Pipelines

Pipeline [tekton.dev/v1]
Task [tekton.dev/v1]
PipelineRun
TaskRun [tekton.dev/v1]
ClusterTask [tekton.dev/v1]
Run [tekton
CustomRun [tekton.dev/v1]
StepAction [tekton.dev/v1]
Verification
ResolutionRequest [resolution.
ApprovalTask [openshiftpipelines.org/v1alpha

Pipeline [tekton.dev/v1]

Description

Pipeline describes a list of Tasks to execute. It expresses how outputs of tasks feed into inputs of subsequent tasks.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the desired state of the Pipeline from the client

.spec

Description

Spec holds the desired state of the Pipeline from the client

Type

`object`

Property	Type	Description
`description`	`string`	Description is a user-facing description of the pipeline that may be used to populate a UI.
`displayName`	`string`	DisplayName is a user-facing name of the pipeline that may be used to populate a UI.

Property	Type	Description
`finally`	`array`	Finally declares the list of Tasks that execute just before leaving the Pipeline i.e. either after all Tasks are finished executing successfully or after a failure which would result in ending the Pipeline
`params`	`array`	Params declares a list of input parameters that must be supplied when this Pipeline is run.
`results`	`array`	Results are values that this pipeline can output once run
`tasks`	`array`	Tasks declares the graph of Tasks that execute when this Pipeline is run.
`workspaces`	`array`	Workspaces declares a set of named workspaces that are expected to be provided by a PipelineRun.

.spec.finally

Description

Finally declares the list of Tasks that execute just before leaving the Pipeline i.e. either after all Tasks are finished executing successfully or after a failure which would result in ending the Pipeline

Type

`array`

.spec.finally[]

Description

PipelineTask defines a task in a Pipeline, passing inputs from both Params and from the output of previous tasks.

Type

`object`

Property	Type	Description
`description`	`string`	Description is the description of this task within the context of a Pipeline. This description may be used to populate a UI.
`displayName`	`string`	DisplayName is the display name of this task within the context of a Pipeline. This display name may be used to populate a UI.
`matrix`	`object`	Matrix declares parameters used to fan out this task.
`name`	`string`	Name is the name of this task within the context of a Pipeline. Name is used as a coordinate with the `from` and `runAfter` fields to establish the execution order of tasks relative to one another.
`onError`	`string`	OnError defines the exiting behavior of a PipelineRun on error can be set to [continue | stopAndFail]
`params`	`array`	Parameters declares parameters passed to this task.

Property	Type	Description
`pipelineRef`	`object`	PipelineRef is a reference to a pipeline definition Note: PipelineRef is in preview mode and not yet supported
`pipelineSpec`		PipelineSpec is a specification of a pipeline Note: PipelineSpec is in preview mode and not yet supported Specifying PipelineSpec can be disabled by setting `disable-inline-spec` feature flag. See Pipeline.spec (API version: tekton.dev/v1)
`retries`	`integer`	Retries represents how many times this task should be retried in case of task failure: ConditionSucceeded set to False
`runAfter`	`array`	RunAfter is the list of PipelineTask names that should be executed before this Task executes. (Used to force a specific ordering in graph execution.)
`taskRef`	`object`	TaskRef is a reference to a task definition.
`taskSpec`		TaskSpec is a specification of a task Specifying TaskSpec can be disabled by setting `disable-inline-spec` feature flag. See Task.spec (API version: tekton.dev/v1)
`timeout`	`string`	Duration after which the TaskRun times out. Defaults to 1 hour. Refer Go's ParseDuration documentation for

Property	Type	Description
		expected format: https://golang.org/pkg/time/#ParseDuration ↗
`when`	`array`	When is a list of when expressions that need to be true for the task to run
`workspaces`	`array`	Workspaces maps workspaces from the pipeline spec to the workspaces declared in the Task.

.spec.finally[].matrix

Description

Matrix declares parameters used to fan out this task.

Type

`object`

Property	Type	Description
`include`	`array`	Include is a list of IncludeParams which allows passing in specific combinations of Parameters into the Matrix.
`params`	`array`	Params is a list of parameters used to fan out the pipelineTask Params takes only `Parameters` of type `"array"` Each array element is supplied to the `PipelineTask` by substituting `params` of type `"string"` in the underlying `Task`. The names of the `params` in the `Matrix` must match the names of the

Property	Type	Description
		`params` in the underlying `Task` that they will be substituting.

.spec.finally[].matrix.include

Description

Include is a list of IncludeParams which allows passing in specific combinations of Parameters into the Matrix.

Type

`array`

.spec.finally[].matrix.include[]

Description

IncludeParams allows passing in a specific combinations of Parameters into the Matrix.

Type

`object`

Property	Type	Description
`name`	`string`	Name the specified combination
`params`	`array`	Params takes only `Parameters` of type `"string"` The names of the `params` must match the names of the `params` in the underlying `Task`

.spec.finally[].matrix.include[].params

Description

Params takes only `Parameters` of type `"string"` The names of the `params` must match the names of the `params` in the underlying `Task`

Type

`array`

.spec.finally[].matrix.include[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.finally[].matrix.params

Description

Params is a list of parameters used to fan out the pipelineTask Params takes only `Parameters` of type `"array"` Each array element is supplied to the `PipelineTask` by substituting `params` of type `"string"` in the underlying `Task`. The names of the `params` in the `Matrix` must match the names of the `params` in the underlying `Task` that they will be substituting.

Type

`array`

.spec.finally[].matrix.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.finally[].params

Description

Parameters declares parameters passed to this task.

Type

`array`

.spec.finally[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.finally[].pipelineRef

Description

PipelineRef is a reference to a pipeline definition Note: PipelineRef is in preview mode and not yet supported

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.finally[].pipelineRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.finally[].pipelineRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.finally[].runAfter

Description

RunAfter is the list of PipelineTask names that should be executed before this Task executes. (Used to force a specific ordering in graph execution.)

Type

`array`

.spec.finally[].runAfter[]

Type

`string`

.spec.finally[].taskRef

Description

TaskRef is a reference to a task definition.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent Note: A Task with non-empty APIVersion and Kind is considered a Custom Task
`kind`	`string`	TaskKind indicates the Kind of the Task: 1. Namespaced Task when Kind is set to "Task". If Kind is "", it defaults to "Task". 2. Custom Task when Kind is non-empty and APIVersion is non-empty
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗

Property	Type	Description
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.finally[].taskRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.finally[].taskRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.finally[].when

Description

When is a list of when expressions that need to be true for the task to run

Type

`array`

.spec.finally[].when[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task

Property	Type	Description
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.spec.finally[].when[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.spec.finally[].when[].values[]

Type

`string`

.spec.finally[].workspaces

Description

Workspaces maps workspaces from the pipeline spec to the workspaces declared in the Task.

Type

`array`

.spec.finally[].workspaces[]

Description

WorkspacePipelineTaskBinding describes how a workspace passed into the pipeline should be mapped to a task's declared workspace.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the workspace as declared by the task
`subPath`	`string`	SubPath is optionally a directory on the volume which should be used for this binding (i.e. the volume will be mounted at this sub directory).
`workspace`	`string`	Workspace is the name of the workspace declared by the pipeline

.spec.params

Description

Params declares a list of input parameters that must be supplied when this Pipeline is run.

Type

`array`

.spec.params[]

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources). Parameter values are provided by users as inputs on a TaskRun or PipelineRun.

Type

`object`

Required

`name`

Property	Type	Description
`default`		Default is the value a parameter takes if no input value is supplied. If default is set, a Task may be executed without a supplied value for the parameter.
`description`	`string`	Description is a user-facing description of the parameter that may be used to populate a UI.
`enum`	`array`	Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.
`name`	`string`	Name declares the name by which a parameter is referenced.
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs parameter.

Property	Type	Description
`type`	`string`	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

.spec.params[].enum

Description

Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.

Type

`array`

.spec.params[].enum[]

Type

`string`

.spec.params[].properties

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

.spec.results

Description

Results are values that this pipeline can output once run

Type

`array`

.spec.results[]

Description

PipelineResult used to describe the results of a pipeline

Type

`object`

Required

`name` `value`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`type`	`string`	Type is the user-specified type of the result. The possible types are 'string', 'array', and 'object', with 'string' as the default. 'array' and 'object' types are alpha features.
`value`		Value the expression used to retrieve the value

.spec.tasks

Description

Tasks declares the graph of Tasks that execute when this Pipeline is run.

Type

`array`

.spec.tasks[]

Description

PipelineTask defines a task in a Pipeline, passing inputs from both Params and from the output of previous tasks.

Type

`object`

Property	Type	Description
`description`	`string`	Description is the description of this task within the context of a Pipeline. This description may be used to populate a UI.
`displayName`	`string`	DisplayName is the display name of this task within the context of a Pipeline. This display name may be used to populate a UI.
`matrix`	`object`	Matrix declares parameters used to fan out this task.
`name`	`string`	Name is the name of this task within the context of a Pipeline. Name is used as a coordinate with the `from` and `runAfter` fields to establish the execution order of tasks relative to one another.

Property	Type	Description
`onError`	`string`	OnError defines the exiting behavior of a PipelineRun on error can be set to [continue | stopAndFail]
`params`	`array`	Parameters declares parameters passed to this task.
`pipelineRef`	`object`	PipelineRef is a reference to a pipeline definition Note: PipelineRef is in preview mode and not yet supported
`pipelineSpec`		PipelineSpec is a specification of a pipeline Note: PipelineSpec is in preview mode and not yet supported Specifying PipelineSpec can be disabled by setting `disable-inline-spec` feature flag. See Pipeline.spec (API version: tekton.dev/v1)
`retries`	`integer`	Retries represents how many times this task should be retried in case of task failure: ConditionSucceeded set to False
`runAfter`	`array`	RunAfter is the list of PipelineTask names that should be executed before this Task executes. (Used to force a specific ordering in graph execution.)
`taskRef`	`object`	TaskRef is a reference to a task definition.

Property	Type	Description
`taskSpec`		TaskSpec is a specification of a task Specifying TaskSpec can be disabled by setting `disable-inline-spec` feature flag. See Task.spec (API version: tekton.dev/v1)
`timeout`	`string`	Duration after which the TaskRun times out. Defaults to 1 hour. Refer Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗
`when`	`array`	When is a list of when expressions that need to be true for the task to run
`workspaces`	`array`	Workspaces maps workspaces from the pipeline spec to the workspaces declared in the Task.

.spec.tasks[].matrix

Description

Matrix declares parameters used to fan out this task.

Type

`object`

Property	Type	Description
`include`	`array`	Include is a list of IncludeParams which allows passing in specific combinations of Parameters into the Matrix.

Property	Type	Description
`params`	`array`	Params is a list of parameters used to fan out the pipelineTask Params takes only `Parameters` of type `"array"` Each array element is supplied to the `PipelineTask` by substituting `params` of type `"string"` in the underlying `Task`. The names of the `params` in the `Matrix` must match the names of the `params` in the underlying `Task` that they will be substituting.

.spec.tasks[].matrix.include

Description

Include is a list of IncludeParams which allows passing in specific combinations of Parameters into the Matrix.

Type

`array`

.spec.tasks[].matrix.include[]

Description

IncludeParams allows passing in a specific combinations of Parameters into the Matrix.

Type

`object`

Property	Type	Description
`name`	`string`	Name the specified combination

Property	Type	Description
`params`	`array`	Params takes only `Parameters` of type `"string"` The names of the `params` must match the names of the `params` in the underlying `Task`

.spec.tasks[].matrix.include[].params

Description

Params takes only `Parameters` of type `"string"` The names of the `params` must match the names of the `params` in the underlying `Task`

Type

`array`

.spec.tasks[].matrix.include[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.tasks[].matrix.params

Description

Params is a list of parameters used to fan out the pipelineTask Params takes only `Parameters` of type `"array"` Each array element is supplied to the `PipelineTask` by substituting `params` of type `"string"` in the underlying `Task`. The names of the `params` in the `Matrix` must match the names of the `params` in the underlying `Task` that they will be substituting.

Type

`array`

.spec.tasks[].matrix.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.tasks[].params

Description

Parameters declares parameters passed to this task.

Type

`array`

.spec.tasks[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.tasks[].pipelineRef

Description

PipelineRef is a reference to a pipeline definition Note: PipelineRef is in preview mode and not yet supported

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include

Property	Type	Description
		"repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.tasks[].pipelineRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.tasks[].pipelineRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.tasks[].runAfter

Description

RunAfter is the list of PipelineTask names that should be executed before this Task executes. (Used to force a specific ordering in graph execution.)

Type

`array`

.spec.tasks[].runAfter[]

Type

`string`

.spec.tasks[].taskRef

Description

TaskRef is a reference to a task definition.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent Note: A Task with non-empty APIVersion and Kind is considered a Custom Task
`kind`	`string`	TaskKind indicates the Kind of the Task: 1. Namespaced Task when Kind is set to "Task". If Kind is "", it defaults to "Task". 2. Custom Task when Kind is non-empty and APIVersion is non-empty

Property	Type	Description
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.tasks[].taskRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.tasks[].taskRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.tasks[].when

Description

When is a list of when expressions that need to be true for the task to run

Type

`array`

.spec.tasks[].when[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗

Property	Type	Description
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.spec.tasks[].when[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.spec.tasks[].when[].values[]

Type

`string`

.spec.tasks[].workspaces

Description

Workspaces maps workspaces from the pipeline spec to the workspaces declared in the Task.

Type

`array`

.spec.tasks[].workspaces[]

Description

WorkspacePipelineTaskBinding describes how a workspace passed into the pipeline should be mapped to a task's declared workspace.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the workspace as declared by the task
`subPath`	`string`	SubPath is optionally a directory on the volume which should be used for this binding (i.e. the volume will be mounted at this sub directory).
`workspace`	`string`	Workspace is the name of the workspace declared by the pipeline

.spec.workspaces

Description

Workspaces declares a set of named workspaces that are expected to be provided by a PipelineRun.

Type

`array`

.spec.workspaces[]

Description

PipelineWorkspaceDeclaration creates a named slot in a Pipeline that a PipelineRun is expected to populate with a workspace binding.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human readable string describing how the workspace will be used in the Pipeline. It can be useful to include a bit of detail about which tasks are intended to have access to the data on the workspace.
`name`	`string`	Name is the name of a workspace to be provided by a PipelineRun.
`optional`	`boolean`	Optional marks a Workspace as not being required in PipelineRuns. By default this field is false and so declared workspaces are required.

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1/namespaces/{namespace}/pipelines`
  - `DELETE`: delete collection of Pipeline
  - `GET`: list objects of kind Pipeline
  - `POST`: create a new Pipeline
- `/apis/tekton.dev/v1/namespaces/{namespace}/pipelines/{name}`
  - `DELETE`: delete the specified Pipeline
  - `GET`: read the specified Pipeline
  - `PATCH`: partially update the specified Pipeline
  - `PUT`: replace the specified Pipeline

/apis/tekton.dev/v1/namespaces/{namespace}/pipelines

HTTP method

`DELETE`

Description

delete collection of Pipeline

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind Pipeline

HTTP responses

HTTP code	Response body
200 - OK	`PipelineList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new Pipeline

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Pipeline` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Pipeline` schema
201 - Created	`Pipeline` schema
202 - Accepted	`Pipeline` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/pipelines/{name}

HTTP method

`DELETE`

Description

delete the specified Pipeline

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified Pipeline

HTTP responses

HTTP code	Response body
200 - OK	`Pipeline` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified Pipeline

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore:

Parameter	Type	Description
		This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`Pipeline` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified Pipeline

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Pipeline` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Pipeline` schema
201 - Created	`Pipeline` schema
401 - Unauthorized	Empty

Task [tekton.dev/v1]

Description

Task represents a collection of sequential steps that are run as part of a Pipeline using a set of inputs and producing a set of outputs. Tasks execute when TaskRuns are created that provide the input parameters and resources and output resources the Task requires.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the desired state of the Task from the client

.spec

Description

Spec holds the desired state of the Task from the client

Type

`object`

Property	Type	Description
`description`	`string`	Description is a user-facing description of the task that may be used to populate a UI.
`displayName`	`string`	DisplayName is a user-facing name of the task that may be used to populate a UI.
`params`	`array`	Params is a list of input parameters required to run the task. Params must be supplied as inputs in TaskRuns unless they declare a default value.

Property	Type	Description
`results`	`array`	Results are values that this Task can output
`sidecars`	`array`	Sidecars are run alongside the Task's step containers. They begin before the steps start and end after the steps complete.
`stepTemplate`	`object`	StepTemplate can be used as the basis for all step containers within the Task, so that the steps inherit settings on the base container.
`steps`	`array`	Steps are the steps of the build; each step is run sequentially with the source mounted into /workspace.
`volumes`		Volumes is a collection of volumes that are available to mount into the steps of the build. See Pod.spec.volumes (API version: v1)
`workspaces`	`array`	Workspaces are the volumes that this Task requires.

.spec.params

Description

Params is a list of input parameters required to run the task. Params must be supplied as inputs in TaskRuns unless they declare a default value.

Type

`array`

.spec.params[]

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources). Parameter values are provided by users as inputs on a TaskRun or PipelineRun.

Type

`object`

Required

`name`

Property	Type	Description
`default`		Default is the value a parameter takes if no input value is supplied. If default is set, a Task may be executed without a supplied value for the parameter.
`description`	`string`	Description is a user-facing description of the parameter that may be used to populate a UI.
`enum`	`array`	Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.
`name`	`string`	Name declares the name by which a parameter is referenced.

Property	Type	Description
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs parameter.
`type`	`string`	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

.spec.params[].enum

Description

Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.

Type

`array`

.spec.params[].enum[]

Type

`string`

.spec.params[].properties

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

.spec.results

Description

Results are values that this Task can output

Type

`array`

.spec.results[]

Description

TaskResult used to describe the results of a task

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs results.
`type`	`string`	Type is the user-specified type of the result. The possible type is currently "string" and will support "array" in following work.

Property	Type	Description
`value`		Value the expression used to retrieve the value of the result from an underlying Step.

.spec.results[].properties

Description

Properties is the JSON Schema properties to support key-value pairs results.

Type

`object`

.spec.sidecars

Description

Sidecars are run alongside the Task's step containers. They begin before the steps start and end after the steps complete.

Type

`array`

.spec.sidecars[]

Description

Sidecar has nearly the same data structure as Step but does not have the ability to timeout.

Type

`object`

Required

`name`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is not provided. Variable references $(VAR_NAME) using the Sidecar's environment. If a variable can resolved, the reference in the input string will be Double $$ are reduced to a single $, which allow the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" string literal "$(VAR_NAME)". Escaped reference expanded, regardless of whether the variable ex Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-appli command-argument-container/#running-a-comm
`command`	`array`	Entrypoint array. Not executed within a shell. The ENTRYPOINT is used if this is not provided. Vari $(VAR_NAME) are expanded using the Sidecar's a variable cannot be resolved, the reference in th will be unchanged. Double $$ are reduced to a s allows for escaping the $(VAR_NAME) syntax: i. "$$(VAR_NAME)" will produce the string literal "$ Escaped references will never be expanded, reg whether the variable exists or not. Cannot be upd https://kubernetes.io/docs/tasks/inject-data-appli command-argument-container/#running-a-comm
`computeResources`	`object`	ComputeResources required by this Sidecar. Ca updated. More info: https://kubernetes.io/docs/concepts/configuration resources-containers/ ↗

Property	Type	Description
`env`	`array`	List of environment variables to set in the Sideca updated.
`envFrom`	`array`	List of sources to populate environment variables The keys defined within a source must be a C_ID invalid keys will be reported as an event when th starting. When a key exists in multiple sources, t associated with the last source will take precede defined by an Env with a duplicate key will take p Cannot be updated.
`image`	`string`	Image reference name. More info: https://kubernetes.io/docs/concepts/containers/ir
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPre to Always if :latest tag is specified, or IfNotPresei Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/ir images ↗
`lifecycle`	`object`	Actions that the management system should tak Sidecar lifecycle events. Cannot be updated.

Property	Type	Description
`livenessProbe`	`object`	Periodic probe of Sidecar liveness. Container wil the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗
`name`	`string`	Name of the Sidecar specified as a DNS_LABEL in a Task must have a unique name (DNS_LABE updated.
`ports`	`array`	List of ports to expose from the Sidecar. Exposin gives the system additional information about the connections a container uses, but is primarily inf specifying a port here DOES NOT prevent that p exposed. Any port which is listening on the defau address inside a container will be accessible fror Cannot be updated.
`readinessProbe`	`object`	Periodic probe of Sidecar service readiness. Cor removed from service endpoints if the probe fails updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗
`restartPolicy`	`string`	RestartPolicy refers to kubernetes RestartPolicy. set for an initContainer and must have it's policy It is currently left optional to help support Kubern prior to 1.29 when this feature was introduced.

Property	Type	Description
`script`	`string`	Script is the contents of an executable file to exe If Script is not empty, the Step cannot have an C Args.
`securityContext`	`object`	SecurityContext defines the security options the be run with. If set, the fields of SecurityContext o equivalent fields of PodSecurityContext. More inf https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`startupProbe`	`object`	StartupProbe indicates that the Pod the Sidecar successfully initialized. If specified, no other prob until this completes successfully. If this probe fail be restarted, just as if the livenessProbe failed. T to provide different probe parameters at the begi lifecycle, when it might take a long time to load d cache, than during steady-state operation. This c updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗
`stdin`	`boolean`	Whether this Sidecar should allocate a buffer for container runtime. If this is not set, reads from st Sidecar will always result in EOF. Default is false
`stdinOnce`	`boolean`	Whether the container runtime should close the s after it has been opened by a single attach. Whe the stdin stream will remain open across multiple

Property	Type	Description
		sessions. If stdinOnce is set to true, stdin is oper start, is empty until the first client attaches to stdi remains open and accepts data until the client di which time stdin is closed and remains closed ur restarted. If this flag is false, a container process from stdin will never receive an EOF. Default is fa
`terminationMessagePath`	`string`	Optional: Path at which the file to which the Side message will be written is mounted into the Side Message written is intended to be brief final statu assertion failure message. Will be truncated by tl greater than 4096 bytes. The total message leng containers will be limited to 12kb. Defaults to /de log. Cannot be updated.
`terminationMessagePolicy`	`string`	Indicate how the termination message should be will use the contents of terminationMessagePath Sidecar status message on both success and fai FallbackToLogsOnError will use the last chunk o output if the termination message file is empty ar exited with an error. The log output is limited to 2 lines, whichever is smaller. Defaults to File. Cann
`tty`	`boolean`	Whether this Sidecar should allocate a TTY for it requires 'stdin' to be true. Default is false.
`volumeDevices`	`array`	volumeDevices is the list of block devices to be u Sidecar.

Property	Type	Description
`volumeMounts`	`array`	Volumes to mount into the Sidecar's filesystem. C updated.
`workingDir`	`string`	Sidecar's working directory. If not specified, the c runtime's default will be used, which might be co container image. Cannot be updated.
`workspaces`	`array`	This is an alpha field. You must set the "enable-a flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Tasl Sidecar wants exclusive access to. Adding a wor list means that any other Step or Sidecar that do request this Workspace will not have access to it

.spec.sidecars[].args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Sidecar's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.sidecars[].args[]

Type

`string`

.spec.sidecars[].command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Sidecar's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.sidecars[].command[]

Type

`string`

.spec.sidecars[].computeResources

Description

ComputeResources required by this Sidecar. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.sidecars[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.sidecars[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.sidecars[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.sidecars[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.sidecars[].env

Description

List of environment variables to set in the Sidecar. Cannot be updated.

Type

`array`

.spec.sidecars[].env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a

Property	Type	Description
		variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.sidecars[].env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Property	Type	Description
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.sidecars[].env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty.

Property	Type	Description
		Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.sidecars[].env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.sidecars[].env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.sidecars[].env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.sidecars[].env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.sidecars[].envFrom

Description

List of sources to populate environment variables in the Sidecar. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the container is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.spec.sidecars[].envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.
`secretRef`	`object`	The Secret to select from

.spec.sidecars[].envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

Property	Type	Description
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.spec.sidecars[].envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.spec.sidecars[].lifecycle

Description

Actions that the management system should take in response to Sidecar lifecycle events. Cannot be updated.

Type

`object`

Property	Type	Description
`postStart`	`object`	PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks ↗
`preStop`	`object`	PreStop is called immediately before a container is terminated due to an API request or management event such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks ↗
`stopSignal`	`string`	StopSignal defines which signal will be sent to a container when it is being stopped. If not specified, the default is defined by the container runtime in use. StopSignal can only be set for Pods with a non-empty .spec.os.name

.spec.sidecars[].lifecycle.postStart

Description

PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in the container.
`httpGet`	`object`	HTTPGet specifies an HTTP GET request to perform.
`sleep`	`object`	Sleep represents a duration that the container should sleep.
`tcpSocket`	`object`	Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

.spec.sidecars[].lifecycle.postStart.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.spec.sidecars[].lifecycle.postStart.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.spec.sidecars[].lifecycle.postStart.exec.command[]

Type

`string`

.spec.sidecars[].lifecycle.postStart.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.spec.sidecars[].lifecycle.postStart.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.spec.sidecars[].lifecycle.postStart.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.spec.sidecars[].lifecycle.postStart.sleep

Description

Sleep represents a duration that the container should sleep.

Type

`object`

Required

`seconds`

Property	Type	Description
`seconds`	`integer`	Seconds is the number of seconds to sleep.

.spec.sidecars[].lifecycle.postStart.tcpSocket

Description

Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.spec.sidecars[].lifecycle.preStop

Description

PreStop is called immediately before a container is terminated due to an API request or management event such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in the container.
`httpGet`	`object`	HTTPGet specifies an HTTP GET request to perform.
`sleep`	`object`	Sleep represents a duration that the container should sleep.
`tcpSocket`	`object`	Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

.spec.sidecars[].lifecycle.preStop.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell

Property	Type	Description
		instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.spec.sidecars[].lifecycle.preStop.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.spec.sidecars[].lifecycle.preStop.exec.command[]

Type

`string`

.spec.sidecars[].lifecycle.preStop.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.spec.sidecars[].lifecycle.preStop.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.spec.sidecars[].lifecycle.preStop.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.spec.sidecars[].lifecycle.preStop.sleep

Description

Sleep represents a duration that the container should sleep.

Type

`object`

Required

`seconds`

Property	Type	Description
`seconds`	`integer`	Seconds is the number of seconds to sleep.

.spec.sidecars[].lifecycle.preStop.tcpSocket

Description

Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.spec.sidecars[].livenessProbe

Description

Periodic probe of Sidecar liveness. Container will be restarted if the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th

Property	Type	Description
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq
`httpGet`	`object`	HTTPGet specifies an HTTP GET request
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru

Property	Type	Description
		pod are sent a termination signal and the ti processes are forcibly halted with a kill sigı value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t (no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceF gate. Minimum value is 1. spec.terminationGracePeriodSeconds is u
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.spec.sidecars[].livenessProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell

Property	Type	Description
		instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.spec.sidecars[].livenessProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.spec.sidecars[].livenessProbe.exec.command[]

Type

`string`

.spec.sidecars[].livenessProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md ↗). If this is not specified, the default behavior is defined by gRPC.

.spec.sidecars[].livenessProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.

Property	Type	Description
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.spec.sidecars[].livenessProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.spec.sidecars[].livenessProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.spec.sidecars[].livenessProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.spec.sidecars[].ports

Description

List of ports to expose from the Sidecar. Exposing a port here gives the system additional information about the network connections a container uses, but is primarily informational. Not specifying a port here DOES NOT prevent that port from being exposed. Any port which is listening on the default "0.0.0.0" address inside a container will be accessible from the network. Cannot be updated.

Type

`array`

.spec.sidecars[].ports[]

Description

ContainerPort represents a network port in a single container.

Type

`object`

Required

`containerPort`

Property	Type	Description
`containerPort`	`integer`	Number of port to expose on the pod's IP address. This must be a valid port number, 0 < x < 65536.
`hostIP`	`string`	What host IP to bind the external port to.
`hostPort`	`integer`	Number of port to expose on the host. If specified, this must be a valid port number, 0 < x < 65536. If HostNetwork is specified, this must match ContainerPort. Most containers do not need this.

