Initial set-up to pull appropriate image from registry on each node :--

• Modify /etc/hosts file

10.132.209.150 noida-registry.scl.com (add this)

Save and exit

• Run the below command to get certificate:

# openssl s\_client -connect noida-registry.scl.com:443 -showcerts </dev/null 2>/dev/null | openssl x509 -

outform PEM | tee /etc/ssl/certs/noida-registry.scl.com.crt

• On redhat/centos/oracle linux

#update-ca-trust

On Ubuntu/debian

#update-ca-certificates

• #service docker restart

• Now you can login to Noida docker registry server

#docker login noida-registry.scl.com

Note: - For MacBook, import following cert in Keychain and restart the system: General -> Files -> noida-

registry.scl.com.crt.zip

Cluster setup:

1. Make host entry on each node to set host-name of the server :

Ex.:

10.132.214.226 - k8s-master

10.132.214.227 - k8s-worker02

10.132.214.224 - k8s-worker01

1. Reboot
2. On master node make a folder to generate public/private rsa key pair

#mkdir /root/.ssh --🡪 directory created

#ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

Enter passphrase (empty for no passphrase):

Enter same passphrase again:

Your identification has been saved in /root/.ssh/id\_rsa.

Your public key has been saved in /root/.ssh/id\_rsa.pub.

The key fingerprint is:

SHA256:ovNc0gXss5i5yln28PIHvWENlPSQtVnKcG+AiQx9LLY root@infra-virtual-machine

The key's randomart image is:

+---[RSA 2048]----+

| .+ ++\*+ . |

| .\* O\* B |

| .o= \* o |

| .E.. . |

| . S..o |

| . \*.++ . |

| o O +o o |

| . B.B o |

| +.+o+. |

+----[SHA256]-----+

1. Open this generated key #cat /root/.ssh/id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDQzE5S1/9figo306BKvH20DHofJoux8vanxL7dRbe74OrmUPPgtNXtzTKNKJbxwUzzHzT5qteCVonLO0VAuo3B8hAG+fqjLHzez5c5Zwhfe7H4r+u5PG4C66sS9uHpLcpB7gCPg1PpKhwvpn7x0oX4StJdB+DG6DRDtpEICkWVVsTj4xvHj3HEZHrgRKxCp0i2sHICTMnLh5+VEvS9leaBX1JZk8sdYVoKClo/cSGbGixZmoQkTV/OBCUH7qycCxAdcfrSfFvLxz6viVBUifVWss+tBYFTxLLPvuZ3SW59av6CP6oowHcv8dMoggszYQgAFRzH62AaMCxB6azsYKhT root@k8s-master

1. Copy this key and paste it on all other nodes by executing below command :

cat >> ~/.ssh/authorized\_keys <<EOF

<above-key>

EOF

**Note**:-- On others node create directory /root/.ssh ,if this is not present.Then perform ssh to

k8s-worker01 k8s-worker02. It should work fine without asking for any password.

**Enable Internet On All Nodes :---**

1. #sudo apt install resolvconf
2. Edit /etc/resolvconf/resolv.conf.d/head and add the following:

# vi /etc/resolvconf/resolv.conf.d/head.

nameserver 8.8.4.4

nameserver 8.8.8.8

1. Restart the resolvconf service

#sudo service resolvconf restart

1. Update with below commands:

#sudo apt-get update

#sudo apt-get install -y apt-transport-https curl

**Note**:- Here can find some failed message as Failed to fetch /kubernetes-xenial/InRelease. We will update all

these by below commands.

1. #curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -

OK

1. root@master:~# cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

EOF ENTER

deb https://apt.kubernetes.io/ kubernetes-xenial main

1. root@master:~# cat /etc/apt/sources.list.d/kubernetes.list

deb https://apt.kubernetes.io/ kubernetes-xenial main

1. root@master:~# sudo apt-get update
2. #sudo apt-get install -y kubelet kubeadm kubectl
3. #systemctl restart kubelet

GO TO MASTER NODE:-

1. root@master:~# swapoff -a ---> disable swap  
   root@master:~# vi /etc/fstab  
   root@master:~# cat /etc/fstab  
    🡺 /etc/fstab: static file system information.  
     
   # <file system> <mount point> <type> <options> <dump> <pass>  
   /dev/mapper/ubuntu--vg-root / ext4 errors=remount-ro 0 1  
   #/dev/mapper/ubuntu--vg-swap\_1 none swap sw 0 0 --> **comment any swap space**
2. root@master:~# **systemctl enable docker.service**

Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.

