

Kubernetes: Containers

Universidad Santo Tomás seccional Tunja
Facultad de Ingeniería de Sistemas
Est. Alix Ivonne Chaparro Vasquez.
Est. Laura Sofia Guio Camargo.
Est. Ing. Luis Felipe Narváez Gómez.
2022-1

INDICE

- ❑ ¿Qué hace Kubernetes?
 - ❑ ¿Qué son los contenedores?
 - ❑ Contenedores en Kubernetes
 - ❑ Imágenes de los Contenedores.
 - ❑ Imagen en Ejecución
- ❑ Imágenes
 - ❑ Nombres de las imágenes
 - ❑ Actualización de las imágenes
 - ❑ Política de extracción de imagen
 - ❑ Política de extracción de imágenes predeterminada
 - ❑ Extracción de imagen requerida
 - ❑ ImagePullBackOff
 - ❑ Imágenes multiarquitectura con índices de imagen
 - ❑ Usar un registro privado
 - ❑ Configuración de nodos para autenticarse en un registro privado
 - ❑ Interpretación de config.json
 - ❑ Imágenes extraídas previamente
 - ❑ Especificación de imagePullSecrets en un pod
 - ❑ Hacer referencia a una imagePullSecrets en un pod
- ❑ Entorno del contenedor
 - ❑ Sistema de archivos
 - ❑ Información del contenedor
 - ❑ Información sobre objetos en el clúster
- ❑ RuntimeClass
 - ❑ Configuración de RuntimeClass
 - ❑ Uso de RuntimeClass
- ❑ Ganchos del ciclo de Vida del contenedor.
 - ❑ Tipos de Controladores de los Ganchos.
 - ❑ Ejecución del controlador del gancho.
 - ❑ ¿Qué pasa si falla un gancho?
 - ❑ Garantías de Entrega de gancho
 - ❑ Crear un Hook.



¿Qué hace Kubernetes?

Esta Plataforma OpenSource se dedica a automatizar muchos de los diferentes procesos manuales involucrados en la implementación, la gestión y el ajuste de las aplicaciones que se alojan en los depósitos de Software.



Kubernetes

Qué Es y Para Qué Sirve?

¿Qué son los contenedores?

Los contenedores son un conjunto de tecnologías que en su conjunción generan depósitos de Software.

Las tecnologías que conforman un contenedor son:

- Namespaces** --> Permite almacenar una app, correrla en el depósito y tener una vista de los recursos del SO.
- Cgroups** --> Permite limitar y medir los recursos del SO.
- Chroot** --> Permite al depósito crear su propio entorno de ejecución, partiendo el SO que lo alberga, con su propio root y su propio home.



Contenedores en Kubernetes.

- ❑ Cada contenedor que se ejecuta es repetible.
- ❑ Existe la estandarización de dependencias.
- ❑ El comportamiento del contenedor es transparente independientemente de donde se ejecute.
- ❑ Los contenedores mantienen las aplicaciones desacopladas de la infraestructura del host subyacente.
- ❑ Son de fácil implementación en diferentes SO o entornos en la nube.



Imágenes de los Contenedores

Las imágenes de los contenedores son paquetes de software especiales listos para la ejecución de las aplicaciones que contienen.

Estos paquetes poseen todo lo necesario para la ejecución de la aplicación, esto incluye el código, tiempos de ejecución, bibliotecas de utilidades, bibliotecas de sistemas, configuración de valores predeterminados y cualquier configuración necesaria para el funcionamiento de la aplicación.



Imagen en Ejecución.

En la naturaleza de los contenedores esta la propiedad de ser inmutables, esto quiere decir que el código dentro de un contenedor no puede ser cambiado una vez se crea y ejecuta.

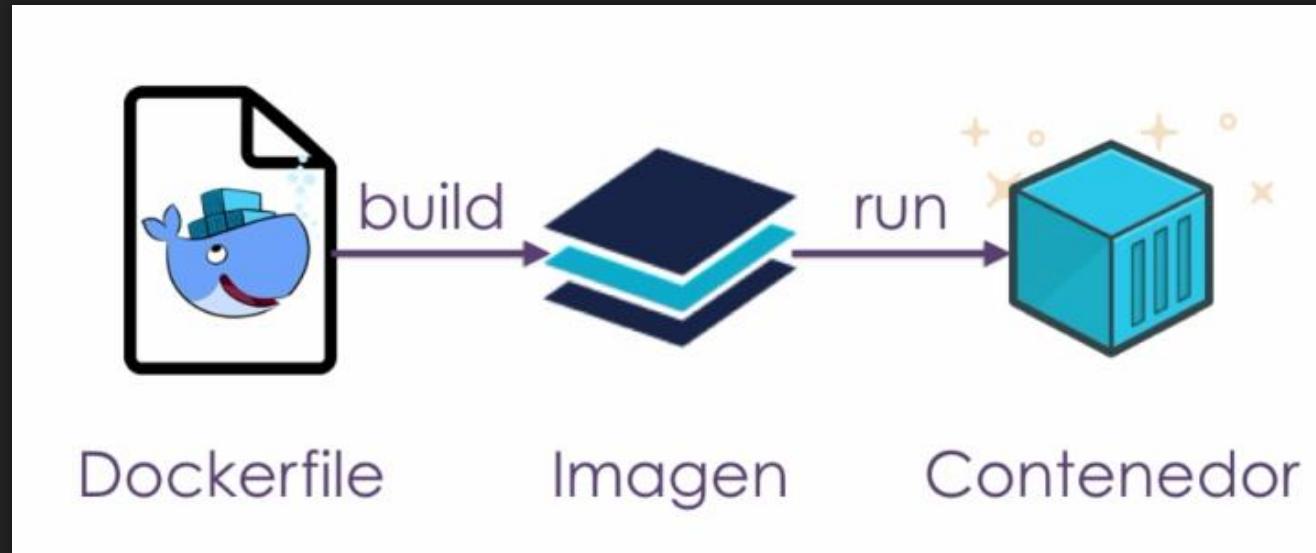
En caso de que queramos hacer cambios en el contenedor, debemos utilizar crear una nueva imagen que contenga las actualizaciones y crear nuevamente el contenedor con estos cambios.



