[# Check your system:](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)

lscpu

[# Install time:](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/)

<https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/>

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

curl -LO "https://dl.k8s.io/release/**$(**curl -L -s https://dl.k8s.io/release/stable.txt**)**/bin/linux/amd64/kubectl.sha256"

# Validate the kubectl binary against the checksum file:

echo "**$(**cat kubectl.sha256**)** kubectl" | sha256sum --check

# 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

Or use this for detailed view of version:

kubectl version --client --output=yaml

Check that kubectl is properly configured by getting the cluster state:

kubectl cluster-info

If you see a message similar to the following, kubectl is not configured correctly or is not able to connect to a Kubernetes cluster.

The connection to the server <server-name:port> was refused - did you specify the right host or port?

For example, if you are intending to run a Kubernetes cluster on your laptop (locally), you will need a tool like Minikube to be installed first and then re-run the commands stated above.

If kubectl cluster-info returns the url response but you can't access your cluster, to check whether it is configured properly, use:

kubectl cluster-info dump

Minikube install:

sudo yum -y install qemu-kvm libvirt libvirt-daemon-kvm

2. Start KVM services and enable to make it persistent across reboot.

[root@kubebase ~]# systemctl start libvirtd

[root@kubebase ~]# systemctl enable libvirtd

virt-host-validate

sudo yum install fuse -y

sudo modprobe fuse

Ensure “firewalld” service is up and running.

systemctl status firewalld

### **Configure Kubernetes Repo:**

cd /etc/yum.repos.d

[root@kubebase yum.repos.d]# cat 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

[root@kubebase yum.repos.d]# pwd

/etc/yum.repos.d

Download the following components from Google repository.

* docker-machine-driver-kvm2
* minikube-linux-amd64 (minikube)

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

7. Modify the file permission and move the binary to the command search path.

[root@kubebase ~]# sudo chmod 755 minikube docker-machine-driver-kvm2

[root@kubebase ~]# sudo mv minikube docker-machine-driver-kvm2 /usr/local/bin/

8. Check the “minikube” version.

[root@kubebase ~]# minikube version

[root@kubebase ~]# kubectl version -o json

https://minikube.sigs.k8s.io/docs/reference/drivers/kvm2/

# Once installed:

kubectl get nodes

NAME STATUS ROLES AGE VERSION

minikube Ready control-plane 4d13h v1.28.3

kubectl cluster-info

Kubernetes control plane is running at https://192.168.49.2:8443

CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

### **Install the Operator**

Apply the Custome Resource Def for Operator:

kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/install/crds.yaml

Apply defaults for cockroach-operator-system namespace:

kubectl apply -f https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/install/crds.yaml

Set current namespace:

kubectl config set-context --current –namespace=cockroach-operator-system

Validation:

kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroach-operator-manager-6fd5c68d58-lvh7x 1/1 Running 0 21m

cockroachdb-0 1/1 Running 0 17m

cockroachdb-1 1/1 Running 0 17m

cockroachdb-2 1/1 Running 0 17m

cockroachdb-client-secure 1/1 Running 0 18m

Here you can see the 3 nodes running and the sql session we have also started up:

Download the example.yaml that will tell Operator how to configure the K8s cluster:

curl -O https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/examples/example.yaml

Now apply the yamlexample:

kubectl apply -f example.yaml

kubectl get pods

NAME READY STATUS RESTARTS AGE

cockroach-operator-manager-6fd5c68d58-lvh7x 1/1 Running 0 21m

cockroachdb-0 1/1 Running 0 17m

cockroachdb-1 1/1 Running 0 17m

cockroachdb-2 1/1 Running 0 17m

cockroachdb-client-secure 1/1 Running 0 18m

To get each pods IP:

kubectl get pod cockroachdb-0 -o custom-columns=NAME:metadata.name,IP:status.podIP

kubectl get pod cockroachdb-1 -o custom-columns=NAME:metadata.name,IP:status.podIP

kubectl get pod cockroachdb-2 -o custom-columns=NAME:metadata.name,IP:status.podIP

$ kubectl get pod cockroach-operator-manager-6fd5c68d58-lvh7x -o custom-columns=NAME:metadata.name,IP:status.podIP

NAME IP

cockroach-operator-manager-6fd5c68d58-lvh7x 10.244.0.4

$ kubectl get pod cockroachdb-0 -o custom-columns=NAME:metadata.name,IP:status.podIP

NAME IP

cockroachdb-0 10.244.0.8

khollman@KH-X20W:~$ kubectl get pod cockroachdb-1 -o custom-columns=NAME:metadata.name,IP:status.podIP

NAME IP

cockroachdb-1 10.244.0.10

khollman@KH-X20W:~$ kubectl get pod cockroachdb-2 -o custom-columns=NAME:metadata.name,IP:status.podIP

NAME IP

cockroachdb-2 10.244.0.9

From within K8s add another 3 nodes (change from 3 to 6 nodes)

https://www.cockroachlabs.com/docs/v23.1/scale-cockroachdb-kubernetes:

vi example.yaml

# nodes refers to the number of crdb pods that are created

# via the statefulset

nodes: 6

kubectl apply -f example.yaml

kubectl get pvc

kubectl get pods

Check Console also for added nodes & Uptime.

