Connector APIs

Connector [accesspolicies.alauda.io/v1alpha1]

`connectors.alauda.io` group

AccessPolicy defines the access strategy for Connectors in a namespace. It specifies which Connectors are covered and what permissions are granted, either automatically (defaultPermission) or after approval checks pass (checkGrantedPermission).

`v1alpha1` version

▼ **spec** `object`

AccessPolicySpec defines the desired state of AccessPolicy.

▼ **checkGrantedPermission** `object`

CheckGrantedPermission defines permissions granted only after approval checks pass.

▼ **spec** `object` required

Spec contains the check rules and the permissions to grant after all checks pass.

▼ **checks** `[]object` required

CheckRule defines a check rule that must pass for a permission to be granted. it contains either a reference to a CheckRuleSpec stored in a ConfigMap or the CheckRuleSpec itself. you can specify either Ref or Spec, but not both.

▼ **name** `string` required

Name is the identifier of this check rule, referenced in AccessRequest status.

▼ **ref** `object`

Ref is a reference to a CheckRuleSpec stored in a ConfigMap.

▼ **configMap** `object` required

ConfigMap references the ConfigMap containing the CheckRuleSpec.

▼ **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** `object`

Spec contains the check rule specification.

▼ **selector** `object` required

Selector specifies how to find the Check Duck Type resource.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ objectRef `object` required

ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

▼ apiVersion `string`

API version of the referent.

▼ fieldPath `string`

If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

▼ kind `string`

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

▼ **name** `string`

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

▼ **resourceVersion** `string`

Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency

▼ **uid** `string`

UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids

▼ **state** `object`

State configures how the check result is computed. If empty, the default duck-type field status.state is used.

▼ **rego** `string`

Rego is an OPA Rego script (package "approval") that receives the full check resource as input and must output status = {"state": "approved|rejected|pending|passed"}. If empty, the default duck-type field status.state is used.

▼ **roleTemplate** `object` required

RoleTemplate defines the rules for the generated Role.

▼ **ref** `object`

Ref specifies a reference to a RoleTemplate

▼ **configMap** `object`

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

▼ **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 ↗

▼ **connector** `object`

Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **names** `[]string`

Names is an explicit list of resource names to match.

▼ **defaultPermission** `object`

DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

▼ **bindingTemplate** `object` required

BindingTemplate defines the subjects for the generated RoleBinding.

▼ **serviceAccounts** `[]object`

ServiceAccountTemplate defines a template for binding ServiceAccounts. it extends rbacv1.Subject with dynamic label-based selectors.

▼ **names** `[]string`

Names is the list of service account names to bind.

▼ **namespaceSelector** `object`

NamespaceSelector selects Namespaces by label and/or name.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **names** `[]string`

Names is an explicit list of resource names to match.

▼ **roleTemplate** `object` required

RoleTemplate defines the rules to include in the generated Role.

▼ **ref** `object`

Ref specifies a reference to a RoleTemplate

▼ **configMap** `object`

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

▼ **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 ↗

▼ **status** `object`

AccessPolicyStatus defines the observed state of AccessPolicy.

▼ **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 `[]object`

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 ↗

▼ 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` required

Status of the condition, one of True, False, Unknown.

▼ type `string` required

Type of condition.

▼ **matchedConnectors** `[]object`

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

▼ **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 ↗

▼ **observedGeneration** `integer`

ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

Connector [accessrequests.alauda.io/v1alpha1]

`connectors.alauda.io` group

AccessRequest represents a subject's access application for a specific Connector, scoped to the lifecycle of a context object (Pod). It tracks matched AccessPolicies, approval check states, and authorization status via conditions.

`v1alpha1` version

▼ **spec** `object`

AccessRequestSpec defines the desired state of AccessRequest.

▼ **connectorRef** `object` required

ConnectorRef references the target Connector in the same namespace. Only Name is required; Namespace is always the same as the AccessRequest.

▼ **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 ↗

▼ **context** `object` required

Context provides lifecycle context for this request. Currently only Kind=Pod is supported.

▼ **objectRef** `object` required

ObjectRef points to the lifecycle object (e.g., a Pod). Currently only Kind=Pod is supported.

▼ **apiVersion** `string`

API version of the referent.

▼ **fieldPath** `string`

If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

▼ **kind** `string`

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ **name** `string`

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗

▼ **resourceVersion** `string`

Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗

▼ **uid** `string`

UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

▼ **subject** `object` required

Subject is the identity requesting access (typically a ServiceAccount).

▼ **apiGroup** `string`

APIGroup holds the API group of the referenced subject. Defaults to "" for ServiceAccount subjects. Defaults to "rbac.authorization.k8s.io" for User and Group subjects.