Imágenes

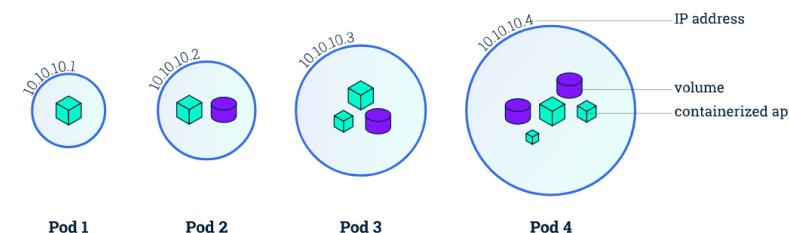
Una imagen de contenedor representa datos binarios que encapsulan una aplicación y todas sus dependencias de software. Las imágenes de contenedor son paquetes de software ejecutables que se pueden ejecutar de forma independiente y que hacen suposiciones muy bien definidas sobre su entorno de tiempo de ejecución.

Por lo general, crea una imagen de contenedor de su aplicación y la envía a un registro antes de hacer referencia a ella en un **Pod**.



Un pod representa un conjunto de contenedores en ejecución en su clúster.

Pods overview



Nombres de las imágenes

Las imágenes de contenedor generalmente reciben un nombre como:

- `pause`
- `example/mycontainer`
- `kube-apiserver`

Las imágenes también pueden incluir un nombre de host de registro:

`fictional.registry.example/imagename`

y posiblemente también un número de puerto:

`fictional.registry.example:10443/imagename`

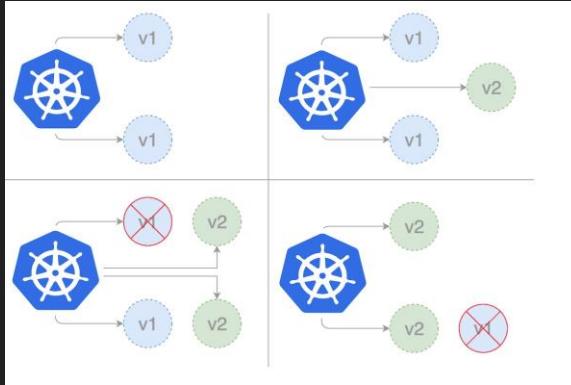
- Si no especifica un nombre de host de registro, Kubernetes asume que se refiere al registro público de Docker.
- Después de la parte del nombre de la imagen, puede agregar una etiqueta (de la misma manera que lo haría con comandos como docker o podman). Las etiquetas le permiten identificar diferentes versiones de la misma serie de imágenes.
- Las etiquetas de imagen consisten en letras minúsculas y mayúsculas, dígitos, guiones bajos (`_`), puntos (`.`) y guiones (`-`).
- Existen reglas adicionales sobre dónde puede colocar los caracteres separadores (`_`, `-` y `.`) dentro de una etiqueta de imagen.
- Si no especifica una etiqueta, Kubernetes asume que se refiere a la etiqueta `latest`.



Actualización de imágenes



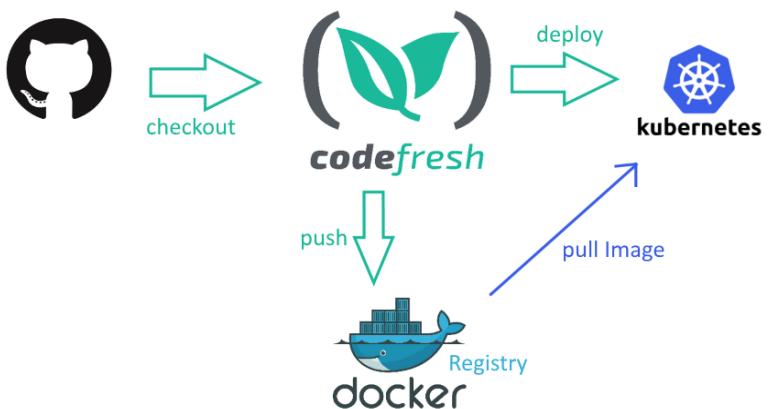
- Deployment: Administra una aplicación replicada en su clúster.
- StatefulSet: Administra la implementación y el escalado de un conjunto de Pods, con almacenamiento duradero e identificadores persistentes para cada Pod.
- Kubelet: Un agente que se ejecuta en cada nodo del clúster. Se asegura de que el contenedor se esté ejecutando en un Pod.



