MySQL-MGR APIs

MySQL MGR

MGR Backup

MGR Resto

MGR Schedule

MGR User

MySQL MGR

`middleware.alauda.io` group

Mysql is the Schema for the mysqls API

`v1` version

▼ spec `object`

MysqlSpec defines the desired state of Mysql

▼ crVersion `string`

Deprecated: use upgradeOption.CrVersion

▼ ipFamilyPrefer `string`

IPFamilyPrefer is the preferred IP family

▼ mgr `object`

MGR use to configure MySQLCluster

▼ affinity `object`

If specified, affinity will define the pod's scheduling constraints

▼ nodeAffinity `object`

Describes node affinity scheduling rules for the pod.

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

An empty preferred scheduling term matches all objects with implicit weight 0 (i.e. it's a no-op). A null preferred scheduling term matches no objects (i.e. is also a no-op).

▼ preference `object` required

A node selector term, associated with the corresponding weight.

▼ matchExpressions `[]object`

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

▼ key `string` required

The label key that the selector applies to.

▼ operator `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ values `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element,

which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ matchFields `[]object`

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

▼ key `string` required

The label key that the selector applies to.

▼ operator `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ values `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **weight** `integer` required

Weight associated with matching the corresponding nodeSelectorTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `object`

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to an update), the system may or may not try to eventually evict the pod from its node.

▼ **nodeSelectorTerms** `[]object` required

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn,

Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-

empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ podAffinity `object`

Describes pod affinity scheduling rules (e.g. co-locate this pod in the same node, zone, etc. as some other pod(s)).

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ podAffinityTerm `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ labelSelector `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ matchExpressions `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling

MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ matchExpressions `[]object`

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

▼ key `string` required

key is the label key that the selector applies to.

▼ operator `string` required

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

▼ values `[]string`

values is an array of string values. If the operator is In or NotIn, the

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches

that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The

same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ matchExpressions `[]object`

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

the key and values.

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **podAntiAffinity** `object`

Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in the same node, zone, etc. as some other pod(s)).

▼ **preferredDuringSchedulingIgnoredDuringExecution** `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ **podAffinityTerm** `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **mismatchLabelKeys** `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin`

`(value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ matchExpressions `[]object`

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

▼ key `string` required

key is the label key that the selector applies to.

▼ operator `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null

namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ matchLabels `object`

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

▼ namespaces `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ topologyKey `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ backupVolumeClaimTemplate `object`

BackupVolumeClaimTemplate allows a user to specify a volume to temporarily store the data for a backup prior to it being shipped to object storage.

▼ apiVersion `string`

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

▼ kind `string`

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ metadata `object`

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

▼ spec `object`

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

▼ accessModes `[]string`

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

▼ dataSource `object`

dataSource field can be used to specify either:

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

▼ apiGroup `string`

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

▼ kind `string` required

Kind is the type of resource being referenced

▼ name `string` required

Name is the name of resource being referenced

▼ dataSourceRef `object`

dataSourceRef specifies the object from which to populate the volume with data, if a non-empty volume is desired. This may be

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

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

▼ apiGroup `string`

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

▼ kind `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **namespace** `string`

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

▼ **resources** `object`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ **selector** `object`

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

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values

array must be empty. This array is replaced during a strategic merge patch.

▼ matchLabels `object`

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

▼ storageClassName `string`

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

▼ volumeAttributesClassName `string`

volumeAttributesClassName may be used to set the VolumeAttributesClass used by this claim. If specified, the CSI driver will create or update the volume with the attributes defined in the corresponding VolumeAttributesClass. This has a different purpose than storageClassName, it can be changed after the claim is created. An empty string value means that no VolumeAttributesClass will be applied to the claim but it's not allowed to reset this field to empty string once it is set. If unspecified and the PersistentVolumeClaim is unbound, the default VolumeAttributesClass will be set by the persistentvolume controller if it exists. If the resource referred to by volumeAttributesClass does not exist, this PersistentVolumeClaim will be set to a Pending state, as reflected by the modifyVolumeStatus field, until such as a resource exists. More

info: https://kubernetes.io/docs/concepts/storage/volume-attributes-classes/ (Beta) Using this field requires the VolumeAttributesClass feature gate to be enabled (off by default).

▼ volumeMode `string`

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

▼ volumeName `string`

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

▼ status `object`

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

▼ accessModes `[]string`

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

▼ allocatedResourceStatuses `object`

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. * Custom resources must use

implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ allocatedResources `object`

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

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ capacity `object`

capacity represents the actual resources of the underlying volume.

▼ conditions `[]object`

PersistentVolumeClaimCondition contains details about state of pvc

▼ lastProbeTime `string`

lastProbeTime is the time we probed the condition.

▼ lastTransitionTime `string`

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

▼ message `string`

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

▼ reason `string`

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

▼ status `string` required

Status is the status of the condition. Can be True, False, Unknown. More info: https://kubernetes.io/docs/reference/kubernetes-api/config-and-storage-resources/persistent-volume-claim-v1/#:~:text=state%20of%20pvc-,conditions.status,-(string)%2C%20required ↗

▼ type `string` required

Type is the type of the condition. More info: https://kubernetes.io/docs/reference/kubernetes-api/config-

and-storage-resources/persistent-volume-claim-v1/#:~:text=set%20to%20%27ResizeStarted%27.-,PersistentVolumeClaimCondition,-contains%20details%20about↗

▼ currentVolumeAttributesClassName `string`

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

▼ modifyVolumeStatus `object`

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

▼ status `string` required

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

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

future. Consumers should check for unknown statuses and fail appropriately.

▼ **targetVolumeAttributesClassName** `string`

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

▼ **phase** `string`

phase represents the current phase of PersistentVolumeClaim.

▼ **baseServerId** `integer`

BaseServerID defines the base number used to create unique server_id for MySQL instances in the cluster. Valid range 1 to 4294967286. If omitted in the manifest file (or set to 0) defaultBaseServerID value will be used.

▼ **certConfig** `object`

CertConfig allows a user to specify custom CA certificate with advanced option.

▼ **DNSNames** `[]string`

DNSNames means the dns name list of the cert.

▼ **duration** `string`

Duration means the duration of the cert.

▼ **secretName** `string`

SecretName means the secret of the cert.

▼ config `object`

Config allows a user to specify a custom configuration file for MySQL.

▼ name `string`

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

▼ configuration `string`

Mysql configuration

▼ enableStorage `boolean`

EnableStorage allow a user to enable mysql storage

▼ image `object`

Image allows a user to specify image.

▼ agentImage `string`

AgentImage means the image of the agent.

▼ routerImage `string`

RouterImage means the image of the router.

▼ **serverImage** `string`

ServerImage means the image of the server.

▼ **slowSQL** `string`

SlowSQL means the image of the slow sql.

▼ **imagePullPolicy** `string`

PullPolicy describes a policy for if/when to pull a container image

▼ **imagePullSecret** `[]object`

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

▼ **name** `string`

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

▼ **ipFamilyPolicy** `string`

IpFamilyPolicy means the IP family policy for cluster

▼ ipFamilyPrefer `string`

IpFamilyPrefer means the preferred IP family for cluster

▼ jemallocConf `string`

JemallocConf means the jemalloc configuration

▼ members `integer`

Members defines the number of MySQL instances in a cluster. InnoDB clustering supports between 1-9 members.

▼ monitor `object`

Monitor define MySQL monitor spec

▼ enable `boolean`

Enable means enable monitor

▼ exporter `object`

Exporter define MySQL exporter spec

▼ image `string`

Image means the image of the exporter.

▼ resources `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers.

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

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

▼ multiClusterReplication `object`

Config this field to support multi cluster replication for disaster recovery.

▼ bootstrapSecret `string`

BootstrapSecret means the secret of the multi cluster replication.

▼ clusterSetSeedAddr `string`

ClusterSetSeedAddr means the cluster set seed address of the multi cluster replication.

▼ enabled `boolean`

Enabled means enable multi cluster replication

▼ hostPort `integer`

HostPort means the host port of the multi cluster replication.

▼ multiMaster `boolean`

MultiMaster defines the mode of the MySQL cluster. If set to true, all instances will be R/W. If false (the default), only a single instance will be R/W and the rest will be R/O.