▼ **kind** `string` required

Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount". If the Authorizer does not recognized the kind value, the Authorizer should report an error.

▼ **name** `string` required

Name of the object being referenced.

▼ **namespace** `string`

Namespace of the referenced object. If the object kind is non-namespace, such as "User" or "Group", and this value is not empty the Authorizer should report an error.

▼ **status** `object`

AccessRequestStatus records the observed state of AccessRequest.

▼ **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** `[]object`

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 ↗

▼ **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` required

Status of the condition, one of True, False, Unknown.

▼ **type** `string` required

Type of condition.

▼ **observedGeneration** `integer`

ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

▼ **policies** `[]object`

AccessPolicyMatchedStatus records a matched AccessPolicy and its check results.

▼ **matchedChecks** `[]object`

MatchedCheck records one matched Check Duck Type resource instance.

▼ **condition** `object` required

Condition records the computed approval condition of this check.

▼ **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` required

Status of the condition, one of True, False, Unknown.

▼ **type** `string` required

Type of condition.

▼ name `string` required

Name matches CheckRule.name in the AccessPolicy.

▼ ref `object` required

Ref identifies the matched Check Duck Type resource instance.

▼ apiVersion `string`

API version of the referent.

▼ fieldPath `string`

If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

▼ kind `string`

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ **name** `string`

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗

▼ **resourceVersion** `string`

Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗

▼ **uid** `string`

UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

▼ **name** `string` required

Name is the AccessPolicy name, used as the list map key.

▼ **permissionSync** `object`

PermissionSync records policy-level permission synchronization condition.

▼ **lastTransitionTime** `string` required

lastTransitionTime is the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.

▼ **message** `string` required

message is a human readable message indicating details about the transition. This may be an empty string.

▼ **observedGeneration** `integer`

observedGeneration represents the .metadata.generation that the condition was set based upon. For instance, if .metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.

▼ **reason** `string` required

reason contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string. This field may not be empty.

▼ **status** `string` required

status of the condition, one of True, False, Unknown.

▼ **type** `string` required

type of condition in CamelCase or in foo.example.com/CamelCase.

▼ **policySpec** `object` required

PolicySpec is the full AccessPolicy spec snapshot at match time.

▼ **checkGrantedPermission** `object`

CheckGrantedPermission defines permissions granted only after approval checks pass.

▼ **spec** `object` required

Spec contains the check rules and the permissions to grant after all checks pass.

▼ **checks** `[]object` required

CheckRule defines a check rule that must pass for a permission to be granted. it contains either a reference to a CheckRuleSpec stored in a ConfigMap or the CheckRuleSpec itself. you can specify either Ref or Spec, but not both.

▼ **name** `string` required

Name is the identifier of this check rule, referenced in AccessRequest status.

▼ **ref** `object`

Ref is a reference to a CheckRuleSpec stored in a ConfigMap.

▼ **configMap** `object` required

ConfigMap references the ConfigMap containing the CheckRuleSpec.

▼ **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** `object`

Spec contains the check rule specification.

▼ **selector** `object` required

Selector specifies how to find the Check Duck Type resource.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string`

required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **objectRef** `object`

required

ObjectRef specifies the reference to the object to check against. kind and apiVersion are required to distinguish different duck types

▼ **apiVersion** `string`

API version of the referent.

▼ **fieldPath** `string`

If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

▼ **kind** `string`

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ **name** `string`

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗

▼ **resourceVersion** `string`

Specific resourceVersion to which this reference is made, if any. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗

▼ **uid** `string`

UID of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

▼ **state** `object`

State configures how the check result is computed. If empty, the default duck-type field status.state is used.

▼ **rego** `string`

Rego is an OPA Rego script (package "approval") that receives the full check resource as input and must output status = {"state": "approved|rejected|pending|passe d"}. If empty, the default duck-type field status.state is used.

▼ **roleTemplate** `object` required

RoleTemplate defines the rules for the generated Role.

▼ **ref** `object`

Ref specifies a reference to a RoleTemplate

▼ **configMap** `object`

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

▼ **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 ↗

▼ connector `object`

Connector specifies which Connectors this policy applies to. If empty, the policy applies to all Connectors in the namespace.

▼ matchExpressions `[]object`

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

▼ key `string` required

key is the label key that the selector applies to.

▼ operator `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ values `[]string`

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.

▼ 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.

▼ **names** `[]string`

Names is an explicit list of resource names to match.

▼ **defaultPermission** `object`

DefaultPermission defines the Role and RoleBinding automatically granted without any approval check.

▼ **bindingTemplate** `object` required

BindingTemplate defines the subjects for the generated RoleBinding.

