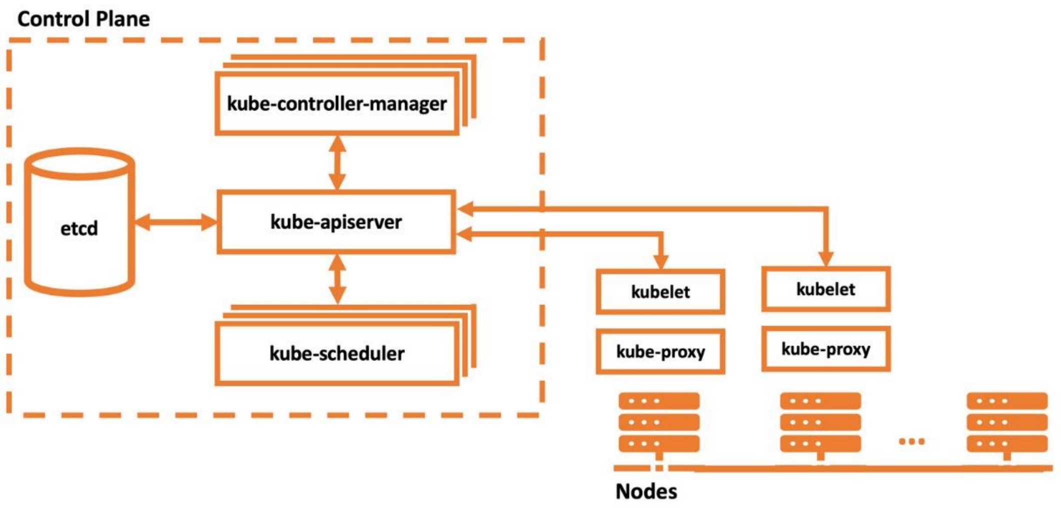
GETTING STARTED WITH KUBERNETES

Overview

In this lab, you will learn about Kubernetes, the most popular container management system in the market. Starting with the basics, architecture, and resources, you will create Kubernetes clusters and deploy real-life applications in them.

Architecture

There are two groups of Kubernetes components—namely, the control plane and the node. The control plane and node components, along with their interactions, are illustrated in the following diagram:



Procedure

Starting a Local Kubernetes Cluster

Kubernetes was initially designed to run on clusters with multiple servers. However, there are many times that you need to run a Kubernetes cluster locally, such as for development or testing. In this exercise, you will install a local Kubernetes provider and then create a Kubernetes cluster.

- Download the latest version of the minikube executable for your Linux operating system and set the binary as executable for your local system by running the following command in your terminal:

```
curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

These preceding commands download the binary for Linux and make it ready to use in the terminal.

- Start a Kubernetes cluster with the following command in your terminal:

```
minikube start
```

The single preceding command executes multiple steps to create a cluster successfully.
The output starts with printing out the version and the environment. Then, the images for Kubernetes components are pulled and started. Finally, you have a locally running Kubernetes cluster after a couple of minutes.

- Connect to the cluster node started by minikube with the following command:

  ```
  minikube ssh
  ```

  With the ssh command, you can continue working on the node running in the cluster.

- Check for each control plane component with the following commands:

  ```
  docker ps --filter "name=kube-apiserver" | grep -v "pause"
  docker ps --filter "name=etcd" | grep -v "pause"
  docker ps --filter "name=scheduler" | grep -v "pause"
  docker ps --filter "name=kube-controller-manager" | grep -v "pause"
  ```

These command checks for the Docker containers and filters with the control plane component names. The output does not contain the pause container, which is responsible for the networking setup of the container groups in Kubernetes. The output shows that four control plane components are running in Docker containers in the minikube node.

- Check for the first node component, kube-proxy, with the following command:

  ```
  docker ps --filter "name=kube-proxy" | grep -v "pause"
  ```

- Check for the second node component, kubelet, with the following command:

  ```
  pgrep -l kubelet
  ```

  This command lists the process with its ID running in minikube:

  ```
  2110 kubelet
  ```

  (You may get different ID output)

Since kubelet communicates between the container runtime and API server, it is configured to run directly on the machine instead of inside a Docker container.

- Disconnect from the minikube node with the following command:

  ```
  exit
  ```

Accessing Kubernetes Clusters with kubectl

To access the clusters securely and reliably, you need a reliable client tool, which is the official client tool of Kubernetes—namely, kubectl. In this exercise, you will install, configure, and use kubectl to explore its capabilities along with the Kubernetes API.

- Download the latest version of the kubectl executable for your Linux operating system and set this as the executable for your local system by running the following command in your terminal:

curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl

- Install kubectl

  sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

- Test to ensure the version you installed is up-to-date:

  kubectl version --client

These preceding commands download the binary for Linux and make it ready to use in the terminal.

- In your terminal, run the following command to configure kubectl to connect to the minikube cluster and use it for further access:

  kubectl config use-context minikube
  The use-context command configures the kubectl context to use the minikube cluster.
  Switched to context "minikube".

- Check for the cluster and client version with the following command:

  kubectl version --short
  This command returns the human-readable client and server version information.

- Check for further information about the cluster with the following command:

  kubectl cluster-info
  This command shows a summary of Kubernetes components, including the master and DNS.

- Get a list of the nodes in the cluster with the following command:

  kubectl get nodes
  Since the cluster is a minikube local cluster, there is only one node named minikube with the master role.

- List the supported resources in the Kubernetes API with the following command:

  kubectl api-resources --output="name"
  This command lists the name field of the api-resources supported in the Kubernetes API server. The long list shows how Kubernetes creates different abstractions to run containerized applications.

Now you have learned how to install, configure, and connect to a Kubernetes cluster. In addition, you have checked its version, the statuses of its nodes, and the available API resources. You have also learned to install and work with kubectl for accessing and managing applications running in Kubernetes.