



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
<code>behavior</code>	<code>object</code>	HorizontalPodAutoscalerBehavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively).
<code>maxReplicas</code>	<code>integer</code>	maxReplicas is the upper limit for the number of replicas to which the autoscaler can scale up. It cannot be less than minReplicas.
<code>metrics</code>	<code>array</code>	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.
<code>minReplicas</code>	<code>integer</code>	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.
<code>scaleTargetRef</code>	<code>object</code>	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
<code>scaleDown</code>	<code>object</code>	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.
<code>scaleUp</code>	<code>object</code>	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
<code>policies</code>	<code>array</code>	<code>policies</code> 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
<code>selectPolicy</code>	<code>string</code>	<code>selectPolicy</code> is used to specify which policy should be used. If not set, the default value Max is used.
<code>stabilizationWindowSeconds</code>	<code>integer</code>	<code>stabilizationWindowSeconds</code> is the number of seconds for which past recommendations should be considered while scaling up or scaling down. <code>StabilizationWindowSeconds</code> 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
<code>periodSeconds</code>	<code>integer</code>	<code>periodSeconds</code> specifies the window of time for which the policy should hold true. <code>PeriodSeconds</code> must be greater than zero and less than or equal to 1800 (30 min).
<code>type</code>	<code>string</code>	<code>type</code> is used to specify the scaling policy.

Property	Type	Description
<code>value</code>	<code>integer</code>	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
<code>policies</code>	<code>array</code>	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
<code>selectPolicy</code>	<code>string</code>	selectPolicy is used to specify which policy should be used. If not set, the default value Max is used.
<code>stabilizationWindowSeconds</code>	<code>integer</code>	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
<code>periodSeconds</code>	<code>integer</code>	<code>periodSeconds</code> specifies the window of time for which the policy should hold true. <code>PeriodSeconds</code> must be greater than zero and less than or equal to 1800 (30 min).
<code>type</code>	<code>string</code>	<code>type</code> is used to specify the scaling policy.
<code>value</code>	<code>integer</code>	<code>value</code> 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
<code>containerResource</code>	<code>object</code>	<code>ContainerResourceMetricSource</code> 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.
<code>external</code>	<code>object</code>	<code>ExternalMetricSource</code> 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).
<code>object</code>	<code>object</code>	<code>ObjectMetricSource</code> indicates how to scale on a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

Property	Type	Description
<code>pods</code>	<code>object</code>	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.
<code>resource</code>	<code>object</code>	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.
<code>type</code>	<code>string</code>	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
<code>container</code>	<code>string</code>	container is the name of the container in the pods of the scaling target
<code>name</code>	<code>string</code>	name is the name of the resource in question.
<code>target</code>	<code>object</code>	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/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 : 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/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 : 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 (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
		Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON. The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><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 value outside the range of approximately +/- 10^30. When a Quantity is parsed from a string, it will remember the type of suffix it encountered. The type is one of: - No precision is lost - No fractional digits will be emitted - The exponent (if any) 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 number. Non-canonical values will still parse as long as they are well formed, but will be canonicalized. This format is intended to make it difficult to use these numbers without writing a parser.
averageValue	string number	
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 JSON. 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 : 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 (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
<code>matchExpressions</code>	<code>array</code>	<code>matchExpressions</code> is a list of label selector requirements. The requirements are ANDed.
<code>matchLabels</code>	<code>object</code>	<code>matchLabels</code> is a map of {key,value} pairs. A single {key,value} in the <code>matchLabels</code> map is equivalent to an element of <code>matchExpressions</code> , 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
<code>key</code>	<code>string</code>	<code>key</code> is the label key that the selector applies to.
<code>operator</code>	<code>string</code>	<code>operator</code> represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.
<code>values</code>	<code>array</code>	<code>values</code> 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/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 : 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 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/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 : 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 (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
		Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON. The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><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 value outside the range of approximately +/- 10^30. When a Quantity is parsed from a string, it will remember the type of suffix it encountered. The type is one of: - Utilization (e.g. 50%) - Value (e.g. 5000000000) - AverageValue (e.g. 5000000000) Before serializing, Quantity will be put in "canonical form". This means that the internal representation is used to generate the canonical form. The canonical form has the following characteristics: - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will be emitted - 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 number. Non-canonical values will still parse as long as they are well formed, but will be converted to canonical form. This format is intended to make it difficult to use these numbers without writing a parser.
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 JSON. 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 i 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 (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, representing the average value.
averageValue	string number	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML. The serialization format is: <pre>(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><digits> <decimalExponent> ::= "e" <signedNumber> "E" <signedNumber> <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 non-integer multiple of its unit. When a Quantity is parsed from a string, it will remember the type of suffix it was given. Before serializing, Quantity will be put in "canonical form". This means that the value is always separated from the unit by a single space, and that canonical units are used. - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will always be emitted - 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 number. Non-canonical values will still parse as long as they are well formed, but will be converted to canonical form. This format is intended to make it difficult to use these numbers without writing a parser.</pre>
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 JSON and YAML.

