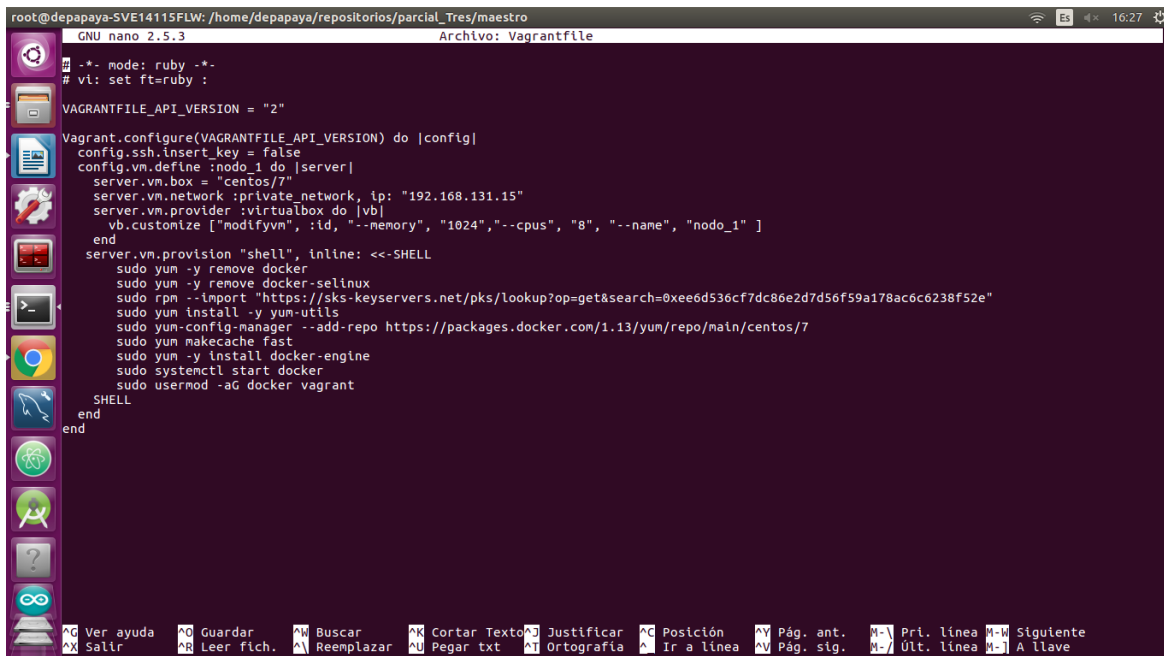


A00274596

Carlos Arredondo

Para realizar el aprovisionamiento de los equipos se decidió utilizar vagrant. A continuación se muestra el archivo Vagrantfile que contiene las configuraciones que se realizaron en todos las maquinas virtuales, uno nodo principal y los otros dos secundarios.

The image shows a terminal window with a dark background and light text. The title bar at the top indicates the file being edited is 'Vagrantfile'. The content of the file is a Vagrant configuration script. It starts with a comment indicating the mode is ruby. Then it sets the VAGRANTFILE_API_VERSION to "2". The main configuration block is enclosed in a 'Vagrant.configure' block. Inside, it sets 'config.ssh.insert_key = false'. It then defines a VM named 'nodo_1' using 'config.vm.define'. For this VM, it sets the box to 'centos/7', the network to 'private_network' with IP '192.168.131.15', and the provider to 'virtualbox'. It also customizes the VM with 'modifyvm' to set memory to '1024', cpus to '8', and name to 'nodo_1'. The provision section uses 'server.vm.provision' with a shell script. This script removes docker and docker-selinux, imports a GPG key from a Google Cloud package repository, installs yum-utils, adds the Docker repository, and then installs docker-engine and starts the docker service. Finally, it sets the user to 'vagrant'. The terminal window has a sidebar on the left with various icons and a bottom status bar with keyboard shortcuts.

