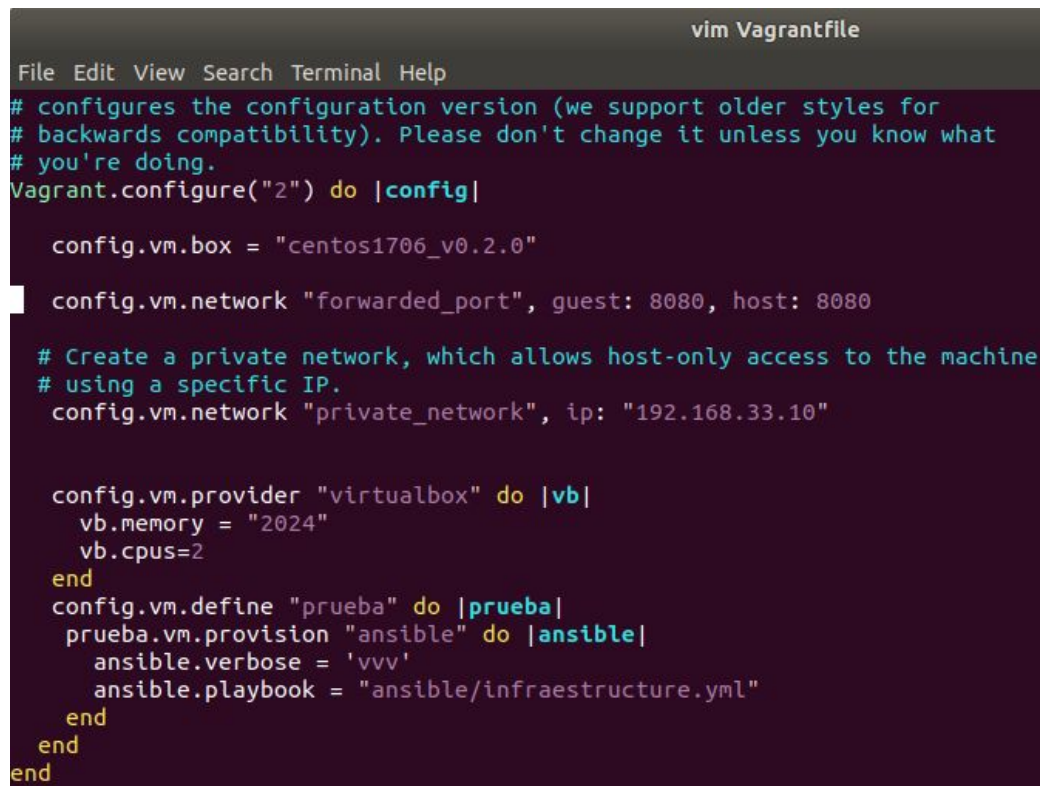


Universidad Icesi  
Sistemas distribuidos  
Angie Lorena Pérez  
A00242068

## Configuración del entorno

Se emplea Vagrant para gestionar las máquinas virtuales y ansible para automatizar.

### Vagrantfile

A screenshot of a terminal window showing a Vagrantfile configuration. The window title is 'vim Vagrantfile'. The editor shows a configuration for a virtual machine named 'prueba' using VirtualBox as the provider. It sets the VM box to 'centos1706\_v0.2.0', configures network forwarding on port 8080, and sets memory to 2024 and CPUs to 2. The VM is provisioned with Ansible, using a verbose output and a specific playbook.

```
vim Vagrantfile
File Edit View Search Terminal Help
# configures the configuration version (we support older styles for
# backwards compatibility). Please don't change it unless you know what
# you're doing.
Vagrant.configure("2") do |config|

  config.vm.box = "centos1706_v0.2.0"

  config.vm.network "forwarded_port", guest: 8080, host: 8080

  # Create a private network, which allows host-only access to the machine
  # using a specific IP.
  config.vm.network "private_network", ip: "192.168.33.10"

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "2024"
    vb.cpus=2
  end
  config.vm.define "prueba" do |prueba|
    prueba.vm.provision "ansible" do |ansible|
      ansible.verbose = 'vvv'
      ansible.playbook = "ansible/infraestructure.yml"
    end
  end
end
```

El directorio ansible contiene el playbook infraestructure.yml para instalar Kubernetes. También contiene el archivo config-default.sh donde se configuran las propiedades del cluster que se está diseñando, como el número de nodos, las ip, se define los minions y el maestro del cluster como se muestra a continuación:

```
# Define all your cluster nodes, MASTER node comes first"
# And separated with blank space like <user_1@ip_1> <user_2@ip_2> <user_3@ip_3>
export nodes=${nodes:-"root@192.168.90.20" "root@192.168.90.22" "root@192.168.90.24"}

# Define all your nodes role: a(master) or i(minion) or ai(both master and minion), must be the order same
role=${roles:-"a" "i" "i"}
# If it practically impossible to set an array as an environment variable
# from a script, so assume variable is a string then convert it to an array
export roles=( $role )

# Define minion numbers
export NUM_NODES=${NUM_NODES:-2}
# define the IP range used for service cluster IPs.
# according to rfc 1918 ref: https://tools.ietf.org/html/rfc1918 choose a private ip range here.
export SERVICE_CLUSTER_IP_RANGE=${SERVICE_CLUSTER_IP_RANGE:-192.168.3.0/24} # formerly PORTAL_NET
# define the IP range used for flannel overlay network, should not conflict with above SERVICE_CLUSTER_IP_RANGE
```

Se definen tres nodos con su respectiva ip, de los cuales uno es “a” = master y dos “i”=minions. Una vez se hace la configuración se procede a iniciar la máquina con el comando **vagrant up**.

Es necesario instalar los siguientes paquetes:

**kubeadm**: comando para arrancar el cluster, se ejecuta en el master.

**kubelet**: este componente se ejecuta en todas las máquinas del clúster y por medio de este se inician pods y contenedores.

**kubectl**: la línea de comando útil para hablar comunicarse con el cluster.

## Creación de pods

### Pod

Un pod es la unidad mínima que es manejada por kubernetes. Este pod es un grupo de uno o más contenedores (normalmente de Docker), con almacenamiento compartido entre ellos y las opciones específicas de cada uno para ejecutarlos.

Los pods pueden utilizarse para realizar escalado horizontal, aunque fomentan el trabajo con microservicios puestos en contenedores diferentes para crear un sistema distribuido mucho más robusto.

Con Kubernetes ya instalado se procede a crear los pods, por medio del comando

```
kubectl run maquina1 --image=nginx --port=80
```

Al ejecutarse este comando también se va a mostrar el **replication controller** que se va a encargar de gestionar el pod, asegura que el pod esté siempre disponible, de modo que si hay muchos pods eliminará algunos y si hay pocos crear nuevos.

A través del replication controller también se pueden crear las réplicas que se deseen del grupo de pods que se tengan con el comando:

```
kubectl scale rc nombre-del-pod --replicas=2
```

Para eliminarlos:

```
kubectl delete rc nombre
```

Ahora una forma para que un pod se pueda comunicar con otro pod, es por medio de un servicio que lo haga accesible.