

Nombre Ruben Felipe Tovar Aviles

Fecha dia mes año

Profesor Jhon Corredor

Materia ADSO - 2900177

Institución SENA - CIES

Curso \_\_\_\_\_ Nota \_\_\_\_\_

## GUIA DE DOCKER CONSOLA

### 1. Introducción y Conceptos Fundamentales

#### ¿Qué es Docker?

Docker es una plataforma de virtualización para contenedores que permite implementar aplicaciones con todos sus dependencias (librerías, configuración, archivos) en contenedores ligeros y portátiles.

#### Ventajas

- **Ligero**: Usa el Kernel del SO
- **Portable**: "Funciona en mi máquina" → "Funciona en todos los máquinas"
- **Reproductible**: Misma Imagen = mismo comportamiento siempre
- **Encapsulado**: Fácil arquitectura y despliegue masivo

#### Arquitectura básica

Activación → Contenedor → Imagen → Docker Engine → SO Host

### 2. Conceptos Esenciales

Concepto	Definición	Analogía
Imagen	Plantilla inmutable de todo software	Mold de Pastel
Contenedor	Instancia ejecutable de una imagen	Pastel hecho del Mold
Docker Hub	Repositorio público de imágenes oficiales	App Store de imágenes
Dockerfile	Archivos de instrucciones para crear imágenes	Receta para el molde
Registry	Servidor que almacena imágenes	Almacén de moldes
Layer	Copia que compone una imagen	Ingrediente del Molde

## Estado de un contenedor

- **Created**: Creado pero no iniciado
- **Running**: En ejecución activa
- **Paused**: Pausado temporalmente
- **Stopped**: Detenido (puede reiniciarse)
- **Exited**: Terminó su ejecución

## 3. Instalación y Verificación

**Definición**: Comandos para verificar que Docker esté correctamente instalado y funcionando en el sistema, así como obtener información sobre su configuración.

### # Verificar instalación

`docker --version` # Versión de Docker

`docker version` # Información detallada cliente / servidores

`docker info` # Estado del daemon y configuración

### # Prueba inicial (ejecuta contenedor de prueba)

`docker run hello-world` # Test básico de funcionamiento

## 4. Gestión de imágenes

**Definición**: Comandos para buscar, descargar, listar, inspeccionar y eliminar imágenes Docker. Son imágenes con plantillas inmutables que contienen el código, runtime, herramientas y configuraciones necesarias para ejecutar una aplicación.

### Buscar y descargar imágenes

#### # Buscar imágenes en Docker Hub

`docker search ubuntu` # Buscar imágenes de ubuntu

`docker search --limit 5 nginx` # Limitar resultado a 5

### Descargar imágenes

`docker pull ubuntu` # Descargar imagen oficial de ubuntu

`docker pull ubuntu:20.04` # Descargar versión específica

`docker pull nginx:alpine` # Descargar nginx con distribución alpine

## Leyton y gestionar imágenes locales

### # Tareas Imágenes

docker images

# Listar las imágenes locales

docker images ubuntu

# Solo imágenes de ubuntu

docker images --all

# Incluir imágenes intermedias

docker images --filter "dangling=true" # Solo imágenes sin nombre

### # Información detallada

docker inspect ubuntu:20.04 # Metadatos completos de la imagen

docker history ubuntu:20.04 # Historial de capas de la imagen

### # Eliminar Imágenes

docker rmi ubuntu

# Por nombre

docker rmi 3b418d7

# Por ID

docker rmi ubuntu:20.04 # Por tag específico

## 5. Gestión de Contenedores

Definición: Comandos para crear, ejecutar, controlar y gestionar contenedores Docker. Los contenedores son instancias en ejecución de las imágenes que ejecutan aplicaciones de forma aislada del sistema host.

### Crear y ejecutar contenedores

#### # Comandos Básicos de ejecución

docker run ubuntu

# Ejecutar y salir inmediatamente

docker run -it

# Modo interactivo con bash

docker run -d nginx

# Ejecutar en segundo plano (daemon)

#### # Opciones comunes de ejecución

docker run --name mi-web nginx

# Con nombre personalizado

docker run -P 8080:80 nginx

# Mapa puerto host: container

docker run -p 127.0.0.1:8080:80 nginx

# Solo en localhost

docker run -v /host/path:/container/path nginx

# Montar volumen

docker run --env VAR=Value nginx

# Variables de entorno

docker run --rm nginx

# Auto-elimina al parar

## Listar y montar los contenedores

### # Listar contenedores

docker ps # Sólo contenedores activos

docker ps -a # Toda los contenedores (activos e inactivos)

docker ps -l # Ultimo container creado

docker ps --format " " # Formato personalizado

# Monitoreo en tiempo real

docker stats # Uso de recursos de todos los contenedores

docker stats mi-contenedor # Uso de recursos de un contenedor

docker top mi-contenedor # Procesos ejecutándose dentro del contenedor

## Controlar Contenedores

### # Iniciar, Parar, Reiniciar

docker start mi-contenedor # Iniciar contenedor parado

docker stop mi-contenedor # Parar contenedor ejecutando

docker restart mi-contenedor # Reiniciar contenedor

docker kill mi-contenedor # Finalizar proceso inmediatamente

docker pause mi-contenedor # Pausar todo los procesos

docker unpause mi-contenedor # Resumir proceso pausado

### # Interactuar con contenedores en ejecución

docker exec -it mi-contenedor bash # Shell shell en container vivo

docker exec mi-contenedor ls -la /app # Ejecutar comando remota

docker attach mi-contenedor # Conectarse a proceso principal

### # Eliminar contenedores

docker rm mi-contenedor # Eliminar contenedor parado

docker rm -f mi-contenedor # Forzar eliminación (cuando esté activo)

docker rm \$(docker ps -q) # Eliminar todos los contenedores

### # Limpieza Automática

docker container prune # Eliminar todos los contenedores parados

docker container prune --filter "until=24h" # Podría hacer más difícil

Nombre	Fecha dia mes año
Profesor	Materia
Institución	Curso Nota

**G. Logs e Inspección**

**Definición:** Comando para monitorear, diagnosticar y obtener información detallada sobre el comportamiento y estado de los contenedores en ejecución o los que han terminado.