▼ nodeSelector `object`

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

▼ paras `object`

Paras provide paras template feature

▼ patches `object`

Patches use to apply patches to the MySQL container

▼ podPatches `object`

PodPatches is a map of pod name to patch

▼ servicePatches `object`

ServicePatches is a map of service name to patch

▼ pause `boolean`

Pause means pause the cluster

▼ podCIDR `string`

PodCIDR allows a user to specify custom CIDR.

▼ readOnlyRootFilesystem `boolean`

ReadOnlyRootFilesystem means the root filesystem is read-only

▼ **repository** `string`

Repository defines the image repository from which to pull the MySQL server image.

▼ **resources** `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers

▼ **agent** `object`

Agent describes the compute resource requirements for agent.

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ **server** `object`

Server describes the compute resource requirements for server.

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ limits `object`

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

▼ requests `object`

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

▼ rootPasswordSecret `object`

If defined, we use this secret for configuring the MYSQL_ROOT_PASSWORD If it is not set we generate a secret dynamically

▼ name `string`

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

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

▼ router `object`

Router defines the MySQL Router

▼ affinity `object`

If specified, affinity will define the pod's scheduling constraints

▼ nodeAffinity `object`

Describes node affinity scheduling rules for the pod.

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

An empty preferred scheduling term matches all objects with implicit weight 0 (i.e. it's a no-op). A null preferred scheduling term matches no objects (i.e. is also a no-op).

####### ▼ preference `object` required

A node selector term, associated with the corresponding weight.

######## ▼ matchExpressions `[]object`

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

######### ▼ key `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **weight** `integer` required

Weight associated with matching the corresponding nodeSelectorTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `object`

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g.

due to an update), the system may or may not try to eventually evict the pod from its node.

▼ **nodeSelectorTerms** `[]object` required

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values

array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ matchFields `[]object`

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

▼ key `string` required

The label key that the selector applies to.

▼ operator `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ values `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an

integer. This array is replaced during a strategic merge patch.

▼ podAffinity `object`

Describes pod affinity scheduling rules (e.g. co-locate this pod in the same node, zone, etc. as some other pod(s)).

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ podAffinityTerm `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ labelSelector `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

####### ▼ matchExpressions `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key

field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing

pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

"value". The requirements are ANDed.

▼ namespaces `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ topologyKey `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ weight `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also,

matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **mismatchLabelKeys** `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

array contains only "value". The requirements are ANDed.

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **podAntiAffinity** `object`

Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in the same node, zone, etc. as some other pod(s)).

▼ **preferredDuringSchedulingIgnoredDuringExecution** `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred

node(s)

▼ **podAffinityTerm** `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for

the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

values is an array of string values. If the operator is In

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

array contains only "value". The requirements are ANDed.

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be

set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where

co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ configuration `string`

Configuration means the configuration of the router.

▼ mysqlPasswordSecret `string`

MySQLPasswordSecret defines the secret name of MySQL user.

▼ mysqlUser `string`

MySQLUser defines the user that connect to MySQL.

▼ nodeSelector `object`

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

▼ replicas `integer`

Replicas defines the replicas of the MySQL Router.

▼ resources `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ securityContext `object`

PodSecurityContext holds pod-level security attributes and common container settings. Some fields are also present in container.securityContext. Field values of container.securityContext take precedence over field values of PodSecurityContext.

▼ appArmorProfile `object`

appArmorProfile is the AppArmor options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ localhostProfile `string`

localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".

▼ type `string` required

type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

▼ fsGroup `integer`

A special supplemental group that applies to all containers in a pod. Some volume types allow the Kubelet to change the ownership of that volume to be owned by the pod:

1. The owning GID will be the FSGroup

2. The setgid bit is set (new files created in the volume will be owned by FSGroup)
3. The permission bits are OR'd with rw-rw----

If unset, the Kubelet will not modify the ownership and permissions of any volume. Note that this field cannot be set when spec.os.name is windows.

▼ fsGroupChangePolicy `string`

fsGroupChangePolicy defines behavior of changing ownership and permission of the volume before being exposed inside Pod. This field will only apply to volume types which support fsGroup based ownership(and permissions). It will have no effect on ephemeral volume types such as: secret, configmaps and emptydir. Valid values are "OnRootMismatch" and "Always". If not specified, "Always" is used. Note that this field cannot be set when spec.os.name is windows.

▼ runAsGroup `integer`

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ runAsNonRoot `boolean`

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in SecurityContext. If set in both SecurityContext and

PodSecurityContext, the value specified in SecurityContext takes precedence.

▼ runAsUser `integer`

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxChangePolicy `string`

seLinuxChangePolicy defines how the container's SELinux label is applied to all volumes used by the Pod. It has no effect on nodes that do not support SELinux or to volumes does not support SELinux. Valid values are "MountOption" and "Recursive".

"Recursive" means relabeling of all files on all Pod volumes by the container runtime. This may be slow for large volumes, but allows mixing privileged and unprivileged Pods sharing the same volume on the same node.

"MountOption" mounts all eligible Pod volumes with `-o context` mount option. This requires all Pods that share the same volume to use the same SELinux label. It is not possible to share the same volume among privileged and unprivileged Pods. Eligible volumes are in-tree FibreChannel and iSCSI volumes, and all CSI volumes whose CSI driver announces SELinux support by setting spec.seLinuxMount: true in their CSIDriver instance. Other volumes are always re-labelled recursively. "MountOption" value is allowed only when SELinuxMount feature gate is enabled.

If not specified and SELinuxMount feature gate is enabled, "MountOption" is used. If not specified and SELinuxMount feature gate is disabled, "MountOption" is used for ReadWriteOncePod volumes and "Recursive" for all other volumes.

This field affects only Pods that have SELinux label set, either in PodSecurityContext or in SecurityContext of all containers.

All Pods that use the same volume should use the same seLinuxChangePolicy, otherwise some pods can get stuck in ContainerCreating state. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxOptions `object`

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

▼ level `string`

Level is SELinux level label that applies to the container.

▼ role `string`

Role is a SELinux role label that applies to the container.

▼ type `string`

Type is a SELinux type label that applies to the container.

▼ user `string`

User is a SELinux user label that applies to the container.

▼ seccompProfile `object`

The seccomp options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ localhostProfile `string`

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

▼ type `string` required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

▼ supplementalGroups `[]integer`

A list of groups applied to the first process run in each container, in addition to the container's primary GID and fsGroup (if specified). If the SupplementalGroupsPolicy feature is enabled, the supplementalGroupsPolicy field determines whether these are in addition to or instead of any group memberships defined in the container image. If unspecified, no additional groups are added, though group memberships defined in the container image may still be used, depending on the supplementalGroupsPolicy field. Note that this field cannot be set when spec.os.name is windows.

▼ supplementalGroupsPolicy `string`

Defines how supplemental groups of the first container processes are calculated. Valid values are "Merge" and "Strict". If not specified, "Merge" is used. (Alpha) Using the field requires the SupplementalGroupsPolicy feature gate to be enabled and the container runtime must implement support for this feature. Note that this field cannot be set when spec.os.name is windows.

▼ sysctls `[]object`

Sysctl defines a kernel parameter to be set

▼ name `string` required

Name of a property to set

▼ value `string` required

Value of a property to set

▼ windowsOptions `object`

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

▼ gmsaCredentialSpec `string`

GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-

gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

▼ **gmsaCredentialSpecName** `string`

GMSACredentialSpecName is the name of the GMSA credential spec to use.

▼ **hostProcess** `boolean`

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

▼ **runAsUserName** `string`

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

▼ **svcRO** `object`

SvcRo defines the read only service of the MySQL Router.

▼ **port** `integer`

Port means the port of the node port.

▼ **type** `string`

Type means the type of the service.

▼ **svcRW** `object`

SvcRW defines the read write service of the MySQL Router.

▼ **port** `integer`

Port means the port of the node port.

▼ **type** `string`

Type means the type of the service.

▼ **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.

▼ runAsRoot `boolean`

RunAsRoot defines whether the MySQL container should run as root.

▼ securityContext `object`

SecurityContext holds Pod-level security attributes and common Container settings.

▼ appArmorProfile `object`

appArmorProfile is the AppArmor options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ localhostProfile `string`

localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".

