Kubernetes + Kubevirt + Ansible

Corriendo sobre Fedora 31

Roberto J. Calva

LATAM Cloud Management & Automation Specialist

Red Hat LATAM

kubernetes

https://kubernetes.io

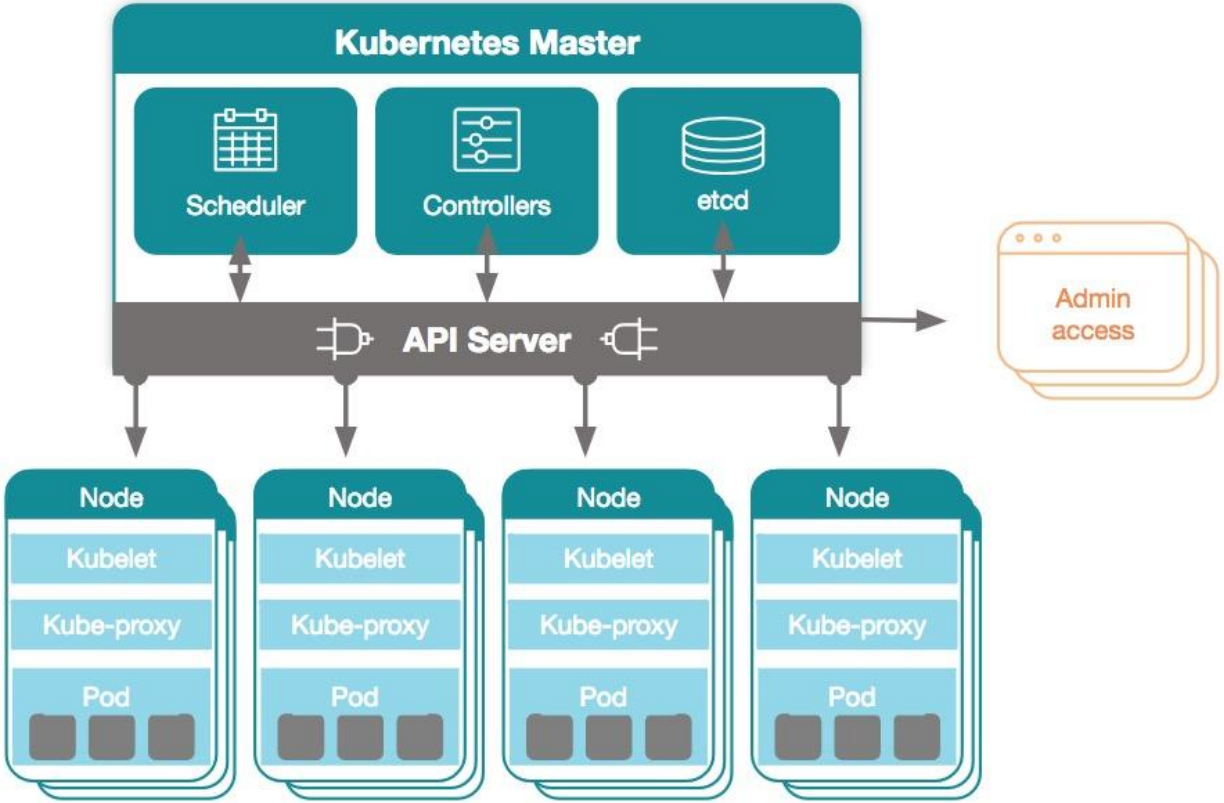
@kubernetesio

Qué es Kubernetes?