▼ **serviceAccounts** `[]object`

ServiceAccountTemplate defines a template for binding ServiceAccounts. it extends rbacv1.Subject with dynamic label-based selectors.

▼ **names** `[]string`

Names is the list of service account names to bind.

▼ **namespaceSelector** `object`

NamespaceSelector selects Namespaces by label and/or name.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **names** `[]string`

Names is an explicit list of resource names to match.

▼ **roleTemplate** `object` required

RoleTemplate defines the rules to include in the generated Role.

▼ **ref** `object`

Ref specifies a reference to a RoleTemplate

▼ **configMap** `object`

ConfigMap specifies a local reference to a ConfigMap whose data["rules"] contains the YAML-encoded list of rbacv1.PolicyRule entries. Only ConfigMaps in the connectors system namespace are supported.

▼ **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 ↗

Connector [connectors.alauda.io/v1alpha1]

`connectors.alauda.io` group

Connector is the Schema for the connectors API

`v1alpha1` version

▼ **spec** `object`

ConnectorSpec defines the desired state of Connector

▼ **address** `string`

Address is connector address

▼ **auth** `object`

Auth represents authenticate data of current connector

▼ **name** `string`

Name represent auth name that configured in ConnectorClass.spec.auth.types[].name

▼ **params** `[]object`

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

▼ **name** `string` required

▼ **value** required

▼ **secretRef** `object`

SecretRef secret reference when doing authentication

▼ **apiVersion** `string`

API version of the referent.

▼ **fieldPath** `string`

If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as desiredState.manifest.containers[2]. For example, if the object reference is to a container within a pod, this would take on a value like: "spec.containers{name}" (where "name" refers to the name of the container that triggered the event) or if no container name is specified "spec.containers[2]" (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

▼ **kind** `string`

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ **name** `string`

Name of the referent. More info:
https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

▼ namespace `string`

Namespace of the referent. More info:
https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗

▼ resourceVersion `string`

Specific resourceVersion to which this reference is made, if any. More info:
https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗

▼ uid `string`

UID of the referent. More info:
https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids ↗

▼ connectorClassName `string`

ConnectorClassName of current connector

▼ params `[]object`

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

▼ name `string` required

▼ **value** required

▼ **status** `object`

ConnectorStatus defines the observed state of Connector

▼ **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.

▼ **api** `object`

API contains the status information for the connector's api

▼ **path** `string`

Path provides the path for the connector API. it is the path of the connector API. it is used to construct the api path for the connector

▼ **conditions** `[]object`

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 ↗

▼ **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` required

Status of the condition, one of True, False, Unknown.

▼ **type** `string` required

Type of condition.

▼ **observedGeneration** `integer`

ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

▼ **proxy** `object`

Proxy contains the status information for the connector's proxy

▼ **httpAddress** `object`

HTTPAddress provides the addressable HTTP endpoint for the connector proxy.

▼ **CACerts** `string`

CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468.

▼ **audience** `string`

Audience is the OIDC audience for this address.

▼ **name** `string`

Name is the name of the address.

▼ **url** `string`

ConnectorClass [connectors.alauda.io/v1alpha1]

`connectors.alauda.io` group

ConnectorClass is the Schema for the connectorclasses API

`v1alpha1` version

▼ **spec** `object`

Spec defines the desired state of ConnectorClass

▼ **address** `object`

Address indicates address param constraints for this ConnectorClass of connectors we only support string param type

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **name** `string` required

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.

▼ **api** `object`

API defines connectorclass plugin api address and openapi specification `api.ref` can be address of plugin api, it should be a kubernetes svc `api.uri` can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref. `api.CACerts` and `api.audience` is not implemented now

▼ **CACerts** `string`

CACerts are Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468 ↗. If set, these CAs are appended to the set of

CAs provided by the Addressable target, if any.

▼ **audience** `string`

Audience is the OIDC audience. This need only be set, if the target is not an Addressable and thus the Audience can't be received from the Addressable itself. In case the Addressable specifies an Audience too, the Destinations Audience takes preference.

▼ **openapi**

OpenAPI defines the openapi specification for the connectorclass api This OpenAPI specification can describe an API customized for the ConnectorClass, or it can describe the original tool's API exposed via the ConnectorClass's proxy. if client request path is not in the OpenAPI specification, it will just forward the request to proxy of the connectorclass.

▼ **permissions** `object`

Permissions defines the request matching rules for different operations on the connectorclass api

▼ **rego** `string`

Rego contains the Rego policy script that maps API requests to permission verbs. The script evaluates incoming requests and determines the required permission level.