▼ type `string` required

type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

▼ fsGroup `integer`

A special supplemental group that applies to all containers in a pod. Some volume types allow the Kubelet to change the ownership of that volume to be owned by the pod:

1. The owning GID will be the FSGroup
2. The setgid bit is set (new files created in the volume will be owned by FSGroup)
3. The permission bits are OR'd with rw-rw----

If unset, the Kubelet will not modify the ownership and permissions of any volume. Note that this field cannot be set when spec.os.name is windows.

▼ fsGroupChangePolicy `string`

fsGroupChangePolicy defines behavior of changing ownership and permission of the volume before being exposed inside Pod. This field will only apply to volume types which support fsGroup based ownership(and permissions). It will have no effect on ephemeral volume types such as: secret, configmaps and emptydir. Valid values are "OnRootMismatch" and

"Always". If not specified, "Always" is used. Note that this field cannot be set when spec.os.name is windows.

▼ runAsGroup `integer`

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ runAsNonRoot `boolean`

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

▼ runAsUser `integer`

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxChangePolicy `string`

seLinuxChangePolicy defines how the container's SELinux label is applied to all volumes used by the Pod. It has no effect on nodes that do not support SELinux or to volumes does not support SELinux. Valid values are "MountOption" and "Recursive".

"Recursive" means relabeling of all files on all Pod volumes by the container runtime. This may be slow for large volumes, but allows mixing privileged and unprivileged Pods sharing the same volume on the same node.

"MountOption" mounts all eligible Pod volumes with `-o context` mount option. This requires all Pods that share the same volume to use the same SELinux label. It is not possible to share the same volume among privileged and unprivileged Pods. Eligible volumes are in-tree FibreChannel and iSCSI volumes, and all CSI volumes whose CSI driver announces SELinux support by setting spec.seLinuxMount: true in their CSIDriver instance. Other volumes are always re-labelled recursively. "MountOption" value is allowed only when SELinuxMount feature gate is enabled.

If not specified and SELinuxMount feature gate is enabled, "MountOption" is used. If not specified and SELinuxMount feature gate is disabled, "MountOption" is used for ReadWriteOncePod volumes and "Recursive" for all other volumes.

This field affects only Pods that have SELinux label set, either in PodSecurityContext or in SecurityContext of all containers.

All Pods that use the same volume should use the same seLinuxChangePolicy, otherwise some pods can get stuck in ContainerCreating state. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxOptions `object`

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

▼ level `string`

Level is SELinux level label that applies to the container.

▼ role `string`

Role is a SELinux role label that applies to the container.

▼ type `string`

Type is a SELinux type label that applies to the container.

▼ user `string`

User is a SELinux user label that applies to the container.

▼ seccompProfile `object`

The seccomp options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ localhostProfile `string`

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

▼ type `string` required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

▼ supplementalGroups `[]integer`

A list of groups applied to the first process run in each container, in addition to the container's primary GID and fsGroup (if specified). If the SupplementalGroupsPolicy feature is enabled, the supplementalGroupsPolicy field determines whether these are in addition to or instead of any group memberships defined in the container image. If unspecified, no additional groups are added, though group memberships defined in the container image may still be used, depending on the supplementalGroupsPolicy field. Note that this field cannot be set when spec.os.name is windows.

▼ supplementalGroupsPolicy `string`

Defines how supplemental groups of the first container processes are calculated. Valid values are "Merge" and "Strict". If not specified, "Merge" is used. (Alpha) Using the field requires the SupplementalGroupsPolicy feature gate to be enabled and the container runtime must implement support for this feature. Note that this field cannot be set when spec.os.name is windows.

▼ sysctls `[]object`

Sysctl defines a kernel parameter to be set

▼ name `string` required

Name of a property to set

▼ value `string` required

Value of a property to set

▼ windowsOptions `object`

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

▼ gmsaCredentialSpec `string`

GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

▼ gmsaCredentialSpecName `string`

GMSACredentialSpecName is the name of the GMSA credential spec to use.

▼ hostProcess `boolean`

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

▼ runAsUserName `string`

The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both

SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

▼ slowSQL `object`

SlowSQL define MySQL slow sql spec

▼ enabled `boolean`

Enabled means enable slow sql

▼ image `string`

Image means the image of the slow sql.

▼ resources `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers.

▼ claims `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ name `string` required

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

▼ request `string`

Request is the name chosen for a request in the referenced claim. If empty, everything from the claim is

made available, otherwise only the result of this request.

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ **serverEndpoint** `string`

ServerEndpoint means the server endpoint of the slow sql.

▼ **sslSecret** `object`

SSLSecret allows a user to specify custom CA certificate, server certificate and server key for group replication SSL.

▼ **name** `string`

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

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

▼ strictSecurityModeEnabled `boolean`

Enable this option to prevent sensitive information from appearing in plain text in environment variables or files in a container.

▼ 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.

▼ upgradeOption `object`

UpgradeOption defines the upgrade option for the MySQL cluster.

▼ autoUpgrade `boolean`

AutoUpgrade means the auto upgrade option

▼ crVersion `string`

CRVersion means the version of the CR

▼ useJemalloc `boolean`

UseJemalloc means use jemalloc for MySQL

▼ version `string`

Version defines the MySQL container image version.

▼ volumeClaimTemplate `object`

VolumeClaimTemplate allows a user to specify how volumes inside a MySQL cluster

▼ apiVersion `string`

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

▼ kind `string`

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ metadata `object`

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

▼ spec `object`

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

▼ accessModes `[]string`

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

▼ dataSource `object`

dataSource field can be used to specify either:

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

▼ apiGroup `string`

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

▼ kind `string` required

Kind is the type of resource being referenced

▼ name `string` required

Name is the name of resource being referenced

▼ dataSourceRef `object`

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

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

▼ apiGroup `string`

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

▼ **kind** `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **namespace** `string`

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

▼ **resources** `object`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ **selector** `object`

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

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **storageClassName** `string`

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

▼ **volumeAttributesClassName** `string`

volumeAttributesClassName may be used to set the VolumeAttributesClass used by this claim. If specified, the CSI driver will create or update the volume with the attributes defined in the corresponding VolumeAttributesClass. This has a different

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

▼ **volumeMode** `string`

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

▼ **volumeName** `string`

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

▼ **status** `object`

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

▼ **accessModes** `[]string`

accessModes contains the actual access modes the volume backing the PVC has. More info:

https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1↗

▼ allocatedResourceStatuses `object`

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

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ allocatedResources `object`

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

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ capacity `object`

capacity represents the actual resources of the underlying volume.

▼ conditions `[]object`

PersistentVolumeClaimCondition contains details about state of pvc

▼ lastProbeTime `string`

lastProbeTime is the time we probed the condition.

▼ lastTransitionTime `string`

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

▼ message `string`

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

▼ reason `string`

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

▼ status `string` required

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

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

▼ **type** `string` required

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

▼ **currentVolumeAttributesClassName** `string`

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

▼ **modifyVolumeStatus** `object`

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

▼ **status** `string` required

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

- Pending Pending indicates that the PersistentVolumeClaim cannot be modified due to

unmet requirements, such as the specified VolumeAttributesClass not existing.

- InProgress InProgress indicates that the volume is being modified.
- Infeasible Infeasible indicates that the request has been rejected as invalid by the CSI driver. To resolve the error, a valid VolumeAttributesClass needs to be specified. Note: New statuses can be added in the future. Consumers should check for unknown statuses and fail appropriately.

▼ **targetVolumeAttributesClassName** `string`

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

▼ **phase** `string`

phase represents the current phase of PersistentVolumeClaim.

▼ **params** `object`

Params is the configuration of MySQL Server

▼ **mysql** `object`

▼ **router** `object`

▼ **paras** `object`

Paras deprecated: this field is deprecated, please use spec.params instead

▼ **pause** `boolean`

Pause is the flag to pause the MySQL cluster

▼ **upgradeOption** `object`

UpgradeOption is the option to upgrade the MySQL cluster

▼ **autoUpgrade** `boolean`

AutoUpgrade is the flag to auto upgrade the MySQL

▼ **crVersion** `string`

CRVersion is the version of the CR

▼ **useSafeConf** `boolean`

UseSafeConf is the flag to use safe configuration

▼ **version** `string` required

Version is the version of the MySQL Server

▼ **status** `object`

MysqlStatus defines the observed state of Mysql

▼ **message** `string`

Message is the message of the MySQL cluster

▼ **mgr** `object`

MGR is the status of MGR

▼ **clusterSetInitialized** `boolean`

Indicate if this cluster has been initialized as a member of innodb clusterset

▼ **clusterSetRole** `string`

Indicate the innodb cluster is primary or replica cluster as a member of innodb clusterset

▼ **conditions** `[]object`

PodCondition contains details for the current condition of this pod.