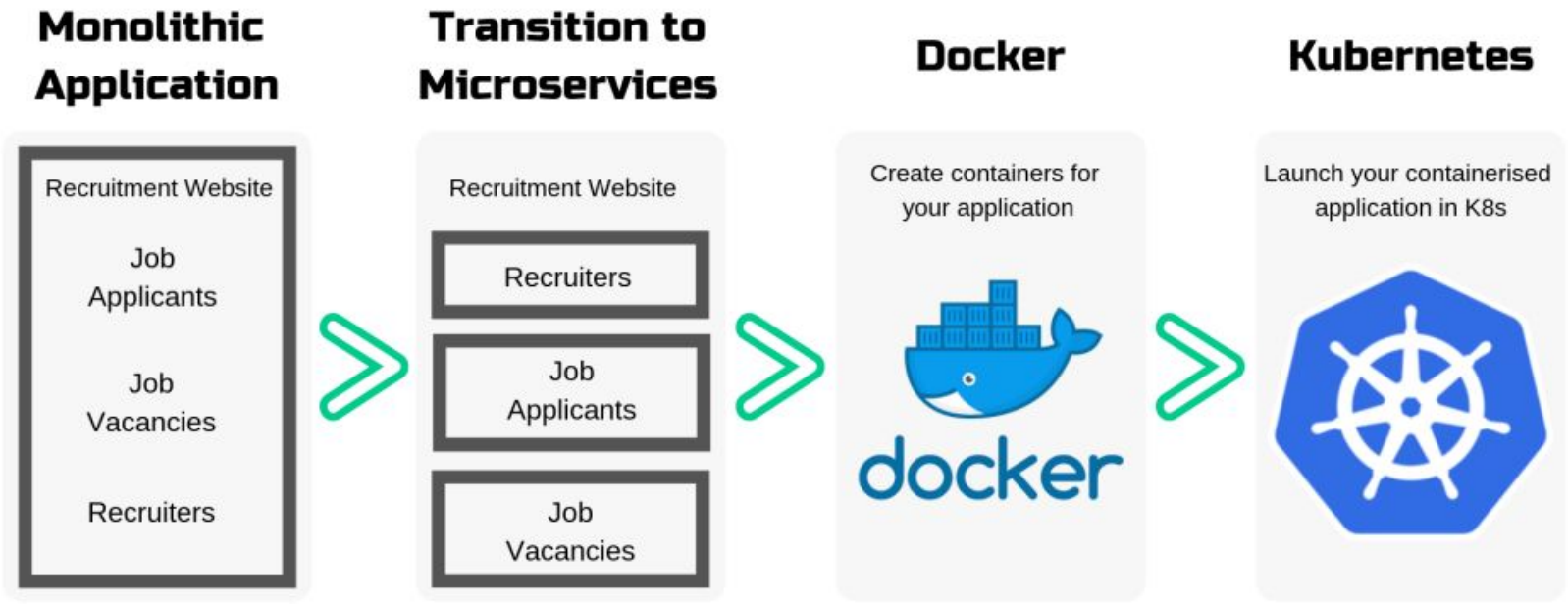
Kubernetes es un sistema de orquestación de contenedores de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones cloud-native. Originalmente fue diseñado por Google, y ahora es mantenido por la Cloud Native Computing Foundation.

Red Hat participa desde el día 1 en el desarrollo de Kubernetes.

