

# MÓDULO 8

---

Creación y publicación de imágenes de contenedores

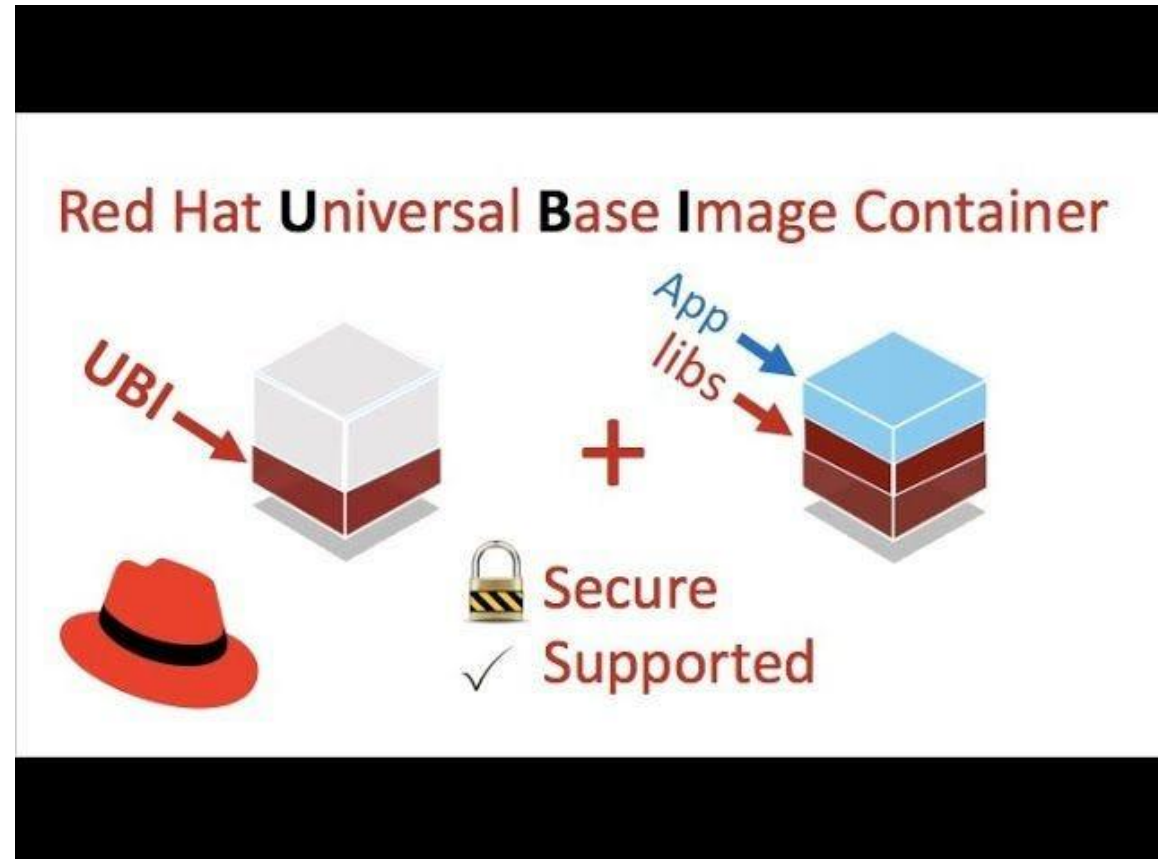
# IMÁGENES DE CONTENEDOR PARA RED HAT OPENSIFT

---



# USO DE RED HAT UNIVERSAL BASE IMAGES

- Al definir imágenes de contenedor personalizadas, Red Hat recomienda usar la imagen base universal (UBI) de Red Hat como imagen de contenedor base para sus aplicaciones.
- Las imágenes UBI son imágenes certificadas, probadas y mantenidas con regularidad que Red Hat proporciona sin costo alguno



# TIPOS DE IMÁGENES UBI

Red Hat proporciona cuatro tipos de imágenes UBI, diseñadas para cubrir la mayoría de los casos de uso:

Tipo de imagen	Nombre de imagen	Descripción
Estándar	ubi	Para la mayoría de las aplicaciones y casos de uso.
Init	ubi-init	Para contenedores que ejecutan varios servicios systemd.
Mínima	ubi-minimal	Imagen más pequeña para aplicaciones que gestionan sus propias dependencias y dependen de menos componentes del sistema operativo.
Micro	ubi-micro	La imagen más pequeña para casos de uso optimizados de espacio de memoria. Para aplicaciones que casi no usan componentes del sistema operativo.



# IMÁGENES UBI DE TIEMPO DE EJECUCIÓN PARA DESARROLLADORES

- Además de las cuatro imágenes UBI principales, Red Hat proporciona imágenes UBI específicas para tiempos de ejecución populares.
- Para cada tiempo de ejecución, Red Hat proporciona imágenes para cada versión principal admitida del tiempo de ejecución.

- OpenJDK
- Node.js
- Python
- PHP
- .NET
- Go
- Ruby



# OPTIMIZAR CONTAINERFILES PARA OPENSIFT

- Comience usando una imagen UBI en la instrucción FROM de su Containerfile.
- Las imágenes UBI están disponibles en el registro de contenedores público [registry.access.redhat.com](https://registry.access.redhat.com).
- El formato del nombre de la imagen es el siguiente:

```
registry.access.redhat.com/NAMESPACE/NAME[:TAG]
```

