Node APIs

Node [v1]

Node [v1]

/kubernetes/{cluster}/api/v1/nodes

Common Parameters

- `pretty` (*in query*): `string`
  If 'true', then the output is pretty printed. Defaults to 'false' unless the user-agent indicates a browser or command-line HTTP tool (curl and wget).

`get`

list or watch objects of kind Node

Parameters

- `allowWatchBookmarks` (*in query*): `boolean`
  allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.
- `continue` (*in query*): `string`
  The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change

on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key".
This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.

- `fieldSelector` (*in query*): `string`
  A selector to restrict the list of returned objects by their fields. Defaults to everything.
- `labelSelector` (*in query*): `string`
  A selector to restrict the list of returned objects by their labels. Defaults to everything.
- `limit` (*in query*): `integer`
  limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true.
  The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
- `resourceVersion` (*in query*): `string`
  resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions ↗ for details.
  Defaults to unset

- `resourceVersionMatch` (*in query*): `string`
  resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions for details.
  Defaults to unset
- `sendInitialEvents` (*in query*): `boolean`
  `sendInitialEvents=true` may be set together with `watch=true`. In that case, the watch stream will begin with synthetic events to produce the current state of objects in the collection. Once all such events have been sent, a synthetic "Bookmark" event will be sent. The bookmark will report the ResourceVersion (RV) corresponding to the set of objects, and be marked with `"k8s.io/initial-events-end": "true"` annotation. Afterwards, the watch stream will proceed as usual, sending watch events corresponding to changes (subsequent to the RV) to objects watched.
  When `sendInitialEvents` option is set, we require `resourceVersionMatch` option to also be set. The semantic of the watch request is as following: - `resourceVersionMatch` = NotOlderThan is interpreted as "data at least as new as the provided `resourceVersion`" and the bookmark event is send when the state is synced to a `resourceVersion` at least as fresh as the one provided by the ListOptions. If `resourceVersion` is unset, this is interpreted as "consistent read" and the bookmark event is send when the state is synced at least to the moment when request started being processed.
    - `resourceVersionMatch` set to any other value or unset Invalid error is returned.

  Defaults to true if `resourceVersion=""` or `resourceVersion="0"` (for backward compatibility reasons) and to false otherwise.
- `timeoutSeconds` (*in query*): `integer`
  Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.
- `watch` (*in query*): `boolean`
  Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

Response

- `200` NodeList: OK

- `401` : Unauthorized

`post`

create a Node

Parameters

- `dryRun` (*in query*): `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
- `fieldManager` (*in query*): `string`
  fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint ↗.
- `fieldValidation` (*in query*): `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.

Request Body

Node

Response

- `200` Node: OK

- `201` Node: Created
- `202` Node: Accepted
- `401` : Unauthorized

`delete`

delete collection of Node

Parameters

- `continue` (*in query*): `string`
  The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key".
  This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.
- `dryRun` (*in query*): `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
- `fieldSelector` (*in query*): `string`
  A selector to restrict the list of returned objects by their fields. Defaults to everything.
- `gracePeriodSeconds` (*in query*): `integer`
  The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace

period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.

- `ignoreStoreReadErrorWithClusterBreakingPotential` (*in query*): `boolean`
  if set to true, it will trigger an unsafe deletion of the resource in case the normal deletion flow fails with a corrupt object error. A resource is considered corrupt if it can not be retrieved from the underlying storage successfully because of a) its data can not be transformed e.g. decryption failure, or b) it fails to decode into an object. NOTE: unsafe deletion ignores finalizer constraints, skips precondition checks, and removes the object from the storage. WARNING: This may potentially break the cluster if the workload associated with the resource being unsafe-deleted relies on normal deletion flow. Use only if you REALLY know what you are doing. The default value is false, and the user must opt in to enable it
- `labelSelector` (*in query*): `string`
  A selector to restrict the list of returned objects by their labels. Defaults to everything.
- `limit` (*in query*): `integer`
  limit is a maximum number of responses to return for a list call. If more items exist, the server will set the `continue` field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true.
  The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.
- `orphanDependents` (*in query*): `boolean`
  Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added

to/removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.