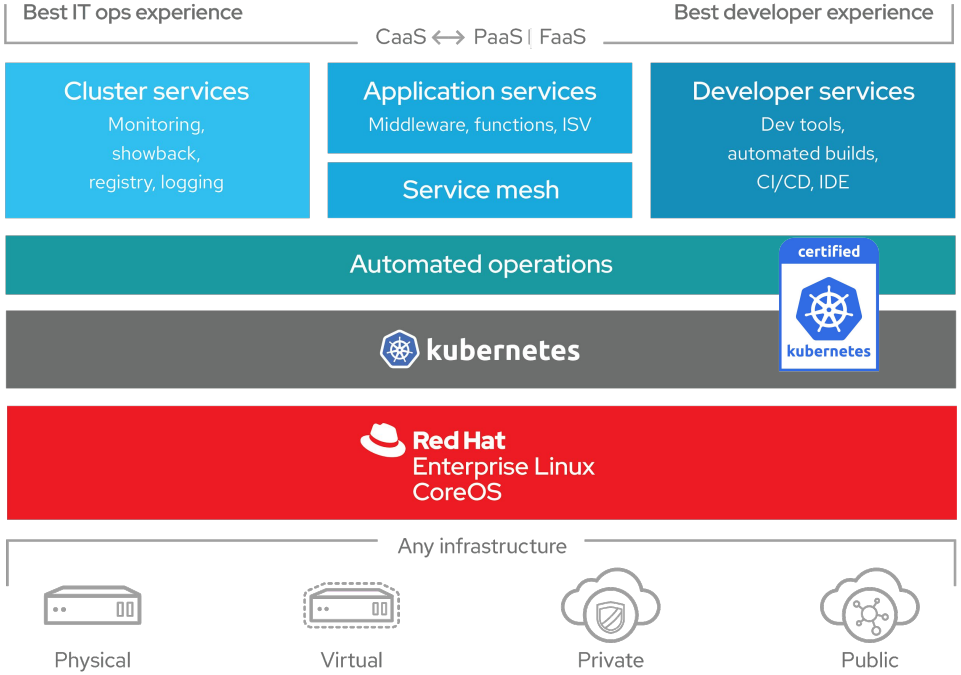
Kubernetes Architecture



De Aplicaciones Monolíticas a Nativas de la Nube



OpenShift 4 - Una plataforma Kubernetes más inteligente



Instalación automatizada de pila completa del host del contenedor para servicios de aplicaciones

Implementación perfecta de Kubernetes en cualquier nube o ambiente local

Escalabilidad automática de recursos de la nube

Actualizaciones con un click para plataforma, servicios y aplicativos

Red Hat

Comandos básicos oc y kubectl

- Obtener recursos - `***kubectl get pods***`
- Describir recursos - `***kubectl describe pod pod_name***`
- Get pod logs - `***kubectl logs pod_name***`
- Trabajar con recurso de un namespace específico `***kubectl -n namespace get pods***`
- Obtener recursos de todos los namespaces - `***kubectl get pods --all-namespaces***`
- Comando en un container - `***kubectl exec pod_name -c container_name -it -- command***`

KubeVirt

https://kubevirt.io

@kubevirt

Qué es KubeVirt?

KubeVirt es un complemento de administración de máquinas virtuales para Kubernetes. El objetivo es proporcionar un terreno común para soluciones de virtualización sobre Kubernetes y que tanto las VMs como los Containers convivan en una misma plataforma.

Qué pasa cuando inicias una VM?

- virt-controller vigila los recursos de la VM, si aparece una nueva VM y el sistema no tiene un pod relevante, virt-controller creará un pod virt-launcher y le dará la propiedad del pod al manejador virt.
- Por cada VM debe existir solo un pod virt-launcher.
- El pod virt-launcher ejecuta un contenedor que incluye el paquete libvirt (puede ejecutar comandos como virsh) y ejecuta el proceso QEMU de la VM relevante.
- virt-launcher pod habla con virt-handler pod a través de la conexión RPC servidor-cliente, de esta manera virt-handler conoce el estado actual de una VM y sabe cómo actualizar una VM al estado deseado.

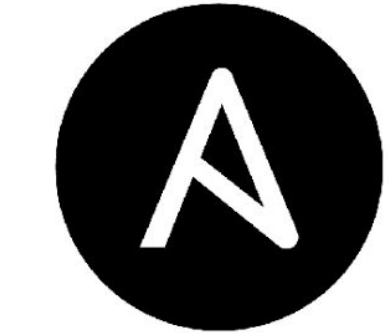
Virtctl (kubectl virt*)

Herramienta que te da la posibilidad de:

- Conectarse a una VM via VNC
- Conectarse a una VM via console
- Start Virtual Machine
- Stop Virtual Machine

* Instalar virt dentro de kubectl usando krew:

https://github.com/kubernetes-sigs/krew/



https://www.ansible.com

@ansible

Qué es Ansible?

Automatización de TI simple y sin agentes que cualquiera puede usar.

Ansible es un lenguaje universal que nos permite automatizar casi cualquier cosa (sistemas operativos, nubes, virtualización, dispositivos de red, dispositivos de seguridad, etc.) sin la necesidad de tener un perfil de desarrollador.

