



## Instalación Cluster OCP UPI vSphere - VMWare

---



29/02/2024

Version: 1.0

Fecha: 29-02-2024

Actores y roles en el requerimiento

Realizador	Empresa	Contacto
Jaime Galdames	INTAC	jgaldames@intaclatam.com
José Pablo Arancibia	INTAC	jparancibia@intaclatam.com

Historial de Cambios

Día de revisión	Revisado por	Razón de cambio
29/02/2024	INTAC	Primera versión del documento

Historial de cambios documentos relacionados

Documentos relacionados

Observaciones

- Las configuraciones en este documento se realizaron en base a la instalación UPI Bare-Metal. Ésta no difiere mucho con la instalación en VMWare, igualmente se añadieron las configuraciones para VMWare en paralelo.
- Se debe agregar más detalle a las configuraciones faltantes para la instalación UPI en VMWare.
- Todas las configuraciones de las máquinas mencionadas en este documento fueron probadas y ejecutadas con éxito, con excepción del *"HAProxy Load Balancer"*.
- En el aparato de *"Nodos"*, solo se instaló y configuró el nodo Bootstrap. Los master y workers deberían seguir los mismos pasos, pero con sus archivos correspondientes. De igual manera queda pendiente documentar el proceso para estas máquinas.

## Pre-Requisitos.

---

- ☒ Definir método de instalación. En este caso se utilizará el método UPI (User Provisioned Infrastructure).
- ☒ Modelar infraestructura del cluster.
- ☒ Descargar Sistema Operativo RHCOS según arquitectura y método de instalación.
- ☒ Definir plataforma en donde se instalará el cluster.
- ☐ Habilitar puerto 443 en vCenter y los hosts ESXi.
- ☐ Permitir sitios requeridos para la instalación de OCP.
- ☐ VMware vSphere v7.0 Update 2 o posterior.
- ☐ vCenter 7.0 Update 2 o posterior.
- ☐ VM version 15 o posterior.
- ☐ No controladores vSphere CSI third party en el cluster.
- ☐ Definir permisos para la cuenta de servicio de vCenter.
- ☐ Crear recursos necesarios para la instalación de OCP en vCenter.
- ☐ Configurar DNS para la instancia de vCenter que contendrá el cluster de OCP.
- ☒ Definir cantidad de nodos y recursos para el cluster.
- ☐ Implementar método de aprobación de los cluster signing requests (CSRs).
- ☒ Configurar DHCP para servir IPs y hostnames a los nodos del cluster.
- ☐ Validar direcciones MAC de las interfaces ethernet de cada VM que cumplan con el rango definido por VMware Organizationally Unique Identifier (OUI).
- ☒ Configurar Load Balancer, su API e Ingress.
- ☐ Validar configuración DNS.

## Descargas

- [Coreos Installer](#)
- [Openshift V4 Arch x86\\_64](#)
- [Openshift V4 Otras Arch](#)
- [Openshift en vSphere/VMWare UPI Installation Required Files](#)

***OBS: Si no se logran visualizar las imágenes, desactivar VPN.***

# 1. Máquinas

## 1. DHCP Y DNS Server

Para el servidor DHCP y DNS se utilizó Windows Server 2016.

### Configuración DHCP

DHCP

dhcp-dns-server

IPv4

Scope [192.168.10.0] OCP

Address Pool

Address Leases

Reservations

Scope Options

Policies

Server Options

Policies

Filters

IPv6

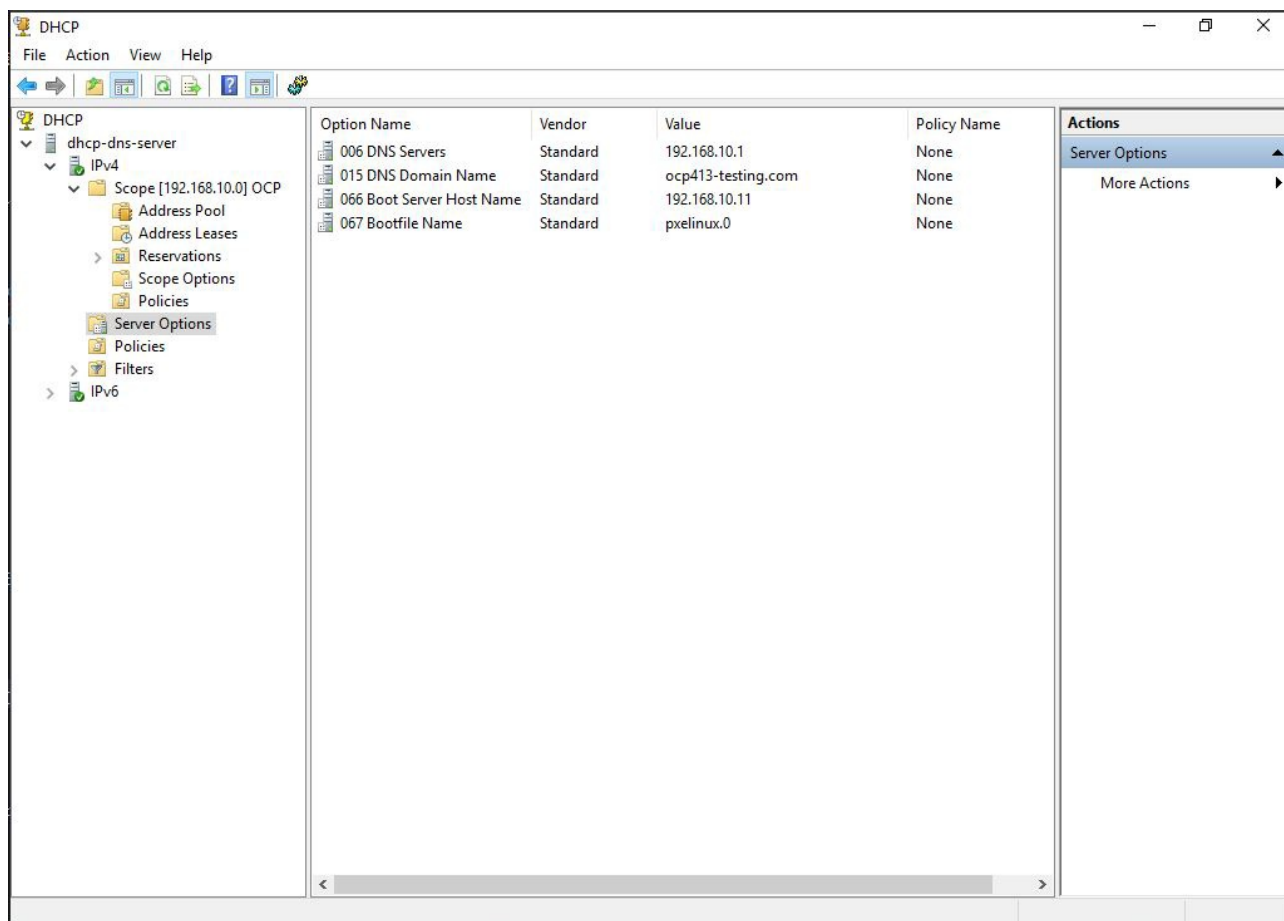
Client IP Address	Name	Lease Expiration	Type	Unique ID
192.168.10.11	bastion.ocp413-test...	07-03-2024 3:26:25	DHCP	080027b35...
192.168.10.12	registry.ocp413-test...	07-03-2024 3:28:02	DHCP	0800278ed...
192.168.10.13		07-03-2024 4:16:07	DHCP	0800271bd...
192.168.10.14	lbhaproxy.ocp413-t...	01-03-2024 10:44:50	DHCP	080027735...

Actions

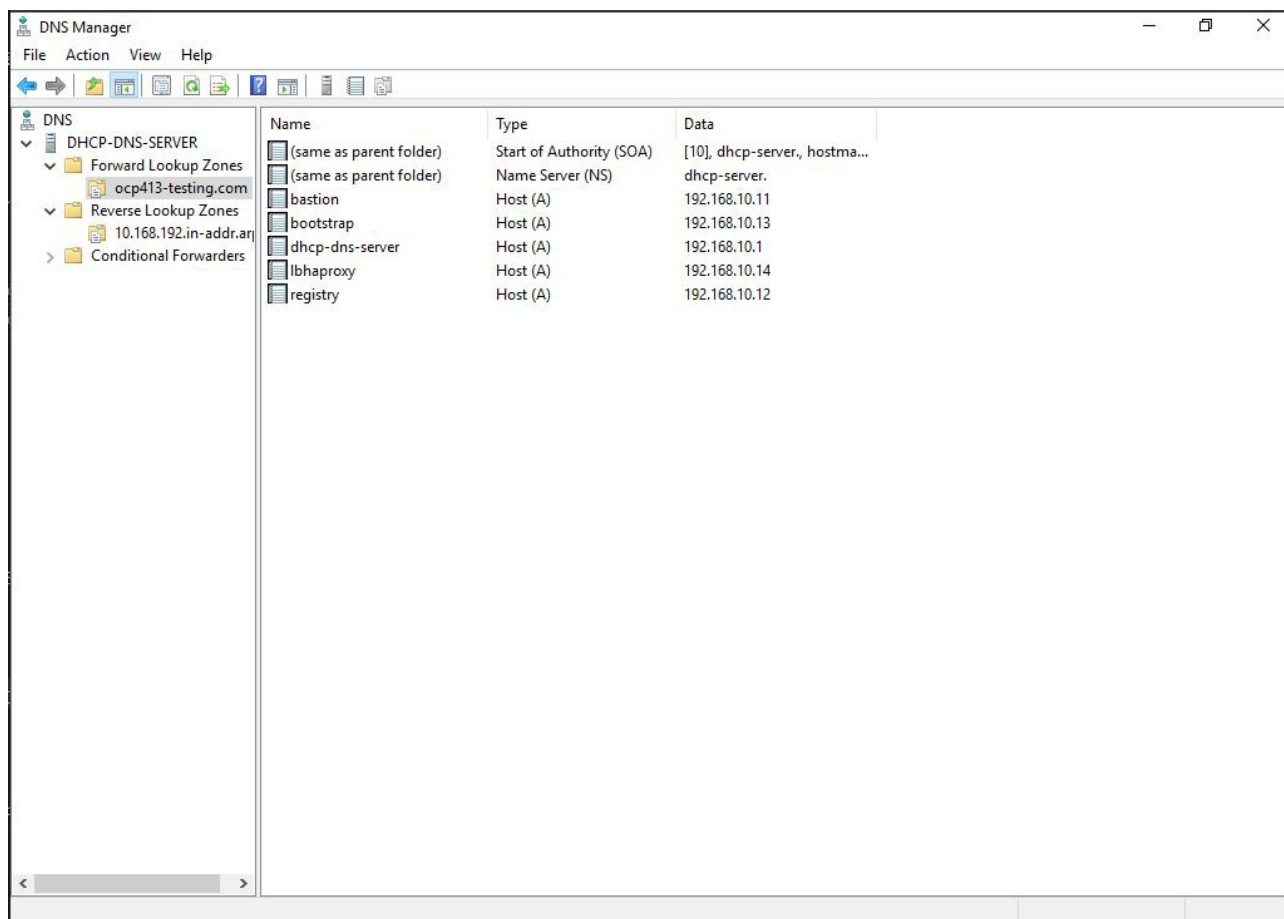
Address Leases

More Actions

### DHCP Server Options



## Configuración DNS



# Instalación Cluster OCP UPI vSphere - VMWare

DNS Manager

File Action View Help

DNS

- ✓ DHCP-DNS-SERVER
  - ✓ Forward Lookup Zones
    - ocp413-testing.com
  - ✓ Reverse Lookup Zones
    - 10.168.192.in-addr.ar
  - > Conditional Forwarders

Name	Type	Data
(same as parent folder)	Start of Authority (SOA)	[17], dhcp-server., hostma...
(same as parent folder)	Name Server (NS)	dhcp-server.
192.168.10.1	Pointer (PTR)	dhcp-dns-server.ocp413-t...
192.168.10.11	Pointer (PTR)	bastion.ocp413-testing.co...
192.168.10.12	Pointer (PTR)	registry.ocp413-testing.co...
192.168.10.13	Pointer (PTR)	bootstrap.ocp413-testing....
192.168.10.14	Pointer (PTR)	lbhaproxy.ocp413-testing....