▼ **lastProbeTime** `string`

Last time we probed the condition.

▼ **lastTransitionTime** `string`

Last time the condition transitioned from one status to another.

▼ **message** `string`

Human-readable message indicating details about last transition.

▼ **reason** `string`

Unique, one-word, CamelCase reason for the condition's last transition.

▼ **status** `string` required

Status is the status of the condition. Can be True, False, Unknown. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-conditions ↗

▼ **type** `string` required

Type is the type of the condition. More info: https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-conditions ↗

▼ **currentVersion** `string`

CurrentVersion represents the current version of the MySQL.

▼ **initVersion** `string`

InitVersion represents the version of the MySQL when it was created.

▼ **operatorVersion** `string`

OperatorVersion means the operator version

▼ **primary** `[]string`

Primary represents the primary member of the MySQL.

▼ **primaryMode** `string`

PrimaryMode represents the primary mode of the MySQL.

▼ **readySize** `integer`

ReadySize represents the number of members online.

▼ **secondary** `[]string`

Secondary represents the secondary members of the MySQL.

▼ **state** `string`

State represents the current state of the MySQLCluster.

▼ **task** `string`

Task represents the current task of the MySQLCluster.

▼ **upgradeStatus** `object`

UpgradeStatus represents the upgrade status of the MySQL.

▼ **crVersion** `string`

CRVersion means the version of the CR

▼ **state** `string`

State is the state of the MySQL cluster

MGR Backup

`mysql.middleware.alauda.io` group

MySQLBackup is the Schema for the mysqlbackups API

`v1` version

▼ spec `object`

MySQLBackupSpec defines the desired state of MySQLBackup

▼ cluster `object`

Cluster is the name of source cluster.

▼ name `string`

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

▼ fullExecutor `string`

Executor is the executor of full backup.

▼ mysqlShellDump `object`

DumpOption is the dump option for backup.

▼ **excludeSchemas** `[]string`

ExcludeSchemas specifies the schemas to exclude from the backup.

▼ **excludeTables** `[]string`

ExcludeTables specifies the tables to exclude from the backup, using the 'schema.table' format.

▼ **includeSchemas** `[]string`

IncludeSchemas specifies the schemas to include in the backup.

▼ **includeTables** `[]string`

IncludeTables specifies the tables to include in the backup, using the 'schema.table' format.

▼ **maxRate** `integer`

MaxRate specifies the maximum rate at which data is transferred during the backup operation.

▼ **threads** `integer`

Threads specifies the number of threads to use for the backup operation.

▼ **storage** `object`

Storage is the storage information of backup for.

▼ **s3** `object`

S3 means s3 compatible object storage

▼ **bucket** `string`

Bucket in which to store the Backup.

▼ **endpoint** `string`

Endpoint (hostname only or fully qualified URI) of S3 compatible storage service.

▼ **region** `string`

Region in which the S3 compatible bucket is located.

▼ **secret** `object`

Secret is a reference to the Secret containing the credentials authenticating with the S3 compatible storage service.

▼ **name** `string`

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

▼ **storeMeta** `boolean`

StoreMeta is the flag of store meta data.

▼ **type** `string`

Type means full or increment backup.

▼ **status** `object`

MySQLBackupStatus defines the observed state of MySQLBackup

▼ **allocated** `string`

Allocated is the name of pod allocate to.

▼ **dataEndTime** `string`

DataEndTime is the stop time of increment backup or full backup data.

▼ **dataStartTime** `string`

DataStartTime is the start time of increment backup data.

▼ **executor** `string`

Executor is the executor of backup.

▼ **finishTime** `string`

FinishTime is the finish time of backup runs.

▼ **gtidEnd** `string`

GtidEnd is the stop gtid of backup data

▼ **gtidFullPrevious** `string`

FullGtid is the gtid of full backup if skip a full backup.

▼ **gtidPrevious** `string`

GtidPrevious is the gtid of previous backup.

▼ **gtidStart** `string`

GtidStart is the start gtid of backup data

▼ **mark** `string`

Mark is the mark of backup

▼ **memberSize** `integer`

MemberSize is the size of cluster member.

▼ **message** `string`

Message is the message of run result of backup

▼ **path** `string`

Path is the path of the file stored in storage.

▼ **startTime** `string`

StartTime is the start time of backup runs.

▼ **state** `string`

State is the backup state and enumeration of success, fail or running.

▼ **storeMeta** `boolean`

StoreMeta is the flag of store meta CR.

MGR Restore

`middleware.alauda.io` group

MySQLRestore is the Schema for the MySQLRestores API

`v1` version

▼ spec `object`

MySQLRestoreSpec defines the desired state of MySQLRestore

▼ mgr `object`

MGR use to create mgr restore schedule

▼ backup `object`

Backup is the name of Backup CR you want to restore.

▼ name `string`

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

▼ cluster `object`

Cluster is the cluster where the data will be restored to.

▼ **name** `string`

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

▼ **mysqlShellLoad** `object`

LoadOption is the load option for restore

▼ **excludeUsers** `[]string`

ExcludeUsers specifies the users to exclude from the backup.

▼ **includeSchemas** `[]string`

IncludeSchemas specifies the schemas to include in the backup.

▼ **includeTables** `[]string`

IncludeTables specifies the tables to include in the backup, using the 'schema.table' format.

▼ **includeUsers** `[]string`

IncludeUsers specifies the users to include in the backup.

▼ **loadUsers** `boolean`

loadUsers specifies whether to load users during the restore operation.

▼ **maxBytesPerTransaction** `integer`

MaxBytesPerTransaction specifies the maximum number of bytes to use per transaction during the restore operation.

▼ **threads** `integer`

Threads specifies the number of threads to use for the restore operation.

▼ **source** `object`

Source is the cluster where the data comes form

▼ **name** `string`

▼ **namespace** `string`

▼ **type** `string`

▼ **timePoint** `string`

TimePoint is the deadline of data you want to restore.

▼ **mysql** `object`

MySQL use to create a new MySQL instance

▼ **crVersion** `string`

Deprecated: use upgradeOption.CrVersion

▼ ipFamilyPrefer `string`

IPFamilyPrefer is the preferred IP family

▼ mgr `object`

MGR use to configure MySQLCluster

▼ affinity `object`

If specified, affinity will define the pod's scheduling constraints

▼ nodeAffinity `object`

Describes node affinity scheduling rules for the pod.

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

An empty preferred scheduling term matches all objects with implicit weight 0 (i.e. it's a no-op). A null preferred scheduling term matches no objects (i.e. is also a no-op).

####### ▼ preference `object` required

A node selector term, associated with the corresponding weight.

######## ▼ matchExpressions `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **weight** `integer` required

Weight associated with matching the corresponding nodeSelectorTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `object`

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to an update), the system may or may not try to eventually evict the pod from its node.

▼ **nodeSelectorTerms** `[]object` required

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values

array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values

array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ podAffinity `object`

Describes pod affinity scheduling rules (e.g. co-locate this pod in the same node, zone, etc. as some other pod(s)).

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ podAffinityTerm `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ labelSelector `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ matchExpressions `[]object`

A label selector requirement is a selector that contains values, a

key, and an operator that relates the key and values.

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels

map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with

`labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

"value". The requirements are ANDed.

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also,

matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **mismatchLabelKeys** `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

array contains only "value". The requirements are ANDed.

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **podAntiAffinity** `object`

Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in the same node, zone, etc. as some other pod(s)).

▼ **preferredDuringSchedulingIgnoredDuringExecution** `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred

node(s)

▼ **podAffinityTerm** `object` required

Required. A pod affinity term, associated with the corresponding weight.

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **matchLabelKeys** `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for

the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

values is an array of string values. If the operator is In

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

array contains only "value". The requirements are ANDed.

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be

set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where

co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ backupVolumeClaimTemplate `object`

BackupVolumeClaimTemplate allows a user to specify a volume to temporarily store the data for a backup prior to it being shipped to object storage.

