

Capacitación Kubernetes

Servicio de capacitación y entrenamiento ARIU

Clase 2

Inicio de Cluster

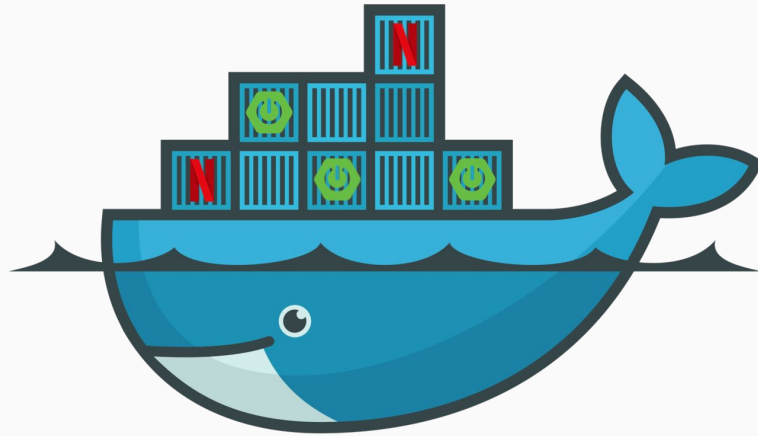
Agenda

- **Repaso última clase**
- Requisitos de inicialización de cluster
- Iniciar Kubernetes
- Instalar Herramientas
- Instalar Addons
- Agregar Control Plane y Worker

- **Inmutabilidad de las imágenes:** Las imágenes de los contenedores son inmutables, los cambios realizados se pierden al reiniciar.
- **Gestión de versiones:** Mantener el control de las versiones de las imágenes de los contenedores para evitar conflictos y problemas de compatibilidad.
- **Configuración y variables de entorno:** Definir y gestionar adecuadamente la configuración y variables de entorno de las aplicaciones en los contenedores.
- **Copias de seguridad y recuperación:** Establecer políticas y procedimientos para respaldar y recuperar datos y configuraciones de los contenedores.

- **Estructura del Dockerfile:**
 - Organiza el Dockerfile de manera clara y legible.
 - Utiliza las instrucciones adecuadas y en el orden correcto.
- **Minimizar el tamaño de la imagen:**
 - Optimiza la imagen para que sea lo más pequeña posible.
 - Elimina archivos innecesarios y utiliza capas eficientes.
- **Seguridad y buenas prácticas:**
 - Considera las mejores prácticas de seguridad al construir la imagen.
 - Evita el uso de contraseñas en texto plano y utiliza usuarios no privilegiados.

- **Pruebas y validaciones:**
 - Realiza pruebas y validaciones de la imagen construida.
 - Asegúrate de que funcione correctamente y cumpla con los requisitos deseados.
- **Escalabilidad:**
 - Diseña y configura tus aplicaciones para aprovechar la escalabilidad horizontal de Kubernetes.
 - Aprovecha la capacidad de escalar automáticamente los pods según la demanda de la carga de trabajo.



Ejecutar Script

- `chmod +x instalar-k8s-debian12-ubuntu2404.sh`
- `sh instalar-k8s-debian12-ubuntu2404.sh`

Reiniciar

- `reboot`

- **Actualización de paquetes:** Se actualizan los paquetes del sistema operativo para garantizar que se tengan las últimas versiones y correcciones de seguridad.
- **Desactivación del uso de Swap:** Ya que no es/era compatible con una instalación de Kubernetes.
- **Configuración del módulo `br_netfilter`:** Se verifica y configura el módulo `br_netfilter` necesario para permitir la comunicación entre los pods de Kubernetes en los nodos del clúster.
- **Activar `Ip_forward`:** Permite forward de paquetes
- **Instalación de `Containerd`:** Demonio que ejecuta contenedores, compatible con Kubernetes.