Property	Type	Description
`name`	`string`	If specified, this must be an IANA_SVC_NAME and unique within the pod. Each named port in a pod must have a unique name. Name for the port that can be referred to by services.
`protocol`	`string`	Protocol for port. Must be UDP, TCP, or SCTP. Defaults to "TCP".

.spec.sidecars[].readinessProbe

Description

Periodic probe of Sidecar service readiness. Container will be removed from service endpoints if the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq

Property	Type	Description
`httpGet`	`object`	HTTPGet specifies an HTTP GET request
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru pod are sent a termination signal and the t processes are forcibly halted with a kill sig value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t

Property	Type	Description
		(no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceP gate. Minimum value is 1. spec.terminationGracePeriodSeconds is us
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.spec.sidecars[].readinessProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.spec.sidecars[].readinessProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.spec.sidecars[].readinessProbe.exec.command[]

Type

`string`

.spec.sidecars[].readinessProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md ↗).

Property	Type	Description
		If this is not specified, the default behavior is defined by gRPC.

.spec.sidecars[].readinessProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

Property	Type	Description
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.spec.sidecars[].readinessProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.spec.sidecars[].readinessProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.spec.sidecars[].readinessProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.spec.sidecars[].securityContext

Description

SecurityContext defines the security options the Sidecar should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its

Property	Type	Description
		parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value

Property	Type	Description
		is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.spec.sidecars[].securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.spec.sidecars[].securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.spec.sidecars[].securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.spec.sidecars[].securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.sidecars[].securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.spec.sidecars[].securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.sidecars[].securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.spec.sidecars[].securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.spec.sidecars[].securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-

Property	Type	Description
		gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.spec.sidecars[].startupProbe

Description

StartupProbe indicates that the Pod the Sidecar is running in has successfully initialized. If specified, no other probes are executed until this completes successfully. If this probe fails, the Pod will be restarted, just as if the livenessProbe failed. This can be used to provide different probe parameters at the beginning of a Pod's lifecycle, when it might take a long time to load data or warm a cache, than during steady-state operation. This cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq
`httpGet`	`object`	HTTPGet specifies an HTTP GET request
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

Property	Type	Description
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru pod are sent a termination signal and the ti processes are forcibly halted with a kill sig value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t (no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceF gate. Minimum value is 1. spec.terminationGracePeriodSeconds is u

Property	Type	Description
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.spec.sidecars[].startupProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.spec.sidecars[].startupProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you

need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.spec.sidecars[].startupProbe.exec.command[]

Type

`string`

.spec.sidecars[].startupProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md ↗). If this is not specified, the default behavior is defined by gRPC.

.spec.sidecars[].startupProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.spec.sidecars[].startupProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.spec.sidecars[].startupProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.spec.sidecars[].startupProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.spec.sidecars[].volumeDevices

Description

volumeDevices is the list of block devices to be used by the Sidecar.

Type

`array`

.spec.sidecars[].volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.spec.sidecars[].volumeMounts

Description

Volumes to mount into the Sidecar's filesystem. Cannot be updated.

Type

`array`

.spec.sidecars[].volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.

Property	Type	Description
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively. If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be

Property	Type	Description
		unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.spec.sidecars[].workspaces

Description

This is an alpha field. You must set the "enable-api-fields" feature flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this Sidecar wants exclusive access to. Adding a workspace to this list means that any other Step or Sidecar that does not also request this Workspace will not have access to it.

Type

`array`

.spec.sidecars[].workspaces[]

Description

WorkspaceUsage is used by a Step or Sidecar to declare that it wants isolated access to a Workspace defined in a Task.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	MountPath is the path that the workspace should be mounted to inside the Step or Sidecar, overriding any MountPath specified in the Task's WorkspaceDeclaration.
`name`	`string`	Name is the name of the workspace this Step or Sidecar wants access to.

.spec.stepTemplate

Description

StepTemplate can be used as the basis for all step containers within the Task, so that the steps inherit settings on the base container.

Type

`object`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is used if t not provided. Variable references $(VAR_NAME) are expa using the Step's environment. If a variable cannot be resol the reference in the input string will be unchanged. Double are reduced to a single $, which allows for escaping the

Property	Type	Description
		$(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce string literal "$(VAR_NAME)". Escaped references will nev expanded, regardless of whether the variable exists or not Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`command`	`array`	Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable refer $(VAR_NAME) are expanded using the Step's environmer variable cannot be resolved, the reference in the input stri be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NA Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. Mo https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`computeResources`	`object`	ComputeResources required by this Step. Cannot be upda More info: https://kubernetes.io/docs/concepts/configuration/manage resources-containers/ ↗
`env`	`array`	List of environment variables to set in the Step. Cannot be updated.
`envFrom`	`array`	List of sources to populate environment variables in the St The keys defined within a source must be a C_IDENTIFIE invalid keys will be reported as an event when the Step is

Property	Type	Description
		starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Valu defined by an Env with a duplicate key will take precedenc Cannot be updated.
`image`	`string`	Image reference name. More info: https://kubernetes.io/docs/concepts/containers/images ↗
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPresent. Def to Always if :latest tag is specified, or IfNotPresent otherwi Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/images#up images ↗
`securityContext`	`object`	SecurityContext defines the security options the Step shou run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`volumeDevices`	`array`	volumeDevices is the list of block devices to be used by th Step.
`volumeMounts`	`array`	Volumes to mount into the Step's filesystem. Cannot be up
`workingDir`	`string`	Step's working directory. If not specified, the container run default will be used, which might be configured in the cont

Property	Type	Description
		image. Cannot be updated.

.spec.stepTemplate.args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Step's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.stepTemplate.args[]

Type

`string`

.spec.stepTemplate.command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Step's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.stepTemplate.command[]

Type

`string`

.spec.stepTemplate.computeResources

Description

ComputeResources required by this Step. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

Property	Type	Description
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.stepTemplate.computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.stepTemplate.computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.stepTemplate.computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.stepTemplate.computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.stepTemplate.env

Description

List of environment variables to set in the Step. Cannot be updated.

Type

`array`

.spec.stepTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".

Property	Type	Description
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.stepTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu,

Property	Type	Description
		requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.stepTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.stepTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.stepTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.stepTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.stepTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.stepTemplate.envFrom

Description

List of sources to populate environment variables in the Step. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.spec.stepTemplate.envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.
`secretRef`	`object`	The Secret to select from

.spec.stepTemplate.envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.spec.stepTemplate.envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.spec.stepTemplate.securityContext

Description

SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its

Property	Type	Description
		parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value

Property	Type	Description
		is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.spec.stepTemplate.securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.spec.stepTemplate.securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.spec.stepTemplate.securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.spec.stepTemplate.securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.stepTemplate.securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.spec.stepTemplate.securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.stepTemplate.securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.spec.stepTemplate.securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.spec.stepTemplate.securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-

Property	Type	Description
		gmsa ↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.spec.stepTemplate.volumeDevices

Description

volumeDevices is the list of block devices to be used by the Step.

Type

`array`

.spec.stepTemplate.volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.spec.stepTemplate.volumeMounts

Description

Volumes to mount into the Step's filesystem. Cannot be updated.

Type

`array`

.spec.stepTemplate.volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively. If ReadOnly is false, this field has no meaning and must be unspecified.

Property	Type	Description
		If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.spec.steps

Description

Steps are the steps of the build; each step is run sequentially with the source mounted into /workspace.

Type

`array`

.spec.steps[]

Description

Step runs a subcomponent of a Task

Type

`object`

Required

`name`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is used if t not provided. Variable references $(VAR_NAME) are expa using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchange Double $$ are reduced to a single $, which allows for esca the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will prod string literal "$(VAR_NAME)". Escaped references will nev expanded, regardless of whether the variable exists or not Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`command`	`array`	Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable refer $(VAR_NAME) are expanded using the container's enviro If a variable cannot be resolved, the reference in the input

Property	Type	Description
		will be unchanged. Double $$ are reduced to a single $, w allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NA Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. Mo https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`computeResources`	`object`	ComputeResources required by this Step. Cannot be upda More info: https://kubernetes.io/docs/concepts/configuration/manage resources-containers/ ↗
`displayName`	`string`	DisplayName is a user-facing name of the step that may b to populate a UI.
`env`	`array`	List of environment variables to set in the Step. Cannot be updated.
`envFrom`	`array`	List of sources to populate environment variables in the St The keys defined within a source must be a C_IDENTIFIE invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Valu defined by an Env with a duplicate key will take precedenc Cannot be updated.

Property	Type	Description
`image`	`string`	Docker image name. More info: https://kubernetes.io/docs/concepts/containers/images ↗
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPresent. Def to Always if :latest tag is specified, or IfNotPresent otherwi Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/images#up images ↗
`name`	`string`	Name of the Step specified as a DNS_LABEL. Each Step Task must have a unique name.
`onError`	`string`	OnError defines the exiting behavior of a container on erro be set to [continue | stopAndFail]
`params`	`array`	Params declares parameters passed to this step action.
`ref`	`object`	Contains the reference to an existing StepAction.
`results`	`array`	Results declares StepResults produced by the Step. It can be used in an inlined Step when used to store Resu $(step.results.resultName.path). It cannot be used when referencing StepActions using [v1.Step.Ref]. The Results declared by the StepActions will be stored here instead.

Property	Type	Description
`script`	`string`	Script is the contents of an executable file to execute. If Script is not empty, the Step cannot have an Command and the Args will be passed to the Script.
`securityContext`	`object`	SecurityContext defines the security options the Step should run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`stderrConfig`	`object`	Stores configuration for the stderr stream of the step.
`stdoutConfig`	`object`	Stores configuration for the stdout stream of the step.
`timeout`	`string`	Timeout is the time after which the step times out. Defaults to never. Refer to Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗
`volumeDevices`	`array`	volumeDevices is the list of block devices to be used by the Step.
`volumeMounts`	`array`	Volumes to mount into the Step's filesystem. Cannot be up

Property	Type	Description
`when`	`array`	When is a list of when expressions that need to be true for task to run
`workingDir`	`string`	Step's working directory. If not specified, the container runt default will be used, which might be configured in the cont image. Cannot be updated.
`workspaces`	`array`	This is an alpha field. You must set the "enable-api-fields" flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this wants exclusive access to. Adding a workspace to this list that any other Step or Sidecar that does not also request t Workspace will not have access to it.

.spec.steps[].args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.steps[].args[]

Type

`string`

.spec.steps[].command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.steps[].command[]

Type

`string`

.spec.steps[].computeResources

Description

ComputeResources required by this Step. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.steps[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.steps[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.steps[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.steps[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.steps[].env

Description

List of environment variables to set in the Step. Cannot be updated.

Type

`array`

.spec.steps[].env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a

Property	Type	Description
		variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.steps[].env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Property	Type	Description
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.steps[].env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty.

Property	Type	Description
		Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.steps[].env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.steps[].env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.steps[].env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.steps[].env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.steps[].envFrom

Description

List of sources to populate environment variables in the Step. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.spec.steps[].envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.
`secretRef`	`object`	The Secret to select from

.spec.steps[].envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

Property	Type	Description
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.spec.steps[].envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.spec.steps[].params

Description

Params declares parameters passed to this step action.

Type

`array`

.spec.steps[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.steps[].ref

Description

Contains the reference to an existing StepAction.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referenced step
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Property	Type	Description
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.steps[].ref.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.steps[].ref.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.steps[].results

Description

Results declares StepResults produced by the Step. It can be used in an inlined Step when used to store Results to $(step.results.resultName.path). It cannot be used when referencing StepActions using [v1.Step.Ref]. The Results declared by the StepActions will be stored here instead.

Type

`array`

.spec.steps[].results[]

Description

StepResult used to describe the Results of a Step.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs results.
`type`	`string`	The possible types are 'string', 'array', and 'object', with 'string' as the default.

.spec.steps[].results[].properties

Description

Properties is the JSON Schema properties to support key-value pairs results.

Type

`object`

.spec.steps[].securityContext

Description

SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note

Property	Type	Description
		that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.spec.steps[].securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.spec.steps[].securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.spec.steps[].securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.spec.steps[].securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.steps[].securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.spec.steps[].securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.steps[].securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.spec.steps[].securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.spec.steps[].securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa ↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

Property	Type	Description
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.spec.steps[].stderrConfig

Description

Stores configuration for the stderr stream of the step.

Type

`object`

Property	Type	Description
`path`	`string`	Path to duplicate stdout stream to on container's local filesystem.

.spec.steps[].stdoutConfig

Description

Stores configuration for the stdout stream of the step.

Type

`object`

Property	Type	Description
`path`	`string`	Path to duplicate stdout stream to on container's local filesystem.

.spec.steps[].volumeDevices

Description

volumeDevices is the list of block devices to be used by the Step.

Type

`array`

.spec.steps[].volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.spec.steps[].volumeMounts

Description

Volumes to mount into the Step's filesystem. Cannot be updated.

Type

`array`

.spec.steps[].volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).

Property	Type	Description
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively. If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.

Property	Type	Description
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.spec.steps[].when

Description

When is a list of when expressions that need to be true for the task to run

Type

`array`

.spec.steps[].when[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.spec.steps[].when[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.spec.steps[].when[].values[]

Type

`string`

.spec.steps[].workspaces

Description

This is an alpha field. You must set the "enable-api-fields" feature flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this Step wants exclusive access to. Adding a workspace to this list means that any other Step or Sidecar that does not also request this Workspace will not have access to it.

Type

`array`

.spec.steps[].workspaces[]

Description

WorkspaceUsage is used by a Step or Sidecar to declare that it wants isolated access to a Workspace defined in a Task.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	MountPath is the path that the workspace should be mounted to inside the Step or Sidecar, overriding any MountPath specified in the Task's WorkspaceDeclaration.
`name`	`string`	Name is the name of the workspace this Step or Sidecar wants access to.

.spec.workspaces

Description

Workspaces are the volumes that this Task requires.

Type

`array`

.spec.workspaces[]

Description

WorkspaceDeclaration is a declaration of a volume that a Task requires.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is an optional human readable description of this volume.
`mountPath`	`string`	MountPath overrides the directory that the volume will be made available at.
`name`	`string`	Name is the name by which you can bind the volume at runtime.
`optional`	`boolean`	Optional marks a Workspace as not being required in TaskRuns. By default this field is false and so declared workspaces are required.

Property	Type	Description
`readOnly`	`boolean`	ReadOnly dictates whether a mounted volume is writable. By default this field is false and so mounted volumes are writable.

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1/namespaces/{namespace}/tasks`
  - `DELETE`: delete collection of Task
  - `GET`: list objects of kind Task
  - `POST`: create a new Task
- `/apis/tekton.dev/v1/namespaces/{namespace}/tasks/{name}`
  - `DELETE`: delete the specified Task
  - `GET`: read the specified Task
  - `PATCH`: partially update the specified Task
  - `PUT`: replace the specified Task

/apis/tekton.dev/v1/namespaces/{namespace}/tasks

HTTP method

`DELETE`

Description

delete collection of Task

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind Task

HTTP responses

HTTP code	Response body
200 - OK	`TaskList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new Task

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last

Parameter	Type	Description
		duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Task` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Task` schema
201 - Created	`Task` schema
202 - Accepted	`Task` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/tasks/{name}

HTTP method

`DELETE`

Description

delete the specified Task

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified Task

HTTP responses

HTTP code	Response body
200 - OK	`Task` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified Task

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`Task` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified Task

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Task` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Task` schema
201 - Created	`Task` schema
401 - Unauthorized	Empty

PipelineRun [tekton.dev/v1]

Description

PipelineRun represents a single execution of a Pipeline. PipelineRuns are how the graph of Tasks declared in a Pipeline are executed; they specify inputs to Pipelines such as parameter values and capture operational aspects of the Tasks execution such as service account and tolerations. Creating a PipelineRun creates TaskRuns for Tasks in the referenced Pipeline.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint

Property	Type	Description
		the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	PipelineRunSpec defines the desired state of PipelineRun
`status`	`object`	PipelineRunStatus defines the observed state of PipelineRun

.spec

Description

PipelineRunSpec defines the desired state of PipelineRun

Type

`object`

Property	Type	Description
`managedBy`	`string`	ManagedBy indicates which controller is responsible for reconciling this resource. If unset or set to "tekton.dev/pipeline", the default Tekton controller will manage this resource. This field is immutable.

Property	Type	Description
`params`	`array`	Params is a list of parameter names and values.
`pipelineRef`	`object`	PipelineRef can be used to refer to a specific instance of a Pipeline.
`pipelineSpec`		Specifying PipelineSpec can be disabled by setting `disable-inline-spec` feature flag. See Pipeline.spec (API version: tekton.dev/v1)
`status`	`string`	Used for cancelling a pipelinerun (and maybe more later on)
`taskRunSpecs`	`array`	TaskRunSpecs holds a set of runtime specs
`taskRunTemplate`	`object`	TaskRunTemplate represent template of taskrun
`timeouts`	`object`	Time after which the Pipeline times out. Currently three keys are accepted in the map pipeline, tasks and finally with Timeouts.pipeline >= Timeouts.tasks + Timeouts.finally
`workspaces`	`array`	Workspaces holds a set of workspace bindings that must match names with those declared in the pipeline.

.spec.params

Description

Params is a list of parameter names and values.

Type

`array`

.spec.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.pipelineRef

Description

PipelineRef can be used to refer to a specific instance of a Pipeline.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.pipelineRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.pipelineRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.taskRunSpecs

Description

TaskRunSpecs holds a set of runtime specs

Type

`array`

.spec.taskRunSpecs[]

Description

PipelineTaskRunSpec can be used to configure specific specs for a concrete Task

Type

`object`

Property	Type	Description
`computeResources`	`object`	Compute resources to use for this TaskRun
`metadata`	`ObjectMeta↗`	PipelineTaskMetadata contains the labels or annotations for an EmbeddedTask

Property	Type	Description
`pipelineTaskName`	`string`	
`podTemplate`	`object`	PodTemplate holds pod specific configuration
`serviceAccountName`	`string`	
`sidecarSpecs`	`array`	
`stepSpecs`	`array`	
`timeout`	`string`	Duration after which the TaskRun times out. Overrides the timeout specified on the Task's spec if specified. Takes lower precedence to PipelineRun's `spec.timeouts.tasks` Refer Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗

.spec.taskRunSpecs[].computeResources

Description

Compute resources to use for this TaskRun

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate.

Property	Type	Description
		This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.taskRunSpecs[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.taskRunSpecs[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.taskRunSpecs[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunSpecs[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunSpecs[].podTemplate

Description

PodTemplate holds pod specific configuration

Type

`object`

Property	Type	Description
`affinity`		If specified, the pod's scheduling constraints Pod.spec.affinity (API version: v1)
`automountServiceAccountToken`	`boolean`	AutomountServiceAccountToken indicates v running as this service account should have token automatically mounted.
`dnsConfig`	`object`	Specifies the DNS parameters of a pod. Par specified here will be merged to the generat configuration based on DNSPolicy.
`dnsPolicy`	`string`	Set DNS policy for the pod. Defaults to "Clu Valid values are 'ClusterFirst', 'Default' or 'N parameters given in DNSConfig will be merg policy selected with DNSPolicy.
`enableServiceLinks`	`boolean`	EnableServiceLinks indicates whether inforr services should be injected into pod's enviro variables, matching the syntax of Docker lin Defaults to true.

Property	Type	Description
`env`	`array`	List of environment variables that can be pr containers belonging to the pod.
`hostAliases`	`array`	HostAliases is an optional list of hosts and I be injected into the pod's hosts file if specifi only valid for non-hostNetwork pods.
`hostNetwork`	`boolean`	HostNetwork specifies whether the pod may node network namespace
`hostUsers`	`boolean`	HostUsers indicates whether the pod will us user namespace. Optional: Default to true. I or not present, the pod will be run in the hos namespace, useful for when the pod needs only available to the host user namespace, loading a kernel module with CAP_SYS_MO When set to false, a new user namespace is the pod. Setting false is useful to mitigating breakout vulnerabilities such as allowing co run as root without their user having root pri the host. This field depends on the kubernet gate UserNamespacesSupport being enable
`imagePullSecrets`	`array`	ImagePullSecrets gives the name of the sec the pod to pull the image if specified

Property	Type	Description
`nodeSelector`	`object`	NodeSelector is a selector which must be tr pod to fit on a node. Selector which must ma labels for the pod to be scheduled on that n info: https://kubernetes.io/docs/concepts/configu pod-node/ ↗
`priorityClassName`	`string`	If specified, indicates the pod's priority. "syst critical" and "system-cluster-critical" are two keywords which indicate the highest prioritie former being the highest priority. Any other r be defined by creating a PriorityClass objec name. If not specified, the pod priority will b zero if there is no default.
`runtimeClassName`	`string`	RuntimeClassName refers to a RuntimeClas the node.k8s.io group, which should be use pod. If no RuntimeClass resource matches t class, the pod will not be run. If unset or em "legacy" RuntimeClass will be used, which i class with an empty definition that uses the runtime handler. More info: https://git.k8s.io/enhancements/keps/sig-no class.md ↗ This is a beta feature as of Kube v1.14.
`schedulerName`	`string`	SchedulerName specifies the scheduler to b dispatch the Pod

Property	Type	Description
`securityContext`		SecurityContext holds pod-level security attı common container settings. Optional: Defau See type description for default values of ea Pod.spec.securityContext (API version: v1)
`tolerations`	`array`	If specified, the pod's tolerations.
`topologySpreadConstraints`	`array`	TopologySpreadConstraints controls how Pc spread across your cluster among failure-dc as regions, zones, nodes, and other user-de topology domains.
`volumes`		List of volumes that can be mounted by con belonging to the pod. More info: https://kubernetes.io/docs/concepts/storage. See Pod.spec.volumes (API version: v1)

.spec.taskRunSpecs[].podTemplate.dnsConfig

Description

Specifies the DNS parameters of a pod. Parameters specified here will be merged to the generated DNS configuration based on DNSPolicy.

Type

`object`

Property	Type	Description
`nameservers`	`array`	A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.
`options`	`array`	A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.
`searches`	`array`	A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

.spec.taskRunSpecs[].podTemplate.dnsConfig.nameservers

Description

A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Type

`array`

.spec.taskRunSpecs[].podTemplate.dnsConfig.nameservers[]

Type

`string`

.spec.taskRunSpecs[].podTemplate.dnsConfig.options

Description

A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.

Type

`array`

.spec.taskRunSpecs[].podTemplate.dnsConfig.options[]

Description

PodDNSConfigOption defines DNS resolver options of a pod.

Type

`object`

Property	Type	Description
`name`	`string`	Name is this DNS resolver option's name. Required.
`value`	`string`	Value is this DNS resolver option's value.

.spec.taskRunSpecs[].podTemplate.dnsConfig.searches

Description

A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

Type

`array`

.spec.taskRunSpecs[].podTemplate.dnsConfig.searches[]

Type

`string`

.spec.taskRunSpecs[].podTemplate.env

Description

List of environment variables that can be provided to the containers belonging to the pod.

Type

`array`

.spec.taskRunSpecs[].podTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a

Property	Type	Description
		variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.taskRunSpecs[].podTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Property	Type	Description
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.taskRunSpecs[].podTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.taskRunSpecs[].podTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".

Property	Type	Description
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.taskRunSpecs[].podTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.

Property	Type	Description
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.taskRunSpecs[].podTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"

Property	Type	Description
`resource`	`string`	Required: resource to select

.spec.taskRunSpecs[].podTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.taskRunSpecs[].podTemplate.hostAliases

Description

HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified. This is only valid for non-hostNetwork pods.

Type

`array`

.spec.taskRunSpecs[].podTemplate.hostAliases[]

Description

HostAlias holds the mapping between IP and hostnames that will be injected as an entry in the pod's hosts file.

Type

`object`

Required

`ip`

Property	Type	Description
`hostnames`	`array`	Hostnames for the above IP address.
`ip`	`string`	IP address of the host file entry.

.spec.taskRunSpecs[].podTemplate.hostAliases[].hostnames

Description

Hostnames for the above IP address.

Type

`array`

.spec.taskRunSpecs[].podTemplate.hostAliases[].hostnames[]

Type

`string`

.spec.taskRunSpecs[].podTemplate.imagePullSecrets

Description

ImagePullSecrets gives the name of the secret used by the pod to pull the image if specified

Type

`array`

.spec.taskRunSpecs[].podTemplate.imagePullSecrets[]

Description

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.taskRunSpecs[].podTemplate.nodeSelector

Description

NodeSelector is a selector which must be true for the pod to fit on a node. Selector which must match a node's labels for the pod to be scheduled on that node. More info: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

Type

`object`

.spec.taskRunSpecs[].podTemplate.tolerations

Description

If specified, the pod's tolerations.

Type

`array`

.spec.taskRunSpecs[].podTemplate.tolerations[]

Description

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator <operator>.

Type

`object`

Property	Type	Description
`effect`	`string`	Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.
`key`	`string`	Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.
`operator`	`string`	Operator represents a key's relationship to the value. Valid operators are Exists, Equal, Lt, and Gt. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category. Lt and Gt perform numeric comparisons (requires feature gate TaintTolerationComparisonOperators).
`tolerationSeconds`	`integer`	TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.
`value`	`string`	Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty,

Property	Type	Description
		otherwise just a regular string.

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints

Description

TopologySpreadConstraints controls how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains.

Type

`array`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[]

Description

TopologySpreadConstraint specifies how to spread matching pods among the given topology.

Type

`object`

Required

`maxSkew` `topologyKey` `whenUnsatisfiable`

Property	Type	Description
`labelSelector`	`object`	LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

Property	Type	Description
`matchLabelKeys`	`array`	MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).
`maxSkew`	`integer`	MaxSkew describes the degree to which pods may be unevenly distributed. When `whenUnsatisfiable=DoNotSchedule`, it is the maximum permitted difference between the number of matching pods in the target topology and the global minimum. The global minimum is the minimum number of matching pods in an eligible domain or zero if the number of eligible domains is less than MinDomains. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 2/2/1: In this case, the global minimum is 1. | zone1 | zone2 | zone3 | | P P | P P | P | • if MaxSkew is 1, incoming pod can only be scheduled to zone3 to become 2/2/2;

Property	Type	Description
		scheduling it onto zone1(zone2) would make the ActualSkew(3-1) on zone1(zone2) violate MaxSkew(1). • if MaxSkew is 2, incoming pod can be scheduled onto any zone. When `whenUnsatisfiable=ScheduleAnyway`, it is used to give higher precedence to topologies that satisfy it. It's a required field. Default value is 1 and 0 is not allowed.
`minDomains`	`integer`	MinDomains indicates a minimum number of eligible domains. When the number of eligible domains with matching topology keys is less than minDomains, Pod Topology Spread treats "global minimum" as 0, and then the calculation of Skew is performed. And when the number of eligible domains with matching topology keys equals or greater than minDomains, this value has no effect on scheduling. As a result, when the number of eligible domains is less than minDomains, scheduler won't schedule more than maxSkew Pods to those domains. If value is nil, the constraint behaves as if MinDomains is equal to 1. Valid values are integers greater than 0. When value is not nil, WhenUnsatisfiable must be DoNotSchedule. For example, in a 3-zone cluster, MaxSkew is set to 2, MinDomains is set to 5 and pods with the same labelSelector spread as 2/2/2: | zone1 | zone2 | zone3 | | P P | P P | P P | The number of domains is less than 5(MinDomains), so "global minimum" is treated as 0. In this situation, new pod with the same labelSelector cannot be scheduled,

Property	Type	Description
		because computed skew will be 3(3 - 0) if new Pod is scheduled to any of the three zones, it will violate MaxSkew.
`nodeAffinityPolicy`	`string`	NodeAffinityPolicy indicates how we will treat Pod's nodeAffinity/nodeSelector when calculating pod topology spread skew. Options are: • Honor: only nodes matching nodeAffinity/nodeSelector are included in the calculations. • Ignore: nodeAffinity/nodeSelector are ignored. All nodes are included in the calculations. If this value is nil, the behavior is equivalent to the Honor policy.
`nodeTaintsPolicy`	`string`	NodeTaintsPolicy indicates how we will treat node taints when calculating pod topology spread skew. Options are: • Honor: nodes without taints, along with tainted nodes for which the incoming pod has a toleration, are included. • Ignore: node taints are ignored. All nodes are included. If this value is nil, the behavior is equivalent to the Ignore policy.
`topologyKey`	`string`	TopologyKey is the key of node labels. Nodes that have a label with this key and identical values are

Property	Type	Description
		considered to be in the same topology. We consider each <key, value> as a "bucket", and try to put balanced number of pods into each bucket. We define a domain as a particular instance of a topology. Also, we define an eligible domain as a domain whose nodes meet the requirements of nodeAffinityPolicy and nodeTaintsPolicy. e.g. If TopologyKey is "kubernetes.io/hostname", each Node is a domain of that topology. And, if TopologyKey is "topology.kubernetes.io/zone", each zone is a domain of that topology. It's a required field.
`whenUnsatisfiable`	`string`	WhenUnsatisfiable indicates how to deal with a pod if it doesn't satisfy the spread constraint. • DoNotSchedule (default) tells the scheduler not to schedule it. • ScheduleAnyway tells the scheduler to schedule the pod in any location, but giving higher precedence to topologies that would help reduce the skew. A constraint is considered "Unsatisfiable" for an incoming pod if and only if every possible node assignment for that pod would violate "MaxSkew" on some topology. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 3/1/1: | zone1 | zone2 | zone3 | | P P P | P | P | If WhenUnsatisfiable is set to DoNotSchedule, incoming pod can only be scheduled to zone2(zone3) to become 3/2/1(3/1/2) as ActualSkew(2-1) on zone2(zone3) satisfies MaxSkew(1). In other words, the cluster can still be imbalanced, but

Property	Type	Description
		scheduler won't make it *more* imbalanced. It's a required field.