Requirements:

1. Must be defined under the 'permissions' package
2. Must produce a 'result' object matching the PermissionMappingResult structure

Input Variables:

- request.path (string): Request path

- request.method (string): HTTP method (GET, POST, PUT, DELETE, etc.)
- request.headers (map<string, list>): Request headers
- request.query (map<string, list>): Request query parameters
- request.body (dynamic): Parsed JSON body (available when Content-Type is application/json)

Output:

- result.verb (string): Permission verb - "read", "write", "delete", or "" (default to http verb mapping)

Example: package permissions import future.keywords.if

result = { "verb": "read" } if { startswith(input.request.path, "/search") input.request.method == "POST" } else = { "verb": "" }

▼ ref `object`

Ref points to an Addressable.

▼ address `string`

Address points to a specific Address Name.

▼ apiVersion `string`

API version of the referent.

▼ group `string`

Group of the API, without the version of the group. This can be used as an alternative to the APIVersion, and then resolved using ResolveGroup. Note: This API is EXPERIMENTAL and might break anytime. For more details: https://github.com/knative/eventing/issues/5086

▼ **kind** `string` required

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ **name** `string` required

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ ↗ This is optional field, it gets defaulted to the object holding it if left out.

▼ **uri** `string`

URI can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref.

▼ **auth** `object`

Auth indicates authentication constraints for this ConnectorClass of connectors

▼ **types** `[]object`

ConnectorClassAuthType represent the authentication types supported by connectors of the current connectorclass type

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **description** `string`

Description is the description of the AuthType

▼ **displayName** `string`

DisplayName is the human readable name of the AuthType

▼ **generator** `object`

Generator specifies how to generate authentication data dynamically. Can be used to implement custom authentication logic.

▼ **rego** `string`

Rego contains the Rego policy script that will be evaluated to generate authentication data. The script must define an 'auth' object that matches the following rules:

1. Define its rules under the 'proxy' package
2. Produce an 'auth' object containing AuthInjection structure.

▼ **name** `string` required

Name of the AuthType Must be unique within the ConnectorClass.

▼ **optional** `boolean`

Optional indicates whether the authentication information is optional for this ConnectorClass of connectors the default value is false

▼ **params** `[]object`

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **name** `string` required

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.

▼ **secretType** `string`

SecretType represents the secret type of the current authentication information follow k8s secret type definition. eg.kubernetes.io/basic-auth, kubernetes.io/ssh-auth, kubernetes.io/opaque

▼ **authProbes** `[]object`

ConnectorClassAuthProbe represents network the detection configuration

▼ **authName** `string` required

AuthName corresponds to `spec.auth.types[].name`, indicating the way to check for the corresponding authentication type

▼ **params** `[]object`

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **name** `string` required

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.

▼ probe `object`

Probe represents current detection configuration

▼ api `object`

API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's `spec.api` configuration. If `spec.api` is not configured or is invalid, no API detection will be performed. The `Path` field within `APIProbeAction` specifies a relative path that is appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`).

▼ path `string`

Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`) when execute the probe

▼ http `object`

Http represents the network detection using the http get method

▼ disableRedirect `boolean`

DisableRedirect indicates whether the probe should follow redirects, default is false

▼ host `string`

Host represents the tool address for the current detection. usually, it would be empty, it will use the `spec.address` of connector

▼ **httpHeaders** `[]object`

HTTPHeader describes a custom header to be used in HTTP probes

▼ **name** `string` required

The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.

▼ **value** `string` required

The header field value

▼ **method** `string`

Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET

▼ **path** `string` required

Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from `spec.address` of connector when execute the probe

▼ **response** `object`

Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.

▼ **cel** `string`

CEL contains a CEL (Common Expression Language) expression for response validation. The expression must evaluate to a boolean value. Available variables:

- response.statusCode (int): HTTP status code
- response.headers (map<string, list>): Response headers
- response.body (dynamic): Parsed JSON body (if Content-Type is application/json and body is valid JSON)
- response.bodyString (string): Response body as string

Example expressions:

- response.statusCode == 200 && response.body.status == 'healthy'
- response.body.uptime > 60 && response.body.errors.size() == 0
- response.body.contains('success') && !response.body.contains('error')
- response.headers['content-type'][0].startsWith('application/json')

CEL is recommended for simple to medium complexity validations:

- Intuitive syntax similar to JavaScript/C/Python
- Better performance for simple expressions
- Native support in Kubernetes ecosystem
- Lower learning curve

When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).

▼ scheme `string`

Scheme to use for connecting to the host. If empty:

- When Host is empty or matches the connector's address, the scheme from the connector's address is used.
- Otherwise, defaults to http. If specified, this value will be used regardless of Host or connector's address.

