## 

## 1.Installing Docker

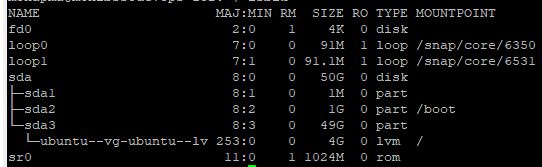
The Docker installation package available in the official Ubuntu 16.04 repository may not be the latest version. To get this latest version, install Docker from the official Docker repository. This section shows you how to do just that.

First remove the old docker repo file by executing below command.

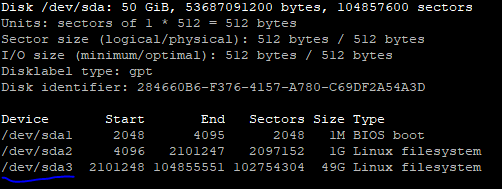
$ sudo apt-get purge docker lxc-docker docker-engine docker.io

$ sudo apt-get install curl apt-transport-https ca-certificates software-properties-common

1. execute lsblk first.



1. execute sudo fdisk -l



**Execute below command for resize the LVM partition**

sudo lvextend -l +8000 /dev/ubuntu-vg/ubuntu-lv

sudo resize2fs /dev/ubuntu-vg/ubuntu-lv

## 2. Setup Docker Repository

Now import dockers official GPG key to verify packages signature before installing them with apt-get. Run the below command on terminal.

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add

$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

## 3. Install Docker on Ubuntu

Your system is now ready for Docker installation. Run the following commands to upgrade apt index and then install Docker community edition on Ubuntu.

$ sudo apt-get update

$ sudo apt-get install docker-ce

After successful installation of Docker community edition, the service will start automatically, Use below command to verify service status and version.

$ sudo systemctl status docker

$ docker --version

Note:- Provide docker access to user from where you want to deploy/install

sudo usermod -aG docker mohapma

## Installing docker-compose (Optional)

For the sake of simplicity of deployment it's also good to have **docker-compose** handy which let's us define a full stack in one file definition.

$ curl -L "https://github.com/docker/compose/releases/download/1.8.1/docker-compose-$(uname -s)-$(uname -m)" > /usr/local/bin/docker-compose

$ chmod +x /usr/local/bin/docker-compose

$ docker-compose -v

## 5. Install Single Node Rancher-Kubernetes

Prepare a Linux host and install Docker onto it. Use any Linux flavor supported by Docker, and install a version of Docker supported by Kubernetes:

18.06.2

Execute the following command to download and run the Rancher server:

$ sudo docker run \--name rancher-server-2.3.2 \-d --restart=unless-stopped -p 80:80 -p 443:443 \-v /data:/var/lib/rancher \rancher/rancher:v2.3.2

Execute below command to check whether rancher is running or not:

$ sudo docker ps | grep rancher



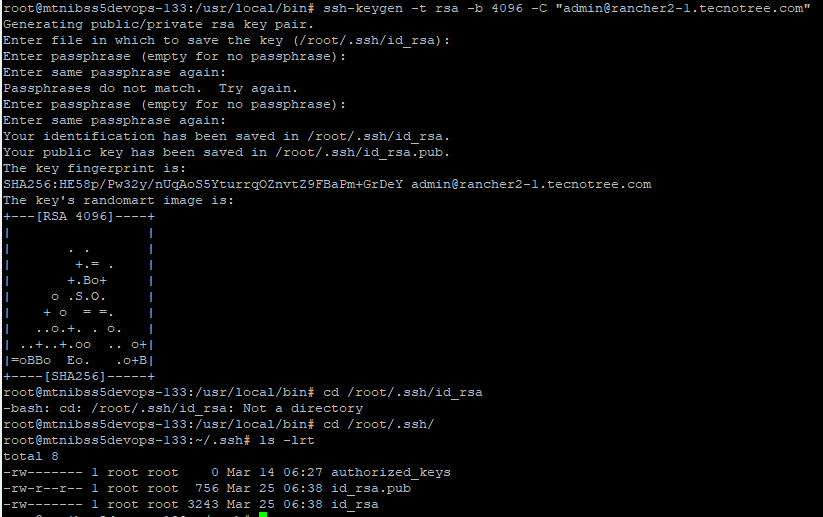
Access the rancher server URL : https://<server\_ip>

Set the password as “admin” for admin user.