Una vez se levantaron las tres maquinas se procedió a instalar **kubeadm**, **kubectl**, **kubelet** en cada una de ellas. Primero se actualizo el repositorio con:

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
EOF
```

Luego, se deshabilitó SELinux con:

```
setenforce 0
sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
```

Por ultimo se realiza la instalación y posterior activación de los servicios

```
yum install -y kubelet
yum install -y kubeadm
yum install -y kubectl
systemctl enable kubelet
systemctl start kubelet
```

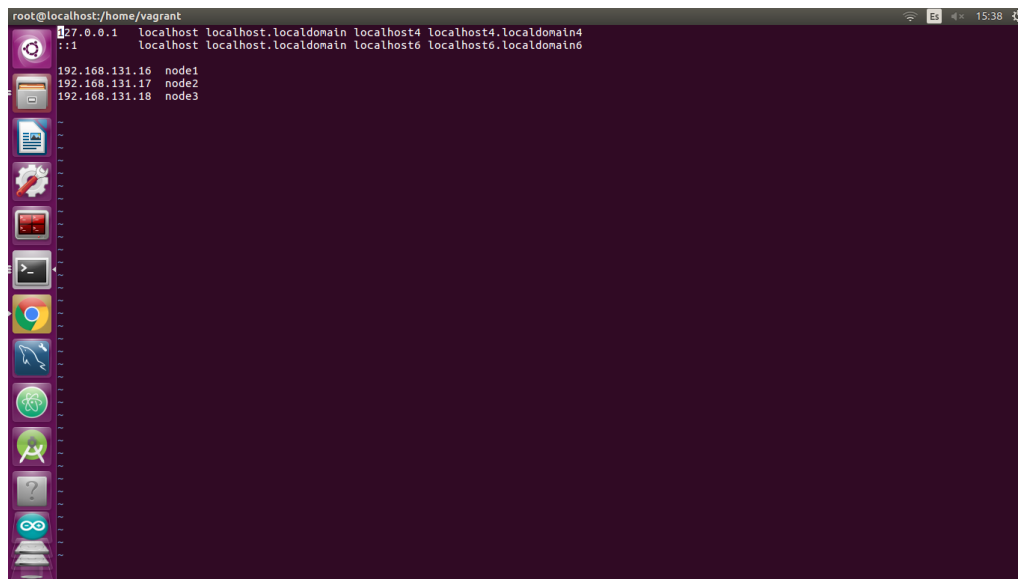
Este comando permite modificar la configuración del kubeadm:

```
sed -i "s/cgroup-driver=systemd/cgroup-driver=cgroupfs/g"  
/etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Este otro deshabilita la configuración del swap:

```
vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf  
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
```

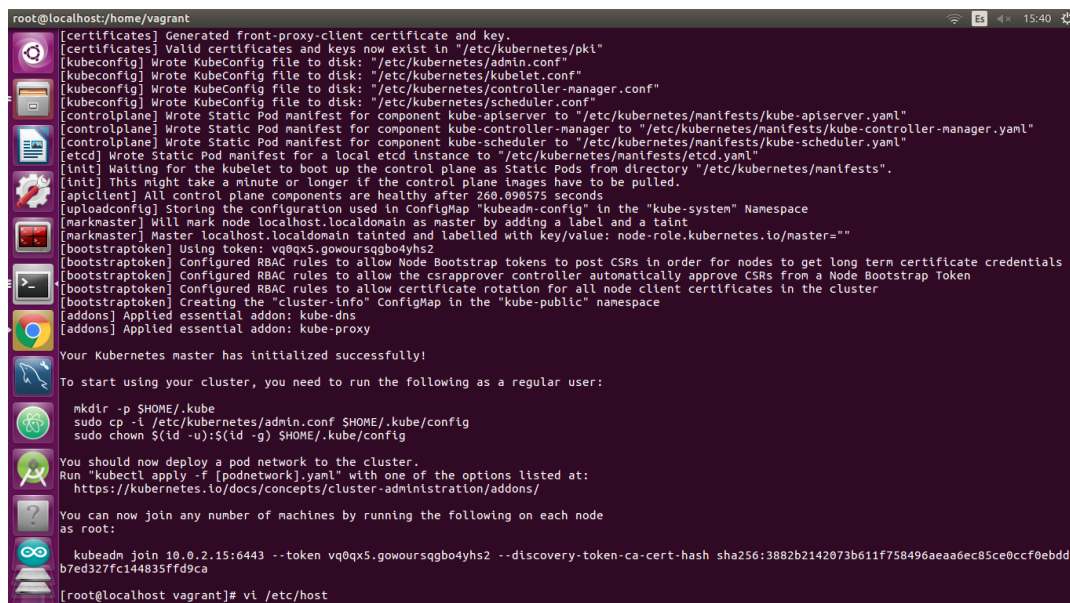
En cada nodo, tanto el principal como los secundarios se añade un archivo que permita la identificación de direcciones ip de cada uno.



```
root@localhost:/home/vagrant  
192.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4  
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6  
192.168.131.16 node1  
192.168.131.17 node2  
192.168.131.18 node3
```