## 2. Registry

Para la configuración del registry se deberá disponer de 2 máquinas:

- Una máquina que será la que contenga el "registry container". Lo ideal es usar un OS como CentOS o RHEL. No importa demasiado, pero es recomendable un OS de Redhat. En este ejemplo se utilizó una VM CentOS 7.
- Una máquina que tenga acceso a internet y a la red interna. Puede ser cualquier máquina, ya sea una VM con cualquier OS o el equipo personal. Lo importante es que tenga acceso a internet y a la red interna. En este ejemplo se utilizó una VM Ubuntu.

### Hay 2 formas de configurar e instalar el contenedor Registry.

- Que el contenedor registry esté totalmente aislado de internet y que el servidor Bastión (o el servidor que tiene acceso a internet para el mirroring de imágenes) no pueda acceder al contenedor registry.
- Que el contenedor registry sea accesible por el servidor Bastión (o el servidor que tiene acceso a internet para el mirroring de imágenes) y que esté aislado de internet.

**OBS:** Para ambos casos el servidor Registry en una primera instancia deberá tener acceso a internet para poder descargar la imagen "registry:2" de docker, necesaria para realizar el mirroring de imágenes y las dependencias necesarias. Una vez terminada la creación del contenedor, se podrá aislar la máquina del registry.

**Primer método: Registry aislado del servidor Bastion. OBS!!:** Este metodo no me funcionó. Al parecer hay un bug con el penúltimo comando que hay que ejecutar (oc image mirror). Dejo referencias del posible bug:

- [Referencia de posible bug en comunidad de Redhat 1](#)
- [Referencia de posible bug en comunidad de Redhat 2](#)

## 1. Creación del contenedor registry

### Instalación de podman

```
[root@home] yum install -y podman
```

### Instalar httpd-tools

```
[root@home] yum install -y httpd-tools
```

### Crear directorios para el registry

```
[root@home] mkdir -p /opt/registry/auth  
[root@home] mkdir -p /opt/registry/certs  
[root@home] mkdir -p /opt/registry/data
```

- El directorio /opt/registry/auth es donde se almacenarán los usuarios y contraseñas para el acceso al registry.
- El directorio /opt/registry/certs es donde se almacenarán los certificados para la autenticación del registry.
- El directorio /opt/registry/data es donde se almacenarán las imágenes del registry.

### Generar credenciales para el acceso al registry.

```
[root@home] htpasswd -bBc /opt/registry/auth/htpasswd <username> <password>
```

- **-b:** Proporciona la contraseña en la línea de comandos.
- **-B:** Utiliza el algoritmo bcrypt para cifrar la contraseña.

# Instalación Cluster OCP UPI vSphere - VMWare

- **-c:** Crea un nuevo archivo de contraseñas. Si el archivo ya existe, lo sobrescribe.
- **username:** Nombre de usuario para el acceso al registry.
- **password:** Contraseña del usuario para el acceso al registry.

Esto generará un archivo con el nombre de usuario y una contraseña en base64 para el acceso al registry como se muestra a continuación:

```
[root@registry auth]# ls -lrt
total 4
-rw-r--r--. 1 root root 68 feb  8 12:16 htpasswd
[root@registry auth]# cat htpasswd
jotape:$2y$05$K.xsRaI0Xd8vTSwF553Lxe8fCotL0NR0kHZaYca1VEuTzkG5kXqNq
```

## Creación del certificado TLS SAN (Subject Alternative Name). En este ejemplo se utilizará un certificado self-signed.

### 1. Ingresar al directorio `/opt/registry/certs`.

### 2. Generar una llave privada

```
[root@certs] openssl genrsa -des3 -out registry.key 2048
```

- **genrsa:** Genera una llave RSA.
- **-des3:** Cifra la llave con el algoritmo DES3.
- **-out:** Especifica el nombre del archivo de salida.
- **2048:** Especifica el tamaño de la llave en bits.

### 3. Generar CSR (Certificate Signing Request)

```
[root@certs] openssl req -new -key registry.key -out registry.csr
```

- **req:** Utiliza el comando de solicitud de certificado.
- **-new:** Crea una nueva solicitud de certificado CSR.
- **-key:** Especifica la llave privada. (registry.key).
- **-out:** Especifica el nombre del archivo de salida. (registry.csr).

Cuando pida ingresar el "Common Name", ingresar el hostname de la máquina. En este ejemplo se utilizó el CN = registry

### 4. Remover Passphrase de la llave

```
[root@certs] cp registry.key registry.key.org
[root@certs] openssl rsa -in registry.key.org -out registry.key
```

### 5. Crear un config file para SAN

```
[root@certs] touch v3.ext
```

- Contenido del archivo:

```
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always,issuer:always
basicConstraints        = CA:TRUE
keyUsage                = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment, keyAgreement,
keyCertSign
subjectAltName          = DNS:registry
issuerAltName           = issuer:copy
```

### 6. Generar el certificado self-signed



```
[root@certs] openssl x509 -req -in registry.csr -signkey registry.key -out registry.crt -days 365 -sha256 -extfile v3.ext
```

- **x509**: Utiliza el comando de certificado x509. x509 es un certificado digital estándar.
- **-req**: Se indica que se está trabajando con una solicitud de certificado CSR como input.
- **-in**: Especifica el archivo de entrada que contiene la solicitud de firma de certificado (CSR) que se quiere firmar. (registry.csr).
- **-signkey**: Especifica el archivo que contiene la llave privada con el que se firmará la solicitud del certificado. (registry.key).
- **-out**: Especifica el nombre del archivo de salida. (registry.crt).
- **-days**: Especifica la cantidad de días que el certificado será válido. (1 año).
- **-sha256**: Especifica el algoritmo de hash que se utilizará para firmar el certificado.
- **-extfile**: Especifica el archivo de configuración que contiene el Subject Alternative Name (SAN). (v3.ext).

## 7. Añadir certificado a los trusted certificates del sistema (CentOS).

```
[root@certs] cp /opt/registry/certs/registry.crt /etc/pki/ca-trust/source/anchors/
```

```
[root@certs] update-ca-trust
```

```
[root@certs] trust list | grep -i "registry"
```

## 8. Iniciar el registry

```
[root@certs] podman run --name registry -p 5000:5000 -v /opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -e "REGISTRY_AUTH=htpasswd" -e "REGISTRY_AUTH_HTPASSWD_REALM=Registry Realm" -e "REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd" -v /opt/registry/certs:/certs:z -e "REGISTRY_HTTP_TLS_CERTIFICATE=/certs/registry.crt" -e "REGISTRY_HTTP_TLS_KEY=/certs/registry.key" -e REGISTRY_COMPATIBILITY_SCHEMA1_ENABLED=true -d registry:2
```

- **run**: Crea y ejecuta un contenedor.
- **--name**: Nombre del contenedor. (registry).
- **-p**: Mapea el puerto del contenedor al puerto del host. El puerto 5000 es un estandar de este tipo de contenedores. (5000:5000).
- **-v /opt/registry/data:/var/lib/registry:z**: Monta un volumen en el contenedor. Esto permite almacenar los datos del registry de Docker en el directorio "/opt/registry/data" del host, que estará disponible en el contenedor en "/var/lib/registry". El modificador "registry:z" indica que se debe establecer el contexto de seguridad SELinux para el volumen.
- **-v /opt/registry/auth:/auth:z**: Monta otro volumen para almacenar la autenticación del registry. Esto permite almacenar los archivos de autenticación en el directorio "/opt/registry/auth" del host, que estará disponible en el contenedor en "/auth".
- **-e "REGISTRY\_AUTH=htpasswd"**: Establece una variable de entorno dentro del contenedor llamada "REGISTRY\_AUTH" con el valor "htpasswd", lo que indica que se utilizará la autenticación basada en htpasswd.
- **-e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm"**: Se indica que las credenciales proporcionadas para acceder al registry de Docker deben aplicarse al "reino" llamado "Registry Realm". Esto significa que cuando un usuario intente acceder al registry, se le solicitarán las credenciales para el "reino" especificado, que en este caso es "Registry Realm".
- **-e "REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd"**: Especifica la ruta dentro del contenedor donde se encuentra el archivo htpasswd para la autenticación.
- **-v /opt/registry/certs:/certs:z**: Monta un volumen para almacenar los certificados TLS. Esto permite almacenar los certificados en el directorio "/opt/registry/certs" del host, que estará disponible en el contenedor en "/certs".
- **-e "REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/registry.crt"**: Especifica la ubicación del certificado TLS para el registry.
- **-e "REGISTRY\_HTTP\_TLS\_KEY=/certs/registry.key"**: Especifica la ubicación de la clave TLS para el registry.
- **-e REGISTRY\_COMPATIBILITY\_SCHEMA1\_ENABLED=true**: Habilita la compatibilidad con el esquema de almacenamiento de Docker versión 1. (Opcional) No sé si es necesario.
- **-d registry:2**: Especifica la imagen del contenedor a ejecutar. En este caso, se utiliza la imagen "registry:2", que es una imagen oficial de Docker para ejecutar un registry de Docker de la versión 2.

## 9. Habilitar puerto 5000

```
[root@home] firewall-cmd --add-port=5000/tcp --zone=internal --permanent

[root@home] firewall-cmd --add-port=5000/tcp --zone=public --permanent

[root@home] firewall-cmd --reload
```

## 10. Descargar pull secret. Una vez descargado, guardarlo en formato JSON.

- [Link Pull Secret](#)

```
[root@home] cat ./pull-secret | jq . > ~/containers/auth.json.
```

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWZhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6",
    },
    "quay.io": {
      "auth": "b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWZhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6",
    },
    "registry.connect.redhat.com": {
      "auth": "fHVoYy1wb29sLTgyNDJiZjFjLWQzYjgtNDdjMyliNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlK",
    },
    "registry.redhat.io": {
      "auth": "fHVoYy1wb29sLTgyNDJiZjFjLWQzYjgtNDdjMyliNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlK",
    },
    "https://registry:5000": {
      "auth": "am90YXB0mpwYWwNTk4"
    }
  }
}
```

- Si no aparecen los registry de redhat, quay.io y el registry local, se deberá agregarlos al registry.

```
[root@home] podman login registry.redhat.io
[root@home] podman login quay.io
[root@home] podman login registry:5000
```

- O ejecutar los siguientes comandos para agregar los registry automáticamente al auth.json.

```
[root@home] podman login registry.redhat.io --authfile ~/containers/auth.json
[root@home] podman login quay.io --authfile ~/containers/auth.json
[root@home] podman login registry:5000 --authfile ~/containers/auth.json
```

## 11. Una vez habilitado el firewall, se puede aislar la máquina (desconectarla de internet, pero que permanezca dentro de la red interna).

## 12. Verificar acceso al contenedor registry.

- Verificar si el contenedor está corriendo

```
[root@home] podman ps
```

- Ejecutar

```
[root@home] curl https://registry:5000/v2/_catalog

OUTPUT: {"repositories":[]}
```

### 3. Configuración Bastión

#### 1. Agregar el certificado generado en la máquina del registry a los trusted certificates del servidor Bastión.

- Ubuntu:

```
[root@home] cd /usr/local/share/ca-certificates
[root@ca-certificates] openssl s_client -connect registry:5000 -servername registry
```

- Copiar el certificado desde ----BEGIN CERTIFICATE---- hasta ----END CERTIFICATE----- , crear un archivo .crt y copiar el contenido dentro.

```
[root@ca-certificates] touch registry.crt
[root@ca-certificates] nano registry.crt
```

- Actualizar los trusted certificates para añadir el certificado del registry.