- `propagationPolicy` (*in query*): `string`
  Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.
- `resourceVersion` (*in query*): `string`
  resourceVersion sets a constraint on what resource versions a request may be served from. See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions ↗ for details.
  Defaults to unset
- `resourceVersionMatch` (*in query*): `string`
  resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set See https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions ↗ for details.
  Defaults to unset
- `sendInitialEvents` (*in query*): `boolean`
  `sendInitialEvents=true` may be set together with `watch=true`. In that case, the watch stream will begin with synthetic events to produce the current state of objects in the collection. Once all such events have been sent, a synthetic "Bookmark" event will be sent. The bookmark will report the ResourceVersion (RV) corresponding to the set of objects, and be marked with `"k8s.io/initial-events-end": "true"` annotation. Afterwards, the watch stream will proceed as usual, sending watch events corresponding to changes (subsequent to the RV) to objects watched.
  When `sendInitialEvents` option is set, we require `resourceVersionMatch` option to also be set. The semantic of the watch request is as following: - `resourceVersionMatch` = NotOlderThan is interpreted as "data at least as new as the provided `resourceVersion`" and the bookmark event is send when the state is synced to a `resourceVersion` at least as fresh as the one provided by the ListOptions. If `resourceVersion` is unset, this is interpreted as "consistent read" and the bookmark

event is send when the state is synced at least to the moment when request started being processed.

- `resourceVersionMatch` set to any other value or unset Invalid error is returned.

Defaults to true if `resourceVersion=""` or `resourceVersion="0"` (for backward compatibility reasons) and to false otherwise.

- `timeoutSeconds` (*in query*): `integer`
  Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

Request Body

DeleteOptions

Response

- `200` Status: OK
- `401` : Unauthorized

NodeList

NodeList is the whole list of all Nodes which have been registered with master.

- `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 ↗
- `items`: `[]Node`
  List of nodes
- `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`: ListMeta
  ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.

Node

Node is a worker node in Kubernetes. Each node will have a unique identifier in the cache (i.e. in etcd).

- `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`: NodeSpec
  NodeSpec describes the attributes that a node is created with.
- `status`: NodeStatus
  NodeStatus is information about the current status of a node.

NodeSpec

NodeSpec describes the attributes that a node is created with.

- `configSource`: NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

- `externalID` : `string`
  Deprecated. Not all kubelets will set this field. Remove field after 1.13. see: https://issues.k8s.io/61966
- `podCIDR` : `string`
  PodCIDR represents the pod IP range assigned to the node.
- `podCIDRs` : `[]string`
  podCIDRs represents the IP ranges assigned to the node for usage by Pods on that node. If this field is specified, the 0th entry must match the podCIDR field. It may contain at most 1 value for each of IPv4 and IPv6.
- `providerID` : `string`
  ID of the node assigned by the cloud provider in the format: ://
- `taints` : `[]Taint`
  If specified, the node's taints.
- `unschedulable` : `boolean`
  Unschedulable controls node schedulability of new pods. By default, node is schedulable. More info: https://kubernetes.io/docs/concepts/nodes/node/#manual-node-administration

NodeConfigSource

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

- `configMap` : ConfigMapNodeConfigSource
  ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration

ConfigMapNodeConfigSource

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration

- `kubeletConfigKey` : `string`
  KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure This field is required in all cases.
- `name` : `string`
  Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.
- `namespace` : `string`
  Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.
- `resourceVersion` : `string`
  ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.
- `uid` : `string`
  UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

Taint

The node this Taint is attached to has the "effect" on any pod that does not tolerate the Taint.

- `effect` : `string`
  Required. The effect of the taint on pods that do not tolerate the taint. Valid effects are NoSchedule, PreferNoSchedule and NoExecute.
  Possible enum values:
  - `"NoExecute"` Evict any already-running pods that do not tolerate the taint. Currently enforced by NodeController.
  - `"NoSchedule"` Do not allow new pods to schedule onto the node unless they tolerate the taint, but allow all pods submitted to Kubelet without going through the scheduler to start, and allow all already-running pods to continue running. Enforced by the scheduler.
  - `"PreferNoSchedule"` Like TaintEffectNoSchedule, but the scheduler tries not to schedule new pods onto the node, rather than prohibiting new pods from scheduling onto the node entirely. Enforced by the scheduler.
- `key` : `string`

Required. The taint key to be applied to a node.

- `timeAdded`: `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.
- `value`: `string`
  The taint value corresponding to the taint key.

Time

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.

NodeStatus

NodeStatus is information about the current status of a node.

