Operator APIs

ConnectorsConfig \[operator.cc ConnectorsCore \[operator.con Connectors

ConnectorsGitLab \[operator.cc ConnectorsGit \[operator.conne Connectors

ConnectorsJFrog \[operator.co ConnectorsK8S \[operator.conr Connectors

ConnectorsNexus \[operator.co ConnectorsNPM \[operator.con Connectors

ConnectorsPyPI \[operator.con ConnectorsSonarQube \[operator.connectors.

ConnectorsConfig [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsConfig is the Schema for the connectorsconfig API.

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`	ConnectorsConfigSpec defines the desired state of ConnectorsConfig.
`status`	`object`	ConnectorsConfigStatus defines the observed state of ConnectorsConfig.

.spec

Description

ConnectorsConfigSpec defines the desired state of ConnectorsConfig.

Type

`object`

Property	Type	Description
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.
`components`	`object`	Components maps component name (e.g. "core", "git", "oci") to its config. Component names must match registered ComponentDefinitions.

Property	Type	Description
		Components not present in the map are enabled with default settings.
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.components

Description

Components maps component name (e.g. "core", "git", "oci") to its config. Component names must match registered ComponentDefinitions. Components not present in the map are enabled with default settings.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.status

Description

ConnectorsConfigStatus defines the observed state of ConnectorsConfig.

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.
`components`	`object`	Components tracks the status of each managed component.
`conditions`	`array`	Conditions the latest available observations of a resource's current state.
`migratedFrom`	`object`	MigratedFrom records the migration source when upgrading from standalone CRs. nil if this is a fresh install (no migration occurred).
`observedGeneration`	`integer`	ObservedGeneration is the 'Generation' of the Service that was last processed by the controller.

.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.components

Description

Components tracks the status of each managed component.

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

Property	Type	Description
		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.migratedFrom

Description

MigratedFrom records the migration source when upgrading from standalone CRs. nil if this is a fresh install (no migration occurred).

Type

`object`

Required

`timestamp`

Property	Type	Description
`components`	`array`	Components lists which components were migrated and from which namespace. Set once by the bootstrap; never mutated. Use this as the audit record.
`pendingCleanup`	`array`	PendingCleanup is the working set of pre-migration standalone CRs the controller still needs to delete. Initially equal to Components; entries are pruned as cleanup converges. Empty list means all leftover standalone CRs have been handled.
`timestamp`	`string`	Timestamp when the migration was performed.

.status.migratedFrom.components

Description

Components lists which components were migrated and from which namespace. Set once by the bootstrap; never mutated. Use this as the audit record.

Type

`array`

.status.migratedFrom.components[]

Description

MigratedComponent records the origin of a migrated component.

Type

`object`

Required

`name` `originalName` `originalNamespace`

Property	Type	Description
`name`	`string`	Name is the component type (e.g. "core", "git", "oci").
`originalName`	`string`	OriginalName is the original CR name (may differ from deterministic name).
`originalNamespace`	`string`	OriginalNamespace is the namespace where the old CR was found.

.status.migratedFrom.pendingCleanup

Description

PendingCleanup is the working set of pre-migration standalone CRs the controller still needs to delete. Initially equal to Components; entries are pruned as cleanup converges. Empty list means all leftover standalone CRs have been handled.

Type

`array`

.status.migratedFrom.pendingCleanup[]

Description

MigratedComponent records the origin of a migrated component.

