AutoScaling APIs

HorizontalPodAutoscaler [autoscaling/v2]

HorizontalPodAutoscaler [autoscaling/v2]

Description

HorizontalPodAutoscaler is the configuration for a horizontal pod autoscaler, which automatically manages the replica count of any resource implementing the scale subresource based on the metrics specified.

Type

`object`

Specification

Property	Type	Description
`apiVersion`	`string`	APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources
`kind`	`string`	Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
`metadata`	`ObjectMeta`	ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.
`spec`	`object`	HorizontalPodAutoscalerSpec describes the desired functionality of the HorizontalPodAutoscaler.
`status`	`object`	HorizontalPodAutoscalerStatus describes the current status of a horizontal pod autoscaler.

.spec

Description

HorizontalPodAutoscalerSpec describes the desired functionality of the HorizontalPodAutoscaler.

Type

`object`

Required

`scaleTargetRef` `maxReplicas`

Property	Type	Description
`behavior`	`object`	HorizontalPodAutoscalerBehavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively).
`maxReplicas`	`integer`	maxReplicas is the upper limit for the number of replicas to which the autoscaler can scale up. It cannot be less that minReplicas.
`metrics`	`array`	metrics contains the specifications for which to use to calculate the desired replica count (the maximum replica count across all metrics will be used). The desired replica count is calculated multiplying the ratio between the target value and the current value by the current number of pods. Ergo, metrics used must decrease as the pod count is increased, and vice-versa. See the individual metric source types for more information about how each type of metric must respond. If not set, the default metric will be set to 80% average CPU utilization.
`minReplicas`	`integer`	minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPAScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.
`scaleTargetRef`	`object`	CrossVersionObjectReference contains enough information to let you identify the referred resource.

.spec.behavior

Description

HorizontalPodAutoscalerBehavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively).

Type

`object`

Property	Type	Description
`scaleDown`	`object`	HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.
`scaleUp`	`object`	HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.

.spec.behavior.scaleDown

Description

HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.

Type

`object`

Property	Type	Description
`policies`	`array`	policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid
`selectPolicy`	`string`	selectPolicy is used to specify which policy should be used. If not set, the default value Max is used.
`stabilizationWindowSeconds`	`integer`	stabilizationWindowSeconds is the number of seconds for which past recommendations should be considered while scaling up or scaling down. StabilizationWindowSeconds must be greater than or equal to zero and less than or equal to 3600 (one hour). If not set, use the default values: - For scale up: 0 (i.e. no stabilization is done). - For scale down: 300 (i.e. the stabilization window is 300 seconds long).

.spec.behavior.scaleDown.policies

Description

policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid

Type

`array`

.spec.behavior.scaleDown.policies[]

Description

HPAScalingPolicy is a single policy which must hold true for a specified past interval.

Type

`object`

Required

`type` `value` `periodSeconds`

Property	Type	Description
`periodSeconds`	`integer`	periodSeconds specifies the window of time for which the policy should hold true. PeriodSeconds must be greater than zero and less than or equal to 1800 (30 min).
`type`	`string`	type is used to specify the scaling policy.

Property	Type	Description
`value`	`integer`	value contains the amount of change which is permitted by the policy. It must be greater than zero

.spec.behavior.scaleUp

Description

HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.

Type

`object`

Property	Type	Description
`policies`	`array`	policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid
`selectPolicy`	`string`	selectPolicy is used to specify which policy should be used. If not set, the default value Max is used.
`stabilizationWindowSeconds`	`integer`	stabilizationWindowSeconds is the number of seconds for which past recommendations should be considered while scaling up or scaling down. StabilizationWindowSeconds must be greater than or equal to zero and less than or equal to 3600 (one hour). If not set, use the default values: - For scale up: 0 (i.e. no stabilization is done). - For scale down: 300 (i.e. the stabilization window is 300 seconds long).

.spec.behavior.scaleUp.policies

Description

policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid

Type

`array`

.spec.behavior.scaleUp.policies[]

Description

HPAScalingPolicy is a single policy which must hold true for a specified past interval.