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector

Description

LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values[]

Type

`string`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].matchLabelKeys

Description

MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector

isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).

Type

`array`

.spec.taskRunSpecs[].podTemplate.topologySpreadConstraints[].matchLabelKeys[]

Type

`string`

.spec.taskRunSpecs[].sidecarSpecs

Type

`array`

.spec.taskRunSpecs[].sidecarSpecs[]

Description

TaskRunSidecarSpec is used to override the values of a Sidecar in the corresponding Task.

Type

`object`

Required

`computeResources` `name`

Property	Type	Description
`computeResources`	`object`	The resource requirements to apply to the Sidecar.

Property	Type	Description
`name`	`string`	The name of the Sidecar to override.

.spec.taskRunSpecs[].sidecarSpecs[].computeResources

Description

The resource requirements to apply to the Sidecar.

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.taskRunSpecs[].sidecarSpecs[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.taskRunSpecs[].sidecarSpecs[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.

Property	Type	Description
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.taskRunSpecs[].sidecarSpecs[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunSpecs[].sidecarSpecs[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunSpecs[].stepSpecs

Type

`array`

.spec.taskRunSpecs[].stepSpecs[]

Description

TaskRunStepSpec is used to override the values of a Step in the corresponding Task.

Type

`object`

Required

`computeResources` `name`

Property	Type	Description
`computeResources`	`object`	The resource requirements to apply to the Step.
`name`	`string`	The name of the Step to override.

.spec.taskRunSpecs[].stepSpecs[].computeResources

Description

The resource requirements to apply to the Step.

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.taskRunSpecs[].stepSpecs[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.taskRunSpecs[].stepSpecs[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.taskRunSpecs[].stepSpecs[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunSpecs[].stepSpecs[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRunTemplate

Description

TaskRunTemplate represent template of taskrun

Type

`object`

Property	Type	Description
`podTemplate`	`object`	PodTemplate holds pod specific configuration
`serviceAccountName`	`string`	

.spec.taskRunTemplate.podTemplate

Description

PodTemplate holds pod specific configuration

Type

`object`

Property	Type	Description
`affinity`		If specified, the pod's scheduling constraints Pod.spec.affinity (API version: v1)
`automountServiceAccountToken`	`boolean`	AutomountServiceAccountToken indicates w running as this service account should have token automatically mounted.
`dnsConfig`	`object`	Specifies the DNS parameters of a pod. Par specified here will be merged to the generat configuration based on DNSPolicy.
`dnsPolicy`	`string`	Set DNS policy for the pod. Defaults to "Clu Valid values are 'ClusterFirst', 'Default' or 'N parameters given in DNSConfig will be merg policy selected with DNSPolicy.
`enableServiceLinks`	`boolean`	EnableServiceLinks indicates whether inforr services should be injected into pod's envirc variables, matching the syntax of Docker lin Defaults to true.
`env`	`array`	List of environment variables that can be pro containers belonging to the pod.
`hostAliases`	`array`	HostAliases is an optional list of hosts and I be injected into the pod's hosts file if specifie

Property	Type	Description
		only valid for non-hostNetwork pods.
`hostNetwork`	`boolean`	HostNetwork specifies whether the pod may node network namespace
`hostUsers`	`boolean`	HostUsers indicates whether the pod will us user namespace. Optional: Default to true. I or not present, the pod will be run in the hos namespace, useful for when the pod needs only available to the host user namespace, loading a kernel module with CAP_SYS_MO When set to false, a new user namespace is the pod. Setting false is useful to mitigating breakout vulnerabilities such as allowing co run as root without their user having root pri the host. This field depends on the kubernet gate UserNamespacesSupport being enable
`imagePullSecrets`	`array`	ImagePullSecrets gives the name of the sec the pod to pull the image if specified
`nodeSelector`	`object`	NodeSelector is a selector which must be tr pod to fit on a node. Selector which must ma labels for the pod to be scheduled on that no info: https://kubernetes.io/docs/concepts/configu pod-node/↗

Property	Type	Description
`priorityClassName`	`string`	If specified, indicates the pod's priority. "syst critical" and "system-cluster-critical" are two keywords which indicate the highest prioritie former being the highest priority. Any other r be defined by creating a PriorityClass objec name. If not specified, the pod priority will b zero if there is no default.
`runtimeClassName`	`string`	RuntimeClassName refers to a RuntimeClas the node.k8s.io group, which should be use pod. If no RuntimeClass resource matches t class, the pod will not be run. If unset or em "legacy" RuntimeClass will be used, which i class with an empty definition that uses the runtime handler. More info: https://git.k8s.io/enhancements/keps/sig-no class.md↗ This is a beta feature as of Kube v1.14.
`schedulerName`	`string`	SchedulerName specifies the scheduler to b dispatch the Pod
`securityContext`		SecurityContext holds pod-level security att common container settings. Optional: Defau See type description for default values of ea Pod.spec.securityContext (API version: v1)

Property	Type	Description
`tolerations`	`array`	If specified, the pod's tolerations.
`topologySpreadConstraints`	`array`	TopologySpreadConstraints controls how Pc spread across your cluster among failure-dc as regions, zones, nodes, and other user-de topology domains.
`volumes`		List of volumes that can be mounted by con belonging to the pod. More info: https://kubernetes.io/docs/concepts/storage See Pod.spec.volumes (API version: v1)

.spec.taskRunTemplate.podTemplate.dnsConfig

Description

Specifies the DNS parameters of a pod. Parameters specified here will be merged to the generated DNS configuration based on DNSPolicy.

Type

`object`

Property	Type	Description
`nameservers`	`array`	A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Property	Type	Description
`options`	`array`	A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.
`searches`	`array`	A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

.spec.taskRunTemplate.podTemplate.dnsConfig.nameservers

Description

A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Type

`array`

.spec.taskRunTemplate.podTemplate.dnsConfig.nameservers[]

Type

`string`

.spec.taskRunTemplate.podTemplate.dnsConfig.options

Description

A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.

Type

`array`

.spec.taskRunTemplate.podTemplate.dnsConfig.options[]

Description

PodDNSConfigOption defines DNS resolver options of a pod.

Type

`object`

Property	Type	Description
`name`	`string`	Name is this DNS resolver option's name. Required.
`value`	`string`	Value is this DNS resolver option's value.

.spec.taskRunTemplate.podTemplate.dnsConfig.searches

Description

A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

Type

`array`

.spec.taskRunTemplate.podTemplate.dnsConfig.searches[]

Type

`string`

.spec.taskRunTemplate.podTemplate.env

Description

List of environment variables that can be provided to the containers belonging to the pod.

Type

`array`

.spec.taskRunTemplate.podTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal

Property	Type	Description
		"$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.taskRunTemplate.podTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Property	Type	Description
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.taskRunTemplate.podTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.taskRunTemplate.podTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.taskRunTemplate.podTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.taskRunTemplate.podTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.taskRunTemplate.podTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.taskRunTemplate.podTemplate.hostAliases

Description

HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified. This is only valid for non-hostNetwork pods.

Type

`array`

.spec.taskRunTemplate.podTemplate.hostAliases[]

Description

HostAlias holds the mapping between IP and hostnames that will be injected as an entry in the pod's hosts file.

Type

`object`

Required

`ip`

Property	Type	Description
`hostnames`	`array`	Hostnames for the above IP address.
`ip`	`string`	IP address of the host file entry.

.spec.taskRunTemplate.podTemplate.hostAliases[].hostnames

Description

Hostnames for the above IP address.

Type

`array`

.spec.taskRunTemplate.podTemplate.hostAliases[].hostnames[]

Type

`string`

.spec.taskRunTemplate.podTemplate.imagePullSecrets

Description

ImagePullSecrets gives the name of the secret used by the pod to pull the image if specified

Type

`array`

.spec.taskRunTemplate.podTemplate.imagePullSecrets[]

Description

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.taskRunTemplate.podTemplate.nodeSelector

Description

NodeSelector is a selector which must be true for the pod to fit on a node. Selector which must match a node's labels for the pod to be scheduled on that node. More info: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

Type

`object`

.spec.taskRunTemplate.podTemplate.tolerations

Description

If specified, the pod's tolerations.

Type

`array`

.spec.taskRunTemplate.podTemplate.tolerations[]

Description

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator <operator>.

Type

`object`

Property	Type	Description
`effect`	`string`	Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.
`key`	`string`	Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.
`operator`	`string`	Operator represents a key's relationship to the value. Valid operators are Exists, Equal, Lt, and Gt. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a

Property	Type	Description
		particular category. Lt and Gt perform numeric comparisons (requires feature gate TaintTolerationComparisonOperators).
`tolerationSeconds`	`integer`	TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.
`value`	`string`	Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints

Description

TopologySpreadConstraints controls how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains.

Type

`array`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[]

Description

TopologySpreadConstraint specifies how to spread matching pods among the given topology.

Type

`object`

Required

`maxSkew` `topologyKey` `whenUnsatisfiable`

Property	Type	Description
`labelSelector`	`object`	LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.
`matchLabelKeys`	`array`	MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).
`maxSkew`	`integer`	MaxSkew describes the degree to which pods may be unevenly distributed. When

Property	Type	Description
		`whenUnsatisfiable=DoNotSchedule`, it is the maximum permitted difference between the number of matching pods in the target topology and the global minimum. The global minimum is the minimum number of matching pods in an eligible domain or zero if the number of eligible domains is less than MinDomains. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 2/2/1: In this case, the global minimum is 1. | zone1 | zone2 | zone3 | | P P | P P | P | • if MaxSkew is 1, incoming pod can only be scheduled to zone3 to become 2/2/2; scheduling it onto zone1(zone2) would make the ActualSkew(3-1) on zone1(zone2) violate MaxSkew(1). • if MaxSkew is 2, incoming pod can be scheduled onto any zone. When `whenUnsatisfiable=ScheduleAnyway`, it is used to give higher precedence to topologies that satisfy it. It's a required field. Default value is 1 and 0 is not allowed.
`minDomains`	`integer`	MinDomains indicates a minimum number of eligible domains. When the number of eligible domains with matching topology keys is less than minDomains, Pod Topology Spread treats "global minimum" as 0, and then the calculation of Skew is performed. And when the number of eligible domains with matching topology keys equals or greater than minDomains, this value has no effect on scheduling. As a result, when the number of eligible domains is less than minDomains,

Property	Type	Description
		scheduler won't schedule more than maxSkew Pods to those domains. If value is nil, the constraint behaves as if MinDomains is equal to 1. Valid values are integers greater than 0. When value is not nil, WhenUnsatisfiable must be DoNotSchedule. For example, in a 3-zone cluster, MaxSkew is set to 2, MinDomains is set to 5 and pods with the same labelSelector spread as 2/2/2: | zone1 | zone2 | zone3 | | P P | P P | P P | The number of domains is less than 5(MinDomains), so "global minimum" is treated as 0. In this situation, new pod with the same labelSelector cannot be scheduled, because computed skew will be 3(3 - 0) if new Pod is scheduled to any of the three zones, it will violate MaxSkew.
`nodeAffinityPolicy`	`string`	NodeAffinityPolicy indicates how we will treat Pod's nodeAffinity/nodeSelector when calculating pod topology spread skew. Options are: • Honor: only nodes matching nodeAffinity/nodeSelector are included in the calculations. • Ignore: nodeAffinity/nodeSelector are ignored. All nodes are included in the calculations. If this value is nil, the behavior is equivalent to the Honor policy.
`nodeTaintsPolicy`	`string`	NodeTaintsPolicy indicates how we will treat node taints when calculating pod topology spread skew.

Property	Type	Description
		Options are: • Honor: nodes without taints, along with tainted nodes for which the incoming pod has a toleration, are included. • Ignore: node taints are ignored. All nodes are included. If this value is nil, the behavior is equivalent to the Ignore policy.
`topologyKey`	`string`	TopologyKey is the key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology. We consider each <key, value> as a "bucket", and try to put balanced number of pods into each bucket. We define a domain as a particular instance of a topology. Also, we define an eligible domain as a domain whose nodes meet the requirements of nodeAffinityPolicy and nodeTaintsPolicy. e.g. If TopologyKey is "kubernetes.io/hostname", each Node is a domain of that topology. And, if TopologyKey is "topology.kubernetes.io/zone", each zone is a domain of that topology. It's a required field.
`whenUnsatisfiable`	`string`	WhenUnsatisfiable indicates how to deal with a pod if it doesn't satisfy the spread constraint. • DoNotSchedule (default) tells the scheduler not to schedule it. • ScheduleAnyway tells the scheduler to schedule the pod in any location, but giving

Property	Type	Description
		higher precedence to topologies that would help reduce the skew. A constraint is considered "Unsatisfiable" for an incoming pod if and only if every possible node assignment for that pod would violate "MaxSkew" on some topology. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 3/1/1: | zone1 | zone2 | zone3 | | P P P | P | P | If WhenUnsatisfiable is set to DoNotSchedule, incoming pod can only be scheduled to zone2(zone3) to become 3/2/1(3/1/2) as ActualSkew(2-1) on zone2(zone3) satisfies MaxSkew(1). In other words, the cluster can still be imbalanced, but scheduler won't make it *more* imbalanced. It's a required field.

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector

Description

LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.

Property	Type	Description
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values[]

Type

`string`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].matchLabelKeys

Description

MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).

Type

`array`

.spec.taskRunTemplate.podTemplate.topologySpreadConstraints[].matchLabelKeys[]

Type

`string`

.spec.timeouts

Description

Time after which the Pipeline times out. Currently three keys are accepted in the map pipeline, tasks and finally with Timeouts.pipeline >= Timeouts.tasks + Timeouts.finally

Type

`object`

Property	Type	Description
`finally`	`string`	Finally sets the maximum allowed duration of this pipeline's finally
`pipeline`	`string`	Pipeline sets the maximum allowed duration for execution of the entire pipeline. The sum of individual timeouts for tasks and finally must not exceed this value.
`tasks`	`string`	Tasks sets the maximum allowed duration of this pipeline's tasks

.spec.workspaces

Description

Workspaces holds a set of workspace bindings that must match names with those declared in the pipeline.

Type

`array`

.spec.workspaces[]

Description

WorkspaceBinding maps a Task's declared workspace to a Volume.

Type

`object`

Required

`name`

Property	Type	Description
`configMap`	`object`	ConfigMap represents a configMap that should popu workspace.
`csi`	`object`	CSI (Container Storage Interface) represents ephem storage that is handled by certain external CSI drive
`emptyDir`	`object`	EmptyDir represents a temporary directory that shar Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes ↗ Either this OR PersistentVolumeClaim can be use
`name`	`string`	Name is the name of the workspace populated by th
`persistentVolumeClaim`	`object`	PersistentVolumeClaimVolumeSource represents a to a PersistentVolumeClaim in the same namespace this OR EmptyDir can be used.
`projected`	`object`	Projected represents a projected volume that should this workspace.

Property	Type	Description
`secret`	`object`	Secret represents a secret that should populate this workspace.
`subPath`	`string`	SubPath is optionally a directory on the volume whic be used for this binding (i.e. the volume will be mour sub directory).
`volumeClaimTemplate`		VolumeClaimTemplate is a template for a claim that created in the same namespace. The PipelineRun c responsible for creating a unique claim for each insta PipelineRun. See PersistentVolumeClaim (API versi

.spec.workspaces[].configMap

Description

ConfigMap represents a configMap that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the

Property	Type	Description
		file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not

be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].csi

Description

CSI (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.
`fsType`	`string`	fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.
`nodePublishSecretRef`	`object`	nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI

Property	Type	Description
		NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.
`readOnly`	`boolean`	readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

.spec.workspaces[].csi.nodePublishSecretRef

Description

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.workspaces[].csi.volumeAttributes

Description

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

Type

`object`

.spec.workspaces[].emptyDir

Description

EmptyDir represents a temporary directory that shares a Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir Either this OR PersistentVolumeClaim can be used.

Type

`object`

Property	Type	Description
`medium`	`string`	medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

Property	Type	Description
`sizeLimit`		sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

.spec.workspaces[].persistentVolumeClaim

Description

PersistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. Either this OR EmptyDir can be used.

Type

`object`

Required

`claimName`

Property	Type	Description
`claimName`	`string`	claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims ↗

Property	Type	Description
`readOnly`	`boolean`	readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

.spec.workspaces[].projected

Description

Projected represents a projected volume that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`sources`	`array`	sources is the list of volume projections. Each entry in this list handles one source.

.spec.workspaces[].projected.sources

Description

sources is the list of volume projections. Each entry in this list handles one source.

Type

`array`

.spec.workspaces[].projected.sources[]

Description

Projection that may be projected along with other supported volume types. Exactly one of these fields must be set.

Type

`object`

Property	Type	Description
`clusterTrustBundle`	`object`	ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Property	Type	Description
`configMap`	`object`	configMap information about the configMap data to project
`downwardAPI`	`object`	downwardAPI information about the downwardAPI data to project
`podCertificate`	`object`	Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries

Property	Type	Description
		are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.
`secret`	`object`	secret information about the secret data to project
`serviceAccountToken`	`object`	serviceAccountToken is information about the serviceAccountToken data to project

.spec.workspaces[].projected.sources[].clusterTrustBundle

Description

ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be

selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Type

`object`

Required

`path`

Property	Type	Description
`labelSelector`	`object`	Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".
`name`	`string`	Select a single ClusterTrustBundle by object name. Mutually-exclusive with signerName and labelSelector.
`optional`	`boolean`	If true, don't block pod startup if the referenced ClusterTrustBundle(s) aren't available. If using name, then the named ClusterTrustBundle is allowed not to exist. If using signerName, then the combination of signerName and labelSelector is allowed to match zero ClusterTrustBundles.
`path`	`string`	Relative path from the volume root to write the bundle.

Property	Type	Description
`signerName`	`string`	Select all ClusterTrustBundles that match this signer name. Mutually-exclusive with name. The contents of all selected ClusterTrustBundles will be unified and deduplicated.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector

Description

Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

Property	Type	Description
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].projected.sources[].configMap

Description

configMap information about the configMap data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].projected.sources[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that

Property	Type	Description
		affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].downwardAPI

Description

downwardAPI information about the downwardAPI data to project

Type

`object`

Property	Type	Description
`items`	`array`	Items is a list of DownwardAPIVolume file

.spec.workspaces[].projected.sources[].downwardAPI.items

Description

Items is a list of DownwardAPIVolume file

Type

`array`

.spec.workspaces[].projected.sources[].downwardAPI.items[]

Description

DownwardAPIVolumeFile represents information to create the file containing the pod field

Type

`object`

Required

`path`

Property	Type	Description
`fieldRef`	`object`	Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.
`mode`	`integer`	Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '..' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'

Property	Type	Description
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

.spec.workspaces[].projected.sources[].downwardAPI.items[].fieldRef

Description

Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.workspaces[].projected.sources[].downwardAPI.items[].resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.workspaces[].projected.sources[].podCertificate

Description

Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will

return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.

Type

`object`

Required

`keyType` `signerName`

Property	Type	Description
`certificateChainPath`	`string`	Write the certificate chain at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.
`credentialBundlePath`	`string`	Write the credential bundle at this path in the projected volume. The credential bundle is a single file that contains multiple PEM blocks. The first PEM block is a PRIVATE KEY block, containing a PKCS#8 private key. The remaining blocks are CERTIFICATE blocks, containing the issued certificate chain from the

Property	Type	Description
		signer (leaf and any intermediates). Using credentialBundlePath lets your Pod's application code make a single atomic read that retrieves a consistent key and certificate chain. If you project them to separate files, your application code will need to additionally check that the leaf certificate was issued to the key.
`keyPath`	`string`	Write the key at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.
`keyType`	`string`	The type of keypair Kubelet will generate for the pod. Valid values are "RSA3072", "RSA4096", "ECDSAP256", "ECDSAP384", "ECDSAP521", and "ED25519".
`maxExpirationSeconds`	`integer`	maxExpirationSeconds is the maximum lifetime permitted for the certificate. Kubelet copies this value verbatim into the PodCertificateRequests it generates for this projection.

Property	Type	Description
		If omitted, kube-apiserver will set it to 86400(24 hours). kube-apiserver will reject values shorter than 3600 (1 hour). The maximum allowable value is 7862400 (91 days). The signer implementation is then free to issue a certificate with any lifetime *shorter* than MaxExpirationSeconds, but no shorter than 3600 seconds (1 hour). This constraint is enforced by kube-apiserver. `kubernetes.io` signers will never issue certificates with a lifetime longer than 24 hours.
`signerName`	`string`	Kubelet's generated CSRs will be addressed to this signer.
`userAnnotations`	`object`	userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field.

Property	Type	Description
		Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

.spec.workspaces[].projected.sources[].podCertificate.userAnnotations

Description

userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

Type

`object`

.spec.workspaces[].projected.sources[].secret

Description

secret information about the secret data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If

Property	Type	Description
		specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional field specify whether the Secret or its key must be defined

.spec.workspaces[].projected.sources[].secret.items

Description

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].serviceAccountToken

Description

serviceAccountToken is information about the serviceAccountToken data to project

Type

`object`

Required

`path`

Property	Type	Description
`audience`	`string`	audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.
`expirationSeconds`	`integer`	expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent of its time to live or if the token is older than 24 hours.Defaults to 1 hour and must be at least 10 minutes.
`path`	`string`	path is the path relative to the mount point of the file to project the token into.

.spec.workspaces[].secret

Description

Secret represents a secret that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`optional`	`boolean`	optional field specify whether the Secret or its keys must be defined
`secretName`	`string`	secretName is the name of the secret in the pod's namespace to use. More info: https://kubernetes.io/docs/concepts/storage/volumes#secret ↗

.spec.workspaces[].secret.items

Description

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.status

Description

PipelineRunStatus defines the observed state of PipelineRun

Type

`object`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`childReferences`	`array`	list of TaskRun and Run names, PipelineTask names, and API versions/kinds for children of this PipelineRun.
`completionTime`	`string`	CompletionTime is the time the PipelineRun completed.

Property	Type	Description
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`finallyStartTime`	`string`	FinallyStartTime is when all non-finally tasks have been completed and only finally tasks are being executed.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`pipelineSpec`		PipelineSpec contains the exact spec used to instantiate the run. See Pipeline.spec (API version: tekton.dev/v1)
`provenance`	`object`	Provenance contains some key authenticated metadata about how a software artifact was built (what sources, what inputs/outputs, etc.).
`results`	`array`	Results are the list of results written out by the pipeline task's containers
`skippedTasks`	`array`	list of tasks that were skipped due to when expressions evaluating to false

Property	Type	Description
`spanContext`	`object`	SpanContext contains tracing span context fields
`startTime`	`string`	StartTime is the time the PipelineRun is actually started.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.childReferences

Description

list of TaskRun and Run names, PipelineTask names, and API versions/kinds for children of this PipelineRun.

Type

`array`

.status.childReferences[]

Description

ChildStatusReference is used to point to the statuses of individual TaskRuns and Runs within this PipelineRun.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	
`displayName`	`string`	DisplayName is a user-facing name of the pipelineTask that may be used to populate a UI.
`kind`	`string`	
`name`	`string`	Name is the name of the TaskRun or Run this is referencing.
`pipelineTaskName`	`string`	PipelineTaskName is the name of the PipelineTask this is referencing.
`whenExpressions`	`array`	WhenExpressions is the list of checks guarding the execution of the PipelineTask

.status.childReferences[].whenExpressions

Description

WhenExpressions is the list of checks guarding the execution of the PipelineTask

Type

`array`

.status.childReferences[].whenExpressions[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.status.childReferences[].whenExpressions[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.status.childReferences[].whenExpressions[].values[]

Type

`string`

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).

Property	Type	Description
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	Type of condition.

.status.provenance

Description

Provenance contains some key authenticated metadata about how a software artifact was built (what sources, what inputs/outputs, etc.).

Type

`object`

Property	Type	Description
`featureFlags`	`object`	FeatureFlags identifies the feature flags that were used during the task/pipeline run
`refSource`	`object`	RefSource identifies the source where a remote task/pipeline came from.

.status.provenance.featureFlags

Description

FeatureFlags identifies the feature flags that were used during the task/pipeline run

Type

`object`

Property	Type	Description
`awaitSidecarReadiness`	`boolean`	
`coschedule`	`string`	
`disableCredsInit`	`boolean`	
`disableInlineSpec`	`string`	
`enableAPIFields`	`string`	
`enableArtifacts`	`boolean`	
`enableCELInWhenExpression`	`boolean`	
`enableConciseResolverSyntax`	`boolean`	
`enableKeepPodOnCancel`	`boolean`	

Property	Type	Description
`enableKubernetesSidecar`	`boolean`	
`enableParamEnum`	`boolean`	
`enableProvenanceInStatus`	`boolean`	
`enableStepActions`	`boolean`	EnableStepActions is a no-op since StepActions are stable
`enableTektonOCIBundles`	`boolean`	DeprecatedEnableTektonOCI is maintained for backward compatibility to allow deletion PipelineRuns created before This field is not used and can removed in a future release o we're confident old PipelineRu have been cleaned up. See is #8359 for context.
`enableWaitExponentialBackoff`	`boolean`	
`enforceNonfalsifiability`	`string`	
`maxResultSize`	`integer`	
`requireGitSSHSecretKnownHosts`	`boolean`	
`resultExtractionMethod`	`string`	
`runningInEnvWithInjectedSidecars`	`boolean`	
`sendCloudEventsForRuns`	`boolean`	
`setSecurityContext`	`boolean`	

Property	Type	Description
`setSecurityContextReadOnlyRootFilesystem`	`boolean`	
`verificationNoMatchPolicy`	`string`	VerificationNoMatchPolicy is t feature flag for "trusted-resou verification-no-match-policy" VerificationNoMatchPolicy ca to "ignore", "warn" and "fail" v ignore: skip trusted resources verification when no matching verification policies found war trusted resources verification no matching verification polici found and log a warning fail: f taskrun or pipelines run if no matching verification policies

.status.provenance.refSource

Description

RefSource identifies the source where a remote task/pipeline came from.