ANSIBLE AUTOMATIZA TECNOLOGÍAS QUE YA USAS

El tiempo para automatizar se mide en minutos

CLOUD

AWS
Azure
CenturyLink
Digital Ocean
Google
OpenStack
Rackspace
+more

VIRT & CONTAINER

Docker
VMware
RHV
OpenStack
OpenShift
+more

STORAGE

NetApp
Red Hat Storage
Infinidat
+more

WINDOWS

ACLs
Files
Packages
IIS
Regedits
Shares
Services
Configs
Users
Domains
+more

NETWORK

Arista
A10
Cumulus
Bigswitch
Cisco
Cumulus
Dell
F5
Juniper
Palo Alto
OpenSwitch
+more

DEVOPS

Jira
GitHub
Vagrant
Jenkins
Bamboo
Atlassian
Subversion
Slack
Hipchat
+more

MONITORING

Dynatrace
Airbrake
BigPanda
Datadog
LogicMonitor
Nagios
New Relic
PagerDuty
Sensu
StackDriver
Zabbix
+more

Instalación de Kubernetes en Fedora 31

Prerrequisitos

- 1 o más máquinas con Fedora 31
- Mínimo 2 CPUs (recomendable 4 CPUs)
- Mínimo 2 GB de memoria RAM (recomendable 8 GB RAM)
- Al menos 40 GB en disco

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/

Prerrequisitos

- SELinux en Permisivo (al principio. Después habría que aplicar booleanos sobre archivos de Kubernetes y Etcd)
- Deshabilitar memoria Swap (swapoff -a)
- Firewall deshabilitado (al final es posible aplicar reglas de Firewall específicas)

Prerrequisitos

- Docker y Fedora 31 tienen un issue utilizando CGroups v2

```
[root@fedora-t420s ~]# dnf -y install grubby
[root@fedora-t420s ~]# grubby --update-kernel=ALL --args="systemd.unified_cgroup_hierarchy=0"
[root@fedora-t420s ~]# dnf -y install moby-engine
[root@fedora-t420s ~]# reboot
[root@fedora-t420s ~]$ sudo systemctl enable --now docker
[root@fedora-t420s ~]$ sudo usermod -aG docker $(whoami)
[root@fedora-t420s ~]$ docker ps
```

Prerrequisitos

- Instalar los siguientes paquetes
    - Kubernetes
    - Kubernetes-client
    - kubernetes-node
    - kubernetes-master
    - Kubernetes-kubeadm
    - openvswitch-ovn-kubernetes
    - Flannel
    - podman

Prerrequisitos

- Modificar Kubelet Service (Fedora 31 & Kubernetes 1.15 issue)

root@fedora-t420s ~]# vim /etc/systemd/system/kubelet.service.d/kubeadm.conf and delete **--allow-privileged=true** and **--fail-swap-on=true** flags.

[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubelet.conf --kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_SYSTEM_PODS_ARGS=--pod-manifest-path=/etc/kubernetes/manifests"
Environment="KUBELET_NETWORK_ARGS=--network-plugin=cni --cni-conf-dir=/etc/cni/net.d --cni-bin-dir=/usr/libexec/cni"
Environment="KUBELET_DNS_ARGS=--cluster-dns=10.96.0.10 --cluster-domain=cluster.local"
Environment="KUBELET_AUTHZ_ARGS=--authorization-mode=Webhook --client-ca-file=/etc/kubernetes/pki/ca.crt"
Environment="KUBELET_EXTRA_ARGS=--cgroup-driver=systemd"
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_SYSTEM_PODS_ARGS $KUBELET_NETWORK_ARGS $KUBELET_DNS_ARGS $KUBELET_AUTHZ_ARGS $KUBELET_EXTRA_ARGS
Restart=always
StartLimitInterval=0
RestartSec=10

Prerrequisitos

- Iniciar Servicio Kubelet

```
[root@fedora-t420s ~]# systemctl enable --now kubelet
```

Instalación

- Comenzar con la instalación utilizando Kubeadm

[root@fedora-t420s ~]# kubeadm init --kubernetes-version=stable --pod-network-cidr=**10.244.0.0/16** *

*** Considerando utilizar Flannel como overlay network**

Instalación

Your **Kubernetes control-plane** 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/