Console:

kubectl port-forward service/cockroachdb-public 8080

kubectl port-forward service/cockroachdb-0 8081

kubectl port-forward --address localhost,192.168.1.27 pod/[cockroachdb-0](https://localhost:8080/" \l "/node/1) 8888:26258

kubectl port-forward pod/[cockroachdb 26258](https://localhost:8080/" \l "/node/1):26258

Use the built-in SQL client

To use the CockroachDB SQL client, first launch a secure pod running the cockroach binary

kubectl create -f <https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/examples/client->[secure-operator.yaml](https://raw.githubusercontent.com/cockroachdb/cockroach-operator/v2.12.0/examples/client-secure-operator.yaml)

Shell into the pod:

kubectl exec -it cockroachdb-client-secure -- ./cockroach sql --certs-dir=/cockroach/cockroach-certs --host=cockroachdb-public

Now to add HAProxy config:

<https://www.cockroachlabs.com/docs/stable/deploy-cockroachdb-on-premises>

HA Proxy config

Use the secure pod:

kubectl exec -it cockroachdb-client-secure -- ./cockroach gen haproxy \

--certs-dir=/cockroach/cockroach-certs \

--host=cockroachdb-public

**LOCAL INSTALL @Vagrant:**

cd crdb

cockroach start --insecure --store=node1 --listen-addr=192.168.1.160:26257 \

--http-addr=192.168.1.160:8080 --join=192.168.1.160:26257,192.168.1.160:26258,192.168.1.160:26259 \

--background --pid-file=node1\_pid

cockroach start --insecure --store=node2 --listen-addr=192.168.1.160:26258 \

--http-addr=192.168.1.160:8081 --join=192.168.1.160:26257,192.168.1.160:26258,192.168.1.160:26259 \

--background --pid-file=node2\_pid

cockroach start --insecure --store=node3 --listen-addr=192.168.1.160:26259 \

--http-addr=192.168.1.160:8082 --join=192.168.1.160:26257,192.168.1.160:26258,192.168.1.160:26259 \

--background --pid-file=node3\_pid

cockroach init --host 192.168.1.160:26258 --insecure

grep 'node starting' node1/logs/cockroach.log -A 11

grep 'node starting' node2/logs/cockroach.log -A 11

grep 'node starting' node3/logs/cockroach.log -A 11

cockroach sql --insecure --host=192.168.1.160:26257

# Generate HAProxy config:

cockroach gen haproxy --insecure --host=192.168.1.160:26257

scp haproxy.cfg 192.168.1.161:/home/vagrant/

@ 192.168.1.161

sudo apt-get install haproxy

haproxy -f haproxy.cfg

(this process doesn’t come back)

TEST: From the original cluster node (192.168.1.160) run the following to conect to the HAProxy in 192.168.1.161:

cockroach sql --insecure --host=192.168.1.161:26257

**Benchmarking:**

There are some various options to load into cockroachdb..becareful as some can be QUITE pc-freezing...

https://www.cockroachlabs.com/docs/v23.1/performance-benchmarking-with-tpcc-large

cockroach sql --insecure --host=192.168.1.161:26257

Let’s try something mini,smaller than small, but larger than the LOCAL setting of 10 warehouses,eg. 250:

cockroach workload fixtures import tpcc --warehouses=250 'postgres://root@192.168.1.161:26257?sslmode=disable'

**SMALL:**

<https://www.cockroachlabs.com/docs/v23.1/performance-benchmarking-with-tpcc-small>

cockroach workload fixtures import tpcc --warehouses=2500 'postgres://root@192.168.1.161:26257?sslmode=disable'

LOCAL:

Import the dataset:

cockroach workload fixtures import tpcc \

--warehouses=10 \

'postgresql://root@192.168.1.161:26257?sslmode=disable’

Run tpcc:

cockroach workload run tpcc \

--warehouses=10 \

--ramp=3m \

--duration=10m \

'postgresql://root@192.168.1.161:26257?sslmode=disable'

cockroach node ls --insecure --host=192.168.1.161:26257

cockroach node status --insecure --host=192.168.1.161:26257

Add a 4th node:

cockroach start --insecure --store=node4 --listen-addr=192.168.1.160:26260 \

--http-addr=192.168.1.160:8083 –join=192.168.1.160:26257,192.168.1.160:26258,192.168.1.160:26259, 192.168.1.160:26260 \

--background –pid-file=node4\_pid

cockroach --insecure --host=192.168.1.161:26257 node decommission 4