- **Instalación de Runc:** Se descarga e instala Runc, un ejecutor de contenedores de bajo nivel necesario para Containerd.
- **Instalación de CNI Plugins:** (Container Networking Interface) permiten la configuración de la red de los contenedores.
- **Instalación de Kubeadm, Kubelet y Kubectl:** Se instalan las herramientas de línea de comandos necesarias para administrar y operar el clúster de Kubernetes.
- **Instalación de Git:** Se instala Git, una herramienta de control de versiones, que será utilizada más adelante.



Agenda

- Repaso última clase
- **Requisitos de inicialización de cluster**
- Iniciar Kubernetes
- Instalar Herramientas
- Instalar Addons
- Agregar Control Plane y Worker

Recursos necesarios para Cluster en Producción

Nombre de cluster y dirección IP:

- Identificación única para el entorno de Kubernetes.
- Conectado a la API de comunicación de todos los Control Plane.
- Ip flotante en entre los Control Plane

Red de Pods:

- La red de Pods es el rango de direcciones IP asignado a los Pods en el cluster. Cada Pod recibirá una dirección IP dentro de este rango.
- La red elegida no debe ser ruteable en la organización.

Red de Servicios:

- La red de Servicios es el rango de direcciones IP utilizado para exponer los servicios dentro del cluster. Cada servicio tendrá una dirección IP dentro de este rango.
- La red elegida no debe ser ruteable en la organización.

Red para LoadBalancer:

- Esta es la gama de direcciones IP reservadas para los balanceadores de carga en el cluster. Los balanceadores de carga recibirán una dirección IP dentro de este rango para permitir el acceso externo a los servicios.
- Red Ruteable en la organización

Ingress para aplicaciones

Direcciones IP privada y pública:

- Estas son las direcciones IP asignadas a los controladores de ingreso (Ingress Controllers).
- Los controladores de ingreso gestionan el enrutamiento del tráfico hacia los servicios tanto desde dentro o fuera de la organización

Configuración DNS con *:

- Permite redirigir el tráfico automáticamente hacia múltiples servicios desplegados en el cluster sin necesidad de configurar cada uno por separado.
- Ofrece flexibilidad y simplifica la gestión de subdominios, facilitando la escalabilidad y la adición de nuevos servicios al cluster de manera ágil.

Ejemplo de recursos asignados cluster UNICEN

- **Nombre de cluster:** k8s.unicen.edu.ar **en ip** 10.254.11.4
- **Red de Pods:** 172.21.0.0/16
- **Red de Servicios:** 10.96.0.0/16
- **Red para LoadBalancer:** 10.254.11.5-10.254.11.254
- **IP de ingress interno** 10.254.11.5, 102.54.11.6, 10.254.11.7
- **IP de ingress externo** 131.221.0.46
- **Dns ingress:** *.unicen.edu.ar

Puertos para Administrar y Acceder a Aplicaciones de Kubernetes

Puertos de Control de Kubernetes:

- TCP 6443: API Server

Puertos para Acceso Externo a las Aplicaciones:

- Según los servicios y aplicaciones específicas desplegadas en el cluster, pueden ser necesarios puertos adicionales para el acceso externo, como HTTP (80) o HTTPS (443).