Type

`object`

Required

`type` `value` `periodSeconds`

Property	Type	Description
`periodSeconds`	`integer`	periodSeconds specifies the window of time for which the policy should hold true. PeriodSeconds must be greater than zero and less than or equal to 1800 (30 min).
`type`	`string`	type is used to specify the scaling policy.
`value`	`integer`	value contains the amount of change which is permitted by the policy. It must be greater than zero

.spec.metrics

Description

metrics contains the specifications for which to use to calculate the desired replica count (the maximum replica count across all metrics will be used). The desired replica count is calculated multiplying the ratio between the target value and the current value by the current number of pods. Ergo, metrics used must decrease as the pod count is increased, and vice-versa. See the individual metric source types for more information about how each type of metric must respond. If not set, the default metric will be set to 80% average CPU utilization.

Type

`array`

.spec.metrics[]

Description

MetricSpec specifies how to scale based on a single metric (only `type` and one other matching field should be set at once).

Type

`object`

Required

`type`

Property	Type	Description
`containerResource`	`object`	ContainerResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.
`external`	`object`	ExternalMetricSource indicates how to scale on a metric not associated with any Kubernetes object (for example length of queue in cloud messaging service, or QPS from loadbalancer running outside of cluster).
`object`	`object`	ObjectMetricSource indicates how to scale on a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

Property	Type	Description
`pods`	`object`	PodsMetricSource indicates how to scale on a metric describing each pod in the current scale target (for example, transactions-processed-per-second). The values will be averaged together before being compared to the target value.
`resource`	`object`	ResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.
`type`	`string`	type is the type of metric source. It should be one of "ContainerResource", "External", "Object", "Pods" or "Resource", each mapping to a matching field in the object.

.spec.metrics[].containerResource

Description

ContainerResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.

Type

`object`

Required

`name` `target` `container`

Property	Type	Description
`container`	`string`	container is the name of the container in the pods of the scaling target
`name`	`string`	name is the name of the resource in question.
`target`	`object`	MetricTarget defines the target value, average value, or average utilization of a specific metric

.spec.metrics[].containerResource.target

Description

MetricTarget defines the target value, average value, or average utilization of a specific metric

Type

`object`

Required

`type`

Property	Type	Description
`averageUtilization`	`integer`	averageUtilization is the target value of the average of the resource metric across all relevant pods, represer
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`type`	`string`	type represents whether the metric type is Utilization, Value, or AverageValue
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.spec.metrics[].external

Description

ExternalMetricSource indicates how to scale on a metric not associated with any Kubernetes object (for example length of queue in cloud messaging service, or QPS from loadbalancer running outside of cluster).

Type

`object`

Required

`metric` `target`

Property	Type	Description
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric
`target`	`object`	MetricTarget defines the target value, average value, or average utilization of a specific metric

.spec.metrics[].external.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.spec.metrics[].external.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.spec.metrics[].external.metric.selector.matchExpressions

Description

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

Type

`array`

.spec.metrics[].external.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.spec.metrics[].external.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.spec.metrics[].external.metric.selector.matchExpressions[].values[]

Type

`string`

.spec.metrics[].external.metric.selector.matchLabels

Description

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

Type

`object`

.spec.metrics[].external.target

Description

MetricTarget defines the target value, average value, or average utilization of a specific metric

Type

`object`

Required

`type`

Property	Type	Description
`averageUtilization`	`integer`	averageUtilization is the target value of the average of the resource metric across all relevant pods, represer

Property	Type	Description
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`type`	`string`	type represents whether the metric type is Utilization, Value, or AverageValue
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.spec.metrics[].object

Description

ObjectMetricSource indicates how to scale on a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

Type

`object`

Required

`describedObject` `target` `metric`

Property	Type	Description
`describedObject`	`object`	CrossVersionObjectReference contains enough information to let you identify the referred resource.
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric
`target`	`object`	MetricTarget defines the target value, average value, or average utilization of a specific metric

.spec.metrics[].object.describedObject

Description

CrossVersionObjectReference contains enough information to let you identify the referred resource.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiVersion`	`string`	apiVersion is the API version of the referent
`kind`	`string`	kind is the kind of the referent; More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
`name`	`string`	name is the name of the referent; More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names ↗

.spec.metrics[].object.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.spec.metrics[].object.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.spec.metrics[].object.metric.selector.matchExpressions

Description

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

Type

`array`

.spec.metrics[].object.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.spec.metrics[].object.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.spec.metrics[].object.metric.selector.matchExpressions[].values[]

Type

`string`

.spec.metrics[].object.metric.selector.matchLabels

Description

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

Type

`object`

.spec.metrics[].object.target

Description

MetricTarget defines the target value, average value, or average utilization of a specific metric

Type

`object`

Required

`type`

Property	Type	Description
`averageUtilization`	`integer`	averageUtilization is the target value of the average of the resource metric across all relevant pods, represer
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin
`type`	`string`	type represents whether the metric type is Utilization, Value, or AverageValue
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.spec.metrics[].pods

Description

PodsMetricSource indicates how to scale on a metric describing each pod in the current scale target (for example, transactions-processed-per-second). The values will be averaged together before being compared to the target value.