Then you can **join any number of worker nodes** by running the following on each **as root**:

```
kubeadm join 192.168.1.73:6443 --token ayk1f2.3fb9ebnie1rx01td \
    --discovery-token-ca-cert-hash sha256:3a2b1f76c9fe220b5d61e0ec09e1790a1c4fd23fef0e32d65592e8c913c961f3
```

Configuración de Red Kubernetes

- Configurar proveedor de red Kubernetes (Flannel)

[azulman@fedora-t420s ~]$ kubectl get all --all-namespaces
[azulman@fedora-t420s ~]# kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
[azulman@fedora-t420s ~]$ kubectl get all --all-namespaces
[azulman@fedora-t420s ~]# kubectl get nodes

Post-instalación

- Habilitar nuestro Master node como Worker node (si estamos trabajando all-in-one)

```
[root@fedora-t420s ~]# kubectl get nodes -o json | jq .items[].spec.taints
[
  {
    "effect": "NoSchedule",
    "key": "node-role.kubernetes.io/master"
  }
]

[root@fedora-t420s ~]# kubectl taint nodes --all node-role.kubernetes.io/master-
node/fedora-t420s untainted

[root@fedora-t420s ~]# kubectl get nodes -o json | jq .items[].spec.taints
null
```

Post-instalación

- Aplicar reglas de Firewall (Master)

```
firewall-cmd --permanent --add-port=6443/tcp
firewall-cmd --permanent --add-port=2379-2380/tcp
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10251/tcp
firewall-cmd --permanent --add-port=10252/tcp
firewall-cmd --permanent --add-port=10255/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --add-masquerade --permanent

# only if you want NodePorts exposed on control plane IP as well
firewall-cmd --permanent --add-port=30000-32767/tcp

systemctl restart firewalld
```

Post-instalación

- Aplicar reglas de Firewall (Nodes)

```
firewall-cmd --permanent --add-port=10250/tcp
firewall-cmd --permanent --add-port=10255/tcp
firewall-cmd --permanent --add-port=8472/udp
firewall-cmd --permanent --add-port=30000-32767/tcp
firewall-cmd --add-masquerade --permanent

systemctl restart firewalld
```

Post-instalación

- Aplicar SELinux contexts:

```
# for kubernetes files
chcon -R -t svirt_sandbox_file_t /etc/kubernetes/

# for etcd files
chcon -R -t svirt_sandbox_file_t /var/lib/etcd
```

Creando nuestro primer Pod

```
# cat pod.json
{
    "apiVersion": "v1",
    "kind": "Pod",
    "metadata": {
        "labels": {
            "app": "apache-centos7"
        },
        "name": "apache-centos7"
    },
    "spec": {
        "containers": [
            {
                "image": "centos/httpd",
                "name": "apache-centos7",
                "ports": [
                    {
                        "containerPort": 80,
                        "hostPort": 80,
                        "protocol": "TCP"
                    }
                ]
            }
        ]
    }
}
```

Creando nuestro primer Pod

```
# kubectl create -f ./pod.json
pod "apache-centos7" created

# kubectl get pod

# kubectl describe pod apache-centos7 | grep 'IP:'
IP:             172.17.0.1

# curl 172.17.0.1
```

Instalación de Kubevirt sobre Kubernetes en Fedora 31

Prerrequisitos

- Obtener la última versión de Kubevirt

```
$ curl -s https://api.github.com/repos/kubevirt/kubevirt/releases/latest | jq -r .tag_name
v0.23.3

$ export KUBEVIRT_VERSION=v0.23.3
```

Prerrequisitos

- Desplegar Kubevirt Operator

```
$ kubectl create -f
https://github.com/kubevirt/kubevirt/releases/download/${KUBEVIRT_VERSION}/kubevirt-operator.yaml
namespace/kubevirt created
...
  deployment.apps/virt-operator created
```

- Esperar a que el Operator Pod esté disponible:

```
$ kubectl wait --for condition=ready pod -l kubevirt.io=virt-operator -n kubevirt --timeout=100s
  pod/virt-operator-5ddb4674b9-6fbrv condition met
```