▼ configurations `[]object`

ConnectorClassConfiguration defines connectorclass configuration

▼ annotations `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata. They are not queryable and should be preserved when modifying objects. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations ↗

▼ data `object`

Data contains the configuration data. Each key must consist of alphanumeric characters, '-', '_' or '.'.

▼ name `string`

Name of the configuration

▼ livenessProbe `object`

LivenessProbe defines liveness probe for this ConnectorClass of connectors

▼ api `object`

API defines a network detection probe that uses the ConnectorClass API. The API endpoint for the probe is resolved from the ConnectorClass's `spec.api` configuration. If `spec.api` is not configured or is invalid, no API detection will be performed. The `Path` field within `APIProbeAction` specifies a relative path that is appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`).

▼ path `string`

Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from `spec.api` (e.g., `spec.api.uri`) when execute the probe

▼ http `object`

Http represents the network detection using the http get method

▼ disableRedirect `boolean`

DisableRedirect indicates whether the probe should follow redirects, default is false

▼ host `string`

Host represents the tool address for the current detection. usually, it would be empty, it will use the `spec.address` of connector

▼ httpHeaders `[]object`

HTTPHeader describes a custom header to be used in HTTP probes

▼ **name** `string` required

The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.

▼ **value** `string` required

The header field value

▼ **method** `string`

Method represents the HTTP method to use for the probe, support method: GET, POST. default is GET

▼ **path** `string` required

Path represents the API address accessed during the current detection it is relative path, it will be appended to the host of URI resolved from `spec.address` of connector when execute the probe

▼ **response** `object`

Response defines the expected response validation rules. If not specified, the probe is considered successful if the request completes without error and returns a 2xx status code.

▼ **cel** `string`

CEL contains a CEL (Common Expression Language) expression for response validation. The expression must evaluate to a boolean value. Available variables:

- response.statusCode (int): HTTP status code
- response.headers (map<string, list>): Response headers

- response.body (dynamic): Parsed JSON body (if Content-Type is application/json and body is valid JSON)
- response.bodyString (string): Response body as string

Example expressions:

- response.statusCode == 200 && response.body.status == 'healthy'
- response.body.uptime > 60 && response.body.errors.size() == 0
- response.body.contains('success') && !response.body.contains('error')
- response.headers['content-type'][0].startsWith('application/json')

CEL is recommended for simple to medium complexity validations:

- Intuitive syntax similar to JavaScript/C/Python
- Better performance for simple expressions
- Native support in Kubernetes ecosystem
- Lower learning curve

When both CEL and built-in rules (Contains, Regex, JSONPath) are specified, all rules must pass (AND logic).

▼ **scheme** `string`

Scheme to use for connecting to the host. If empty:

- When Host is empty or matches the connector's address, the scheme from the connector's address is used.
- Otherwise, defaults to http. If specified, this value will be used regardless of Host or connector's address.

▼ **params** `[]object`

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **name** `string` required

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.

▼ **proxy** `object`

Proxy defines the proxy configuration for this ConnectorClass. Specifies how network traffic should be routed through a proxy server.

▼ **CACerts** `string`

CACerts are Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468↗. If set, these CAs are appended to the set of CAs provided by the Addressable target, if any.

▼ **audience** `string`

Audience is the OIDC audience. This need only be set, if the target is not an Addressable and thus the Audience can't be received from the Addressable itself. In case the Addressable specifies an Audience too, the Destinations Audience takes preference.

▼ **authExtractor** `object`

AuthExtractor specifies the method used to extract authentication data from incoming requests. The AuthExtractor is responsible for extracting the token required to perform proxy permission validation.

▼ **rego** `string`

Rego contains the Rego policy script that will be evaluated to extract proxy authentication data from the request. The script must define an 'auth' object that matches the following rules:

1. Define its rules under the 'proxy' package

2. Produce an 'auth' object has structure same as ProxyAuthExtractorResult. Example: package proxy auth = { "token": input.request.headers["Private-Token"][0] }

▼ ref `object`

Ref points to an Addressable.

▼ address `string`

Address points to a specific Address Name.

▼ apiVersion `string`

API version of the referent.

▼ group `string`

Group of the API, without the version of the group. This can be used as an alternative to the APIVersion, and then resolved using ResolveGroup. Note: This API is EXPERIMENTAL and might break anytime. For more details: https://github.com/knative/eventing/issues/5086

▼ kind `string` required

Kind of the referent. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

▼ name `string` required

Name of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names

▼ **namespace** `string`

Namespace of the referent. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/ This is optional field, it gets defaulted to the object holding it if left out.