▼ apiVersion `string`

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

▼ kind `string`

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ metadata `object`

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

▼ **spec** `object`

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

▼ **accessModes** `[]string`

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

▼ **dataSource** `object`

dataSource field can be used to specify either:

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

▼ **apiGroup** `string`

APIGroup is the group for the resource being referenced. If APIGroup is not specified, the

specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

▼ **kind** `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **dataSourceRef** `object`

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

- While dataSource only allows two specific types of objects, dataSourceRef allows any non-core object, as well as PersistentVolumeClaim objects.

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

▼ apiGroup `string`

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

▼ kind `string` required

Kind is the type of resource being referenced

▼ name `string` required

Name is the name of resource being referenced

▼ namespace `string`

Namespace is the namespace of resource being referenced Note that when a namespace is specified, a gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details.

(Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

▼ resources `object`

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

▼ limits `object`

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

▼ requests `object`

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

▼ **selector** `object`

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

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ storageClassName `string`

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

▼ volumeAttributesClassName `string`

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

VolumeAttributesClass feature gate to be enabled (off by default).

▼ **volumeMode** `string`

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

▼ **volumeName** `string`

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

▼ **status** `object`

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

▼ **accessModes** `[]string`

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

▼ **allocatedResourceStatuses** `object`

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-prefixed keys: - storage - the capacity of the volume. *

Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ allocatedResources `object`

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

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ capacity `object`

capacity represents the actual resources of the underlying volume.

▼ **conditions** `[]object`

PersistentVolumeClaimCondition contains details about state of pvc

▼ **lastProbeTime** `string`

lastProbeTime is the time we probed the condition.

▼ **lastTransitionTime** `string`

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

▼ **message** `string`

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

▼ **reason** `string`

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

▼ **status** `string` required

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

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

▼ **type** `string` required

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

▼ **currentVolumeAttributesClassName** `string`

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

▼ **modifyVolumeStatus** `object`

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

▼ **status** `string` required

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

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

▼ **targetVolumeAttributesClassName** `string`

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

▼ **phase** `string`

phase represents the current phase of PersistentVolumeClaim.

▼ **baseServerId** `integer`

BaseServerID defines the base number used to create unique server_id for MySQL instances in the cluster. Valid range 1 to 4294967286. If omitted in

the manifest file (or set to 0) defaultBaseServerID value will be used.

▼ certConfig `object`

CertConfig allows a user to specify custom CA certificate with advanced option.

▼ DNSNames `[]string`

DNSNames means the dns name list of the cert.

▼ duration `string`

Duration means the duration of the cert.

▼ secretName `string`

SecretName means the secret of the cert.

▼ config `object`

Config allows a user to specify a custom configuration file for MySQL.

▼ name `string`

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

▼ **configuration** `string`

Mysql configuration

▼ **enableStorage** `boolean`

EnableStorage allow a user to enable mysql storage

▼ **image** `object`

Image allows a user to specify image.

▼ **agentImage** `string`

AgentImage means the image of the agent.

▼ **routerImage** `string`

RouterImage means the image of the router.

▼ **serverImage** `string`

ServerImage means the image of the server.

▼ **slowSQL** `string`

SlowSQL means the image of the slow sql.

▼ **imagePullPolicy** `string`

PullPolicy describes a policy for if/when to pull a container image

▼ imagePullSecret `[]object`

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

▼ name `string`

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

▼ ipFamilyPolicy `string`

IpFamilyPolicy means the IP family policy for cluster

▼ ipFamilyPrefer `string`

IpFamilyPrefer means the preferred IP family for cluster

▼ jemallocConf `string`

JemallocConf means the jemalloc configuration

▼ members `integer`

Members defines the number of MySQL instances in a cluster. InnoDB clustering supports between 1-9 members.

▼ monitor `object`

Monitor define MySQL monitor spec

▼ **enable** `boolean`

Enable means enable monitor

▼ **exporter** `object`

Exporter define MySQL exporter spec

▼ **image** `string`

Image means the image of the exporter.

▼ **resources** `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers.

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

Request is the name chosen for a request in the referenced claim. If empty,

everything from the claim is made available, otherwise only the result of this request.

▼ limits `object`

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

▼ requests `object`

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

▼ multiClusterReplication `object`

Config this field to support multi cluster replication for disaster recovery.

▼ bootstrapSecret `string`

BootstrapSecret means the secret of the multi cluster replication.

▼ clusterSetSeedAddr `string`

ClusterSetSeedAddr means the cluster set seed address of the multi cluster replication.

▼ **enabled** `boolean`

Enabled means enable multi cluster replication

▼ **hostPort** `integer`

HostPort means the host port of the multi cluster replication.

▼ **multiMaster** `boolean`

MultiMaster defines the mode of the MySQL cluster. If set to true, all instances will be R/W. If false (the default), only a single instance will be R/W and the rest will be R/O.

▼ **nodeSelector** `object`

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

▼ **paras** `object`

Paras provide paras template feature

▼ **patches** `object`

Patches use to apply patches to the MySQL container

▼ **podPatches** `object`

PodPatches is a map of pod name to patch

▼ **servicePatches** `object`

ServicePatches is a map of service name to patch

▼ **pause** `boolean`

Pause means pause the cluster

▼ **podCIDR** `string`

PodCIDR allows a user to specify custom CIDR.

▼ **readOnlyRootFilesystem** `boolean`

ReadOnlyRootFilesystem means the root filesystem is read-only

▼ **repository** `string`

Repository defines the image repository from which to pull the MySQL server image.

▼ **resources** `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers

▼ **agent** `object`

Agent describes the compute resource requirements for agent.

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

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

▼ server `object`

Server describes the compute resource requirements for server.

▼ claims `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ name `string` required

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

▼ request `string`

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

▼ limits `object`

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

▼ **requests** `object`

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

▼ **rootPasswordSecret** `object`

If defined, we use this secret for configuring the MYSQL_ROOT_PASSWORD If it is not set we generate a secret dynamically

▼ **name** `string`

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

▼ **router** `object`

Router defines the MySQL Router

▼ **affinity** `object`

If specified, affinity will define the pod's scheduling constraints

▼ **nodeAffinity** `object`

Describes node affinity scheduling rules for the pod.

▼ **preferredDuringSchedulingIgnoredDuringExecution** `[]object`

An empty preferred scheduling term matches all objects with implicit weight 0 (i.e. it's a no-op). A null preferred scheduling term matches no objects (i.e. is also a no-op).

▼ **preference** `object` required

A node selector term, associated with the corresponding weight.

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

The label key that the selector applies to.

▼ **operator** `string` required

Represents a key's relationship to a set of values. Valid operators are

In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string`

required

The label key that the selector applies to.

▼ **operator** `string`

required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **weight** `integer` required

Weight associated with matching the corresponding nodeSelectorTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `object`

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to an update), the system may or may not try to eventually evict the pod from its node.

▼ **nodeSelectorTerms** `[]object`

required

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

The label key that the selector applies to.

▼ **operator** `string`

required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ **matchFields** `[]object`

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

▼ **key** `string`

required

The label key that the selector applies to.

▼ **operator** `string`

required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist. Gt, and Lt.

▼ **values** `[]string`

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

▼ podAffinity `object`

Describes pod affinity scheduling rules (e.g. co-locate this pod in the same node, zone, etc. as some other pod(s)).

▼ preferredDuringSchedulingIgnoredDuringExecution `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ podAffinityTerm `object`

required

Required. A pod affinity term, associated with the corresponding weight.

▼ labelSelector `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

####### ▼ matchExpressions `[]object`

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

######## ▼ key `string`

required

key is the label key that the selector applies to.

▼ **operator**

`string`

required

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

▼ **values**

`[]string`

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

▼ **matchLabels**

`object`

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

▼ **matchLabelKeys**

`[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set

when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector**
`object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions**
`[]object`

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

▼ **key** `string`
required

key is the label key that the selector applies to.

▼ **operator**
`string`
required

operator represents a key's

relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

▼ **values**
`[]string`

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

▼ **matchLabels**
`object`

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and

the values array contains only "value". The requirements are ANDed.

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string`

required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both

mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ **namespaceSelector** `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies

to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **podAntiAffinity** `object`

Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in the same node, zone, etc. as some other pod(s)).

▼ **preferredDuringSchedulingIgnoredDuringExecution** `[]object`

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

▼ podAffinityTerm `object`

required

Required. A pod affinity term, associated with the corresponding weight.