**Generate ssh keys for the user running rke:**

$ ssh-keygen -t rsa -b 4096 -C [admin@rancher2-1.tecnotree.com](mailto:admin@rancher2-1.tecnotree.com)

Save the key to /root/.ssh/id\_rsa

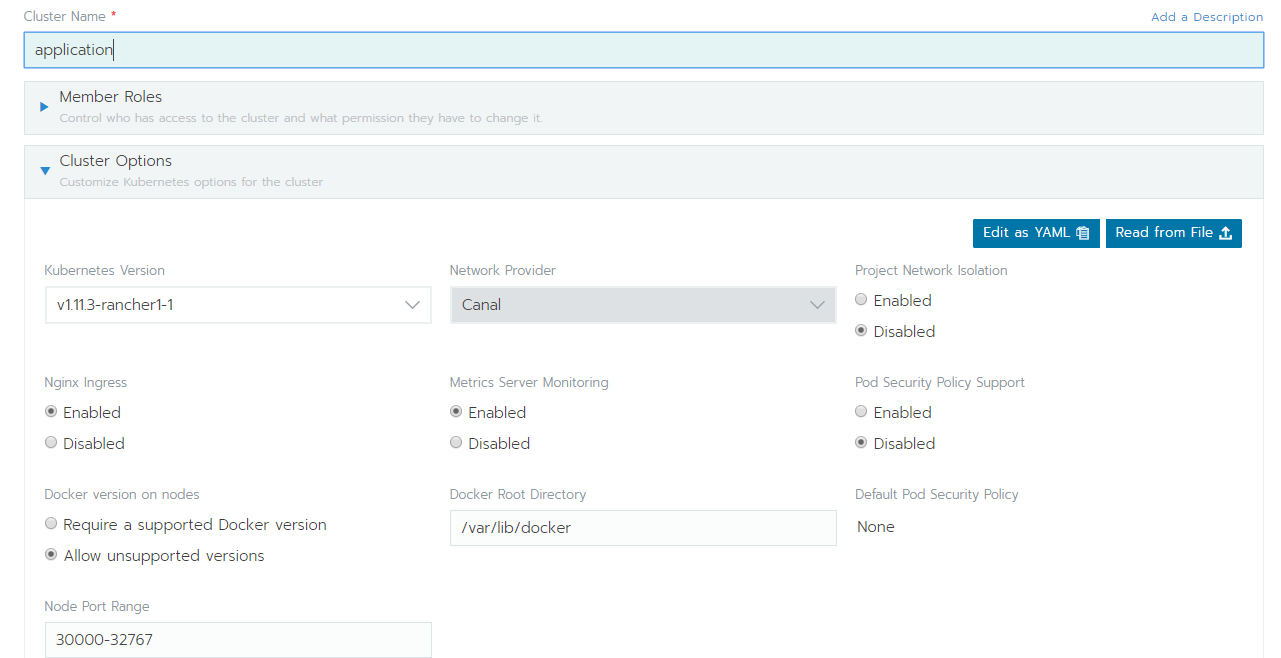


Copy the public ssh key to node server

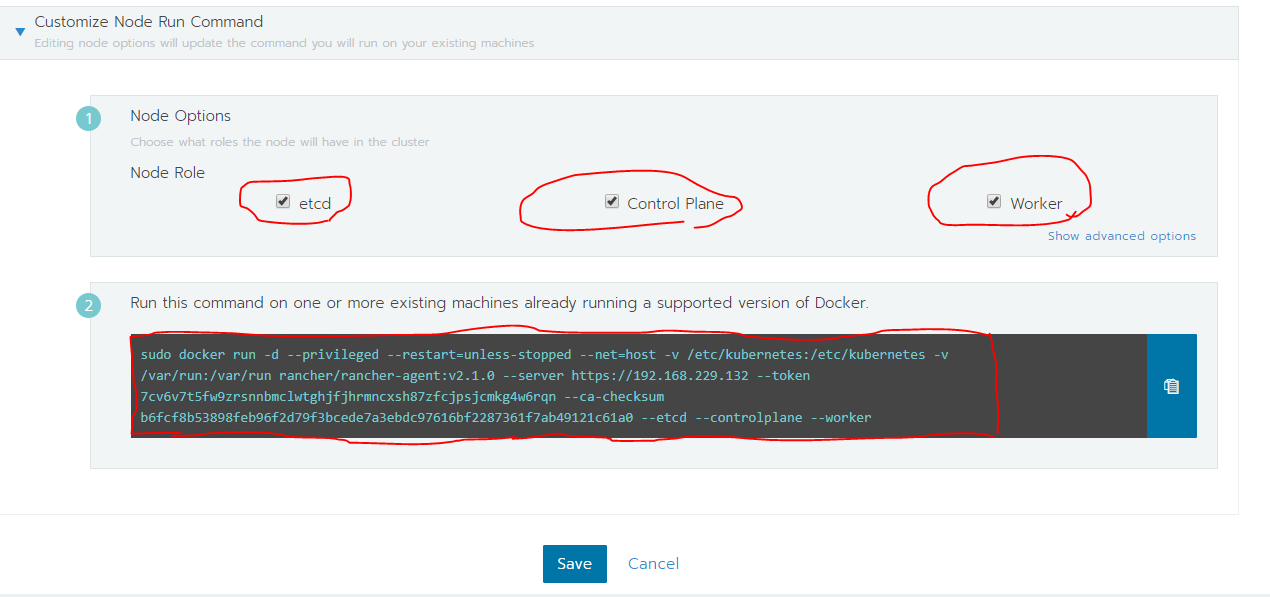
$ ssh-copy-id [mohapma@192.168.229.132](mailto:mohapma@192.168.229.132)