```
[root@ca-certificates] update-ca-certificates
```

#### 2. Descargar pull secret. Una vez descargado, guardarlo en formato JSON.

- [Link Pull Secret](#)

```
[root@home] cat ./pull-secret | jq . > ~/containers/auth.json.
```

```
{
  "auths": {
    "cloud.openshift.com": {
      "auth": "b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWRhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6"
    },
    "quay.io": {
      "auth": "b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWRhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6"
    },
    "registry.connect.redhat.com": {
      "auth": "fHV0Yy1wb29sLTgyNDJiZjFjLWQzYjgtNDdjMyliNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlK"
    },
    "registry.redhat.io": {
      "auth": "fHV0Yy1wb29sLTgyNDJiZjFjLWQzYjgtNDdjMyliNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlK"
    },
    "https://registry:5000": {
      "auth": "am90YXB0mpwYwWwNTk4"
    }
  }
}
```

- Si no aparecen los registry de redhat, quay.io y el registry local, se deberá agregarlos al registry.

```
[root@home] podman login registry.redhat.io
[root@home] podman login quay.io
[root@home] podman login registry:5000
```

- O ejecutar los siguientes comandos para agregar los registry automáticamente al auth.json.

```
[root@home] podman login registry.redhat.io --authfile ~/containers/auth.json
[root@home] podman login quay.io --authfile ~/containers/auth.json
[root@home] podman login registry:5000 --authfile ~/containers/auth.json
```

#### 3. En el archivo .bashrc del usuario root, agregar las siguientes variables de

## entorno.

```
export OCP_RELEASE=4.13.10
export LOCAL_REGISTRY=registry:5000
export LOCAL_REPOSITORY=ocp4/openshift4
export PRODUCT_REPO=openshift-release-dev
export RELEASE_NAME=ocp-release
export LOCAL_SECRET_JSON=~/.containers/auth.json
export ARCHITECTURE=x86_64
export REMOVABLE_MEDIA_PATH=~/.images
```

- **OCP\_RELEASE:** Versión de OCP a instalar.
- **LOCAL\_REGISTRY:** Dominio del contenedor registry local y puerto de la máquina que contiene el contenedor registry.
- **LOCAL\_REPOSITORY:** Nombre que tendrá el repositorio en el registry local.
- **PRODUCT\_REPO:** Repositorio de Redhat. Para producción se ocupa openshift-release.
- **RELEASE\_NAME:** Nombre del release. Para producción se ocupa ocp-release.
- **LOCAL\_SECRET\_JSON:** Ruta del archivo auth.json.
- **ARCHITECTURE:** Arquitectura de la máquina. Pueden ser x86\_64, aarch64, s390x, or ppc64le.
- **REMOVABLE\_MEDIA\_PATH:** Ruta donde se almacenarán las imágenes.

## 4. Una vez configuradas las variables de entorno, ejecutar el siguiente comando.

```
[root@~] source .bashrc
```

## 5. Verificado lo anterior, ejecutar el siguiente comando.

- Este comando lleva un parametro "--dry-run" que permite verificar si el comando se ejecutará correctamente y no ejecutará nada.
- OBS: Ejecutar este comando en el servidor registry, para hacer esto, se requiere que el servidor registry tenga instalado el CLI "oc", "oc-mirror" y "kubectl". Tambien es necesario que se configuren las mismas variables de entorno que se configuraron en el servidor Bastion, solo si es que el servidor Bastion no tiene acceso al contenedor registry.

```
[root@home] oc adm release mirror -a ${LOCAL_SECRET_JSON} --
from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} -
-to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE} --dry-run
```

## 6. Al terminar la ejecución del comando anterior, entregará un output como el siguiente:

To use the new mirrored repository to install, add the following section to the install-config.yaml:

```
imageContentSources:
  - mirrors:
    - registry:5000/ocp4/openshift4
      source: quay.io/openshift-release-dev/ocp-release
  - mirrors:
    - registry:5000/ocp4/openshift4
      source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

To use the new mirrored repository for upgrades, use the following to create an ImageContentSourcePolicy:

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: example
spec:
  repositoryDigestMirrors:
    - mirrors:
      - registry:5000/ocp4/openshift4
        source: quay.io/openshift-release-dev/ocp-release
    - mirrors:
      - registry:5000/ocp4/openshift4
        source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

- Es importante guardar este output ya que se utilizará en el archivo de configuración "install-config.yaml" para la instalación del cluster.

## 7. Descargar las imágenes.

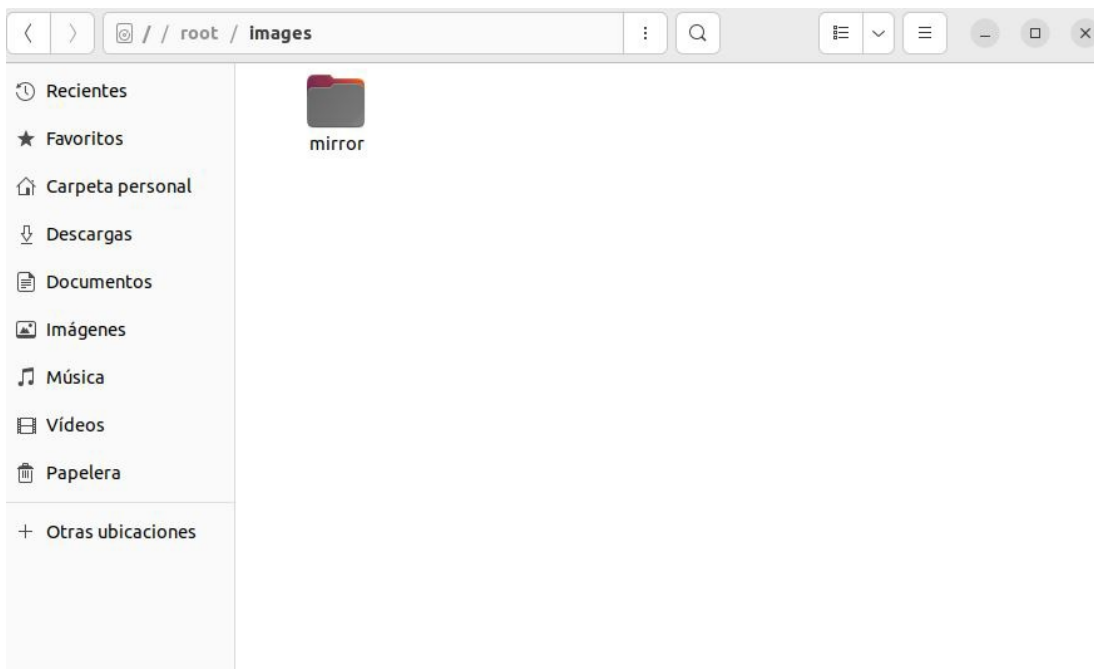
- En este caso se descargó la version 4.13 de OCP. Fueron alrededor de 18 GB de imágenes que demoró más o menos 30 minutos.

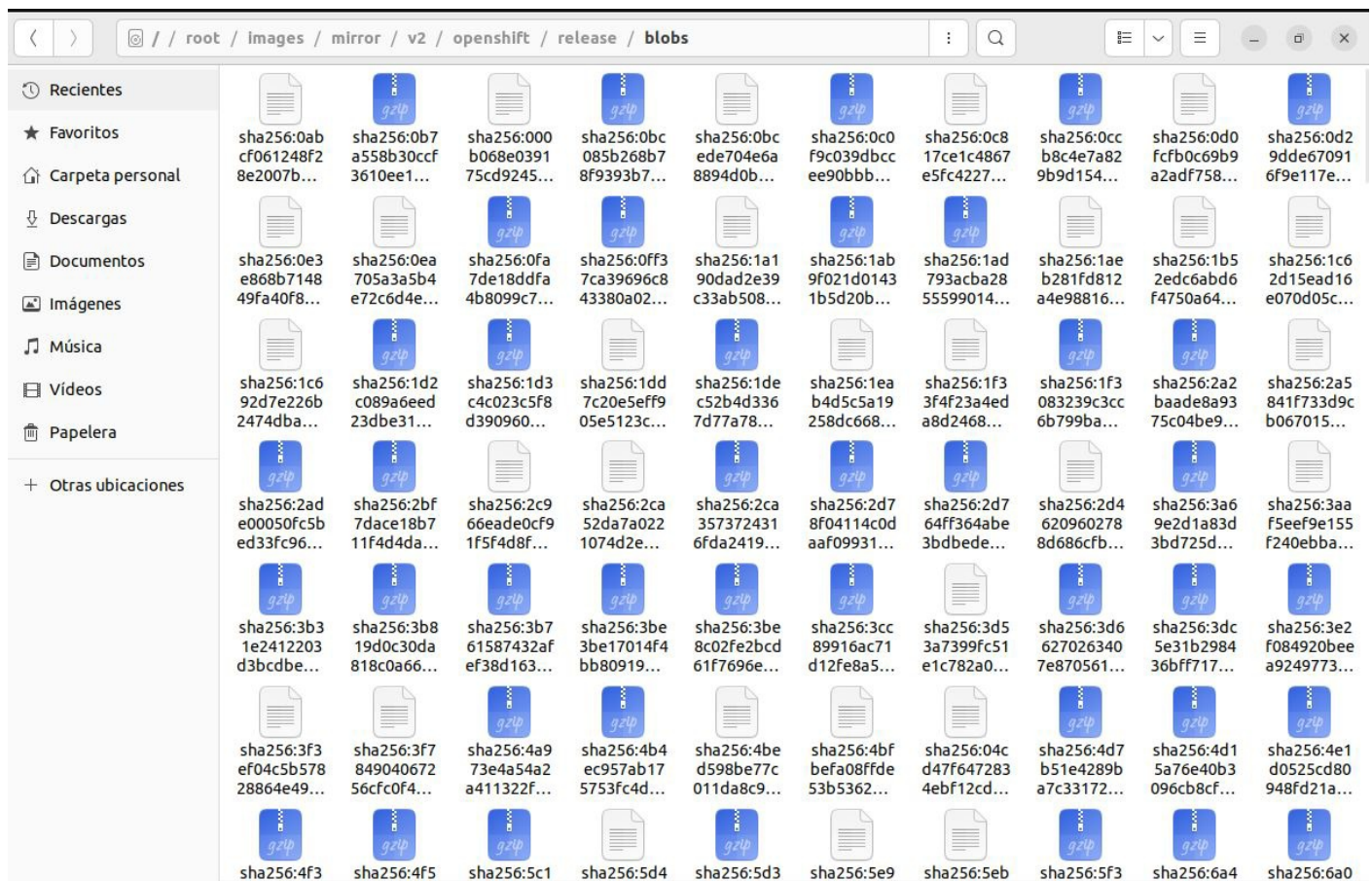
```
[root@home] oc adm release mirror -a ${LOCAL_SECRET_JSON} --to-dir=${REMOVABLE_MEDIA_PATH}/mirror
quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE}
```

## 8. Al terminar la ejecución del comando anterior, verificar si las imágenes se descargaron correctamente.

- Debería haber una carpeta con el nombre "mirror" en la ruta especificada en la variable de entorno "REMOVABLE\_MEDIA\_PATH".

```
[root@home] ls -l ~/images/mirror
```





## 9. Una vez terminada la ejecución del comando anterior, traspasar las imágenes al servidor registry.

- OJO: Es necesario habilitar una conexión SSH entre el servidor Bastión y el servidor registry. Esta operación también demoró más o menos 30 minutos.

```
[root@~] scp -r ./images root@192.168.10.12:~/images
```

## 10. Al terminar el traspaso de los archivos, ir al servidor registry y hacer un deploy de las imágenes dentro del contenedor creado previamente.

- OBS: Éste es el comando que se mencionó al principio que genera conflicto. Es el único comando que no logré ejecutar al realizar este método.

```
[root@~] oc image mirror -a ${LOCAL_SECRET_JSON} --from-dir=${REMOVABLE_MEDIA_PATH}/mirror
"file://openshift/release:${OCP_RELEASE}*" ${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} --skip-missing
```

## 11. Verificar imágenes dentro del registry.

- Una vez finalizado el paso anterior, nos ubicamos en el servidor Registry. Verificar si las imágenes se encuentran en el registry.

