High Available Kubernetes Cluster Setup using Kubespray

Kubespray is an *Ansible* based kubernetes provisioner. It helps us to setup a production grade, highly available and highly scalable Kubernetes cluster.

Prerequisites

Hardware Pre requisites

- 4 Nodes: Virtual/Physical Machines
- Memory: 2GB
- CPU: 1 Core
- Hard disk: 20GB available

Software Pre Requisites

`On All Nodes`

- Ubuntu 16.04 OS
- Python
- SSH Server
- Privileged user

`On Ansible Control Node`

- Ansible version 2.4 or greater

- Jinja

Networking Pre Requisites

- Internet access to download docker images and install softwares
- IPv4 Forwarding should be enabled
- Firewall should allow ssh access as well as ports required by Kubernetes. Internally open all the ports between node.

Architecture of a high available kubernetes cluster

Preparing the nodes

Run instructions in the section `On all nodes` in the cluster. This includes Ansible controller too.

Install Python

Ansible needs python to be installed on all the machines.

```
sudo apt update
sudo apt install python
```

Enable IPv4 Forwarding

```
On all nodes
```

Enalbe IPv4 forwarding by uncommenting the following line

```
echo "net.ipv4.ip_forward=1" >> /etc/sysctl.conf
```

Disable Swap

```
swapoff -a
```

Setup passwordless SSH between ansible controller and kubernetes nodes

`On control node`

Ansible uses passwordless ssh[1] to create the cluster. Let us see how to set it up from your *control node*.

Generate ssh keypair if not present already using the following command.

```
ssh-keygen -t rsa
```

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_r
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.
The key fingerprint is:
```

```
SHA256:yC4Tl6RYc+saTPcLKFdGlTLOWOIuDgO1my/NrMBnRxA ubuntu@n
The key's randomart image is:
+---[RSA 2048]----+
|    E    ..      |
|   . o +..       |
| . +o*+o         |
|. .o+Bo+         |
|. .++.X S        |
|+ +ooX .         |
|.=.OB.+ .        |
| .=o*= . .       |
|  .o.   .        |
+----[SHA256]-----+
```

Just leave the fields to defaults. This command will generate a public key and private key for you.

Copy over the public key to all nodes.

Example, assuming **ubuntu** as the user which has a privileged access on the node with ip address **10.10.1.101**,

```
ssh-copy-id ubuntu@10.10.1.101
```

This will copy our newly generated public key to the remote machine. After running this command you will be able to SSH into the machine directly without using a password. Replace *10.40.1.26* with your respective machine's IP.

e.g.

```
ssh ubuntu@10.10.1.101
```

Make sure to copy the public key to all kubernetes nodes. `Replace username with the actual user on your system`. If the above mentioned command fails, then copy your public key and paste it in the remote machine's `~/.ssh/authorized_keys` file.

e.g. (Only if ssh-copy-id fails)

```
cat ~/.ssh/id_rsa.pub
ssh ubunut@10.10.1.101
vim ~/.ssh/authorized_keys
# Paste the public key
```

Setup Ansible Control node and Kubespray

`On control node`

Set Locale

```
export LC_ALL="en_US.UTF-8"
export LC_CTYPE="en_US.UTF-8"
sudo dpkg-reconfigure locales
```

Do no select any other locale in the menu. Just press (**OK**) in the next two screens.

Setup kubespray

Kubespray is hosted on GitHub. Let us the clone the official repository.

```
git clone https://github.com/kubernetes-incubator/kubespray
cd kubespray
```

Install Prerequisites

Install the python dependencies. **This step installs Ansible as well. You do not need to install Ansible separately**.

```
sudo apt install python-pip -y
sudo pip install -r requirements.txt
```

Set Remote User for Ansible

Add the following section in ansible.cfg file

```
remote_user=ubuntu
```

If the user you are going to connect is diffent, use that instead.

Your *ansible.cfg* file should look like this.

```
[ssh_connection]
pipelining=True
ssh_args = -o ControlMaster=auto -o ControlPersist=30m -o C
#control_path = ~/.ssh/ansible-%%r@%%h:%%p
[defaults]
host_key_checking=False
gathering = smart
fact_caching = jsonfile
fact_caching_connection = /tmp
stdout_callback = skippy
library = ./library
callback_whitelist = profile_tasks
roles_path = roles:$VIRTUAL_ENV/usr/local/share/kubespray/r
deprecation_warnings=False
remote_user=ubuntu
```

Create Inventory

```
cp -rfp inventory/sample inventory/prod
```

where **prod** is the custom configuration name. Replace is with whatever name you would like to assign to the current cluster.

To build the inventory file, execute the inventory script along with the IP addresses of our cluster as arguments

```
CONFIG_FILE=inventory/prod/hosts.ini python3 contrib/invent
```

Where replace the IP addresses (e.g. 10.10.1.101) with the actual IPs of your nodes

Once its run, you should see an inventory file generated which may look similar to below

```
file: inventory/prod/hosts.ini


[all]
node1    ansible_host=10.10.1.101 ip=10.10.1.101
node2    ansible_host=10.10.1.102 ip=10.10.1.102
node3    ansible_host=10.10.1.103 ip=10.10.1.103
node4    ansible_host=10.10.1.104 ip=10.10.1.104

[kube-master]
node1
node2

[kube-node]
node1
node2
node3
node4

[etcd]
node1
node2
```

```
node3

[k8s-cluster:children]
kube-node
kube-master

[calico-rr]

[vault]
node1
node2
node3
```

Customise Kubernetes Cluster Configs

There are two configs files in your inventroy directory's group_vars (e.g. inventory/prod/group_vars) viz.

- all.yml
- k8s-cluster.yml

Ansible is data driven, and most of the configurations of the cluster can be tweaked by changing the variable values from the above files.

Few of the configurations you may want to modify

```
file: inventory/prod/group_vars/k8s-cluster.yml


kubelet_max_pods: 100
```

```
cluster_name: prod
helm_enabled: true
```

Provisioning kubernetes cluster with kubespray

`On control node`

We are set to provision the cluster. Run the following ansible-playbook command to provision our Kubernetes cluster.