Type

`object`

Property	Type	Description
`digest`	`object`	Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}

Property	Type	Description
`entryPoint`	`string`	EntryPoint identifies the entry point into the build. This is often a path to a build definition file and/or a target label within that file. Example: "task/git-clone/0.10/git-clone.yaml"
`uri`	`string`	URI indicates the identity of the source of the build definition. Example: "https://github.com/tektoncd/catalog↗"

.status.provenance.refSource.digest

Description

Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}

Type

`object`

.status.results

Description

Results are the list of results written out by the pipeline task's containers

Type

`array`

.status.results[]

Description

PipelineRunResult used to describe the results of a pipeline

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name is the result's name as declared by the Pipeline
`value`		Value is the result returned from the execution of this PipelineRun

.status.skippedTasks

Description

list of tasks that were skipped due to when expressions evaluating to false

Type

`array`

.status.skippedTasks[]

Description

SkippedTask is used to describe the Tasks that were skipped due to their When Expressions evaluating to False. This is a struct because we are looking into including more details about the When Expressions that caused this Task to be skipped.

Type

`object`

Required

`name` `reason`

Property	Type	Description
`name`	`string`	Name is the Pipeline Task name
`reason`	`string`	Reason is the cause of the PipelineTask being skipped.
`whenExpressions`	`array`	WhenExpressions is the list of checks guarding the execution of the PipelineTask

.status.skippedTasks[].whenExpressions

Description

WhenExpressions is the list of checks guarding the execution of the PipelineTask

Type

`array`

.status.skippedTasks[].whenExpressions[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the

Property	Type	Description
		result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.status.skippedTasks[].whenExpressions[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.status.skippedTasks[].whenExpressions[].values[]

Type

`string`

.status.spanContext

Description

SpanContext contains tracing span context fields

Type

`object`

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns`
  - `DELETE`: delete collection of PipelineRun
  - `GET`: list objects of kind PipelineRun
  - `POST`: create a new PipelineRun
- `/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns/{name}`
  - `DELETE`: delete the specified PipelineRun
  - `GET`: read the specified PipelineRun
  - `PATCH`: partially update the specified PipelineRun
  - `PUT`: replace the specified PipelineRun
- `/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns/{name}/status`
  - `GET`: read status of the specified PipelineRun
  - `PATCH`: partially update status of the specified PipelineRun
  - `PUT`: replace status of the specified PipelineRun

/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns

HTTP method

`DELETE`

Description

delete collection of PipelineRun

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind PipelineRun

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRunList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PipelineRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
201 - Created	`PipelineRun` schema
202 - Accepted	`PipelineRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns/{name}

HTTP method

`DELETE`

Description

delete the specified PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified PipelineRun

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PipelineRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
201 - Created	`PipelineRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/pipelineruns/{name}/status

HTTP method

`GET`

Description

read status of the specified PipelineRun

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields.

Parameter	Type	Description
		This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified PipelineRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object,

Parameter	Type	Description
		and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`PipelineRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`PipelineRun` schema
201 - Created	`PipelineRun` schema
401 - Unauthorized	Empty

TaskRun [tekton.dev/v1]

Description

TaskRun represents a single execution of a Task. TaskRuns are how the steps specified in a Task are executed; they specify the parameters and resources used to run the steps in a Task.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	TaskRunSpec defines the desired state of TaskRun
`status`	`object`	TaskRunStatus defines the observed state of TaskRun

.spec

Description

TaskRunSpec defines the desired state of TaskRun

Type

`object`

Property	Type	Description
`computeResources`	`object`	Compute resources to use for this TaskRun
`debug`	`object`	TaskRunDebug defines the breakpoint config for a particular TaskRun

Property	Type	Description
`managedBy`	`string`	ManagedBy indicates which controller is responsible for reconciling this resource. If unset or set to "tekton.dev/pipeline", the default Tekton controller will manage this resource. This field is immutable.
`params`	`array`	Params is a list of Param
`podTemplate`	`object`	PodTemplate holds pod specific configuration
`retries`	`integer`	Retries represents how many times this TaskRun should be retried in the event of task failure.
`serviceAccountName`	`string`	
`sidecarSpecs`	`array`	Specs to apply to Sidecars in this TaskRun. If a field is specified in both a Sidecar and a SidecarSpec, the value from the SidecarSpec will be used. This field is only supported when the alpha feature gate is enabled.
`status`	`string`	Used for cancelling a TaskRun (and maybe more later on)
`statusMessage`	`string`	Status message for cancellation.

Property	Type	Description
`stepSpecs`	`array`	Specs to apply to Steps in this TaskRun. If a field is specified in both a Step and a StepSpec, the value from the StepSpec will be used. This field is only supported when the alpha feature gate is enabled.
`taskRef`	`object`	no more than one of the TaskRef and TaskSpec may be specified.
`taskSpec`		Specifying TaskSpec can be disabled by setting `disable-inline-spec` feature flag. See Task.spec (API version: tekton.dev/v1)
`timeout`	`string`	Time after which one retry attempt times out. Defaults to 1 hour. Refer Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗
`workspaces`	`array`	Workspaces is a list of WorkspaceBindings from volumes to workspaces.

.spec.computeResources

Description

Compute resources to use for this TaskRun

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.debug

Description

TaskRunDebug defines the breakpoint config for a particular TaskRun

Type

`object`

Property	Type	Description
`breakpoints`	`object`	TaskBreakpoints defines the breakpoint config for a particular Task

.spec.debug.breakpoints

Description

TaskBreakpoints defines the breakpoint config for a particular Task

Type

`object`

Property	Type	Description
`beforeSteps`	`array`	

Property	Type	Description
`onFailure`	`string`	if enabled, pause TaskRun on failure of a step failed step will not exit

.spec.debug.breakpoints.beforeSteps

Type

`array`

.spec.debug.breakpoints.beforeSteps[]

Type

`string`

.spec.params

Description

Params is a list of Param

Type

`array`

.spec.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.podTemplate

Description

PodTemplate holds pod specific configuration

Type

`object`

Property	Type	Description
`affinity`		If specified, the pod's scheduling constraints Pod.spec.affinity (API version: v1)
`automountServiceAccountToken`	`boolean`	AutomountServiceAccountToken indicates w running as this service account should have token automatically mounted.
`dnsConfig`	`object`	Specifies the DNS parameters of a pod. Par specified here will be merged to the generat configuration based on DNSPolicy.
`dnsPolicy`	`string`	Set DNS policy for the pod. Defaults to "Clu Valid values are 'ClusterFirst', 'Default' or 'N parameters given in DNSConfig will be merg policy selected with DNSPolicy.

Property	Type	Description
`enableServiceLinks`	`boolean`	EnableServiceLinks indicates whether inforr services should be injected into pod's enviro variables, matching the syntax of Docker lin Defaults to true.
`env`	`array`	List of environment variables that can be pro containers belonging to the pod.
`hostAliases`	`array`	HostAliases is an optional list of hosts and I be injected into the pod's hosts file if specifie only valid for non-hostNetwork pods.
`hostNetwork`	`boolean`	HostNetwork specifies whether the pod may node network namespace
`hostUsers`	`boolean`	HostUsers indicates whether the pod will us user namespace. Optional: Default to true. I or not present, the pod will be run in the hos namespace, useful for when the pod needs only available to the host user namespace, s loading a kernel module with CAP_SYS_MO When set to false, a new user namespace is the pod. Setting false is useful to mitigating breakout vulnerabilities such as allowing co run as root without their user having root pri the host. This field depends on the kubernet gate UserNamespacesSupport being enable

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets gives the name of the sec the pod to pull the image if specified
`nodeSelector`	`object`	NodeSelector is a selector which must be tr pod to fit on a node. Selector which must ma labels for the pod to be scheduled on that n info: https://kubernetes.io/docs/concepts/configu pod-node/↗
`priorityClassName`	`string`	If specified, indicates the pod's priority. "syst critical" and "system-cluster-critical" are two keywords which indicate the highest prioritie former being the highest priority. Any other r be defined by creating a PriorityClass objec name. If not specified, the pod priority will b zero if there is no default.
`runtimeClassName`	`string`	RuntimeClassName refers to a RuntimeClas the node.k8s.io group, which should be use pod. If no RuntimeClass resource matches t class, the pod will not be run. If unset or em "legacy" RuntimeClass will be used, which i class with an empty definition that uses the runtime handler. More info: https://git.k8s.io/enhancements/keps/sig-no class.md↗ This is a beta feature as of Kube v1.14.

Property	Type	Description
`schedulerName`	`string`	SchedulerName specifies the scheduler to b dispatch the Pod
`securityContext`		SecurityContext holds pod-level security attı common container settings. Optional: Defau See type description for default values of ea Pod.spec.securityContext (API version: v1)
`tolerations`	`array`	If specified, the pod's tolerations.
`topologySpreadConstraints`	`array`	TopologySpreadConstraints controls how Pc spread across your cluster among failure-dc as regions, zones, nodes, and other user-de topology domains.
`volumes`		List of volumes that can be mounted by con belonging to the pod. More info: https://kubernetes.io/docs/concepts/storage See Pod.spec.volumes (API version: v1)

.spec.podTemplate.dnsConfig

Description

Specifies the DNS parameters of a pod. Parameters specified here will be merged to the generated DNS configuration based on DNSPolicy.

Type

`object`

Property	Type	Description
`nameservers`	`array`	A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.
`options`	`array`	A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.
`searches`	`array`	A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

.spec.podTemplate.dnsConfig.nameservers

Description

A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Type

`array`

.spec.podTemplate.dnsConfig.nameservers[]

Type

`string`

.spec.podTemplate.dnsConfig.options

Description

A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.

Type

`array`

.spec.podTemplate.dnsConfig.options[]

Description

PodDNSConfigOption defines DNS resolver options of a pod.

Type

`object`

Property	Type	Description
`name`	`string`	Name is this DNS resolver option's name. Required.
`value`	`string`	Value is this DNS resolver option's value.

.spec.podTemplate.dnsConfig.searches

Description

A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

Type

`array`

.spec.podTemplate.dnsConfig.searches[]

Type

`string`

.spec.podTemplate.env

Description

List of environment variables that can be provided to the containers belonging to the pod.

Type

`array`

.spec.podTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a

Property	Type	Description
		single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.podTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Property	Type	Description
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.podTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty.

Property	Type	Description
		Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.podTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.podTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.podTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.podTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.podTemplate.hostAliases

Description

HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified. This is only valid for non-hostNetwork pods.

Type

`array`

.spec.podTemplate.hostAliases[]

Description

HostAlias holds the mapping between IP and hostnames that will be injected as an entry in the pod's hosts file.

Type

`object`

Required

`ip`

Property	Type	Description
`hostnames`	`array`	Hostnames for the above IP address.
`ip`	`string`	IP address of the host file entry.

.spec.podTemplate.hostAliases[].hostnames

Description

Hostnames for the above IP address.

Type

`array`

.spec.podTemplate.hostAliases[].hostnames[]

Type

`string`

.spec.podTemplate.imagePullSecrets

Description

ImagePullSecrets gives the name of the secret used by the pod to pull the image if specified

Type

`array`

.spec.podTemplate.imagePullSecrets[]

Description

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.podTemplate.nodeSelector

Description

NodeSelector is a selector which must be true for the pod to fit on a node. Selector which must match a node's labels for the pod to be scheduled on that node. More info: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

Type

`object`

.spec.podTemplate.tolerations

Description

If specified, the pod's tolerations.

Type

`array`

.spec.podTemplate.tolerations[]

Description

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator <operator>.

Type

`object`

Property	Type	Description
`effect`	`string`	Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.
`key`	`string`	Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.
`operator`	`string`	Operator represents a key's relationship to the value. Valid operators are Exists, Equal, Lt, and Gt. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category. Lt and Gt perform numeric comparisons (requires feature gate TaintTolerationComparisonOperators).
`tolerationSeconds`	`integer`	TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By

Property	Type	Description
		default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.
`value`	`string`	Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

.spec.podTemplate.topologySpreadConstraints

Description

TopologySpreadConstraints controls how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[]

Description

TopologySpreadConstraint specifies how to spread matching pods among the given topology.

Type

`object`

Required

`maxSkew` `topologyKey` `whenUnsatisfiable`

Property	Type	Description
`labelSelector`	`object`	LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.
`matchLabelKeys`	`array`	MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).
`maxSkew`	`integer`	MaxSkew describes the degree to which pods may be unevenly distributed. When `whenUnsatisfiable=DoNotSchedule`, it is the maximum permitted difference between the number of matching pods in the target topology and the global minimum. The global minimum is the minimum number of matching pods in an eligible domain or zero if the number of eligible domains is less than MinDomains. For example, in

Property	Type	Description
		a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 2/2/1: In this case, the global minimum is 1. | zone1 | zone2 | zone3 | | P P | P P | P | • if MaxSkew is 1, incoming pod can only be scheduled to zone3 to become 2/2/2; scheduling it onto zone1(zone2) would make the ActualSkew(3-1) on zone1(zone2) violate MaxSkew(1). • if MaxSkew is 2, incoming pod can be scheduled onto any zone. When `whenUnsatisfiable=ScheduleAnyway`, it is used to give higher precedence to topologies that satisfy it. It's a required field. Default value is 1 and 0 is not allowed.
`minDomains`	`integer`	MinDomains indicates a minimum number of eligible domains. When the number of eligible domains with matching topology keys is less than minDomains, Pod Topology Spread treats "global minimum" as 0, and then the calculation of Skew is performed. And when the number of eligible domains with matching topology keys equals or greater than minDomains, this value has no effect on scheduling. As a result, when the number of eligible domains is less than minDomains, scheduler won't schedule more than maxSkew Pods to those domains. If value is nil, the constraint behaves as if MinDomains is equal to 1. Valid values are integers greater than 0. When value is not nil, WhenUnsatisfiable must be DoNotSchedule.

Property	Type	Description
		For example, in a 3-zone cluster, MaxSkew is set to 2, MinDomains is set to 5 and pods with the same labelSelector spread as 2/2/2: | zone1 | zone2 | zone3 | | P P | P P | P P | The number of domains is less than 5(MinDomains), so "global minimum" is treated as 0. In this situation, new pod with the same labelSelector cannot be scheduled, because computed skew will be 3(3 - 0) if new Pod is scheduled to any of the three zones, it will violate MaxSkew.
`nodeAffinityPolicy`	`string`	NodeAffinityPolicy indicates how we will treat Pod's nodeAffinity/nodeSelector when calculating pod topology spread skew. Options are: • Honor: only nodes matching nodeAffinity/nodeSelector are included in the calculations. • Ignore: nodeAffinity/nodeSelector are ignored. All nodes are included in the calculations. If this value is nil, the behavior is equivalent to the Honor policy.
`nodeTaintsPolicy`	`string`	NodeTaintsPolicy indicates how we will treat node taints when calculating pod topology spread skew. Options are: • Honor: nodes without taints, along with tainted nodes for which the incoming pod has a toleration, are included. • Ignore: node taints are ignored. All nodes are included.

Property	Type	Description
		If this value is nil, the behavior is equivalent to the Ignore policy.
`topologyKey`	`string`	TopologyKey is the key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology. We consider each <key, value> as a "bucket", and try to put balanced number of pods into each bucket. We define a domain as a particular instance of a topology. Also, we define an eligible domain as a domain whose nodes meet the requirements of nodeAffinityPolicy and nodeTaintsPolicy. e.g. If TopologyKey is "kubernetes.io/hostname", each Node is a domain of that topology. And, if TopologyKey is "topology.kubernetes.io/zone", each zone is a domain of that topology. It's a required field.
`whenUnsatisfiable`	`string`	WhenUnsatisfiable indicates how to deal with a pod if it doesn't satisfy the spread constraint. • DoNotSchedule (default) tells the scheduler not to schedule it. • ScheduleAnyway tells the scheduler to schedule the pod in any location, but giving higher precedence to topologies that would help reduce the skew. A constraint is considered "Unsatisfiable" for an incoming pod if and only if every possible node assignment for that pod would violate "MaxSkew" on some topology. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same

Property	Type	Description
		labelSelector spread as 3/1/1: | zone1 | zone2 | zone3 | | P P P | P | P | If WhenUnsatisfiable is set to DoNotSchedule, incoming pod can only be scheduled to zone2(zone3) to become 3/2/1(3/1/2) as ActualSkew(2-1) on zone2(zone3) satisfies MaxSkew(1). In other words, the cluster can still be imbalanced, but scheduler won't make it *more* imbalanced. It's a required field.

.spec.podTemplate.topologySpreadConstraints[].labelSelector

Description

LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

Property	Type	Description
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values[]

Type

`string`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.podTemplate.topologySpreadConstraints[].matchLabelKeys

Description

MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].matchLabelKeys[]

Type

`string`

.spec.sidecarSpecs

Description

Specs to apply to Sidecars in this TaskRun. If a field is specified in both a Sidecar and a SidecarSpec, the value from the SidecarSpec will be used. This field is only supported when the alpha feature gate is enabled.

Type

`array`

.spec.sidecarSpecs[]

Description

TaskRunSidecarSpec is used to override the values of a Sidecar in the corresponding Task.

Type

`object`

Required

`computeResources` `name`

Property	Type	Description
`computeResources`	`object`	The resource requirements to apply to the Sidecar.
`name`	`string`	The name of the Sidecar to override.

.spec.sidecarSpecs[].computeResources

Description

The resource requirements to apply to the Sidecar.

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.sidecarSpecs[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.sidecarSpecs[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.sidecarSpecs[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.sidecarSpecs[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.stepSpecs

Description

Specs to apply to Steps in this TaskRun. If a field is specified in both a Step and a StepSpec, the value from the StepSpec will be used. This field is only supported when the alpha feature gate is enabled.

Type

`array`

.spec.stepSpecs[]

Description

TaskRunStepSpec is used to override the values of a Step in the corresponding Task.

Type

`object`

Required

`computeResources` `name`

Property	Type	Description
`computeResources`	`object`	The resource requirements to apply to the Step.
`name`	`string`	The name of the Step to override.

.spec.stepSpecs[].computeResources

Description

The resource requirements to apply to the Step.

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.spec.stepSpecs[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.spec.stepSpecs[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.spec.stepSpecs[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.stepSpecs[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.spec.taskRef

Description

no more than one of the TaskRef and TaskSpec may be specified.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent Note: A Task with non-empty APIVersion and Kind is considered a Custom Task
`kind`	`string`	TaskKind indicates the Kind of the Task: 1. Namespaced Task when Kind is set to "Task". If Kind is "", it defaults to "Task". 2. Custom Task when Kind is non-empty and APIVersion is non-empty

Property	Type	Description
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.taskRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.taskRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.workspaces

Description

Workspaces is a list of WorkspaceBindings from volumes to workspaces.

Type

`array`

.spec.workspaces[]

Description

WorkspaceBinding maps a Task's declared workspace to a Volume.

Type

`object`

Required

`name`

Property	Type	Description
`configMap`	`object`	ConfigMap represents a configMap that should popu workspace.
`csi`	`object`	CSI (Container Storage Interface) represents ephem storage that is handled by certain external CSI drive

Property	Type	Description
`emptyDir`	`object`	EmptyDir represents a temporary directory that shar Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes ↗ Either this OR PersistentVolumeClaim can be use
`name`	`string`	Name is the name of the workspace populated by th
`persistentVolumeClaim`	`object`	PersistentVolumeClaimVolumeSource represents a to a PersistentVolumeClaim in the same namespace this OR EmptyDir can be used.
`projected`	`object`	Projected represents a projected volume that should this workspace.
`secret`	`object`	Secret represents a secret that should populate this workspace.
`subPath`	`string`	SubPath is optionally a directory on the volume whic be used for this binding (i.e. the volume will be mour sub directory).
`volumeClaimTemplate`		VolumeClaimTemplate is a template for a claim that created in the same namespace. The PipelineRun c responsible for creating a unique claim for each insta PipelineRun. See PersistentVolumeClaim (API versi

.spec.workspaces[].configMap

Description

ConfigMap represents a configMap that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info:

Property	Type	Description
		https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].csi

Description

CSI (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.
`fsType`	`string`	fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.
`nodePublishSecretRef`	`object`	nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.
`readOnly`	`boolean`	readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

.spec.workspaces[].csi.nodePublishSecretRef

Description

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.workspaces[].csi.volumeAttributes

Description

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

Type

`object`

.spec.workspaces[].emptyDir

Description

EmptyDir represents a temporary directory that shares a Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir Either this OR PersistentVolumeClaim can be used.

Type

`object`

Property	Type	Description
`medium`	`string`	medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗
`sizeLimit`		sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

.spec.workspaces[].persistentVolumeClaim

Description

PersistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. Either this OR EmptyDir can be used.

Type

`object`

Required

`claimName`

Property	Type	Description
`claimName`	`string`	claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims ↗
`readOnly`	`boolean`	readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

.spec.workspaces[].projected

Description

Projected represents a projected volume that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`sources`	`array`	sources is the list of volume projections. Each entry in this list handles one source.

.spec.workspaces[].projected.sources

Description

sources is the list of volume projections. Each entry in this list handles one source.

Type

`array`

.spec.workspaces[].projected.sources[]

Description

Projection that may be projected along with other supported volume types. Exactly one of these fields must be set.

Type

`object`

Property	Type	Description
`clusterTrustBundle`	`object`	ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector.

Property	Type	Description
		Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.
`configMap`	`object`	configMap information about the configMap data to project
`downwardAPI`	`object`	downwardAPI information about the downwardAPI data to project
`podCertificate`	`object`	Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp.

Property	Type	Description
		Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.
`secret`	`object`	secret information about the secret data to project
`serviceAccountToken`	`object`	serviceAccountToken is information about the serviceAccountToken data to project

.spec.workspaces[].projected.sources[].clusterTrustBundle

Description

ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Type

`object`

Required

`path`

Property	Type	Description
`labelSelector`	`object`	Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".
`name`	`string`	Select a single ClusterTrustBundle by object name. Mutually-exclusive with signerName and labelSelector.
`optional`	`boolean`	If true, don't block pod startup if the referenced ClusterTrustBundle(s) aren't available. If using name, then the named ClusterTrustBundle is allowed not to exist. If using signerName, then the combination of

Property	Type	Description
		signerName and labelSelector is allowed to match zero ClusterTrustBundles.
`path`	`string`	Relative path from the volume root to write the bundle.
`signerName`	`string`	Select all ClusterTrustBundles that match this signer name. Mutually-exclusive with name. The contents of all selected ClusterTrustBundles will be unified and deduplicated.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector

Description

Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is

Property	Type	Description
		"key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.

Property	Type	Description
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].projected.sources[].configMap

Description

configMap information about the configMap data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

Property	Type	Description
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].projected.sources[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.

Property	Type	Description
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].downwardAPI

Description

downwardAPI information about the downwardAPI data to project

Type

`object`

Property	Type	Description
`items`	`array`	Items is a list of DownwardAPIVolume file

.spec.workspaces[].projected.sources[].downwardAPI.items

Description

Items is a list of DownwardAPIVolume file

Type

`array`

.spec.workspaces[].projected.sources[].downwardAPI.items[]

Description

DownwardAPIVolumeFile represents information to create the file containing the pod field

Type

`object`

Required

`path`

Property	Type	Description
`fieldRef`	`object`	Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.
`mode`	`integer`	Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`path`	`string`	Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '..' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

.spec.workspaces[].projected.sources[].downwardAPI.items[].fieldRef

Description

Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".

Property	Type	Description
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.workspaces[].projected.sources[].downwardAPI.items[].resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.workspaces[].projected.sources[].podCertificate

Description

Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.

Type

`object`

Required

`keyType` `signerName`

Property	Type	Description
`certificateChainPath`	`string`	Write the certificate chain at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.

Property	Type	Description
`credentialBundlePath`	`string`	Write the credential bundle at this path in the projected volume. The credential bundle is a single file that contains multiple PEM blocks. The first PEM block is a PRIVATE KEY block, containing a PKCS#8 private key. The remaining blocks are CERTIFICATE blocks, containing the issued certificate chain from the signer (leaf and any intermediates). Using credentialBundlePath lets your Pod's application code make a single atomic read that retrieves a consistent key and certificate chain. If you project them to separate files, your application code will need to additionally check that the leaf certificate was issued to the key.
`keyPath`	`string`	Write the key at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.
`keyType`	`string`	The type of keypair Kubelet will generate for the pod.

Property	Type	Description
		Valid values are "RSA3072", "RSA4096", "ECDSAP256", "ECDSAP384", "ECDSAP521", and "ED25519".
`maxExpirationSeconds`	`integer`	maxExpirationSeconds is the maximum lifetime permitted for the certificate. Kubelet copies this value verbatim into the PodCertificateRequests it generates for this projection. If omitted, kube-apiserver will set it to 86400(24 hours). kube-apiserver will reject values shorter than 3600 (1 hour). The maximum allowable value is 7862400 (91 days). The signer implementation is then free to issue a certificate with any lifetime *shorter* than MaxExpirationSeconds, but no shorter than 3600 seconds (1 hour). This constraint is enforced by kube-apiserver. `kubernetes.io` signers will never issue certificates with a lifetime longer than 24 hours.
`signerName`	`string`	Kubelet's generated CSRs will be addressed to this signer.
`userAnnotations`	`object`	userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way.

Property	Type	Description
		These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

.spec.workspaces[].projected.sources[].podCertificate.userAnnotations

Description

userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

Type

`object`

.spec.workspaces[].projected.sources[].secret

Description

secret information about the secret data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional field specify whether the Secret or its key must be defined

.spec.workspaces[].projected.sources[].secret.items

Description

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not

be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

Property	Type	Description
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].serviceAccountToken

Description

serviceAccountToken is information about the serviceAccountToken data to project

Type

`object`

Required

`path`

Property	Type	Description
`audience`	`string`	audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.
`expirationSeconds`	`integer`	expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent of its time to live or if

Property	Type	Description
		the token is older than 24 hours.Defaults to 1 hour and must be at least 10 minutes.
`path`	`string`	path is the path relative to the mount point of the file to project the token into.

.spec.workspaces[].secret

Description

Secret represents a secret that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is

Property	Type	Description
		specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`optional`	`boolean`	optional field specify whether the Secret or its keys must be defined
`secretName`	`string`	secretName is the name of the secret in the pod's namespace to use. More info: https://kubernetes.io/docs/concepts/storage/volumes#secret ↗

.spec.workspaces[].secret.items

Description

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.status

Description

TaskRunStatus defines the observed state of TaskRun

Type

`object`

Required

`podName`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`artifacts`	`object`	Artifacts are the list of artifacts written out by the task's containers
`completionTime`	`string`	CompletionTime is the time the build completed.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`podName`	`string`	PodName is the name of the pod responsible for executing this task's steps.
`provenance`	`object`	Provenance contains some key authenticated metadata about how a software artifact was built (what sources, what inputs/outputs, etc.).

Property	Type	Description
`results`	`array`	Results are the list of results written out by the task's containers
`retriesStatus`		RetriesStatus contains the history of TaskRunStatus in case of a retry in order to keep record of failures. All TaskRunStatus stored in RetriesStatus will have no date within the RetriesStatus as is redundant.
`sidecars`	`array`	The list has one entry per sidecar in the manifest. Each entry is represents the imageid of the corresponding sidecar.
`spanContext`	`object`	SpanContext contains tracing span context fields
`startTime`	`string`	StartTime is the time the build is actually started.
`steps`	`array`	Steps describes the state of each build step container.
`taskSpec`	`object`	TaskSpec contains the Spec from the dereferenced Task definition used to instantiate this TaskRun.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.artifacts

Description

Artifacts are the list of artifacts written out by the task's containers

Type

`object`

Property	Type	Description
`inputs`	`array`	
`outputs`	`array`	

.status.artifacts.inputs

Type

`array`

.status.artifacts.inputs[]

Description

Artifact represents an artifact within a system, potentially containing multiple values associated with it.

Type

`object`

Property	Type	Description
`buildOutput`	`boolean`	Indicate if the artifact is a build output or a by-product
`name`	`string`	The artifact's identifying category name
`values`	`array`	A collection of values related to the artifact

.status.artifacts.inputs[].values

Description

A collection of values related to the artifact

Type

`array`

.status.artifacts.inputs[].values[]

Description

ArtifactValue represents a specific value or data element within an Artifact.

Type

`object`

Property	Type	Description
`digest`	`object`	
`uri`	`string`	

.status.artifacts.inputs[].values[].digest

Type

`object`

.status.artifacts.outputs

Type

`array`

.status.artifacts.outputs[]

Description

Artifact represents an artifact within a system, potentially containing multiple values associated with it.