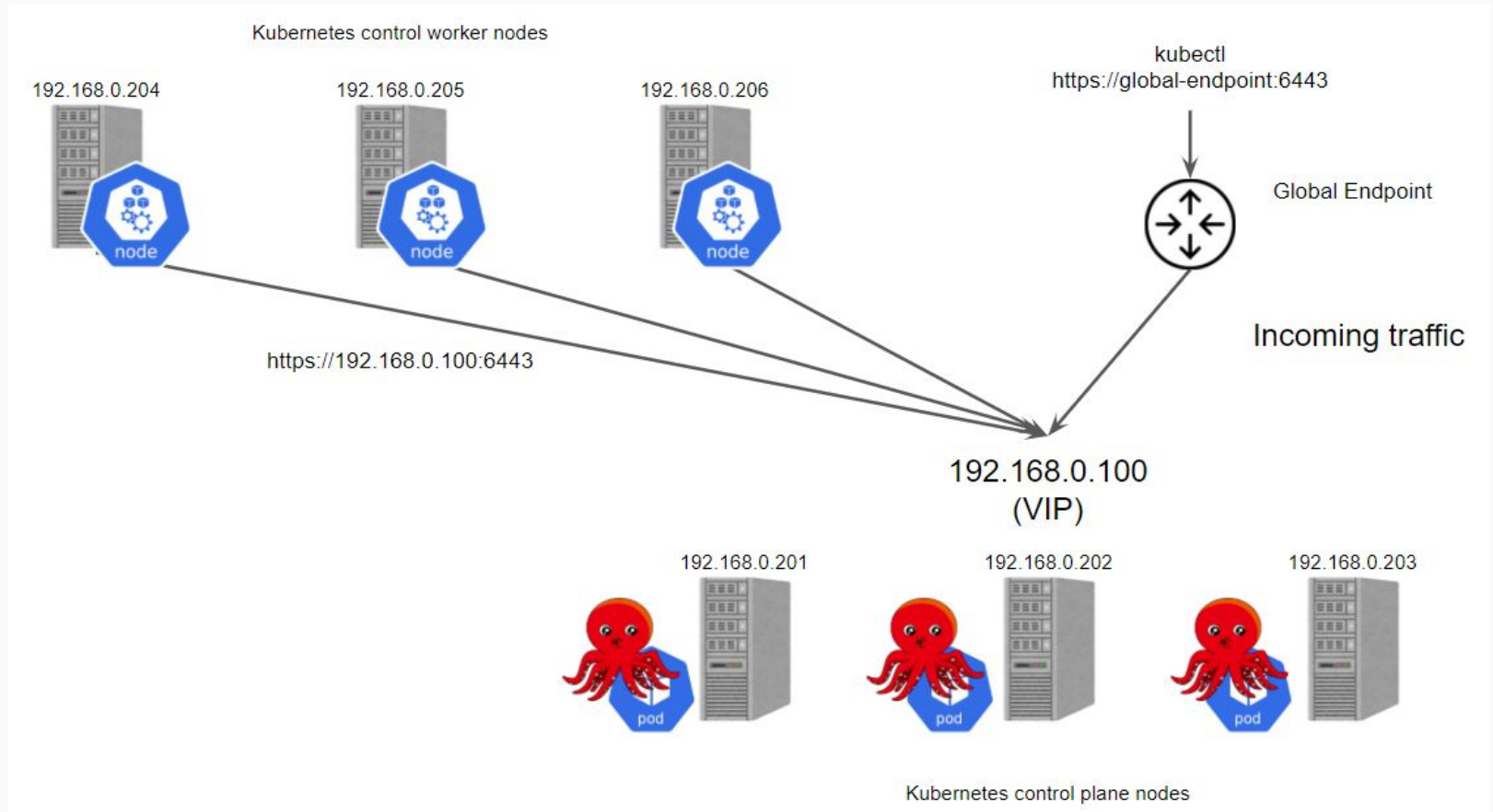
Agenda

- Repaso última clase
- Requisitos de inicialización de cluster
- **Iniciar Kubernetes**
- Instalar Herramientas
- Instalar Addons
- Agregar Control Plane y Worker

Para poder dar alta disponibilidad es necesario un **Load Balancer** y una **ip flotante** sobre el Control Plane activo.

- Utilizar kube-vip para manejar una IP virtual, proporcionando alta disponibilidad y balanceo de carga.
- Kube-vip actúa como Haproxy y Keepalived combinados.
- Si el nodo con la IP virtual falla, kube-vip la mueve a otro nodo disponible.

- Colocar el YAML del Pod en /etc/kubernetes/manifests para que se inicie en el control plane al inicio.
- Editar la placa de red en el archivo kube-vip.yaml para especificar dónde mover la IP virtual, si es necesario.
- Las peticiones entrantes se distribuyen entre los Pods de la API disponibles.