- `addresses`: `[]NodeAddress`
  List of addresses reachable to the node. Queried from cloud provider, if available. More info: https://kubernetes.io/docs/reference/node/node-status/#addresses ↗ Note: This field is declared as mergeable, but the merge key is not sufficiently unique, which can cause data corruption when it is merged. Callers should instead use a full-replacement patch. See https://pr.k8s.io/79391 ↗ for an example. Consumers should assume that addresses can change during the lifetime of a Node. However, there are some exceptions where this may not be possible, such as Pods that inherit a Node's address in its own status or consumers of the downward API (status.hostIP).
- `allocatable`: `map[string]Quantity`
  Allocatable represents the resources of a node that are available for scheduling. Defaults to Capacity.
- `capacity`: `map[string]Quantity`
  Capacity represents the total resources of a node. More info: https://kubernetes.io/docs/reference/node/node-status/#capacity ↗
- `conditions`: `[]NodeCondition`

Conditions is an array of current observed node conditions. More info: https://kubernetes.io/docs/reference/node/node-status/#condition ↗

- `config`: NodeConfigStatus
  NodeConfigStatus describes the status of the config assigned by Node.Spec.ConfigSource.
- `daemonEndpoints`: NodeDaemonEndpoints
  NodeDaemonEndpoints lists ports opened by daemons running on the Node.
- `features`: NodeFeatures
  NodeFeatures describes the set of features implemented by the CRI implementation. The features contained in the NodeFeatures should depend only on the cri implementation independent of runtime handlers.
- `images`: `[]ContainerImage`
  List of container images on this node
- `nodeInfo`: NodeSystemInfo
  NodeSystemInfo is a set of ids/uuids to uniquely identify the node.
- `phase`: `string`
  NodePhase is the recently observed lifecycle phase of the node. More info: https://kubernetes.io/docs/concepts/nodes/node/#phase ↗ The field is never populated, and now is deprecated.
  Possible enum values:
  - `"Pending"` means the node has been created/added by the system, but not configured.
  - `"Running"` means the node has been configured and has Kubernetes components running.
  - `"Terminated"` means the node has been removed from the cluster.
- `runtimeHandlers`: `[]NodeRuntimeHandler`
  The available runtime handlers.
- `volumesAttached`: `[]AttachedVolume`
  List of volumes that are attached to the node.
- `volumesInUse`: `[]string`
  List of attachable volumes in use (mounted) by the node.

NodeAddress

NodeAddress contains information for the node's address.

- `address`: `string`
  The node address.
- `type`: `string`
  Node address type, one of Hostname, ExternalIP or InternalIP.

Quantity

Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML, in addition to String() and AsInt64() accessors.