Type

`object`

Required

`name` `originalName` `originalNamespace`

Property	Type	Description
`name`	`string`	Name is the component type (e.g. "core", "git", "oci").
`originalName`	`string`	OriginalName is the original CR name (may differ from deterministic name).
`originalNamespace`	`string`	OriginalNamespace is the namespace where the old CR was found.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs`
  - `DELETE`: delete collection of ConnectorsConfig
  - `GET`: list objects of kind ConnectorsConfig
  - `POST`: create a new ConnectorsConfig
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs/{name}`
  - `DELETE`: delete the specified ConnectorsConfig
  - `GET`: read the specified ConnectorsConfig
  - `PATCH`: partially update the specified ConnectorsConfig

- `PUT`: replace the specified ConnectorsConfig

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs/{name}/status`
  - `GET`: read status of the specified ConnectorsConfig
  - `PATCH`: partially update status of the specified ConnectorsConfig
  - `PUT`: replace status of the specified ConnectorsConfig

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs

HTTP method

`DELETE`

Description

delete collection of ConnectorsConfig

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsConfig

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsConfig

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`	`ConnectorsConfig` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsConfig

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 ConnectorsConfig

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsConfig

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	`ConnectorsConfig` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsConfig

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`	`ConnectorsConfig` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsconfigs/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsConfig

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsConfig

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	`ConnectorsConfig` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsConfig

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`	`ConnectorsConfig` schema	`application/json` formatted

HTTP responses

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

ConnectorsCore [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsCore is the Schema for the connectorscore API.

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`	ConnectorsCoreSpec defines the desired state of ConnectorsCore.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsCoreSpec defines the desired state of ConnectorsCore.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`featureFlags`	`object`	FeatureFlags overrides feature flag values in the connectors-config ConfigMap. Keys must match the feature flag names defined in connectors-config. Values are string booleans ("true" or "false").
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-

Property	Type	Description
		version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the

Property	Type	Description
		prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.featureFlags

Description

FeatureFlags overrides feature flag values in the connectors-config ConfigMap. Keys must match the feature flag names defined in connectors-config. Values are string booleans ("true" or "false").

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on

a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.

Property	Type	Description
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores`
  - `DELETE`: delete collection of ConnectorsCore
  - `GET`: list objects of kind ConnectorsCore
  - `POST`: create a new ConnectorsCore
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores/{name}`
  - `DELETE`: delete the specified ConnectorsCore
  - `GET`: read the specified ConnectorsCore
  - `PATCH`: partially update the specified ConnectorsCore
  - `PUT`: replace the specified ConnectorsCore
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores/{name}/status`
  - `GET`: read status of the specified ConnectorsCore
  - `PATCH`: partially update status of the specified ConnectorsCore
  - `PUT`: replace status of the specified ConnectorsCore

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores

HTTP method

`DELETE`

Description

delete collection of ConnectorsCore

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 ConnectorsCore

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsCore

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`	`ConnectorsCore` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsCore

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 ConnectorsCore

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsCore

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	`ConnectorsCore` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsCore

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`	`ConnectorsCore` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorscores/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsCore

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsCore

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	`ConnectorsCore` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsCore

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`	`ConnectorsCore` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsCore` schema
201 - Created	`ConnectorsCore` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsGitHub [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsGitHub is the Schema for the ConnectorsGitHubs API.

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`	ConnectorsGitHubSpec defines the desired state of ConnectorsGitHub.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsGitHubSpec defines the desired state of ConnectorsGitHub.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs`
  - `DELETE`: delete collection of ConnectorsGitHub
  - `GET`: list objects of kind ConnectorsGitHub
  - `POST`: create a new ConnectorsGitHub
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs/{name}`
  - `DELETE`: delete the specified ConnectorsGitHub
  - `GET`: read the specified ConnectorsGitHub
  - `PATCH`: partially update the specified ConnectorsGitHub
  - `PUT`: replace the specified ConnectorsGitHub
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs/{name}/status`
  - `GET`: read status of the specified ConnectorsGitHub
  - `PATCH`: partially update status of the specified ConnectorsGitHub
  - `PUT`: replace status of the specified ConnectorsGitHub

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs

HTTP method

`DELETE`

Description

delete collection of ConnectorsGitHub

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsGitHub

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsGitHub

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`	`ConnectorsGitHub` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsGitHub

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 ConnectorsGitHub

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsGitHub

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	`ConnectorsGitHub` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsGitHub

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`	`ConnectorsGitHub` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgithubs/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsGitHub

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsGitHub

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	`ConnectorsGitHub` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsGitHub

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`	`ConnectorsGitHub` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsGitHub` schema
201 - Created	`ConnectorsGitHub` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsGitLab [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsGitLab is the Schema for the ConnectorsGitLabs API.

