Guías

Pasos detallados para montar cosas desde cero. Cuando necesito recordar cómo hice algo sin volver a improvisar.

- Crea tu página web desde cero en Docker
- Dockcheck: Automatiza las actualizaciones de Docker
- Instalación de NVIDIA Container Toolkit para su uso en Docker
- Instalación y primeros pasos con Docker

Crea tu página web desde cero en Docker

Vamos a montar una página web desde cero utilizando Docker, combinando un archivo `docker-compose.yml`, un `Dockerfile`, y una configuración personalizada de Nginx.

Requisitos previos

Antes de empezar, asegúrate de tener instalados los siguientes programas:

- **Docker**: para crear y gestionar contenedores.
- **Docker Compose**: para orquestar tus servicios.

¿Cómo instalarlos?

Puedes consultar la documentación oficial de Docker para guías detalladas de instalación:

- Instalar Docker
- Instalar Docker Compose

Si tu sistema está basado en **APT** (como Debian o Ubuntu), también puedes seguir este tutorial paso a paso:

Instalación y primeros pasos con Docker

También necesitarás los archivos de tu página web en un directorio local.

Ventajas de dockerizar tu página web

Dockerizar una página web ofrece una serie de beneficios importantes:

1. **Portabilidad**
   Tu aplicación funcionará de la misma manera en cualquier sistema que tenga Docker instalado, independientemente de las diferencias en el entorno.

2. **Consistencia entre entornos**
   Evita problemas típicos de configuración al garantizar que el entorno de desarrollo, pruebas y producción sean idénticos.
3. **Fácil despliegue**
   Con un simple comando (`docker compose up -d --build`), puedes levantar la infraestructura necesaria para tu aplicación sin complicaciones.
4. **Aislamiento**
   Cada contenedor opera de forma independiente, evitando conflictos con otros servicios o aplicaciones en el mismo sistema.
5. **Ligereza**
   Los contenedores consumen menos recursos que las máquinas virtuales, ya que comparten el kernel del sistema operativo.
6. **Escalabilidad**
   Permite escalar horizontalmente al replicar contenedores según las necesidades de tráfico.
7. **Gestión de versiones**
   Facilita la gestión de versiones al permitir etiquetar diferentes estados de la aplicación y realizar rollbacks en caso necesario.
8. **Integración con CI/CD**
   Docker se integra fácilmente con herramientas de integración continua, automatizando pruebas y despliegues.
9. **Simplificación del manejo de dependencias**
   No necesitas instalar dependencias o configuraciones específicas en tu máquina local; todo está definido en el `Dockerfile`.
10. **Facilidad para pruebas locales**
    Permite realizar pruebas en un entorno controlado sin afectar otros servicios.
11. **Ecosistema rico**
    Docker ofrece acceso a imágenes preconstruidas en Docker Hub, como servidores web o bases de datos, para agilizar configuraciones.
12. **Resiliencia**
    Configuraciones como `restart: unless-stopped` aseguran que el contenedor se reinicie automáticamente en caso de fallos.
13. **Compatibilidad futura**
    Docker es una tecnología moderna y ampliamente adoptada, asegurando relevancia a largo plazo.