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
```

NodeCondition

NodeCondition contains condition information for a node.

- `lastHeartbeatTime`: `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.

- `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`
  Human readable message indicating details about last transition.
- `reason` : `string`
  (brief) reason for the condition's last transition.
- `status` : `string`
  Status of the condition, one of True, False, Unknown.
- `type` : `string`
  Type of node condition.

NodeConfigStatus

NodeConfigStatus describes the status of the config assigned by Node.Spec.ConfigSource.

- `active` : NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22
- `assigned` : NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22
- `error` : `string`
  Error describes any problems reconciling the Spec.ConfigSource to the Active config. Errors may occur, for example, attempting to checkpoint Spec.ConfigSource to the local Assigned record, attempting to checkpoint the payload associated with Spec.ConfigSource, attempting to load or validate the Assigned config, etc. Errors may occur at different points while syncing config. Earlier errors (e.g. download or checkpointing errors) will not result in a rollback to LastKnownGood, and may resolve across Kubelet retries. Later errors (e.g. loading or validating a checkpointed config) will result in a rollback to LastKnownGood. In

the latter case, it is usually possible to resolve the error by fixing the config assigned in Spec.ConfigSource. You can find additional information for debugging by searching the error message in the Kubelet log. Error is a human-readable description of the error state; machines can check whether or not Error is empty, but should not rely on the stability of the Error text across Kubelet versions.

- `lastKnownGood`: NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

NodeDaemonEndpoints

NodeDaemonEndpoints lists ports opened by daemons running on the Node.

- `kubeletEndpoint`: DaemonEndpoint
  DaemonEndpoint contains information about a single Daemon endpoint.

DaemonEndpoint

DaemonEndpoint contains information about a single Daemon endpoint.

- `Port`: `integer`
  Port number of the given endpoint.

NodeFeatures

NodeFeatures describes the set of features implemented by the CRI implementation. The features contained in the NodeFeatures should depend only on the cri implementation independent of runtime handlers.

- `supplementalGroupsPolicy`: `boolean`
  SupplementalGroupsPolicy is set to true if the runtime supports SupplementalGroupsPolicy and ContainerUser.

ContainerImage

Describe a container image

- `names` : `[]string`
  Names by which this image is known. e.g. ["kubernetes.example/hyperkube:v1.0.7", "cloud-vendor.registry.example/cloud-vendor/hyperkube:v1.0.7"]
- `sizeBytes` : `integer`
  The size of the image in bytes.

NodeSystemInfo

NodeSystemInfo is a set of ids/uuids to uniquely identify the node.

- `architecture` : `string`
  The Architecture reported by the node
- `bootID` : `string`
  Boot ID reported by the node.
- `containerRuntimeVersion` : `string`
  ContainerRuntime Version reported by the node through runtime remote API (e.g. containerd://1.4.2).
- `kernelVersion` : `string`
  Kernel Version reported by the node from 'uname -r' (e.g. 3.16.0-0.bpo.4-amd64).
- `kubeProxyVersion` : `string`
  Deprecated: KubeProxy Version reported by the node.
- `kubeletVersion` : `string`
  Kubelet Version reported by the node.
- `machineID` : `string`
  MachineID reported by the node. For unique machine identification in the cluster this field is preferred. Learn more from man(5) machine-id: http://man7.org/linux/man-pages/man5/machine-id.5.html ↗
- `operatingSystem` : `string`
  The Operating System reported by the node

- `osImage` : `string`
  OS Image reported by the node from /etc/os-release (e.g. Debian GNU/Linux 7 (wheezy)).
- `systemUUID` : `string`
  SystemUUID reported by the node. For unique machine identification MachineID is preferred. This field is specific to Red Hat hosts https://access.redhat.com/documentation/en-us/red_hat_subscription_management/1/html/rhsm/uuid ↗

NodeRuntimeHandler

NodeRuntimeHandler is a set of runtime handler information.

- `features` : NodeRuntimeHandlerFeatures
  NodeRuntimeHandlerFeatures is a set of features implemented by the runtime handler.
- `name` : `string`
  Runtime handler name. Empty for the default runtime handler.

NodeRuntimeHandlerFeatures

NodeRuntimeHandlerFeatures is a set of features implemented by the runtime handler.

- `recursiveReadOnlyMounts` : `boolean`
  RecursiveReadOnlyMounts is set to true if the runtime handler supports RecursiveReadOnlyMounts.
- `userNamespaces` : `boolean`
  UserNamespaces is set to true if the runtime handler supports UserNamespaces, including for volumes.

AttachedVolume

AttachedVolume describes a volume attached to a node

- `devicePath` : `string`

DevicePath represents the device path where the volume should be available

- `name`: `string`
  Name of the attached volume

ListMeta

ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.

- `continue`: `string`
  continue may be set if the user set a limit on the number of items returned, and indicates that the server has more data available. The value is opaque and may be used to issue another request to the endpoint that served this list to retrieve the next set of available objects. Continuing a consistent list may not be possible if the server configuration has changed or more than a few minutes have passed. The resourceVersion field returned when using this continue value will be identical to the value in the first response, unless you have received this token from an error message.
- `remainingItemCount`: `integer`
  remainingItemCount is the number of subsequent items in the list which are not included in this list response. If the list request contained label or field selectors, then the number of remaining items is unknown and the field will be left unset and omitted during serialization. If the list is complete (either because it is not chunking or because this is the last chunk), then there are no more remaining items and this field will be left unset and omitted during serialization. Servers older than v1.15 do not set this field. The intended use of the remainingItemCount is *estimating* the size of a collection. Clients should not rely on the remainingItemCount to be set or to be exact.
- `resourceVersion`: `string`