Práctica en conjunto: Ejecución de Comandos para iniciar Kubernetes

- Para seguir las actividades, pueden abrir el archivo “README.md” ubicado en Moodle, Clase 2.
- Dentro de dicho archivo, encontrarán los comandos que utilizaremos.
- Les solicitamos abrir el archivo y copiar los comandos, adaptándolos a sus respectivos entornos.

Kube-vip

Modificar nombre de placa de red e IP en kube-vip.yaml

- `sudo cp kube-vip.yaml /etc/kubernetes/manifests/`
- `sudo chmod 600 /etc/kubernetes/manifests/kube-vip.yaml`

Iniciar Kubernetes

Paso previo:

```
echo "170.210.5.8 k8s.riu.edu.ar" | sudo tee -a /etc/hosts
```

Opción 1:

```
sudo kubeadm init  
--control-plane-endpoint=ep-k8s-manager.siu.edu.ar
```

Opción 2:

```
sudo kubeadm init --pod-network-cidr=172.21.0.0/16  
--service-cidr=10.96.0.0/16 --service-dns-domain=cluster.local  
--control-plane-endpoint=k8s.riu.edu.ar
```

Verificar contenedores:

```
crictl ps -a
```

Conectar el cliente kubectl con el cluster

Ejecutar:

```
chmod +x conectar-cliente.sh  
conectar-cliente.sh
```

Contenido:

```
mkdir -p $HOME/.kube  
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Verificar

```
kubectl get nodes (va a fallar el nodo)
```

Conectar el cliente kubectl con el cluster en Pc Local

Instalar cliente

```
apt update -y  
apt upgrade -y  
apt install kubectl=1.33.4-00 -y
```

Copiar credenciales

```
mkdir -p $HOME/.kube  
scp root@<k8s-server>:/etc/kubernetes/admin.conf $HOME/.kube/config  
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Clusters:

- Certificado: Datos del certificado de autoridad que se utiliza para validar la identidad del servidor.
- Server: URL del servidor que apunta al API Server del clúster
- Nombre: Definición que usa el usuario para conectarse

Contexts:

- Los contextos permiten configurar diferentes entornos o escenarios de trabajo dentro del clúster. Un contexto se define como una **combinación de cluster, namespace y usuario**.

Users:

- Proporciona detalles sobre cada usuario y cómo autenticarse con el clúster usando certificado de cliente y clave privada

Agenda

- Repaso última clase
- Requisitos de inicialización de cluster
- Iniciar Kubernetes
- **Instalar Herramientas**
- Instalar Addons
- Agregar Control Plane y Worker

Instalar Helm

Herramienta de línea de comandos para Kubernetes que facilita la instalación y gestión de aplicaciones mediante paquetes preconfigurados (charts).

Url:

<https://helm.sh/>

Ejecutar:

instalar-helm.sh

Instalar krew y plugins de Kubectl

Administrador de plugins de kubectl

Url:

<https://krew.sigs.k8s.io/>

Ejecutar:

```
instalar-krew.sh
```

```
Agregar export a .bashrc
```

```
kubectl krew install ns
```

```
kubectl krew install get-all
```

```
kubectl krew install stern
```

```
kubectl krew install tree
```

Crear Alias de kubectl como k

Útil cuando trabajas con Kubernetes frecuentemente y necesitas ingresar comandos repetitivos

```
alias k='kubectl'
```

Agregar esta última línea a .bashrc

Cargar bashrc

```
source .bashrc
```

Agenda

- Repaso última clase
- Requisitos de inicialización de cluster
- Iniciar Kubernetes
- Instalar Herramientas
- **Instalar Addons**
- Agregar Control Plane y Worker

Calico:

- Solución de red y seguridad de código abierto para Kubernetes.
- Proporciona una red plana y escalable para conectar pods. Enrutamiento y control de acceso basado iptables.

MetalLB:

- Controlador de servicios LoadBalancer de capa 2.
- Permite exponer servicios con direcciones IP externas en clústeres locales.