Type

`object`

Property	Type	Description
`buildOutput`	`boolean`	Indicate if the artifact is a build output or a by-product
`name`	`string`	The artifact's identifying category name
`values`	`array`	A collection of values related to the artifact

.status.artifacts.outputs[].values

Description

A collection of values related to the artifact

Type

`array`

.status.artifacts.outputs[].values[]

Description

ArtifactValue represents a specific value or data element within an Artifact.

Type

`object`

Property	Type	Description
`digest`	`object`	
`uri`	`string`	

.status.artifacts.outputs[].values[].digest

Type

`object`

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.

Property	Type	Description
`type`	`string`	Type of condition.

.status.provenance

Description

Provenance contains some key authenticated metadata about how a software artifact was built (what sources, what inputs/outputs, etc.).

Type

`object`

Property	Type	Description
`featureFlags`	`object`	FeatureFlags identifies the feature flags that were used during the task/pipeline run
`refSource`	`object`	RefSource identifies the source where a remote task/pipeline came from.

.status.provenance.featureFlags

Description

FeatureFlags identifies the feature flags that were used during the task/pipeline run

Type

`object`

Property	Type	Description
`awaitSidecarReadiness`	`boolean`	
`coschedule`	`string`	
`disableCredsInit`	`boolean`	
`disableInlineSpec`	`string`	
`enableAPIFields`	`string`	
`enableArtifacts`	`boolean`	
`enableCELInWhenExpression`	`boolean`	
`enableConciseResolverSyntax`	`boolean`	
`enableKeepPodOnCancel`	`boolean`	
`enableKubernetesSidecar`	`boolean`	
`enableParamEnum`	`boolean`	
`enableProvenanceInStatus`	`boolean`	
`enableStepActions`	`boolean`	EnableStepActions is a no-op since StepActions are stable

Property	Type	Description
`enableTektonOCIBundles`	`boolean`	DeprecatedEnableTektonOCI is maintained for backward compatibility to allow deletion PipelineRuns created before v This field is not used and can removed in a future release o we're confident old PipelineRu have been cleaned up. See is #8359 for context.
`enableWaitExponentialBackoff`	`boolean`	
`enforceNonfalsifiability`	`string`	
`maxResultSize`	`integer`	
`requireGitSSHSecretKnownHosts`	`boolean`	
`resultExtractionMethod`	`string`	
`runningInEnvWithInjectedSidecars`	`boolean`	
`sendCloudEventsForRuns`	`boolean`	
`setSecurityContext`	`boolean`	
`setSecurityContextReadOnlyRootFilesystem`	`boolean`	
`verificationNoMatchPolicy`	`string`	VerificationNoMatchPolicy is t feature flag for "trusted-resou verification-no-match-policy" VerificationNoMatchPolicy ca to "ignore", "warn" and "fail" v ignore: skip trusted resources verification when no matching

Property	Type	Description
		verification policies found war trusted resources verification no matching verification polici found and log a warning fail: f taskrun or pipelines run if no matching verification policies

.status.provenance.refSource

Description

RefSource identifies the source where a remote task/pipeline came from.

Type

`object`

Property	Type	Description
`digest`	`object`	Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}
`entryPoint`	`string`	EntryPoint identifies the entry point into the build. This is often a path to a build definition file and/or a target label within that file. Example: "task/git-clone/0.10/git-clone.yaml"
`uri`	`string`	URI indicates the identity of the source of the build definition. Example: "https://github.com/tektoncd/catalog↗"

.status.provenance.refSource.digest

Description

Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}

Type

`object`

.status.results

Description

Results are the list of results written out by the task's containers

Type

`array`

.status.results[]

Description

TaskRunResult used to describe the results of a task

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name the given name

Property	Type	Description
`type`	`string`	Type is the user-specified type of the result. The possible type is currently "string" and will support "array" in following work.
`value`		Value the given value of the result

.status.sidecars

Description

The list has one entry per sidecar in the manifest. Each entry is represents the imageid of the corresponding sidecar.

Type

`array`

.status.sidecars[]

Description

SidecarState reports the results of running a sidecar in a Task.

Type

`object`

Property	Type	Description
`container`	`string`	
`imageID`	`string`	
`name`	`string`	

Property	Type	Description
`running`	`object`	Details about a running container
`terminated`	`object`	Details about a terminated container
`waiting`	`object`	Details about a waiting container

.status.sidecars[].running

Description

Details about a running container

Type

`object`

Property	Type	Description
`startedAt`	`string`	Time at which the container was last (re-)started

.status.sidecars[].terminated

Description

Details about a terminated container

Type

`object`

Required

`exitCode`

Property	Type	Description
`containerID`	`string`	Container's ID in the format '://<container_id>'
`exitCode`	`integer`	Exit status from the last termination of the container
`finishedAt`	`string`	Time at which the container last terminated
`message`	`string`	Message regarding the last termination of the container
`reason`	`string`	(brief) reason from the last termination of the container
`signal`	`integer`	Signal from the last termination of the container
`startedAt`	`string`	Time at which previous execution of the container started

.status.sidecars[].waiting

Description

Details about a waiting container

Type

`object`

Property	Type	Description
`message`	`string`	Message regarding why the container is not yet running.
`reason`	`string`	(brief) reason the container is not yet running.

.status.spanContext

Description

SpanContext contains tracing span context fields

Type

`object`

.status.steps

Description

Steps describes the state of each build step container.

Type

`array`

.status.steps[]

Description

StepState reports the results of running a step in a Task.

Type

`object`

Property	Type	Description
`container`	`string`	

Property	Type	Description
`imageID`	`string`	
`inputs`	`array`	
`name`	`string`	
`outputs`	`array`	
`provenance`	`object`	Provenance contains metadata about resources used in the TaskRun/PipelineRun such as the source from where a remote build definition was fetched. This field aims to carry minimum amoumt of metadata in *Run status so that Tekton Chains can capture them in the provenance.
`results`	`array`	
`running`	`object`	Details about a running container
`terminated`	`object`	Details about a terminated container
`terminationReason`	`string`	
`waiting`	`object`	Details about a waiting container

.status.steps[].inputs

Type

`array`

.status.steps[].inputs[]

Description

Artifact represents an artifact within a system, potentially containing multiple values associated with it.

Type

`object`

Property	Type	Description
`buildOutput`	`boolean`	Indicate if the artifact is a build output or a by-product
`name`	`string`	The artifact's identifying category name
`values`	`array`	A collection of values related to the artifact

.status.steps[].inputs[].values

Description

A collection of values related to the artifact

Type

`array`

.status.steps[].inputs[].values[]

Description

ArtifactValue represents a specific value or data element within an Artifact.

Type

`object`

Property	Type	Description
`digest`	`object`	
`uri`	`string`	

.status.steps[].inputs[].values[].digest

Type

`object`

.status.steps[].outputs

Type

`array`

.status.steps[].outputs[]

Description

Artifact represents an artifact within a system, potentially containing multiple values associated with it.

Type

`object`

Property	Type	Description
`buildOutput`	`boolean`	Indicate if the artifact is a build output or a by-product
`name`	`string`	The artifact's identifying category name

Property	Type	Description
`values`	`array`	A collection of values related to the artifact

.status.steps[].outputs[].values

Description

A collection of values related to the artifact

Type

`array`

.status.steps[].outputs[].values[]

Description

ArtifactValue represents a specific value or data element within an Artifact.

Type

`object`

Property	Type	Description
`digest`	`object`	
`uri`	`string`	

.status.steps[].outputs[].values[].digest

Type

`object`

.status.steps[].provenance

Description

Provenance contains metadata about resources used in the TaskRun/PipelineRun such as the source from where a remote build definition was fetched. This field aims to carry minimum amoumt of metadata in *Run status so that Tekton Chains can capture them in the provenance.

Type

`object`

Property	Type	Description
`featureFlags`	`object`	FeatureFlags identifies the feature flags that were used during the task/pipeline run
`refSource`	`object`	RefSource identifies the source where a remote task/pipeline came from.

.status.steps[].provenance.featureFlags

Description

FeatureFlags identifies the feature flags that were used during the task/pipeline run

Type

`object`

Property	Type	Description
`awaitSidecarReadiness`	`boolean`	
`coschedule`	`string`	
`disableCredsInit`	`boolean`	
`disableInlineSpec`	`string`	

Property	Type	Description
`enableAPIFields`	`string`	
`enableArtifacts`	`boolean`	
`enableCELInWhenExpression`	`boolean`	
`enableConciseResolverSyntax`	`boolean`	
`enableKeepPodOnCancel`	`boolean`	
`enableKubernetesSidecar`	`boolean`	
`enableParamEnum`	`boolean`	
`enableProvenanceInStatus`	`boolean`	
`enableStepActions`	`boolean`	EnableStepActions is a no-op since StepActions are stable
`enableTektonOCIBundles`	`boolean`	DeprecatedEnableTektonOCI is maintained for backward compatibility to allow deletion PipelineRuns created before This field is not used and can removed in a future release o we're confident old PipelineR have been cleaned up. See is #8359 for context.
`enableWaitExponentialBackoff`	`boolean`	
`enforceNonfalsifiability`	`string`	
`maxResultSize`	`integer`	

Property	Type	Description
`requireGitSSHSecretKnownHosts`	`boolean`	
`resultExtractionMethod`	`string`	
`runningInEnvWithInjectedSidecars`	`boolean`	
`sendCloudEventsForRuns`	`boolean`	
`setSecurityContext`	`boolean`	
`setSecurityContextReadOnlyRootFilesystem`	`boolean`	
`verificationNoMatchPolicy`	`string`	VerificationNoMatchPolicy is t feature flag for "trusted-resou verification-no-match-policy" VerificationNoMatchPolicy ca to "ignore", "warn" and "fail" v ignore: skip trusted resources verification when no matching verification policies found war trusted resources verification no matching verification polici found and log a warning fail: f taskrun or pipelines run if no matching verification policies

.status.steps[].provenance.refSource

Description

RefSource identifies the source where a remote task/pipeline came from.

Type

`object`

Property	Type	Description
`digest`	`object`	Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}
`entryPoint`	`string`	EntryPoint identifies the entry point into the build. This is often a path to a build definition file and/or a target label within that file. Example: "task/git-clone/0.10/git-clone.yaml"
`uri`	`string`	URI indicates the identity of the source of the build definition. Example: "https://github.com/tektoncd/catalog↗"

.status.steps[].provenance.refSource.digest

Description

Digest is a collection of cryptographic digests for the contents of the artifact specified by URI. Example: {"sha1": "f99d13e554ffcb696dee719fa85b695cb5b0f428"}

Type

`object`

.status.steps[].results

Type

`array`

.status.steps[].results[]

Description

TaskRunResult used to describe the results of a task

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name the given name
`type`	`string`	Type is the user-specified type of the result. The possible type is currently "string" and will support "array" in following work.
`value`		Value the given value of the result

.status.steps[].running

Description

Details about a running container

Type

`object`

Property	Type	Description
`startedAt`	`string`	Time at which the container was last (re-)started

.status.steps[].terminated

Description

Details about a terminated container

Type

`object`

Required

`exitCode`

Property	Type	Description
`containerID`	`string`	Container's ID in the format '://<container_id>'
`exitCode`	`integer`	Exit status from the last termination of the container
`finishedAt`	`string`	Time at which the container last terminated
`message`	`string`	Message regarding the last termination of the container
`reason`	`string`	(brief) reason from the last termination of the container
`signal`	`integer`	Signal from the last termination of the container
`startedAt`	`string`	Time at which previous execution of the container started

.status.steps[].waiting

Description

Details about a waiting container

Type

`object`

Property	Type	Description
`message`	`string`	Message regarding why the container is not yet running.
`reason`	`string`	(brief) reason the container is not yet running.

.status.taskSpec

Description

TaskSpec contains the Spec from the dereferenced Task definition used to instantiate this TaskRun.

Type

`object`

Property	Type	Description
`description`	`string`	Description is a user-facing description of the task that may be used to populate a UI.
`displayName`	`string`	DisplayName is a user-facing name of the task that may be used to populate a UI.
`params`	`array`	Params is a list of input parameters required to run the task. Params must be supplied as inputs in TaskRuns

Property	Type	Description
		unless they declare a default value.
`results`	`array`	Results are values that this Task can output
`sidecars`	`array`	Sidecars are run alongside the Task's step containers. They begin before the steps start and end after the steps complete.
`stepTemplate`	`object`	StepTemplate can be used as the basis for all step containers within the Task, so that the steps inherit settings on the base container.
`steps`	`array`	Steps are the steps of the build; each step is run sequentially with the source mounted into /workspace.
`volumes`		Volumes is a collection of volumes that are available to mount into the steps of the build. See Pod.spec.volumes (API version: v1)
`workspaces`	`array`	Workspaces are the volumes that this Task requires.

.status.taskSpec.params

Description

Params is a list of input parameters required to run the task. Params must be supplied as inputs in TaskRuns unless they declare a default value.

Type

`array`

.status.taskSpec.params[]

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources). Parameter values are provided by users as inputs on a TaskRun or PipelineRun.

Type

`object`

Required

`name`

Property	Type	Description
`default`		Default is the value a parameter takes if no input value is supplied. If default is set, a Task may be executed without a supplied value for the parameter.
`description`	`string`	Description is a user-facing description of the parameter that may be used to populate a UI.
`enum`	`array`	Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.

Property	Type	Description
`name`	`string`	Name declares the name by which a parameter is referenced.
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs parameter.
`type`	`string`	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

.status.taskSpec.params[].enum

Description

Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.

Type

`array`

.status.taskSpec.params[].enum[]

Type

`string`

.status.taskSpec.params[].properties

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

.status.taskSpec.results

Description

Results are values that this Task can output

Type

`array`

.status.taskSpec.results[]

Description

TaskResult used to describe the results of a task

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs results.

Property	Type	Description
`type`	`string`	Type is the user-specified type of the result. The possible type is currently "string" and will support "array" in following work.
`value`		Value the expression used to retrieve the value of the result from an underlying Step.

.status.taskSpec.results[].properties

Description

Properties is the JSON Schema properties to support key-value pairs results.

Type

`object`

.status.taskSpec.sidecars

Description

Sidecars are run alongside the Task's step containers. They begin before the steps start and end after the steps complete.

Type

`array`

.status.taskSpec.sidecars[]

Description

Sidecar has nearly the same data structure as Step but does not have the ability to timeout.

Type

`object`

Required

`name`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is not provided. Variable references $(VAR_NAME) using the Sidecar's environment. If a variable can resolved, the reference in the input string will be Double $$ are reduced to a single $, which allow the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" string literal "$(VAR_NAME)". Escaped reference expanded, regardless of whether the variable ex Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-appli command-argument-container/#running-a-comm
`command`	`array`	Entrypoint array. Not executed within a shell. The ENTRYPOINT is used if this is not provided. Vari $(VAR_NAME) are expanded using the Sidecar's a variable cannot be resolved, the reference in th will be unchanged. Double $$ are reduced to a s allows for escaping the $(VAR_NAME) syntax: i. "$$(VAR_NAME)" will produce the string literal "$ Escaped references will never be expanded, reg whether the variable exists or not. Cannot be upd https://kubernetes.io/docs/tasks/inject-data-appli command-argument-container/#running-a-comm

Property	Type	Description
`computeResources`	`object`	ComputeResources required by this Sidecar. Ca updated. More info: https://kubernetes.io/docs/concepts/configuratior resources-containers/ ↗
`env`	`array`	List of environment variables to set in the Sideca updated.
`envFrom`	`array`	List of sources to populate environment variables The keys defined within a source must be a C_ID invalid keys will be reported as an event when th starting. When a key exists in multiple sources, t associated with the last source will take precede defined by an Env with a duplicate key will take p Cannot be updated.
`image`	`string`	Image reference name. More info: https://kubernetes.io/docs/concepts/containers/ir
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPre to Always if :latest tag is specified, or IfNotPresei Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/ir images ↗

Property	Type	Description
`lifecycle`	`object`	Actions that the management system should tak Sidecar lifecycle events. Cannot be updated.
`livenessProbe`	`object`	Periodic probe of Sidecar liveness. Container wil the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗
`name`	`string`	Name of the Sidecar specified as a DNS_LABEL in a Task must have a unique name (DNS_LABE updated.
`ports`	`array`	List of ports to expose from the Sidecar. Exposin gives the system additional information about the connections a container uses, but is primarily inf specifying a port here DOES NOT prevent that p exposed. Any port which is listening on the defau address inside a container will be accessible fror Cannot be updated.
`readinessProbe`	`object`	Periodic probe of Sidecar service readiness. Cor removed from service endpoints if the probe fails updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗

Property	Type	Description
`restartPolicy`	`string`	RestartPolicy refers to kubernetes RestartPolicy. set for an initContainer and must have it's policy It is currently left optional to help support Kubern prior to 1.29 when this feature was introduced.
`script`	`string`	Script is the contents of an executable file to exe If Script is not empty, the Step cannot have an C Args.
`securityContext`	`object`	SecurityContext defines the security options the be run with. If set, the fields of SecurityContext o equivalent fields of PodSecurityContext. More inf https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`startupProbe`	`object`	StartupProbe indicates that the Pod the Sidecar successfully initialized. If specified, no other prob until this completes successfully. If this probe fail be restarted, just as if the livenessProbe failed. T to provide different probe parameters at the begi lifecycle, when it might take a long time to load d cache, than during steady-state operation. This c updated. More info: https://kubernetes.io/docs/concepts/workloads/p lifecycle#container-probes ↗
`stdin`	`boolean`	Whether this Sidecar should allocate a buffer for container runtime. If this is not set, reads from st

Property	Type	Description
		Sidecar will always result in EOF. Default is false
`stdinOnce`	`boolean`	Whether the container runtime should close the s after it has been opened by a single attach. Whe the stdin stream will remain open across multiple sessions. If stdinOnce is set to true, stdin is oper start, is empty until the first client attaches to stdi remains open and accepts data until the client di which time stdin is closed and remains closed ur restarted. If this flag is false, a container process from stdin will never receive an EOF. Default is fa
`terminationMessagePath`	`string`	Optional: Path at which the file to which the Side message will be written is mounted into the Side Message written is intended to be brief final statu assertion failure message. Will be truncated by tl greater than 4096 bytes. The total message leng containers will be limited to 12kb. Defaults to /de log. Cannot be updated.
`terminationMessagePolicy`	`string`	Indicate how the termination message should be will use the contents of terminationMessagePath Sidecar status message on both success and fai FallbackToLogsOnError will use the last chunk o output if the termination message file is empty ar exited with an error. The log output is limited to 2 lines, whichever is smaller. Defaults to File. Cann

Property	Type	Description
`tty`	`boolean`	Whether this Sidecar should allocate a TTY for it requires 'stdin' to be true. Default is false.
`volumeDevices`	`array`	volumeDevices is the list of block devices to be u Sidecar.
`volumeMounts`	`array`	Volumes to mount into the Sidecar's filesystem. C updated.
`workingDir`	`string`	Sidecar's working directory. If not specified, the c runtime's default will be used, which might be co container image. Cannot be updated.
`workspaces`	`array`	This is an alpha field. You must set the "enable-a flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Tasl Sidecar wants exclusive access to. Adding a wor list means that any other Step or Sidecar that do request this Workspace will not have access to it

.status.taskSpec.sidecars[].args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Sidecar's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are

reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.sidecars[].args[]

Type

`string`

.status.taskSpec.sidecars[].command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Sidecar's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.sidecars[].command[]

Type

`string`

.status.taskSpec.sidecars[].computeResources

Description

ComputeResources required by this Sidecar. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.status.taskSpec.sidecars[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.status.taskSpec.sidecars[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.status.taskSpec.sidecars[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.sidecars[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.sidecars[].env

Description

List of environment variables to set in the Sidecar. Cannot be updated.

Type

`array`

.status.taskSpec.sidecars[].env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.status.taskSpec.sidecars[].env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.

Property	Type	Description
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.status.taskSpec.sidecars[].env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.status.taskSpec.sidecars[].env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.status.taskSpec.sidecars[].env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers.

Property	Type	Description
		If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.status.taskSpec.sidecars[].env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars

Property	Type	Description
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.status.taskSpec.sidecars[].env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

Property	Type	Description
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.status.taskSpec.sidecars[].envFrom

Description

List of sources to populate environment variables in the Sidecar. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the container is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.status.taskSpec.sidecars[].envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.

Property	Type	Description
`secretRef`	`object`	The Secret to select from

.status.taskSpec.sidecars[].envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.status.taskSpec.sidecars[].envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.status.taskSpec.sidecars[].lifecycle

Description

Actions that the management system should take in response to Sidecar lifecycle events. Cannot be updated.

Type

`object`

Property	Type	Description
`postStart`	`object`	PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks ↗
`preStop`	`object`	PreStop is called immediately before a container is terminated due to an API request or management event

Property	Type	Description
		such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks ↗
`stopSignal`	`string`	StopSignal defines which signal will be sent to a container when it is being stopped. If not specified, the default is defined by the container runtime in use. StopSignal can only be set for Pods with a non-empty .spec.os.name

.status.taskSpec.sidecars[].lifecycle.postStart

Description

PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in the container.
`httpGet`	`object`	HTTPGet specifies an HTTP GET request to perform.
`sleep`	`object`	Sleep represents a duration that the container should sleep.
`tcpSocket`	`object`	Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

.status.taskSpec.sidecars[].lifecycle.postStart.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to

Property	Type	Description
		explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.status.taskSpec.sidecars[].lifecycle.postStart.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.status.taskSpec.sidecars[].lifecycle.postStart.exec.command[]

Type

`string`

.status.taskSpec.sidecars[].lifecycle.postStart.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.status.taskSpec.sidecars[].lifecycle.postStart.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.status.taskSpec.sidecars[].lifecycle.postStart.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.status.taskSpec.sidecars[].lifecycle.postStart.sleep

Description

Sleep represents a duration that the container should sleep.

Type

`object`

Required

`seconds`

Property	Type	Description
`seconds`	`integer`	Seconds is the number of seconds to sleep.

.status.taskSpec.sidecars[].lifecycle.postStart.tcpSocket

Description

Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.status.taskSpec.sidecars[].lifecycle.preStop

Description

PreStop is called immediately before a container is terminated due to an API request or management event such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in the container.
`httpGet`	`object`	HTTPGet specifies an HTTP GET request to perform.
`sleep`	`object`	Sleep represents a duration that the container should sleep.
`tcpSocket`	`object`	Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

.status.taskSpec.sidecars[].lifecycle.preStop.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell

Property	Type	Description
		instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.status.taskSpec.sidecars[].lifecycle.preStop.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.status.taskSpec.sidecars[].lifecycle.preStop.exec.command[]

Type

`string`

.status.taskSpec.sidecars[].lifecycle.preStop.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.status.taskSpec.sidecars[].lifecycle.preStop.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.status.taskSpec.sidecars[].lifecycle.preStop.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.status.taskSpec.sidecars[].lifecycle.preStop.sleep

Description

Sleep represents a duration that the container should sleep.

Type

`object`

Required

`seconds`

Property	Type	Description
`seconds`	`integer`	Seconds is the number of seconds to sleep.

.status.taskSpec.sidecars[].lifecycle.preStop.tcpSocket

Description

Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for backward compatibility. There is no validation of this field and lifecycle hooks will fail at runtime when it is specified.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.status.taskSpec.sidecars[].livenessProbe

Description

Periodic probe of Sidecar liveness. Container will be restarted if the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq
`httpGet`	`object`	HTTPGet specifies an HTTP GET request
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu

Property	Type	Description
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru pod are sent a termination signal and the t processes are forcibly halted with a kill sig value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t (no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceP gate. Minimum value is 1. spec.terminationGracePeriodSeconds is u
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.status.taskSpec.sidecars[].livenessProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.status.taskSpec.sidecars[].livenessProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.status.taskSpec.sidecars[].livenessProbe.exec.command[]

Type

`string`

.status.taskSpec.sidecars[].livenessProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md↗). If this is not specified, the default behavior is defined by gRPC.

.status.taskSpec.sidecars[].livenessProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.

Property	Type	Description
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.status.taskSpec.sidecars[].livenessProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.status.taskSpec.sidecars[].livenessProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.status.taskSpec.sidecars[].livenessProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.status.taskSpec.sidecars[].ports

Description

List of ports to expose from the Sidecar. Exposing a port here gives the system additional information about the network connections a container uses, but is primarily informational. Not specifying a port here DOES NOT prevent that port from being exposed. Any port which is listening on the default "0.0.0.0" address inside a container will be accessible from the network. Cannot be updated.

Type

`array`

.status.taskSpec.sidecars[].ports[]

Description

ContainerPort represents a network port in a single container.

Type

`object`

Required

`containerPort`

Property	Type	Description
`containerPort`	`integer`	Number of port to expose on the pod's IP address. This must be a valid port number, 0 < x < 65536.
`hostIP`	`string`	What host IP to bind the external port to.

Property	Type	Description
`hostPort`	`integer`	Number of port to expose on the host. If specified, this must be a valid port number, 0 < x < 65536. If HostNetwork is specified, this must match ContainerPort. Most containers do not need this.
`name`	`string`	If specified, this must be an IANA_SVC_NAME and unique within the pod. Each named port in a pod must have a unique name. Name for the port that can be referred to by services.
`protocol`	`string`	Protocol for port. Must be UDP, TCP, or SCTP. Defaults to "TCP".

.status.taskSpec.sidecars[].readinessProbe

Description

Periodic probe of Sidecar service readiness. Container will be removed from service endpoints if the probe fails. Cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th

Property	Type	Description
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq
`httpGet`	`object`	HTTPGet specifies an HTTP GET request
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru

Property	Type	Description
		pod are sent a termination signal and the ti processes are forcibly halted with a kill sigı value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t (no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceF gate. Minimum value is 1. spec.terminationGracePeriodSeconds is u
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.status.taskSpec.sidecars[].readinessProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell

Property	Type	Description
		instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.status.taskSpec.sidecars[].readinessProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.status.taskSpec.sidecars[].readinessProbe.exec.command[]

Type

`string`

.status.taskSpec.sidecars[].readinessProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md ↗). If this is not specified, the default behavior is defined by gRPC.

.status.taskSpec.sidecars[].readinessProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.

Property	Type	Description
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.status.taskSpec.sidecars[].readinessProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.status.taskSpec.sidecars[].readinessProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.status.taskSpec.sidecars[].readinessProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.status.taskSpec.sidecars[].securityContext

Description

SecurityContext defines the security options the Sidecar should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container

Property	Type	Description
		runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes

Property	Type	Description
		precedence. Note that this field cannot be set when spec.os.name is windows.
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes

Property	Type	Description
		precedence. Note that this field cannot be set when spec.os.name is windows.
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.status.taskSpec.sidecars[].securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.status.taskSpec.sidecars[].securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.status.taskSpec.sidecars[].securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.status.taskSpec.sidecars[].securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.sidecars[].securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.status.taskSpec.sidecars[].securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.sidecars[].securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.status.taskSpec.sidecars[].securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.status.taskSpec.sidecars[].securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the

value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa ↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the

Property	Type	Description
		value specified in SecurityContext takes precedence.

