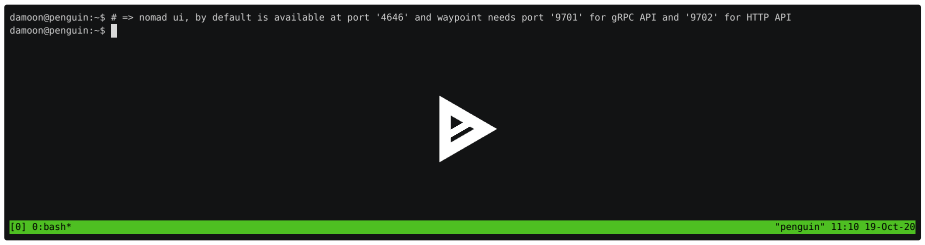
remote environment init

overview

in this section, we are going to provision a fresh debian google compute instance, provision it and install lxd and setup nomad cluster playground on it.

provisioning infrastructure



```
damoon@penguin:~$ # => nomad ui, by default is available at port '4646' and waypoint needs port '9701' for gRPC API and '9702' for HTTP API
damoon@penguin:~$
[0] 0:bash*                                                                              "penguin" 11:10 19-Oct-20
```

I am going to use google cloud sdk (`gcloud`) to create a compute engine. feel free to skip this step if you already has a remote server you can ssh into.

```
gcloud compute instances create "waypoint-demo" \
 --zone "us-central1-c" \
 --machine-type "n1-standard-16" \
 --subnet "default" \
 --network-tier "PREMIUM" \
 --image "debian-10-buster-v20200521" \
 --image-project "debian-cloud" \
 --boot-disk-size "40GB" \
 --boot-disk-type "pd-ssd" \
 --boot-disk-device-name "waypoint-demo"
```

now, we will create firewall rules to allow waypoint and nomad traffic. nomad ui, by default is available at port `4646` and waypoint needs port `9701` for gRPC API and `9702` for HTTP API so we need to setup a rule that allows ingress and egress traffic on these ports.

```
gcloud compute firewall-rules create waypoint-ingress \
  --allow tcp:4646,tcp:9701,tcp:9702 \
  --target-tags "waypoint-ingress" \
  --direction "INGRESS"
gcloud compute firewall-rules create waypoint-egress \
  --allow tcp:4646,tcp:9701,tcp:9702 \
  --destination-ranges '0.0.0.0/0' \
  --target-tags "waypoint-egress" \
  --direction "EGRESS"
```

lets add these tags to the compute instance we just created so the firewall allows port access

```
gcloud compute instances add-tags waypoint-demo --zone "us-central1-c" --tags=waypoint-egres
```

let's update ssh config file to make sure we can ssh into the remote we just created:

```
gcloud compute config-ssh
```

environment provisioning

```
Selecting previously unselected package libonig5:amd64.
Preparing to unpack .../084-libonig5_6.9.1-1_amd64.deb ...
Unpacking libonig5:amd64 (6.9.1-1) ...
Selecting previously unselected package libjq1:amd64.
Preparing to unpack .../085-libjq1_1.5+dfsg-2+b1_amd64.deb ...
Unpacking libjq1:amd64 (1.5+dfsg-2+b1) ...
Selecting previously unselected package jq.
Preparing to unpack .../086-jq_1.5+dfsg-2+b1_amd64.deb ...
Unpacking jq (1.5+dfsg-2+b1) ...
Selecting previously unselected package libalgorithm-diff-perl.
Preparing to unpack .../087-libalgorithm-diff-perl_1.19.03-2_all.deb ...
Unpacking libalgorithm-diff-perl (1.19.03-2) ...
Selecting previously unselected package libalgorithm-diff-xs-perl.
Preparing to unpack .../088-libalgorithm-diff-xs-perl_0.04-5+b1_amd64.deb ...
Unpacking libalgorithm-diff-xs-perl (0.04-5+b1) ...

[0] 0:ssh*                                                                "penguin" 11:26 19-Oct-20
```