Cuando crea por primera vez un **Deployment**, **StatefulSet**, Pod u otro objeto que incluya una plantilla de Pod, entonces, de manera predeterminada, la política de extracción de todos los contenedores en ese Pod se establecerá en `IfNotPresent` si no se especifica explícitamente. Esta política hace que el **kubelet** omita extraer una imagen si ya existe.

Política de extracción de imágenes

La `imagePullPolicy` para un contenedor y la etiqueta de la imagen afectan cuando el kubelet intenta extraer (descargar) la imagen especificada.



Valores que puede establecer para `imagePullPolicy`

- **IfNotPresent:** la imagen se extrae solo si aún no está presente localmente.
- **Always:** cada vez que el kubelet lanza un contenedor, el kubelet consulta el registro de imágenes del contenedor para resolver el nombre en una imagen **digest**. Si el kubelet tiene una imagen de contenedor con esa misma **digest** almacenado en caché localmente, el kubelet usa su imagen en caché; de lo contrario, el kubelet extrae la imagen con el resumen resuelto y usa esa imagen para iniciar el contenedor.
- **Never:** el kubelet no intenta obtener la imagen. Si la imagen ya está presente localmente, el kubelet intenta iniciar el contenedor; de lo contrario, el inicio falla.

La semántica de almacenamiento en caché del proveedor de imágenes subyacente hace que incluso `imagePullPolicy: Always` sea eficiente, siempre que el registro sea accesible de manera confiable. El tiempo de ejecución de su contenedor puede notar que las capas de imagen ya existen en el nodo, por lo que no es necesario descargarlas nuevamente.

Política de extracción de imágenes

Nota: Debes evitar usar el la etiqueta :latest al implementar contenedores en producción, ya que es más difícil rastrear qué versión de la imagen se está ejecutando y más difícil de revertir correctamente.

En su lugar, especifique una etiqueta significativa como v1.42.0.

Kubernetes imagepullpolicy



www.educba.com

Para asegurarse de que el Pod siempre use la misma versión de una imagen de contenedor, puede especificar el resumen de la imagen; reemplazar <image-name>:<tag> con <image-name>@<digest>.

Ej:

image@sha256:45b23dee08af5e43a7fea6c4cf9c25ccf269ee113168c19722f87876677c5cb2

Al usar etiquetas de imagen, si el registro de imágenes cambiara el código que representa la etiqueta en esa imagen, podría terminar con una combinación de pods que ejecutan el código antiguo y el nuevo. Un resumen de imagen identifica de forma única una versión específica de la imagen, por lo que Kubernetes ejecuta el mismo código cada vez que inicia un contenedor con ese nombre de imagen y resumen especificados. Especificar una imagen por resumen corrige el código que ejecuta para que un cambio en el registro no pueda conducir a esa combinación de versiones.

Hay controladores de admisión de terceros que mutan los pods (y las plantillas de pods) cuando se crean, de modo que la carga de trabajo en ejecución se define en función de un resumen de imagen en lugar de una etiqueta. Eso podría ser útil si desea asegurarse de que toda su carga de trabajo ejecute el mismo código sin importar qué cambios de etiquetas ocurran en el registro.

Política de extracción de imágenes predeterminada

Cuando usted (o un controlador) envía un nuevo Pod al servidor API, su clúster establece el campo `imagePullPolicy` cuando se cumplen condiciones específicas:

- si omite el campo `imagePullPolicy` y la etiqueta de la imagen del contenedor es `:latest`, `imagePullPolicy` se establece automáticamente en `Always`;
- si omite el campo `imagePullPolicy` y no especifica la etiqueta para la imagen del contenedor, `imagePullPolicy` se establece automáticamente en `Always`;
- si omite el campo `imagePullPolicy` y especifica la etiqueta para la imagen del contenedor que no es `:latest`, `imagePullPolicy` se establece automáticamente en `IfNotPresent`.

Nota: El valor de `imagePullPolicy` del contenedor siempre se establece cuando el objeto se crea por primera vez y no se actualiza si la etiqueta de la imagen cambia más adelante.

Por ejemplo, si crea una implementación con una imagen cuya etiqueta no es `:latest` y luego actualiza la imagen de esa implementación a una etiqueta `:latest`, el campo `imagePullPolicy` no cambiará a `Always`. Debe cambiar manualmente la política de extracción de cualquier objeto después de su creación inicial.



Extracción de imagen requerida

Si desea forzar siempre un pull, puede realizar una de las siguientes acciones:

- Establezca `imagePullPolicy` del contenedor en `Always`.
- Omita `imagePullPolicy` y use `:latest` como etiqueta para la imagen a usar; Kubernetes establecerá la política en `Always` cuando envíe el Pod.
- Omita `imagePullPolicy` y la etiqueta de la imagen a usar; Kubernetes establecerá la política en `Always` cuando envíe el Pod.
- Habilite el controlador de admisión `AlwaysPullImages`.