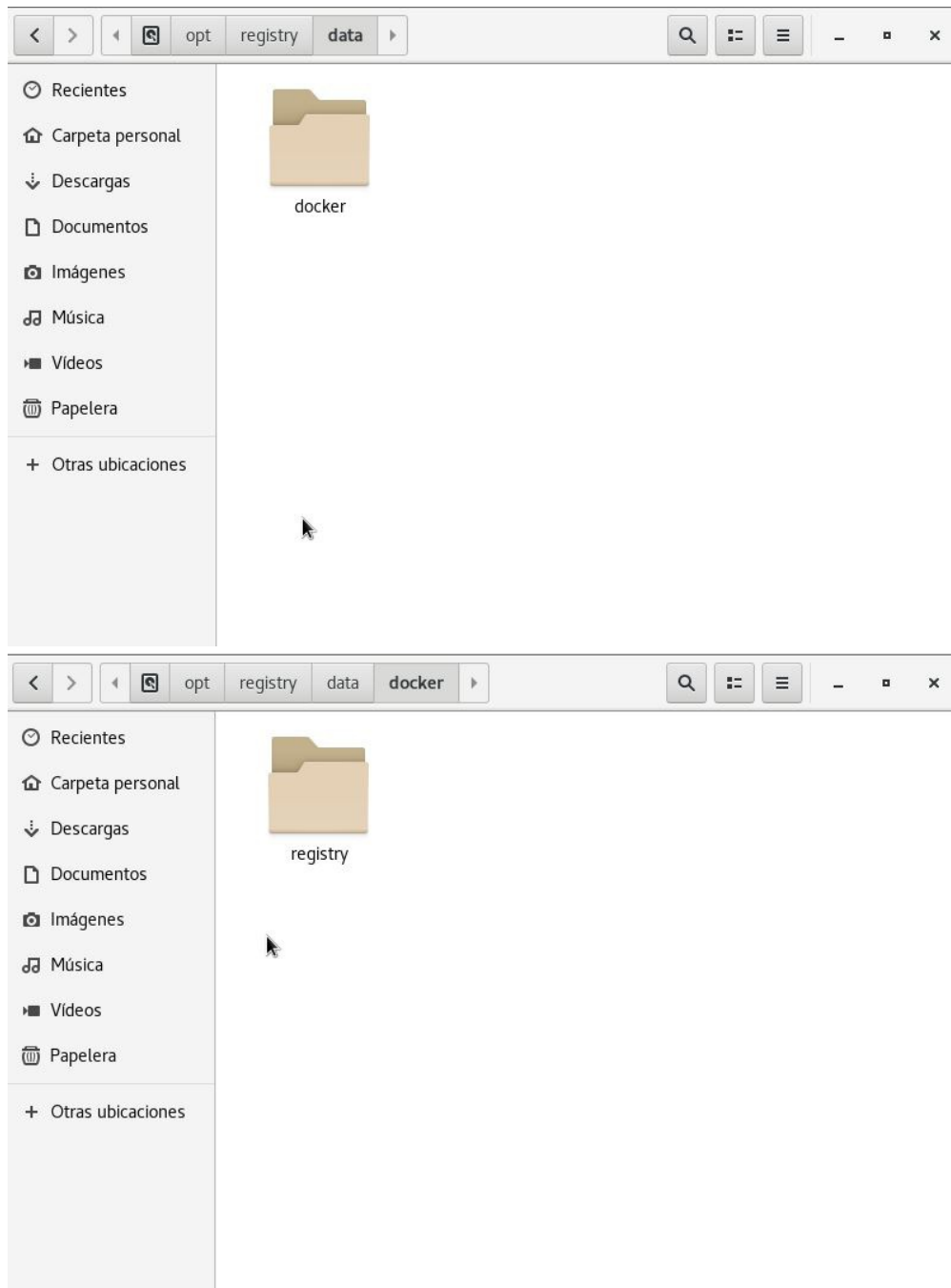
```
[root@home] curl -u usuario:password https://registry:5000/v2/_catalog

{"repositories":["ocp4/openshift4"]}
```

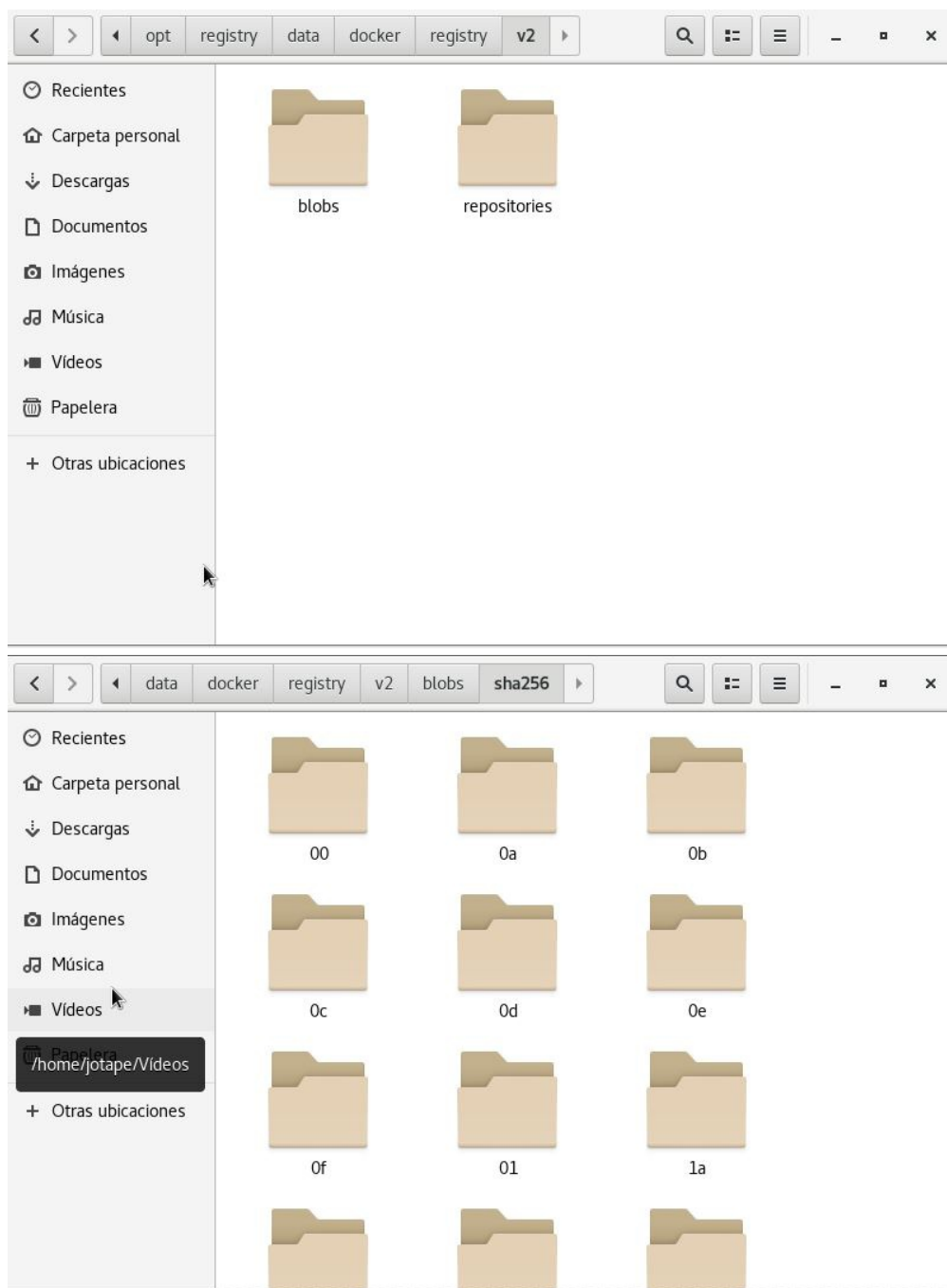
```
[root@home] curl -u usuario:password https://registry:5000/v2/ocp4/openshift4/tags/list
```

OUTPUT: JSON con las imágenes desplegadas en el contenedor.