1. Now create a file called config.yaml
2. root@master:~# vi /var/lib/kubelet/config.yaml

**insert mode and paste below data** :---

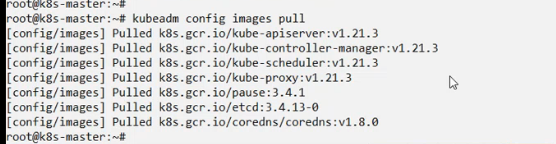
apiVersion: kubelet.config.k8s.io/v1beta1

kind: KubeletConfiguration

cgroupDriver: systemd

1. Now pull the K8s component images

#kubeadm config images pull



NOTE:- if you get error, please enter into DNS

root@master:~# cat /etc/hosts

127.0.0.1 localhost

127.0.1.1 master

# The following lines are desirable for IPv6 capable hosts

::1 ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

10.132.214.224 master

10.132.214.226 node01

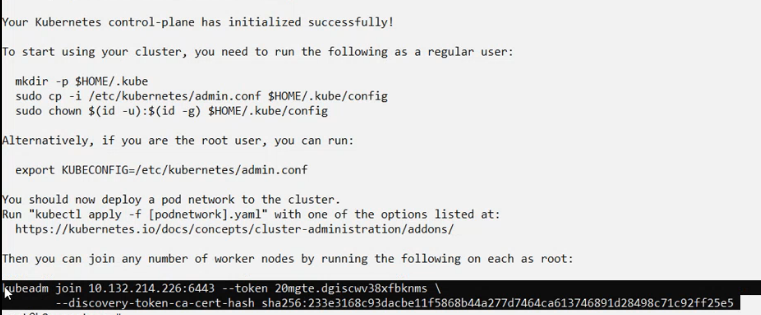
10.132.214.227 node02

74.125.133.82 k8s.gcr.io

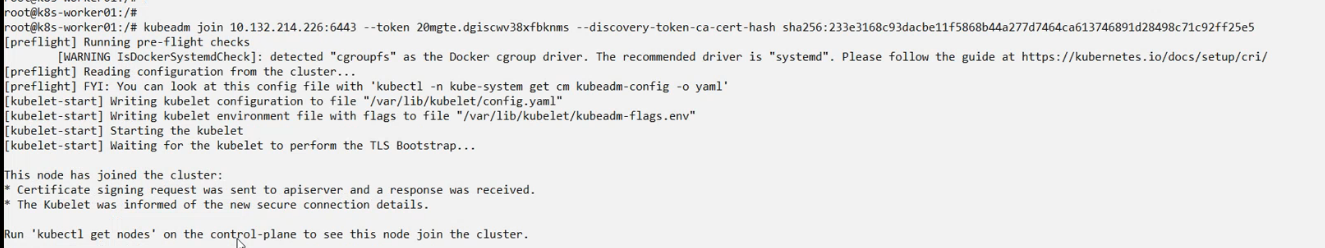
root@master#**kubeadm config images pull** --> this should be able to pull image

Create Cluster :--

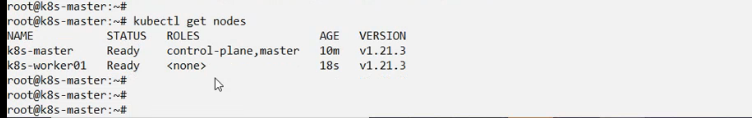
1. kubeadm init --apiserver-advertise-address=<ip-mster-server> --pod-network-cidr=10.244.0.0/16



1. Now follow the instruction as given in output on Worker nodes to join them.



1. Now you can verify on master as

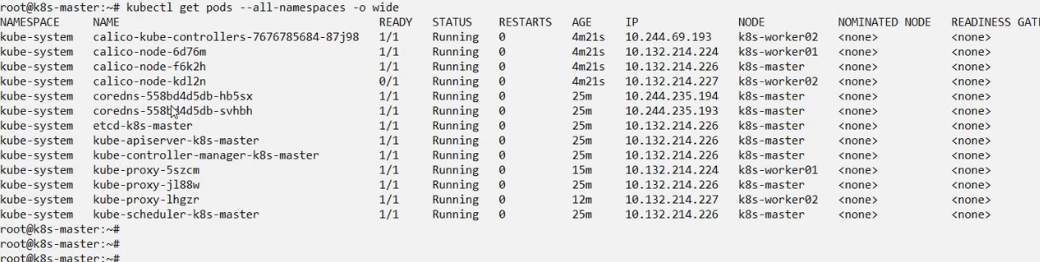


1. On Master server run below command to install CNI (container network interface)

root@master:~# kubectl apply -f <https://docs.projectcalico.org/v3.14/manifests/calico.yaml>  
configmap/calico-config created