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`	ConnectorsGitLabSpec defines the desired state of ConnectorsGitLab.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsGitLabSpec defines the desired state of ConnectorsGitLab.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs`
  - `DELETE`: delete collection of ConnectorsGitLab
  - `GET`: list objects of kind ConnectorsGitLab
  - `POST`: create a new ConnectorsGitLab
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs/{name}`
  - `DELETE`: delete the specified ConnectorsGitLab
  - `GET`: read the specified ConnectorsGitLab
  - `PATCH`: partially update the specified ConnectorsGitLab
  - `PUT`: replace the specified ConnectorsGitLab
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs/{name}/status`
  - `GET`: read status of the specified ConnectorsGitLab
  - `PATCH`: partially update status of the specified ConnectorsGitLab
  - `PUT`: replace status of the specified ConnectorsGitLab

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs

HTTP method

`DELETE`

Description

delete collection of ConnectorsGitLab

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsGitLab

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsGitLab

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`	`ConnectorsGitLab` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsGitLab

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 ConnectorsGitLab

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsGitLab

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	`ConnectorsGitLab` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsGitLab

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`	`ConnectorsGitLab` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgitlabs/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsGitLab

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsGitLab

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	`ConnectorsGitLab` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsGitLab

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`	`ConnectorsGitLab` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsGitLab` schema
201 - Created	`ConnectorsGitLab` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsGit [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsGit is the Schema for the ConnectorsGits API.

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`	ConnectorsGitSpec defines the desired state of ConnectorsGit.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsGitSpec defines the desired state of ConnectorsGit.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits`
  - `DELETE`: delete collection of ConnectorsGit
  - `GET`: list objects of kind ConnectorsGit
  - `POST`: create a new ConnectorsGit
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits/{name}`
  - `DELETE`: delete the specified ConnectorsGit
  - `GET`: read the specified ConnectorsGit
  - `PATCH`: partially update the specified ConnectorsGit
  - `PUT`: replace the specified ConnectorsGit
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits/{name}/status`
  - `GET`: read status of the specified ConnectorsGit
  - `PATCH`: partially update status of the specified ConnectorsGit
  - `PUT`: replace status of the specified ConnectorsGit

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits

HTTP method

`DELETE`

Description

delete collection of ConnectorsGit

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsGit

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsGit

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`	`ConnectorsGit` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsGit

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 ConnectorsGit

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsGit

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	`ConnectorsGit` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsGit

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`	`ConnectorsGit` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsgits/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsGit

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsGit

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	`ConnectorsGit` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsGit

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`	`ConnectorsGit` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsGit` schema
201 - Created	`ConnectorsGit` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsHarbor [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsHarbor is the Schema for the ConnectorsHarbors API.

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`	ConnectorsHarborSpec defines the desired state of ConnectorsHarbor.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsHarborSpec defines the desired state of ConnectorsHarbor.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`expose`	`object`	Expose defines the expose configuration for the connector
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+.

Property	Type	Description
		Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.

Property	Type	Description
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.expose

Description

Expose defines the expose configuration for the connector

Type

`object`

Required

`type`

Property	Type	Description
`domain`	`string`	Domain specifies the access domain for the connector service. Required when Type is Ingress or NodePort: • For Ingress: Set to your ingress domain (e.g., "connectors.example.com") • For NodePort: Set to your cluster node IP (e.g., "192.168.1.100")

Property	Type	Description
`ingress`	`object`	Ingress configuration for external access via ingress controller. Required when Type is Ingress.
`nodePort`	`object`	NodePort configuration for external access via node port. Required when Type is NodePort.
`type`	`string`	Type specifies the exposure method for the OCI connector service. • ClusterIP: Internal cluster access only (default) • NodePort: Access via node IP and port from outside the cluster • Ingress: Access via ingress controller with domain name

.spec.expose.ingress

Description

Ingress configuration for external access via ingress controller. Required when Type is Ingress.