.status.taskSpec.sidecars[].startupProbe

Description

StartupProbe indicates that the Pod the Sidecar is running in has successfully initialized. If specified, no other probes are executed until this completes successfully. If this probe fails, the Pod will be restarted, just as if the livenessProbe failed. This can be used to provide different probe parameters at the beginning of a Pod's lifecycle, when it might take a long time to load data or warm a cache, than during steady-state operation. This cannot be updated. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes

Type

`object`

Property	Type	Description
`exec`	`object`	Exec specifies a command to execute in th
`failureThreshold`	`integer`	Minimum consecutive failures for the probe considered failed after having succeeded. Minimum value is 1.
`grpc`	`object`	GRPC specifies a GRPC HealthCheckReq
`httpGet`	`object`	HTTPGet specifies an HTTP GET request

Property	Type	Description
`initialDelaySeconds`	`integer`	Number of seconds after the container has before liveness probes are initiated. More i https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗
`periodSeconds`	`integer`	How often (in seconds) to perform the prob 10 seconds. Minimum value is 1.
`successThreshold`	`integer`	Minimum consecutive successes for the pr considered successful after having failed. I Must be 1 for liveness and startup. Minimu
`tcpSocket`	`object`	TCPSocket specifies a connection to a TC

Property	Type	Description
`terminationGracePeriodSeconds`	`integer`	Optional duration in seconds the pod need gracefully upon probe failure. The grace pe duration in seconds after the processes ru pod are sent a termination signal and the t processes are forcibly halted with a kill sig value longer than the expected cleanup tim process. If this value is nil, the pod's terminationGracePeriodSeconds will be us Otherwise, this value overrides the value p pod spec. Value must be non-negative inte value zero indicates stop immediately via t (no opportunity to shut down). This is a bet requires enabling ProbeTerminationGraceP gate. Minimum value is 1. spec.terminationGracePeriodSeconds is u
`timeoutSeconds`	`integer`	Number of seconds after which the probe t Defaults to 1 second. Minimum value is 1. https://kubernetes.io/docs/concepts/worklo lifecycle#container-probes ↗

.status.taskSpec.sidecars[].startupProbe.exec

Description

Exec specifies a command to execute in the container.

Type

`object`

Property	Type	Description
`command`	`array`	Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

.status.taskSpec.sidecars[].startupProbe.exec.command

Description

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

Type

`array`

.status.taskSpec.sidecars[].startupProbe.exec.command[]

Type

`string`

.status.taskSpec.sidecars[].startupProbe.grpc

Description

GRPC specifies a GRPC HealthCheckRequest.

Type

`object`

Required

`port`

Property	Type	Description
`port`	`integer`	Port number of the gRPC service. Number must be in the range 1 to 65535.
`service`	`string`	Service is the name of the service to place in the gRPC HealthCheckRequest (see https://github.com/grpc/grpc/blob/master/doc/health-checking.md ↗). If this is not specified, the default behavior is defined by gRPC.

.status.taskSpec.sidecars[].startupProbe.httpGet

Description

HTTPGet specifies an HTTP GET request to perform.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.

Property	Type	Description
`httpHeaders`	`array`	Custom headers to set in the request. HTTP allows repeated headers.
`path`	`string`	Path to access on the HTTP server.
`port`		Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.
`scheme`	`string`	Scheme to use for connecting to the host. Defaults to HTTP.

.status.taskSpec.sidecars[].startupProbe.httpGet.httpHeaders

Description

Custom headers to set in the request. HTTP allows repeated headers.

Type

`array`

.status.taskSpec.sidecars[].startupProbe.httpGet.httpHeaders[]

Description

HTTPHeader describes a custom header to be used in HTTP probes

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.
`value`	`string`	The header field value

.status.taskSpec.sidecars[].startupProbe.tcpSocket

Description

TCPSocket specifies a connection to a TCP port.

Type

`object`

Required

`port`

Property	Type	Description
`host`	`string`	Optional: Host name to connect to, defaults to the pod IP.
`port`		Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

.status.taskSpec.sidecars[].volumeDevices

Description

volumeDevices is the list of block devices to be used by the Sidecar.

Type

`array`

.status.taskSpec.sidecars[].volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.status.taskSpec.sidecars[].volumeMounts

Description

Volumes to mount into the Sidecar's filesystem. Cannot be updated.

Type

`array`

.status.taskSpec.sidecars[].volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively.

Property	Type	Description
		If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.status.taskSpec.sidecars[].workspaces

Description

This is an alpha field. You must set the "enable-api-fields" feature flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this Sidecar wants exclusive access to. Adding a workspace to this list means that any other Step or Sidecar that does not also request this Workspace will not have access to it.

Type

`array`

.status.taskSpec.sidecars[].workspaces[]

Description

WorkspaceUsage is used by a Step or Sidecar to declare that it wants isolated access to a Workspace defined in a Task.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	MountPath is the path that the workspace should be mounted to inside the Step or Sidecar, overriding any MountPath specified in the Task's WorkspaceDeclaration.
`name`	`string`	Name is the name of the workspace this Step or Sidecar wants access to.

.status.taskSpec.stepTemplate

Description

StepTemplate can be used as the basis for all step containers within the Task, so that the steps inherit settings on the base container.

Type

`object`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is used if t not provided. Variable references $(VAR_NAME) are expa using the Step's environment. If a variable cannot be resol the reference in the input string will be unchanged. Double are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce string literal "$(VAR_NAME)". Escaped references will nev expanded, regardless of whether the variable exists or not Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`command`	`array`	Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable refer $(VAR_NAME) are expanded using the Step's environmer variable cannot be resolved, the reference in the input stri be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NA Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. Mo https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-

Property	Type	Description
`computeResources`	`object`	ComputeResources required by this Step. Cannot be upda More info: https://kubernetes.io/docs/concepts/configuration/manage resources-containers/ ↗
`env`	`array`	List of environment variables to set in the Step. Cannot be updated.
`envFrom`	`array`	List of sources to populate environment variables in the St The keys defined within a source must be a C_IDENTIFIE invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Valu defined by an Env with a duplicate key will take precedenc Cannot be updated.
`image`	`string`	Image reference name. More info: https://kubernetes.io/docs/concepts/containers/images ↗
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPresent. Def to Always if :latest tag is specified, or IfNotPresent otherwi Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/images#up images ↗
`securityContext`	`object`	SecurityContext defines the security options the Step shou run with. If set, the fields of SecurityContext override the

Property	Type	Description
		equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`volumeDevices`	`array`	volumeDevices is the list of block devices to be used by th Step.
`volumeMounts`	`array`	Volumes to mount into the Step's filesystem. Cannot be up
`workingDir`	`string`	Step's working directory. If not specified, the container run default will be used, which might be configured in the cont image. Cannot be updated.

.status.taskSpec.stepTemplate.args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Step's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.stepTemplate.args[]

Type

`string`

.status.taskSpec.stepTemplate.command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the Step's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.stepTemplate.command[]

Type

`string`

.status.taskSpec.stepTemplate.computeResources

Description

ComputeResources required by this Step. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.status.taskSpec.stepTemplate.computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.status.taskSpec.stepTemplate.computeResources.claims []

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.status.taskSpec.stepTemplate.computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.stepTemplate.computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.stepTemplate.env

Description

List of environment variables to set in the Step. Cannot be updated.

Type

`array`

.status.taskSpec.stepTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.

Property	Type	Description
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.status.taskSpec.stepTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`,

Property	Type	Description
		`metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.status.taskSpec.stepTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.status.taskSpec.stepTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".

Property	Type	Description
`fieldPath`	`string`	Path of the field to select in the specified API version.

.status.taskSpec.stepTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.

Property	Type	Description
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.status.taskSpec.stepTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"

Property	Type	Description
`resource`	`string`	Required: resource to select

.status.taskSpec.stepTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.status.taskSpec.stepTemplate.envFrom

Description

List of sources to populate environment variables in the Step. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.status.taskSpec.stepTemplate.envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.
`secretRef`	`object`	The Secret to select from

.status.taskSpec.stepTemplate.envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.status.taskSpec.stepTemplate.envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.status.taskSpec.stepTemplate.securityContext

Description

SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name

Property	Type	Description
		is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and

Property	Type	Description
		PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.status.taskSpec.stepTemplate.securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.status.taskSpec.stepTemplate.securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.status.taskSpec.stepTemplate.securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.status.taskSpec.stepTemplate.securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.stepTemplate.securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.status.taskSpec.stepTemplate.securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.stepTemplate.securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.status.taskSpec.stepTemplate.securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured

Property	Type	Description
		seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.status.taskSpec.stepTemplate.securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.status.taskSpec.stepTemplate.volumeDevices

Description

volumeDevices is the list of block devices to be used by the Step.

Type

`array`

.status.taskSpec.stepTemplate.volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.status.taskSpec.stepTemplate.volumeMounts

Description

Volumes to mount into the Step's filesystem. Cannot be updated.

Type

`array`

.status.taskSpec.stepTemplate.volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively.

Property	Type	Description
		If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.status.taskSpec.steps

Description

Steps are the steps of the build; each step is run sequentially with the source mounted into /workspace.

Type

`array`

.status.taskSpec.steps[]

Description

Step runs a subcomponent of a Task

Type

`object`

Required

`name`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is used if t not provided. Variable references $(VAR_NAME) are expa using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchange Double $$ are reduced to a single $, which allows for esca the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will prod string literal "$(VAR_NAME)". Escaped references will nev expanded, regardless of whether the variable exists or not Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-

Property	Type	Description
`command`	`array`	Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable refer $(VAR_NAME) are expanded using the container's enviro If a variable cannot be resolved, the reference in the input will be unchanged. Double $$ are reduced to a single $, w allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NA Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. Mo https://kubernetes.io/docs/tasks/inject-data-application/def command-argument-container/#running-a-command-in-a-
`computeResources`	`object`	ComputeResources required by this Step. Cannot be upda More info: https://kubernetes.io/docs/concepts/configuration/manage resources-containers/ ↗
`displayName`	`string`	DisplayName is a user-facing name of the step that may b to populate a UI.
`env`	`array`	List of environment variables to set in the Step. Cannot be updated.
`envFrom`	`array`	List of sources to populate environment variables in the St The keys defined within a source must be a C_IDENTIFIE invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Valu

Property	Type	Description
		defined by an Env with a duplicate key will take precedenc Cannot be updated.
`image`	`string`	Docker image name. More info: https://kubernetes.io/docs/concepts/containers/images ↗
`imagePullPolicy`	`string`	Image pull policy. One of Always, Never, IfNotPresent. Def to Always if :latest tag is specified, or IfNotPresent otherwi Cannot be updated. More info: https://kubernetes.io/docs/concepts/containers/images#up images ↗
`name`	`string`	Name of the Step specified as a DNS_LABEL. Each Step Task must have a unique name.
`onError`	`string`	OnError defines the exiting behavior of a container on erro be set to [continue | stopAndFail]
`params`	`array`	Params declares parameters passed to this step action.
`ref`	`object`	Contains the reference to an existing StepAction.
`results`	`array`	Results declares StepResults produced by the Step. It can be used in an inlined Step when used to store Resu $(step.results.resultName.path). It cannot be used when

Property	Type	Description
		referencing StepActions using [v1.Step.Ref]. The Results declared by the StepActions will be stored here instead.
`script`	`string`	Script is the contents of an executable file to execute. If Script is not empty, the Step cannot have an Command the Args will be passed to the Script.
`securityContext`	`object`	SecurityContext defines the security options the Step shou run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗
`stderrConfig`	`object`	Stores configuration for the stderr stream of the step.
`stdoutConfig`	`object`	Stores configuration for the stdout stream of the step.
`timeout`	`string`	Timeout is the time after which the step times out. Defaults never. Refer to Go's ParseDuration documentation for exp format: https://golang.org/pkg/time/#ParseDuration ↗
`volumeDevices`	`array`	volumeDevices is the list of block devices to be used by th Step.

Property	Type	Description
`volumeMounts`	`array`	Volumes to mount into the Step's filesystem. Cannot be up
`when`	`array`	When is a list of when expressions that need to be true for task to run
`workingDir`	`string`	Step's working directory. If not specified, the container runt default will be used, which might be configured in the cont image. Cannot be updated.
`workspaces`	`array`	This is an alpha field. You must set the "enable-api-fields" flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this wants exclusive access to. Adding a workspace to this list that any other Step or Sidecar that does not also request t Workspace will not have access to it.

.status.taskSpec.steps[].args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.steps[].args[]

Type

`string`

.status.taskSpec.steps[].command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.status.taskSpec.steps[].command[]

Type

`string`

.status.taskSpec.steps[].computeResources

Description

ComputeResources required by this Step. Cannot be updated. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

Property	Type	Description
`claims`	`array`	Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.
`limits`	`object`	Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗
`requests`	`object`	Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ ↗

.status.taskSpec.steps[].computeResources.claims

Description

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container. This field depends on the DynamicResourceAllocation feature gate. This field is immutable. It can only be set for containers.

Type

`array`

.status.taskSpec.steps[].computeResources.claims[]

Description

ResourceClaim references one entry in PodSpec.ResourceClaims.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.
`request`	`string`	Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is made available, otherwise only the result of this request.

.status.taskSpec.steps[].computeResources.limits

Description

Limits describes the maximum amount of compute resources allowed. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.steps[].computeResources.requests

Description

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

Type

`object`

.status.taskSpec.steps[].env

Description

List of environment variables to set in the Step. Cannot be updated.

Type

`array`

.status.taskSpec.steps[].env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the

Property	Type	Description
		container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.status.taskSpec.steps[].env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.

Property	Type	Description
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.status.taskSpec.steps[].env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.status.taskSpec.steps[].env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.status.taskSpec.steps[].env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers.

Property	Type	Description
		If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.status.taskSpec.steps[].env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars

Property	Type	Description
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.status.taskSpec.steps[].env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

Property	Type	Description
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.status.taskSpec.steps[].envFrom

Description

List of sources to populate environment variables in the Step. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the Step is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

Type

`array`

.status.taskSpec.steps[].envFrom[]

Description

EnvFromSource represents the source of a set of ConfigMaps or Secrets

Type

`object`

Property	Type	Description
`configMapRef`	`object`	The ConfigMap to select from
`prefix`	`string`	Optional text to prepend to the name of each environment variable. May consist of any printable ASCII characters except '='.

Property	Type	Description
`secretRef`	`object`	The Secret to select from

.status.taskSpec.steps[].envFrom[].configMapRef

Description

The ConfigMap to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap must be defined

.status.taskSpec.steps[].envFrom[].secretRef

Description

The Secret to select from

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret must be defined

.status.taskSpec.steps[].params

Description

Params declares parameters passed to this step action.

Type

`array`

.status.taskSpec.steps[].params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	

Property	Type	Description
`value`		

.status.taskSpec.steps[].ref

Description

Contains the reference to an existing StepAction.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referenced step
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.status.taskSpec.steps[].ref.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.status.taskSpec.steps[].ref.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.status.taskSpec.steps[].results

Description

Results declares StepResults produced by the Step. It can be used in an inlined Step when used to store Results to $(step.results.resultName.path). It cannot be used when referencing StepActions using [v1.Step.Ref]. The Results declared by the StepActions will be stored here instead.

Type

`array`

.status.taskSpec.steps[].results[]

Description

StepResult used to describe the Results of a Step.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs results.
`type`	`string`	The possible types are 'string', 'array', and 'object', with 'string' as the default.

.status.taskSpec.steps[].results[].properties

Description

Properties is the JSON Schema properties to support key-value pairs results.

Type

`object`

.status.taskSpec.steps[].securityContext

Description

SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host.

Property	Type	Description
		Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to

Property	Type	Description
		start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options.

Property	Type	Description
		Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.status.taskSpec.steps[].securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.status.taskSpec.steps[].securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities
`drop`	`array`	Removed capabilities

.status.taskSpec.steps[].securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.status.taskSpec.steps[].securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.steps[].securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.status.taskSpec.steps[].securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.status.taskSpec.steps[].securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.status.taskSpec.steps[].securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.status.taskSpec.steps[].securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-

Property	Type	Description
		gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.status.taskSpec.steps[].stderrConfig

Description

Stores configuration for the stderr stream of the step.

Type

`object`

Property	Type	Description
`path`	`string`	Path to duplicate stdout stream to on container's local filesystem.

.status.taskSpec.steps[].stdoutConfig

Description

Stores configuration for the stdout stream of the step.

Type

`object`

Property	Type	Description
`path`	`string`	Path to duplicate stdout stream to on container's local filesystem.

.status.taskSpec.steps[].volumeDevices

Description

volumeDevices is the list of block devices to be used by the Step.

Type

`array`

.status.taskSpec.steps[].volumeDevices[]

Description

volumeDevice describes a mapping of a raw block device within a container.

Type

`object`

Required

`devicePath` `name`

Property	Type	Description
`devicePath`	`string`	devicePath is the path inside of the container that the device will be mapped to.
`name`	`string`	name must match the name of a persistentVolumeClaim in the pod

.status.taskSpec.steps[].volumeMounts

Description

Volumes to mount into the Step's filesystem. Cannot be updated.

Type

`array`

.status.taskSpec.steps[].volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively. If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the

Property	Type	Description
		pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

.status.taskSpec.steps[].when

Description

When is a list of when expressions that need to be true for the task to run

Type

`array`

.status.taskSpec.steps[].when[]

Description

WhenExpression allows a PipelineTask to declare expressions to be evaluated before the Task is run to determine whether the Task should be executed or skipped

Type

`object`

Property	Type	Description
`cel`	`string`	CEL is a string of Common Language Expression, which can be used to conditionally execute the task based on the result of the expression evaluation More info about CEL syntax: https://github.com/google/cel-spec/blob/master/doc/langdef.md ↗
`input`	`string`	Input is the string for guard checking which can be a static input or an output from a parent Task
`operator`	`string`	Operator that represents an Input's relationship to the values
`values`	`array`	Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

.status.taskSpec.steps[].when[].values

Description

Values is an array of strings, which is compared against the input, for guard checking It must be non-empty

Type

`array`

.status.taskSpec.steps[].when[].values[]

Type

`string`

.status.taskSpec.steps[].workspaces

Description

This is an alpha field. You must set the "enable-api-fields" feature flag to "alpha" for this field to be supported. Workspaces is a list of workspaces from the Task that this Step wants exclusive access to. Adding a workspace to this list means that any other Step or Sidecar that does not also request this Workspace will not have access to it.

Type

`array`

.status.taskSpec.steps[].workspaces[]

Description

WorkspaceUsage is used by a Step or Sidecar to declare that it wants isolated access to a Workspace defined in a Task.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	MountPath is the path that the workspace should be mounted to inside the Step or Sidecar, overriding any MountPath specified in the Task's WorkspaceDeclaration.
`name`	`string`	Name is the name of the workspace this Step or Sidecar wants access to.

.status.taskSpec.workspaces

Description

Workspaces are the volumes that this Task requires.

Type

`array`

.status.taskSpec.workspaces[]

Description

WorkspaceDeclaration is a declaration of a volume that a Task requires.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is an optional human readable description of this volume.

Property	Type	Description
`mountPath`	`string`	MountPath overrides the directory that the volume will be made available at.
`name`	`string`	Name is the name by which you can bind the volume at runtime.
`optional`	`boolean`	Optional marks a Workspace as not being required in TaskRuns. By default this field is false and so declared workspaces are required.
`readOnly`	`boolean`	ReadOnly dictates whether a mounted volume is writable. By default this field is false and so mounted volumes are writable.

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1/namespaces/{namespace}/taskruns`
  - `DELETE`: delete collection of TaskRun
  - `GET`: list objects of kind TaskRun
  - `POST`: create a new TaskRun
- `/apis/tekton.dev/v1/namespaces/{namespace}/taskruns/{name}`
  - `DELETE`: delete the specified TaskRun
  - `GET`: read the specified TaskRun

- `PATCH`: partially update the specified TaskRun
- `PUT`: replace the specified TaskRun

- `/apis/tekton.dev/v1/namespaces/{namespace}/taskruns/{name}/status`
  - `GET`: read status of the specified TaskRun
  - `PATCH`: partially update status of the specified TaskRun
  - `PUT`: replace status of the specified TaskRun

/apis/tekton.dev/v1/namespaces/{namespace}/taskruns

HTTP method

`DELETE`

Description

delete collection of TaskRun

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind TaskRun

HTTP responses

HTTP code	Response body
200 - OK	`TaskRunList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`TaskRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
201 - Created	`TaskRun` schema
202 - Accepted	`TaskRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/taskruns/{name}

HTTP method

`DELETE`

Description

delete the specified TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified TaskRun

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a

Parameter	Type	Description
		warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently

Parameter	Type	Description
		dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`TaskRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
201 - Created	`TaskRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1/namespaces/{namespace}/taskruns/{name}/status

HTTP method

`GET`

Description

read status of the specified TaskRun

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server

Parameter	Type	Description
		will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified TaskRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the

Parameter	Type	Description
		request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`TaskRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`TaskRun` schema
201 - Created	`TaskRun` schema
401 - Unauthorized	Empty

ClusterTask [tekton.dev/v1]

Run [tekton.dev/v1]

Description

Run represents a single execution of a Custom Task.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

Property	Type	Description
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta ↗`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	RunSpec defines the desired state of Run
`status`	`object`	RunStatus defines the observed state of Run

.spec

Description

RunSpec defines the desired state of Run

Type

`object`

Property	Type	Description
`params`	`array`	Params is a list of Param

Property	Type	Description
`podTemplate`	`object`	PodTemplate holds pod specific configuration
`ref`	`object`	TaskRef can be used to refer to a specific instance of a task.
`retries`	`integer`	Used for propagating retries count to custom tasks
`serviceAccountName`	`string`	
`spec`	`object`	Spec is a specification of a custom task
`status`	`string`	Used for cancelling a run (and maybe more later on)
`statusMessage`	`string`	Status message for cancellation.
`timeout`	`string`	Time after which the custom-task times out. Refer Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗
`workspaces`	`array`	Workspaces is a list of WorkspaceBindings from volumes to workspaces.

.spec.params

Description

Params is a list of Param

Type

`array`

.spec.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.podTemplate

Description

PodTemplate holds pod specific configuration

Type

`object`

Property	Type	Description
`affinity`		If specified, the pod's scheduling constraints Pod.spec.affinity (API version: v1)
`automountServiceAccountToken`	`boolean`	AutomountServiceAccountToken indicates w running as this service account should have token automatically mounted.
`dnsConfig`	`object`	Specifies the DNS parameters of a pod. Par specified here will be merged to the generat configuration based on DNSPolicy.
`dnsPolicy`	`string`	Set DNS policy for the pod. Defaults to "Clu Valid values are 'ClusterFirst', 'Default' or 'N parameters given in DNSConfig will be merg policy selected with DNSPolicy.
`enableServiceLinks`	`boolean`	EnableServiceLinks indicates whether inforr services should be injected into pod's envirc variables, matching the syntax of Docker lin Defaults to true.
`env`	`array`	List of environment variables that can be pro containers belonging to the pod.
`hostAliases`	`array`	HostAliases is an optional list of hosts and I be injected into the pod's hosts file if specifie

Property	Type	Description
		only valid for non-hostNetwork pods.
`hostNetwork`	`boolean`	HostNetwork specifies whether the pod may node network namespace
`hostUsers`	`boolean`	HostUsers indicates whether the pod will us user namespace. Optional: Default to true. I or not present, the pod will be run in the hos namespace, useful for when the pod needs only available to the host user namespace, loading a kernel module with CAP_SYS_MO When set to false, a new user namespace is the pod. Setting false is useful to mitigating breakout vulnerabilities such as allowing co run as root without their user having root pri the host. This field depends on the kubernet gate UserNamespacesSupport being enable
`imagePullSecrets`	`array`	ImagePullSecrets gives the name of the sec the pod to pull the image if specified
`nodeSelector`	`object`	NodeSelector is a selector which must be tr pod to fit on a node. Selector which must ma labels for the pod to be scheduled on that no info: https://kubernetes.io/docs/concepts/configu pod-node/↗

Property	Type	Description
`priorityClassName`	`string`	If specified, indicates the pod's priority. "syst critical" and "system-cluster-critical" are two keywords which indicate the highest prioritie former being the highest priority. Any other r be defined by creating a PriorityClass objec name. If not specified, the pod priority will b zero if there is no default.
`runtimeClassName`	`string`	RuntimeClassName refers to a RuntimeClas the node.k8s.io group, which should be use pod. If no RuntimeClass resource matches t class, the pod will not be run. If unset or em "legacy" RuntimeClass will be used, which i class with an empty definition that uses the runtime handler. More info: https://git.k8s.io/enhancements/keps/sig-no class.md↗ This is a beta feature as of Kube v1.14.
`schedulerName`	`string`	SchedulerName specifies the scheduler to b dispatch the Pod
`securityContext`		SecurityContext holds pod-level security attr common container settings. Optional: Defau See type description for default values of ea Pod.spec.securityContext (API version: v1)

Property	Type	Description
`tolerations`	`array`	If specified, the pod's tolerations.
`topologySpreadConstraints`	`array`	TopologySpreadConstraints controls how Pc spread across your cluster among failure-dc as regions, zones, nodes, and other user-de topology domains.
`volumes`		List of volumes that can be mounted by con belonging to the pod. More info: https://kubernetes.io/docs/concepts/storage See Pod.spec.volumes (API version: v1)

.spec.podTemplate.dnsConfig

Description

Specifies the DNS parameters of a pod. Parameters specified here will be merged to the generated DNS configuration based on DNSPolicy.

Type

`object`

Property	Type	Description
`nameservers`	`array`	A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Property	Type	Description
`options`	`array`	A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.
`searches`	`array`	A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

.spec.podTemplate.dnsConfig.nameservers

Description

A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

Type

`array`

.spec.podTemplate.dnsConfig.nameservers[]

Type

`string`

.spec.podTemplate.dnsConfig.options

Description

A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.

Type

`array`

.spec.podTemplate.dnsConfig.options[]

Description

PodDNSConfigOption defines DNS resolver options of a pod.

Type

`object`

Property	Type	Description
`name`	`string`	Name is this DNS resolver option's name. Required.
`value`	`string`	Value is this DNS resolver option's value.

.spec.podTemplate.dnsConfig.searches

Description

A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

Type

`array`

.spec.podTemplate.dnsConfig.searches[]

Type

`string`

.spec.podTemplate.env

Description

List of environment variables that can be provided to the containers belonging to the pod.

Type

`array`

.spec.podTemplate.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".

Property	Type	Description
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.podTemplate.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu,

Property	Type	Description
		requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.podTemplate.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.podTemplate.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.podTemplate.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.podTemplate.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.podTemplate.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.podTemplate.hostAliases

Description

HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified. This is only valid for non-hostNetwork pods.

Type

`array`

.spec.podTemplate.hostAliases[]

Description

HostAlias holds the mapping between IP and hostnames that will be injected as an entry in the pod's hosts file.

Type

`object`

Required

`ip`

Property	Type	Description
`hostnames`	`array`	Hostnames for the above IP address.
`ip`	`string`	IP address of the host file entry.

.spec.podTemplate.hostAliases[].hostnames

Description

Hostnames for the above IP address.

Type

`array`

.spec.podTemplate.hostAliases[].hostnames[]

Type

`string`

.spec.podTemplate.imagePullSecrets

Description

ImagePullSecrets gives the name of the secret used by the pod to pull the image if specified

Type

`array`

.spec.podTemplate.imagePullSecrets[]

Description

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.podTemplate.nodeSelector

Description

NodeSelector is a selector which must be true for the pod to fit on a node. Selector which must match a node's labels for the pod to be scheduled on that node. More info: https://kubernetes.io/docs/concepts/configuration/assign-pod-node/

Type

`object`

.spec.podTemplate.tolerations

Description

If specified, the pod's tolerations.

Type

`array`

.spec.podTemplate.tolerations[]

Description

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator <operator>.

Type

`object`

Property	Type	Description
`effect`	`string`	Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.
`key`	`string`	Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.
`operator`	`string`	Operator represents a key's relationship to the value. Valid operators are Exists, Equal, Lt, and Gt. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category. Lt and Gt perform numeric comparisons (requires feature gate TaintTolerationComparisonOperators).
`tolerationSeconds`	`integer`	TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint

Property	Type	Description
		forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.
`value`	`string`	Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

.spec.podTemplate.topologySpreadConstraints

Description

TopologySpreadConstraints controls how Pods are spread across your cluster among failure-domains such as regions, zones, nodes, and other user-defined topology domains.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[]

Description

TopologySpreadConstraint specifies how to spread matching pods among the given topology.