1. Now check all pods on Master

#kubectl get po –all-namespaces



Note :- Faced this issue 🡪one calico pods was not running as error was “**calico/node is not ready: BIRD is not ready: BGP not established with 10.0. 0.1. In most cases, this “unready” status error in Kubernetes means that a particular peer is unreachable in the cluster. Check that BGP connectivity between the two peers is allowed in the environment.** “

**Resolution:**

root@k8s-master:~# kubectl set env daemonset/calico-node -n kube-system IP\_AUTODETECTION\_METHOD=interface=ens\\*

daemonset.apps/calico-node env updated

**SET UP COTS** :-

* MySQL
* MongoDB
* RabbitMQ
* Redis
* Gitea

NOTE:-- Follow the COTS Setup.docx guide to setup above services.

1. For GITEA follow below mentioned compose file.

version: "2"

networks:

gitea:

external: false

services:

server:

image: gitea/gitea:1.12.6

container\_name: gitea

environment:

- USER\_UID=1000

- USER\_GID=1000

- DB\_TYPE=mysql

- DB\_HOST=10.132.214.227

- DB\_NAME=gitea

- DB\_USER=idcAdmin

- DB\_PASSWD=Idemia@123

- SSH\_PORT=2222

restart: always

networks:

- gitea

volumes:

- ./gitea:/data

- /etc/timezone:/etc/timezone:ro

- /etc/localtime:/etc/localtime:ro

ports:

- "3000:3000"

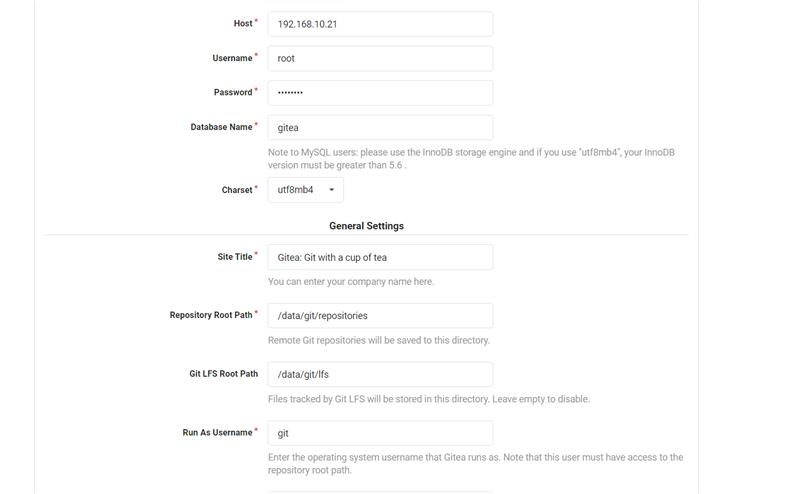
- "2222:2222"

Save and exit.

1. #docker-compose up –d
2. Create DB schema for the GITEA server on native mysql.

#mysql -uroot –ppassword -h<hostname>

1. mysql>create database gitea;
2. Exit;
3. From the browser access http://<gitea-container-host-ip>:3000/
4. Click on the Registration.

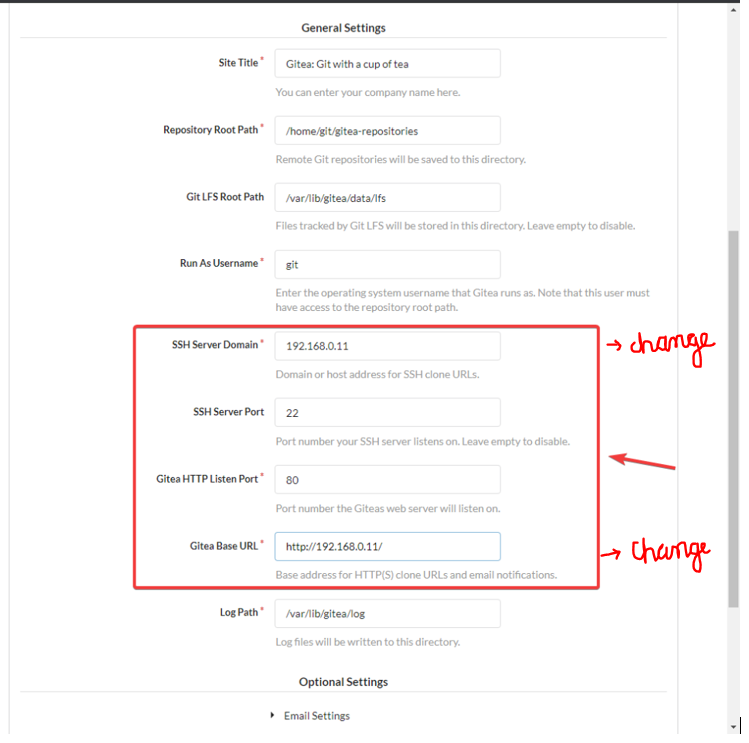


1. - SSH Server Domain = Host-IP-of-Gitea-Container-Host.