  String that identifies the server's internal version of this object that can be used by clients to determine when objects have changed. Value must be treated as opaque by clients and passed unmodified back to the server. Populated by the system. Read-only. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗
- `selfLink`: `string`
  Deprecated: selfLink is a legacy read-only field that is no longer populated by the system.

Status

Status is a return value for calls that don't return other objects.

- `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 ↗
- `code` : `integer`
  Suggested HTTP return code for this status, 0 if not set.
- `details` : StatusDetails
  StatusDetails is a set of additional properties that MAY be set by the server to provide additional information about a response. The Reason field of a Status object defines what attributes will be set. Clients must ignore fields that do not match the defined type of each attribute, and should assume that any attribute may be empty, invalid, or under defined.
- `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 ↗
- `message` : `string`
  A human-readable description of the status of this operation.
- `metadata` : ListMeta
  ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.
- `reason` : `string`
  A machine-readable description of why this operation is in the "Failure" status. If this value is empty there is no information available. A Reason clarifies an HTTP status code but does not override it.
- `status` : `string`
  Status of the operation. One of: "Success" or "Failure". More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status ↗

StatusDetails

StatusDetails is a set of additional properties that MAY be set by the server to provide additional information about a response. The Reason field of a Status object defines what attributes will be set. Clients must ignore fields that do not match the defined type of each attribute, and should assume that any attribute may be empty, invalid, or under defined.

- `causes` : `[]StatusCause`
  The Causes array includes more details associated with the StatusReason failure. Not all StatusReasons may provide detailed causes.
- `group` : `string`
  The group attribute of the resource associated with the status StatusReason.
- `kind` : `string`
  The kind attribute of the resource associated with the status StatusReason. On some operations may differ from the requested resource Kind. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
- `name` : `string`
  The name attribute of the resource associated with the status StatusReason (when there is a single name which can be described).
- `retryAfterSeconds` : `integer`
  If specified, the time in seconds before the operation should be retried. Some errors may indicate the client must take an alternate action - for those errors this field may indicate how long to wait before taking the alternate action.
- `uid` : `string`
  UID of the resource. (when there is a single resource which can be described). More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names#uids ↗

StatusCause

StatusCause provides more information about an api.Status failure, including cases when multiple errors are encountered.

- `field` : `string`

The field of the resource that has caused this error, as named by its JSON serialization. May include dot and postfix notation for nested attributes. Arrays are zero-indexed. Fields may appear more than once in an array of causes due to fields having multiple errors. Optional.
Examples: "name" - the field "name" on the current resource "items[0].name" - the field "name" on the first array entry in "items"

- `message` : `string`
  A human-readable description of the cause of the error. This field may be presented as-is to a reader.
- `reason` : `string`
  A machine-readable description of the cause of the error. If this value is empty there is no information available.

/kubernetes/{cluster}/api/v1/nodes/{name}

Common Parameters

- `name` (*in path*): `string` required
  name of the Node
- `pretty` (*in query*): `string`
  If 'true', then the output is pretty printed. Defaults to 'false' unless the user-agent indicates a browser or command-line HTTP tool (curl and wget).

`get`

read the specified Node

Response

- `200` Node: OK
- `401` : Unauthorized

`put`

replace the specified Node

Parameters

- `dryRun` (*in query*): `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
- `fieldManager` (*in query*): `string`
  fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint ↗.
- `fieldValidation` (*in query*): `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.

Request Body

Node

Response

- `200` Node: OK
- `201` Node: Created
- `401` : Unauthorized

`delete`

delete a Node

Parameters

- `dryRun` (*in query*): `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
- `gracePeriodSeconds` (*in query*): `integer`
  The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.
- `ignoreStoreReadErrorWithClusterBreakingPotential` (*in query*): `boolean`
  if set to true, it will trigger an unsafe deletion of the resource in case the normal deletion flow fails with a corrupt object error. A resource is considered corrupt if it can not be retrieved from the underlying storage successfully because of a) its data can not be transformed e.g. decryption failure, or b) it fails to decode into an object. NOTE: unsafe deletion ignores finalizer constraints, skips precondition checks, and removes the object from the storage. WARNING: This may potentially break the cluster if the workload associated with the resource being unsafe-deleted relies on normal deletion flow. Use only if you REALLY know what you are doing. The default value is false, and the user must opt in to enable it
- `orphanDependents` (*in query*): `boolean`
  Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added to/removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.
- `propagationPolicy` (*in query*): `string`
  Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

Request Body

DeleteOptions

Response

- `200` Status: OK
- `202` Status: Accepted
- `401` : Unauthorized

`patch`

partially update the specified Node

Parameters

- `dryRun` (*in query*): `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
- `fieldManager` (*in query*): `string`
  fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by https://golang.org/pkg/unicode/#IsPrint ↗. This field is required for apply requests (application/apply-patch) but optional for non-apply patch types (JsonPatch, MergePatch, StrategicMergePatch).