## 12. Evidencias de archivos en el registry.

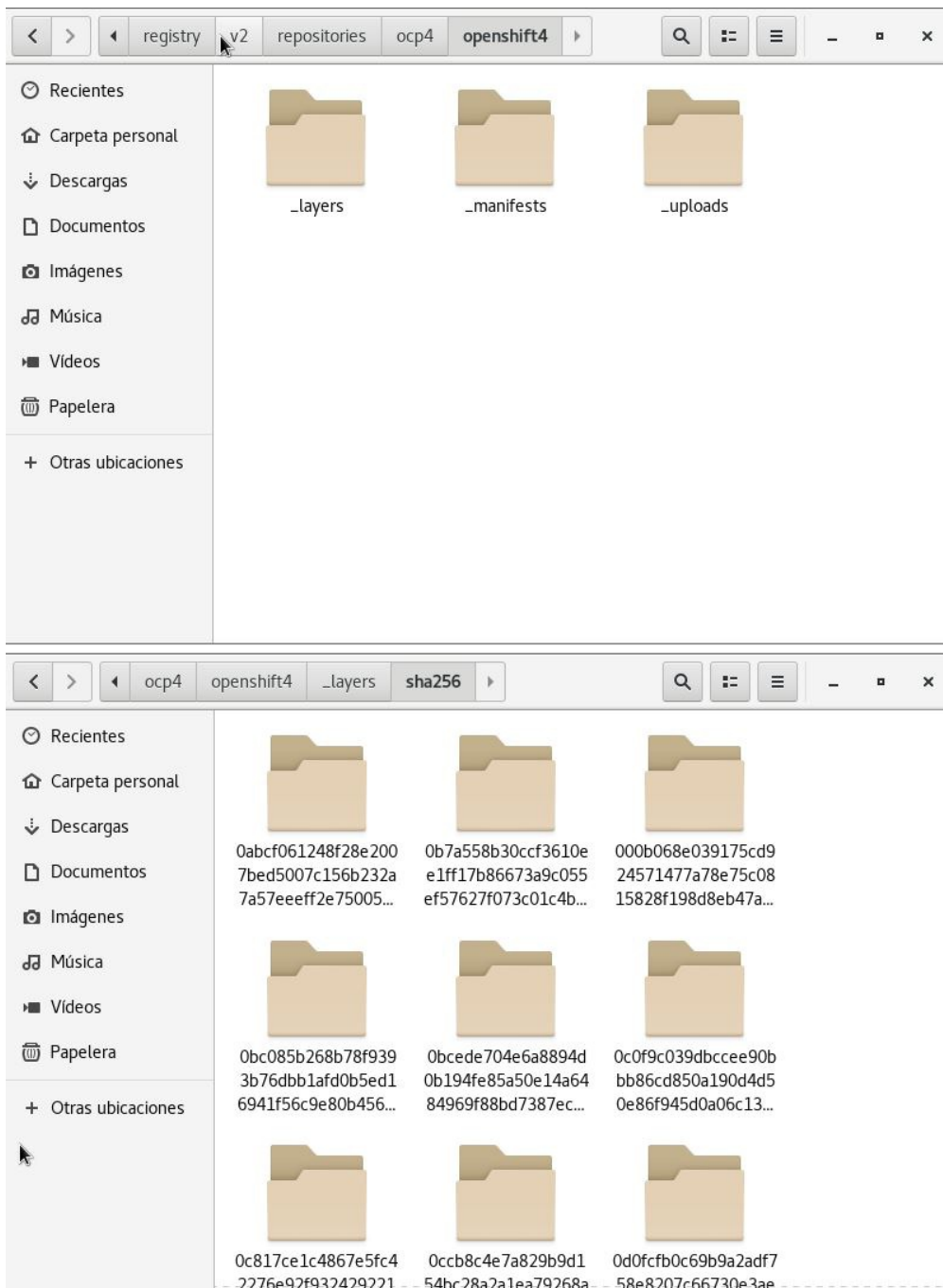


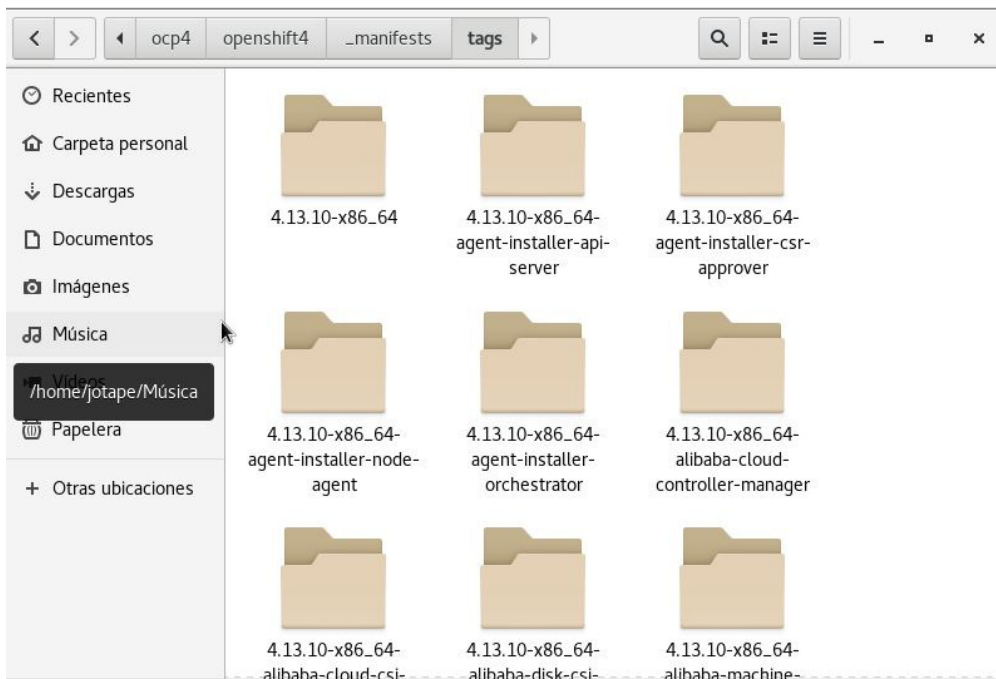






# Instalación Cluster OCP UPI vSphere - VMWare





## Segundo método: Contenedor registry accesible por el servidor Bastión.

- Este metodo es muy parecido al primero, solo que no es necesario descargar las imágenes en el servidor Bastion en un directorio y luego traspasarlas al servidor Registry. En este caso, despues de ejecutar el comando (que esta vez se puede ejecutar directamente desde el servidor Bastion.):

```
[root@home] oc adm release mirror -a ${LOCAL_SECRET_JSON} --  
from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} -  
-to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE} --dry-run
```

- Se procede a hacer el mirroring desde el mismo Bastion, obteniendo las imágenes directamente desde el repositorio oficial de RedHat, ya que el contenedor pertenece a la red interna en la que se está trabajando y que el servidor Bastión tiene conexión a Internet. (Sin el parametro "--dry-run"). Al terminar, las imágenes deberían estar en la ruta configurada al principio en el servidor registry `/opt/registry/data`.

```
[root@home] oc adm release mirror -a ${LOCAL_SECRET_JSON} --  
from=quay.io/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE}-${ARCHITECTURE} --to=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY} -  
-to-release-image=${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}
```

To use the new mirrored repository to install, add the following section to the install-config.yaml:

```
imageContentSources:  
  - mirrors:  
    - registry:5000/ocp4/openshift4  
      source: quay.io/openshift-release-dev/ocp-release  
  - mirrors:  
    - registry:5000/ocp4/openshift4  
      source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

To use the new mirrored repository for upgrades, use the following to create an ImageContentSourcePolicy:

```
apiVersion: operator.openshift.io/v1alpha1  
kind: ImageContentSourcePolicy  
metadata:  
  name: example  
spec:  
  repositoryDigestMirrors:  
    - mirrors:  
      - registry:5000/ocp4/openshift4  
        source: quay.io/openshift-release-dev/ocp-release  
    - mirrors:  
      - registry:5000/ocp4/openshift4  
        source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

## 3. Nodos

---

### 1. Bootstrap

#### 1. Configuración

- Para el efecto de este ejercicio, se utilizó el servidor Bastión (Ubuntu) como bootstrap.

#### 1. Descargar syslinux - tftpd-hpa - apache2 - pxelinux

```
[root@home] apt-update  
[root@home] apt install tftpd-hpa apache2 syslinux pxelinux
```

#### 2. Obtener ignition files. Para obtenerlos, primero hay que crear el archivo "install-config.yaml".

- Crear directorio donde alojar el archivo "install-config.yaml", archivos ignition e imágenes para instalar RHCOS.

```
[root@home] mkdir ~/ignition-files | cd ~/ignition-files  
  
[root@ignition-files] touch install-config.yaml  
  
[root@ignition-files] nano install-config.yaml
```

- install-config.yaml

# Instalación Cluster OCP UPI vSphere - VMWare

```
apiVersion: v1
baseDomain: ocp413-testing.com
compute:
  - hyperthreading: Enabled
    name: worker
    replicas: 0 # Debe ser 0 ya que en instalación UPI los workers se crean manualmente.
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: test # Nombre del cluster
imageContentSources:
  - mirrors:
    - registry:5000/ocp4/openshift4
      source: quay.io/openshift-release-dev/ocp-release
  - mirrors:
    - registry:5000/ocp4/openshift4
      source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
networking:
  clusterNetwork:
    - cidr: 10.128.0.0/14 # Un bloque de direcciones IP a partir del cual se asignan las direcciones IP de los pods.
      Este bloque no debe solaparse con redes físicas existentes. Estas direcciones IP se utilizan para la red de pods. Si
      necesita acceder a los pods desde una red externa, debe configurar equilibradores de carga y enrutadores para gestionar
      el tráfico.
    hostPrefix: 23
    networkType: OpenShiftSDN
    serviceNetwork:
      - 172.30.0.0/16 # El grupo de direcciones IP a utilizar para las direcciones IP de servicio. Sólo se puede
        introducir un grupo de direcciones IP. Este bloque no debe solaparse con redes físicas existentes. Si necesita acceder a
        los servicios desde una red externa, configure balanceadores de carga y enrutadores para gestionar el tráfico.
  platform:
    none: {} # Aquí debería ir la configuración de vSphere y vCenter. En este caso se deja en blanco.
  fips: false
  pullSecret: '{
    "auths": {
      "cloud.openshift.com": {
        "auth":
"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWRhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6Qk9IU1UwVjZFS0VYUUVZYUEE2T1dKSUcyW
        "email": "jparancibialinker@gmail.com"
      },
      "quay.io": {
        "auth":
"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K29jbV9hY2Nlc3NfOWU3MTlhZWRhMmY1NDRIzjk2Mjc4MTFjNzVmMTNjOGM6Qk9IU1UwVjZFS0VYUUVZYUEE2T1dKSUcyW
        "email": "jparancibialinker@gmail.com"
      },
      "registry.connect.redhat.com": {
"auth": "fHVoYy1wb29sLTgyNDJiNzFjLWQzYjgtNDdjMy1iNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJaU9pSXpNV0UxWX
        "email": "jparancibialinker@gmail.com"
      },
      "registry.redhat.io": {
"auth": "fHVoYy1wb29sLTgyNDJiNzFjLWQzYjgtNDdjMy1iNGNiLWExMDI3ZjAzZmY2NjpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJaU9pSXpNV0UxWX
        "email": "jparancibialinker@gmail.com"
      },
      "registry.5000": {
        "auth": "am90YXBLOmpwYWwwNTk4",
        "email": "jparancibialinker@gmail.com"
      }
    }
  }' # Se copia el contenido del archivo auth.json configurado previamente.
sshKey: "ssh-ed25519 AAAA..." # La clave pública SSH para el usuario core en Red Hat Enterprise Linux CoreOS
(RHCOS). En mi caso puse la llave pública de mi servidor Bastión.
```

- En el caso de que la instalación se realice en una VMWare, modificar el siguiente parametro del archivo "install-config.yaml"

```
platform:
  vsphere:
    failureDomains:
      - name: <failure_domain_name>
        region: <default_region_name>
        server: <fully_qualified_domain_name>
    topology:
      computeCluster: "/<datacenter>/host/<cluster>"
      datacenter: <datacenter>
      datastore: "/<datacenter>/datastore/<datastore>"
      networks:
        - <VM_Network_name>
      resourcePool: "/<datacenter>/host/<cluster>/Resources/<resourcePool>"
      folder: "/<datacenter_name>/vm/<folder_name>/<subfolder_name>"
      zone: <default_zone_name>
  vcenters:
    - datacenters:
      - <datacenter>
      password: <password>
      port: 443
      server: <fully_qualified_domain_name>
      user: administrator@vsphere.local
    diskType: thin
```

## Segun documentación:

- **failureDomain:** Establece las relaciones entre una región y una zona. Un dominio de fallos se define mediante objetos vCenter, como un objeto datastore. Un dominio de fallos define la ubicación de vCenter para los nodos de clúster de OpenShift Container Platform.
- **datacenter:** VMWare / vSphere datacenter.
- **datastore:** La ruta al almacén de datos de vSphere que contiene archivos de máquinas virtuales, plantillas e imágenes ISO. **IMPORTANTE:** Puede especificar la ruta de cualquier almacén de datos que exista en un clúster de almacenes de datos. Por defecto, Storage vMotion se habilita automáticamente para un cluster de datastore. Red Hat no soporta Storage vMotion, por lo que debe desactivar Storage vMotion para evitar problemas de pérdida de datos en su cluster OpenShift Container Platform. Si debe especificar VMs a través de múltiples datastores, utilice un objeto datastore para especificar un dominio de fallo en el archivo de configuración install-config.yaml de su cluster. Para obtener más información, consulte "Habilitación de regiones y zonas de VMware vSphere".
- **resourcePool:** Opcional: Para la infraestructura aprovisionada por el instalador, la ruta absoluta de un pool de recursos existente donde el programa de instalación crea las máquinas virtuales, por ejemplo, /<nombre\_centro\_de\_datos>/host/<nombre\_cluster>/Recursos/<nombre\_pool\_de\_recursos>/<nombre\_pool\_de\_recursos\_anidado\_opcional>. Si no se especifica ningún valor, los recursos se instalan en la raíz del clúster /ejemplo\_centro\_de\_datos/host/ejemplo\_cluster/Recursos.
- **folder:** Opcional: Para la infraestructura aprovisionada por el instalador, la ruta absoluta de una carpeta existente en la que el programa de instalación crea las máquinas virtuales, por ejemplo, /<nombre\_centro\_de\_datos>/vm/<nombre\_carpeta>/<nombre\_subcarpeta>. Si no proporciona este valor, el programa de instalación crea una carpeta de nivel superior en la carpeta de máquinas virtuales del centro de datos que se denomina con el ID de infraestructura. Si está proporcionando la infraestructura para el clúster y no desea utilizar el objeto StorageClass predeterminado, denominado thin, puede omitir el parámetro de carpeta del archivo install-config.yaml.
- **password:** La contraseña asociada al usuario VMWare / vSphere.
- **server:** El nombre de host completo o la dirección IP del servidor vCenter.
- **diskType:** El método de aprovisionamiento de disco de VMWare / vSphere.