Type

`object`

Property	Type	Description
`ingressClassName`	`string`	IngressClassName specifies the ingress controller class to use. Leave empty to use the default ingress controller. eg: "nginx"
`secretName`	`string`	SecretName specifies the TLS secret for HTTPS access. Example: "connectors-tls-secret"

.spec.expose.nodePort

Description

NodePort configuration for external access via node port. Required when Type is NodePort.

Type

`object`

Property	Type	Description
`port`	`integer`	Port specifies the node port number for external access. Must be between 30000-32767 (Kubernetes node port range). Recommended: Use ports 30000-31000 to avoid conflicts. Example: 30080 for HTTP, 30443 for HTTPS

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the

manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy

Property	Type	Description
		value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.
`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.

.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).

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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors`
  - `DELETE`: delete collection of ConnectorsHarbor
  - `GET`: list objects of kind ConnectorsHarbor
  - `POST`: create a new ConnectorsHarbor
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors/{name}`

- `DELETE` : delete the specified ConnectorsHarbor
- `GET` : read the specified ConnectorsHarbor
- `PATCH` : partially update the specified ConnectorsHarbor
- `PUT` : replace the specified ConnectorsHarbor

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors/{name}/status`
  - `GET` : read status of the specified ConnectorsHarbor
  - `PATCH` : partially update status of the specified ConnectorsHarbor
  - `PUT` : replace status of the specified ConnectorsHarbor

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors

HTTP method

`DELETE`

Description

delete collection of ConnectorsHarbor

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsHarbor

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsHarbor

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`	`ConnectorsHarbor` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsHarbor

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 ConnectorsHarbor

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsHarbor

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	`ConnectorsHarbor` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsHarbor

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`	`ConnectorsHarbor` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsharbors/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsHarbor

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsHarbor

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	`ConnectorsHarbor` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsHarbor

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`	`ConnectorsHarbor` schema	`application/json` formatted

HTTP responses

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

ConnectorsJFrog [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsJFrog is the Schema for the ConnectorsJFrogs API.

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`	ConnectorsJFrogSpec defines the desired state of ConnectorsJFrog.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsJFrogSpec defines the desired state of ConnectorsJFrog.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs`
  - `DELETE`: delete collection of ConnectorsJFrog
  - `GET`: list objects of kind ConnectorsJFrog
  - `POST`: create a new ConnectorsJFrog
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs/{name}`
  - `DELETE`: delete the specified ConnectorsJFrog
  - `GET`: read the specified ConnectorsJFrog
  - `PATCH`: partially update the specified ConnectorsJFrog
  - `PUT`: replace the specified ConnectorsJFrog
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs/{name}/status`
  - `GET`: read status of the specified ConnectorsJFrog
  - `PATCH`: partially update status of the specified ConnectorsJFrog
  - `PUT`: replace status of the specified ConnectorsJFrog

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs

HTTP method

`DELETE`

Description

delete collection of ConnectorsJFrog

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsJFrog

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsJFrog

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`	`ConnectorsJFrog` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsJFrog

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 ConnectorsJFrog

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsJFrog

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	`ConnectorsJFrog` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsJFrog

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`	`ConnectorsJFrog` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsjfrogs/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsJFrog

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsJFrog

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	`ConnectorsJFrog` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsJFrog

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`	`ConnectorsJFrog` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsJFrog` schema
201 - Created	`ConnectorsJFrog` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsK8S [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsK8S is the Schema for the ConnectorsK8Ss API.

