Equinix k8s Deployer

The goal of this how-to is to guide you to use this project to create an Equinix Machine (On-demand), after that, create 03 KVM virtual machines inside Equinix Machine and deploy a Kubernetes Cluster to do some demo or some tests.

Usage

1. On your local machine clone equinix-k8s-deployer project

```
git clone https://github.com/richardsonlima/equinix-k8s-deployer.git
```

2. Go to terraform-equinix-metalbox folder and use terraform project to create your base machine on equinix

```
cd terraform-equinix-metalbox
terraform init
terraform plan
terraform apply
```

> **ATTENTION:** Before performing the terraform commands on terraform-equinix-metalbox folder you will need to do some changes on terraform files: Firstly edit the variables.tf file in line 03 and put your equinix api token, after that, edit main.tf file in line 28 and put your equinix project_id de6bc1c3-7500-4a0a-9725-4ca6140e1f24.

```
# Terraform principal commands

This paragraph have a short explanation about terraform commands that we
will need for your project.

## Init
The terraform init command is used to initialize a working directory
containing Terraform configuration files. This is the first command that
should be run after writing a new Terraform configuration or cloning an
existing one from version control. It is safe to run this command multiple
times.

## Plan
The terraform plan command creates an execution plan. By default, creating
a plan consists of:

Reading the current state of any already-existing remote objects to make
sure that the Terraform state is up-to-date.
Comparing the current configuration to the prior state and noting any
differences.
```

```
Proposing a set of change actions that should, if applied, make the remote
objects match the configuration.

## Apply
The terraform apply command executes the actions proposed in a Terraform
plan.
```

3. After 3 minutes try to get the equinix machine public ip performing an API call against equinix platform:

```
curl -X GET --header 'Accept: application/json' --header 'X-Auth-Token:
${YOUR-API-TOKEN-HERE}'
'https://api.equinix.com/metal/v1/projects/de6bc1c3-7500-4a0a-9725-
4ca6140e1f24/devices' | jq |grep address
```

> ***NOTE:*** Take a look on Equinix API documentation if you need to undestand in details how it works. Please access this link https://metal.equinix.com/developers/api/ to get more details.

4. Access the new Equinix MetalBox machine from your local machine via ssh protocol:

```
ssh root@${EQUINIX-MACHINE-PUBLIC-IP}
```

> **ATTENTION:** For the security purpose is recommended that you create a normal user on the server and configure sudo for it, but as we know it's we will create a machine only for demo or test, then you don't need to matter about it during this setup.

5. Inside the new Equinix machine run the command below to install some packages needed for our setup:

```
sudo apt update && apt-get -y install pip python sshpass bridge-utils
qemu-kvm qemu virt-manager net-tools openssh-server mlocate libvirt-
clients libvirt-daemon libvirt-daemon-driver-storage-zfs python3-libvirt
virt-manager virtinst
```

6. On your Equinix machine clone our kubespray custom project:

```
cd /root/
git clone -b calico_bpf_enabled
https://github.com/richardsonlima/kubespray.git
```

6. Install python requirements for kubespray

```
cd /root/kubespray/
pip install -r requirements.txt
```

7. Create a public and private ssh key pair and create an encrypted password for the ansible user:

```
cd /root/kubespray/scripts/kvm-equinix-metalbox/k8s-provisioner-on-kvm/
ssh-keygen -t rsa -C ansible@host -f id_rsa
python ./encrypt-pw.py
```

8. Update the ubuntu.ks file inside on /root/kubespray/scripts/kvm-equinix-metalbox/k8s-provisioner-on-kvm/ directory, with the encrypted password and the generated public key.

> ***NOTE:***
> The Kickstart file already contains a public key for which the private key is provided and an encrypted password of which the plaintext is Welcome01. As indicated, these are for example purposes. Do not use them for production!

9. Create our 3 KVM virtual machines inside Equinix Server

```
cd /root/kubespray/scripts/kvm-equinix-metalbox/k8s-provisioner-on-kvm/
./call_create_vm.sh
```

> ***NOTE:***
> Waiting 5 minutes before run the script below to start our VMs

```
./start-vm.sh
```

10. Create our inventory configuration from kubespray sample files

```
cd /root/kubespray/
rm -Rf inventory/mycluster/
cp -rfp inventory/sample inventory/mycluster
```

```
declare -a IPS=($(for n in $(seq 1 3); do /root/kubespray/scripts/kvm-equinix-metalbox/k8s-provisioner-on-kvm/get-vm-ip.sh node$n; done))
```

```
echo ${IPS[@]}
```

```
CONFIG_FILE=inventory/mycluster/hosts.yml \
   python3 contrib/inventory_builder/inventory.py ${IPS[@]}
```

> ***NOTE:*** We're using the get-vm-ip.sh script to obtain the KVM virtual machines IP addresses. These should be used to generate a hosts.yml file indicating what should be installed where.

11. Let kubespray it copy an admin.conf to a host directory which after cluster created it should be used as ~/.kube/config:

```
echo '  vars:' >>  inventory/mycluster/hosts.yml
echo '    kubeconfig_localhost: true' >>  inventory/mycluster/hosts.yml
```

12. Execute Ansible to provision the kubernetes cluster on the kvm virtual machines using the previously generated ssh key for ansible user.

Firstly export the ANSIBLE_REMOTE_USER environment variable

```
export ANSIBLE_REMOTE_USER=ansible
```

```
ansible-playbook -i inventory/mycluster/hosts.yml --become --become-
user=root cluster.yml --private-key=/root/kubespray/scripts/kvm-equinix-
metalbox/k8s-provisioner-on-kvm/id_rsa
```

> ***ATTENTION:*** If you created a password for ansible user using encrypt-pw.py script please try the command below to create our kubernetes cluster. It will ask you to put the ansible password.

```
ansible-playbook -i inventory/mycluster/hosts.yml --become --become-
user=root cluster.yml --private-key=/root/kubespray/scripts/kvm-equinix-
metalbox/k8s-provisioner-on-kvm/id_rsa --ask-pass
```

13. Create the config file for kubectl on equinix machine:

```
mkdir -p ~/.kube/
cp -rip inventory/mycluster/artifacts/admin.conf ~/.kube/config
```

14. Install kubectl (for Kubernetes)

```
sudo snap install kubectl --classic
```

15. Allow the kube-system:clusterrole-aggregation-controller to access the dashboard:

```
kubectl create clusterrolebinding dashboard-admin -n default --
clusterrole=cluster-admin --serviceaccount=kube-system:clusterrole-
aggregation-controller
```

16. Get a token to access the dashboard:

```
kubectl -n kube-system describe secrets `kubectl -n kube-system get
secrets | awk '/clusterrole-aggregation-controller/ {print $1}'` | awk
'/token:/ {print $2}'
```

17. Deploy the kubernetes dashboard and do kubectl proxy to be able to access it at localhost:8001 or from your local machine:

```
kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.3.1/aio/deploy/r
ecommended.yaml

kubectl proxy --address='0.0.0.0' --port=8001 --accept-hosts='.*'
```

Access it from equinix machine using the URL below:

http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#/login

To access it from local machine using the URL below:

http://EQUINIX-MACHINE-PUBLIC-IP:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/#/login

18. Login and enjoy!

Try it on your new cluster: https://docs.projectcalico.org/security/tutorials/calico-policy

19. After tests performed please destroy the environment:

On your local machine do that

```
cd terraform-equinix-metalbox
terraform destroy
```

References

- Kubernetes documentation
- Kubespray
- KVM
- Terraform
- Ansible
- Equinix Metal