3. Ejecutar el siguiente comando para obtener los ignition files. Este comando "consume" el archivo "install-config.yaml". En este ejercicio éste último desapareció, no sé si es correcto que pase eso, si ese es el caso respaldar el contenido del archivo "install-config.yaml" para crearlo nuevamente para cuando se proceda a instalar el cluster.

```
[root@ignition-files] openshift-install create ignition-configs

time="2024-02-14T14:40:59-03:00" level=debug msg="Generating Metadata..."
time="2024-02-14T14:40:59-03:00" level=info msg="Ignition-Configs created in: . and auth"
```

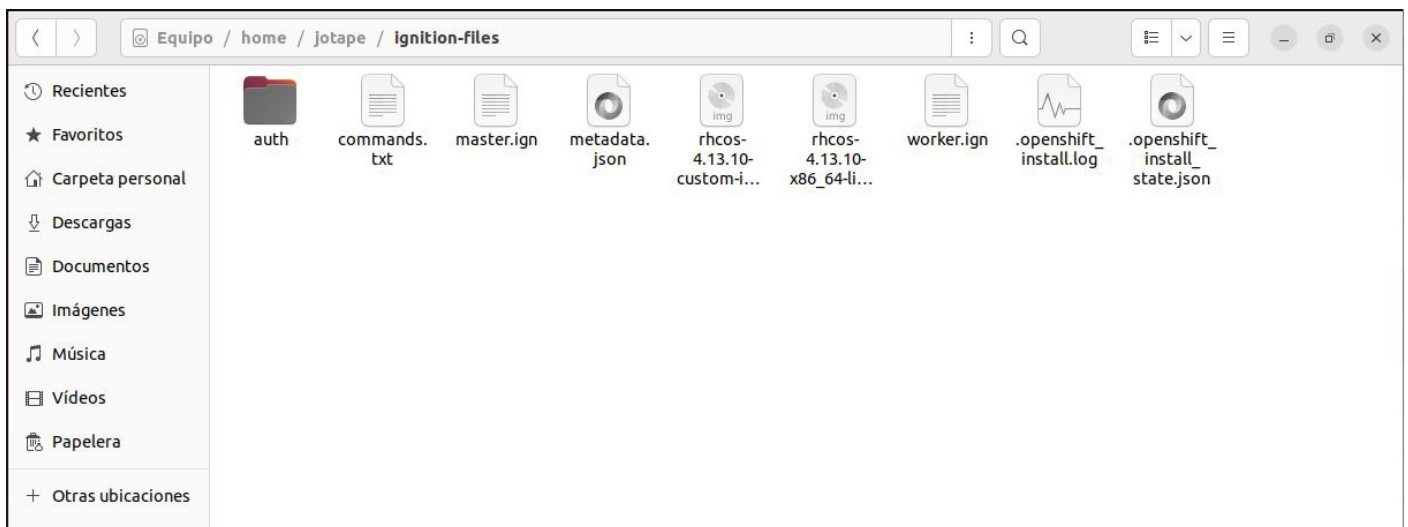
4. En el mismo directorio donde se encuentra el archivo "install-config.yaml", poner los archivos "bootstrap.ign" y "rhcos-4.13.10-x86\_64-live-initramfs.x86\_64" para crear una configuración personalizada del PXE Boot.

# Instalación Cluster OCP UPI vSphere - VMWare

Ejecutar el siguiente comando:

```
[root@ignition-files] coreos-installer pxe customize rhcos-4.13.10-x86_64-live-initramfs.x86_64.img(1) --dest-ignition /home/jotape/ignition-files/bootstrap.ign(2) --dest-console tty0(3) --dest-console ttyS0,115200n8(4) --dest-device /dev/sda(5) -o rhcos-4.13.10-x86_64-custom-initramfs.x86_64.img(6)
```

- (1). Esto invoca el comando coreos-installer, que es una herramienta para instalar CoreOS. pxe customize indica que se está personalizando una imagen para arranque por red (PXE). rhcos-4.13.10-x86\_64-live-initramfs.x86\_64.img es el nombre de la imagen de inicio de CoreOS que se personalizará.
- (2). Esto especifica el destino de la configuración de Ignition, que es un sistema de inicialización de CoreOS
- (3). Esto indica que la salida de la consola se enviará a la primera consola virtual (tty0).
- (4). Aquí, ttyS0 se refiere al primer puerto serie del sistema. Este parámetro es útil en entornos donde la interacción con el sistema se realiza a través de una conexión serie, como en sistemas embebidos o servidores sin interfaz gráfica. 115200n8 especifica la velocidad de transmisión (baudrate) y la configuración del terminal para el puerto serie.
- (5). Esto especifica el punto de montaje donde se instalará CoreOS.
- (6). Esto indica el nombre del archivo de salida de la imagen personalizada.
- [Referencia de lo realizado anteriormente.](#)



## 5. Configurar el TFTP Server

```
[root@home] nano /etc/default/tftpd-hpa
```

- Ingresar lo siguiente

```
TFTP_USERNAME="tftp"  
TFTP_DIRECTORY="/var/lib/tftpboot"  
TFTP_ADDRESS="0.0.0.0:69"  
TFTP_OPTIONS="--secure --create"
```

- Guardar y salir.
- Crear directorio */var/lib/tftpboot*

```
[root@home] mkdir /var/lib/tftpboot
```

- Copiar archivos SYSLINUX dentro del directorio *var/lib/tftpboot*

```
[root@home] cp /usr/lib/syslinux/modules/bios/* /var/lib/tftpboot/
```

- Descargar imagen para pantalla de booteo. (Opcional)

# Instalación Cluster OCP UPI vSphere - VMWare

```
[root@home] wget https://raw.githubusercontent.com/leoaraujo/openshift_pxe_boot_menu/main/files/bg-ocp.png -O /var/lib/tftpboot/bg-ocp.png
```

- Copiar archivo *pxelinux.0* dentro del directorio */var/lib/tftpboot*.

```
[root@home] cp /usr/lib/PXELINUX/pxelinux.0 /var/lib/tftpboot/
```

- Mover o copiar los archivos "rhcos-4.13.10-x86\_64-live-kernel-x86\_64" y "rhcos-4.13.10-x86\_64-custom-initramfs.x86\_64.img" dentro del directorio *\*/var/lib/tftpboot\_*

```
[root@home] mv /home/jotape/ignition-files/rhcos-4.13.10-x86_64-live-kernel-x86_64 /var/lib/tftpboot/
```

```
[root@home] mv /home/jotape/ignition-files/rhcos-4.13.10-x86_64-custom-initramfs.x86_64.img /var/lib/tftpboot/
```

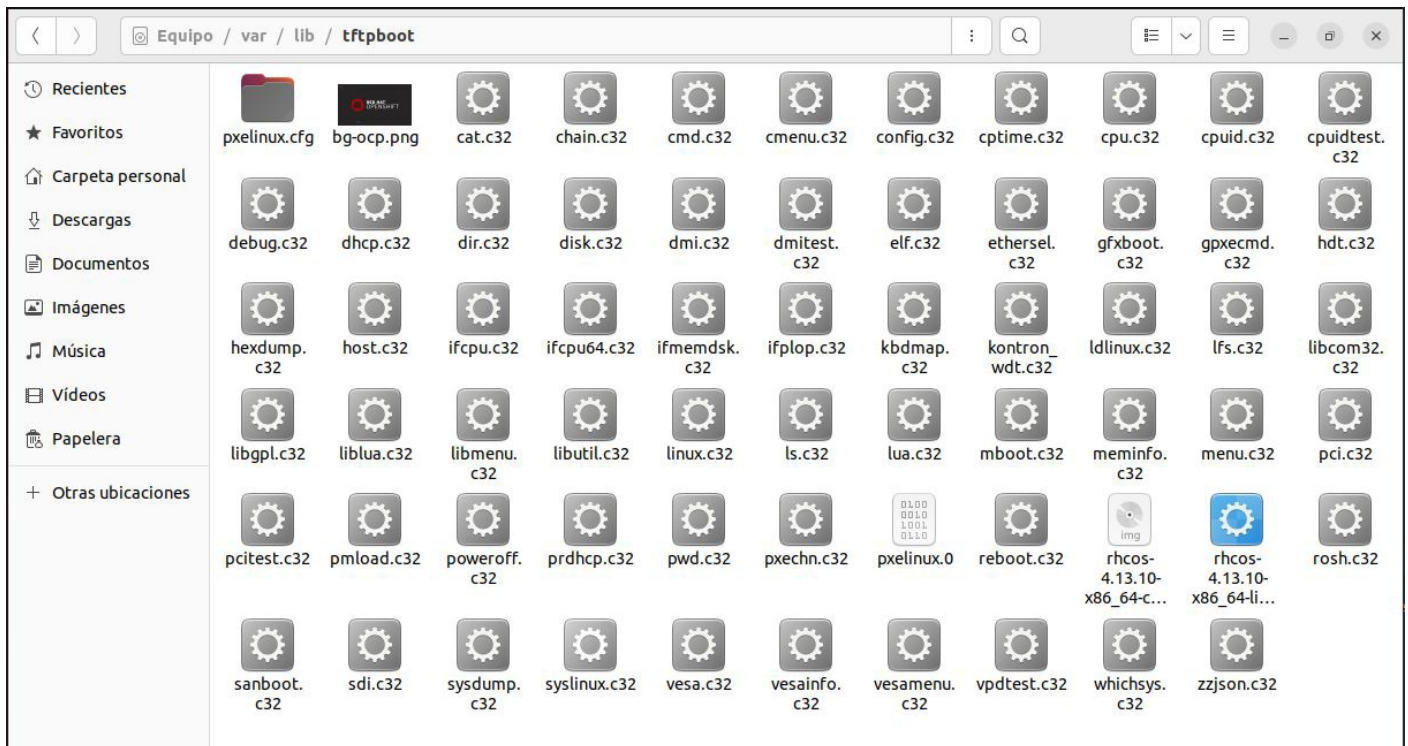
- Crear directorio *pxelinux.cfg* y dentro de él crear un archivo *default*.

```
[root@tftpboot] mkdir pxelinux.cfg | cd pxelinux.cfg
```

```
[root@pxelinux.cfg] touch default
```

- El directorio debería quedar de la siguiente manera:

```
var
|__lib
|__tftpboot
|__pxelinux.cfg
|    | default
|    | pxelinux.0
|    | bg-ocp.png
|    | rhcos-4.13.10-x86_64-live-kernel-x86_64
|    | rhcos-4.13.10-x86_64-custom-initramfs.x86_64.img
|    | syslinux files...
```



- Quitarle permisos al directorio tftpboot y contenido.

OBS: Para efecto de este ejercicio se le quitaron todos los permisos las rutas, no es la mejor opción pero aun no está claro qué permisos necesita el booteo PXE para obtener el archivo de arranque.



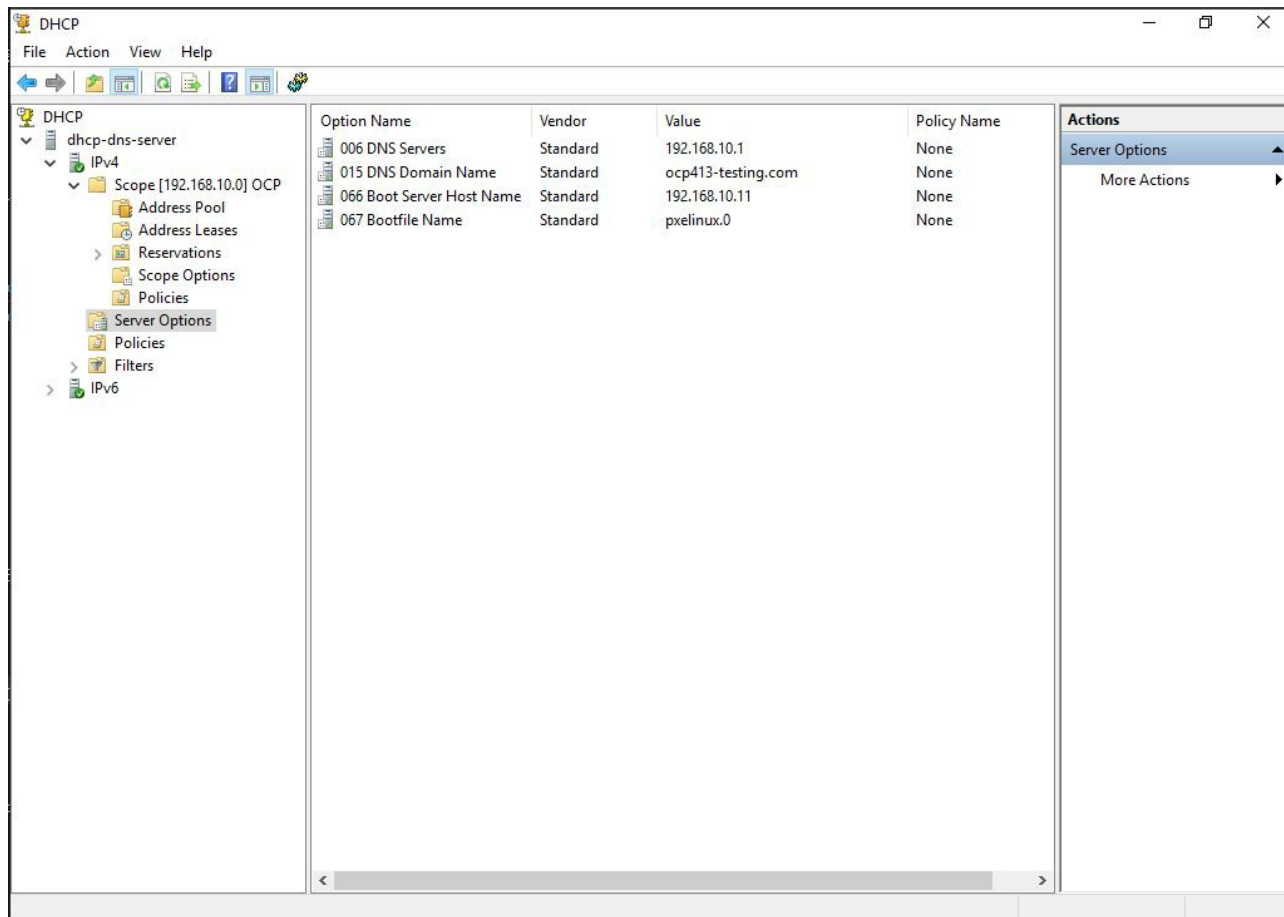
# Instalación Cluster OCP UPI vSphere - VMWare

```
[root@home] chown -R nobody:nogroup /var/lib/tftpboot  
[root@home] chmod -R 777 /var/lib/tftpboot
```

- Reiniciar servicio tftpd-hpa

```
[root@home] systemctl restart tftpd-hpa
```

- Configurar Servidor DHCP. Añadir server options 066 y 067 con la IP del tftp server (Ip de la máquina ubuntu en este caso) y el nombre del archivo de booteo, como se muestra en la imagen



## 6. Configurar servidor HTTP Apache en servidor Bastión.

- Ir al directorio de configuración de Apache.