- `fieldValidation` (*in query*): `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.

- `force` (*in query*): `boolean`
  Force is going to "force" Apply requests. It means user will re-acquire conflicting fields owned by other people. Force flag must be unset for non-apply patch requests.

Request Body

Patch

Response

- `200` Node: OK
- `201` Node: Created
- `401` : Unauthorized

Node

Node is a worker node in Kubernetes. Each node will have a unique identifier in the cache (i.e. in etcd).

- `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`: NodeSpec
  NodeSpec describes the attributes that a node is created with.
- `status`: NodeStatus
  NodeStatus is information about the current status of a node.

NodeSpec

NodeSpec describes the attributes that a node is created with.

- `configSource`: NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22
- `externalID`: `string`
  Deprecated. Not all kubelets will set this field. Remove field after 1.13. see: https://issues.k8s.io/61966 ↗
- `podCIDR`: `string`
  PodCIDR represents the pod IP range assigned to the node.
- `podCIDRs`: `[]string`
  podCIDRs represents the IP ranges assigned to the node for usage by Pods on that node. If this field is specified, the 0th entry must match the podCIDR field. It may contain at most 1 value for each of IPv4 and IPv6.
- `providerID`: `string`
  ID of the node assigned by the cloud provider in the format: ://
- `taints`: `[]Taint`
  If specified, the node's taints.
- `unschedulable`: `boolean`
  Unschedulable controls node schedulability of new pods. By default, node is schedulable. More info: https://kubernetes.io/docs/concepts/nodes/node/#manual-node-administration ↗

NodeConfigSource

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

- `configMap`: ConfigMapNodeConfigSource
  ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration ↗

ConfigMapNodeConfigSource

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration ↗

- `kubeletConfigKey`: `string`
  KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure This field is required in all cases.
- `name`: `string`
  Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.
- `namespace`: `string`
  Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.
- `resourceVersion`: `string`
  ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.
- `uid`: `string`
  UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

Taint

The node this Taint is attached to has the "effect" on any pod that does not tolerate the Taint.

- `effect` : `string`
  Required. The effect of the taint on pods that do not tolerate the taint. Valid effects are NoSchedule, PreferNoSchedule and NoExecute.
  Possible enum values:
  - `"NoExecute"` Evict any already-running pods that do not tolerate the taint. Currently enforced by NodeController.
  - `"NoSchedule"` Do not allow new pods to schedule onto the node unless they tolerate the taint, but allow all pods submitted to Kubelet without going through the scheduler to start, and allow all already-running pods to continue running. Enforced by the scheduler.
  - `"PreferNoSchedule"` Like TaintEffectNoSchedule, but the scheduler tries not to schedule new pods onto the node, rather than prohibiting new pods from scheduling onto the node entirely. Enforced by the scheduler.
- `key` : `string`
  Required. The taint key to be applied to a node.
- `timeAdded` : `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.
- `value` : `string`
  The taint value corresponding to the taint key.

Time

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.

NodeStatus

NodeStatus is information about the current status of a node.

- `addresses` : `[]NodeAddress`

List of addresses reachable to the node. Queried from cloud provider, if available. More info: https://kubernetes.io/docs/reference/node/node-status/#addresses ↗ Note: This field is declared as mergeable, but the merge key is not sufficiently unique, which can cause data corruption when it is merged. Callers should instead use a full-replacement patch. See https://pr.k8s.io/79391 ↗ for an example. Consumers should assume that addresses can change during the lifetime of a Node. However, there are some exceptions where this may not be possible, such as Pods that inherit a Node's address in its own status or consumers of the downward API (status.hostIP).

- `allocatable` : `map[string]Quantity`
  Allocatable represents the resources of a node that are available for scheduling. Defaults to Capacity.
- `capacity` : `map[string]Quantity`
  Capacity represents the total resources of a node. More info: https://kubernetes.io/docs/reference/node/node-status/#capacity ↗
- `conditions` : `[]NodeCondition`
  Conditions is an array of current observed node conditions. More info: https://kubernetes.io/docs/reference/node/node-status/#condition ↗
- `config` : NodeConfigStatus
  NodeConfigStatus describes the status of the config assigned by Node.Spec.ConfigSource.
- `daemonEndpoints` : NodeDaemonEndpoints
  NodeDaemonEndpoints lists ports opened by daemons running on the Node.
- `features` : NodeFeatures
  NodeFeatures describes the set of features implemented by the CRI implementation. The features contained in the NodeFeatures should depend only on the cri implementation independent of runtime handlers.
- `images` : `[]ContainerImage`
  List of container images on this node
- `nodeInfo` : NodeSystemInfo
  NodeSystemInfo is a set of ids/uuids to uniquely identify the node.
- `phase` : `string`
  NodePhase is the recently observed lifecycle phase of the node. More info: https://kubernetes.io/docs/concepts/nodes/node/#phase ↗ The field is never populated, and now is deprecated.

Possible enum values:

- `"Pending"` means the node has been created/added by the system, but not configured.
- `"Running"` means the node has been configured and has Kubernetes components running.
- `"Terminated"` means the node has been removed from the cluster.

- `runtimeHandlers`: `[]NodeRuntimeHandler`
  The available runtime handlers.
- `volumesAttached`: `[]AttachedVolume`
  List of volumes that are attached to the node.
- `volumesInUse`: `[]string`
  List of attachable volumes in use (mounted) by the node.

NodeAddress

NodeAddress contains information for the node's address.

- `address`: `string`
  The node address.
- `type`: `string`
  Node address type, one of Hostname, ExternalIP or InternalIP.

Quantity

Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML, in addition to String() and AsInt64() accessors.

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
```