Type

`object`

Required

`metric` `target`

Property	Type	Description
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric
`target`	`object`	MetricTarget defines the target value, average value, or average utilization of a specific metric

.spec.metrics[].pods.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.spec.metrics[].pods.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.spec.metrics[].pods.metric.selector.matchExpressions

Description

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

Type

`array`

.spec.metrics[].pods.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.spec.metrics[].pods.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.spec.metrics[].pods.metric.selector.matchExpressions[].values[]

Type

`string`

.spec.metrics[].pods.metric.selector.matchLabels

Description

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

Type

`object`

.spec.metrics[].pods.target

Description

MetricTarget defines the target value, average value, or average utilization of a specific metric

Type

`object`

Required

`type`

Property	Type	Description
`averageUtilization`	`integer`	averageUtilization is the target value of the average of the resource metric across all relevant pods, represer

Property	Type	Description
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`type`	`string`	type represents whether the metric type is Utilization, Value, or AverageValue
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.spec.metrics[].resource

Description

ResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.

Type

`object`

Required

`name` `target`

Property	Type	Description
`name`	`string`	name is the name of the resource in question.
`target`	`object`	MetricTarget defines the target value, average value, or average utilization of a specific metric

.spec.metrics[].resource.target

Description

MetricTarget defines the target value, average value, or average utilization of a specific metric

Type

`object`

Required

`type`

Property	Type	Description
`averageUtilization`	`integer`	averageUtilization is the target value of the average of the resource metric across all relevant pods, represer
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`type`	`string`	type represents whether the metric type is Utilization, Value, or AverageValue
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO

Property	Type	Description
		The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi

.spec.scaleTargetRef

Description

CrossVersionObjectReference contains enough information to let you identify the referred resource.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiVersion`	`string`	apiVersion is the API version of the referent
`kind`	`string`	kind is the kind of the referent; More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

Property	Type	Description
`name`	`string`	name is the name of the referent; More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names

.status

Description

HorizontalPodAutoscalerStatus describes the current status of a horizontal pod autoscaler.

Type

`object`

Required

`desiredReplicas`

Property	Type	Description
`conditions`	`array`	conditions is the set of conditions required for this autoscaler to scale its target, and indicates whether or not those conditions are met.
`currentMetrics`	`array`	currentMetrics is the last read state of the metrics used by this autoscaler.
`currentReplicas`	`integer`	currentReplicas is current number of replicas of pods managed by this autoscaler, as last seen by the autoscaler.
`desiredReplicas`	`integer`	desiredReplicas is the desired number of replicas of pods managed by this autoscaler, as last calculated by the autoscaler.
`lastScaleTime`	`string`	Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
`observedGeneration`	`integer`	observedGeneration is the most recent generation observed by this autoscaler.

.status.conditions

Description

conditions is the set of conditions required for this autoscaler to scale its target, and indicates whether or not those conditions are met.

Type

`array`

.status.conditions[]

Description

HorizontalPodAutoscalerCondition describes the state of a HorizontalPodAutoscaler at a certain point.

Type

`object`

Required

`type` `status`

Property	Type	Description
`lastTransitionTime`	`string`	Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
`message`	`string`	message is a human-readable explanation containing details about the transition
`reason`	`string`	reason is the reason for the condition's last transition.
`status`	`string`	status is the status of the condition (True, False, Unknown)
`type`	`string`	type describes the current condition

.status.currentMetrics

Description

currentMetrics is the last read state of the metrics used by this autoscaler.

Type

`array`

.status.currentMetrics[]

Description

MetricStatus describes the last-read state of a single metric.