Type

`object`

Required

`maxSkew` `topologyKey` `whenUnsatisfiable`

Property	Type	Description
`labelSelector`	`object`	LabelSelector is used to find matching pods. Pods that match this label selector are counted to

Property	Type	Description
		determine the number of pods in their corresponding topology domain.
`matchLabelKeys`	`array`	MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).
`maxSkew`	`integer`	MaxSkew describes the degree to which pods may be unevenly distributed. When `whenUnsatisfiable=DoNotSchedule`, it is the maximum permitted difference between the number of matching pods in the target topology and the global minimum. The global minimum is the minimum number of matching pods in an eligible domain or zero if the number of eligible domains is less than MinDomains. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 2/2/1: In this case, the global minimum is 1. | zone1 | zone2 | zone3 | | P P | P P | P |

Property	Type	Description
		• if MaxSkew is 1, incoming pod can only be scheduled to zone3 to become 2/2/2; scheduling it onto zone1(zone2) would make the ActualSkew(3-1) on zone1(zone2) violate MaxSkew(1). • if MaxSkew is 2, incoming pod can be scheduled onto any zone. When `whenUnsatisfiable=ScheduleAnyway`, it is used to give higher precedence to topologies that satisfy it. It's a required field. Default value is 1 and 0 is not allowed.
`minDomains`	`integer`	MinDomains indicates a minimum number of eligible domains. When the number of eligible domains with matching topology keys is less than minDomains, Pod Topology Spread treats "global minimum" as 0, and then the calculation of Skew is performed. And when the number of eligible domains with matching topology keys equals or greater than minDomains, this value has no effect on scheduling. As a result, when the number of eligible domains is less than minDomains, scheduler won't schedule more than maxSkew Pods to those domains. If value is nil, the constraint behaves as if MinDomains is equal to 1. Valid values are integers greater than 0. When value is not nil, WhenUnsatisfiable must be DoNotSchedule. For example, in a 3-zone cluster, MaxSkew is set to 2, MinDomains is set to 5 and pods with the same labelSelector spread as 2/2/2: | zone1 | zone2 | zone3 | | P P | P P | P P | The number of

Property	Type	Description
		domains is less than 5(MinDomains), so "global minimum" is treated as 0. In this situation, new pod with the same labelSelector cannot be scheduled, because computed skew will be 3(3 - 0) if new Pod is scheduled to any of the three zones, it will violate MaxSkew.
`nodeAffinityPolicy`	`string`	NodeAffinityPolicy indicates how we will treat Pod's nodeAffinity/nodeSelector when calculating pod topology spread skew. Options are: • Honor: only nodes matching nodeAffinity/nodeSelector are included in the calculations. • Ignore: nodeAffinity/nodeSelector are ignored. All nodes are included in the calculations. If this value is nil, the behavior is equivalent to the Honor policy.
`nodeTaintsPolicy`	`string`	NodeTaintsPolicy indicates how we will treat node taints when calculating pod topology spread skew. Options are: • Honor: nodes without taints, along with tainted nodes for which the incoming pod has a toleration, are included. • Ignore: node taints are ignored. All nodes are included. If this value is nil, the behavior is equivalent to the Ignore policy.

Property	Type	Description
`topologyKey`	`string`	TopologyKey is the key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology. We consider each <key, value> as a "bucket", and try to put balanced number of pods into each bucket. We define a domain as a particular instance of a topology. Also, we define an eligible domain as a domain whose nodes meet the requirements of nodeAffinityPolicy and nodeTaintsPolicy. e.g. If TopologyKey is "kubernetes.io/hostname", each Node is a domain of that topology. And, if TopologyKey is "topology.kubernetes.io/zone", each zone is a domain of that topology. It's a required field.
`whenUnsatisfiable`	`string`	WhenUnsatisfiable indicates how to deal with a pod if it doesn't satisfy the spread constraint. • DoNotSchedule (default) tells the scheduler not to schedule it. • ScheduleAnyway tells the scheduler to schedule the pod in any location, but giving higher precedence to topologies that would help reduce the skew. A constraint is considered "Unsatisfiable" for an incoming pod if and only if every possible node assignment for that pod would violate "MaxSkew" on some topology. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 3/1/1: | zone1 | zone2 | zone3 | | P P P | P | P | If WhenUnsatisfiable is set to DoNotSchedule, incoming pod can only be scheduled to zone2(zone3) to become

Property	Type	Description
		3/2/1(3/1/2) as ActualSkew(2-1) on zone2(zone3) satisfies MaxSkew(1). In other words, the cluster can still be imbalanced, but scheduler won't make it *more* imbalanced. It's a required field.

.spec.podTemplate.topologySpreadConstraints[].labelSelector

Description

LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchExpressions[].values[]

Type

`string`

.spec.podTemplate.topologySpreadConstraints[].labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.podTemplate.topologySpreadConstraints[].matchLabelKeys

Description

MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-

value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector. MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector. This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).

Type

`array`

.spec.podTemplate.topologySpreadConstraints[].matchLabelKeys[]

Type

`string`

.spec.ref

Description

TaskRef can be used to refer to a specific instance of a task.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent Note: A Task with non-empty APIVersion and Kind is considered a Custom Task

Property	Type	Description
`bundle`	`string`	Bundle url reference to a Tekton Bundle. Deprecated: Please use ResolverRef with the bundles resolver instead. The field is staying there for go client backward compatibility, but is not used/allowed anymore.
`kind`	`string`	TaskKind indicates the Kind of the Task: 1. Namespaced Task when Kind is set to "Task". If Kind is "", it defaults to "Task". 2. Custom Task when Kind is non-empty and APIVersion is non-empty
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.ref.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.ref.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.spec

Description

Spec is a specification of a custom task

Type

`object`

Property	Type	Description
`apiVersion`	`string`	
`kind`	`string`	

Property	Type	Description
`metadata`	`ObjectMeta`	PipelineTaskMetadata contains the labels or annotations for an EmbeddedTask
`spec`	`object`	Spec is a specification of a custom task

.spec.spec.spec

Description

Spec is a specification of a custom task

Type

`object`

.spec.workspaces

Description

Workspaces is a list of WorkspaceBindings from volumes to workspaces.

Type

`array`

.spec.workspaces[]

Description

WorkspaceBinding maps a Task's declared workspace to a Volume.

Type

`object`

Required

`name`

Property	Type	Description
`configMap`	`object`	ConfigMap represents a configMap that should popu workspace.
`csi`	`object`	CSI (Container Storage Interface) represents ephem storage that is handled by certain external CSI drive
`emptyDir`	`object`	EmptyDir represents a temporary directory that shar Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes ↗ Either this OR PersistentVolumeClaim can be use
`name`	`string`	Name is the name of the workspace populated by th
`persistentVolumeClaim`	`object`	PersistentVolumeClaimVolumeSource represents a to a PersistentVolumeClaim in the same namespace this OR EmptyDir can be used.
`projected`	`object`	Projected represents a projected volume that should this workspace.
`secret`	`object`	Secret represents a secret that should populate this workspace.
`subPath`	`string`	SubPath is optionally a directory on the volume whic be used for this binding (i.e. the volume will be mour

Property	Type	Description
		sub directory).
`volumeClaimTemplate`		VolumeClaimTemplate is a template for a claim that created in the same namespace. The PipelineRun c responsible for creating a unique claim for each insta PipelineRun. See PersistentVolumeClaim (API versi

.spec.workspaces[].configMap

Description

ConfigMap represents a configMap that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified

Property	Type	Description
		paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].csi

Description

CSI (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.
`fsType`	`string`	fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.
`nodePublishSecretRef`	`object`	nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.
`readOnly`	`boolean`	readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes stores driver-specific properties that are passed to the CSI driver.

Property	Type	Description
		Consult your driver's documentation for supported values.

.spec.workspaces[].csi.nodePublishSecretRef

Description

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.workspaces[].csi.volumeAttributes

Description

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

Type

`object`

.spec.workspaces[].emptyDir

Description

EmptyDir represents a temporary directory that shares a Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir Either this OR PersistentVolumeClaim can be used.

Type

`object`

Property	Type	Description
`medium`	`string`	medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗
`sizeLimit`		sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

.spec.workspaces[].persistentVolumeClaim

Description

PersistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. Either this OR EmptyDir can be used.

Type

`object`

Required

`claimName`

Property	Type	Description
`claimName`	`string`	claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims ↗
`readOnly`	`boolean`	readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

.spec.workspaces[].projected

Description

Projected represents a projected volume that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within

Property	Type	Description
		the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`sources`	`array`	sources is the list of volume projections. Each entry in this list handles one source.

.spec.workspaces[].projected.sources

Description

sources is the list of volume projections. Each entry in this list handles one source.

Type

`array`

.spec.workspaces[].projected.sources[]

Description

Projection that may be projected along with other supported volume types. Exactly one of these fields must be set.

Type

`object`

Property	Type	Description
`clusterTrustBundle`	`object`	ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate.

Property	Type	Description
		ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.
`configMap`	`object`	configMap information about the configMap data to project
`downwardAPI`	`object`	downwardAPI information about the downwardAPI data to project
`podCertificate`	`object`	Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec.

Property	Type	Description
		Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.
`secret`	`object`	secret information about the secret data to project

Property	Type	Description
`serviceAccountToken`	`object`	serviceAccountToken is information about the serviceAccountToken data to project

.spec.workspaces[].projected.sources[].clusterTrustBundle

Description

ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Type

`object`

Required

`path`

Property	Type	Description
`labelSelector`	`object`	Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Property	Type	Description
`name`	`string`	Select a single ClusterTrustBundle by object name. Mutually-exclusive with signerName and labelSelector.
`optional`	`boolean`	If true, don't block pod startup if the referenced ClusterTrustBundle(s) aren't available. If using name, then the named ClusterTrustBundle is allowed not to exist. If using signerName, then the combination of signerName and labelSelector is allowed to match zero ClusterTrustBundles.
`path`	`string`	Relative path from the volume root to write the bundle.
`signerName`	`string`	Select all ClusterTrustBundles that match this signer name. Mutually-exclusive with name. The contents of all selected ClusterTrustBundles will be unified and deduplicated.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector

Description

Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].projected.sources[].configMap

Description

configMap information about the configMap data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost

Property	Type	Description
		certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].projected.sources[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].downwardAPI

Description

downwardAPI information about the downwardAPI data to project

Type

`object`

Property	Type	Description
`items`	`array`	Items is a list of DownwardAPIVolume file

.spec.workspaces[].projected.sources[].downwardAPI.items

Description

Items is a list of DownwardAPIVolume file

Type

`array`

.spec.workspaces[].projected.sources[].downwardAPI.items[]

Description

DownwardAPIVolumeFile represents information to create the file containing the pod field

Type

`object`

Required

`path`

Property	Type	Description
`fieldRef`	`object`	Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.
`mode`	`integer`	Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the

Property	Type	Description
		file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '..' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

.spec.workspaces[].projected.sources[].downwardAPI.items[].fieldRef

Description

Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.workspaces[].projected.sources[].downwardAPI.items[].resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.workspaces[].projected.sources[].podCertificate

Description

Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.

Type

`object`

Required

`keyType` `signerName`

Property	Type	Description
`certificateChainPath`	`string`	Write the certificate chain at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to

Property	Type	Description
		check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.
`credentialBundlePath`	`string`	Write the credential bundle at this path in the projected volume. The credential bundle is a single file that contains multiple PEM blocks. The first PEM block is a PRIVATE KEY block, containing a PKCS#8 private key. The remaining blocks are CERTIFICATE blocks, containing the issued certificate chain from the signer (leaf and any intermediates). Using credentialBundlePath lets your Pod's application code make a single atomic read that retrieves a consistent key and certificate chain. If you project them to separate files, your application code will need to additionally check that the leaf certificate was issued to the key.
`keyPath`	`string`	Write the key at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.

Property	Type	Description
`keyType`	`string`	The type of keypair Kubelet will generate for the pod. Valid values are "RSA3072", "RSA4096", "ECDSAP256", "ECDSAP384", "ECDSAP521", and "ED25519".
`maxExpirationSeconds`	`integer`	maxExpirationSeconds is the maximum lifetime permitted for the certificate. Kubelet copies this value verbatim into the PodCertificateRequests it generates for this projection. If omitted, kube-apiserver will set it to 86400(24 hours). kube-apiserver will reject values shorter than 3600 (1 hour). The maximum allowable value is 7862400 (91 days). The signer implementation is then free to issue a certificate with any lifetime *shorter* than MaxExpirationSeconds, but no shorter than 3600 seconds (1 hour). This constraint is enforced by kube-apiserver. `kubernetes.io` signers will never issue certificates with a lifetime longer than 24 hours.
`signerName`	`string`	Kubelet's generated CSRs will be addressed to this signer.

Property	Type	Description
`userAnnotations`	`object`	userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

.spec.workspaces[].projected.sources[].podCertificate.userAnnotations

Description

userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

Type

`object`

.spec.workspaces[].projected.sources[].secret

Description

secret information about the secret data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional field specify whether the Secret or its key must be defined

.spec.workspaces[].projected.sources[].secret.items

Description

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that

Property	Type	Description
		affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].serviceAccountToken

Description

serviceAccountToken is information about the serviceAccountToken data to project

Type

`object`

Required

`path`

Property	Type	Description
`audience`	`string`	audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.
`expirationSeconds`	`integer`	expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin

Property	Type	Description
		will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent of its time to live or if the token is older than 24 hours.Defaults to 1 hour and must be at least 10 minutes.
`path`	`string`	path is the path relative to the mount point of the file to project the token into.

.spec.workspaces[].secret

Description

Secret represents a secret that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a

Property	Type	Description
		file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`optional`	`boolean`	optional field specify whether the Secret or its keys must be defined
`secretName`	`string`	secretName is the name of the secret in the pod's namespace to use. More info: https://kubernetes.io/docs/concepts/storage/volumes#secret ↗

.spec.workspaces[].secret.items

Description

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.status

Description

RunStatus defines the observed state of Run

Type

`object`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`completionTime`	`string`	CompletionTime is the time the build completed.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`extraFields`	`object`	ExtraFields holds arbitrary fields provided by the custom task controller.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`results`	`array`	Results reports any output result values to be consumed by later tasks in a pipeline.
`retriesStatus`	`array`	RetriesStatus contains the history of RunStatus, in case of a retry.

Property	Type	Description
`startTime`	`string`	StartTime is the time the build is actually started.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	Type of condition.

.status.extraFields

Description

ExtraFields holds arbitrary fields provided by the custom task controller.

Type

`object`

.status.results

Description

Results reports any output result values to be consumed by later tasks in a pipeline.

Type

`array`

.status.results[]

Description

RunResult used to describe the results of a task

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name the given name
`value`	`string`	Value the given value of the result

.status.retriesStatus

Description

RetriesStatus contains the history of RunStatus, in case of a retry.

Type

`array`

.status.retriesStatus[]

Type

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs`
  - `DELETE`: delete collection of Run
  - `GET`: list objects of kind Run
  - `POST`: create a new Run
- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs/{name}`
  - `DELETE`: delete the specified Run
  - `GET`: read the specified Run
  - `PATCH`: partially update the specified Run
  - `PUT`: replace the specified Run
- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs/{name}/status`
  - `GET`: read status of the specified Run
  - `PATCH`: partially update status of the specified Run
  - `PUT`: replace status of the specified Run

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs

HTTP method

`DELETE`

Description

delete collection of Run

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind Run

HTTP responses

HTTP code	Response body
200 - OK	`RunList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing

Parameter	Type	Description
		unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Run` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
201 - Created	`Run` schema
202 - Accepted	`Run` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs/{name}

HTTP method

`DELETE`

Description

delete the specified Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified Run

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Run` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
201 - Created	`Run` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/runs/{name}/status

HTTP method

`GET`

Description

read status of the specified Run

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun

Parameter	Type	Description
		directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified Run

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`Run` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`Run` schema
201 - Created	`Run` schema

HTTP code	Response body
401 - Unauthorized	Empty

CustomRun [tekton.dev/v1]

Description

CustomRun represents a single execution of a Custom Task.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

Property	Type	Description
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta ↗`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	CustomRunSpec defines the desired state of CustomRun
`status`	`object`	CustomRunStatus defines the observed state of CustomRun

.spec

Description

CustomRunSpec defines the desired state of CustomRun

Type

`object`

Property	Type	Description
`customRef`	`object`	TaskRef can be used to refer to a specific instance of a task.

Property	Type	Description
`customSpec`	`object`	Spec is a specification of a custom task
`params`	`array`	Params is a list of Param
`retries`	`integer`	Used for propagating retries count to custom tasks
`serviceAccountName`	`string`	
`status`	`string`	Used for cancelling a customrun (and maybe more later on)
`statusMessage`	`string`	Status message for cancellation.
`timeout`	`string`	Time after which the custom-task times out. Refer Go's ParseDuration documentation for expected format: https://golang.org/pkg/time/#ParseDuration ↗
`workspaces`	`array`	Workspaces is a list of WorkspaceBindings from volumes to workspaces.

.spec.customRef

Description

TaskRef can be used to refer to a specific instance of a task.

Type

`object`

Property	Type	Description
`apiVersion`	`string`	API version of the referent Note: A Task with non-empty APIVersion and Kind is considered a Custom Task
`bundle`	`string`	Bundle url reference to a Tekton Bundle. Deprecated: Please use ResolverRef with the bundles resolver instead. The field is staying there for go client backward compatibility, but is not used/allowed anymore.
`kind`	`string`	TaskKind indicates the Kind of the Task: 1. Namespaced Task when Kind is set to "Task". If Kind is "", it defaults to "Task". 2. Custom Task when Kind is non-empty and APIVersion is non-empty
`name`	`string`	Name of the referent; More info: http://kubernetes.io/docs/user-guide/identifiers#names ↗
`params`	`array`	Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Property	Type	Description
`resolver`	`string`	Resolver is the name of the resolver that should perform resolution of the referenced Tekton resource, such as "git".

.spec.customRef.params

Description

Params contains the parameters used to identify the referenced Tekton resource. Example entries might include "repo" or "path" but the set of params ultimately depends on the chosen resolver.

Type

`array`

.spec.customRef.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.customSpec

Description

Spec is a specification of a custom task

Type

`object`

Property	Type	Description
`apiVersion`	`string`	
`kind`	`string`	
`metadata`	`ObjectMeta`	PipelineTaskMetadata contains the labels or annotations for an EmbeddedTask
`spec`	`object`	Spec is a specification of a custom task

.spec.customSpec.spec

Description

Spec is a specification of a custom task

Type

`object`

.spec.params

Description

Params is a list of Param

Type

`array`

.spec.params[]

Description

Param declares an ParamValues to use for the parameter called name.

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	
`value`		

.spec.workspaces

Description

Workspaces is a list of WorkspaceBindings from volumes to workspaces.

Type

`array`

.spec.workspaces[]

Description

WorkspaceBinding maps a Task's declared workspace to a Volume.

Type

`object`

Required

`name`

Property	Type	Description
`configMap`	`object`	ConfigMap represents a configMap that should popu workspace.
`csi`	`object`	CSI (Container Storage Interface) represents ephem storage that is handled by certain external CSI drive
`emptyDir`	`object`	EmptyDir represents a temporary directory that shar Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes ↗ Either this OR PersistentVolumeClaim can be use
`name`	`string`	Name is the name of the workspace populated by th
`persistentVolumeClaim`	`object`	PersistentVolumeClaimVolumeSource represents a to a PersistentVolumeClaim in the same namespace this OR EmptyDir can be used.
`projected`	`object`	Projected represents a projected volume that should this workspace.
`secret`	`object`	Secret represents a secret that should populate this workspace.
`subPath`	`string`	SubPath is optionally a directory on the volume whic be used for this binding (i.e. the volume will be mour

Property	Type	Description
		sub directory).
`volumeClaimTemplate`		VolumeClaimTemplate is a template for a claim that created in the same namespace. The PipelineRun c responsible for creating a unique claim for each insta PipelineRun. See PersistentVolumeClaim (API versi

.spec.workspaces[].configMap

Description

ConfigMap represents a configMap that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified

Property	Type	Description
		paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].csi

Description

CSI (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers.

Type

`object`

Required

`driver`

Property	Type	Description
`driver`	`string`	driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.
`fsType`	`string`	fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.
`nodePublishSecretRef`	`object`	nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.
`readOnly`	`boolean`	readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).
`volumeAttributes`	`object`	volumeAttributes stores driver-specific properties that are passed to the CSI driver.

Property	Type	Description
		Consult your driver's documentation for supported values.

.spec.workspaces[].csi.nodePublishSecretRef

Description

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

Type

`object`

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.workspaces[].csi.volumeAttributes

Description

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

Type

`object`

.spec.workspaces[].emptyDir

Description

EmptyDir represents a temporary directory that shares a Task's lifetime. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir Either this OR PersistentVolumeClaim can be used.

Type

`object`

Property	Type	Description
`medium`	`string`	medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗
`sizeLimit`		sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: https://kubernetes.io/docs/concepts/storage/volumes#emptydir ↗

.spec.workspaces[].persistentVolumeClaim

Description

PersistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. Either this OR EmptyDir can be used.

Type

`object`

Required

`claimName`

Property	Type	Description
`claimName`	`string`	claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims ↗
`readOnly`	`boolean`	readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

.spec.workspaces[].projected

Description

Projected represents a projected volume that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within

Property	Type	Description
		the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`sources`	`array`	sources is the list of volume projections. Each entry in this list handles one source.

.spec.workspaces[].projected.sources

Description

sources is the list of volume projections. Each entry in this list handles one source.

Type

`array`

.spec.workspaces[].projected.sources[]

Description

Projection that may be projected along with other supported volume types. Exactly one of these fields must be set.

Type

`object`

Property	Type	Description
`clusterTrustBundle`	`object`	ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate.

Property	Type	Description
		ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.
`configMap`	`object`	configMap information about the configMap data to project
`downwardAPI`	`object`	downwardAPI information about the downwardAPI data to project
`podCertificate`	`object`	Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec.

Property	Type	Description
		Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.
`secret`	`object`	secret information about the secret data to project

Property	Type	Description
`serviceAccountToken`	`object`	serviceAccountToken is information about the serviceAccountToken data to project

.spec.workspaces[].projected.sources[].clusterTrustBundle

Description

ClusterTrustBundle allows a pod to access the `.spec.trustBundle` field of ClusterTrustBundle objects in an auto-updating file. Alpha, gated by the ClusterTrustBundleProjection feature gate. ClusterTrustBundle objects can either be selected by name, or by the combination of signer name and a label selector. Kubelet performs aggressive normalization of the PEM contents written into the pod filesystem. Esoteric PEM features such as inter-block comments and block headers are stripped. Certificates are deduplicated. The ordering of certificates within the file is arbitrary, and Kubelet may change the order over time.

Type

`object`

Required

`path`

Property	Type	Description
`labelSelector`	`object`	Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Property	Type	Description
`name`	`string`	Select a single ClusterTrustBundle by object name. Mutually-exclusive with signerName and labelSelector.
`optional`	`boolean`	If true, don't block pod startup if the referenced ClusterTrustBundle(s) aren't available. If using name, then the named ClusterTrustBundle is allowed not to exist. If using signerName, then the combination of signerName and labelSelector is allowed to match zero ClusterTrustBundles.
`path`	`string`	Relative path from the volume root to write the bundle.
`signerName`	`string`	Select all ClusterTrustBundles that match this signer name. Mutually-exclusive with name. The contents of all selected ClusterTrustBundles will be unified and deduplicated.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector

Description

Select all ClusterTrustBundles that match this label selector. Only has effect if signerName is set. Mutually-exclusive with name. If unset, interpreted as "match nothing". If set but empty, interpreted as "match everything".

Type

`object`

Property	Type	Description
`matchExpressions`	`array`	matchExpressions is a list of label selector requirements. The requirements are ANDed.
`matchLabels`	`object`	matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions

Description

matchExpressions is a list of label selector requirements. The requirements are ANDed.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[]

Description

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

Type

`object`

Required

`key` `operator`

Property	Type	Description
`key`	`string`	key is the label key that the selector applies to.
`operator`	`string`	operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
`values`	`array`	values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values

Description

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

Type

`array`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchExpressions[].values[]

Type

`string`

.spec.workspaces[].projected.sources[].clusterTrustBundle.labelSelector.matchLabels

Description

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

Type

`object`

.spec.workspaces[].projected.sources[].configMap

Description

configMap information about the configMap data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost

Property	Type	Description
		certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	optional specify whether the ConfigMap or its keys must be defined

.spec.workspaces[].projected.sources[].configMap.items

Description

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].configMap.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].downwardAPI

Description

downwardAPI information about the downwardAPI data to project

Type

`object`

Property	Type	Description
`items`	`array`	Items is a list of DownwardAPIVolume file

.spec.workspaces[].projected.sources[].downwardAPI.items

Description

Items is a list of DownwardAPIVolume file

Type

`array`

.spec.workspaces[].projected.sources[].downwardAPI.items[]

Description

DownwardAPIVolumeFile represents information to create the file containing the pod field

Type

`object`

Required

`path`

Property	Type	Description
`fieldRef`	`object`	Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.
`mode`	`integer`	Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the

Property	Type	Description
		file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '..' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

.spec.workspaces[].projected.sources[].downwardAPI.items[].fieldRef

Description

Required: Selects a field of the pod: only annotations, labels, name, namespace and uid are supported.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.workspaces[].projected.sources[].downwardAPI.items[].resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.workspaces[].projected.sources[].podCertificate

Description

Projects an auto-rotating credential bundle (private key and certificate chain) that the pod can use either as a TLS client or server. Kubelet generates a private key and uses it to send a PodCertificateRequest to the named signer. Once the signer approves the request and issues a certificate chain, Kubelet writes the key and certificate chain to the pod filesystem. The pod does not start until certificates have been issued for each podCertificate projected volume source in its spec. Kubelet will begin trying to rotate the certificate at the time indicated by the signer using the PodCertificateRequest.Status.BeginRefreshAt timestamp. Kubelet can write a single file, indicated by the credentialBundlePath field, or separate files, indicated by the keyPath and certificateChainPath fields. The credential bundle is a single file in PEM format. The first PEM entry is the private key (in PKCS#8 format), and the remaining PEM entries are the certificate chain issued by the signer (typically, signers will return their certificate chain in leaf-to-root order). Prefer using the credential bundle format, since your application code can read it atomically. If you use keyPath and certificateChainPath, your application must make two separate file reads. If these coincide with a certificate rotation, it is possible that the private key and leaf certificate you read may not correspond to each other. Your application will need to check for this condition, and re-read until they are consistent. The named signer controls chooses the format of the certificate it issues; consult the signer implementation's documentation to learn how to use the certificates it issues.

Type

`object`

Required

`keyType` `signerName`

Property	Type	Description
`certificateChainPath`	`string`	Write the certificate chain at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to

Property	Type	Description
		check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.
`credentialBundlePath`	`string`	Write the credential bundle at this path in the projected volume. The credential bundle is a single file that contains multiple PEM blocks. The first PEM block is a PRIVATE KEY block, containing a PKCS#8 private key. The remaining blocks are CERTIFICATE blocks, containing the issued certificate chain from the signer (leaf and any intermediates). Using credentialBundlePath lets your Pod's application code make a single atomic read that retrieves a consistent key and certificate chain. If you project them to separate files, your application code will need to additionally check that the leaf certificate was issued to the key.
`keyPath`	`string`	Write the key at this path in the projected volume. Most applications should use credentialBundlePath. When using keyPath and certificateChainPath, your application needs to check that the key and leaf certificate are consistent, because it is possible to read the files mid-rotation.

Property	Type	Description
`keyType`	`string`	The type of keypair Kubelet will generate for the pod. Valid values are "RSA3072", "RSA4096", "ECDSAP256", "ECDSAP384", "ECDSAP521", and "ED25519".
`maxExpirationSeconds`	`integer`	maxExpirationSeconds is the maximum lifetime permitted for the certificate. Kubelet copies this value verbatim into the PodCertificateRequests it generates for this projection. If omitted, kube-apiserver will set it to 86400(24 hours). kube-apiserver will reject values shorter than 3600 (1 hour). The maximum allowable value is 7862400 (91 days). The signer implementation is then free to issue a certificate with any lifetime *shorter* than MaxExpirationSeconds, but no shorter than 3600 seconds (1 hour). This constraint is enforced by kube-apiserver. `kubernetes.io` signers will never issue certificates with a lifetime longer than 24 hours.
`signerName`	`string`	Kubelet's generated CSRs will be addressed to this signer.