▼ **uri** `string`

URI can be an absolute URL(non-empty scheme and non-empty host) pointing to the target or a relative URI. Relative URIs will be resolved using the base URI retrieved from Ref.

▼ **status** `object`

Status defines the actual state of ConnectorClass

▼ **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.

▼ **api** `object`

API represents status of connectorclass api it will resolved based on `spec.api` if `spec.api` is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any api service.

▼ **address** `object`

Address is a single Addressable address.

▼ **CACerts** `string`

CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468.

▼ **audience** `string`

Audience is the OIDC audience for this address.

▼ **name** `string`

Name is the name of the address.

▼ **url** `string`

▼ **conditions** `[]object`

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

▼ **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` required

Status of the condition, one of True, False, Unknown.

▼ **type** `string` required

Type of condition.

▼ **observedGeneration** `integer`

ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

▼ **proxy** `object`

Proxy represents status of connectorclass proxy it will resolved based on `spec.proxy` if `spec.proxy` is empty or invalid, it will not be set if current field is empty, the connectorclass cannot provides any proxy service.

▼ **httpAddress** `object`

HttpAddress is a single Addressable address.

▼ CACerts `string`

CACerts is the Certification Authority (CA) certificates in PEM format according to https://www.rfc-editor.org/rfc/rfc7468↗.

▼ audience `string`

Audience is the OIDC audience for this address.

▼ name `string`

Name is the name of the address.

▼ url `string`

Connector [resourceinterfaces.alauda.io/v1alpha1]

`connectors.alauda.io` group

ResourceInterface is the Schema for the resourceinterfaces API ResourceInterface defines a reusable template for creating PipelineIntegrations with specific parameters and outputs. It supports inheritance through the extends field and provides a way to standardize resource definitions across different connector implementations.

`v1alpha1` version

▼ **spec** `object`

Spec defines the desired state of ResourceInterface

▼ **attributes** `[]object`

ResourceInterfaceAttributeSpec defines attribute specification for ResourceInterface

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **dependsOn** `[]string`

DependsOn defines the dependencies of this parameter. it could be the name of other parameters or connectors

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **expression** `string`

Expression defines a custom expression to compute the value. The expression can access connector and params of current PipelineIntegration.

▼ **name** `string` required

Name declares the name by which a parameter is referenced.

▼ **parameterize** `object`

Parameterize defines the parameterization configuration for this parameter.

▼ **disable** `boolean`

Disable indicates whether parameterization is disabled for this parameter/workspace.

▼ **name** `string`

Name is the default parameter name when this parameter/workspace is parameterized during PipelineResource construction.

▼ **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.

▼ **configurations** `object`

Configurations defines the additional configuration for this ResourceInterface, which will be passed to the PipelineIntegration instance as annotations.

▼ **params** `[]object`

ParamSpec defines arbitrary parameters needed beyond typed inputs (such as resources).

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata.

▼ **default**

Default is the value a parameter takes if no input value is supplied.

▼ **description** `string`

Description is a user-facing description of the parameter that may be used to populate a UI.

▼ **enum** `[]string`

Enum declares a set of allowed param input values. If Enum is not set, no input validation is performed for the param.

▼ **name** `string` required

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.

▼ **workspaces** `[]object`

ResourceInterfaceWorkspaceSpec defines a workspace specification for ResourceInterface

▼ **annotations** `object`

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata. More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations ↗

▼ **description** `string`

Description provides a human-readable description of the workspace.

▼ **name** `string` required

Name of the workspace.

▼ **value** `object`

Value defines the volume source for this workspace.

▼ **configMap** `object`

ConfigMap represents a configMap that should populate this workspace.

▼ **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** `[]object`

Maps a string key to a path within a volume.

▼ **key** `string` required

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` required

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 '..'.

▼ **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

▼ **csi** `object`

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

▼ **driver** `string` required

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.

▼ **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 ↗

▼ **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.

▼ **emptyDir** `object`

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.

▼ **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 ↗

▼ persistentVolumeClaim `object`

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

▼ claimName `string` required

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.

▼ projected `object`

Projected represents a projected volume that should populate this workspace.

▼ 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 `[]object`

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

▼ 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.

▼ 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".

▼ matchExpressions `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **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` required

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.

▼ **configMap** `object`

configMap information about the configMap data to project

▼ **items** `[]object`

Maps a string key to a path within a volume.

▼ **key** `string` required

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` required

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 '..'.

▼ **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

▼ **downwardAPI** `object`

downwardAPI information about the downwardAPI data to project

▼ **items** `[]object`

DownwardAPIVolumeFile represents information to create the file containing the pod field

▼ **fieldRef** `object`

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

▼ **apiVersion** `string`

Version of the schema the FieldPath is written in terms of, defaults to "v1".