Después en el nodo principal se ejecuta el siguiente comando que se puede apreciar en la captura de pantalla para que inicie el cluster y los nodos se puedan unir a él.

```
kubeadm init --apiserver-advertise-address $(hostname -i)
```



```
root@localhost:/home/vagrant  
[certificates] Generated front-proxy-client certificate and key.  
[certificates] Valid certificates and keys now exist in "/etc/kubernetes/pki"  
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/admin.conf"  
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/kubelet.conf"  
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/controller-manager.conf"  
[kubeconfig] Wrote KubeConfig file to disk: "/etc/kubernetes/scheduler.conf"  
[controlplane] Wrote Static Pod manifest for component kube-apiserver to "/etc/kubernetes/manifests/kube-apiserver.yaml"  
[controlplane] Wrote Static Pod manifest for component kube-controller-manager to "/etc/kubernetes/manifests/kube-controller-manager.yaml"  
[controlplane] Wrote Static Pod manifest for component kube-scheduler to "/etc/kubernetes/manifests/kube-scheduler.yaml"  
[etcd] Wrote Static Pod manifest for a local etcd instance to "/etc/kubernetes/manifests/etcd.yaml"  
[init] Waiting for the kubelet to boot up the control plane as Static Pods from directory "/etc/kubernetes/manifests".  
[init] This might take a minute or longer if the control plane images have to be pulled.  
[apiclient] All control plane components are healthy after 260.090575 seconds  
[uploadconfig] Storing the configuration used in ConfigMap "kubeadm-config" in the "kube-system" Namespace  
[markmaster] Will mark node localhost.localdomain as master by adding a label and a taint  
[markmaster] Master localhost.localdomain tainted and labelled with key/value: node-role.kubernetes.io/master=""  
[bootstraptoken] Using token: vq0qx5.gowoursqgb04yhs2  
[bootstraptoken] Configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order for nodes to get long term certificate credentials  
[bootstraptoken] Configured RBAC rules to allow the csrapprover controller automatically approve CSRs from a Node Bootstrap Token  
[bootstraptoken] Configured RBAC rules to allow certificate rotation for all node client certificates in the cluster  
[bootstraptoken] Creating the "cluster-info" ConfigMap in the "kube-public" namespace  
[addons] Applied essential addon: kube-dns  
[addons] Applied essential addon: kube-proxy  
  
Your Kubernetes master has initialized successfully!  
  
To start using your cluster, you need to run the following as a regular user:  
  
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config  
  
You should now deploy a pod network to the cluster.  
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:  
https://kubernetes.io/docs/concepts/cluster-administration/addons/  
  
You can now join any number of machines by running the following on each node  
as root:  
  
kubeadm join 192.0.2.15:6443 --token vq0qx5.gowoursqgb04yhs2 --discovery-token-ca-cert-hash sha256:3882b2142073b611f758496aea6ec85ce0ccf0ebdd  
b7ed327fc144835ffd9ca  
[root@localhost vagrant]# vi /etc/host
```

Una vez iniciado el cluster, los nodos se podrán unir a el a través de la siguiente instrucción

```
[root@localhost vagrant]# kubeadm join --token vq0qx5.gowoursqgbo4yhs2 --discovery-token-ca-cert-hash sha256:3882b2142073b611f758496aeaa6ec85c50ccf0ebddb7ed327fc144835ffd9ca
```

Problemas encontrados

La versión de virtual box se encontraba desactualizada, por lo tanto no era posible correr ninguna box de vagrant con el comando **vagrant up**. En un principio las direcciones ip para unirse al cluster no eran compatibles con la dirección del nodo maestro, pero se pudo reconocer que el comando **kubeadm init --apiserver-advertise-address \$(hostname -i)** genera una dirección por la cual se puede conectarse sin ningún inconveniente.