Paso 1: Preparar el archivo `docker-compose.yml`

Este archivo define cómo se orquestará tu contenedor web. Aquí tienes un ejemplo:

```
services:
  web: # Define un servicio llamado "web". Este será el contenedor que ejecutará tu página web.
    build: . # Indica que se construirá una imagen Docker utilizando el Dockerfile que está en el directorio actual.
    network_mode: bridge # Configura el contenedor para usar el modo de red "bridge". Este es el modo predeterminado, que aísla la red del contenedor y la conecta a través de un puente virtual al host.
    volumes: # Define volúmenes que conectan el sistema de archivos del host con el contenedor.
      - ./website:/usr/share/nginx/html # Monta la carpeta local "./website" (donde está tu página web) en "/usr/share/nginx/html" dentro del contenedor, para que Nginx pueda servir los archivos.
      - /etc/localtime:/etc/localtime:ro # Sincroniza la hora local del host con el contenedor para que compartan la misma zona horaria. Se monta en modo "ro" (solo lectura) para evitar cambios accidentales.
    ports:
      - "8080:80" # Mapea el puerto 80 del contenedor (donde escucha Nginx) al puerto 8080 del host, haciendo accesible la aplicación en "http://localhost:8080".
    restart: unless-stopped # Configura el contenedor para reiniciarse automáticamente a menos que sea detenido manualmente. Esto asegura alta disponibilidad en caso de reinicios o fallos.
    container_name: my_web_service # Asigna un nombre personalizado al contenedor, facilitando su identificación en comandos como "docker ps".
    environment: # Define variables de entorno que se pasarán al contenedor.
      - TZ=Europe/Madrid # Configura la zona horaria dentro del contenedor para que coincida con la local (Europa/Madrid en este caso).
```

Paso 2: Crear el `Dockerfile`

El `Dockerfile` define cómo se construirá la imagen de tu contenedor. Aquí tienes un ejemplo:

```
# Usa una imagen base ligera de servidor web
FROM nginx:alpine

# Copia los archivos de la página al directorio de Nginx
COPY ./website /usr/share/nginx/html
```

```
# Copia el archivo de configuración personalizado de Nginx
COPY nginx.conf /etc/nginx/nginx.conf

# Exponer el puerto 80 para que se pueda acceder desde fuera del contenedor
EXPOSE 80
```

Paso 3: Configurar Nginx

Aquí tienes un ejemplo de un archivo `nginx.conf` personalizado, adaptado para este proyecto:

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    include       mime.types;
    default_type  application/octet-stream;

    client_max_body_size 10M;

    limit_req_zone $binary_remote_addr zone=mylimit:10m rate=20r/m;

    server {
        listen 80;
        server_name localhost;

        root /usr/share/nginx/html;
        index index.html;

        error_page 400 401 403 404 500 502 503 504 /error.html;
        location = /error.html {
            root /usr/share/nginx/html;
            internal;
        }

        location / {
```

```
            limit_req zone=mylimit burst=5 nodelay;
            try_files $uri $uri/ =404;
        }

        location /resources/ {
            limit_req zone=mylimit burst=20 delay=5;
            try_files $uri $uri/ =404;
        }

        location ~ /\.ht { deny all; }
        location ~* \.(conf|ini|log|yml|)$ { deny all; }
        location ~ /\. { deny all; }
        location ~* /(admin|dashboard) { deny all; }
    }
}
```

Paso 4: Levantar tu entorno

Con los archivos listos (`docker-compose.yml`, `Dockerfile`, `nginx.conf`, y tus archivos web en el directorio `website`), ejecuta el siguiente comando:

```
docker compose up -d --build
```

Este comando construye la imagen, levanta el contenedor en segundo plano y actualiza el entorno si ya estaba corriendo.

Exponer tu página al exterior

Si deseas que tu página sea accesible desde Internet, puedes exponerla utilizando un proxy inverso como Caddy. Sigue esta guía de Caddy para gestionar dominios de forma sencilla, con soporte para HTTPS incluido.

Verificar el contenedor

Para asegurarte de que todo funciona correctamente, ejecuta:

```
docker ps
```

Deberías ver tu contenedor `my_web_service` en ejecución, con el puerto 8080 expuesto.

Notas finales

Este setup es ideal para entornos de desarrollo. Si planeas usarlo en producción, considera:

- Utilizar HTTPS para conexiones seguras.
- Añadir reglas de seguridad avanzadas en Nginx.
- Configurar una solución para gestionar los logs de tu contenedor.

Dockcheck: Automatiza las actualizaciones de Docker

Dockcheck es una herramienta ligera que revisa y actualiza imágenes de Docker de manera automática. Su uso evita la acumulación de imágenes obsoletas y garantiza que los contenedores se mantengan al día sin intervención manual.

Instalación

1. Preparar el entorno

Antes de instalar Dockcheck, asegúrate de que el directorio `~/.local/bin` existe. Si no es así, créalo con el siguiente comando:

