Operator APIs

Alauda AI

AmlCluster [amlclusters.aml.dev/v1alpha1]

`amlclusters.aml.dev` group

`v1alpha1` version

▼ **spec** `object`

▼ **amlExperimental** `object`

▼ **components** `object`

▼ **knativeServing** `object`

Knative-Serving component configuration. Requires KServeless operator to be installed. Requires Service Mesh (istio).

▼ **ingressGateway** `object`

IngressGateway allows to customize some parameters for the Istio Ingress Gateway that is bound to KNative-Serving.

▼ **certificate** `object`

Certificate specifies configuration of the TLS certificate securing communication for the gateway.

▼ **secretName** `string`

SecretName specifies the name of the Kubernetes Secret resource that contains a TLS certificate secure HTTP communications for the KNative network.

▼ type `string`

Type specifies if the TLS certificate should be generated automatically, or if the certificate is provided by the user. Allowed values are:

- SelfSigned: A certificate is going to be generated using an own private key.
- Provided: Pre-existence of the TLS Secret (see SecretName) with a valid certificate is assumed.
- ACPDefaultIngress: Default ingress certificate configured for ACP (ALB)

▼ domain `string` required

Domain specifies the host name for intercepting incoming requests. Most likely, you will want to use a wildcard name, like *.example.com. If you choose to generate a certificate, this is the domain used for the certificate request.

▼ managementState `string`

▼ namespace `string`

▼ values `object`

▼ kserve `object`

Kserve component configuration. Requires KServeless operator to be installed.

▼ managementState `string`

▼ namespace `string`

▼ values `object`

▼ values `object` required

▼ amlApiServer `object`

▼ tolerations `[]object`

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

▼ effect `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ key `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists;

this combination means to match all values and all keys.

▼ operator `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ tolerationSeconds `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ value `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ amlServer `object`

▼ nodeSelector `[]string`

▼ tolerations `[]object`

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

▼ effect `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ key `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

▼ operator `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ tolerationSeconds `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ value `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ amlService `object`

▼ appsBaseImage `string`

应用镜像构建的基础镜像

▼ appsGradioVersions `string`

应用镜像构建可选的 Gradio 的版本列表

▼ appsPipMirror `string`

应用镜像构建使用的 pip 镜像源

▼ appsStreamlitVersions `string`

应用镜像构建可选的 Streamlit 的版本列表

▼ gitDatasetUpdateCallbackUrl `string`

gitlab 回调 amlservice 的访问地址，用于在数据集更新之后，触发自动更新 preview 数据。

▼ imageBuildPvcWorkspaceSize `string`

镜像构建任务用于存放工作区文件的空间大小，比如存放模型文件

▼ imageBuildUsedPvcName `string`

镜像构建服务使用的pvc名称（非NFS）,需提前创建好，建议申请大一点的空间用于缓存构建镜像，比如：150G

▼ nodeSelector `[]string`

▼ notebookStorageClass `string`

自动创建的 notebook 使用的存储类

▼ tolerations `[]object`

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

▼ effect `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ key `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

▼ operator `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ tolerationSeconds `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ **value** `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ **trainingPvcShareInNamespace** `string`

训练使用的 PVC 是否在命名空间内共享，即一个命名空间仅使用一个通用的 PVC 缓存训练时的模型、数据集

▼ **trainingPvcSize** `string`

训练 PVC 大小。如果是命名空间共享的缓存PVC，建议使用较大的空间，足够命名空间所有训练任务使用

▼ **trainingPvcStorageClass** `string`

训练 PVC 使用的存储类

▼ **buildkitd** `object`

▼ **storage** `object`

▼ **emptyDir** `object`

当 type 为 emptyDir 时，需要提供以下配置

▼ **sizeLimit** `integer`

▼ **pvc** `object`

当 type 为 pvc 时，需要提供以下配置

▼ **accessMode** `string`

▼ **storageClassName** `string`

▼ **storageSize** `string`

▼ **type** `string`

存储类型，支持：emptyDir,pvc

▼ **tolerations** `[]object`

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

▼ **effect** `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ **key** `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

▼ **operator** `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ **tolerationSeconds** `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ **value** `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ **experimentalFeatures** `object`

▼ agents `boolean`

▼ datasets `boolean`

▼ imageBuilder `boolean`

▼ tuneModels `boolean`

▼ global `object`

▼ defaultNamespace `string`

默认纳管的命名空间

▼ defaultNodeSelector `[]string`

▼ defaultTolerations `[]object`

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

▼ effect `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ key `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

▼ operator `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ tolerationSeconds `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ value `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ deployFlavor `string`

DeployFlavor for single node or HA-cluster

▼ gitUserEmail `string`

后台 git push 使用的 email，仅在 git log 存储

▼ gitlabAdminTokenSecretRef `object` required

Gitlab Admin 密钥地址

▼ name `string` required

Name is the name of resource being referenced

▼ namespace `string` required

Namespace is the namespace of resource being referenced

▼ gitlabBaseUrl `string` required

Gitlab 访问地址

▼ imageRegistryPrefix `string`

harbor 项目名字

▼ images `object`

▼ labelBaseDomain `string`

▼ mysql `object`

MySQL configuration.

▼ **database** `string` required

Database to use.

▼ **host** `string` required

Host where the database server is located.

▼ **passwordSecretRef** `object`

Password reference to a secret containing the password.

