Goal

This blog post describes the steps required to setup a multi node Kubernetes cluster for development purposes. This setup provides a production-like cluster that can be setup on your local machine.

multi node cluster setup requirement

Multi node Kubernetes clusters offer a production-like environment which has various advantages. Even though Minikube provides an excellent platform for getting started, it doesn’t provide the opportunity to work with multi node clusters which can help solve problems or bugs that are related to application design and architecture. For instance, Ops can reproduce an issue in a multi node cluster environment, Testers can deploy multiple versions of an application for executing test cases and verifying changes. These benefits enable teams to resolve issues faster which make the more agile.

Usage of Vagrant and Ansible

Vagrant is a tool that will allow us to create a virtual environment easily and it eliminates pitfalls that cause the works-on-my-machine phenomenon. It can be used with multiple providers such as Oracle VirtualBox, VMware, Docker, and so on. It allows us to create a disposable environment by making use of configuration files.

Ansible is an infrastructure automation engine that automates software configuration management. It is agentless and allows us to use SSH keys for connecting to remote machines. Ansible playbooks are written in yaml and offer inventory management in simple text files.

Prerequisites

* Vagrant should be installed on your machine. Installation binaries can be found [here](https://www.vagrantup.com/downloads.html).
* Oracle VirtualBox can be used as a Vagrant provider or make use of similar providers as described in Vagrant’s official [documentation](https://www.vagrantup.com/docs/providers/).
* Ansible should be installed in your machine. Refer to the [Ansible installation guide](https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html" \t "_blank) for platform specific installation.

Setup overview

We will be setting up a Kubernetes cluster that will consist of one master and two worker nodes. All the nodes will run Ubuntu Xenial 64-bit OS and Ansible playbooks will be used for provisioning.

Step 1: Creating a Vagrantfile

Use the text editor of your choice and create a file with named Vagrantfile, inserting the code below. The value of N denotes the number of nodes present in the cluster, it can be modified accordingly. In the below example, we are setting the value of N as 2.

IMAGE\_NAME = "bento/ubuntu-16.04"

N = 2

Vagrant.configure("2") **do** |config|

config.ssh.insert\_key = false

config.vm.provider "virtualbox" **do** |v|

v.memory = 1024

v.cpus = 2

**end**

config.vm.define "k8s-master" **do** |master|

master.vm.box = IMAGE\_NAME

master.vm.network "private\_network", ip: "192.168.50.10"

master.vm.hostname = "k8s-master"

master.vm.provision "ansible" **do** |ansible|

ansible.playbook = "kubernetes-setup/master-playbook.yml"

ansible.extra\_vars = {

node\_ip: "192.168.50.10",

}

**end**

**end**

(1..N).each **do** |i|

config.vm.define "node-**#{**i**}**" **do** |node|

node.vm.box = IMAGE\_NAME

node.vm.network "private\_network", ip: "192.168.50.**#{**i + 10**}**"

node.vm.hostname = "node-**#{**i**}**"

node.vm.provision "ansible" **do** |ansible|

ansible.playbook = "kubernetes-setup/node-playbook.yml"

ansible.extra\_vars = {

node\_ip: "192.168.50.**#{**i + 10**}**",

}

**end**

**end**

**end**

Step 2: Create an Ansible playbook for Kubernetes master.

Create a directory named kubernetes-setup in the same directory as the Vagrantfile. Create two files named master-playbook.yml and node-playbook.yml in the directory kubernetes-setup.

In the file master-playbook.yml, add the code below.

Step 2.1: Install Docker and its dependent components.

We will be installing the following packages, and then adding a user named “vagrant” to the “docker” group. - docker-ce - docker-ce-cli - containerd.io

---

- hosts: all

become: **true**

tasks:

- name: Install packages that allow apt to be used over HTTPS

apt:

name: "{{ packages }}"

state: present

update\_cache: yes

vars:

packages:

- apt-transport-https

- ca-certificates

- curl

- gnupg-agent

- software-properties-common

- name: Add an apt signing key for Docker

apt\_key:

url: https://download.docker.com/linux/ubuntu/gpg

state: present

- name: Add apt repository for stable version

apt\_repository:

repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable

state: present

- name: Install docker and its dependecies

apt:

name: "{{ packages }}"

state: present

update\_cache: yes

vars:

packages:

- docker-ce

- docker-ce-cli

- containerd.io

notify:

- docker status

- name: Add vagrant user to docker group

user:

name: vagrant

group: docker

Step 2.2: Kubelet will not start if the system has swap enabled, so we are disabling swap using the below code.