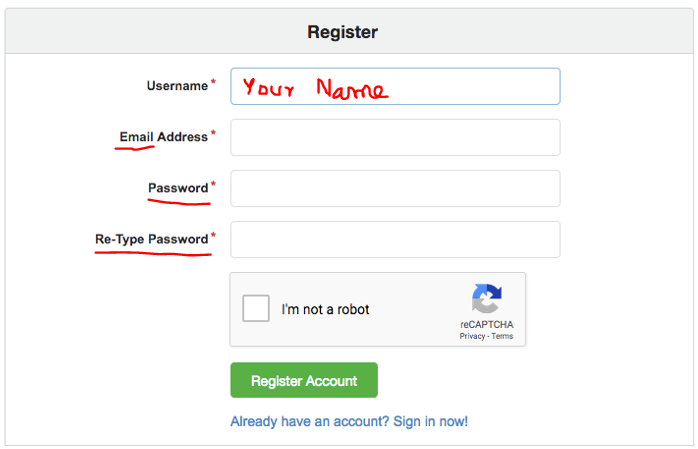
- SSH Server Port = 2222  as given in docker-compose.yaml file

- HTTP Listen Port = 3000 -> as given in docker-compose.yaml

- GIT Base URL = http://<Host-IP-of-Gitea-Container-Host>



1. Now proceed and go to next page of registration and fill the details.



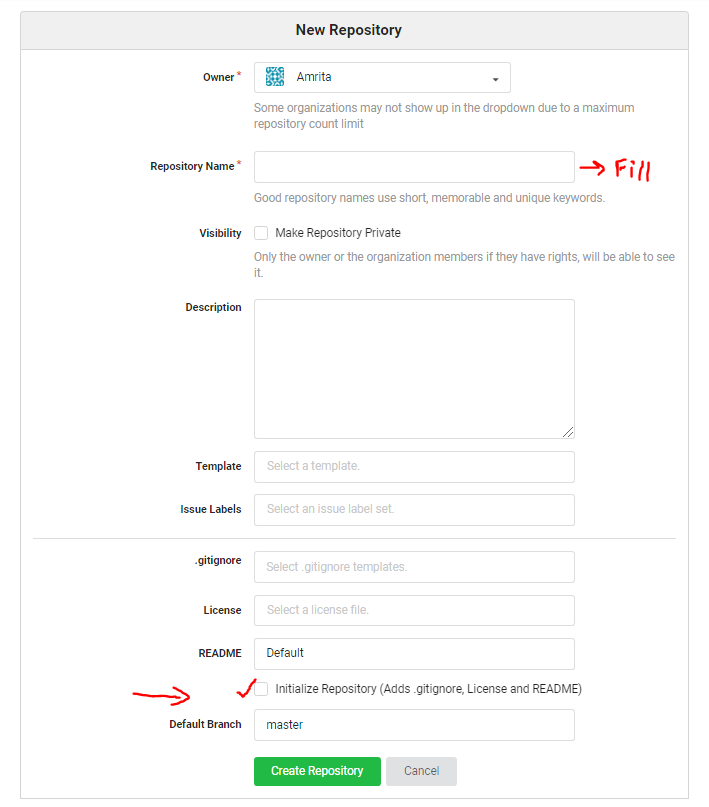
1. Create a repo where you want to upload your properties file –

Right Hand Side Top  Click on “+” symbol  New Repository 

Enter Repository Name

Select “Initiate Repository” checkbox

Create



1. Now on local machine create a folder and put all properties files there.
2. Go to browser (Gitea) and upload all **properties files** in the repository you have created before.

**Deployment of Micro-services on Master node :-**

* Create service and endpoints of COTS as well as pods for micro-services using below mentioned definition files.
* Link for definition files:-

<https://confluence.oberthur.com/display/PUNE/ID-MANAGE_6.7-deployment+on+K8s>

* Kubectl create -f example.yaml 🡪 to create pod, services, end-points
* Kubectl get po 🡪 to check pods
* Kubectl get svc 🡪 to check sevices
* Kubectl get ep 🡪 to check endpoints
* kubectl create secret generic ssl-keystore-cert --from-file=./<name\_of\_the\_file> 🡪 to create secret