▼ **fieldPath** `string`

required

Path of the field to select in the specified API version.

▼ **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

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.

▼ **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

Required: resource to select

▼ **secret** `object`

secret information about the secret data to project

▼ **items** `[]object`

Maps a string key to a path within a volume.

▼ **key** `string` required

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` required

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 '..'.

▼ **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

▼ **serviceAccountToken** `object`

serviceAccountToken is information about the serviceAccountToken data to project

▼ **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` required

path is the path relative to the mount point of the file to project the token into.

▼ **secret** `object`

Secret represents a secret that should populate this workspace.

▼ **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** `[]object`

Maps a string key to a path within a volume.

▼ **key** `string` required

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` required

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 '..'.

▼ **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

▼ **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).

▼ **volumeClaimTemplate** `object`

VolumeClaimTemplate is a template for a claim that will be created in the same namespace.

▼ **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: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds

▼ **metadata** `object`

Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata

▼ **spec** `object`

spec defines the desired characteristics of a volume requested by a pod author. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims

▼ **accessModes** `[]string`

accessModes contains the desired access modes the volume should have. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

▼ **dataSource** `object`

dataSource field can be used to specify either:

- An existing VolumeSnapshot object (snapshot.storage.k8s.io/VolumeSnapshot)
- An existing PVC (PersistentVolumeClaim) If the provisioner or an external controller can support the specified data source, it will create a new volume based on the contents of the specified data source. When the AnyVolumeDataSource feature gate is enabled, dataSource contents will be copied to dataSourceRef, and dataSourceRef contents will be copied to dataSource when dataSourceRef.namespace is not specified. If the namespace is specified, then dataSourceRef will not be copied to dataSource.

▼ **apiGroup** `string`

APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

▼ **kind** `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **dataSourceRef** `object`

dataSourceRef specifies the object from which to populate the volume with data, if a non-empty volume is desired. This may be any object from a non-empty API group (non core object) or a PersistentVolumeClaim object. When this field is specified, volume binding will only succeed if the type of the specified object matches some installed volume populator or dynamic provisioner. This field will replace the functionality of the dataSource field and as such if both fields are non-empty, they must have the same value. For backwards compatibility, when namespace isn't specified in dataSourceRef, both fields (dataSource and dataSourceRef) will be set to the same value automatically if one of them is empty and the other is non-empty. When namespace is specified in dataSourceRef, dataSource isn't set to the same value and must be empty. There are three important differences between dataSource and dataSourceRef:

- While dataSource only allows two specific types of objects, dataSourceRef allows any non-core object, as well as PersistentVolumeClaim objects.
- While dataSource ignores disallowed values (dropping them), dataSourceRef preserves all values, and generates an error if a disallowed value is specified.
- While dataSource only allows local objects, dataSourceRef allows objects in any namespaces. (Beta) Using this field requires the AnyVolumeDataSource feature gate to be enabled. (Alpha) Using the namespace field of dataSourceRef requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

▼ **apiGroup** `string`

APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

▼ **kind** `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **namespace** `string`

Namespace is the namespace of resource being referenced Note that when a namespace is specified, a gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. (Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

▼ resources `object`

resources represents the minimum resources the volume should have. If RecoverVolumeExpansionFailure feature is enabled users are allowed to specify resource requirements that are lower than previous value but must still be higher than capacity recorded in the status field of the claim. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#resources↗

▼ 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/

▼ **selector** `object`

selector is a label query over volumes to consider for binding.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values** `[]string`

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.

▼ **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.

▼ **storageClassName** `string`

storageClassName is the name of the StorageClass required by the claim. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#class-1 ↗

▼ **volumeAttributesClassName** `string`

volumeAttributesClassName may be used to set the VolumeAttributesClass used by this claim. If specified, the CSI driver will create or update the volume with the attributes defined in the corresponding VolumeAttributesClass. This has a different purpose than storageClassName, it can be changed after the claim is created. An empty string value means that no VolumeAttributesClass will be applied to the claim but it's not allowed to reset this field to empty string once it is set. If unspecified and the PersistentVolumeClaim is unbound, the default VolumeAttributesClass will be set by the persistentvolume controller if it exists. If the resource referred to by volumeAttributesClass does not exist, this PersistentVolumeClaim will be set to a Pending state, as reflected by the modifyVolumeStatus field, until such as a resource exists. More info: https://kubernetes.io/docs/concepts/storage/volumeattributes-classes/ (Beta) Using this field requires the VolumeAttributesClass feature gate to be enabled (off by default).

