The Kubernetes API

Archived: 2026-04-05 20:23:27 UTC

The Kubernetes API lets you query and manipulate the state of objects in Kubernetes. The core of Kubernetes' control plane is the API server and the HTTP API that it exposes. Users, the different parts of your cluster, and external components all communicate with one another through the API server.

The core of Kubernetes' control plane is the API server. The API server exposes an HTTP API that lets end users, different parts of your cluster, and external components communicate with one another.

The Kubernetes API lets you query and manipulate the state of API objects in Kubernetes (for example: Pods, Namespaces, ConfigMaps, and Events).

Most operations can be performed through the kubectl command-line interface or other command-line tools, such as kubeadm, which in turn use the API. However, you can also access the API directly using REST calls. Kubernetes provides a set of client libraries for those looking to write applications using the Kubernetes API.

Each Kubernetes cluster publishes the specification of the APIs that the cluster serves. There are two mechanisms that Kubernetes uses to publish these API specifications; both are useful to enable automatic interoperability. For example, the `kubectl` tool fetches and caches the API specification for enabling command-line completion and other features. The two supported mechanisms are as follows:

- The Discovery API provides information about the Kubernetes APIs: API names, resources, versions, and supported operations. This is a Kubernetes specific term as it is a separate API from the Kubernetes OpenAPI. It is intended to be a brief summary of the available resources and it does not detail specific schema for the resources. For reference about resource schemas, please refer to the OpenAPI document.
- The Kubernetes OpenAPI Document provides (full) OpenAPI v2.0 and 3.0 schemas for all Kubernetes API endpoints. The OpenAPI v3 is the preferred method for accessing OpenAPI as it provides a more comprehensive and accurate view of the API. It includes all the available API paths, as well as all resources consumed and produced for every operations on every endpoints. It also includes any extensibility components that a cluster supports. The data is a complete specification and is significantly larger than that from the Discovery API.

Discovery API

Kubernetes publishes a list of all group versions and resources supported via the Discovery API. This includes the following for each resource:

- Name
- Cluster or namespaced scope
- Endpoint URL and supported verbs
- Alternative names

- Group, version, kind

The API is available in both aggregated and unaggregated form. The aggregated discovery serves two endpoints, while the unaggregated discovery serves a separate endpoint for each group version.

Aggregated discovery

FEATURE STATE: `Kubernetes v1.30 [stable]` (enabled by default)

Kubernetes offers stable support for *aggregated discovery*, publishing all resources supported by a cluster through two endpoints (`/api` and `/apis`). Requesting this endpoint drastically reduces the number of requests sent to fetch the discovery data from the cluster. You can access the data by requesting the respective endpoints with an `Accept` header indicating the aggregated discovery resource: `Accept: application/json;v=v2;g=apidiscovery.k8s.io;as=APIGroupDiscoveryList` .

Without indicating the resource type using the `Accept` header, the default response for the `/api` and `/apis` endpoint is an unaggregated discovery document.

The [discovery document] for the built-in resources can be found in the Kubernetes GitHub repository. This Github document can be used as a reference of the base set of the available resources if a Kubernetes cluster is not available to query.

The endpoint also supports ETag and protobuf encoding.

Unaggregated discovery

Without discovery aggregation, discovery is published in levels, with the root endpoints publishing discovery information for downstream documents.

A list of all group versions supported by a cluster is published at the `/api` and `/apis` endpoints. Example:

```
{
 "kind": "APIGroupList",
 "apiVersion": "v1",
 "groups": [
   {
     "name": "apiregistration.k8s.io",
     "versions": [
       {
         "groupVersion": "apiregistration.k8s.io/v1",
         "version": "v1"
       }
     ],
     "preferredVersion": {
       "groupVersion": "apiregistration.k8s.io/v1",
       "version": "v1"
     }
```

```
    },
    {
      "name": "apps",
      "versions": [
        {
          "groupVersion": "apps/v1",
          "version": "v1"
        }
      ],
      "preferredVersion": {
        "groupVersion": "apps/v1",
        "version": "v1"
      }
    },
    ...
}
```

Additional requests are needed to obtain the discovery document for each group version at `/apis/<group>/<version>` (for example: `/apis/rbac.authorization.k8s.io/v1alpha1`), which advertises the list of resources served under a particular group version. These endpoints are used by kubectl to fetch the list of resources supported by a cluster.

OpenAPI interface definition

For details about the OpenAPI specifications, see the OpenAPI documentation.

Kubernetes serves both OpenAPI v2.0 and OpenAPI v3.0. OpenAPI v3 is the preferred method of accessing the OpenAPI because it offers a more comprehensive (lossless) representation of Kubernetes resources. Due to limitations of OpenAPI version 2, certain fields are dropped from the published OpenAPI including but not limited to `default`, `nullable`, `oneOf`.

OpenAPI V2

The Kubernetes API server serves an aggregated OpenAPI v2 spec via the `/openapi/v2` endpoint. You can request the response format using request headers as follows:

Header	Possible values	Notes
`Accept-Encoding`	`gzip`	*not supplying this header is also acceptable*
`Accept`	`application/com.github.proto-openapi.spec.v2@v1.0+protobuf`	*mainly for intra-cluster use*
	`application/json`	*default*

Header	Possible values	Notes
	`*`	*serves* `application/json`

Warning:

The validation rules published as part of OpenAPI schemas may not be complete, and usually aren't. Additional validation occurs within the API server. If you want precise and complete verification, a `kubectl apply --dry-run=server` runs all the applicable validation (and also activates admission-time checks).