NodeCondition

NodeCondition contains condition information for a node.

- `lastHeartbeatTime`: `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.

- `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`
  Human readable message indicating details about last transition.
- `reason` : `string`
  (brief) reason for the condition's last transition.
- `status` : `string`
  Status of the condition, one of True, False, Unknown.
- `type` : `string`
  Type of node condition.

NodeConfigStatus

NodeConfigStatus describes the status of the config assigned by Node.Spec.ConfigSource.

- `active` : NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22
- `assigned` : NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22
- `error` : `string`
  Error describes any problems reconciling the Spec.ConfigSource to the Active config. Errors may occur, for example, attempting to checkpoint Spec.ConfigSource to the local Assigned record, attempting to checkpoint the payload associated with Spec.ConfigSource, attempting to load or validate the Assigned config, etc. Errors may occur at different points while syncing config. Earlier errors (e.g. download or checkpointing errors) will not result in a rollback to LastKnownGood, and may resolve across Kubelet retries. Later errors (e.g. loading or validating a checkpointed config) will result in a rollback to LastKnownGood. In

the latter case, it is usually possible to resolve the error by fixing the config assigned in Spec.ConfigSource. You can find additional information for debugging by searching the error message in the Kubelet log. Error is a human-readable description of the error state; machines can check whether or not Error is empty, but should not rely on the stability of the Error text across Kubelet versions.

- `lastKnownGood`: NodeConfigSource
  NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

NodeDaemonEndpoints

NodeDaemonEndpoints lists ports opened by daemons running on the Node.

- `kubeletEndpoint`: DaemonEndpoint
  DaemonEndpoint contains information about a single Daemon endpoint.

DaemonEndpoint

DaemonEndpoint contains information about a single Daemon endpoint.

- `Port`: `integer`
  Port number of the given endpoint.

NodeFeatures

NodeFeatures describes the set of features implemented by the CRI implementation. The features contained in the NodeFeatures should depend only on the cri implementation independent of runtime handlers.

- `supplementalGroupsPolicy`: `boolean`
  SupplementalGroupsPolicy is set to true if the runtime supports SupplementalGroupsPolicy and ContainerUser.

ContainerImage

Describe a container image

- `names` : `[]string`
  Names by which this image is known. e.g. ["kubernetes.example/hyperkube:v1.0.7", "cloud-vendor.registry.example/cloud-vendor/hyperkube:v1.0.7"]
- `sizeBytes` : `integer`
  The size of the image in bytes.

NodeSystemInfo

NodeSystemInfo is a set of ids/uuids to uniquely identify the node.

- `architecture` : `string`
  The Architecture reported by the node
- `bootID` : `string`
  Boot ID reported by the node.
- `containerRuntimeVersion` : `string`
  ContainerRuntime Version reported by the node through runtime remote API (e.g. containerd://1.4.2).
- `kernelVersion` : `string`
  Kernel Version reported by the node from 'uname -r' (e.g. 3.16.0-0.bpo.4-amd64).
- `kubeProxyVersion` : `string`
  Deprecated: KubeProxy Version reported by the node.
- `kubeletVersion` : `string`
  Kubelet Version reported by the node.
- `machineID` : `string`
  MachineID reported by the node. For unique machine identification in the cluster this field is preferred. Learn more from man(5) machine-id: http://man7.org/linux/man-pages/man5/machine-id.5.html ↗
- `operatingSystem` : `string`
  The Operating System reported by the node

- `osImage` : `string`
  OS Image reported by the node from /etc/os-release (e.g. Debian GNU/Linux 7 (wheezy)).
- `systemUUID` : `string`
  SystemUUID reported by the node. For unique machine identification MachineID is preferred. This field is specific to Red Hat hosts https://access.redhat.com/documentation/en-us/red_hat_subscription_management/1/html/rhsm/uuid ↗

NodeRuntimeHandler

NodeRuntimeHandler is a set of runtime handler information.

- `features` : NodeRuntimeHandlerFeatures
  NodeRuntimeHandlerFeatures is a set of features implemented by the runtime handler.
- `name` : `string`
  Runtime handler name. Empty for the default runtime handler.