Ingress:

- Controlador de acceso externo a servicios en el clúster.
- Enruta tráfico externo hacia servicios según nombre url.

Letsencrypt:

- Let's Encrypt proporciona certificados SSL/TLS gratuitos y automatizados
- Los certificados se renuevan automáticamente sin intervención manual, evitando expiraciones.
- Se integra con controlador de Ingress para gestionar la emisión y renovación de certificados de forma transparente.

Instalar Calico

Url

<https://docs.tigera.io/>

Ejecutar

```
kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

Comprobar

```
kubectl get nodes
```

Permitir Pod en control-plane

```
k taint nodes k8s1
```

```
node-role.kubernetes.io/control-plane:NoSchedule-
```

Instalar Metallb

Url

`https://metallb.universe.tf/`

Ejecutar

```
kubectl apply -f  
https://raw.githubusercontent.com/metallb/metallb/v0.15.2/con  
fig/manifests/metallb-native.yaml
```

Setear pool de ip para LoadBalancer

Editar archivo y aplicar

```
kubectl apply -f ipaddresspool.yaml -n metallb-system
```

Comprobar

```
kubectl apply -f example-loadbalancer.yaml -n metallb-system  
kubectl get svc  
kubectl delete svc nginx
```

Ingress:

Url:

<https://kubernetes.github.io/ingress-nginx/>

Ejecutar

Usar la primera ip del pool de LoadBalancer en el comando

```
helm upgrade --install ingress-nginx ingress-nginx --repo  
https://kubernetes.github.io/ingress-nginx --namespace  
ingress-nginx --create-namespace --set  
controller.replicaCount=1,controller.service.loadBalancerIP=172.2  
7.101.240
```

Verificar

```
k get svc -n ingress-nginx
```

Certificados Letsencrypt

Url

<https://cert-manager.io/>

Instalar

```
helm repo add jetstack https://charts.jetstack.io  
helm repo update  
helm install \  
  cert-manager jetstack/cert-manager \  
  --namespace cert-manager \  
  --create-namespace \  
  --version v1.18.2 \  
  --set installCRDs=true
```

Levantar clusterissuers con challenge http de letsencrypt provistos

Completar con email válido antes de aplicar

```
kubectly apply -n cert-manager -f ClusterIssuer/.
```

Agenda

- Repaso última clase
- Requisitos de inicialización de cluster
- Iniciar Kubernetes
- Instalar Herramientas
- Instalar Addons
- **Agregar Control Plane y Worker**

- Repetir los pasos de instalación para el primer nodo hasta la instalación de kubeadm, kubelet y kubectl
- Regenerar los certificados del Control Plane en el primer nodo creado con:
 - `kubeadm init phase upload-certs --upload-certs`
- Obtener nuevo token de Join en el primer nodo creado
 - `kubeadm token create --print-join-command`
- En nodo a unir ejecutar:
 - `kubeadm join <ip-address>:<port> --token <token>
--discovery-token-ca-cert-hash sha256:<hash> --control-plane
--certificate-key <certificate-key>`

Quitar nodo control manager

- Ejecutar en el nodo a quitar:
 - `kubeadm reset`
- En el cluster ejecutar:
 - `kubectrl delete node <Nombre>`

Agregar Nodo Worker

- Repetir los pasos de instalación para el primer nodo hasta la instalación de kubeadm, kubelet y kubectl
- Ejecutar en un nodo master:
 - `kubeadm token create --print-join-command`
- Ejecutar en el nodo a agregar la salida del comando anterior



Preparar Cluster

Preparación para la siguiente
clase

Mirar video “de noob a pro”

- Link lo pueden encontrar en el Moodle

Preparar Cluster en entorno local

- Seguir los pasos realizados en clase
- Agregar los nodos Worker y Control Plane que consideren necesarios.
- Descarguen los archivos de la práctica de la siguiente clase en una PC con acceso al cluster para realizar práctica.
- Se usará el cluster para aprender los recurso de kubernetes

A R I U

Asociación Redes de Interconexión Universitaria