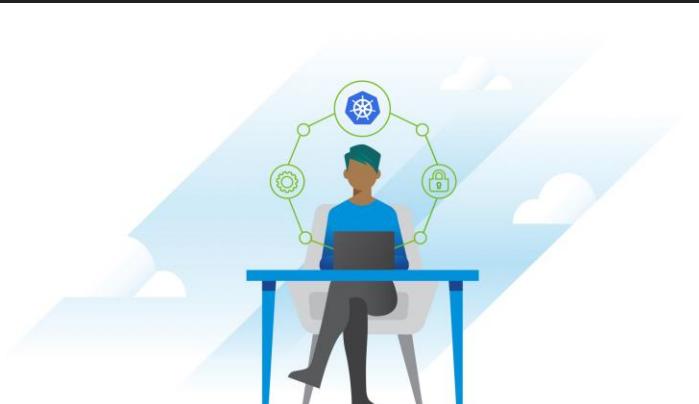


ImagePullBackOff

Cuando un kubelet comienza a crear contenedores para un pod mediante un tiempo de ejecución de contenedor, es posible que el contenedor esté en estado de Waiting debido a ImagePullBackOff.

El estado ImagePullBackOff significa que un contenedor no pudo iniciarse porque Kubernetes no pudo extraer una imagen del contenedor (por motivos como un nombre de imagen no válido o extracción de un registro privado sin imagePullSecret). La parte BackOff indica que Kubernetes seguirá intentando extraer la imagen, con un retraso de retroceso cada vez mayor.

Kubernetes aumenta la demora entre cada intento hasta que alcanza un límite compilado, que es de 300 segundos (5 minutos).



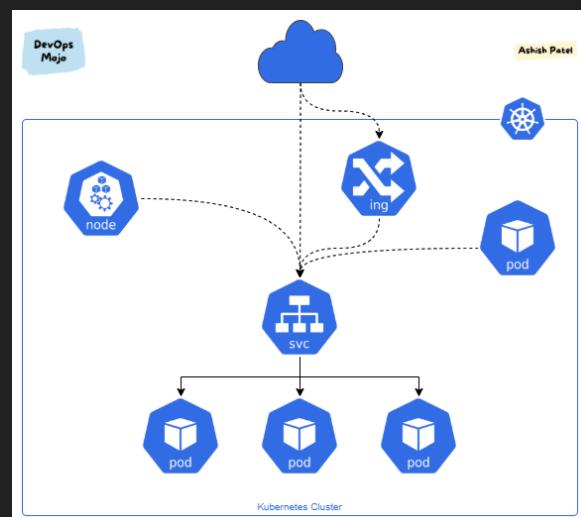
Imágenes multiarquitectura con índices de imagen

Además de proporcionar imágenes binarias, un registro de contenedor también puede proporcionar un índice de imagen de contenedor. Un índice de imágenes puede apuntar a múltiples manifiestos de imágenes para versiones específicas de la arquitectura de un contenedor. La idea es que pueda tener un nombre para una imagen

pause, example/mycontainer, kube-apiserver

y permitir que diferentes sistemas obtengan la imagen binaria correcta para la arquitectura de la máquina que están usando.

El propio Kubernetes normalmente nombra imágenes de contenedores con un sufijo `-$(ARCH)`. Para compatibilidad con versiones anteriores, genere las imágenes más antiguas con sufijos. La idea es generar, por ejemplo, una imagen de pausa que tenga el manifiesto para todos los arcos y, por ejemplo, `pausa-amd64`, que es compatible con configuraciones anteriores o archivos YAML que pueden tener codificadas las imágenes con sufijos.



Usar un registro privado

Los registros privados pueden requerir claves para leer imágenes de ellos.

Las credenciales se pueden proporcionar de varias maneras:



kubernetes

- Configuración de nodos para autenticarse en un registro privado
 - Todos los pods pueden leer cualquier registro privado configurado
 - Requiere la configuración del nodo por parte del administrador del clúster
- Imágenes extraídas previamente
 - Todos los pods pueden usar cualquier imagen almacenada en caché en un nodo
 - Requiere acceso raíz a todos los nodos para configurar
- Especificación de ImagePullSecrets en un pod
 - Solo los pods que proporcionan claves propias pueden acceder al registro privado
- Extensiones locales o específicas del proveedor
 - si usa una configuración de nodo personalizada, usted (o su proveedor de la nube) puede implementar su mecanismo para autenticar el nodo en el registro del contenedor.

Configuración de nodos para autenticarse en un registro privado

Las instrucciones específicas para configurar las credenciales dependen del tiempo de ejecución del contenedor y del registro que elija usar. Debe consultar la documentación de su solución para obtener la información más precisa.



Interpretación de config.json

La interpretación de config.json varía entre la implementación original de Docker y la interpretación de Kubernetes. En Docker, las claves de autenticación solo pueden especificar direcciones URL raíz, mientras que Kubernetes permite direcciones URL globales, así como rutas con coincidencia de prefijo. Esto significa que un config.json como este es válido:

```
{  
  "auths": {  
    "*my-registry.io/images": {  
      "auth": "..."  
    }  
  }  
}
```

```
pattern:  
  { term }  
  
term:  
  '*'      matches any sequence of non-Separator characters  
  '?'      matches any single non-Separator character  
  '[' [ '^' ] { character-range } ']'  
          character class (must be non-empty)  
  c        matches character c (c != '*', '?', '\\\\', '[')  
  '\\\\' c    matches character c  
  
character-range:  
  c        matches character c (c != '\\\\', '-', ']')  
  '\\\\' c    matches character c  
  lo '-' hi  matches character c for lo <= c <= hi
```