▼ **name** `string` required

Name is the name of resource being referenced

▼ **namespace** `string` required

Namespace is the namespace of resource being referenced

▼ **port** `integer`

MySQL port to use, default is usually OK. (default: 3306)

▼ **user** `string` required

User to log in as.

▼ **registry** `object`

▼ **address** `string`

▼ **mysql** `object`

▼ **pvc** `object`

▼ **storageClassName** `string`

▼ **storageSize** `integer`

▼ **tolerations** `[]object`

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

▼ **effect** `string`

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

▼ **key** `string`

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

▼ **operator** `string`

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

▼ **tolerationSeconds** `integer`

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

▼ **value** `string`

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

▼ **status** `object`

▼ **components** `object`

Expose component's specific status

▼ **aml** `object`

▼ **conditions** `[]object`

Condition represents an observation of an object's state. Conditions are an extension mechanism intended to be used when the details of an

observation are not a priori known or would not apply to all instances of a given Kind.

Conditions should be added to explicitly convey properties that users and components care about rather than requiring those properties to be inferred from other observations. Once defined, the meaning of a Condition can not be changed arbitrarily - it becomes part of the API, and has the same backwards- and forwards-compatibility concerns of any other part of the API.

▼ **lastTransitionTime** `string`

▼ **message** `string`

▼ **reason** `string`

ConditionReason is intended to be a one-word, CamelCase representation of the category of cause of the current status. It is intended to be used in concise output, such as one-line kubectl get output, and in summarizing occurrences of causes.

▼ **status** `string` required

▼ **type** `string` required

ConditionType is the type of the condition and is typically a CamelCased word or short phrase.

Condition types should indicate state in the "abnormal-true" polarity. For example, if the condition indicates when a policy is invalid, the "is valid" case is probably the norm, so the condition should be called "Invalid".

▼ **deployedRelease** `object`

▼ **name** `string`

▼ **managementState** `string`

▼ **kServe** `object`

▼ **conditions** `[]object`

Condition represents an observation of an object's state. Conditions are an extension mechanism intended to be used when the details of an observation are not a priori known or would not apply to all instances of a given Kind.

Conditions should be added to explicitly convey properties that users and components care about rather than requiring those properties to be inferred from other observations. Once defined, the meaning of a Condition can not be changed arbitrarily - it becomes part of the API, and has the same backwards- and forwards-compatibility concerns of any other part of the API.

▼ **lastTransitionTime** `string`

▼ **message** `string`

▼ **reason** `string`

ConditionReason is intended to be a one-word, CamelCase representation of the category of cause of the current status. It is intended to be used in concise output, such as one-line kubectl get output, and in summarizing occurrences of causes.

▼ **status** `string` required

▼ **type** `string` required

ConditionType is the type of the condition and is typically a CamelCased word or short phrase.

Condition types should indicate state in the "abnormal-true" polarity. For example, if the condition indicates when a policy is invalid, the "is valid" case is probably the norm, so the condition should be called "Invalid".

▼ **deployedRelease** `object`

▼ **name** `string`

▼ **managementState** `string`

▼ **knativeServing** `object`

▼ **conditions** `[]object`

Condition represents an observation of an object's state. Conditions are an extension mechanism intended to be used when the details of an

observation are not a priori known or would not apply to all instances of a given Kind.

Conditions should be added to explicitly convey properties that users and components care about rather than requiring those properties to be inferred from other observations. Once defined, the meaning of a Condition can not be changed arbitrarily - it becomes part of the API, and has the same backwards- and forwards-compatibility concerns of any other part of the API.

▼ **lastTransitionTime** `string`

▼ **message** `string`

▼ **reason** `string`

ConditionReason is intended to be a one-word, CamelCase representation of the category of cause of the current status. It is intended to be used in concise output, such as one-line kubectl get output, and in summarizing occurrences of causes.

▼ **status** `string` required

▼ **type** `string` required

ConditionType is the type of the condition and is typically a CamelCased word or short phrase.

Condition types should indicate state in the "abnormal-true" polarity. For example, if the condition indicates when a policy is invalid, the "is valid" case is probably the norm, so the condition should be called "Invalid".

▼ **deployedRelease** `object`

▼ **name** `string`

▼ **managementState** `string`

▼ **conditions** `[]object`

Condition contains details for one aspect of the current state of this API Resource.

This struct is intended for direct use as an array at the field path .status.conditions. For example,

```
type FooStatus struct{
    // Represents the observations of a foo's current state.
    // Known .status.conditions.type are: "Available", "Progressing"
    // +patchMergeKey=type
    // +patchStrategy=merge
    // +listType=map
    // +listMapKey=type
    Conditions []metav1.Condition `json:"conditions,omitempty" patch


    // other fields
}
```

▼ lastTransitionTime `string` required

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

▼ message `string` required

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

▼ observedGeneration `integer`

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

▼ reason `string` required

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

▼ status `string` required

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

▼ type `string` required

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

Many .condition.type values are consistent across resources like Available, but because arbitrary conditions can be useful (see .node.status.conditions), the ability to deconflict is important. The regex it matches is (dns1123SubdomainFmt/)? (qualifiedNameFmt)

▼ lastHandledReconcileAt `string`

LastHandledReconcileAt holds the value of the most recent reconcile request value, so a change of the annotation value can be detected.

▼ observedGeneration `integer`

The observed generation of the object. When this matches the object's metadata.generation, it indicates the status is up-to-date.