▼ **volumeMode** `string`

volumeMode defines what type of volume is required by the claim. Value of Filesystem is implied when not included in claim spec.

▼ **volumeName** `string`

volumeName is the binding reference to the PersistentVolume backing this claim.

▼ **status** `object`

status represents the current information/status of a persistent volume claim. Read-only. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims

▼ **accessModes** `[]string`

accessModes contains the actual access modes the volume backing the PVC has. More info: https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1

▼ **allocatedResourceStatuses** `object`

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Unprefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

ClaimResourceStatus can be in any of following states: - ControllerResizeInProgress: State set when resize controller starts resizing the volume in control-plane. - ControllerResizeFailed: State set when resize has failed in resize controller with a terminal error. - NodeResizePending: State set when resize controller has finished resizing the volume but further resizing of volume is needed on the node. - NodeResizeInProgress: State set when kubelet starts resizing the volume. - NodeResizeFailed: State set when resizing has failed in kubelet with a terminal error. Transient errors don't set NodeResizeFailed. For example: if expanding a PVC for more capacity - this field can be one of the following states: - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeFailed" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizePending" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeInProgress" - pvc.status.allocatedResourceStatus['storage'] = "NodeResizeFailed" When this field is not set, it means that no resize operation is in progress for the given PVC.

A controller that receives PVC update with previously unknown resourceName or ClaimResourceStatus should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC.

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ allocatedResources `object`

allocatedResources tracks the resources allocated to a PVC including its capacity. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

Capacity reported here may be larger than the actual capacity when a volume expansion operation is requested. For storage quota, the larger value from allocatedResources and PVC.spec.resources is used. If allocatedResources is not set, PVC.spec.resources alone is used for quota calculation. If a volume expansion capacity request is lowered, allocatedResources is only lowered if there are no expansion operations in progress and if the actual volume capacity is equal or lower than the requested capacity.

A controller that receives PVC update with previously unknown resourceName should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC.

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ capacity `object`

capacity represents the actual resources of the underlying volume.

▼ **conditions** `[]object`

PersistentVolumeClaimCondition contains details about state of pvc

▼ **lastProbeTime** `string`

lastProbeTime is the time we probed the condition.

▼ **lastTransitionTime** `string`

lastTransitionTime is the time the condition transitioned from one status to another.

▼ **message** `string`

message is the human-readable message indicating details about last transition.

▼ **reason** `string`

reason is a unique, this should be a short, machine understandable string that gives the reason for condition's last transition. If it reports "Resizing" that means the underlying persistent volume is being resized.

▼ **status** `string` required

Status is the status of the condition. Can be True, False, Unknown. More info:

https://kubernetes.io/docs/reference/kubernetes-api/config-and-storage-resources/persistent-volume-claim-v1/#:~:text=state%20of%20pvc-,conditions.status,-(string)%2C%20required ↗

▼ **type** `string` required

Type is the type of the condition. More info: https://kubernetes.io/docs/reference/kubernetes-api/config-and-storage-resources/persistent-volume-claim-v1/#:~:text=set%20to%20%27ResizeStarted%27.-,PersistentVolumeClaimCondition,-contains%20details%20about ↗

▼ **currentVolumeAttributesClassName** `string`

currentVolumeAttributesClassName is the current name of the VolumeAttributesClass the PVC is using. When unset, there is no VolumeAttributeClass applied to this PersistentVolumeClaim This is a beta field and requires enabling VolumeAttributesClass feature (off by default).

▼ **modifyVolumeStatus** `object`

ModifyVolumeStatus represents the status object of ControllerModifyVolume operation. When this is unset, there is no ModifyVolume operation being attempted. This is a beta field and requires enabling VolumeAttributesClass feature (off by default).

▼ **status** `string` required

status is the status of the ControllerModifyVolume operation. It can be in any of following states:

- Pending Pending indicates that the PersistentVolumeClaim cannot be modified due to unmet requirements, such as the specified VolumeAttributesClass not existing.
- InProgress InProgress indicates that the volume is being modified.
- Infeasible Infeasible indicates that the request has been rejected as invalid by the CSI driver. To resolve the error, a valid VolumeAttributesClass needs to be specified. Note: New statuses can be added in the future. Consumers should check for unknown statuses and fail appropriately.

▼ **targetVolumeAttributesClassName** `string`

targetVolumeAttributesClassName is the name of the VolumeAttributesClass the PVC currently being reconciled

▼ **phase** `string`

phase represents the current phase of PersistentVolumeClaim.

▼ **workspaceMapping** `object`

WorkspaceMapping defines the workspace mapping configuration for this workspace.

▼ **name** `string`

Name is the default parameter name when this workspace is mapped during PipelineIntegration construction.