```
ansible-playbook -b -v -i inventory/prod/hosts.ini cluster.
```

Option -i = Inventory file path
Option -b = Become as root user
Option -v = Give verbose output

If you face this following error, while running `ansible-playbook` command, you can fix it by running following instructions

`ERROR`:

```
ERROR! Unexpected Exception, this is probably a bug: (crypt
```

`FIX`:

```
sudo pip install --upgrade pip
sudo pip uninstall cryptography
sudo pip install cryptography
ansible-playbook -b -v -i inventory/prod/hosts.ini cluster.
```

This Ansible run will take around 30 mins to complete.

Kubectl Configs

```
On kube master node
```

Once the cluster setup is done, copy the configuration and setup the permissions.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Check the State of the Cluster

```
On the node where kubectl is setup
```

Let us check the state of the cluster by running,

```
kubectl cluster-info

Kubernetes master is running at https://10.10.1.101:6443
KubeDNS is running at https://10.10.1.101:6443/api/v1/names

To further debug and diagnose cluster problems, use 'kubect
```

```
kubectl get nodes
```

```
NAME      STATUS    ROLES          AGE       VERSION
node1     Ready     master,node    21h       v1.9.0+coreos.0
node2     Ready     master,node    21h       v1.9.0+coreos.0
node3     Ready     node           21h       v1.9.0+coreos.0
node4     Ready     node           21h       v1.9.0+coreos.0
```

If you are able to see this, your cluster has been set up successfully.

[1] You can use private key / password instead of passwordless ssh. But it requires additional knowledge in using Ansible.

Access Kubernetes Cluster Remotely (Optional)

```
On your local machine
```

You could also install kubectl on your laptop/workstation. To learn how to install it for your OS, refer to the procedure here.

e.g. To install **kubectl** on Ubuntu,

```
sudo apt-get update && sudo apt-get install -y apt-transpor
curl -s https://packages.cloud.google.com/apt/doc/apt-key.g
```

```
apt-key add -

sudo touch /etc/apt/sources.list.d/kubernetes.list

echo "deb http://apt.kubernetes.io/ kubernetes-xenial main"

sudo apt-get update

sudo apt-get install -y kubectl
```

Copy kubernetes config to your local machine

Copy `kubeconfig` file to your local machine

```
mkdir ~/.kube
scp -r ubuntu@MASTER_HOST_IP:/etc/kubernetes/admin.conf  ~/

kubectl get nodes
```

Deploy Kubernetes Objects

Since its a new cluster, which is differnt than what you have created with kubeadm earlier, or if this is the first time you are creating a kubernetes cluster with kubespray as part of **Advanced Workshop**, you need to deploy services which have been covered as part of the previous topics.

In order to do that, use the following commands on the node where you have configured kubectl

```
git clone https://github.com/schoolofdevops/k8s-code.git

cd k8s-code/projects/instavote

kubectl apply -f instavote-ns.yaml
kubectl apply -f prod/
```

Switch to **instavote** namespace and validate,

```
kubectl config set-context $(kubectl config current-context

kubectl get pods,deploy,svc
```

where,

- --cluster=prod : prod is the cluter name you created earlier. If not, use the correct name of the cluster (kubectl config view)
- --user=admin-prod: is the admin user created by default while installing with kubespray
- --namespace=instavote : the namespace you just created to deploy instavote app stack

[sample output]

```
$ kubectl get pods,deploy,svc

NAME                              READY     STATUS    RESTARTS
pod/db-66496667c9-qggzd           1/1       Running   0
pod/redis-6555998885-4k5cr        1/1       Running   0
pod/redis-6555998885-fb8rk        1/1       Running   0
pod/result-5c7569bcb7-4fptr       1/1       Running   0
pod/result-5c7569bcb7-s4rdx       1/1       Running   0
pod/vote-5d88d47fc8-gbzbq         1/1       Running   0
pod/vote-5d88d47fc8-q4vj6         1/1       Running   0
pod/worker-7c98c96fb4-7tzzw       1/1       Running   0

NAME                               DESIRED   CURRENT   UP-TO-DA
deployment.extensions/db           1         1         1
deployment.extensions/redis        2         2         2
deployment.extensions/result       2         2         2
deployment.extensions/vote         2         2         2
deployment.extensions/worker       1         1         1

NAME                TYPE        CLUSTER-IP       EXTERNAL-IP
service/db          ClusterIP   10.233.16.207    <none>
service/redis       ClusterIP   10.233.14.61     <none>
service/result      NodePort    10.233.22.10     <none>
service/vote        NodePort    10.233.19.111    <none>
```

References

- Installing Kubernetes On Premises/On Cloud with Kubespray
- Kubespray on Github