▼ labelSelector `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ matchExpressions `[]object`

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

▼ key `string`

required

key is the label key that the selector applies to.

▼ operator `string`

required

operator represents a key's relationship to a set of values. Valid

operators are In, NotIn, Exists and DoesNotExist.

▼ **values**
`[]string`

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

▼ **matchLabels**
`object`

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

only "value". The requirements are ANDed.

▼ matchLabelKeys

`[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also, matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys

`[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will

be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions**
`[]object`

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

▼ **key** `string`
required

key is the label key that the selector applies to.

▼ **operator**
`string`
required

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

▼ **values**
`[]string`

values is an array of string values. If the operator is In or NotIn, the values array must

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

▼ **matchLabels** `object`

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

▼ **namespaces** `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null

namespaceSelector means "this pod's namespace".

▼ **topologyKey** `string`

required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ **weight** `integer` required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

▼ **requiredDuringSchedulingIgnoredDuringExecution** `[]object`

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the

label with key matches that of any node on which a pod of the set of pods is running

▼ **labelSelector** `object`

A label query over a set of resources, in this case pods. If it's null, this PodAffinityTerm matches with no Pods.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

values is an array of string values. If the operator is In

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

▼ matchLabels `object`

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

▼ matchLabelKeys `[]string`

MatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key in (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both matchLabelKeys and labelSelector. Also,

matchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ mismatchLabelKeys `[]string`

MismatchLabelKeys is a set of pod label keys to select which pods will be taken into consideration. The keys are used to lookup values from the incoming pod labels, those key-value labels are merged with `labelSelector` as `key notin (value)` to select the group of existing pods which pods will be taken into consideration for the incoming pod's pod (anti) affinity. Keys that don't exist in the incoming pod labels will be ignored. The default value is empty. The same key is forbidden to exist in both mismatchLabelKeys and labelSelector. Also, mismatchLabelKeys cannot be set when labelSelector isn't set. This is a beta field and requires enabling MatchLabelKeysInPodAffinity feature gate (enabled by default).

▼ namespaceSelector `object`

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this

pod's namespace". An empty selector ({}) matches all namespaces.

▼ **matchExpressions** `[]object`

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

▼ **key** `string`

required

key is the label key that the selector applies to.

▼ **operator** `string`

required

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

▼ **values** `[]string`

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This

array is replaced during a strategic merge patch.

▼ matchLabels `object`

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

▼ namespaces `[]string`

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

▼ topologyKey `string` required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of

any node on which any of the selected pods is running. Empty topologyKey is not allowed.

▼ configuration `string`

Configuration means the configuration of the router.

▼ mysqlPasswordSecret `string`

MySQLPasswordSecret defines the secret name of MySQL user.

▼ mysqlUser `string`

MySQLUser defines the user that connect to MySQL.

▼ nodeSelector `object`

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

▼ replicas `integer`

Replicas defines the replicas of the MySQL Router.

▼ resources `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers

▼ **claims** `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

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

▼ securityContext `object`

PodSecurityContext holds pod-level security attributes and common container settings. Some fields are also present in container.securityContext. Field values of container.securityContext take precedence over field values of PodSecurityContext.

▼ appArmorProfile `object`

appArmorProfile is the AppArmor options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ localhostProfile `string`

localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".

▼ type `string` required

type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

▼ fsGroup `integer`

A special supplemental group that applies to all containers in a pod. Some volume types allow the Kubelet to change the ownership of that volume to be owned by the pod:

1. The owning GID will be the FSGroup
2. The setgid bit is set (new files created in the volume will be owned by FSGroup)
3. The permission bits are OR'd with rw-rw----

If unset, the Kubelet will not modify the ownership and permissions of any volume. Note that this field cannot be set when spec.os.name is windows.

▼ fsGroupChangePolicy `string`

fsGroupChangePolicy defines behavior of changing ownership and permission of the volume before being exposed inside Pod. This field will only apply to volume types which support fsGroup based ownership(and permissions). It will have no effect on ephemeral volume types such as: secret, configmaps and emptydir. Valid values are "OnRootMismatch" and "Always". If not specified, "Always" is used. Note that this field cannot be set when spec.os.name is windows.

▼ runAsGroup `integer`

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ runAsNonRoot `boolean`

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

▼ runAsUser `integer`

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxChangePolicy `string`

seLinuxChangePolicy defines how the container's SELinux label is applied to all volumes used by the Pod. It has no effect on nodes that do not support SELinux or to volumes does not support SELinux. Valid values are "MountOption" and "Recursive".

"Recursive" means relabeling of all files on all Pod volumes by the container runtime. This may be slow for large volumes, but allows mixing privileged and unprivileged Pods sharing the same volume on the same node.

"MountOption" mounts all eligible Pod volumes with `-o context` mount option. This requires all Pods that share the same volume to use the same SELinux label. It is not

possible to share the same volume among privileged and unprivileged Pods. Eligible volumes are in-tree FibreChannel and iSCSI volumes, and all CSI volumes whose CSI driver announces SELinux support by setting spec.seLinuxMount: true in their CSIDriver instance. Other volumes are always re-labelled recursively. "MountOption" value is allowed only when SELinuxMount feature gate is enabled.

If not specified and SELinuxMount feature gate is enabled, "MountOption" is used. If not specified and SELinuxMount feature gate is disabled, "MountOption" is used for ReadWriteOncePod volumes and "Recursive" for all other volumes.

This field affects only Pods that have SELinux label set, either in PodSecurityContext or in SecurityContext of all containers.

All Pods that use the same volume should use the same seLinuxChangePolicy, otherwise some pods can get stuck in ContainerCreating state. Note that this field cannot be set when spec.os.name is windows.

▼ **seLinuxOptions** `object`

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

▼ **level** `string`

Level is SELinux level label that applies to the container.

▼ **role** `string`

Role is a SELinux role label that applies to the container.

▼ **type** `string`

Type is a SELinux type label that applies to the container.

▼ **user** `string`

User is a SELinux user label that applies to the container.

▼ **seccompProfile** `object`

The seccomp options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ **localhostProfile** `string`

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

▼ **type** `string` required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

▼ supplementalGroups `[]integer`

A list of groups applied to the first process run in each container, in addition to the container's primary GID and fsGroup (if specified). If the SupplementalGroupsPolicy feature is enabled, the supplementalGroupsPolicy field determines whether these are in addition to or instead of any group memberships defined in the container image. If unspecified, no additional groups are added, though group memberships defined in the container image may still be used, depending on the supplementalGroupsPolicy field. Note that this field cannot be set when spec.os.name is windows.

▼ supplementalGroupsPolicy `string`

Defines how supplemental groups of the first container processes are calculated. Valid values are "Merge" and "Strict". If not specified, "Merge" is used. (Alpha) Using the field requires the SupplementalGroupsPolicy feature gate to be enabled and the container runtime must implement support for this feature. Note that this field cannot be set when spec.os.name is windows.

▼ sysctls `[]object`

Sysctl defines a kernel parameter to be set

▼ **name** `string` required

Name of a property to set

▼ **value** `string` required

Value of a property to set

▼ **windowsOptions** `object`

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

▼ **gmsaCredentialSpec** `string`

GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

▼ **gmsaCredentialSpecName** `string`

GMSACredentialSpecName is the name of the GMSA credential spec to use.

▼ **hostProcess** `boolean`

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

▼ **runAsUserName** `string`

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

▼ **svcRO** `object`

SvcRo defines the read only service of the MySQL Router.

▼ **port** `integer`

Port means the port of the node port.

▼ **type** `string`

Type means the type of the service.

▼ **svcRW** `object`

SvcRW defines the read write service of the MySQL Router.

▼ **port** `integer`

Port means the port of the node port.

▼ **type** `string`

Type means the type of the service.

▼ **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.

▼ runAsRoot `boolean`

RunAsRoot defines whether the MySQL container should run as root.

▼ securityContext `object`

SecurityContext holds Pod-level security attributes and common Container settings.

▼ appArmorProfile `object`

appArmorProfile is the AppArmor options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ **localhostProfile** `string`

localhostProfile indicates a profile loaded on the node that should be used. The profile must be preconfigured on the node to work. Must match the loaded name of the profile. Must be set if and only if type is "Localhost".