**OBS:** Está configuración es para un sistema Ubuntu.

```
[root@home] cd /etc/apache2/sites-available
```

- Modificar el archivo de configuración default de Apache. OBS: Se puede crear un sitio nuevo creando un archivo de configuración nuevo.

# Instalación Cluster OCP UPI vSphere - VMWare

```
[root@sites-available] nano 000-default.conf

<VirtualHost 192.168.10.11:80>
    ServerAdmin root@bastion.ocp413-testing.com
    DocumentRoot /var/www/html
    ServerName bastion.ocp413-testing.com
    ErrorLog ${APACHE_LOG_DIR}/images-server-error.log
    CustomLog ${APACHE_LOG_DIR}/images-server-access.log common
    <Directory /pxeboot/boot >
        Options Indexes MultiViews
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

- Crear directorio `/pxeboot/boot/` dentro de la ruta `/var/www/html`.

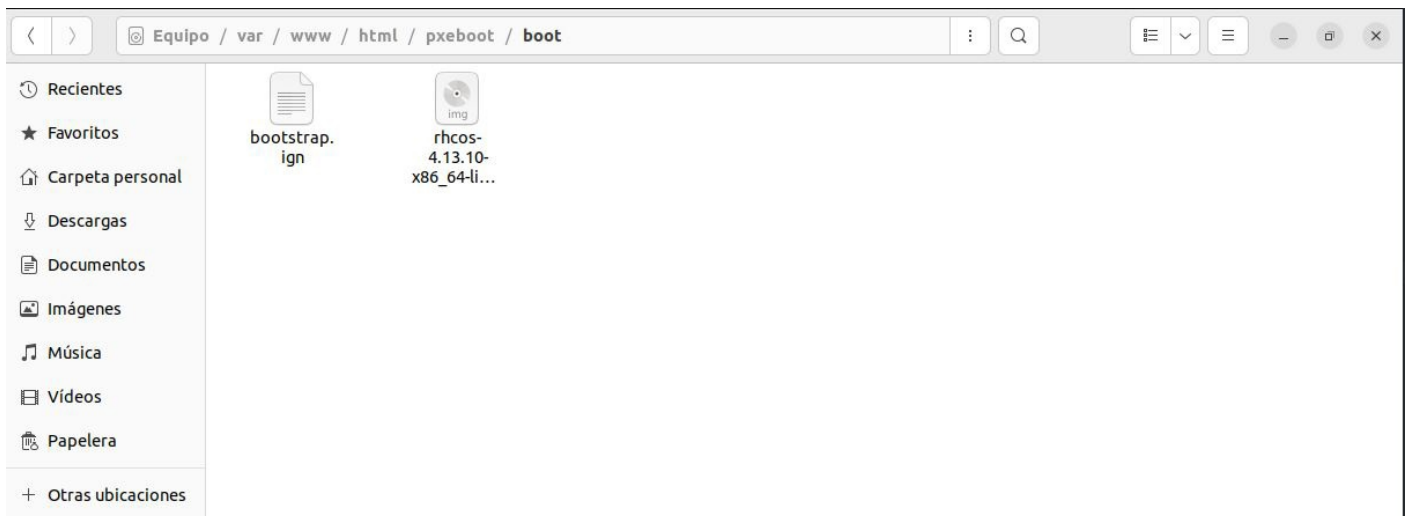
```
[root@html] mkdir pxeboot | cd /pxeboot
[root@pxeboot] mkdir boot | cd /boot
```

- Mover o copiar archivos "bootstrap.ign" y "rhcos-4.13.10-x86\_64-live-rootfs.x86\_64.img" dentro del directorio `_boot`.

El directorio debería quedar de la siguiente manera:

```
var
|___www
|   |___html
|       |___pxeboot
|           |___boot
|               | bootstrap.ign
|               | rhcos-4.13.10-x86_64-live-rootfs.x86_64.img
```

- OBS: Las razones por las cuales estos archivos no van en el directorio `tftpboot` son las siguientes:
  - El instalador del SO RCHOS solicita sí o sí que el archivo "rhcos-4.13.10-x86\_64-live-rootfs.x86\_64.img" sea transferido a través de HTTP,TFTP o HTTPS.
  - La solución más rápida es HTTP ya que la TFTP demoró alrededor de 2 horas en obtener el archivo. (El archivo pesa 1 GB aprox.).
  - Existen más opciones de seguridad al realizar la transferencia por HTTP o HTTPS que no fueron exploradas en éste ejercicio.
  - Al momento de iniciar el booteo del SO RCHOS, la máquina, por alguna razón que desconozco, bloquea las llamadas entrantes desde el servidor donde están alojadas las imágenes y solo se pueden obtener directamente desde el directorio `/tftpboot`. Los 2 archivos esenciales para que comience la instalación del SO RHCOS, son "rhcos-4.13.10-x86\_64-live-kernel-x86\_64" y "rhcos-4.13.10-x86\_64-custom-initramfs.x86\_64.img".



- Quitarle permisos al directorio `/var/www/html/pxeboot/boot` y contenido.

OBS: Para efecto de este ejercicio se le quitaron todos los permisos las rutas, no es la mejor opción pero aun no está claro qué permisos necesita el servidor HTTP para servir los archivos a la máquina booteable.

# Instalación Cluster OCP UPI vSphere - VMWare

```
[root@var] chown -R nobody:nogroup /var/www
```

```
[root@var] chmod -R 777 /var/www
```

- Configurar firewall para el servidor TFTP y HTTP (Apache).

**OBS:** Estos puertos son los predeterminados para los protocolos TFTP y HTTP. En teoría, si es requerido, se puede configurar cada uno para que sirva en un puerto distinto.

```
[root@home] ufw allow Apache
```

```
[root@home] ufw allow from any to any proto udp port 69
```

```
[root@home] ufw reload
```

```
[root@home] ufw status
```

```
root@bastion:~# ufw status
Estado: activo

Hasta      Acción    Desde
-----
Anywhere on enp0s8  DENY      192.168.10.0/24
22         ALLOW     Anywhere
8080        ALLOW     Anywhere
22/tcp     ALLOW     Anywhere
69/udp     ALLOW     Anywhere
Apache     ALLOW     Anywhere
22 (v6)    ALLOW     Anywhere (v6)
8080 (v6)  ALLOW     Anywhere (v6)
22/tcp (v6) ALLOW     Anywhere (v6)
69/udp (v6) ALLOW     Anywhere (v6)
Apache (v6) ALLOW     Anywhere (v6)
```

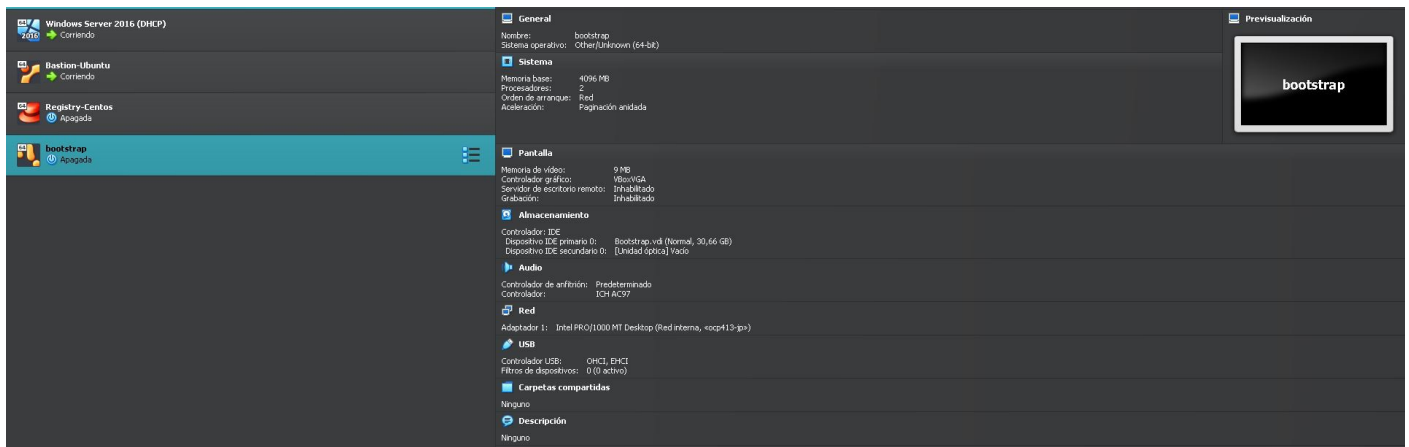
## 7. Configurar archivo default en el directorio `/var/lib/tftpboot/pxelinux.cfg`.

Por el momento, solo se tiene el `ignition` file del `bootstrap` configurado.

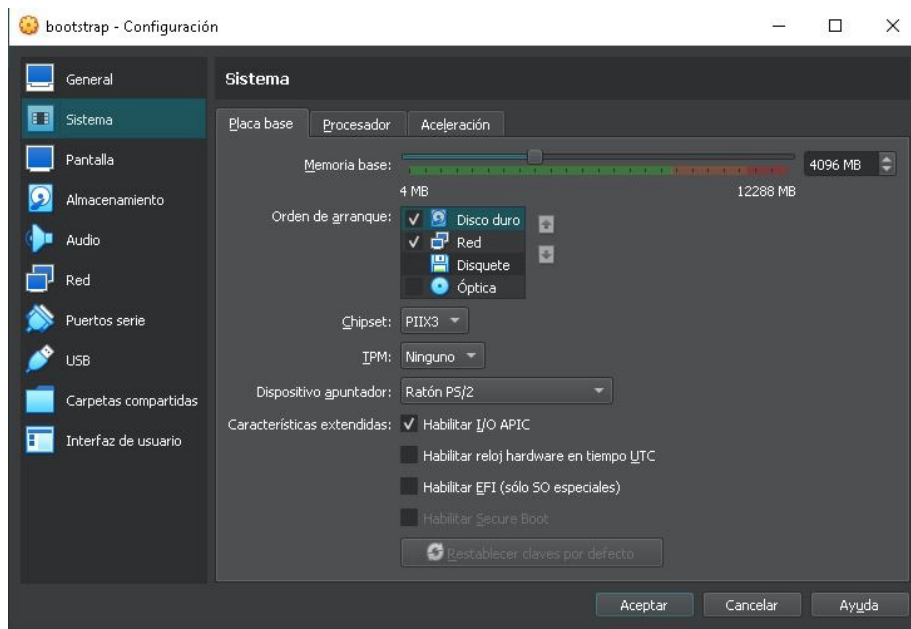
```
UI vesamenu.c32
MENU BACKGROUND      bg-ocp.png
MENU COLOR sel        4 #ffffff std
MENU COLOR title      1 #ffffff

MENU TITLE OPENSIFT 4.13.10 INSTALLATION PXE MENU
LABEL RedHat CoreOS
    MENU LABEL RedHat CoreOS
    KERNEL rhcos-4.13.10-x86_64-live-kernel-x86_64
    APPEND initrd=rhcos-4.13.10-x86_64-custom-initramfs.x86_64.img
coreos.live.rootfs_url=http://192.168.10.11/pxeboot/boot/rhcos-4.13.10-x86_64-live-rootfs.x86_64.img ignition.firstboot
ignition.platform.id=metal
```

## 8. Una vez configurado el archivo. Crear maquina virtual vacía. (En este ejercicio se utilizó VirtualBox).



## 9. Configurar bootstrap para bootear por red.



## 10. Iniciar la máquina virtual. Al bootear debería aparecer el siguiente menú.