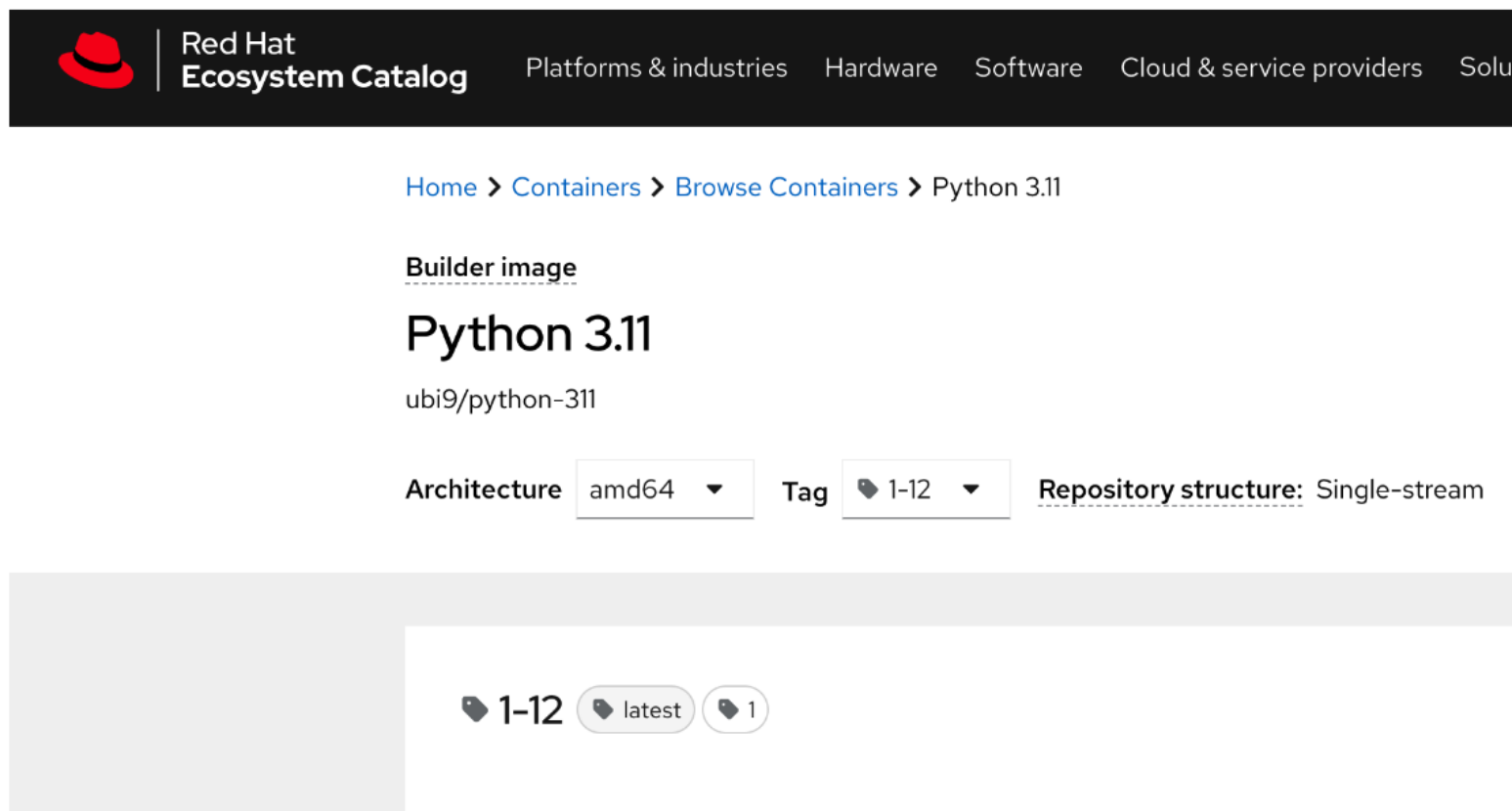
- La parte **NAMESPACE** especifica la versión de UBI que usa, como `ubi8` o `ubi9`

```
FROM registry.access.redhat.com/ubi9/nodejs-18-minimal:1-56
```



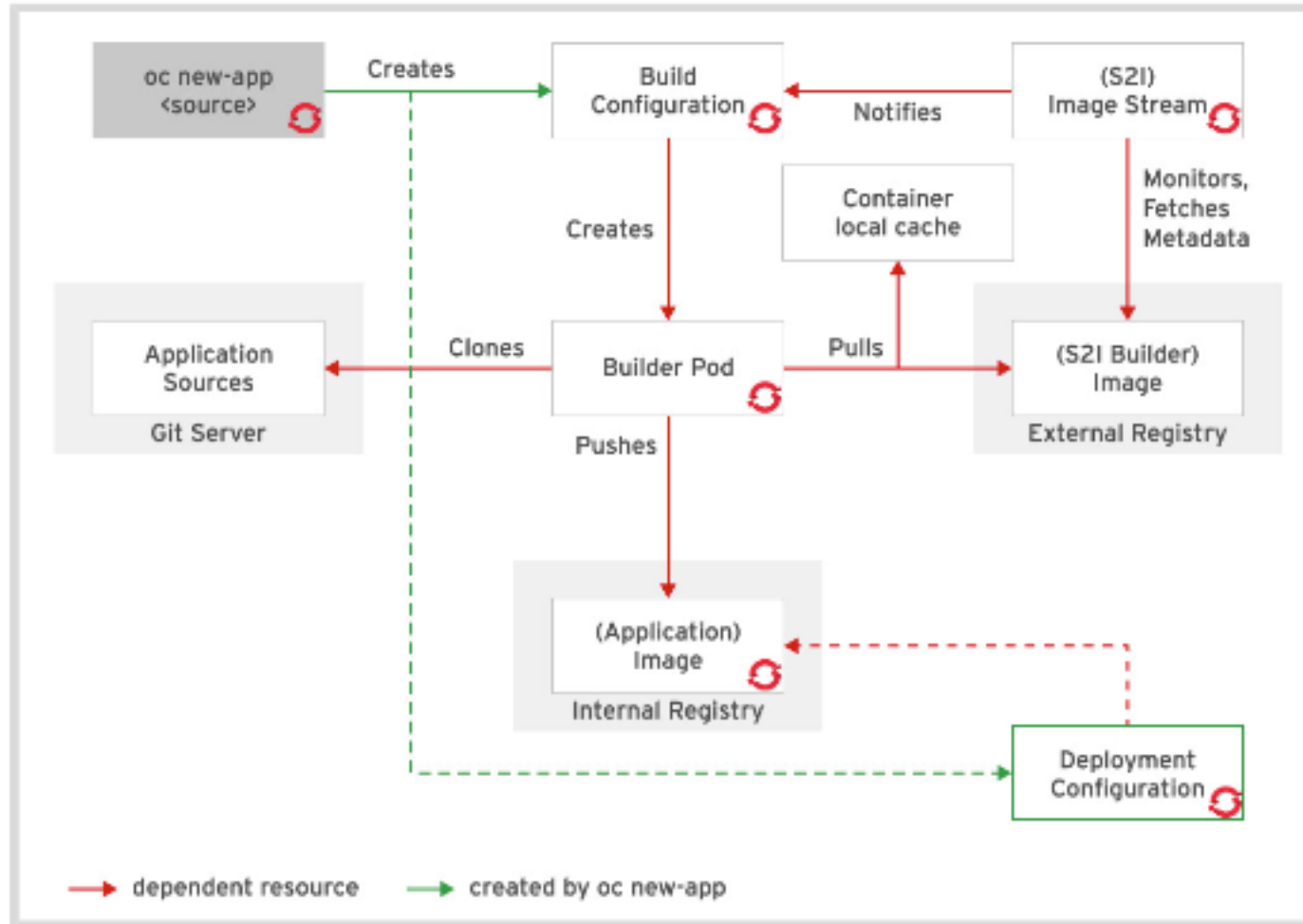
# USO DE INSTRUCCIONES ESPECÍFICAS DE UBI

