Kubernetes

- Kubernetes : Minikube on CentOS 7
- Istio: Install
- Operator Framework

Kubernetes : Minikube on CentOS 7

Deploy Kubernetes on a developer's workstation.

This example is on CentOS 7with KVM. It is a VM. To follow along, you'll need metal or nested KVM.

It can be used to develop applications locally and then publish them to OpenShift, GKE or even Azure AKS.

Install the required packages and start libvirt

```
yum -y install qemu-kvm libvirt libvirt-daemon-kvm
systemctl enable --now libvirtd
```

Setup the repo for Kubernetes.

```
cat <<'EOF' > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Install Kubectl

```
yum -y install kubectl
```

Download the minikube binary and docker machine driver

```
wget https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 -O minikube
wget https://storage.googleapis.com/minikube/releases/latest/docker-machine-driver-kvm2
chmod 755 minikube docker-machine-driver-kvm2
```

```
mv minikube docker-machine-driver-kvm2 /usr/local/bin/
```

Run a quick minikube check to make sure it's working.

```
minikube version
```

`minikube version: v1.0.1`

Run a kubectl check

```
kubectl -o json
```

```
{
  "clientVersion": {
    "major": "1",
    "minor": "14",
    "gitVersion": "v1.14.1",
    "gitCommit": "b7394102d6ef778017f2ca4046abbaa23b88c290",
    "gitTreeState": "clean",
    "buildDate": "2019-04-08T17:11:31Z",
    "goVersion": "go1.12.1",
    "compiler": "gc",
    "platform": "linux/amd64"
  }
}
```

Start minikube

```
minikube start --vm-driver kvm2
```

After the the start command runs, check the status

```
minikube status
```

You should see out put like this.

```
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.39.33
```

Setup the environment

```
minikube docker-env
```

Output will look like this:

```
Kubernetes master is running at https://192.168.39.33:8443
KubeDNS is running at https://192.168.39.33:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
```

Log into the minikube

```
minikube ssh
```

Check the docker status

```
docker ps
```

At this point you are in a normal VM shell. All tools work as expected.

To stop minikube.

```
minikube stop
```

To remove minikube, do like follows

```
minikube delete
```

Istio: Install

Setup and Configure

Download Istio by running the following command:

```
curl -L https://istio.io/downloadIstio | sh -
```

Move to the Istio package directory. For example, if the package is istio-1.11.2:

```
cd istio-1.14.1
```

Add the istioctl client tool to the PATH for your workstation.

```
export PATH=$PWD/bin:$PATH
```

Validate if the cluster meets Istio install requirements by running the precheck:

```
istioctl x precheck
```

Output should look simalar to:

```
✔ No issues found when checking the cluster.
Istio is safe to install or upgrade!
```

Install and Adding Integration

- Istio

```
istioctl install
```

- Prometheus and Grafana

```
kubectl apply -f samples/addons/prometheus.yaml
kubectl apply -f samples/addons/grafana.yaml
```

- Jaeger

```
kubectl apply -f samples/addons/jaeger.yaml
```

- Kiali

```
kubectl apply -f samples/addons/kiali.yaml
```

Dashboards

```
istioctl dashboard <service>
```

Where `<service>` is one of:

- prometheus
- grafana
- jaeger
- kaili

Additional Setup

Gateway

Install the Gateway

```
kubectl create namespace istio-ingress
helm install istio-ingress istio/gateway -n istio-ingress
```

Operator Framework

Operator SDK

Download the release binary

Set platform information:

```
export ARCH=$(case $(uname -m) in x86_64) echo -n amd64 ;; aarch64) echo -n arm64 ;; *) echo -n $(uname -m) ;; esac)
export OS=$(uname | awk '{print tolower($0)}')
```

Download the binary for your platform:

```
export OPERATOR_SDK_DL_URL=https://github.com/operator-framework/operator-sdk/releases/download/v1.22.2
curl -LO ${OPERATOR_SDK_DL_URL}/operator-sdk_${OS}_${ARCH}
```

Verify the downloaded binary

Import the operator-sdk release GPG key from keyserver.ubuntu.com:

```
gpg --keyserver keyserver.ubuntu.com --recv-keys 052996E2A20B5C7E
```

Verify the signature:

```
curl -LO ${OPERATOR_SDK_DL_URL}/checksums.txt
curl -LO ${OPERATOR_SDK_DL_URL}/checksums.txt.asc
gpg -u "Operator SDK (release) <cncf-operator-sdk@cncf.io>" --verify checksums.txt.asc
```

You should see something similar:

```
gpg: assuming signed data in 'checksums.txt'
gpg: Signature made Fri 30 Oct 2020 12:15:15 PM PDT
gpg:                using RSA key ADE83605E945FA5A1BD8639C59E5B4762496218
gpg: Good signature from "Operator SDK (release) <cncf-operator-sdk@cncf.io>" [ultimate]
```

Make sure the checksums match:

```
grep operator-sdk_${OS}_${ARCH} checksums.txt | sha256sum -c -
```

You should see something similar to the following:

```
operator-sdk_linux_amd64: OK
```

Install the release binary in your PATH

```
chmod +x operator-sdk_${OS}_${ARCH} && sudo mv operator-sdk_${OS}_${ARCH} /usr/local/bin/operator-sdk
```

Bash completion

```
operator-sdk completion bash > /etc/bash_completion.d/operator-sdk
```

Install OLM

```
operator-sdk olm install
```

Operators

Prometheus

Clone the operator from Github

```
git clone https://github.com/prometheus-operator/kube-prometheus.git
cd kube-prometheus
```

Create the namespace and CRDs, and then wait for them to be availble before creating the remaining resources

```
kubectl create -f manifests/setup
```

Wait until the "servicemonitors" CRD is created. The message "No resources found" means success in this context.

```
until kubectl get servicemonitors --all-namespaces ; do date; sleep 1; echo ""; done
```

Deploy the rest of the operator manifests

```
kubectl create -f manifests/
```