- name: Remove swapfile from /etc/fstab

mount:

name: "{{ item }}"

fstype: swap

state: absent

with\_items:

- swap

- none

- name: Disable swap

command: swapoff -a

when: ansible\_swaptotal\_mb *> 0*

Step 2.3: Installing kubelet, kubeadm and kubectl using the below code.

- name: Add an apt signing key for Kubernetes

apt\_key:

url: https://packages.cloud.google.com/apt/doc/apt-key.gpg

state: present

- name: Adding apt repository for Kubernetes

apt\_repository:

repo: deb https://apt.kubernetes.io/ kubernetes-xenial main

state: present

filename: kubernetes.list

- name: Install Kubernetes binaries

apt:

name: "{{ packages }}"

state: present

update\_cache: yes

vars:

packages:

- kubelet

- kubeadm

- kubectl

- name: Configure node ip

lineinfile:

path: /etc/default/kubelet

line: KUBELET\_EXTRA\_ARGS=--node-ip={{ node\_ip }}

- name: Restart kubelet

service:

name: kubelet

daemon\_reload: yes

state: restarted

Step 2.3: Initialize the Kubernetes cluster with kubeadm using the below code (applicable only on master node).

- name: Initialize the Kubernetes cluster using kubeadm

command: kubeadm init --apiserver-advertise-address="192.168.50.10" --apiserver-cert-extra-sans="192.168.50.10" --node-name k8s-master --pod-network-cidr=192.168.0.0/16

Step 2.4: Setup the kube config file for the vagrant user to access the Kubernetes cluster using the below code.

- name: Setup kubeconfig for vagrant user

command: "{{ item }}"

with\_items:

- mkdir -p /home/vagrant/.kube

- cp -i /etc/kubernetes/admin.conf /home/vagrant/.kube/config

- chown vagrant:vagrant /home/vagrant/.kube/config

Step 2.5: Setup the container networking provider and the network policy engine using the below code.

- name: Install calico pod network

become: **false**

command: kubectl create -f https://docs.projectcalico.org/v3.4/getting-started/kubernetes/installation/hosted/calico.yaml

Step 2.6: Generate kube join command for joining the node to the Kubernetes cluster and store the command in the file named join-command.

- name: Generate join command

command: kubeadm token create --print-join-command

register: join\_command

- name: Copy join command to local file

local\_action: copy content="{{ join\_command.stdout\_lines[0] }}" dest="./join-command"

Step 2.7: Setup a handler for checking Docker daemon using the below code.

handlers:

- name: docker status

service: name=docker state=started

Step 3: Create the Ansible playbook for Kubernetes node.

Create a file named node-playbook.yml in the directory kubernetes-setup.

Add the code below into node-playbook.yml

Step 3.1: Start adding the code from Steps 2.1 till 2.3.

Step 3.2: Join the nodes to the Kubernetes cluster using below code.

- name: Copy the join command to server location

copy: src=join-command dest=/tmp/join-command.sh mode=0777

- name: Join the node to cluster

command: sh /tmp/join-command.sh

Step 3.3: Add the code from step 2.7 to finish this playbook.

Step 4: Upon completing the Vagrantfile and playbooks follow the below steps.

$ cd /path/to/Vagrantfile

$ vagrant up

Upon completion of all the above steps, the Kubernetes cluster should be up and running. We can login to the master or worker nodes using Vagrant as follows:

$ *## Accessing master*

$ vagrant ssh k8s-master

vagrant@k8s-master:~$ kubectl get nodes

NAME STATUS ROLES AGE VERSION

k8s-master Ready master 18m v1.13.3

node-1 Ready <none> 12m v1.13.3

node-2 Ready <none> 6m22s v1.13.3

$ *## Accessing nodes*

$ vagrant ssh node-1

$ vagrant ssh node-2