- La pestaña Descripción general del Catálogo de ecosistemas de Red Hat proporciona instrucciones para usar una imagen :



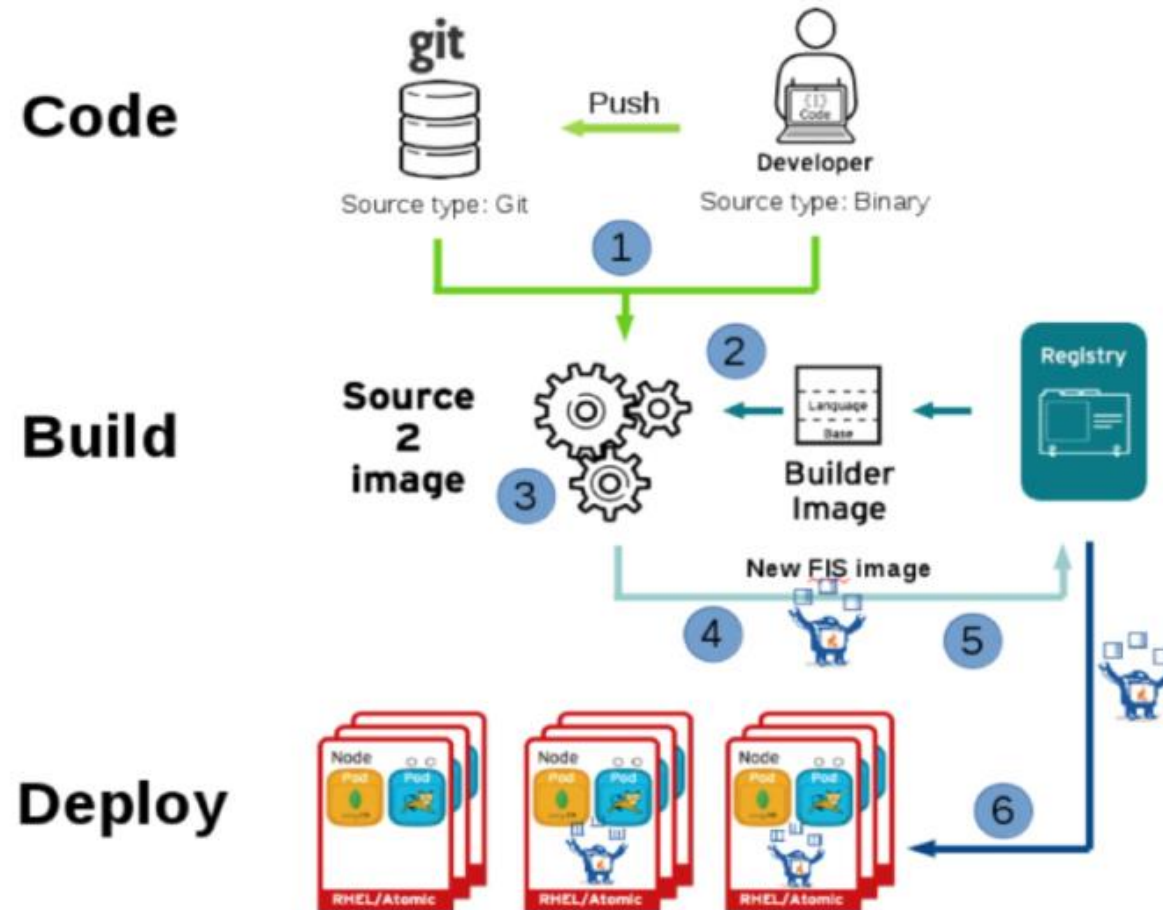
The screenshot displays the Red Hat Ecosystem Catalog interface. At the top, there is a navigation bar with the Red Hat logo and the text "Red Hat Ecosystem Catalog". To the right of the logo, there are links for "Platforms & industries", "Hardware", "Software", "Cloud & service providers", and "Solutions". Below the navigation bar, a breadcrumb trail shows the path: "Home > Containers > Browse Containers > Python 3.11". The main heading is "Builder image" followed by "Python 3.11" and the image identifier "ubi9/python-311". Below this, there are filters for "Architecture" (set to "amd64"), "Tag" (set to "1-12"), and "Repository structure" (set to "Single-stream"). At the bottom, there are buttons for "1-12", "latest", and "1".

# CREACIÓN DE APLICACIONES CON SOURCE-TO-IMAGE





# ¿CÓMO PUEDE AYUDARTE EL SOURCE TO IMAGE (S2I)?



# AGREGADO DE SOPORTE PARA USUARIOS NO ROOT ARBITRARIOS

- Red Hat OpenShift, de manera predeterminada, no respeta la instrucción **USER** definida por una imagen de contenedor.
- Por motivos de seguridad, OpenShift usa un ID de usuario aleatorio diferente al ID de usuario root (0) para ejecutar contenedores.
- OpenShift convierte a este usuario aleatorio en miembro del grupo root, que corresponde al ID de grupo 0.
- instrucción **RUN** en Containerfile de manera recurrente, se definen los permisos de un directorio para permitir que los usuarios del grupo **root** accedan a este directorio y sus contenidos en el contenedor.

```
RUN chgrp -R 0 directory && \ ❶  
chmod -R g=u directory ❷
```

- ❶ El comando `chgrp` cambia la propiedad del grupo de `directory` al grupo root (ID de grupo 0). La cuenta de usuario arbitraria que ejecuta el contenedor en OpenShift siempre es miembro del grupo root.
- ❷ Iguala los permisos del grupo con los permisos del usuario propietario, que de manera predeterminada son de lectura y escritura. Puede usar el argumento `g+rwX` con los mismos resultados.