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`	ConnectorsK8SSpec defines the desired state of ConnectorsK8S.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsK8SSpec defines the desired state of ConnectorsK8S.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses`
  - `DELETE`: delete collection of ConnectorsK8S
  - `GET`: list objects of kind ConnectorsK8S
  - `POST`: create a new ConnectorsK8S
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses/{name}`
  - `DELETE`: delete the specified ConnectorsK8S
  - `GET`: read the specified ConnectorsK8S
  - `PATCH`: partially update the specified ConnectorsK8S
  - `PUT`: replace the specified ConnectorsK8S
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses/{name}/status`
  - `GET`: read status of the specified ConnectorsK8S
  - `PATCH`: partially update status of the specified ConnectorsK8S
  - `PUT`: replace status of the specified ConnectorsK8S

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses

HTTP method

`DELETE`

Description

delete collection of ConnectorsK8S

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsK8S

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8SList` schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new ConnectorsK8S

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`	`ConnectorsK8S` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8S` schema
201 - Created	`ConnectorsK8S` schema
202 - Accepted	`ConnectorsK8S` schema
401 - Unauthorized	Empty

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsK8S

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 ConnectorsK8S

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8S` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update the specified ConnectorsK8S

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	`ConnectorsK8S` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsK8S

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`	`ConnectorsK8S` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8S` schema
201 - Created	`ConnectorsK8S` schema
401 - Unauthorized	Empty

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsk8ses/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsK8S

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8S` schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsK8S

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	`ConnectorsK8S` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsK8S

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`	`ConnectorsK8S` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsK8S` schema
201 - Created	`ConnectorsK8S` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsMaven [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsMaven is the Schema for the ConnectorsMavens API.

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`	ConnectorsMavenSpec defines the desired state of ConnectorsMaven.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsMavenSpec defines the desired state of ConnectorsMaven.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens`
  - `DELETE`: delete collection of ConnectorsMaven
  - `GET`: list objects of kind ConnectorsMaven
  - `POST`: create a new ConnectorsMaven
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens/{name}`
  - `DELETE`: delete the specified ConnectorsMaven
  - `GET`: read the specified ConnectorsMaven
  - `PATCH`: partially update the specified ConnectorsMaven
  - `PUT`: replace the specified ConnectorsMaven
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens/{name}/status`
  - `GET`: read status of the specified ConnectorsMaven
  - `PATCH`: partially update status of the specified ConnectorsMaven
  - `PUT`: replace status of the specified ConnectorsMaven

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens

HTTP method

`DELETE`

Description

delete collection of ConnectorsMaven

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsMaven

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsMaven

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`	`ConnectorsMaven` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsMaven

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 ConnectorsMaven

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsMaven

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	`ConnectorsMaven` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsMaven

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`	`ConnectorsMaven` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsmavens/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsMaven

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsMaven

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	`ConnectorsMaven` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsMaven

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`	`ConnectorsMaven` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsMaven` schema
201 - Created	`ConnectorsMaven` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsNexus [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsNexus is the Schema for the ConnectorsNexuses API.

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`	ConnectorsNexusSpec defines the desired state of ConnectorsNexus.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsNexusSpec defines the desired state of ConnectorsNexus.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses`
  - `DELETE`: delete collection of ConnectorsNexus
  - `GET`: list objects of kind ConnectorsNexus
  - `POST`: create a new ConnectorsNexus
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses/{name}`
  - `DELETE`: delete the specified ConnectorsNexus
  - `GET`: read the specified ConnectorsNexus
  - `PATCH`: partially update the specified ConnectorsNexus
  - `PUT`: replace the specified ConnectorsNexus
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses/{name}/status`
  - `GET`: read status of the specified ConnectorsNexus
  - `PATCH`: partially update status of the specified ConnectorsNexus
  - `PUT`: replace status of the specified ConnectorsNexus

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses

HTTP method

`DELETE`

Description

delete collection of ConnectorsNexus

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsNexus

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsNexus

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`	`ConnectorsNexus` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsNexus

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 ConnectorsNexus

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsNexus

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	`ConnectorsNexus` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsNexus

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`	`ConnectorsNexus` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnexuses/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsNexus

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsNexus

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	`ConnectorsNexus` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsNexus

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`	`ConnectorsNexus` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsNexus` schema
201 - Created	`ConnectorsNexus` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsNPM [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsNPM is the Schema for the ConnectorsNPMs API.