▼ **type** `string` required

type indicates which kind of AppArmor profile will be applied. Valid options are: Localhost - a profile pre-loaded on the node. RuntimeDefault - the container runtime's default profile. Unconfined - no AppArmor enforcement.

▼ **fsGroup** `integer`

A special supplemental group that applies to all containers in a pod. Some volume types allow the Kubelet to change the ownership of that volume to be owned by the pod:

1. The owning GID will be the FSGroup
2. The setgid bit is set (new files created in the volume will be owned by FSGroup)
3. The permission bits are OR'd with rw-rw----

If unset, the Kubelet will not modify the ownership and permissions of any volume. Note that this field cannot be set when spec.os.name is windows.

▼ **fsGroupChangePolicy** `string`

fsGroupChangePolicy defines behavior of changing ownership and permission of the volume before being exposed inside Pod. This field will only apply to volume types which support fsGroup based ownership(and permissions). It will have no effect on ephemeral

volume types such as: secret, configmaps and emptydir. Valid values are "OnRootMismatch" and "Always". If not specified, "Always" is used. Note that this field cannot be set when spec.os.name is windows.

▼ **runAsGroup** `integer`

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ **runAsNonRoot** `boolean`

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

▼ **runAsUser** `integer`

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ **seLinuxChangePolicy** `string`

seLinuxChangePolicy defines how the container's SELinux label is applied to all volumes used by the Pod. It has no effect on nodes that do not support SELinux or to volumes does not support SELinux. Valid values are "MountOption" and "Recursive".

"Recursive" means relabeling of all files on all Pod volumes by the container runtime. This may be slow for large volumes, but allows mixing privileged and unprivileged Pods sharing the same volume on the same node.

"MountOption" mounts all eligible Pod volumes with `-o context` mount option. This requires all Pods that share the same volume to use the same SELinux label. It is not possible to share the same volume among privileged and unprivileged Pods. Eligible volumes are in-tree FibreChannel and iSCSI volumes, and all CSI volumes whose CSI driver announces SELinux support by setting spec.seLinuxMount: true in their CSIDriver instance. Other volumes are always re-labelled recursively. "MountOption" value is allowed only when SELinuxMount feature gate is enabled.

If not specified and SELinuxMount feature gate is enabled, "MountOption" is used. If not specified and SELinuxMount feature gate is disabled, "MountOption" is used for ReadWriteOncePod volumes and "Recursive" for all other volumes.

This field affects only Pods that have SELinux label set, either in PodSecurityContext or in SecurityContext of all containers.

All Pods that use the same volume should use the same seLinuxChangePolicy, otherwise some pods can get stuck in ContainerCreating state. Note that this field cannot be set when spec.os.name is windows.

▼ seLinuxOptions `object`

The SELinux context to be applied to all containers. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in

SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

▼ **level** `string`

Level is SELinux level label that applies to the container.

▼ **role** `string`

Role is a SELinux role label that applies to the container.

▼ **type** `string`

Type is a SELinux type label that applies to the container.

▼ **user** `string`

User is a SELinux user label that applies to the container.

▼ **seccompProfile** `object`

The seccomp options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

▼ **localhostProfile** `string`

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

▼ **type** `string` required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used. RuntimeDefault - the container runtime default profile should be used. Unconfined - no profile should be applied.

▼ **supplementalGroups** `[]integer`

A list of groups applied to the first process run in each container, in addition to the container's primary GID and fsGroup (if specified). If the SupplementalGroupsPolicy feature is enabled, the supplementalGroupsPolicy field determines whether these are in addition to or instead of any group memberships defined in the container image. If unspecified, no additional groups are added, though group memberships defined in the container image may still be used, depending on the supplementalGroupsPolicy field. Note that this field cannot be set when spec.os.name is windows.

▼ **supplementalGroupsPolicy** `string`

Defines how supplemental groups of the first container processes are calculated. Valid values are "Merge" and "Strict". If not specified, "Merge" is used. (Alpha) Using the field requires the SupplementalGroupsPolicy feature gate to be enabled and the container runtime must implement support for this feature. Note that this field cannot be set when spec.os.name is windows.

▼ **sysctls** `[]object`

Sysctl defines a kernel parameter to be set

▼ **name** `string` required

Name of a property to set

▼ **value** `string` required

Value of a property to set

▼ **windowsOptions** `object`

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

▼ **gmsaCredentialSpec** `string`

GMSACredentialSpec is where the GMSA admission webhook (https://github.com/kubernetes-sigs/windows-gmsa↗) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

▼ **gmsaCredentialSpecName** `string`

GMSACredentialSpecName is the name of the GMSA credential spec to use.

▼ **hostProcess** `boolean`

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-

HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

▼ runAsUserName `string`

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

▼ slowSQL `object`

SlowSQL define MySQL slow sql spec

▼ enabled `boolean`

Enabled means enable slow sql

▼ image `string`

Image means the image of the slow sql.

▼ resources `object`

Resources holds ResourceRequirements for the MySQL Agent & Server Containers.

▼ claims `[]object`

ResourceClaim references one entry in PodSpec.ResourceClaims.

▼ **name** `string` required

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

▼ **request** `string`

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

▼ **limits** `object`

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

▼ **requests** `object`

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

▼ **serverEndpoint** `string`

ServerEndpoint means the server endpoint of the slow sql.

▼ sslSecret `object`

SSLSecret allows a user to specify custom CA certificate, server certificate and server key for group replication SSL.

▼ name `string`

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

▼ strictSecurityModeEnabled `boolean`

Enable this option to prevent sensitive information from appearing in plain text in environment variables or files in a container.

▼ 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.

▼ upgradeOption `object`

UpgradeOption defines the upgrade option for the MySQL cluster.

▼ autoUpgrade `boolean`

AutoUpgrade means the auto upgrade option

▼ crVersion `string`

CRVersion means the version of the CR

▼ useJemalloc `boolean`

UseJemalloc means use jemalloc for MySQL

▼ version `string`

Version defines the MySQL container image version.

▼ volumeClaimTemplate `object`

VolumeClaimTemplate allows a user to specify how volumes inside a MySQL cluster

▼ apiVersion `string`

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources ↗

▼ kind `string`

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

▼ metadata `object`

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

▼ spec `object`

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

▼ accessModes `[]string`

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

▼ dataSource `object`

dataSource field can be used to specify either:

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

▼ **apiGroup** `string`

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

▼ **kind** `string` required

Kind is the type of resource being referenced

▼ **name** `string` required

Name is the name of resource being referenced

▼ **dataSourceRef** `object`

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

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

▼ apiGroup `string`

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

▼ kind `string` required

Kind is the type of resource being referenced

▼ name `string` required

Name is the name of resource being referenced

▼ namespace `string`

Namespace is the namespace of resource being referenced Note that when a namespace is specified, a

gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. (Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

▼ resources `object`

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

▼ limits `object`

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

▼ requests `object`

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

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

▼ **selector** `object`

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

▼ **matchExpressions** `[]object`

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

▼ **key** `string` required

key is the label key that the selector applies to.

▼ **operator** `string` required

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

▼ **values** `[]string`

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

▼ **matchLabels** `object`

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

▼ **storageClassName** `string`

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

▼ **volumeAttributesClassName** `string`

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

attributes-classes/↗ (Beta) Using this field requires the VolumeAttributesClass feature gate to be enabled (off by default).

▼ **volumeMode** `string`

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

▼ **volumeName** `string`

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

▼ **status** `object`

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

▼ **accessModes** `[]string`

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

▼ **allocatedResourceStatuses** `object`

allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either: * Un-

prefixed keys: - storage - the capacity of the volume. * Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource" Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ allocatedResources `object`

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

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

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

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

▼ capacity `object`

capacity represents the actual resources of the underlying volume.

▼ **conditions** `[]object`

PersistentVolumeClaimCondition contains details about state of pvc

▼ **lastProbeTime** `string`

lastProbeTime is the time we probed the condition.

▼ **lastTransitionTime** `string`

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

▼ **message** `string`

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

▼ **reason** `string`

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

▼ **status** `string` required

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

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

▼ **type** `string` required

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

▼ **currentVolumeAttributesClassName** `string`

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

▼ **modifyVolumeStatus** `object`

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

▼ **status** `string` required

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

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

▼ **targetVolumeAttributesClassName** `string`

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

▼ **phase** `string`

phase represents the current phase of PersistentVolumeClaim.