# SEÑALES DE INTERRUPCIÓN POR PARTE DE LOS CONTENEDORES

- OpenShift espera que las instancias de la aplicación se apaguen correctamente antes de que el clúster elimine las instancias del balanceador de carga.
- OpenShift envía una señal **SIGTERM** a los procesos en el contenedor para finalizar una aplicación

```
#!/bin/env bash

function graceful_shutdown() {
    kill -SIGTERM "$java_pid"
    wait "$java_pid"
    exit 0
}

# Trap the SIGTERM signal
trap graceful_shutdown SIGTERM

...script omitted...

# Start the application
java -jar example.jar &
java_pid=$!

...script omitted...

# Wait for the process to finish
wait "$java_pid"
```

# CICLO DE VIDA DEL POD : PRESTOP

- Puede usar el enlace (hook) del ciclo de vida del pod **preStop** para iniciar un cierre ordenado de su aplicación

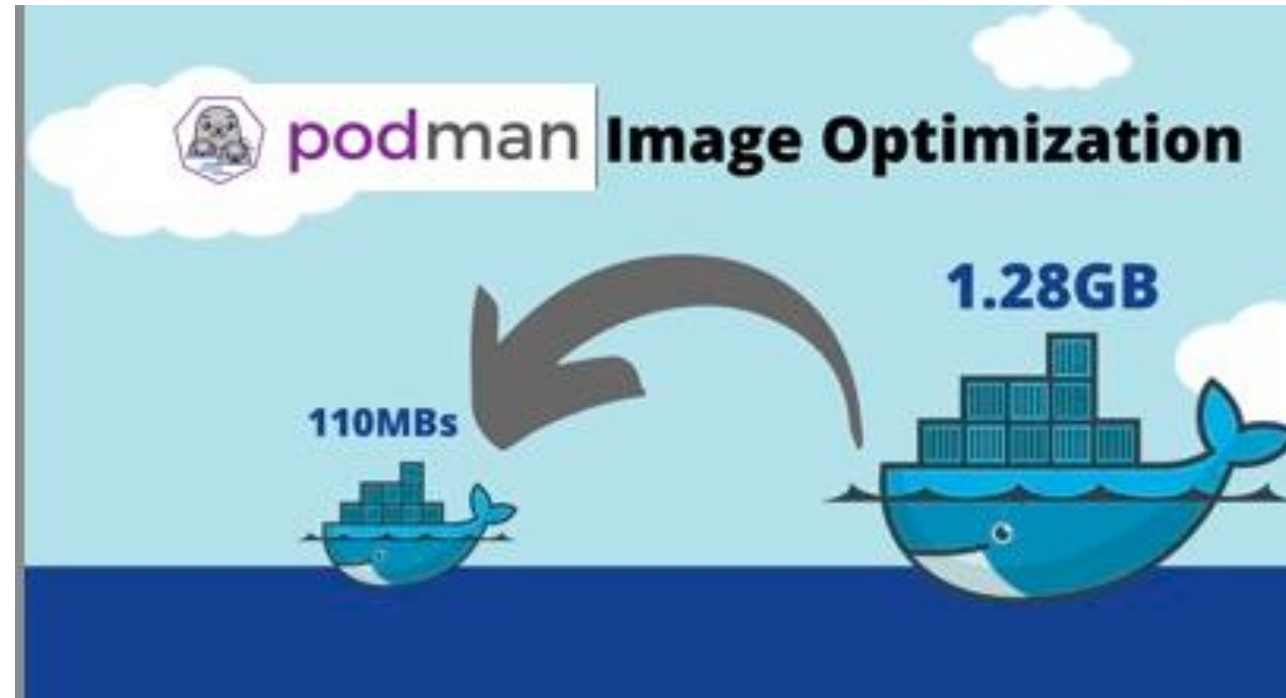
```
apiVersion: v1
kind: Pod
metadata:
  name: example-pod
spec:
  containers:
    - name: my-container
      image: example.com/myimage
      lifecycle:
        preStop:
          httpGet:
            path: /shutdown
            port: 8080
```



# REDUCCIÓN DEL TAMAÑO DE LA IMAGEN

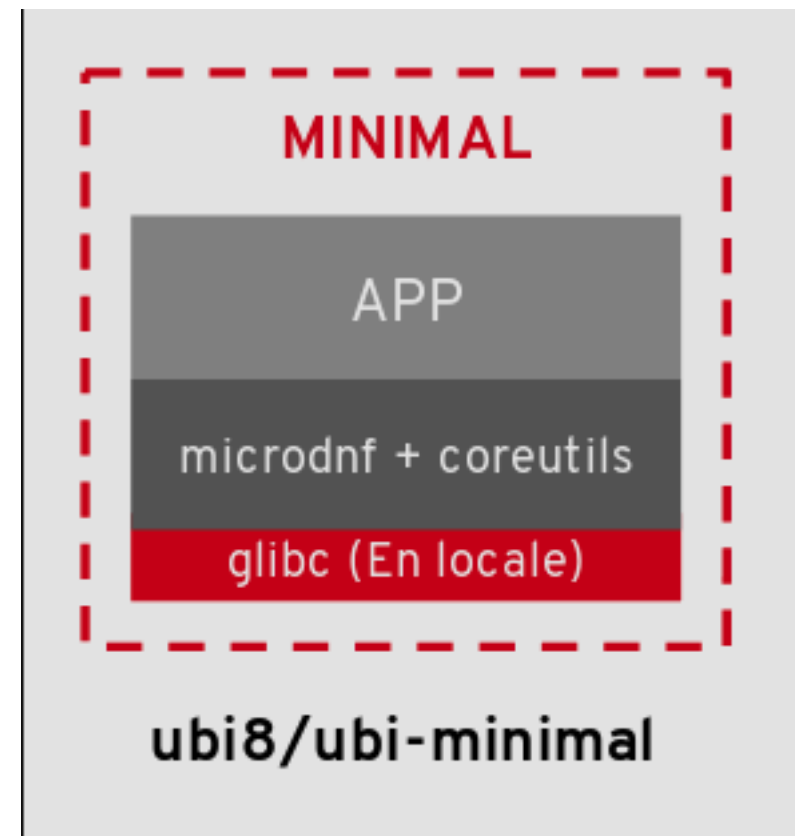
Reducir el tamaño de los contenedores personalizados mediante las siguientes técnicas:

- Evite tener demasiadas instrucciones RUN en un Containerfile.
- Excluya archivos y directorios del contexto de compilación, con `.containerignore` o `.dockerignore`.
- Utilice Containerfiles de varias etapas



# USO DE UNA IMAGEN MÍNIMA

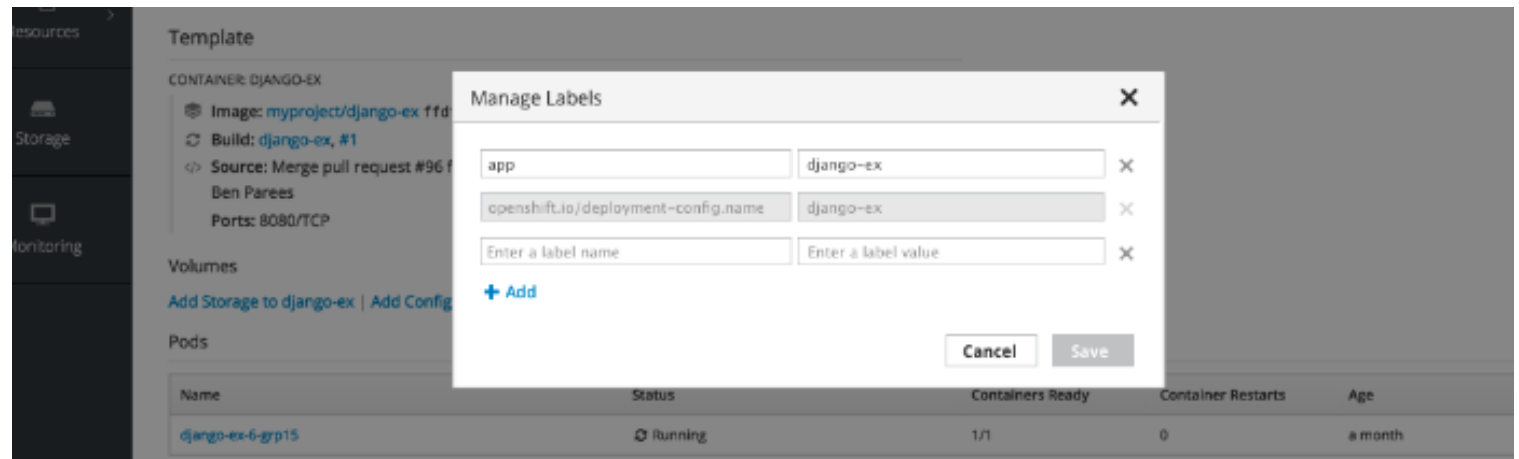
- Siempre que sea posible, use una imagen UBI mínima para reducir el tamaño de sus contenedores
- El nombre de las imágenes mínimas puede diferir según el tiempo de ejecución. Ejemplo “ubi9/nodejs-18-minimal”



# DEFINICIÓN DE ETIQUETAS DE METADATOS

- La instrucción LABEL para definir información, advertencias y sugerencias que OpenShift puede mostrar a los usuarios.
- Los espacios de nombres de las etiquetas están definidos con los prefijos io.openshift o io.k8s

```
LABEL io.openshift.min-cpu 2
```



# USO DE DIRECTORIOS DE TRABAJO

- Red Hat recomienda usar rutas absolutas en las instrucciones **WORKDIR**.
- Use **WORKDIR** en lugar de varias instrucciones **RUN**, donde puede cambiar directorios y, luego, ejecutar algunos comandos.

## What is **WORKDIR** in Dockerfile

**WORKDIR**

**Mention this is in Dockerfile**

**/USR/APP**

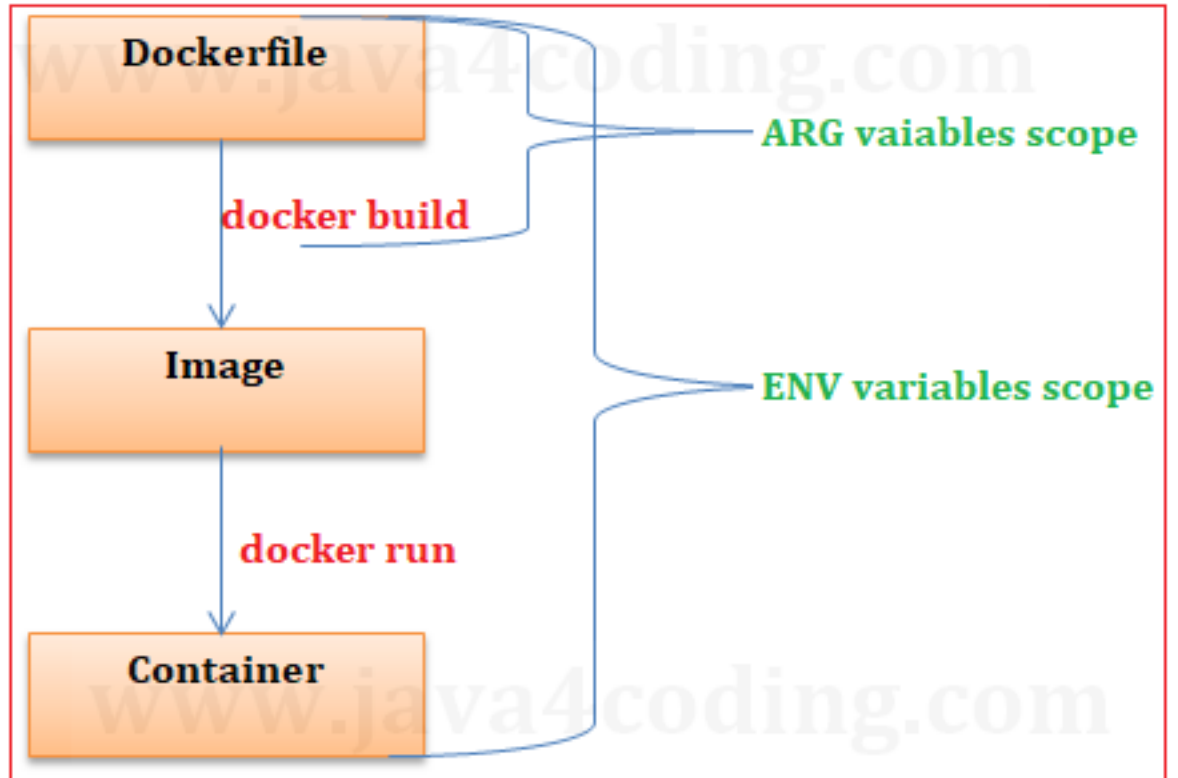
**This will be working directory inside container**