NodeRuntimeHandlerFeatures

NodeRuntimeHandlerFeatures is a set of features implemented by the runtime handler.

- `recursiveReadOnlyMounts` : `boolean`
  RecursiveReadOnlyMounts is set to true if the runtime handler supports RecursiveReadOnlyMounts.
- `userNamespaces` : `boolean`
  UserNamespaces is set to true if the runtime handler supports UserNamespaces, including for volumes.

AttachedVolume

AttachedVolume describes a volume attached to a node

- `devicePath` : `string`

DevicePath represents the device path where the volume should be available

- `name` : `string`
  Name of the attached volume

Status

Status is a return value for calls that don't return other objects.

- `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 ↗
- `code` : `integer`
  Suggested HTTP return code for this status, 0 if not set.
- `details` : StatusDetails
  StatusDetails is a set of additional properties that MAY be set by the server to provide additional information about a response. The Reason field of a Status object defines what attributes will be set. Clients must ignore fields that do not match the defined type of each attribute, and should assume that any attribute may be empty, invalid, or under defined.
- `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 ↗
- `message` : `string`
  A human-readable description of the status of this operation.
- `metadata` : ListMeta
  ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.
- `reason` : `string`
  A machine-readable description of why this operation is in the "Failure" status. If this value is empty there is no information available. A Reason clarifies an HTTP status code but does not override it.

- `status` : `string`
  Status of the operation. One of: "Success" or "Failure". More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status ↗

StatusDetails

StatusDetails is a set of additional properties that MAY be set by the server to provide additional information about a response. The Reason field of a Status object defines what attributes will be set. Clients must ignore fields that do not match the defined type of each attribute, and should assume that any attribute may be empty, invalid, or under defined.

- `causes` : `[]StatusCause`
  The Causes array includes more details associated with the StatusReason failure. Not all StatusReasons may provide detailed causes.
- `group` : `string`
  The group attribute of the resource associated with the status StatusReason.
- `kind` : `string`
  The kind attribute of the resource associated with the status StatusReason. On some operations may differ from the requested resource Kind. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds ↗
- `name` : `string`
  The name attribute of the resource associated with the status StatusReason (when there is a single name which can be described).
- `retryAfterSeconds` : `integer`
  If specified, the time in seconds before the operation should be retried. Some errors may indicate the client must take an alternate action - for those errors this field may indicate how long to wait before taking the alternate action.
- `uid` : `string`
  UID of the resource. (when there is a single resource which can be described). More info: https://kubernetes.io/docs/concepts/overview/working-with-objects/names#uids ↗

StatusCause

StatusCause provides more information about an api.Status failure, including cases when multiple errors are encountered.

- `field` : `string`
  The field of the resource that has caused this error, as named by its JSON serialization. May include dot and postfix notation for nested attributes. Arrays are zero-indexed. Fields may appear more than once in an array of causes due to fields having multiple errors. Optional.
  Examples: "name" - the field "name" on the current resource "items[0].name" - the field "name" on the first array entry in "items"
- `message` : `string`
  A human-readable description of the cause of the error. This field may be presented as-is to a reader.
- `reason` : `string`
  A machine-readable description of the cause of the error. If this value is empty there is no information available.

ListMeta

ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.

- `continue` : `string`
  continue may be set if the user set a limit on the number of items returned, and indicates that the server has more data available. The value is opaque and may be used to issue another request to the endpoint that served this list to retrieve the next set of available objects. Continuing a consistent list may not be possible if the server configuration has changed or more than a few minutes have passed. The resourceVersion field returned when using this continue value will be identical to the value in the first response, unless you have received this token from an error message.
- `remainingItemCount` : `integer`
  remainingItemCount is the number of subsequent items in the list which are not included in this list response. If the list request contained label or field selectors, then the number of

remaining items is unknown and the field will be left unset and omitted during serialization. If the list is complete (either because it is not chunking or because this is the last chunk), then there are no more remaining items and this field will be left unset and omitted during serialization. Servers older than v1.15 do not set this field. The intended use of the remainingItemCount is *estimating* the size of a collection. Clients should not rely on the remainingItemCount to be set or to be exact.

- `resourceVersion`: `string`
  String that identifies the server's internal version of this object that can be used by clients to determine when objects have changed. Value must be treated as opaque by clients and passed unmodified back to the server. Populated by the system. Read-only. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency ↗
- `selfLink`: `string`
  Deprecated: selfLink is a legacy read-only field that is no longer populated by the system.

Patch

Patch is provided to give a concrete name and type to the Kubernetes PATCH request body.