let's install base dependencies

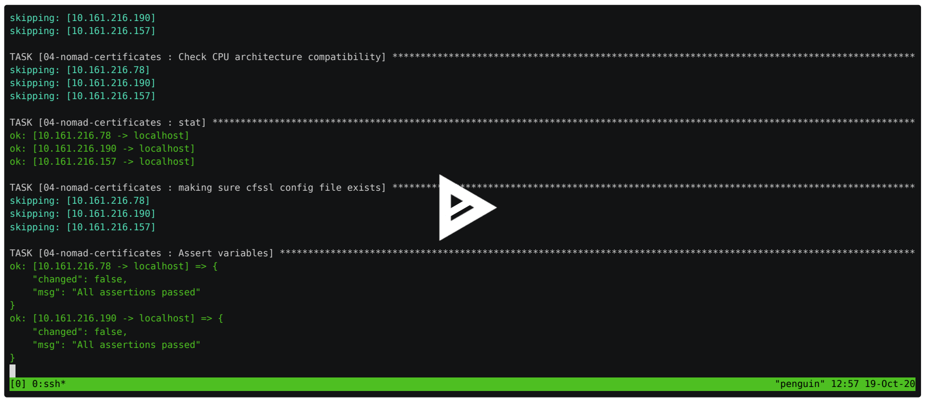
```
sudo apt update && sudo apt install -y neofetch make build-essential git snapd jq sshpass ar
```

now, we will clone this repo and use the `make init` target to initialize lxd. `make init` target uses `contrib/scripts/env-init` for bootstraping and installing needed tools. run `contrib/scripts/env-init --help` to learn more about how the command line interface works.

- Keep in mind that `make init` creates a priviledged lxc container. in case lxd is configured to block priviledge containers (like in chromeos), the target will fail.
- Make sure that you have already generated a ssh-key with `ssh-keygen` command before running `make init` target.
- in case lxd was not installed before running `make init` , the command would fail for the first time. you would have to login by running `newgrp lxd` and then running `make init` again.

```
git clone https://github.com/da-moon/waypoint-playground
pushd waypoint-playground
make init
popd
```

nomad playground environment setup



```
skipping: [10.161.216.190]
skipping: [10.161.216.157]

TASK [04-nomad-certificates : Check CPU architecture compatibility] *********************************************************************************************
skipping: [10.161.216.78]
skipping: [10.161.216.190]
skipping: [10.161.216.157]

TASK [04-nomad-certificates : stat] *****************************************************************************************************************************
ok: [10.161.216.78 -> localhost]
ok: [10.161.216.190 -> localhost]
ok: [10.161.216.157 -> localhost]

TASK [04-nomad-certificates : making sure cfssl config file exists] *********************************************************************************************
skipping: [10.161.216.78]
skipping: [10.161.216.190]
skipping: [10.161.216.157]

TASK [04-nomad-certificates : Assert variables] *****************************************************************************************************************
ok: [10.161.216.78 -> localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [10.161.216.190 -> localhost] => {
    "changed": false,
    "msg": "All assertions passed"
}

[0] 0:ssh*                                                                                                                              "penguin" 12:57 19-Oct-20
```

let’s clone `nomad-cluster-playbook`

```
git clone https://github.com/da-moon/nomad-cluster-playbook
```

before running any targets, we must setup an encryption password for `ansible-vault`. use the following snippet to generate a random password :

```
head -c16 < /dev/urandom| xxd -p -u | tee ~/.vault_pass.txt > /dev/null
```

now, lets setup nomad cluster containers

```
make -j`nproc` init
```

lets provision those containers with ansible

```
make pre-staging
```

- optional : port forwarding : since our compute engine allows incoming connections from public internet, to allow access to nomad api through the internet, we will use ip tables to forward host’s port `4646` to a single server container’s port `4646`. the following snippet will do the trick

```
sudo iptables -t nat -A PREROUTING -i $(ip link | awk -F: '$0 !~ "lo|vir|wl|lxd|docker|^[^0-
```

let’s confirm the rule has been added :

```
sudo iptables --table nat --list
```