Interpretación de config.json

La URL raíz (*my-registry.io) se compara con la siguiente sintaxis:

Interpretación de config.json

Las operaciones de obtención de imágenes ahora pasarán las credenciales al tiempo de ejecución del contenedor CRI para cada patrón válido. Por ejemplo, los siguientes nombres de imágenes de contenedores coincidirían correctamente:

- `my-registry.io/images`
- `my-registry.io/images/my-image`
- `my-registry.io/images/another-image`
- `sub.my-registry.io/images/my-image`
- `a.sub.my-registry.io/images/my-image`

Interpretación de config.json

El kubelet realiza extracciones de imágenes secuencialmente para cada credencial encontrada. Esto significa que también son posibles varias entradas en config.json:

```
{  
  "auths": {  
    "my-registry.io/images": {  
      "auth": "..."  
    },  
    "my-registry.io/images/subpath": {  
      "auth": "..."  
    }  
  }  
}
```

Si ahora un contenedor especifica que se extraiga una imagen my-registry.io/images/subpath/my-image, entonces el kubelet intentará descargarlos de ambas fuentes de autenticación si una de ellas falla.

Imágenes extraídas previamente

Nota: este enfoque es adecuado si puede controlar la configuración del nodo. No funcionará de manera confiable si su proveedor de nube administra los nodos y los reemplaza automáticamente.

De forma predeterminada, el kubelet intenta extraer cada imagen del registro especificado. Sin embargo, si la propiedad `imagePullPolicy` del contenedor se establece en `IfNotPresent` o `Never`, se usa una imagen local (de manera preferencial o exclusiva, respectivamente).

Si desea confiar en las imágenes extraídas previamente como sustituto de la autenticación del registro, debe asegurarse de que todos los nodos del clúster tengan las mismas imágenes extraídas previamente.

Esto se puede usar para precargar ciertas imágenes para aumentar la velocidad o como una alternativa a la autenticación en un registro privado.

Todos los pods tendrán acceso de lectura a cualquier imagen extraída previamente.

Especificación de imagePullSecrets en un pod

Nota: Este es el enfoque recomendado para ejecutar contenedores basados en imágenes en registros privados.

Kubernetes admite la especificación de claves de registro de imágenes de contenedores en un pod.

Crear un secreto con una configuración de Docker

Debe conocer el nombre de usuario, la contraseña del registro y la dirección de correo electrónico del cliente para autenticarse en el registro, así como su nombre de host. Ejecute el siguiente comando, sustituyendo los valores en mayúscula apropiados:

```
kubectl create secret docker-registry <name>
--docker-server=DOCKER_REGISTRY_SERVER
--docker-username=DOCKER_USER
--docker-password=DOCKER_PASSWORD
--docker-email=DOCKER_EMAIL
```

Si ya tiene un archivo de credenciales de Docker, en lugar de usar el comando anterior, puede importar el archivo de credenciales como secretos de Kubernetes.

Esto es especialmente útil si utiliza varios registros de contenedores privados, ya que kubectl create secret docker-registry crea un secreto que solo funciona con un único registro privado.

Nota: Los pods solo pueden hacer referencia a secretos de extracción de imágenes en su propio espacio de nombres, por lo que este proceso debe realizarse una vez por espacio de nombres.

Hacer referencia a una imagenPullSecrets en un pod

Ahora, puede crear pods que hagan referencia a ese secreto agregando una sección imagePullSecrets a una definición de pod.

```
cat <<EOF > pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: foo
  namespace: awesomeapps
spec:
  containers:
    - name: foo
      image: janedoe/awesomeapp:v1
  imagePullSecrets:
    - name: myregistrykey
EOF

cat <<EOF >> ./kustomization.yaml
resources:
- pod.yaml
EOF
```

Esto debe hacerse para cada pod que utilice un registro privado.

Sin embargo, la configuración de este campo se puede automatizar configurando imagePullSecrets en un recurso ServiceAccount.

Puede usar esto junto con un .docker/config.json por nodo. Las credenciales se fusionarán.

Entorno del contenedor

Proporciona elementos importantes para los contenedores:

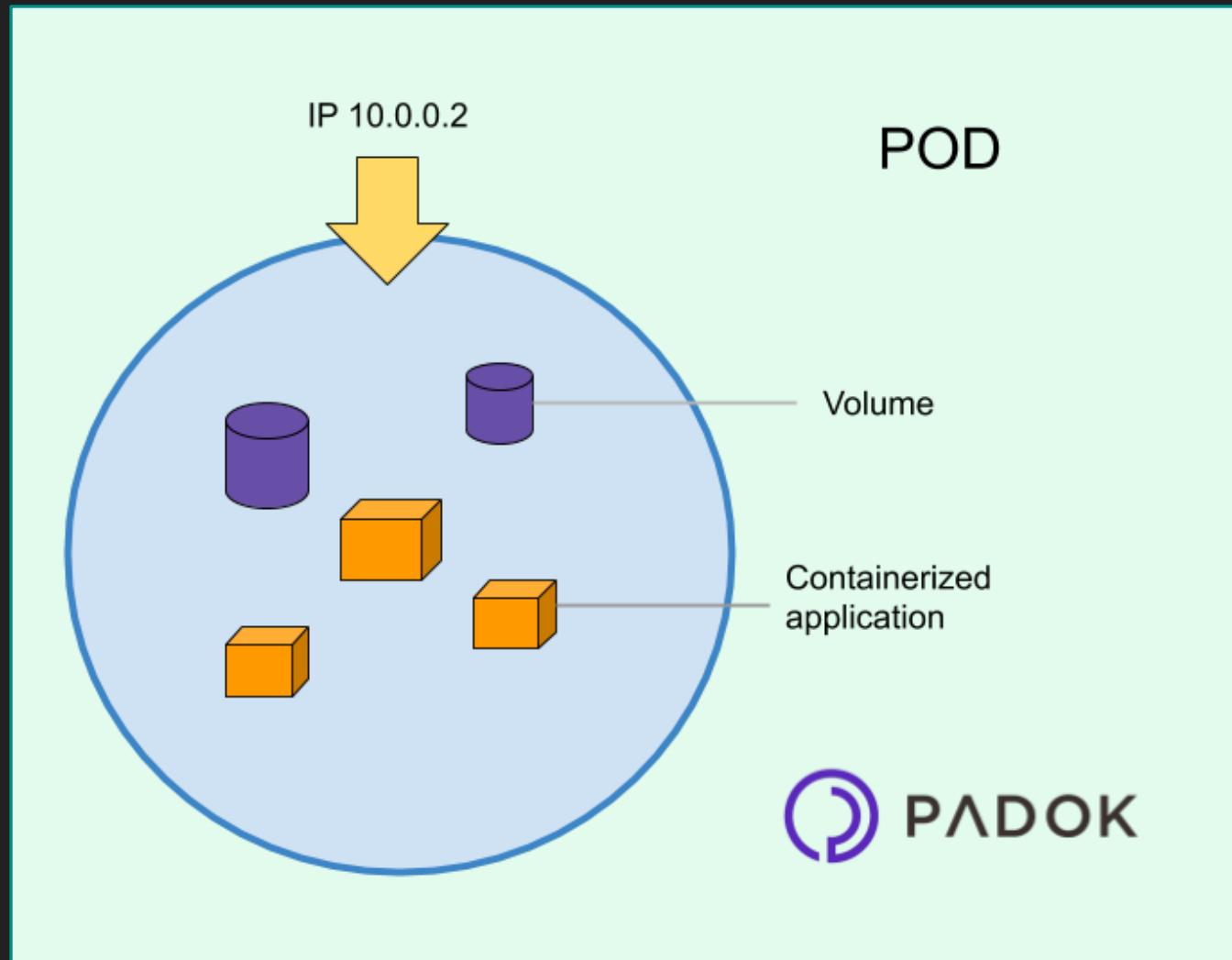
- Sistema de archivos**
- Información del contenedor**
- Información sobre objetos en el clúster**



Sistema de archivos

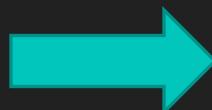
Combinación de una imagen con uno o más volúmenes.

- ❑ **Imagen** → Paquetes de software ejecutables
- ❑ **Volumen** → Directorio al que pueden acceder los contenedores de un POD.
- ❑ POD → Conjunto de uno o más contenedores



Información del contenedor

Nombre del Host → nombre del POD en el que se ejecuta contenedor.



Para obtenerlo se ejecuta:

- Hostname**
- gethostname**

Está disponible como variable de entorno en la API descendente.