Prerrequisitos

- Si se está trabajando en máquinas virtuales, habilitar VM Emulation:

```
$ kubectl create configmap kubevirt-config -n kubevirt --from-literal debug.useEmulation=true
configmap/kubevirt-config created
```

Instalación

- Desplegar Kubevirt vía Operator (Custom Resource):

```
$ kubectl create -f
https://github.com/kubevirt/kubevirt/releases/download/${KUBEVIRT_VERSION}/kubevirt-cr.yaml
  kubevirt.kubevirt.io/kubevirt created
```

Instalación

- Dar un vistazo al despliegue

```
$ kubectl get pods -n kubevirt
```

```
NAME                              READY   STATUS    RESTARTS   AGE
virt-api-7fc57db6dd-g4s4w         1/1     Running   0          3m
virt-api-7fc57db6dd-zd95q         1/1     Running   0          3m
virt-controller-6849d45bcc-88zd4  1/1     Running   0          3m
virt-controller-6849d45bcc-cmfzk  1/1     Running   0          3m
virt-handler-fvsqw                1/1     Running   0          3m
virt-operator-5649f67475-gmphg    1/1     Running   0          4m
virt-operator-5649f67475-sw78k    1/1     Running   0          4m
```

Post-Instalación

- Esperar a que todos los pods estén disponibles:

```
$ kubectl wait --for condition=ready pod -l kubevirt.io=virt-api -n kubevirt --timeout=100s
pod/virt-api-5ddb4674b9-6fbrv condition met

$ kubectl wait --for condition=ready pod -l kubevirt.io=virt-controller -n kubevirt --timeout=100s
pod/virt-controller-p3d4o-1fvfz condition met

$ kubectl wait --for condition=ready pod -l kubevirt.io=virt-handler -n kubevirt --timeout=100s
  pod/virt-handler-1b4n3z4674b9-sf1rl condition met
```

Post-Instalación

- Ahora sí es posible crear una VM:

```
$ kubectl apply -f https://raw.githubusercontent.com/kubevirt/demo/master/manifests/vm.yaml
virtualmachine.kubevirt.io/testvm created
```

- Para obtener las VMs:

```
$ kubectl get vms
NAME     AGE RUNNING VOLUME
 testvm  22s false
```

- Para describir una VM:

```
$ kubectl describe vm vm_name
```

https://kubevirt.io/labs/kubernetes/lab1

Post-Instalación

- Inicializar la VM:

```
$ kubectl virt start testvm
VM testvm was scheduled to start
```

- Para revisar el estatus de las VMs:

```
$ kubectl get vmis
NAME      AGE       PHASE     IP            NODENAME
testvm    1m        Running   10.32.0.11    master
```

Post-Instalación

- Para eliminar la VM:

```
$ kubectl virt stop testvm
VM testvm was scheduled to stop
```

```
$ kubectl delete vms testvm
virtualmachine.kubevirt.io "testvm" deleted
```

Siguientes Pasos

Experiment with the Containerized Data Importer (CDI)

https://kubevirt.io/labs/kubernetes/lab2.html

Ansible y Kubernetes

k8s – Manage Kubernetes (K8s) objects

- Use el cliente OpenShift Python para realizar operaciones CRUD en objetos K8.
- Pase la definición del objeto desde un archivo fuente o en línea. Vea ejemplos para leer archivos y usar plantillas Jinja o archivos cifrados en vault.
- Acceso a la gama completa de API de K8.
- Utilice el módulo k8s_info para obtener una lista de elementos sobre un objeto de tipo tipo
- Autentíquese usando un archivo de configuración, certificados, contraseña o token.
- Admite el modo de verificación.

https://docs.ansible.com/ansible/latest/modules/k8s_module.html

k8s – Manage Kubernetes (K8s) objects

```
- name: Create a k8s namespace
  k8s:
    name: testing
    api_version: v1
    kind: Namespace
     state: present
```

https://docs.ansible.com/ansible/latest/modules/k8s_module.html

PREGUNTAS?

linkedin.com/in/roberto-j-calva

Instagram: @robert.j.calva

Gracias!

linkedin.com/in/roberto-j-calva

Instagram: @robert.j.calva