Property	Type	Description
		The serialization format is: <pre>(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 i 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 (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</pre>

.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
<code>name</code>	<code>string</code>	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
<code>conditions</code>	<code>array</code>	conditions is the set of conditions required for this autoscaler to scale its target, and indicates whether or not those conditions are met.
<code>currentMetrics</code>	<code>array</code>	currentMetrics is the last read state of the metrics used by this autoscaler.
<code>currentReplicas</code>	<code>integer</code>	currentReplicas is current number of replicas of pods managed by this autoscaler, as last seen by the autoscaler.
<code>desiredReplicas</code>	<code>integer</code>	desiredReplicas is the desired number of replicas of pods managed by this autoscaler, as last calculated by the autoscaler.
<code>lastScaleTime</code>	<code>string</code>	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.
<code>observedGeneration</code>	<code>integer</code>	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
<code>external</code>	<code>object</code>	ExternalMetricStatus indicates the current value of a global metric not associated with any Kubernetes object.
<code>object</code>	<code>object</code>	ObjectMetricStatus indicates the current value of a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).
<code>pods</code>	<code>object</code>	PodsMetricStatus indicates the current value of a metric describing each pod in the current scale target (for example, transactions-processed-per-second).
<code>resource</code>	<code>object</code>	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.
<code>type</code>	<code>string</code>	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
<code>container</code>	<code>string</code>	container is the name of the container in the pods of the scaling target
<code>current</code>	<code>object</code>	MetricValueStatus holds the current value for a metric
<code>name</code>	<code>string</code>	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 JSON and YAML. The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><digits> (International System of units; See: http://physics.nist.gov/cuu/Units/binary) <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 value outside the range of approximately +/- 10^308. When a Quantity is parsed from a string, it will remember the type of suffix it was given and will try to preserve the original form when serializing. However, when a Quantity is parsed from a string, it will be normalized to a canonical form. This means that Equivalent quantities may be serialized differently depending on the input format. Before serializing, Quantity will be put in "canonical form". This means that Equivalent quantities may be serialized differently depending on the input format. - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will be emitted when the number is not an integer 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 number. It is intended to be able to parse quantities with large exponents or decimal digits without overflowing internal limited-precision representations. Non-canonical values will still parse as long as they are well formed, but will be normalized to canonical form. This format is intended to make it difficult to use these numbers without writing a parser.
value	string number	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML. 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 : 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 (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