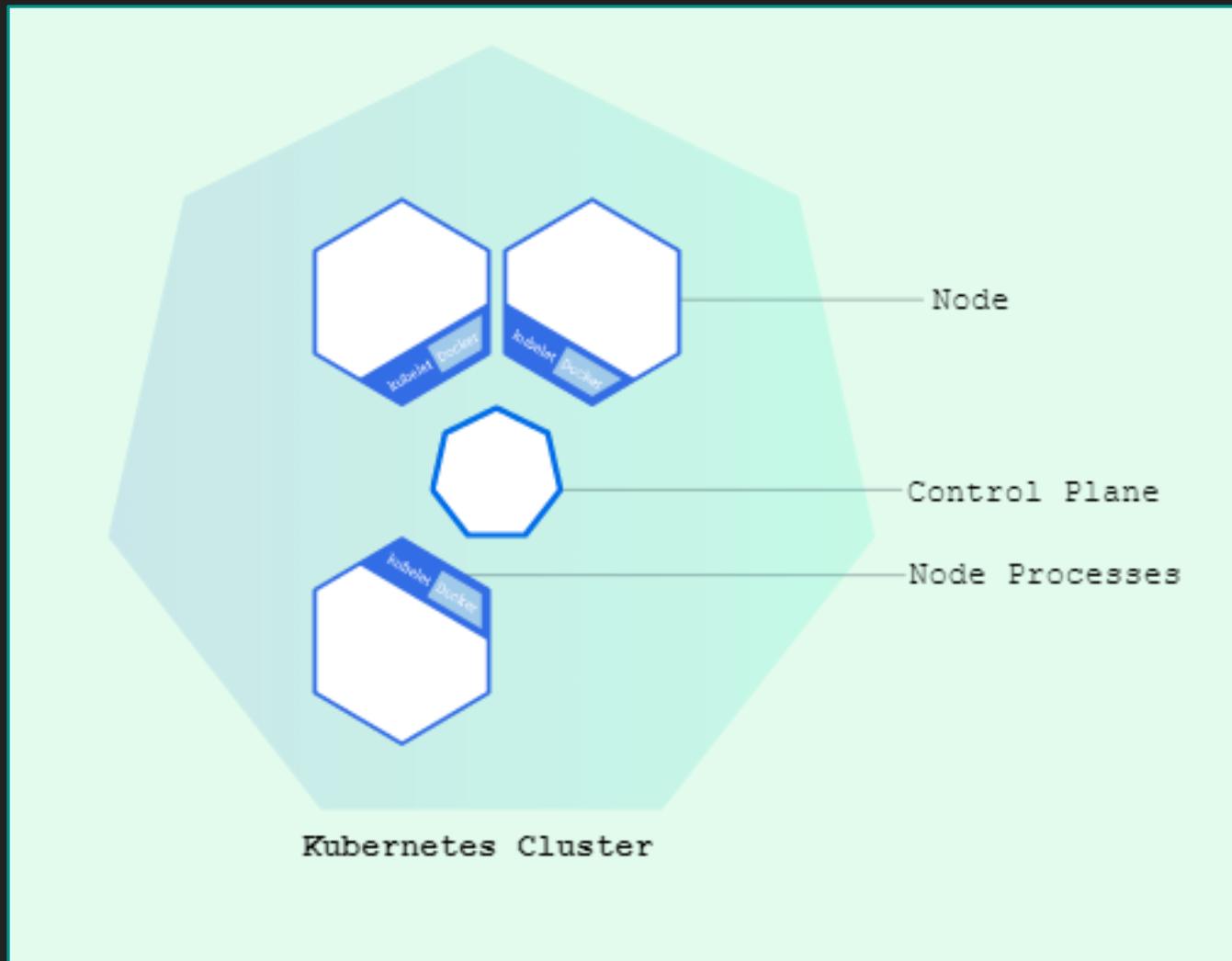
Las variables de entorno establecidas por el usuario también están presentes para el contenedor.

Información sobre objetos en el clúster

Todos los servicios que fueron ejecutados a la hora de crear un contenedor se encuentran como variables de entorno del mismo.

Aquí se encuentran los servicios que comparten el mismo namespace que el POD, cuentan con un host y un puerto.

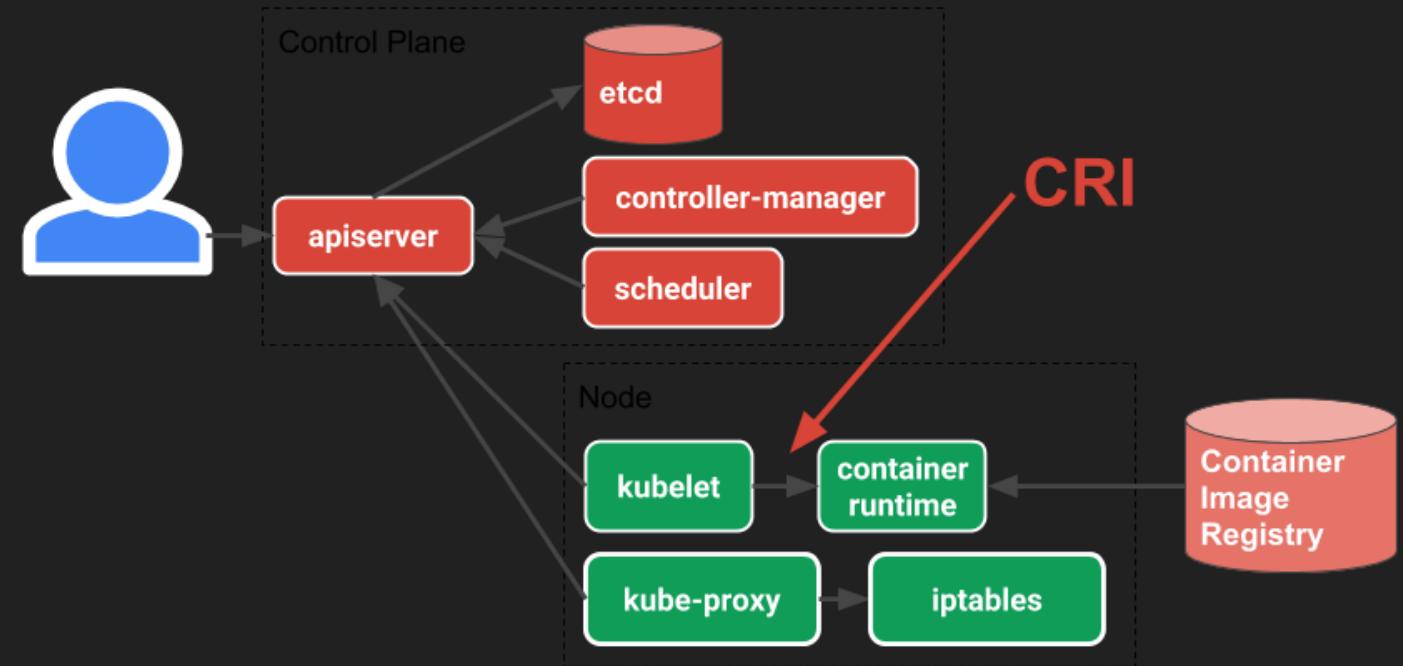
Se encuentran disponibles en el contendor por medio de su dirección IP, siempre y cuando el complemento DNS esté activo.