### **Then add cluster from Rancher UI**

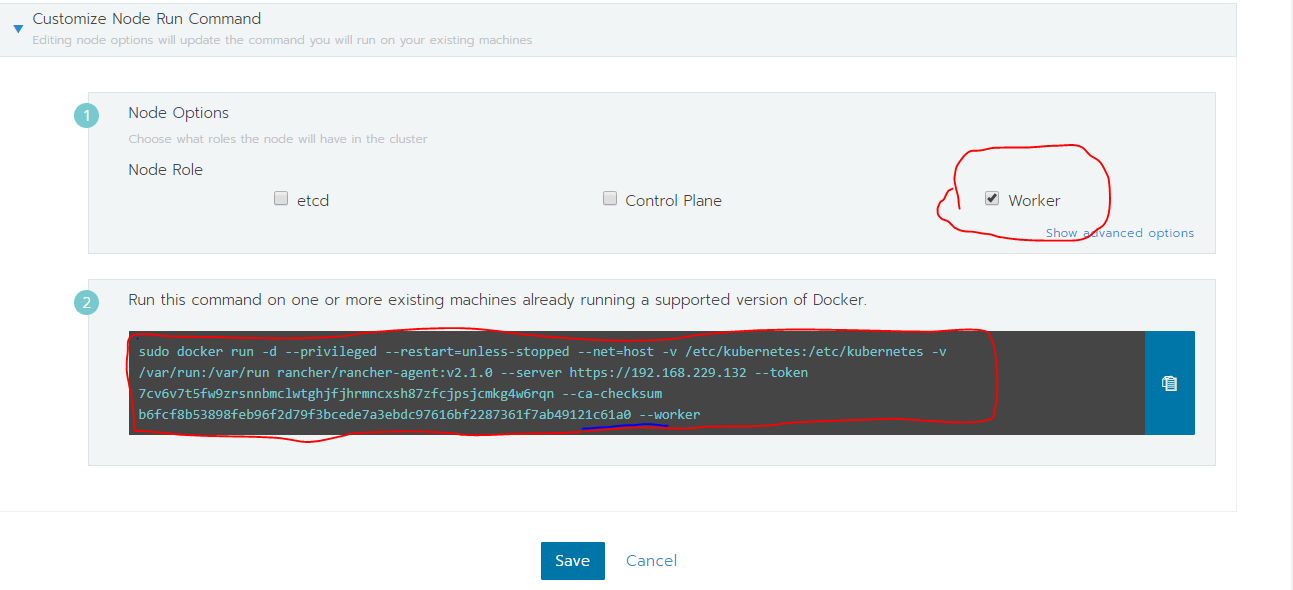
Add cluster -> custom



Then customise the node command. Copy the command & run it in master.



Then customise the node command. Copy the command & run it in worker.