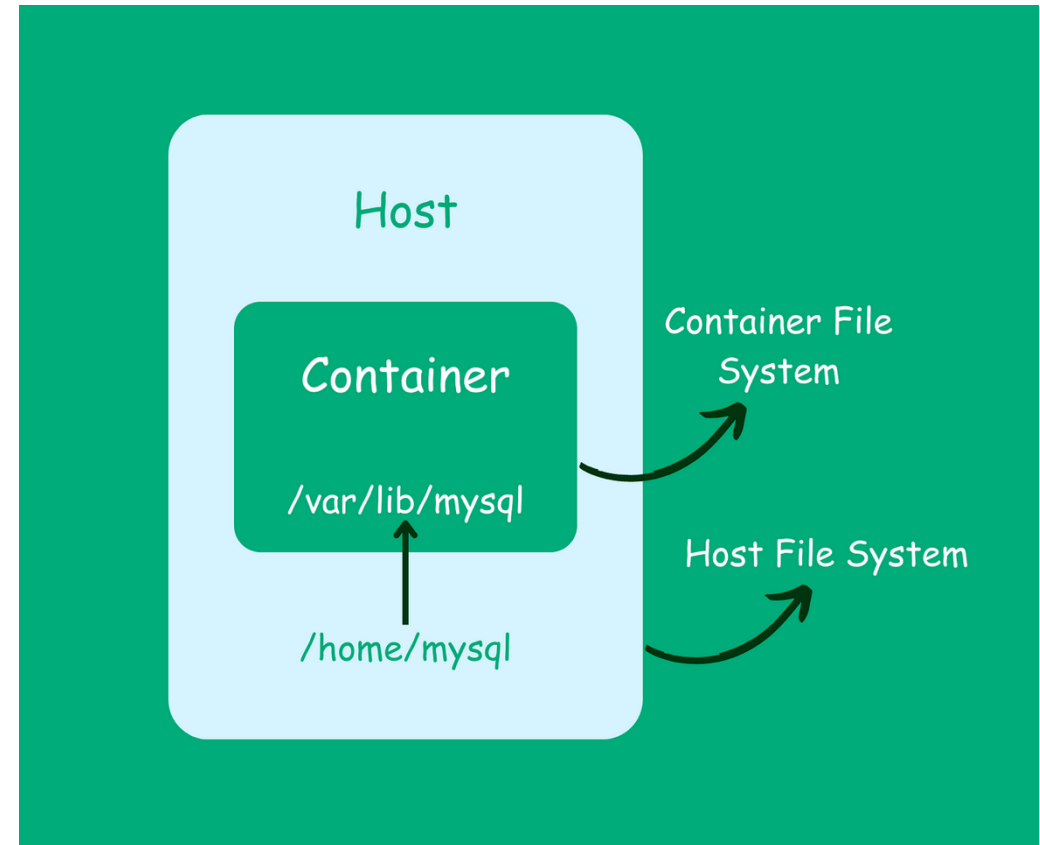
# DEFINICIÓN DE VARIABLES DEL ENTORNO

- Utiliza las instrucciones **ENV** para definir rutas de archivos y directorios en lugar de reutilizar rutas fijas en las instrucciones de Containerfile.
- Las variables del entorno son útiles para definir la configuración de la aplicación, almacenar información, como números de versión de software
- El uso de la instrucción **ARG** para establecer variables de entorno en el momento de la compilación



# DECLARACIÓN DE VOLÚMENES

- Definir la instrucción VOLUME hace que los consumidores de la imagen comprendan más fácilmente los volúmenes que pueden definir cuándo ejecutan su imagen.



# EXPOSICIÓN DE PUERTOS

- Exponga los puertos no privilegiados con la instrucción EXPOSE.
- La definición de la instrucción EXPOSE facilita que los consumidores de imágenes comprendan qué puertos expone su aplicación.
- No exponer puertos por debajo de 1024, que requieren acceso con privilegios.
- La consola web de OpenShift reconoce los puertos definidos por la instrucción EXPOSE y los muestra en la lista desplegable