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`	ConnectorsNPMSpec defines the desired state of ConnectorsNPM.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsNPMSpec defines the desired state of ConnectorsNPM.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms`
  - `DELETE`: delete collection of ConnectorsNPM
  - `GET`: list objects of kind ConnectorsNPM
  - `POST`: create a new ConnectorsNPM
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms/{name}`
  - `DELETE`: delete the specified ConnectorsNPM
  - `GET`: read the specified ConnectorsNPM
  - `PATCH`: partially update the specified ConnectorsNPM
  - `PUT`: replace the specified ConnectorsNPM
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms/{name}/status`
  - `GET`: read status of the specified ConnectorsNPM
  - `PATCH`: partially update status of the specified ConnectorsNPM
  - `PUT`: replace status of the specified ConnectorsNPM

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms

HTTP method

`DELETE`

Description

delete collection of ConnectorsNPM

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsNPM

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsNPM

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`	`ConnectorsNPM` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsNPM

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 ConnectorsNPM

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsNPM

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	`ConnectorsNPM` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsNPM

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`	`ConnectorsNPM` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsnpms/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsNPM

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsNPM

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	`ConnectorsNPM` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsNPM

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`	`ConnectorsNPM` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsNPM` schema
201 - Created	`ConnectorsNPM` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsOCI [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsOCI is the Schema for the ConnectorsOCIs API.

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`	ConnectorsOCISpec defines the desired state of ConnectorsOCI.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsOCISpec defines the desired state of ConnectorsOCI.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`expose`	`object`	Expose defines the expose configuration for the connector
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+.

Property	Type	Description
		Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.

Property	Type	Description
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.expose

Description

Expose defines the expose configuration for the connector

Type

`object`

Required

`type`

Property	Type	Description
`domain`	`string`	Domain specifies the access domain for the connector service. Required when Type is Ingress or NodePort: • For Ingress: Set to your ingress domain (e.g., "connectors.example.com") • For NodePort: Set to your cluster node IP (e.g., "192.168.1.100")

Property	Type	Description
`ingress`	`object`	Ingress configuration for external access via ingress controller. Required when Type is Ingress.
`nodePort`	`object`	NodePort configuration for external access via node port. Required when Type is NodePort.
`type`	`string`	Type specifies the exposure method for the OCI connector service. • ClusterIP: Internal cluster access only (default) • NodePort: Access via node IP and port from outside the cluster • Ingress: Access via ingress controller with domain name

.spec.expose.ingress

Description

Ingress configuration for external access via ingress controller. Required when Type is Ingress.

Type

`object`

Property	Type	Description
`ingressClassName`	`string`	IngressClassName specifies the ingress controller class to use. Leave empty to use the default ingress controller. eg: "nginx"
`secretName`	`string`	SecretName specifies the TLS secret for HTTPS access. Example: "connectors-tls-secret"

.spec.expose.nodePort

Description

NodePort configuration for external access via node port. Required when Type is NodePort.

Type

`object`

Property	Type	Description
`port`	`integer`	Port specifies the node port number for external access. Must be between 30000-32767 (Kubernetes node port range). Recommended: Use ports 30000-31000 to avoid conflicts. Example: 30080 for HTTP, 30443 for HTTPS

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the

manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy

Property	Type	Description
		value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.
`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.

.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).