## 6 Install HA Rancher-Kubernetes

### Install RKE

**Create directory for rke files:**

|  |
| --- |
| $ mkdir /root/k8s-cluster  $ sudo -i  $ cd /root/k8s-cluster |

**Install RKE:**

$ curl -LOk <https://github.com/rancher/rke/releases/download/v0.1.17/rke_linux-amd64>

$ chmod +x rke\_linux-amd64

$ mv rke\_linux-amd64 /usr/local/bin/rke

### Setup kubectl:

mkdir k8s-cluster

$ cd /root/k8s-cluster

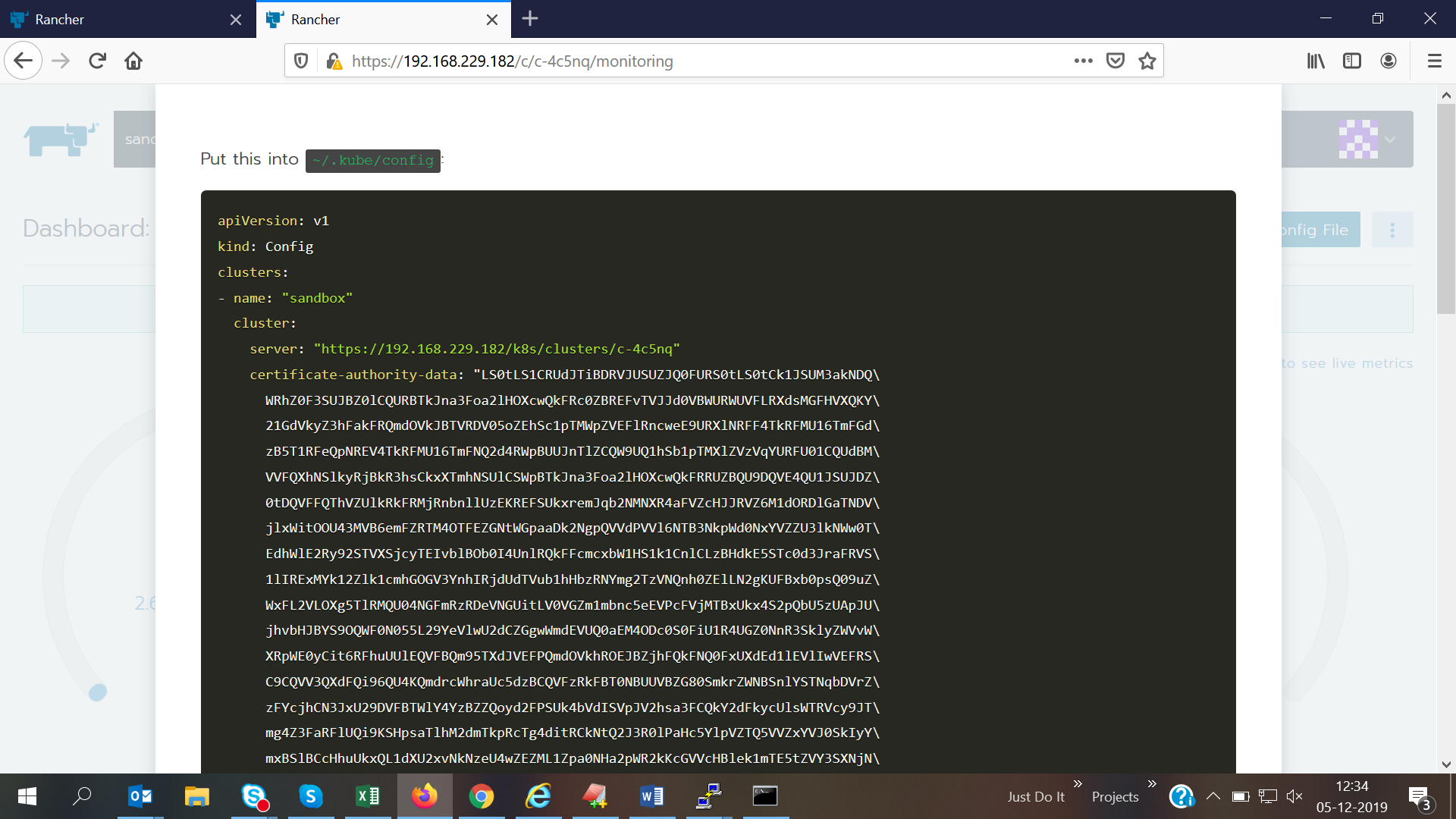
$ curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.12.0/bin/linux/amd64/kubectl

$ chmod +x kubectl

$ mv kubectl /usr/local/bin/

$ mkdir ~/.kube

Vi confing

* Copy the ./kube/config in this above file.
* 

### Install Helm:-

$ cd /root/k8s-cluster

$ curl -LOk <https://storage.googleapis.com/kubernetes-helm/helm-v2.13.0-linux-amd64.tar.gz>