**Advanced options > Target port**

Red Hat OpenShift Online

Project: ncacheproject

**Name \***  
myroute  
A unique name for the route within the project.

**Hostname**  
www.example.com  
Public hostname for the route. If not specified, a hostname is generated.

**Path**  
/  
Path that the router watches to route traffic to the service.

**Service \***  
cacheserver  
Service to route to.

**Target Port \***  
8251 -> 8251 (TCP)  
Target port for traffic.

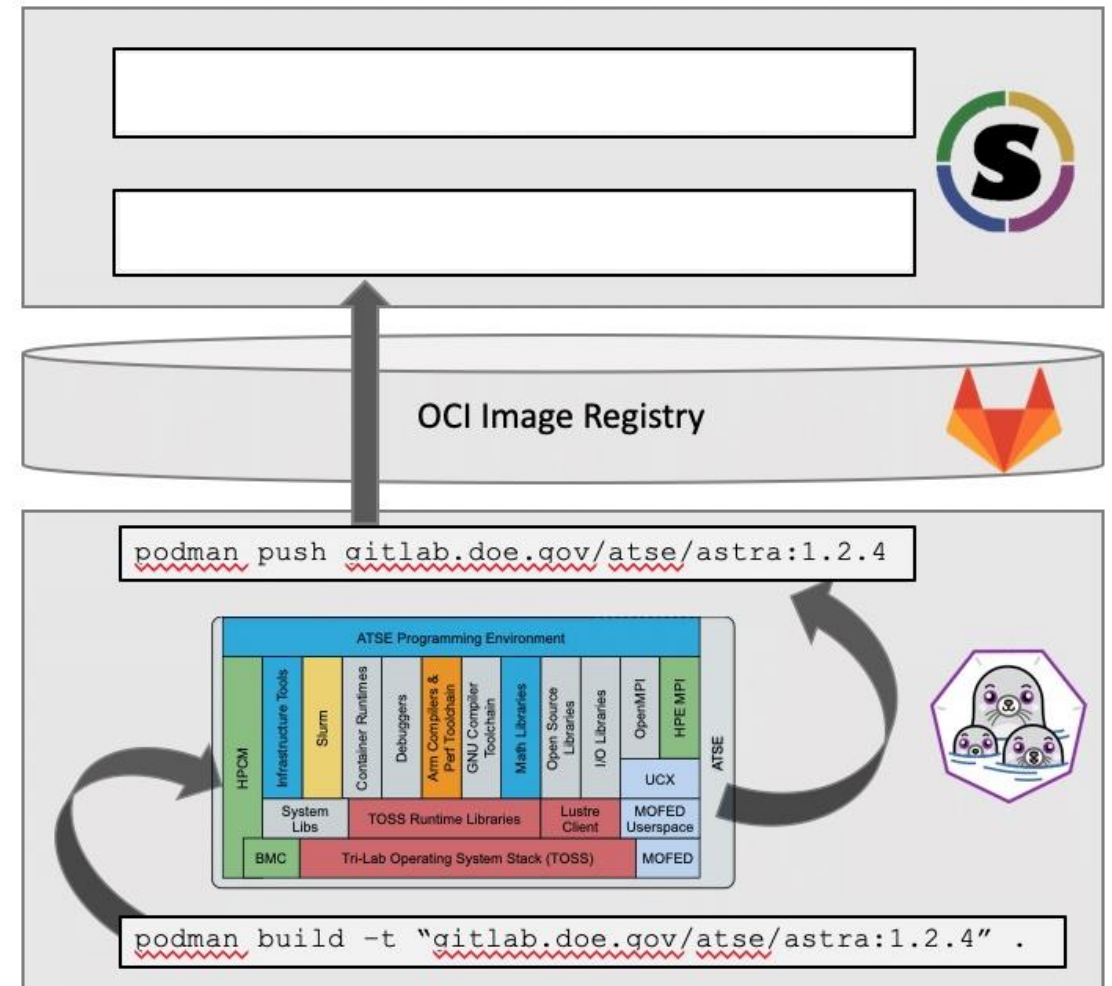
# COMPILACIÓN Y ENVÍO DE IMÁGENES CON PODMAN

- Se puede usar una herramienta, como Podman, para crear una imagen de contenedor en una ubicación local y enviarla a un registro de contenedor.
- Use el comando `podman build` para crear una imagen de contenedor a partir de un Containerfile

```
[user@host ~]$ podman build CONTEXT_DIR -t IMAGE
```

- Después de compilar la imagen del contenedor de forma local, envíe la imagen a un registro de contenedor mediante el comando `podman push`.

```
[user@host ~]$ podman push IMAGE
```



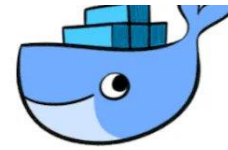
# USO DE REGISTROS EXTERNOS EN RED HAT OPENSIFT

---



# REGISTROS DE CONTENEDOR

- OpenShift, hay muchos tipos de registros de contenedor:
  - Registros públicosRegistros
  - Registros privadosLos
  - Registros
  - Registros internos de Openshift



DockerHub



Amazon ECR



DigitalOcean  
Container Registry



Azure  
Container Registry



Harbor  
Container Registry



GitLab  
Container Registry



Google  
Container Registry



IBM  
Container Registry



Alibaba Cloud  
Container Registry



JFrog  
Container Registry



Quay  
Container Registry



# AUTENTICACIÓN DE OPENSIFT CON REGISTROS PRIVADOS

- Para implementar imágenes de contenedor de esos registros, OpenShift debe estar autenticado con el registro externo.
- Para autorizar OpenShift con un registro externo, almacene las credenciales para autorizar el registro en OpenShift y asocie las credenciales con su cuenta de servicio.



# CREACIÓN DE CREDENCIALES DE REGISTRO EN OPENS SHIFT

- Puede almacenar credenciales para un registro remoto en un objeto Secret.
- Los secrets y mapas de configuración (ConfigMap) son objetos con espacios de nombres que le permiten externalizar datos de sus aplicaciones en su clúster de OpenShift.
- Puede usar el comando `oc create` para crear un secret o desde la consola web.

```
[user@host ~]$ oc create secret generic example-secret \
--from-literal=user=developer --namespace=example-ns
secret/example-secret created
```

Type	Created
kubernetes.io/dockercfg	Jul 27, 2023, 4:07 AM
kubernetes.io/service-account-token	Jul 27, 2023, 4:07 AM
kubernetes.io/dockercfg	Jul 27, 2023, 4:07 AM
kubernetes.io/service-account-token	Jul 27, 2023, 4:07 AM
kubernetes.io/dockercfg	Jul 27, 2023, 4:07 AM
kubernetes.io/service-account-token	Jul 27, 2023, 4:07 AM



# KUBERNETES SECRETS DOCKER-REGISTRY

- Kubernetes proporciona el tipo de secreto **docker-registry** para almacenar las credenciales para la autenticación con el registro del contenedor

```
[user@host ~]$ oc create secret docker-registry SECRET_NAME \
--docker-server REGISTRY_URL \
--docker-username USER \
--docker-password PASSWORD \
--docker-email=EMAIL
secret/SECRET_NAME created
```

Project: example-project ▼

## Create image pull secret

Image pull secrets let you authenticate against a private image registry.