# Ver logs de contenedores

```
docker logs mi-contenedor # Ver todos los logs
docker logs -f mi-contenedor # Seguir logs en tiempo real (follow)
docker logs --tail 50 mi-contenedor # Últimos 50 líneas
docker logs --since="2023-01-01" mi-contenedor # Desde fecha específica
```

# Inspección detallada

```
docker inspect mi-contenedor # Metadatos completo (JSON)
docker port mi-contenedor # Mapas de puerto
docker diff mi-contenedor # Cambios en el contenido de archivos
```

**F. Construcción de imágenes personalizadas**

**Definición:** Comando que tiene para crear imágenes Docker personalizadas usando Dockerfile. Esto permite especificar opciones específicas con sus dependencias y configuraciones particulares.

Dockerfile básico

# Imagen base

```
FROM ubuntu:20.04
```

# Metadatos

```
LABEL maintainer = "Tu-email@"
LABEL version = "1.0"
```

# Variables de Entorno

```
ENV APP_HOME=/app
ENV PYTHON_VERSION=3.8
```

## # Instalar dependencias

```
RUN apt-get update && apt-get install -y \
    python3 \
    python3-pip \
&& rm -rf /var/lib/apt/lists/*
```

## # Crear directorio de trabajo

```
WORKDIR $APP_HOME
```

## # Copiar archivos

```
COPY requirement.txt.
```

```
COPY src/.isrc/
```

## # Instalar dependencias de Python

```
RUN pip3 install -r requirements.txt
```

## # Exponer puerto

```
EXPOSE 8080
```

## # Crear user root (Required)

```
RUN useradd -m appuser
```

```
USER appuser
```

## # Comando por defecto

```
CMD ["python3", "src/appy.py"]
```

## Comandos de Construcción

### # Continuar imagen desde Dockerfile

```
docker build -t mi-app:1.0.
```

# Con Tag específico

```
docker build -t mi-app:latest.
```

# Tag por defecto

```
docker build -f Dockerfile.dev -t mi-app .
```

# Dockerfile específico

```
docker build --no-cache -t mi-app.
```

# Sin usar cache

```
docker build -b build -a my VERSION=1.0 -t mi-app .
```

# Con argumentos

## # Etiquetar Imagenes

```
docker tag mi-app:1.0 mi-app:latest # Crear Tag adicional
```

```
docker tag mi-app:1.0 mi-app
```

## 8. Redes y Volumenes

**Definición:** Comandos para gestionar la comunicación entre contenedores (redes) y los persistentes de datos (volumenes). Las redes permiten que contenedores se comuniquen entre sí, mientras que los volumenes guardan datos más allá del ciclo de vida del contenedor.

### Gestión de redes

#### # Típico y crear redes

```
docker network ls
```

# Listar redes disponibles

```
docker network create mi-red
```

# Crear red personalizada

```
docker network create --driver bridge mi-red-bridge
```

# Ejecutar comando

#### # Conectar contenedores de redes

```
docker run -d --name web --network mi-red nginx
```

```
docker network connect mi-red mi-con-exis
```

```
docker network disconnect mi-red mi-con
```

#### # Inspeccionar redes

```
docker network inspect mi-red
```

# Detalles de la red

### Gestión de Volumenes

#### # Crear y listar Volumenes

```
docker volume create mi-volumen
```

# Crear volumen nombreable

```
docker volume ls
```

# Listar volumenes

```
docker volume inspect mi-volumen
```

# Inspección volumen

#### # Unir Volumenes en Contenedores

```
docker run -v mi-volumen:/data nginx
```

# Volumen nombreable

```
docker run -v /host/path:/container/path nginx
```

# Bind mount

```
docker run --mount source=mi-volumen target=/data
```

# Síntaxis mount

#### # Eliminar Volumenes

```
docker volume rm mi-volumen
```

# Eliminar volumen específico

```
docker volume prune
```

# Eliminar volumenes sin usar

## 10. Limpieza y Mantenimiento

Definición:

## 10. Buena Práctica y Consejos

Seguridad

- No ejecutar como root: Crear un usuario dentro del contenedor
- Usar imágenes oficiales: Muy seguras y mantenibles
- Evitar Vulnerabilidades: docker scan mi-image
- Separar entornos: No mezclar contenidos Dockerfile

Optimización

- Usar .dockerrun.sh: Excluir archivos innecesarios
- Imágenes multi-stage: Reducir el tamaño final
- Limpiar en lo mínimo copia:
- Usar tags específicos: Evitar latest en producción

Desarrollo

- Un proceso por contenedor: Principio de responsabilidad única
- Contenedores efimeros: Nunca para ser fácilmente reutilizables
- Logear a stdart/stderr: Para mejores observabilidades
- Health check: Nuestra verificación de disponibilidad