$ tar xvf helm-v2.13.0-linux-amd64.tar.gz

$ mv linux-amd64/helm /usr/local/bin/

$ rm -rf linux-amd64

$ helm init --kubeconfig ~/.kube/config --service-account tiller

Install nodes

1. Install three nodes with minimal Photon OS.
2. Start and enable docker.
3. Edit /etc/ssh/sshd\_config and set AllowTcpForwarding yes.
4. Enable passwordless login from rke node by adding public key of the user running rke to each node's /root/.ssh/authorized\_keys.
5. Open required ports:

**/etc/systemd/scripts/iptables:**

|  |
| --- |
| #RKE ports:  iptables -A OUTPUT -p tcp --match multiport --dports 22,6443 -j ACCEPT    #Etcd node ports:  iptables -A INPUT -p tcp --match multiport --dports 2376,2379,2380,9099,10250 -j ACCEPT  iptables -A INPUT -p udp --match multiport --dports 8472 -j ACCEPT  iptables -A OUTPUT -p tcp --match multiport --dports 443,2379,6443,9099 -j ACCEPT  iptables -A OUTPUT -p udp --match multiport --dports 8472 -j ACCEPT    #Additional Controlplane node ports:  iptables -A INPUT -p tcp --match multiport --dports 80,443,6443,10254 -j ACCEPT  iptables -A OUTPUT -p tcp --match multiport --dports 2380,9099,10250,10254 -j ACCEPT |

**Verify cluster:**

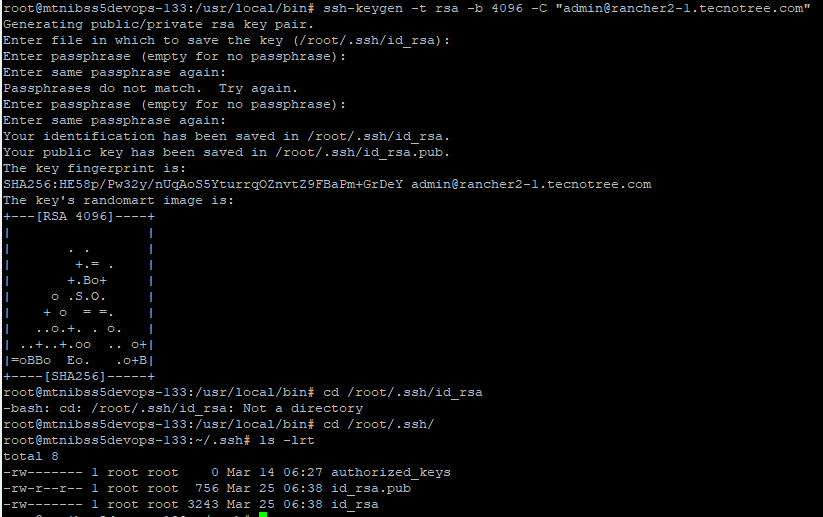
$ kubectl get nodes -o wide

$ kubectl get all --all-namespaces

**Generate ssh keys for the user running rke:**

$ ssh-keygen -t rsa -b 4096 -C [admin@rancher2-1.tecnotree.com](mailto:admin@rancher2-1.tecnotree.com)

Save the key to /root/.ssh/id\_rsa



Copy the public ssh key to node server

$ ssh-copy-id [mohapma@192.168.229.132](mailto:mohapma@192.168.229.132)

Or copy the id\_rsa.pub(from where the RKE is running) and paste the key in authorized\_keys of cluster servers.

$ vi rancher-cluster.yml --- add below contains to this file

nodes:

- address: 192.168.229.132

hostname\_override: k8-master

user: mohapma

role: [controlplane,etcd]

docker\_socket: /var/run/docker.sock

ssh\_key: ""

ssh\_key\_path: ""

- address: 192.168.229.133

hostname\_override: k8-worker

user: mohapma

role: [worker]

docker\_socket: /var/run/docker.sock

ssh\_key: ""

ssh\_key\_path: ""

services:

etcd:

snapshot: true

creation: 6h

retention: 24h

image: quay.io/coreos/etcd:latest

# extra\_args: {}

# kube-api:

# image: rancher/k8s:v1.8.3-rancher2

# extra\_args: {}

# service\_cluster\_ip\_range: 10.233.0.0/18

# kube-controller:

# image: rancher/k8s:v1.8.3-rancher2

# extra\_args: {}

# cluster\_cidr: 10.233.64.0/18

# service\_cluster\_ip\_range: 10.233.0.0/18

# scheduler:

# image: rancher/k8s:v1.8.3-rancher2

# extra\_args: {}

# kubelet:

# image: rancher/k8s:v1.8.3-rancher2

# extra\_args: {}

# cluster\_domain: cluster.local

# infra\_container\_image: gcr.io/google\_containers/pause-amd64:3.0

# cluster\_dns\_server: ""

# kubeproxy:

# image: rancher/k8s:v1.8.3-rancher2

# extra\_args: {}

network:

plugin: calico

options: {}

auth:

strategy: x509

options: {}

addons: ""

system\_images: {}

ssh\_key\_path: /home/mohapma/.ssh/id\_rsa

ignore\_docker\_version: true

$ rke up --config ./rancher-cluster.yml

$ cp -f kube\_config\_rancher-cluster.yml ~/.kube/config

Note:- On Google Kubernetes Engine you configure cluster autoscaler either on cluster creation or update or when creating a particular node pool (which you want to be autoscaled) by passing flags **--enable-autoscaling** **--min-nodes** and **--max-nodes** to the corresponding **gcloud** commands.

### **Creating the Cluster Configuration File using rke config**

$ rke config --name cluster.yml

$ rke config --empty --name cluster.yml

### **Verify cluster:**

$ kubectl get nodes -o wide

$ kubectl get all --all-namespaces

### **Create service account**:

$ kubectl -n kube-system create serviceaccount tiller

$ kubectl create clusterrolebinding tiller \

--clusterrole cluster-admin \

--serviceaccount=kube-system:tiller

## 6.2 Install Rancher

## $ helm repo add rancher-stable <https://releases.rancher.com/server-charts/stable>

$ helm install rancher-stable/rancher \

--kubeconfig ~/.kube/config \

--name rancher \

--namespace cattle-system \

--set hostname=rancher2.tecnotree.com \

--set ingress.tls.source=secret

### 6.3 **If you are getting below error during installation then follow the below steps**

Error: release rancher failed: namespaces "cattle-system" is forbidden: User "system:serviceaccount:kube-system:default"

cannot get resource "namespaces" in API group "" in the namespace "cattle-system"+++

++++++++++++++++++++++++++++++++++++++++++++++++++

$ helm reset --force

$ cat rbac-config.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: tiller

namespace: default

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: tiller

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: tiller

namespace: default

$ helm init --service-account tiller --upgrade -i registry.cn-hangzhou.aliyuncs.com/google\_containers/tiller:v2.13.0 --stable-repo-url https://kubernetes.oss-cn-hangzhou.aliyuncs.com/charts

$ helm version

$ kubectl get deployments --all-namespaces

$ kubectl --namespace kube-system create serviceaccount tiller

$ kubectl create clusterrolebinding tiller-cluster-rule \

--clusterrole=cluster-admin --serviceaccount=kube-system:tiller

$ kubectl --namespace kube-system patch deploy tiller-deploy \

-p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'