metriccurrent

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,
		Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML. The serialization format is: (Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><digits> (International System of units; See: http://physics.nist.gov/cuu/Units/binary) <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 value outside the range of approximately +/- 1e308. When a Quantity is parsed from a string, it will remember the type of suffix it parsed, and will serialize it back to the canonical representation for that type. Before serializing, Quantity will be put in "canonical form". This means that the internal representation is used to fill in the missing bits of the canonical representation. - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will be chosen to avoid fractional digits 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 number. All internal calculations are done on integers. Non-canonical values will still parse as long as they are well formed, but will be serialized into canonical form. This format is intended to make it difficult to use these numbers without writing a parser.
averageValue	string number	No matter which of the three exponent forms is used, no quantity may represent a value outside the range of approximately +/- 1e308. When a Quantity is parsed from a string, it will remember the type of suffix it parsed, and will serialize it back to the canonical representation for that type. Before serializing, Quantity will be put in "canonical form". This means that the internal representation is used to fill in the missing bits of the canonical representation. - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will be chosen to avoid fractional digits 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 number. All internal calculations are done on integers. Non-canonical values will still parse as long as they are well formed, but will be serialized into canonical form. This format is intended to make it difficult to use these numbers without writing a parser.
value	string number	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML. 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 : 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 (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
		Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON. The serialization format is: <pre>(Note that <suffix> may be empty, from the "" case in <decimalSI>.) <digit> ::= 0 1 ... 9 <digits> ::= <digit> <digit><digit> (International System of units; See: http://physics.nist.gov/cuu/Units/binary) <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 floating point number. When a Quantity is parsed from a string, it will remember the type of suffix it has. Before serializing, Quantity will be put in "canonical form". This means that E notation will be used for large numbers, and scientific notation will be used for small numbers. - No precision is lost - No fractional digits will be emitted - The exponent (or suffix) will be emitted if it is non-zero. 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 number. Non-canonical values will still parse as long as they are well formed, but will be put in canonical form. This format is intended to make it difficult to use these numbers without writing a parser.</pre>
averageValue	string number	
value	string number	Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON. 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 : 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 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
<code>name</code>	<code>string</code>	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
<code>name</code>	<code>string</code>	name is the name of the given metric
<code>selector</code>	<code>object</code>	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
<code>matchExpressions</code>	<code>array</code>	matchExpressions is a list of label selector requirements. The requirements are ANDed.
<code>matchLabels</code>	<code>object</code>	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 : 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 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/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 : 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 (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 : 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 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/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 : 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 (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	<code>Status</code> schema
401 - Unauthorized	Empty

HTTP method

`GET`

Description

list objects of kind HorizontalPodAutoscaler

HTTP responses

HTTP code	Response body
200 - OK	<code>HorizontalPodAutoscalerList</code> schema
401 - Unauthorized	Empty

HTTP method

`POST`

Description

create a new HorizontalPodAutoscaler

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	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
<code>fieldValidation</code>	<code>string</code>	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
<code>body</code>	<code>HorizontalPodAutoscaler</code> schema	<code>application/json</code> formatted

HTTP responses

HTTP code	Response body
200 - OK	<code>HorizontalPodAutoscaler</code> schema
201 - Created	<code>HorizontalPodAutoscaler</code> schema
202 - Accepted	<code>HorizontalPodAutoscaler</code> 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
<code>dryRun</code>	<code>string</code>	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	<code>Status</code> schema
202 - Accepted	<code>Status</code> schema
401 - Unauthorized	Empty

HTTP method

GET

Description

read the specified HorizontalPodAutoscaler

HTTP responses

HTTP code	Response body
200 - OK	<code>HorizontalPodAutoscaler</code> schema
401 - Unauthorized	Empty

HTTP method

PATCH

Description

partially update the specified HorizontalPodAutoscaler

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> 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
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> 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 <code>BadRequest</code> 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	<code>HorizontalPodAutoscaler</code> schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace the specified `HorizontalPodAutoscaler`

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	When present, indicates that modifications should not be persisted. An invalid or unrecognized <code>dryRun</code> 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
<code>fieldValidation</code>	<code>string</code>	<code>fieldValidation</code> 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 <code>BadRequest</code> 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
<code>body</code>	<code>HorizontalPodAutoscaler</code> schema	<code>application/json</code> formatted

HTTP responses

HTTP code	Response body
200 - OK	<code>HorizontalPodAutoscaler</code> schema
201 - Created	<code>HorizontalPodAutoscaler</code> 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	<code>HorizontalPodAutoscaler</code> schema
401 - Unauthorized	Empty

HTTP method

`PATCH`

Description

partially update status of the specified HorizontalPodAutoscaler

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	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
<code>fieldValidation</code>	<code>string</code>	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	<code>HorizontalPodAutoscaler</code> schema
401 - Unauthorized	Empty

HTTP method

`PUT`

Description

replace status of the specified HorizontalPodAutoscaler

Query parameters

Parameter	Type	Description
<code>dryRun</code>	<code>string</code>	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
<code>fieldValidation</code>	<code>string</code>	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
<code>body</code>	<code>HorizontalPodAutoscaler</code> schema	<code>application/json</code> formatted

HTTP responses

HTTP code	Response body
200 - OK	<code>HorizontalPodAutoscaler</code> schema
201 - Created	<code>HorizontalPodAutoscaler</code> schema
401 - Unauthorized	Empty