Type

`object`

Required

`type`

Property	Type	Description
`containerResource`	`object`	ContainerResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing a single container in each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

Property	Type	Description
`external`	`object`	ExternalMetricStatus indicates the current value of a global metric not associated with any Kubernetes object.
`object`	`object`	ObjectMetricStatus indicates the current value of a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).
`pods`	`object`	PodsMetricStatus indicates the current value of a metric describing each pod in the current scale target (for example, transactions-processed-per-second).
`resource`	`object`	ResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.
`type`	`string`	type is the type of metric source. It will be one of "ContainerResource", "External", "Object", "Pods" or "Resource", each corresponds to a matching field in the object.

.status.currentMetrics[].containerResource

Description

ContainerResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing a single container in each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

Type

`object`

Required

`name` `current` `container`

Property	Type	Description
`container`	`string`	container is the name of the container in the pods of the scaling target
`current`	`object`	MetricValueStatus holds the current value for a metric
`name`	`string`	name is the name of the resource in question.

.status.currentMetrics[].containerResource.current

Description

MetricValueStatus holds the current value for a metric

Type

`object`

Property	Type	Description
`averageUtilization`	`integer`	currentAverageUtilization is the current value of the average of the resource metric across all relevant pods,
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.status.currentMetrics[].external

Description

ExternalMetricStatus indicates the current value of a global metric not associated with any Kubernetes object.

Type

`object`

Required

`metric` `current`

Property	Type	Description
`current`	`object`	MetricValueStatus holds the current value for a metric
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric

.status.currentMetrics[].external.current

Description

MetricValueStatus holds the current value for a metric

Type

`object`

Property	Type	Description
`averageUtilization`	`integer`	currentAverageUtilization is the current value of the average of the resource metric across all relevant pods,
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.status.currentMetrics[].external.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.status.currentMetrics[].external.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.status.currentMetrics[].external.metric.selector.matchExpressions

Description

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

Type

`array`

.status.currentMetrics[].external.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.status.currentMetrics[].external.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.status.currentMetrics[].external.metric.selector.matchExpressions[].values[]

Type

`string`

.status.currentMetrics[].external.metric.selector.matchLabels

Description

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

Type

`object`

.status.currentMetrics[].object

Description

ObjectMetricStatus indicates the current value of a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

Type

`object`

Required

`metric` `current` `describedObject`

Property	Type	Description
`current`	`object`	MetricValueStatus holds the current value for a metric
`describedObject`	`object`	CrossVersionObjectReference contains enough information to let you identify the referred resource.
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric

.status.currentMetrics[].object.current

Description

MetricValueStatus holds the current value for a metric

Type

`object`

Property	Type	Description
`averageUtilization`	`integer`	currentAverageUtilization is the current value of the average of the resource metric across all relevant pods,

Property	Type	Description
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.status.currentMetrics[].object.describedObject

Description

CrossVersionObjectReference contains enough information to let you identify the referred resource.

Type

`object`

Required

`kind` `name`

Property	Type	Description
`apiVersion`	`string`	apiVersion is the API version of the referent
`kind`	`string`	kind is the kind of the referent; More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗

Property	Type	Description
`name`	`string`	name is the name of the referent; More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names

.status.currentMetrics[].object.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.status.currentMetrics[].object.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.status.currentMetrics[].object.metric.selector.matchExpressions

Description

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

Type

`array`

.status.currentMetrics[].object.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.status.currentMetrics[].object.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.status.currentMetrics[].object.metric.selector.matchExpressions[].values[]

Type

`string`

.status.currentMetrics[].object.metric.selector.matchLabels

Description

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

Type

`object`

.status.currentMetrics[].pods

Description

PodsMetricStatus indicates the current value of a metric describing each pod in the current scale target (for example, transactions-processed-per-second).

Type

`object`

Required

`metric` `current`

Property	Type	Description
`current`	`object`	MetricValueStatus holds the current value for a metric
`metric`	`object`	MetricIdentifier defines the name and optionally selector for a metric

.status.currentMetrics[].pods.current

Description

MetricValueStatus holds the current value for a metric

Type

`object`

Property	Type	Description
`averageUtilization`	`integer`	currentAverageUtilization is the current value of the average of the resource metric across all relevant pods,
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

.status.currentMetrics[].pods.metric

Description

MetricIdentifier defines the name and optionally selector for a metric

Type

`object`

Required

`name`

Property	Type	Description
`name`	`string`	name is the name of the given metric
`selector`	`object`	A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