Property	Type	Description
`userAnnotations`	`object`	userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

.spec.workspaces[].projected.sources[].podCertificate.userAnnotations

Description

userAnnotations allow pod authors to pass additional information to the signer implementation. Kubernetes does not restrict or validate this metadata in any way. These values are copied verbatim into the `spec.unverifiedUserAnnotations` field of the PodCertificateRequest objects that Kubelet creates. Entries are subject to the same validation as object metadata annotations, with the addition that all keys must be domain-prefixed. No restrictions are placed on values, except an overall size limitation on the entire field. Signers should document the keys and values they support. Signers should deny requests that contain keys they do not recognize.

Type

`object`

.spec.workspaces[].projected.sources[].secret

Description

secret information about the secret data to project

Type

`object`

Property	Type	Description
`items`	`array`	items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names↗
`optional`	`boolean`	optional field specify whether the Secret or its key must be defined

.spec.workspaces[].projected.sources[].secret.items

Description

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].projected.sources[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that

Property	Type	Description
		affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.spec.workspaces[].projected.sources[].serviceAccountToken

Description

serviceAccountToken is information about the serviceAccountToken data to project

Type

`object`

Required

`path`

Property	Type	Description
`audience`	`string`	audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.
`expirationSeconds`	`integer`	expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin

Property	Type	Description
		will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent of its time to live or if the token is older than 24 hours.Defaults to 1 hour and must be at least 10 minutes.
`path`	`string`	path is the path relative to the mount point of the file to project the token into.

.spec.workspaces[].secret

Description

Secret represents a secret that should populate this workspace.

Type

`object`

Property	Type	Description
`defaultMode`	`integer`	defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`items`	`array`	items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a

Property	Type	Description
		file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.
`optional`	`boolean`	optional field specify whether the Secret or its keys must be defined
`secretName`	`string`	secretName is the name of the secret in the pod's namespace to use. More info: https://kubernetes.io/docs/concepts/storage/volumes#secret ↗

.spec.workspaces[].secret.items

Description

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

Type

`array`

.spec.workspaces[].secret.items[]

Description

Maps a string key to a path within a volume.

Type

`object`

Required

`key` `path`

Property	Type	Description
`key`	`string`	key is the key to project.
`mode`	`integer`	mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.
`path`	`string`	path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '..'.

.status

Description

CustomRunStatus defines the observed state of CustomRun

Type

`object`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`completionTime`	`string`	CompletionTime is the time the build completed.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`extraFields`		ExtraFields holds arbitrary fields provided by the custom task controller.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`results`	`array`	Results reports any output result values to be consumed by later tasks in a pipeline.
`retriesStatus`		RetriesStatus contains the history of CustomRunStatus, in case of a retry. See CustomRun.status (API version: tekton.dev/v1beta1)

Property	Type	Description
`startTime`	`string`	StartTime is the time the build is actually started.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	Type of condition.

.status.results

Description

Results reports any output result values to be consumed by later tasks in a pipeline.

Type

`array`

.status.results[]

Description

CustomRunResult used to describe the results of a task

Type

`object`

Required

`name` `value`

Property	Type	Description
`name`	`string`	Name the given name
`value`	`string`	Value the given value of the result

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns`
  - `DELETE`: delete collection of CustomRun
  - `GET`: list objects of kind CustomRun
  - `POST`: create a new CustomRun
- `/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns/{name}`
  - `DELETE`: delete the specified CustomRun
  - `GET`: read the specified CustomRun

- `PATCH`: partially update the specified CustomRun
- `PUT`: replace the specified CustomRun

- `/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns/{name}/status`
  - `GET`: read status of the specified CustomRun
  - `PATCH`: partially update status of the specified CustomRun
  - `PUT`: replace status of the specified CustomRun

/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns

HTTP method

`DELETE`

Description

delete collection of CustomRun

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind CustomRun

HTTP responses

HTTP code	Response body
200 - OK	`CustomRunList` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`CustomRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
201 - Created	`CustomRun` schema
202 - Accepted	`CustomRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns/{name}

HTTP method

`DELETE`

Description

delete the specified CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
202 - Accepted	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified CustomRun

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore:

Parameter	Type	Description
		This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

Parameter	Type	Description
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`CustomRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
201 - Created	`CustomRun` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1beta1/namespaces/{namespace}/customruns/{name}/status

HTTP method

`GET`

Description

read status of the specified CustomRun

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields.

Parameter	Type	Description
		This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified CustomRun

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object,

Parameter	Type	Description
		and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`CustomRun` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`CustomRun` schema
201 - Created	`CustomRun` schema
401 - Unauthorized	Empty

StepAction [tekton.dev/v1]

Description

StepAction represents the actionable components of Step. The Step can only reference it from the cluster or using remote resolution.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info:

Property	Type	Description
		https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the desired state of the Step from the client

.spec

Description

Spec holds the desired state of the Step from the client

Type

`object`

Property	Type	Description
`args`	`array`	Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-

Property	Type	Description
		application/define-command-argument-container/#running-a-command-in-a-shell ↗
`command`	`array`	Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell ↗
`description`	`string`	Description is a user-facing description of the stepaction that may be used to populate a UI.
`env`	`array`	List of environment variables to set in the container. Cannot be updated.
`image`	`string`	Image reference name to run for this StepAction. More info: https://kubernetes.io/docs/concepts/containers/images ↗

Property	Type	Description
`params`	`array`	Params is a list of input parameters required to run the stepAction. Params must be supplied as inputs in Steps unless they declare a defaultvalue.
`results`	`array`	Results are values that this StepAction can output
`script`	`string`	Script is the contents of an executable file to execute. If Script is not empty, the Step cannot have an Command and the Args will be passed to the Script.
`securityContext`	`object`	SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ ↗ The value set in StepAction will take precedence over the value from Task.
`volumeMounts`	`array`	Volumes to mount into the Step's filesystem. Cannot be updated.
`workingDir`	`string`	Step's working directory. If not specified, the container runtime's default will be used, which might be configured in the container image. Cannot be updated.

.spec.args

Description

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.args[]

Type

`string`

.spec.command

Description

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references $(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal "$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell

Type

`array`

.spec.command[]

Type

`string`

.spec.env

Description

List of environment variables to set in the container. Cannot be updated.

Type

`array`

.spec.env[]

Description

EnvVar represents an environment variable present in a Container.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name of the environment variable. May consist of any printable ASCII characters except '='.
`value`	`string`	Variable references $(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double $$ are reduced to a single $, which allows for escaping the $(VAR_NAME) syntax: i.e. "$$(VAR_NAME)" will produce the string literal

Property	Type	Description
		"$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".
`valueFrom`	`object`	Source for the environment variable's value. Cannot be used if value is not empty.

.spec.env[].valueFrom

Description

Source for the environment variable's value. Cannot be used if value is not empty.

Type

`object`

Property	Type	Description
`configMapKeyRef`	`object`	Selects a key of a ConfigMap.
`fieldRef`	`object`	Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.
`fileKeyRef`	`object`	FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Property	Type	Description
`resourceFieldRef`	`object`	Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.
`secretKeyRef`	`object`	Selects a key of a secret in the pod's namespace

.spec.env[].valueFrom.configMapKeyRef

Description

Selects a key of a ConfigMap.

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key to select.

Property	Type	Description
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the ConfigMap or its key must be defined

.spec.env[].valueFrom.fieldRef

Description

Selects a field of the pod: supports metadata.name, metadata.namespace, `metadata.labels['<KEY>']`, `metadata.annotations['<KEY>']`, spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

Type

`object`

Required

`fieldPath`

Property	Type	Description
`apiVersion`	`string`	Version of the schema the FieldPath is written in terms of, defaults to "v1".
`fieldPath`	`string`	Path of the field to select in the specified API version.

.spec.env[].valueFrom.fileKeyRef

Description

FileKeyRef selects a key of the env file. Requires the EnvFiles feature gate to be enabled.

Type

`object`

Required

`key` `path` `volumeName`

Property	Type	Description
`key`	`string`	The key within the env file. An invalid key will prevent the pod from starting. The keys defined within a source may consist of any printable ASCII characters except '='. During Alpha stage of the EnvFiles feature gate, the key size is limited to 128 characters.
`optional`	`boolean`	Specify whether the file or its key must be defined. If the file or key does not exist, then the env var is not published. If optional is set to true and the specified key does not exist, the environment variable will not be set in the Pod's containers. If optional is set to false and the specified key does not exist, an error will be returned during Pod creation.
`path`	`string`	The path within the volume from which to select the file. Must be relative and may not contain the '..' path or start with '..'.

Property	Type	Description
`volumeName`	`string`	The name of the volume mount containing the env file.

.spec.env[].valueFrom.resourceFieldRef

Description

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

Type

`object`

Required

`resource`

Property	Type	Description
`containerName`	`string`	Container name: required for volumes, optional for env vars
`divisor`		Specifies the output format of the exposed resources, defaults to "1"
`resource`	`string`	Required: resource to select

.spec.env[].valueFrom.secretKeyRef

Description

Selects a key of a secret in the pod's namespace

Type

`object`

Required

`key`

Property	Type	Description
`key`	`string`	The key of the secret to select from. Must be a valid secret key.
`name`	`string`	Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗
`optional`	`boolean`	Specify whether the Secret or its key must be defined

.spec.params

Description

Params is a list of input parameters required to run the stepAction. Params must be supplied as inputs in Steps unless they declare a defaultvalue.

Type

`array`

.spec.params[]

Description

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources). Parameter values are provided by users as inputs on a TaskRun or PipelineRun.

Type

`object`

Required

`name`

Property	Type	Description
`default`		Default is the value a parameter takes if no input value is supplied. If default is set, a Task may be executed without a supplied value for the parameter.
`description`	`string`	Description is a user-facing description of the parameter that may be used to populate a UI.
`enum`	`array`	Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.
`name`	`string`	Name declares the name by which a parameter is referenced.
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs parameter.

Property	Type	Description
`type`	`string`	Type is the user-specified type of the parameter. The possible types are currently "string", "array" and "object", and "string" is the default.

.spec.params[].enum

Description

Enum declares a set of allowed param input values for tasks/pipelines that can be validated. If Enum is not set, no input validation is performed for the param.

Type

`array`

.spec.params[].enum[]

Type

`string`

.spec.params[].properties

Description

Properties is the JSON Schema properties to support key-value pairs parameter.

Type

`object`

.spec.results

Description

Results are values that this StepAction can output

Type

`array`

.spec.results[]

Description

StepResult used to describe the Results of a Step.

Type

`object`

Required

`name`

Property	Type	Description
`description`	`string`	Description is a human-readable description of the result
`name`	`string`	Name the given name
`properties`	`object`	Properties is the JSON Schema properties to support key-value pairs results.
`type`	`string`	The possible types are 'string', 'array', and 'object', with 'string' as the default.

.spec.results[].properties

Description

Properties is the JSON Schema properties to support key-value pairs results.

Type

`object`

.spec.securityContext

Description

SecurityContext defines the security options the Step should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: https://kubernetes.io/docs/tasks/configure-pod-container/security-context/ The value set in StepAction will take precedence over the value from Task.

Type

`object`

Property	Type	Description
`allowPrivilegeEscalation`	`boolean`	AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1. run as Privileged 2. has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.
`appArmorProfile`	`object`	appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Property	Type	Description
`capabilities`	`object`	The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.
`privileged`	`boolean`	Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.
`procMount`	`string`	procMount denotes the type of proc mount to use for the containers. The default value is Default which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.
`readOnlyRootFilesystem`	`boolean`	Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.
`runAsGroup`	`integer`	The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in

Property	Type	Description
		PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`runAsNonRoot`	`boolean`	Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.
`runAsUser`	`integer`	The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seLinuxOptions`	`object`	The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set

Property	Type	Description
		in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.
`seccompProfile`	`object`	The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.
`windowsOptions`	`object`	The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

.spec.securityContext.appArmorProfile

Description

appArmorProfile is the AppArmor options to use by this container. If set, this profile overrides the pod's appArmorProfile. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".
`type`	`string`	type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

.spec.securityContext.capabilities

Description

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`add`	`array`	Added capabilities

Property	Type	Description
`drop`	`array`	Removed capabilities

.spec.securityContext.capabilities.add

Description

Added capabilities

Type

`array`

.spec.securityContext.capabilities.add[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.securityContext.capabilities.drop

Description

Removed capabilities

Type

`array`

.spec.securityContext.capabilities.drop[]

Description

Capability represent POSIX capabilities type

Type

`string`

.spec.securityContext.seLinuxOptions

Description

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Property	Type	Description
`level`	`string`	Level is SELinux level label that applies to the container.
`role`	`string`	Role is a SELinux role label that applies to the container.
`type`	`string`	Type is a SELinux type label that applies to the container.
`user`	`string`	User is a SELinux user label that applies to the container.

.spec.securityContext.seccompProfile

Description

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

Type

`object`

Required

`type`

Property	Type	Description
`localhostProfile`	`string`	localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.
`type`	`string`	type indicates which kind of seccomp profile will be applied. Valid options are: Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

.spec.securityContext.windowsOptions

Description

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

Type

`object`

Property	Type	Description
`gmsaCredentialSpec`	`string`	GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.
`gmsaCredentialSpecName`	`string`	GMSACredentialSpecName is the name of the GMSA credential spec to use.
`hostProcess`	`boolean`	HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.
`runAsUserName`	`string`	The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

.spec.volumeMounts

Description

Volumes to mount into the Step's filesystem. Cannot be updated.

Type

`array`

.spec.volumeMounts[]

Description

VolumeMount describes a mounting of a Volume within a container.

Type

`object`

Required

`mountPath` `name`

Property	Type	Description
`mountPath`	`string`	Path within the container at which the volume should be mounted. Must not contain ':'.
`mountPropagation`	`string`	mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10. When RecursiveReadOnly is set to IfPossible or to Enabled, MountPropagation must be None or unspecified (which defaults to None).

Property	Type	Description
`name`	`string`	This must match the Name of a Volume.
`readOnly`	`boolean`	Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.
`recursiveReadOnly`	`string`	RecursiveReadOnly specifies whether read-only mounts should be handled recursively. If ReadOnly is false, this field has no meaning and must be unspecified. If ReadOnly is true, and this field is set to Disabled, the mount is not made recursively read-only. If this field is set to IfPossible, the mount is made recursively read-only, if it is supported by the container runtime. If this field is set to Enabled, the mount is made recursively read-only if it is supported by the container runtime, otherwise the pod will not be started and an error will be generated to indicate the reason. If this field is set to IfPossible or Enabled, MountPropagation must be set to None (or be unspecified, which defaults to None). If this field is not specified, it is treated as an equivalent of Disabled.

Property	Type	Description
`subPath`	`string`	Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).
`subPathExpr`	`string`	Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references $(VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/stepactions`
  - `DELETE`: delete collection of StepAction
  - `GET`: list objects of kind StepAction
  - `POST`: create a new StepAction
- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/stepactions/{name}`
  - `DELETE`: delete the specified StepAction
  - `GET`: read the specified StepAction
  - `PATCH`: partially update the specified StepAction
  - `PUT`: replace the specified StepAction

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/stepactions

HTTP method

`DELETE`

Description

delete collection of StepAction

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind StepAction

HTTP responses

HTTP code	Response body
200 - OK	`StepActionList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new StepAction

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`StepAction` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`StepAction` schema
201 - Created	`StepAction` schema

HTTP code	Response body
202 - Accepted	`StepAction` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/stepactions/{name}

HTTP method

`DELETE`

Description

delete the specified StepAction

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status` ↗ schema
202 - Accepted	`Status` ↗ schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified StepAction

HTTP responses

HTTP code	Response body
200 - OK	`StepAction` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified StepAction

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server

Parameter	Type	Description
		will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`StepAction` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified StepAction

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the

Parameter	Type	Description
		request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`StepAction` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`StepAction` schema
201 - Created	`StepAction` schema
401 - Unauthorized	Empty

VerificationPolicy [tekton.dev/v1alpha1]

Description

VerificationPolicy defines the rules to verify Tekton resources. VerificationPolicy can config the mapping from resources to a list of public keys, so when verifying the resources we can use the corresponding public keys.

Type

`object`

Required

`spec`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

Property	Type	Description
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the desired state of the VerificationPolicy.

.spec

Description

Spec holds the desired state of the VerificationPolicy.

Type

`object`

Required

`authorities` `resources`

Property	Type	Description
`authorities`	`array`	Authorities defines the rules for validating signatures.

Property	Type	Description
`mode`	`string`	Mode controls whether a failing policy will fail the taskrun/pipelinerun, or only log the warnings enforce - fail the taskrun/pipelinerun if verification fails (default) warn - don't fail the taskrun/pipelinerun if verification fails but log warnings
`resources`	`array`	Resources defines the patterns of resources sources that should be subject to this policy. For example, we may want to apply this Policy from a certain GitHub repo. Then the ResourcesPattern should be valid regex. E.g. If using gitresolver, and we want to config keys from a certain git repo. `ResourcesPattern` can be `https://github.com/tektoncd/catalog.git`, we will use regex to filter out those resources.

.spec.authorities

Description

Authorities defines the rules for validating signatures.

Type

`array`

.spec.authorities[]

Description

The Authority block defines the keys for validating signatures.

Type

`object`

Required

`name`

Property	Type	Description
`key`	`object`	Key contains the public key to validate the resource.
`name`	`string`	Name is the name for this authority.

.spec.authorities[].key

Description

Key contains the public key to validate the resource.

Type

`object`

Property	Type	Description
`data`	`string`	Data contains the inline public key.
`hashAlgorithm`	`string`	HashAlgorithm always defaults to sha256 if the algorithm has
`kms`	`string`	KMS contains the KMS url of the public key Supported format could be: gcpkms://projects/[PROJECT]/locations/[LOCATION]>/keyRin For more examples please refer https://docs.sigstore.dev/cosi
`secretRef`	`object`	SecretRef sets a reference to a secret with the key.

.spec.authorities[].key.secretRef

Description

SecretRef sets a reference to a secret with the key.

Type

`object`

Property	Type	Description
`name`	`string`	name is unique within a namespace to reference a secret resource.
`namespace`	`string`	namespace defines the space within which the secret name must be unique.

.spec.resources

Description

Resources defines the patterns of resources sources that should be subject to this policy. For example, we may want to apply this Policy from a certain GitHub repo. Then the ResourcesPattern should be valid regex. E.g. If using gitresolver, and we want to config keys from a certain git repo. `ResourcesPattern` can be `https://github.com/tektoncd/catalog.git`, we will use regex to filter out those resources.

Type

`array`

.spec.resources[]

Description

ResourcePattern defines the pattern of the resource source

Type

`object`

Required

`pattern`

Property	Type	Description
`pattern`	`string`	Pattern defines a resource pattern. Regex is created to filter resources based on `Pattern` Example patterns: GitHub resource: https://github.com/tektoncd/catalog.git ↗, https://github.com/tektoncd/ ↗* Bundle resource: gcr.io/tekton-releases/catalog/upstream/git-clone, gcr.io/tekton-releases/catalog/upstream/* Hub resource: https://artifacthub.io/ ↗*,

API Endpoints

The following API endpoints are available:

- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/verificationpolicies`
  - `DELETE`: delete collection of VerificationPolicy
  - `GET`: list objects of kind VerificationPolicy
  - `POST`: create a new VerificationPolicy
- `/apis/tekton.dev/v1alpha1/namespaces/{namespace}/verificationpolicies/{name}`
  - `DELETE`: delete the specified VerificationPolicy
  - `GET`: read the specified VerificationPolicy
  - `PATCH`: partially update the specified VerificationPolicy
  - `PUT`: replace the specified VerificationPolicy

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/verificationpolicies

HTTP method

`DELETE`

Description

delete collection of VerificationPolicy

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind VerificationPolicy

HTTP responses

HTTP code	Response body
200 - OK	`VerificationPolicyList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new VerificationPolicy

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`VerificationPolicy` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`VerificationPolicy` schema
201 - Created	`VerificationPolicy` schema

HTTP code	Response body
202 - Accepted	`VerificationPolicy` schema
401 - Unauthorized	Empty

/apis/tekton.dev/v1alpha1/namespaces/{namespace}/verificationpolicies/{name}

HTTP method

`DELETE`

Description

delete the specified VerificationPolicy

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified VerificationPolicy

HTTP responses

HTTP code	Response body
200 - OK	`VerificationPolicy` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified VerificationPolicy

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server

Parameter	Type	Description
		will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`VerificationPolicy` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified VerificationPolicy

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the

Parameter	Type	Description
		request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`VerificationPolicy` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`VerificationPolicy` schema
201 - Created	`VerificationPolicy` schema
401 - Unauthorized	Empty

ResolutionRequest [resolution.tekton.dev/v1beta1]

Description

ResolutionRequest is an object for requesting the content of a Tekton resource like a pipeline.yaml.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In

Property	Type	Description
		CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the information for the request part of the resource request.
`status`	`object`	Status communicates the state of the request and, ultimately, the content of the resolved resource.

.spec

Description

Spec holds the information for the request part of the resource request.

Type

`object`

Property	Type	Description
`params`	`object`	Parameters are the runtime attributes passed to the resolver to help it figure out how to resolve the resource being requested. For example: repo URL, commit SHA, path to file, the kind of authentication to leverage, etc.

.spec.params

Description

Parameters are the runtime attributes passed to the resolver to help it figure out how to resolve the resource being requested. For example: repo URL, commit SHA, path to file, the kind of authentication to leverage, etc.

Type

`object`

.status

Description

Status communicates the state of the request and, ultimately, the content of the resolved resource.

Type

`object`

Required

`data` `refSource`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.

Property	Type	Description
`data`	`string`	Data is a string representation of the resolved content of the requested resource in-lined into the ResolutionRequest object.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`refSource`		RefSource is the source reference of the remote data that records where the remote file came from including the url, digest and the entrypoint.

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.

Property	Type	Description
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	Type of condition.

API Endpoints

The following API endpoints are available:

- `/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests`
  - `DELETE`: delete collection of ResolutionRequest
  - `GET`: list objects of kind ResolutionRequest
  - `POST`: create a new ResolutionRequest
- `/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests/{name}`
  - `DELETE`: delete the specified ResolutionRequest
  - `GET`: read the specified ResolutionRequest
  - `PATCH`: partially update the specified ResolutionRequest
  - `PUT`: replace the specified ResolutionRequest
- `/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests/{name}/status`
  - `GET`: read status of the specified ResolutionRequest
  - `PATCH`: partially update status of the specified ResolutionRequest
  - `PUT`: replace status of the specified ResolutionRequest

/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests

HTTP method

`DELETE`

Description

delete collection of ResolutionRequest

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind ResolutionRequest

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequestList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ResolutionRequest` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
201 - Created	`ResolutionRequest` schema

HTTP code	Response body
202 - Accepted	`ResolutionRequest` schema
401 - Unauthorized	Empty

/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests/{name}

HTTP method

`DELETE`

Description

delete the specified ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified ResolutionRequest

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server

Parameter	Type	Description
		will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the

Parameter	Type	Description
		request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ResolutionRequest` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
201 - Created	`ResolutionRequest` schema
401 - Unauthorized	Empty

/apis/resolution.tekton.dev/v1alpha1/namespaces/{namespace}/resolutionrequests/{name}/status

HTTP method

`GET`

Description

read status of the specified ResolutionRequest

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ResolutionRequest

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ResolutionRequest` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ResolutionRequest` schema
201 - Created	`ResolutionRequest` schema
401 - Unauthorized	Empty

ApprovalTask [openshiftpipelines.org/v1alpha1]

Description

ApprovalTask is a "wait for manual approval" Task.

Type

`object`

Required

`status`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

Property	Type	Description
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`metadata`	`ObjectMeta` ↗	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	Spec holds the desired state of the TaskGroup from the client
`status`	`object`	

.spec

Description

Spec holds the desired state of the TaskGroup from the client

Type

`object`

Required

`approvers` `numberOfApprovalsRequired`

Property	Type	Description
`approvers`	`array`	
`description`	`string`	

Property	Type	Description
`numberOfApprovalsRequired`	`integer`	

.spec.approvers

Type

`array`

.spec.approvers[]

Type

`object`

Required

`input` `name` `type`

Property	Type	Description
`input`	`string`	
`message`	`string`	
`name`	`string`	
`type`	`string`	
`users`	`array`	

.spec.approvers[].users

Type

`array`

.spec.approvers[].users[]

Type

`object`

Required

`input` `name`

Property	Type	Description
`input`	`string`	
`name`	`string`	

.status

Type

`object`

Required

`state`

Property	Type	Description
`annotations`	`object`	Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.
`approvalsReceived`	`integer`	ApprovalsReceived is the number of approvals received so far

Property	Type	Description
`approvalsRequired`	`integer`	ApprovalsRequired is the number of approvals required for the task
`approvers`	`array`	
`approversResponse`	`array`	
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.
`startTime`	`string`	StartTime is the time the build is actually started.
`state`	`string`	

.status.annotations

Description

Annotations is additional Status fields for the Resource to save some additional State as well as convey more information to the user. This is roughly akin to Annotations on any k8s resource, just the reconciler conveying richer information outwards.

Type

`object`

.status.approvers

Type

`array`

.status.approvers[]

Type

`string`

.status.approversResponse

Type

`array`

.status.approversResponse[]

Type

`object`

Required

`name` `response` `type`

Property	Type	Description
`groupMembers`	`array`	
`message`	`string`	
`name`	`string`	
`response`	`string`	
`type`	`string`	

.status.approversResponse[].groupMembers

Type

`array`

.status.approversResponse[].groupMembers[]

Type

`object`

Required

`name` `response`

Property	Type	Description
`message`	`string`	
`name`	`string`	
`response`	`string`	

.status.conditions

Description

Conditions the latest available observations of a resource's current state.

Type

`array`

.status.conditions[]

Description

Condition defines a readiness condition for a Knative resource. See: https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#typical-status-properties

Type

`object`

Required

`status` `type`

Property	Type	Description
`lastTransitionTime`	`string`	LastTransitionTime is the last time the condition transitioned from one status to another. We use VolatileTime in place of metav1.Time to exclude this from creating equality.Semantic differences (all other things held constant).
`message`	`string`	A human readable message indicating details about the transition.
`reason`	`string`	The reason for the condition's last transition.
`severity`	`string`	Severity with which to treat failures of this type of condition. When this is not specified, it defaults to Error.
`status`	`string`	Status of the condition, one of True, False, Unknown.
`type`	`string`	Type of condition.

API Endpoints

The following API endpoints are available:

- `/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks`
  - `DELETE` : delete collection of ApprovalTask
  - `GET` : list objects of kind ApprovalTask
  - `POST` : create a new ApprovalTask
- `/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks/{name}`
  - `DELETE` : delete the specified ApprovalTask
  - `GET` : read the specified ApprovalTask
  - `PATCH` : partially update the specified ApprovalTask
  - `PUT` : replace the specified ApprovalTask
- `/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks/{name}/status`
  - `GET` : read status of the specified ApprovalTask
  - `PATCH` : partially update status of the specified ApprovalTask
  - `PUT` : replace status of the specified ApprovalTask

/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks

HTTP method

`DELETE`

Description

delete collection of ApprovalTask

HTTP responses

HTTP code	Response body
200 - OK	`Status` schema

HTTP code	Response body
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind ApprovalTask

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTaskList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a

Parameter	Type	Description
		warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ApprovalTask` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
201 - Created	`ApprovalTask` schema
202 - Accepted	`ApprovalTask` schema
401 - Unauthorized	Empty

/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks/{name}

HTTP method

`DELETE`

Description

delete the specified ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

HTTP responses

HTTP code	Response body
200 - OK	`Status↗` schema
202 - Accepted	`Status↗` schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

read the specified ApprovalTask

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ApprovalTask` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
201 - Created	`ApprovalTask` schema
401 - Unauthorized	Empty

/apis/openshiftpipelines.org/v1alpha1/namespaces/{namespace}/approvaltasks/{name}/status

HTTP method

`GET`

Description

read status of the specified ApprovalTask

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further

Parameter	Type	Description
		processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ApprovalTask

Query parameters

Parameter	Type	Description
`dryRun`	`string`	When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed
`fieldValidation`	`string`	fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are: - Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23. - Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+ - Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

Body parameters

Parameter	Type	Description
`body`	`ApprovalTask` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ApprovalTask` schema
201 - Created	`ApprovalTask` schema

HTTP code	Response body
401 - Unauthorized	Empty