OpenAPI V3

FEATURE STATE: `Kubernetes v1.27 [stable]` (enabled by default)

Kubernetes supports publishing a description of its APIs as OpenAPI v3.

A discovery endpoint `/openapi/v3` is provided to see a list of all group/versions available. This endpoint only returns JSON. These group/versions are provided in the following format:

```
 {
    "paths": {
        ...,
        "api/v1": {
            "serverRelativeURL": "/openapi/v3/api/v1?hash=CC0E9BFD992D8C59AEC98A1E2336F899E8318D3CF4C68944C3DEC(
        },
        "apis/admissionregistration.k8s.io/v1": {
            "serverRelativeURL": "/openapi/v3/apis/admissionregistration.k8s.io/v1?hash=E19CC93A116982CE5422FC4
        },
        ....
    }
}
```

The relative URLs are pointing to immutable OpenAPI descriptions, in order to improve client-side caching. The proper HTTP caching headers are also set by the API server for that purpose (`Expires` to 1 year in the future, and `Cache-Control` to `immutable`). When an obsolete URL is used, the API server returns a redirect to the newest URL.

The Kubernetes API server publishes an OpenAPI v3 spec per Kubernetes group version at the `/openapi/v3/apis/<group>/<version>?hash=<hash>` endpoint.

Refer to the table below for accepted request headers.

Header	Possible values	Notes
`Accept-Encoding`	`gzip`	*not supplying this header is also acceptable*
`Accept`	`application/com.github.proto-openapi.spec.v3@v1.0+protobuf`	*mainly for intra-cluster use*
	`application/json`	*default*
	`*`	*serves* `application/json`

A Golang implementation to fetch the OpenAPI V3 is provided in the package `k8s.io/client-go/openapi3`.

Kubernetes 1.35 publishes OpenAPI v2.0 and v3.0; there are no plans to support 3.1 in the near future.

Protobuf serialization

Kubernetes implements an alternative Protobuf based serialization format that is primarily intended for intra-cluster communication. For more information about this format, see the Kubernetes Protobuf serialization design proposal and the Interface Definition Language (IDL) files for each schema located in the Go packages that define the API objects.

Persistence

Kubernetes stores the serialized state of objects by writing them into etcd.

API groups and versioning

To make it easier to eliminate fields or restructure resource representations, Kubernetes supports multiple API versions, each at a different API path, such as `/api/v1` or `/apis/rbac.authorization.k8s.io/v1alpha1`.

Versioning is done at the API level rather than at the resource or field level to ensure that the API presents a clear, consistent view of system resources and behavior, and to enable controlling access to end-of-life and/or experimental APIs.

To make it easier to evolve and to extend its API, Kubernetes implements API groups that can be enabled or disabled.

API resources are distinguished by their API group, resource type, namespace (for namespaced resources), and name. The API server handles the conversion between API versions transparently: all the different versions are actually representations of the same persisted data. The API server may serve the same underlying data through multiple API versions.

For example, suppose there are two API versions, `v1` and `v1beta1`, for the same resource. If you originally created an object using the `v1beta1` version of its API, you can later read, update, or delete that object using

either the `v1beta1` or the `v1` API version, until the `v1beta1` version is deprecated and removed. At that point you can continue accessing and modifying the object using the `v1` API.

API changes

Any system that is successful needs to grow and change as new use cases emerge or existing ones change. Therefore, Kubernetes has designed the Kubernetes API to continuously change and grow. The Kubernetes project aims to *not* break compatibility with existing clients, and to maintain that compatibility for a length of time so that other projects have an opportunity to adapt.

In general, new API resources and new resource fields can be added often and frequently. Elimination of resources or fields requires following the API deprecation policy.

Kubernetes makes a strong commitment to maintain compatibility for official Kubernetes APIs once they reach general availability (GA), typically at API version `v1`. Additionally, Kubernetes maintains compatibility with data persisted via *beta* API versions of official Kubernetes APIs, and ensures that data can be converted and accessed via GA API versions when the feature goes stable.

If you adopt a beta API version, you will need to transition to a subsequent beta or stable API version once the API graduates. The best time to do this is while the beta API is in its deprecation period, since objects are simultaneously accessible via both API versions. Once the beta API completes its deprecation period and is no longer served, the replacement API version must be used.

Note:

Although Kubernetes also aims to maintain compatibility for *alpha* APIs versions, in some circumstances this is not possible. If you use any alpha API versions, check the release notes for Kubernetes when upgrading your cluster, in case the API did change in incompatible ways that require deleting all existing alpha objects prior to upgrade.

Refer to API versions reference for more details on the API version level definitions.

API Extension

The Kubernetes API can be extended in one of two ways:

1. Custom resources let you declaratively define how the API server should provide your chosen resource API.
2. You can also extend the Kubernetes API by implementing an aggregation layer.

What's next

- Learn how to extend the Kubernetes API by adding your own CustomResourceDefinition.
- Controlling Access To The Kubernetes API describes how the cluster manages authentication and authorization for API access.
- Learn about API endpoints, resource types and samples by reading API Reference.

- Learn about what constitutes a compatible change, and how to change the API, from API changes.

Source: https://kubernetes.io/docs/concepts/overview/kubernetes-api/