Kubeconfig

Summary

To access a Kubernetes cluster a **kubeconfig** file is required. A kubeconfig file has all the necessary cluster metadata and authentication details, that grants the user permission to query the cluster objects. Usually, the kubeconfig file is stored locally under the `~/.kube/config` file. However, k3s places the kubeconfig file within `/etc/rancher/k3s/k3s.yaml` path. Additionally, the location of a kubeconfig file can be set through the `--kubeconfig` kubectl flag or via the `KUBECONFIG` environmental variable.
A Kubeconfig file has 3 main distinct sections:

- **Cluster** - encapsulates the metadata for a cluster, such as the name of the cluster, API server endpoint, and certificate authority used to check the identity of the user.
- **User** - contains the user details that want access to the cluster, including the user name, and any authentication metadata, such as username, password, token or client, and key certificates.
- **Context** - links a user to a cluster. If the user credentials are valid and the cluster is up, access to resources is granted. Also, a `current-context` can be specified, which instructs which context (cluster and user) should be used to query the cluster.
Here is an example of a kubeconfig file:

```
apiVersion: v1
# define the cluster metadata
clusters:
- cluster:
    certificate-authority-data: {{ CA }}
    server: https://127.0.0.1:63668
  name: udacity-cluster
```

```
# define the user details
users:
# `udacity-user` user authenticates using client and key certificates
- name: udacity-user
  user:
    client-certificate-data: {{ CERT }}
    client-key-data: {{ KEY }}
# `green-user` user authenticates using a token
- name: green-user
  user:
    token: {{ TOKEN }}
# define the contexts
contexts:
- context:
    cluster: udacity-cluster
    user: udacity-user
  name: udacity-context
# set the current context
current-context: udacity-context
```

Once you start handling multiple clusters, you'll find a lot of useful information in this article

Kubeconfig Walkthrough

In this demo, the instructor uses a cluster bootstrapped with kind. Throughout this course, the students will use k3s to provision a cluser. However, in this demo *kina* is used to highlight how different tools provision the kubeconfig files.

If the students chose to follows this demo, these are the instructions to create a cluster using *kina*:

Note: kina can be installed directly on your local machine

- Ensure Docker is installed and running. Use the `docker --version` command to verify if Docker is installed.
- Install kind by using the official installation documentation
- Create a kind cluster using the `kind create cluster --name demo` command

Throughout the demo, the following kubectl commands are used:

```
# Inspect  the endpoints for the cluster and installed add-ons
kubectl cluster-info

# List all the nodes in the cluster.
# To get a more detailed view of the nodes, the `-o wide` flag can be passed
kubectl get nodes [-o wide]

# Describe a cluster node.
# Typical configuration: node IP, capacity (CPU and memory), a list of running pods on the node, podCIDR, etc.
kubectl describe node {{ NODE NAME }}
```

New terms

- **Kubeconfig** - a metadata file that grants a user access to a Kubernetes cluster

Further reading

- Organizing Cluster Access Using kubeconfig Files

NEXT