RuntimeClass

Recurso con el que se puede seleccionar la configuración de tiempo de ejecución de los contenedores de un POD.

Esta configuración puede ser distinta en los PODs con el fin de proporcionar cierto rendimiento y seguridad.



Configuración de RuntimeClass

1. Configurar implementación de CRI en nodos

- Cada configuración tiene su correspondiente nombre de controlador (handler).
- Para configurar el CRI → /etc/containerd/config.toml o también por medio de CRI-O.

2. Crear los recursos RuntimeClass correspondientes

- Cada configuración tiene un handler y cada controlador debe contar con un objeto RuntimeClass.
- Este recurso tiene dos campos importantes, nombre del RuntimeClass (`metadata.name`) y su controlador (handler)

```
apiVersion: node.k8s.io/v1 # RuntimeClass is defined in the node.k8s.io API group
kind: RuntimeClass
metadata:
  name: myclass # The name the RuntimeClass will be referenced by
  # RuntimeClass is a non-namespaced resource
  handler: myconfiguration # The name of the corresponding CRI configuration
```



Uso de RuntimeClass

Una vez sean configuradas se pueden usar de la siguiente manera

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  runtimeClassName: myclass
  # ...
```

En las especificaciones del pod se especifica el runtimeClassName, lo cual indica al kubelet (agente de nodo principal que se ejecuta en cada nodo) usar a RuntimeClass nombrada. Si esta no existe o el CRI no puede ejecutar el controlador aparecerá error.

Si este no se especifica se utilizará el predeterminado que es equivalente a deshabilitar la característica RuntimeClass

Ganchos del ciclo de vida del contenedor

Kubernetes ofrece enlaces de ciclo de vida o ganchos de ciclo de vida, los cuales permiten estar al tanto de la administración y ejecución del código (como un historial del tiempo de vida del contenedor).

Hay dos clases de ganchos para los controladores:

- PostStart
- PreStop



Tipos de Controladores de Ganchos

Existen dos tipos de controladores principales para la configuración de los ganchos:

- ❑ **Exec:** este se encarga de ejecutar un comando específico, esto dentro de los Cgroups y NamesSpaces.
- ❑ **HTTP:** este ejecuta una solicitud de tipo http contra una finalización de ejecución prevista en el contenedor.



Ejecución del controlador de gancho.

Dependiendo de la acción que se esté ejecutando, el Kubernetes ejecutara el controlador consecuente, ya sea HttpGet, tcpSocket o exec.

La llamadas que se dan al controlador son sincronas dentro del contexto de POD del contenedor.

Lo anterior quiere decir que un comando PostStart ejecutara el gancho y el contenedor de forma separada y asíncrona, salvo el mismo demore mucho en responder y no alcance el estado de running



¿Qué pasa si Falla un Gancho?

Si un gancho **PostStart** o **PreStop** falla, mata al Contenedor.

Si un **PreStop** gancho se cuelga durante la ejecución, la fase del Pod será **Terminating** y permanecerá allí hasta que el Pod se elimine después de que **terminationGracePeriodSeconds** expire.



Garantías de entrega de gancho

Los registros de un controlador Hook no se exponen en los eventos Pod.

Si un controlador falla por alguna razón, transmite un evento

Ejemplo:

- Para PostStart ---> FailedPostStartHook
- Para PreStop, ---> FailedPreStopHook



Generar un Hook

Para generar un `FailedPreStopHook` evento fallido usted mismo, modifique el archivo `lifecycle-events.yaml` para cambiar el comando `postStart` a "badcommand" y aplicarlo.



REFERENCIAS

1. <https://www.redhat.com/es/topics/containers/what-is-kubernetes>
2. https://platzi.com/contributions/guia-del-curso-de-docker/?utm_source=google&utm_medium=cpc&utm_campaign=12915366154&utm_adgroup=&utm_content=&gclid=CjwKCAjw9LSSBhBsEiwAKtf0n28N41MG31Wo_GKX_eWvKlyEfQk3rSUK8lt9c6LefPR-V3HP_tL64xoCsdEQAvD_BwE&gclsrc=aw.ds
3. <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>
4. <https://kubernetes.io/docs/concepts/containers/>
5. <https://kodekloud.com/learning-path-kubernetes/>
6. <https://cri-o.io/#what-is-cri-o>
7. <https://github.com/kubernetes/community/blob/master/contributors/design-sig-node/container-runtime-interface.md>
8. <https://kubernetes.io/docs/concepts/containers/runtime-class/>
9. <https://kubernetes.io/docs/concepts/containers/container-environment/>





UNIVERSIDAD SANTO TOMÁS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
SECCIONAL TUNJA

VIGILADA MINEDUCACIÓN - SNIES 1732

Acreditación Institucional
Internacional
OTORGADA POR EL IAC CINDA ACUERDO 55 DEL 9 DE MAYO-VIGENCIA 5 AÑOS