```
iPXE (PCI E2:00.0) starting execution...ok
iPXE initialising devices...ok

iPXE 1.21.1 -- Open Source Network Boot Firmware -- http://ipxe.org
Features: DNS TFTP PXE PXEXT

net0: 08:00:27:1b:dc:48 using 82540em on 0000:00:03.0 (open)
  [Link:down, TX:0 TXE:0 RX:0 RXE:0]
  [Link status: Down (http://ipxe.org/38086101)]
Waiting for link-up on net0.... ok
Configuring (net0 08:00:27:1b:dc:48)..... ok
net0: 192.168.10.13/255.255.255.0
Next server: 192.168.10.11
Filename: pxelinux.0
tftp://192.168.10.11/pxelinux.0... ok
pxelinux.0 : 42584 bytes [PXE-NBP]

PXELINUX 6.04 PXE 20210811 Copyright (C) 1994-2015 H. Peter Anvin et al
-
```



## Instalación Cluster OCP UPI vSphere - VMWare

```
[ 9.577227] systemd[1]: Starting dracut initqueue hook...
[ OK ] Finished dracut initqueue hook.[ 9.801191] systemd[1]: Finished dracut initqueue hook.

[ OK ] Reached target Preparation for Remote File Systems.[ 9.809230] systemd[1]: Reached target Preparation for Remote File Systems.

[ OK ] Reached target Remote Encrypted Volumes.[ 9.816348] systemd[1]: Reached target Remote Encrypted Volumes.

[ OK ] Reached target Remote File Systems.[ 9.824168] systemd[1]: Reached target Remote File Systems.

Starting Acquire Live PXE rootfs Image...
[ 9.838206] systemd[1]: Starting Acquire Live PXE rootfs Image...
[ 9.842191] systemd[1]: Starting dracut pre-mount hook...
Starting dracut pre-mount hook...
[ 9.878230] coreos-livepxe-rootfs[707]: Fetching rootfs image from http://192.168.10.11/pxeboot/boot/rhcos-4.13.10-x86_64-live-rootfs.x86_64.img...
[ OK ] Finished dracut pre-mount hook.[ 9.897204] systemd[1]: Finished dracut pre-mount hook.

[ 9.991242] coreos-livepxe-rootfs[735]: bsdtar: Failed to set default locale
[***] A start job is running for Acquire Live PXE rootfs Image (9s / no limit)
```

```
Red Hat Enterprise Linux CoreOS 413.92.202307260246-0 (Plow) 4.13
SSH host key: SHA256:hHZQDfH9ZIr3sup4MmbJy0zzzUQ4ZahgddtE1BW7/xI (ED25519)
SSH host key: SHA256:U6ZKP2hF5TKYxFrSZwrN5o8cJ1IErn5USDDexbWrD8I (ECDSA)
SSH host key: SHA256:xSxSTFdQDXuktWoDkcI5muLp1ptwdBR8NmtLMo60zbg (RSA)
enp0s3: 192.168.10.13 fe80::d11d:1e83:2ead:ab33
Ignition: ran on 2024/02/28 12:01:33 UTC (this boot)
Ignition: user-provided config was applied
bootstrap login:
```

```
bootstrap login: core
Password:
Last login: Wed Feb 28 12:12:51 on pts/0
Red Hat Enterprise Linux CoreOS 413.92.202307260246-0
Part of OpenShift 4.13, RHCOS is a Kubernetes native operating system
managed by the Machine Config Operator ('clusteroperator/machine-config').

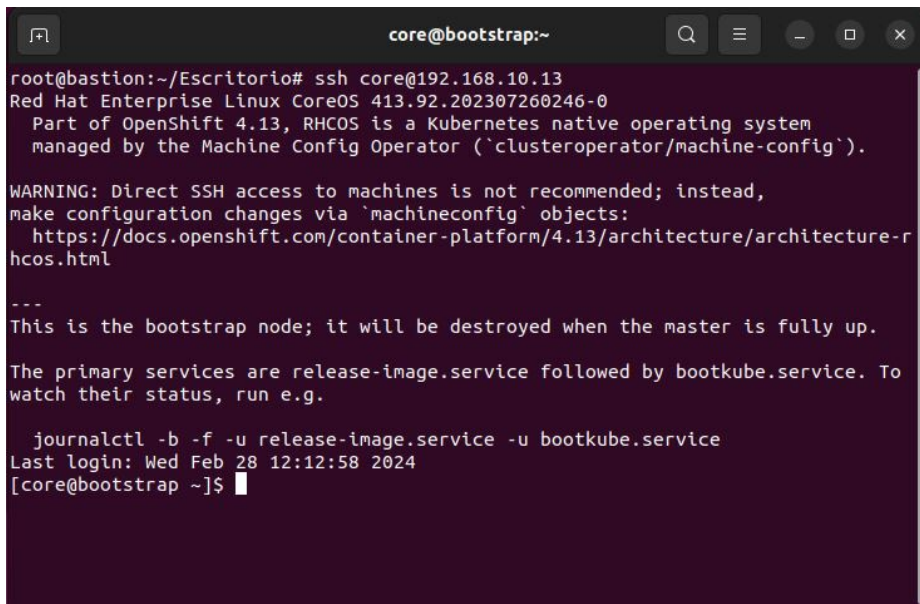
WARNING: Direct SSH access to machines is not recommended; instead,
make configuration changes via 'machineconfig' objects:
https://docs.openshift.com/container-platform/4.13/architecture/architecture-rhcos.html

---
This is the bootstrap node; it will be destroyed when the master is fully up.

The primary services are release-image.service followed by bootkube.service. To
watch their status, run e.g.

journalctl -b -f -u release-image.service -u bootkube.service
[core@bootstrap ~]$
[core@bootstrap ~]$
[core@bootstrap ~]$
[core@bootstrap ~]$
[core@bootstrap ~]$
[core@bootstrap ~]$
```

Conexión a nodo bootstrap por SSH desde máquina Bastión.

A terminal window titled 'core@bootstrap:~' with standard window controls. The terminal shows a successful SSH login from a bastion host to a bootstrap node. The output includes system information (Red Hat Enterprise Linux CoreOS), a warning about direct SSH access, and instructions for managing the node. The user 'core' is logged in and the prompt is '[core@bootstrap ~]\$'.

## NODO BOOTSTRAP OBS

- En este ejercicio, cuando se instaló el SO en el nodo bootstrap, los archivos de configuración de éste (ignition, initramfs, etc..) no proveían una contraseña para el usuario "core", por lo que se tuvo que ingresar desde la máquina Bastión por SSH, y asignarle una contraseña desde esa instancia.

## NODO BOOTSTRAP OBS 2

- En este ejercicio, solo se tenía las máquinas (DHCP,BASTION y BOOTSTRAP) levantadas, por lo que al bootear el nodo Bootstrap, éste intento hacer un pull de las imágenes al registry que se configuró previamente, por lo que en producción, cuando se instalen y configuren los nodos, todas las máquinas deben estar operativas.

## 4. HAProxy Load Balancer

- En este ejercicio se utilizó una máquina virtual CentOS 7 para el Load Balancer.

### 1. Levanta máquina con conexión a Internet para instalar HAProxy.

```
[root@home] yum install haproxy
```

### 2. Una vez instalado el paquete, se puede dar de baja la interfaz que da acceso a internet.

### 3. Configurar el archivo de configuración de HAProxy.

```
[root@home] nano /etc/haproxy/haproxy.cfg
```

### 4. Para éste ejercicio la configuración sería la siguiente:

```
global
    log          127.0.0.1 local2
    chroot       /var/lib/haproxy
    pidfile      /var/run/haproxy.pid
    maxconn      4000
    stats socket /run/haproxy/admin.sock mode 600 level admin
    daemon

defaults
    mode          http
    log           global
    option        httplog
    option        dontlognull
    option http-server-close
    option        redispatch
    retries       3
    timeout http-request 10s
    timeout queue 1m
    timeout connect 10s
    timeout client 1m
    timeout server 1m
    timeout http-keep-alive 10s
    timeout check 10s
    maxconn       3000

frontend ocp413-testing-k8s-api-fe
    bind :6443
    default_backend ocp413-testing-k8s-api-be
    mode tcp
    option tcplog

backend ocp413-testing-k8s-api-be
    bind *:6443
    mode tcp
    option httpchk GET /readyz HTTP/1.0
    option http-health-checks
    balance roundrobin
    server bootstrap bootstrap.ocp413-testing.com:6443 verify none check chk-ssl inter 10s fall rise 3 backup

frontend ocp413-testing-machine-config-server-fe
    bind :22623
    default_backend ocp413-testing-machine-config-server-be
    mode tcp
    option tcplog

backend ocp413-testing-machine-config-server-be
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp413-testing.com:22623 check inter 1s backup
```

- Según la documentación oficial, el archivo debería quedar de la siguiente forma:

```
global
    log      127.0.0.1 local2
    chroot   /var/lib/haproxy
    pidfile   /var/run/haproxy.pid
    maxconn   4000
    stats socket /run/haproxy/admin.sock mode 600 level admin
    daemon

defaults
    mode                http
    log                  global
    option               httplog
    option               dontlognull
    option http-server-close
    option               redispatch
    retries              3
    timeout http-request 10s
    timeout queue        1m
    timeout connect      10s
    timeout client       1m
    timeout server       1m
    timeout http-keep-alive 10s
    timeout check        10s
    maxconn              3000

frontend ocp413-testing-k8s-api-fe
    bind :6443
    default_backend ocp413-testing-k8s-api-be
    mode tcp
    option tcplog

backend ocp413-testing-k8s-api-be
    bind *:6443
    mode tcp
    option httpchk GET /readyz HTTP/1.0
    option log-health-checks
    balance roundrobin
    server bootstrap bootstrap.ocp413-testingcom:6443 verify none check check-ssl inter 10s fall 2 rise 3 backup
    server master0 master0.ocp413-testingcom:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
    server master1 master1.ocp413-testingcom:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3
    server master2 master2.ocp413-testingcom:6443 weight 1 verify none check check-ssl inter 10s fall 2 rise 3

frontend ocp413-testing-machine-config-server-fe
    bind :22623
    default_backend ocp413-testing-machine-config-server-be
    mode tcp
    option tcplog

backend ocp413-testing-machine-config-server-be
    bind *:22623
    mode tcp
    server bootstrap bootstrap.ocp413-testingcom:22623 check inter 1s backup
    server master0 master0.ocp413-testingcom:22623 check inter 1s
    server master1 master1.ocp413-testingcom:22623 check inter 1s
    server master2 master2.ocp413-testingcom:22623 check inter 1s

frontend ocp413-testing-http-ingress-traffic-fe
    bind :443
    default_backend ocp413-testing-ingress-router-https-be
    mode tcp
    option tcplog

backend ocp413-testing-ingress-router-https-be
    bind *:443
    mode tcp
    balance source
    server worker0 worker0.ocp413-testingcom:443 check inter 1s
    server worker1 worker1.ocp413-testingcom:443 check inter 1s

frontend ocp413-testing-http-ingress-traffic-fe
    bind :80
    default_backend ocp413-testing-ingress-router-http-be
    mode tcp
    option tcplog

backend ocp413-testing-ingress-router-http-be
    bind *:80
    mode tcp
    balance source
    server worker0 worker0.ocp413-testingcom:80 check inter 1s
    server worker1 worker1.ocp413-testingcom:80 check inter 1s
```