▼ **params** `object`

Params is the configuration of MySQL Server

▼ **mysql** `object`

▼ **router** `object`

▼ **paras** `object`

Paras deprecated: this field is deprecated, please use spec.params instead

▼ **pause** `boolean`

Pause is the flag to pause the MySQL cluster

▼ **runAsRoot** `boolean`

RunAsRoot is the flag to run MySQL Server as root

▼ **upgradeOption** `object`

UpgradeOption is the option to upgrade the MySQL cluster

▼ **autoUpgrade** `boolean`

AutoUpgrade is the flag to auto upgrade the MySQL

▼ **crVersion** `string`

CRVersion is the version of the CR

▼ **useSafeConf** `boolean`

UseSafeConf is the flag to use safe configuration

▼ **version** `string` required

Version is the version of the MySQL Server

▼ **status** `object`

MySQLRestoreStatus defines the observed state of MySQLRestore

▼ **mgr** `object`

MySQLRestoreStatus defines the observed state of MySQLRestore

▼ **allocated** `string`

Allocated is the name of pod allocate to.

▼ **message** `string`

Message is the message of run result of restore

▼ **progress** `integer`

Progress is the current restore progress in running.

▼ **restoreSchedule** `[]object`

▼ **backupInfo** `object`

BackupInfo is the backup information for restore

▼ **name** `string`

Name is the name of Backup CR.

▼ **spec** `object`

Spec is the spec of Backup CR.

▼ **cluster** `object`

Cluster is the name of source cluster.

▼ **name** `string`

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

▼ **fullExecutor** `string`

Executor is the executor of full backup.

▼ **mysqlShellDump** `object`

DumpOption is the dump option for backup.

▼ **excludeSchemas** `[]string`

ExcludeSchemas specifies the schemas to exclude from the backup.

▼ **excludeTables** `[]string`

ExcludeTables specifies the tables to exclude from the backup, using the 'schema.table' format.

▼ **includeSchemas** `[]string`

IncludeSchemas specifies the schemas to include in the backup.

▼ **includeTables** `[]string`

IncludeTables specifies the tables to include in the backup, using the 'schema.table' format.

▼ **maxRate** `integer`

MaxRate specifies the maximum rate at which data is transferred during the backup operation.

▼ **threads** `integer`

Threads specifies the number of threads to use for the backup operation.

▼ **storage** `object`

Storage is the storage information of backup for.

▼ **s3** `object`

S3 means s3 compatible object storage

▼ **bucket** `string`

Bucket in which to store the Backup.

▼ **endpoint** `string`

Endpoint (hostname only or fully qualified URI) of S3 compatible storage service.

▼ **region** `string`

Region in which the S3 compatible bucket is located.

▼ **secret** `object`

Secret is a reference to the Secret containing the credentials authenticating with the S3 compatible storage service.

▼ **name** `string`

Name of the referent. This field is effectively required, but due to backwards compatibility is allowed to be empty. Instances of this type with an empty value here are almost certainly wrong. More info: https://kubernetes.io/docs/concept

s/overview/working-with-objects/names/#names↗

▼ **storeMeta** `boolean`

StoreMeta is the flag of store meta data.

▼ **type** `string`

Type means full or increment backup.

▼ **status** `object`

Status is the status of backup CR.

▼ **allocated** `string`

Allocated is the name of pod allocate to.

▼ **dataEndTime** `string`

DataEndTime is the stop time of increment backup or full backup data.

▼ **dataStartTime** `string`

DataStartTime is the start time of increment backup data.

▼ **executor** `string`

Executor is the executor of backup.

▼ **finishTime** `string`

FinishTime is the finish time of backup runs.

▼ **gtidEnd** `string`

GtidEnd is the stop gtid of backup data

▼ **gtidFullPrevious** `string`

FullGtid is the gtid of full backup if skip a full backup.

▼ **gtidPrevious** `string`

GtidPrevious is the gtid of previous backup.

▼ **gtidStart** `string`

GtidStart is the start gtid of backup data

▼ **mark** `string`

Mark is the mark of backup

▼ **memberSize** `integer`

MemberSize is the size of cluster member.

▼ **message** `string`

Message is the message of run result of backup

▼ **path** `string`

Path is the path of the file stored in storage.

▼ **startTime** `string`

StartTime is the start time of backup runs.

▼ **state** `string`

State is the backup state and enumeration of success, fail or running.

▼ **storeMeta** `boolean`

StoreMeta is the flag of store meta CR.

▼ **version** `integer`

Version is the mark of continues backup.

▼ **message** `string`

Message is the message of run result of restore

▼ **state** `string`

State is the state for restore

▼ **state** `string`

State is the overall state for restore

▼ **total** `integer`

Total is the total restore schedule.

▼ **state** `string`

MGR Schedule

`mysql.middleware.alauda.io` group

MySQLSchedule is the Schema for the mysqlschedules API

`v1` version

▼ spec `object`

MySQLScheduleSpec defines the desired state of MySQLSchedule

▼ cluster `object`

Cluster is the name of source cluster.

▼ name `string`

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

▼ expiryDays `integer`

ExpiryDays is the days of the latest backup data hold.

▼ full `object`

Full is the full backup schedule.

▼ **cron** `string`

Cron is the scheduled backup time expressed using cron.

▼ **enable** `boolean`

Enable is enable or disable schedule backup.

▼ **fullExecutor** `string`

Executor is the executor of full backup.

▼ **incr** `object`

Incr is the increment backup schedule.

▼ **cron** `string`

Cron is the scheduled backup time expressed using cron.

▼ **enable** `boolean`

Enable is enable or disable schedule backup.

▼ **mysqlShellDump** `object`

DumpOption is the dump option for backup.

▼ **excludeSchemas** `[]string`

ExcludeSchemas specifies the schemas to exclude from the backup.

▼ **excludeTables** `[]string`

ExcludeTables specifies the tables to exclude from the backup, using the 'schema.table' format.

▼ **includeSchemas** `[]string`

IncludeSchemas specifies the schemas to include in the backup.

▼ **includeTables** `[]string`

IncludeTables specifies the tables to include in the backup, using the 'schema.table' format.

▼ **maxRate** `integer`

MaxRate specifies the maximum rate at which data is transferred during the backup operation.

▼ **threads** `integer`

Threads specifies the number of threads to use for the backup operation.

▼ **storage** `object`

Storage is the storage information of backup for.

▼ **s3** `object`

S3 means s3 compatible object storage

▼ **bucket** `string`

Bucket in which to store the Backup.

▼ **endpoint** `string`

Endpoint (hostname only or fully qualified URI) of S3 compatible storage service.

▼ **region** `string`

Region in which the S3 compatible bucket is located.

▼ **secret** `object`

Secret is a reference to the Secret containing the credentials authenticating with the S3 compatible storage service.

▼ **name** `string`

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

▼ **storeMeta** `boolean`

StoreMeta is the flag of store meta data.

▼ status `object`

MySQLScheduleStatus defines the observed state of MySQLSchedule

▼ fullLatestTime `string`

FullLatestTime is the full backup latest run time.

▼ incrLatestTime `string`

IncrLatestTime is the increment backup latest run time.

MGR User

`middleware.alauda.io` group

MysqlUser is the Schema for the mysqlusers API

`v1` version

▼ spec `object`

MysqlUserSpec defines the desired state of MysqlUser

▼ host `string` required

Host is the host of the MySQL User

▼ mysql `string` required

Mysql is the name of the MySQL Cluster

▼ privileges `[]object`

Privilege defines the desired mysql Privilege

▼ grants `[]string`

Grants is the list of grants for the user

▼ targets `[]string`

Targets is the list target of MySQL Databases or Tables

▼ **secretName** `string`

SecretName is the name of the Secret for the MySQL User Password

▼ **user** `string` required

INSERT ADDITIONAL SPEC FIELDS - desired state of cluster Important: Run "make" to regenerate code after modifying this file User is the name of the MySQL User

▼ **status** `object`

MysqlUserStatus defines the observed state of MysqlUser

▼ **lastSyncTime** `string`

LastSyncTime is the last time the MySQL User sync

▼ **reason** `string`

Reason is the reason of the MySQL User sync failed

▼ **secretRef** `object`

SecretRef is the reference of the Secret for the MySQL User Password

▼ **name** `string`

▼ **namespace** `string`

▼ state `string`

State is the state of the MySQL User sync