Secret name \*

Unique name of the new secret.

Authentication type

Image registry credentials ▼

Registry server address \*

For example quay.io or docker.io

Username \*

Password \*

Email

# CREACIÓN DE IMAGE STREAMS

---



# IMAGE STREAMS RED HAT OPENSSHIFT

- Un flujo de imágenes es una colección de etiquetas de flujo de imágenes relacionadas. Para usar una secuencia de imágenes, haga referencia a una etiqueta de secuencia de imágenes.
- Esto es similar a una imagen de contenedor en un registro de imágenes.
- Pero a diferencia de las etiquetas en un registro de imágenes, las etiquetas de flujo de imágenes apuntan a la misma imagen incluso si se actualiza la etiqueta de imagen del contenedor original.

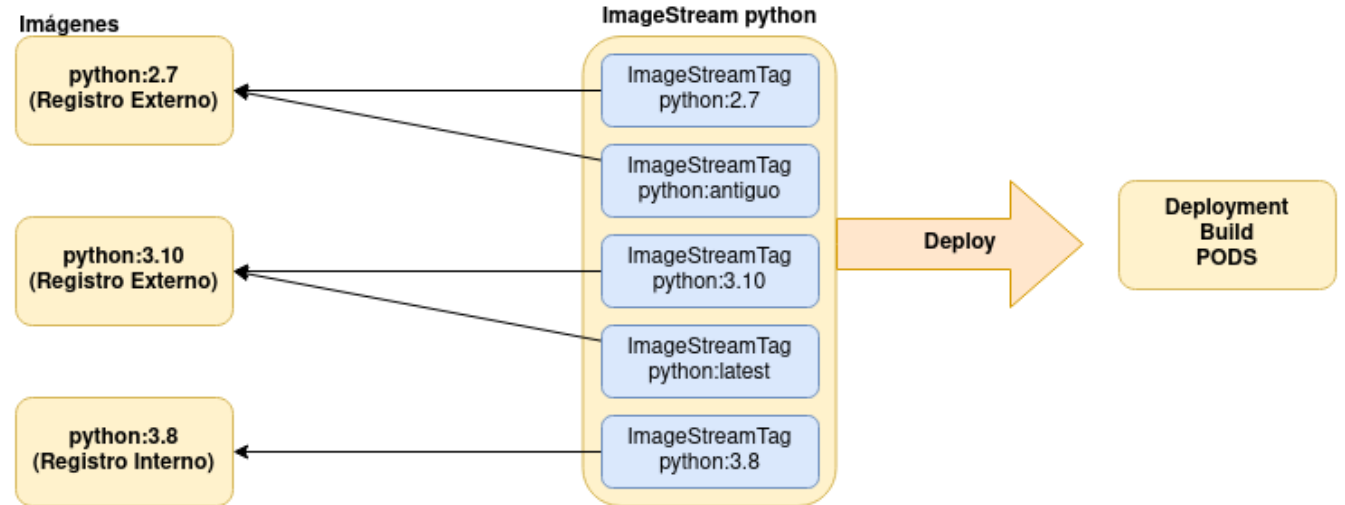
## ImageStream



# IMAGESTREAMS

- Un flujo de imágenes es una colección de etiquetas de flujo de imágenes relacionadas.
- Para usar una secuencia de imágenes, haga referencia a una etiqueta de secuencia de imágenes.
- Esto es similar a una imagen de contenedor en un registro de imágenes.
- Pero a diferencia de las etiquetas en un registro de imágenes, las etiquetas de flujo de imágenes apuntan a la misma imagen incluso si se actualiza la etiqueta de imagen del contenedor original.

## ImageStream



# NOMBRES, ETIQUETAS E ID DE IMAGEN

- Para visualizar mejor las relaciones entre tipos de recursos, el siguiente comando `oc describe is php` muestra la imagen de origen y el ID de imagen actual para cada etiqueta de flujo de imágenes:

```
[user@host ~]$ oc describe is php -n openshift
Name:                php 1
Namespace:           openshift
...output omitted...
Tags:                6
7.3 (latest) 2
  tagged from registry.redhat.io/rhsc1/php-73-rhel7:latest 3
...output omitted...
  * registry.redhat.io/rhsc1/php-73-rhel7@sha256:22ba...09b5 4
...output omitted...
7.2
  tagged from registry.redhat.io/rhsc1/php-72-rhel7:latest
...output omitted...
  * registry.redhat.io/rhsc1/php-72-rhel7@sha256:e8d6...e615
...output omitted...
```

- <sup>1</sup> El nombre del flujo de imágenes
- <sup>2</sup> El nombre de la etiqueta, que también se etiqueta como la imagen latest
- <sup>3</sup> El nombre de la imagen
- <sup>4</sup> El ID de la imagen actual para la etiqueta 7.3



# ADMINISTRACIÓN DE FLUJOS Y ETIQUETAS DE IMÁGENES

- Una etiqueta de Imagestreams realiza un seguimiento de sus últimos ID de imagen recuperados.
- Enviar una nueva imagen a un registro externo no actualiza automáticamente una etiqueta de flujo de imágenes.
- comando `import-image`, importa una imagen de contenedor `my-app-stream` desde un registro de contenedor externo y busca actualizaciones periódicamente:

```
[user@host ~]$ oc import-image myimagestream --confirm --scheduled=true \
--from example.com/example-repo/my-app-image
```

- Crear un recurso de etiqueta de flujo de imágenes para cada etiqueta de imagen de contenedor que exista en el servidor de registro de origen, agregue la opción `--all` al comando `oc import-image`

```
[user@host ~]$ oc import-image myimagestream --confirm --all \
--from registry/myorg/myimage
```



**LAB 8**

# **CREACIÓN Y PUBLICACIÓN DE IMÁGENES DE CONTENEDORES**

---