.status.currentMetrics[].pods.metric.selector

Description

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

Type

`object`

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

.status.currentMetrics[].pods.metric.selector.matchExpressions

Description

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

Type

`array`

.status.currentMetrics[].pods.metric.selector.matchExpressions[]

Description

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

Type

`object`

Required

`key` `operator`

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

.status.currentMetrics[].pods.metric.selector.matchExpressions[].values

Description

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

Type

`array`

.status.currentMetrics[].pods.metric.selector.matchExpressions[].values[]

Type

`string`

.status.currentMetrics[].pods.metric.selector.matchLabels

Description

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

Type

`object`

.status.currentMetrics[].resource

Description

ResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

Type

`object`

Required

`name` `current`

Property	Type	Description
`current`	`object`	MetricValueStatus holds the current value for a metric
`name`	`string`	name is the name of the resource in question.

.status.currentMetrics[].resource.current

Description

MetricValueStatus holds the current value for a metric

Type

`object`

Property	Type	Description
`averageUtilization`	`integer`	currentAverageUtilization is the current value of the average of the resource metric across all relevant pods,
`averageValue`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO

Property	Type	Description
		The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/bina <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that E - No precision is lost - No fractional digits will be emitted - The exponent (o The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writi
`value`	`string|number`	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSO The serialization format is:

Property	Type	Description
		(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 | 1 | ... | 9 <digits> ::= <digit> | <digit><d (International System of units; See: http://physics.nist.gov/cuu/Units/binar <decimalSI> ::= m | "" | k | M | G | T | P | E (Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.) <decimalExponent> ::= "e" <signedNumber> | "E" <signedNumber> ``` No matter which of the three exponent forms is used, no quantity may represent a When a Quantity is parsed from a string, it will remember the type of suffix it Before serializing, Quantity will be put in "canonical form". This means that Ex - No precision is lost - No fractional digits will be emitted - The exponent (or The sign will be omitted unless the number is negative. Examples: - 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi" Note that the quantity will NEVER be internally represented by a floating point Non-canonical values will still parse as long as they are well formed, but will This format is intended to make it difficult to use these numbers without writin

API Endpoints

The following API endpoints are available:

- `/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers`
  - `DELETE` : delete collection of HorizontalPodAutoscaler
  - `GET` : list objects of kind HorizontalPodAutoscaler
  - `POST` : create a new HorizontalPodAutoscaler
- `/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}`
  - `DELETE` : delete the specified HorizontalPodAutoscaler
  - `GET` : read the specified HorizontalPodAutoscaler
  - `PATCH` : partially update the specified HorizontalPodAutoscaler
  - `PUT` : replace the specified HorizontalPodAutoscaler
- `/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}/status`
  - `GET` : read status of the specified HorizontalPodAutoscaler
  - `PATCH` : partially update status of the specified HorizontalPodAutoscaler
  - `PUT` : replace status of the specified HorizontalPodAutoscaler

/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers

HTTP method

`DELETE`

Description

delete collection of HorizontalPodAutoscaler

HTTP responses

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

HTTP method

`GET`

Description

list objects of kind HorizontalPodAutoscaler

HTTP responses

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

HTTP method

`POST`

Description

create a new HorizontalPodAutoscaler

Query parameters

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

Body parameters

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

HTTP responses

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

/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}

HTTP method

`DELETE`

Description

delete the specified HorizontalPodAutoscaler

Query parameters

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

HTTP responses

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

HTTP method

`GET`

Description

read the specified HorizontalPodAutoscaler

HTTP responses

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

HTTP method

`PATCH`

Description

partially update the specified HorizontalPodAutoscaler

Query parameters

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

HTTP responses

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

HTTP method

`PUT`

Description

replace the specified HorizontalPodAutoscaler

Query parameters

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

Body parameters

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

HTTP responses

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

HTTP code	Response body
401 - Unauthorized	Empty

/kubernetes/{cluster}/apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

HTTP method

`GET`

Description

read status of the specified HorizontalPodAutoscaler

HTTP responses

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

HTTP method

`PATCH`

Description

partially update status of the specified HorizontalPodAutoscaler

Query parameters

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

HTTP responses

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

HTTP method

`PUT`

Description

replace status of the specified HorizontalPodAutoscaler

Query parameters

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

Body parameters

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

HTTP responses

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