Introduction to kubectl

Get introduced to kubectl in Kubernetes.

We'll cover the following

Overview

`kubectl` is a powerful command line tool for interacting with Kubernetes. We can use `kubectl` to create resources, list running objects, and more. It provides a convenient way to learn, operate, and manage Kubernetes. We can easily complete most management tasks via `kubectl`.

What is `kubectl`?

From a novice programmer's view, `kubectl` is a swiss army knife—one tool that accomplish many different tasks. It can help us manage Kubernetes, query the system to see what is happening there, display objects with our interested fields, monitor resource usages, and more.

From a veteran programmer's view, `kubectl` is a client that can talk to Kubernetes. As we all know, Kubernetes exposes RESTful HTTP APIs. These APIs are not only used by Kubernetes components, but also for external accesses. These APIs give us total leverage over Kubernetes. This kind of design helps ensure that each Kubernetes process can be revealed as an API endpoint with an HTTP request. Therefore, what `kubectl` primarily does is make HTTP requests to the `kube-apiserver`.

How to use `kubectl`

Below are all the commands in `kubectl`. We can invoke a sub-command to perform our operation. One of the most commonly used sub-commands is `get`, which can be used to get a specified object of a kind, list a group of resources, or send out a `WATCH` request to keep track of resource changes.

```
kubectl -h
```

The help command

The output will be as follows:

```
kubectl controls the Kubernetes cluster manager.

```

```
 Find more information at: https://kubernetes.io/docs/reference/kubectl/overview/

Basic Commands (Beginner):
  create          Create a resource from a file or from stdin
  expose          Take a replication controller, service, deployment or pod and expose it as a new Kuber
  run             Run a particular image on the cluster
  set             Set specific features on objects

Basic Commands (Intermediate):
  explain         Get documentation for a resource
  get             Display one or many resources
  edit            Edit a resource on the server
  delete          Delete resources by file names, stdin, resources and names, or by resources and label

Deploy Commands:
  rollout         Manage the rollout of a resource
  scale           Set a new size for a deployment, replica set, or replication controller
  autoscale       Auto-scale a deployment, replica set, stateful set, or replication controller

Cluster Management Commands:
  certificate     Modify certificate resources.
  cluster-info    Display cluster information
  top             Display resource (CPU/memory) usage
  cordon          Mark node as unschedulable
  uncordon        Mark node as schedulable
  drain           Drain node in preparation for maintenance
  taint           Update the taints on one or more nodes

Troubleshooting and Debugging Commands:
```

The help message of kubectl

We can view the client version by running the command `kubectl version --client` in the terminal below:

Click to Connect...

The output will be as follows:

```
Client Version: version.Info{Major:"1", Minor:"24", GitVersion:"v1.24.0", GitCommit:"4ce5a8954017644c542
```

```
2  Kustomize Version: v4.5.4
```

The output of kubectl version --client

When using `kubectl` to access a Kubernetes cluster, we normally need a kubeconfig file, where the endpoint of the `kube-apiserver` and related credentials are stored. We can explicitly specify such a kubeconfig file with the flag `--kubeconfig`. If this flag is not specified, `kubectl` will search for a valid kubeconfig file with a precedence order. It will try to find out whether an environment `KUBECONFIG` is specified and then resolve the default file located at `~/.kube/config`. Below is the code snippet for the loading precedence of kubeconfig files for `kubectl`:

```
1  // staging/src/k8s.io/client-go/tools/clientcmd/loader.go
2
3  func (o *PathOptions) GetLoadingPrecedence() []string {
4      if o.IsExplicitFile() {
5          return []string{o.GetExplicitFile()}
6      }
7
8      if envVarFiles := o.GetEnvVarFiles(); len(envVarFiles) > 0 {
9          return envVarFiles
10     }
11     return []string{o.GlobalFile}
12 }
13
14 ...
15
16 func NewDefaultPathOptions() *PathOptions {
17     ret := &PathOptions{
18         GlobalFile:       RecommendedHomeFile,
19         EnvVar:           RecommendedConfigPathEnvVar,
20         ExplicitFileFlag: RecommendedConfigPathFlag,
21
22         GlobalFileSubpath: path.Join(RecommendedHomeDir, RecommendedFileName),
23
24         LoadingRules: NewDefaultClientConfigLoadingRules(),
25     }
26     ret.LoadingRules.DoNotResolvePaths = true
27
28     return ret
29 }
```

The loading precedence of the kubeconfig file for kubectl

If none of these files are found, the error below is returned.

```
1  The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

No valid kubeconfig file is specified

This error means that `kubectl` is accessing the insecure HTTP endpoint in the `localhost`.

Next

How kubectl Plugins Work