$ helm repo add rancher-stable https://releases.rancher.com/server-charts/stable

"rancher-stable" has been added to your repositories

$ helm search rancher

$ helm install rancher-stable/rancher \

--kubeconfig ~/.kube/config \

--name rancher \

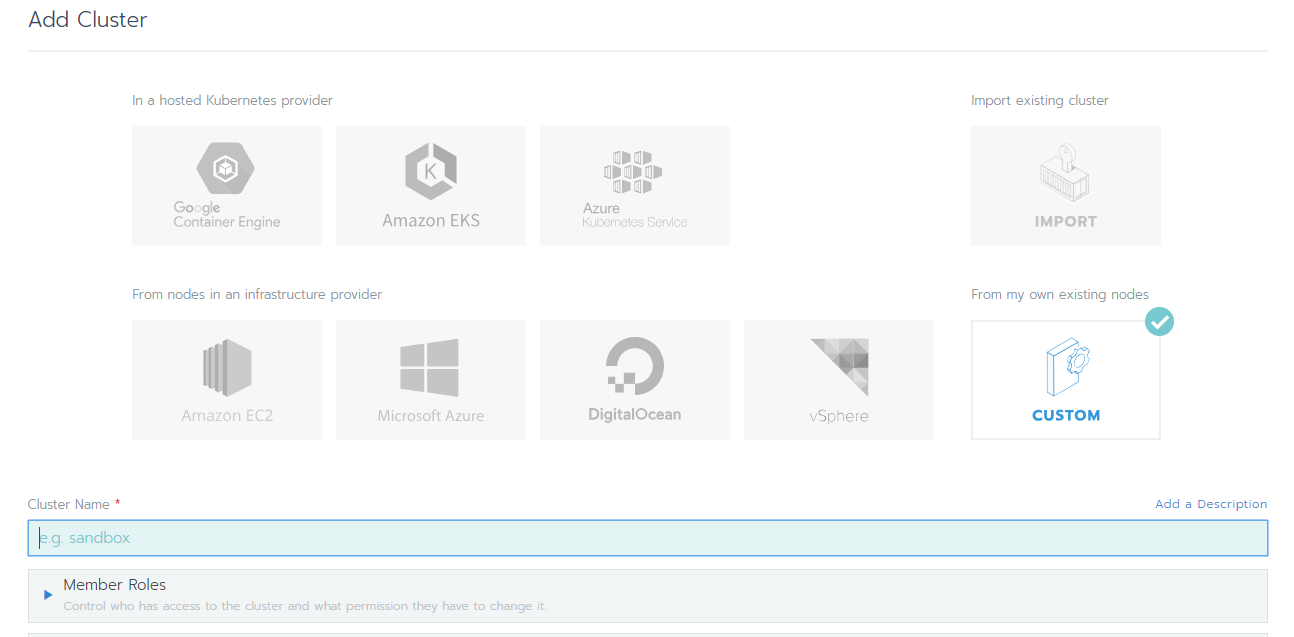
--namespace cattle-system \

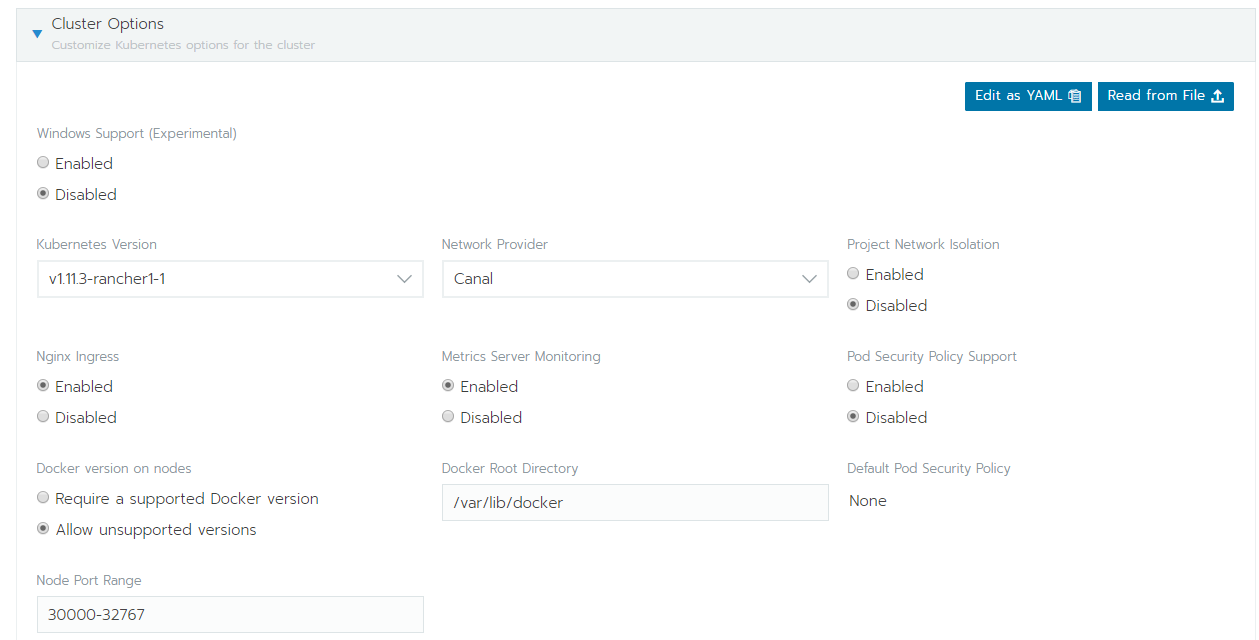
--set hostname=rancher2.test-devops.com \