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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis`
  - `DELETE`: delete collection of ConnectorsOCI
  - `GET`: list objects of kind ConnectorsOCI
  - `POST`: create a new ConnectorsOCI
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis/{name}`

- `DELETE`: delete the specified ConnectorsOCI
- `GET`: read the specified ConnectorsOCI
- `PATCH`: partially update the specified ConnectorsOCI
- `PUT`: replace the specified ConnectorsOCI

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis/{name}/status`
  - `GET`: read status of the specified ConnectorsOCI
  - `PATCH`: partially update status of the specified ConnectorsOCI
  - `PUT`: replace status of the specified ConnectorsOCI

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis

HTTP method

`DELETE`

Description

delete collection of ConnectorsOCI

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsOCI

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsOCI

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`	`ConnectorsOCI` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsOCI

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 ConnectorsOCI

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsOCI

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	`ConnectorsOCI` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsOCI

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`	`ConnectorsOCI` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorsocis/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsOCI

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsOCI

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	`ConnectorsOCI` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsOCI

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`	`ConnectorsOCI` schema	`application/json` formatted

HTTP responses

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

ConnectorsPyPI [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsPyPI is the Schema for the ConnectorsPyPIs API.

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`	ConnectorsPyPISpec defines the desired state of ConnectorsPyPI.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsPyPISpec defines the desired state of ConnectorsPyPI.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis`
  - `DELETE`: delete collection of ConnectorsPyPI
  - `GET`: list objects of kind ConnectorsPyPI
  - `POST`: create a new ConnectorsPyPI
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis/{name}`
  - `DELETE`: delete the specified ConnectorsPyPI
  - `GET`: read the specified ConnectorsPyPI
  - `PATCH`: partially update the specified ConnectorsPyPI
  - `PUT`: replace the specified ConnectorsPyPI
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis/{name}/status`
  - `GET`: read status of the specified ConnectorsPyPI
  - `PATCH`: partially update status of the specified ConnectorsPyPI
  - `PUT`: replace status of the specified ConnectorsPyPI

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis

HTTP method

`DELETE`

Description

delete collection of ConnectorsPyPI

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsPyPI

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsPyPI

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`	`ConnectorsPyPI` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsPyPI

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 ConnectorsPyPI

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsPyPI

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	`ConnectorsPyPI` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsPyPI

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`	`ConnectorsPyPI` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorspypis/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsPyPI

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsPyPI

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	`ConnectorsPyPI` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsPyPI

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`	`ConnectorsPyPI` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsPyPI` schema
201 - Created	`ConnectorsPyPI` schema

HTTP code	Response body
401 - Unauthorized	Empty

ConnectorsSonarQube [operator.connectors.alauda.io/v1alpha1]

Description

ConnectorsSonarQube is the Schema for the ConnectorsSonarQubes API.

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`	ConnectorsSonarQubeSpec defines the desired state of ConnectorsSonarQube.
`status`	`object`	ComponentStatus defines the observed state of Connector

.spec

Description

ConnectorsSonarQubeSpec defines the desired state of ConnectorsSonarQube.

Type

`object`

Property	Type	Description
`additionalManifests`	`string`	A means to specify the additional manifests to install, eg: http://example.com/manifests.yaml ↗
`annotations`	`object`	Annotations overrides annotations for the namespace and its template.

Property	Type	Description
`imagePullSecrets`	`array`	ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.
`installFlags`	`object`	InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own shape. Per-install-flag transformers

Property	Type	Description
		(pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.
`labels`	`object`	Labels overrides labels for the namespace and its template.
`registry`	`string`	Registry overrides the default registry for the workloads. The registry is used to replace the prefix of the image. the image reference format is /:@.
`workloads`	`array`	Workloads overrides workloads configurations such as resources and replicas.

.spec.annotations

Description

Annotations overrides annotations for the namespace and its template.

Type

`object`

.spec.imagePullSecrets

Description

ImagePullSecrets lists Secret names used to pull private images for the component workloads. Each name must reference a kubernetes.io/dockerconfigjson Secret that already exists in the namespace where the workload runs. The names are injected into every workload pod template's imagePullSecrets, merged with (and deduplicated against) any the manifest already declares. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR to add component-specific pull secrets. The effective list is the union of the cluster-wide and per-component names.

