# Kubernetes - API

Kubernetes API serves as a foundation for declarative configuration schema for the system. **Kubectl** command-line tool can be used to create, update, delete, and get API object. Kubernetes API acts a communicator among different components of Kubernetes.

## Adding API to Kubernetes

Adding a new API to Kubernetes will add new features to Kubernetes, which will increase the functionality of Kubernetes. However, alongside it will also increase the cost and maintainability of the system. In order to create a balance between the cost and complexity, there are a few sets defined for it.

The API which is getting added should be useful to more than 50% of the users. There is no other way to implement the functionality in Kubernetes. Exceptional circumstances are discussed in the community meeting of Kubernetes, and then API is added.

## API Changes

In order to increase the capability of Kubernetes, changes are continuously introduced to the system. It is done by Kubernetes team to add the functionality to Kubernetes without removing or impacting the existing functionality of the system.

To demonstrate the general process, here is an (hypothetical) example −

A user POSTs a Pod object to **/api/v7beta1/...**

The JSON is unmarshalled into a **v7beta1.Pod** structure

Default values are applied to the **v7beta1.Pod**

The **v7beta1.Pod** is converted to an **api.Pod** structure

The **api.Pod** is validated, and any errors are returned to the user

The **api.Pod** is converted to a v6.Pod (because v6 is the latest stable version)

The **v6.Pod** is marshalled into JSON and written to **etcd**

Now that we have the Pod object stored, a user can GET that object in any supported API version. For example −

A user GETs the Pod from **/api/v5/...**

The JSON is read from **etcd** and **unmarshalled** into a **v6.Pod** structure

Default values are applied to the **v6.Pod**

The **v6.Pod** is converted to an api.Pod structure

The **api.Pod** is converted to a **v5.Pod** structure

The **v5.Pod** is marshalled into JSON and sent to the user

The implication of this process is that API changes must be done carefully and backward compatibly.

## API Versioning

To make it easier to support multiple structures, Kubernetes supports multiple API versions each at different API path such as **/api/v1** or **/apsi/extensions/v1beta1**

Versioning standards at Kubernetes are defined in multiple standards.

### Alpha Level

This version contains alpha (e.g. v1alpha1)

This version may be buggy; the enabled version may have bugs

Support for bugs can be dropped at any point of time.

Recommended to be used in short term testing only as the support may not be present all the time.

### Beta Level

The version name contains beta (e.g. v2beta3)

The code is fully tested and the enabled version is supposed to be stable.

The support of the feature will not be dropped; there may be some small changes.

Recommended for only non-business-critical uses because of the potential for incompatible changes in subsequent releases.

### Stable Level

The version name is **vX** where **X** is an integer.

Stable versions of features will appear in the released software for many subsequent versions.