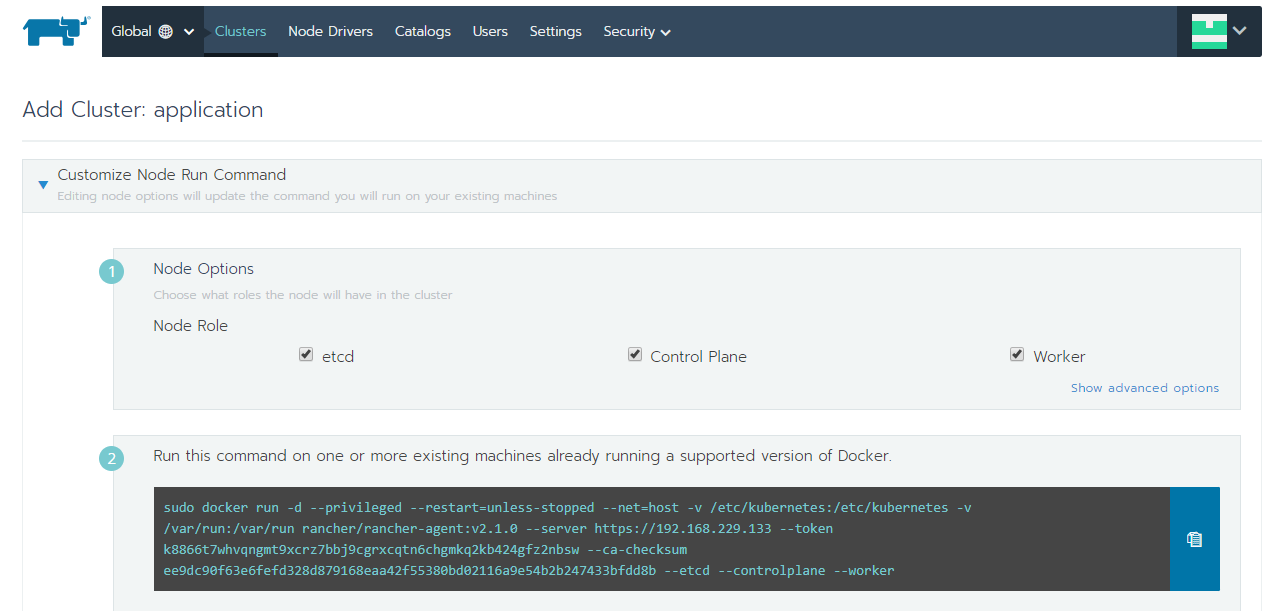
--set ingress.tls.source=secret

**Add Cluster:-**

Create cluster in Rancher UI as custom.







Copy the command from UI and execute that in cluster node.

sudo docker run -d --privileged --restart=unless-stopped --net=host -v /etc/kubernetes:/etc/kubernetes -v /var/run:/var/run rancher/rancher-agent:v2.1.0 --server https://192.168.229.133 --token k8866t7whvqngmt9xcrz7bbj9cgrxcqtn6chgmkq2kb424gfz2nbsw --ca-checksum ee9dc90f63e6fefd328d879168eaa42f55380bd02116a9e54b2b247433bfdd8b --etcd --controlplane --worker

## 7.Prometheus Monitoring On Kubernetes

$ git clone <https://github.com/bibinwilson/kubernetes-prometheus>

$ kubectl create namespace monitoring

### **Create A Prometheus Deployment**

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

  name: prometheus-deployment

  namespace: monitoring

spec:

  replicas: 1

  template:

    metadata:

      labels:

        app: prometheus-server

    spec:

      containers:

        - name: prometheus

          image: prom/prometheus:v2.1.0

          args:

            - "--config.file=/etc/prometheus/prometheus.yml"

            - "--storage.tsdb.path=/prometheus/"

          ports:

            - containerPort: 9090

          volumeMounts:

            - name: prometheus-config-volume

              mountPath: /etc/prometheus/

            - name: prometheus-storage-volume

              mountPath: /prometheus/

      volumes:

        - name: prometheus-config-volume

          configMap:

            defaultMode: 420

            name: prometheus-server-conf

        - name: prometheus-storage-volume

          emptyDir: {}

$ kubectl create  -f prometheus-deployment.yaml --namespace=monitoring

$ kubectl get deployments --namespace=monitoring

$ kubectl get pods --namespace=monitoring

$ kubectl port-forward prometheus-monitoring-3331088907-hm5n1 8080:9090 -n

#### **Exposing Prometheus As A Service**

Create a file named prometheus-service.yaml

apiVersion: v1

kind: Service

metadata:

  name: prometheus-service

spec:

  selector:

    app: prometheus-server

  type: NodePort

  ports:

    - port: 8080

      targetPort: 9090

      nodePort: 30000

$ kubectl create -f prometheus-service.yaml --namespace=monitoring