Type

`array`

.spec.imagePullSecrets[]

Type

`string`

.spec.installFlags

Description

InstallFlags supplies per-install-flag admin configuration for install-time gating. Map key is a kebab-case install-flag name; value is an arbitrary JSON value whose shape is documented per flag. Currently registered flag: pipeline-integration: "<=N" // string, semver constraint on the // ResourceInterface schema-version label. // e.g. "<=2" installs v1 and v2; skips v3+. Future flags may use other value shapes (boolean toggle, enum string, struct, …); the CRD schema does not constrain inner value structure on purpose so each flag can own its own

shape. Per-install-flag transformers (pkg/controllers/transformer/installflags/) parse the JSON value at Transform time; parse failures degrade to the maturity default for that flag (Alpha: skip all matched resources; Beta: install all). GA / Deprecated maturities are locked and ignore admin overrides entirely. This is NOT a runtime feature flag (see ConnectorsCoreSpec.FeatureFlags for that) — it acts at install/Transform time and decides whether install-time manifests get applied to the cluster, not whether a running binary's code paths are enabled. Set on ConnectorsConfig.spec to broadcast cluster-wide; set on a per-component CR (e.g. ConnectorsHarbor.spec.installFlags) to override the cluster-wide value on the matching key for THAT component only.

Type

`object`

.spec.labels

Description

Labels overrides labels for the namespace and its template.

Type

`object`

.spec.workloads

Description

Workloads overrides workloads configurations such as resources and replicas.

Type

`array`

.spec.workloads[]

Description

WorkloadOverride overrides the workload configurations.

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	Name is the name of the deployment to override.
`replicas`	`integer`	Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.
`template`		PodTemplateSpecJSON describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

.status

Description

ComponentStatus defines the observed state of Connector

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.

Property	Type	Description
`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.

.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.

API Endpoints

The following API endpoints are available:

- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes`
  - `DELETE`: delete collection of ConnectorsSonarQube
  - `GET`: list objects of kind ConnectorsSonarQube
  - `POST`: create a new ConnectorsSonarQube
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes/{name}`
  - `DELETE`: delete the specified ConnectorsSonarQube
  - `GET`: read the specified ConnectorsSonarQube
  - `PATCH`: partially update the specified ConnectorsSonarQube
  - `PUT`: replace the specified ConnectorsSonarQube
- `/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes/{name}/status`
  - `GET`: read status of the specified ConnectorsSonarQube
  - `PATCH`: partially update status of the specified ConnectorsSonarQube
  - `PUT`: replace status of the specified ConnectorsSonarQube

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes

HTTP method

`DELETE`

Description

delete collection of ConnectorsSonarQube

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind ConnectorsSonarQube

HTTP responses

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

HTTP method

`POST`

Description

create a new ConnectorsSonarQube

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`	`ConnectorsSonarQube` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes/{name}

HTTP method

`DELETE`

Description

delete the specified ConnectorsSonarQube

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 ConnectorsSonarQube

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified ConnectorsSonarQube

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	`ConnectorsSonarQube` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified ConnectorsSonarQube

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`	`ConnectorsSonarQube` schema	`application/json` formatted

HTTP responses

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

/apis/operator.connectors.alauda.io/v1alpha1/namespaces/{namespace}/connectorssonarqubes/{name}/status

HTTP method

`GET`

Description

read status of the specified ConnectorsSonarQube

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified ConnectorsSonarQube

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	`ConnectorsSonarQube` schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified ConnectorsSonarQube

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`	`ConnectorsSonarQube` schema	`application/json` formatted

HTTP responses

HTTP code	Response body
200 - OK	`ConnectorsSonarQube` schema
201 - Created	`ConnectorsSonarQube` schema

HTTP code	Response body
401 - Unauthorized	Empty