```
mkdir -p ~/.local/bin
```

---

2. Descargar Dockcheck

Tienes dos opciones para descargar el script: `wget` o `curl`.

Opción 1: Usando `wget` (recomendada)

```
wget -O ~/.local/bin/dockcheck.sh
"https://raw.githubusercontent.com/mag37/dockcheck/main/dockcheck.sh" && chmod +x
~/.local/bin/dockcheck.sh
```

Opción 2: Usando `curl`

```
curl -L https://raw.githubusercontent.com/mag37/dockcheck/main/dockcheck.sh -o
~/.local/bin/dockcheck.sh
chmod +x ~/.local/bin/dockcheck.sh
```

---

3. Ejecutar Dockcheck

Para ejecutar el script, usa:

```
~/.local/bin/dockcheck.sh
```

Si deseas ejecutarlo sin especificar la ruta completa, añade `~/.local/bin` a tu `$PATH`:

1. Abre tu archivo `~/.bashrc` o `~/.zshrc` con un editor de texto:
   ```
   nano ~/.bashrc
   ```
2. Agrega la siguiente línea al final del archivo:
   ```
   export PATH="$HOME/.local/bin:$PATH"
   ```
3. Aplica los cambios ejecutando:
   ```
   source ~/.bashrc
   ```

Ahora, simplemente puedes ejecutar Dockcheck con:

```
dockcheck.sh
```

Uso básico

1. Verificar actualizaciones

Para comprobar si hay actualizaciones disponibles para los contenedores de Docker, usa:

```
dockcheck.sh
```

Esto listará los contenedores con nuevas versiones disponibles.

---

2. Actualizar contenedores automáticamente

Si quieres actualizar todos los contenedores automáticamente, ejecuta:

```
dockcheck.sh -p -a
```

Esto realizará las siguientes acciones:

- Descargar las versiones más recientes de las imágenes.
- Eliminar las imágenes antiguas.

- Reiniciar los contenedores con las nuevas versiones.

---

3. Opciones avanzadas

- **Excluir contenedores específicos**: Para omitir ciertos contenedores en la actualización, usa:

  ```
  dockcheck.sh --exclude nombre_del_contenedor
  ```

- **Ver la ayuda completa**: Muestra todas las opciones disponibles con:

  ```
  dockcheck.sh -h
  ```

Repositorio oficial

Para más información, visita el repositorio oficial de Dockcheck.

Instalación de NVIDIA Container Toolkit para su uso en Docker

---

Integrar GPU NVIDIA con Docker (NVIDIA Container Toolkit 1.17.0)

Integrar GPUs NVIDIA con Docker permite ejecutar aplicaciones aceleradas por GPU dentro de contenedores, ideal para tareas como aprendizaje profundo y cómputo intensivo. Aquí se detalla cómo instalar y configurar el **NVIDIA Container Toolkit 1.17.0** en sistemas Ubuntu/Debian.

Requisitos previos

Hardware

- **GPU NVIDIA:** Compatible con CUDA. Revisa la lista oficial.
- **Memoria suficiente:** Asegúrate de que la GPU tenga memoria para las tareas que quieres correr.

Software

1. **Sistema operativo:** Ubuntu 18.04/20.04/22.04 o Debian Stretch/Buster.
2. **Controladores NVIDIA instalados:** Verifica con:

   ```
   nvidia-smi
   ```

3. **Docker instalado:** Comprueba con:

   ```
   docker --version
   ```

   Si no lo tienes, sigue la documentación oficial.

4. **CUDA Toolkit (opcional):** Solo si vas a ejecutar código CUDA fuera de Docker:

```
nvcc --version
```

Instalación del NVIDIA Container Toolkit

1. Configurar el repositorio de NVIDIA

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
  && curl -fsSL https://nvidia.github.io/libnvidia-container/gpgkey | sudo gpg --dearmor -o /usr/share/keyrings/nvidia-container-toolkit-keyring.gpg \
  && curl -s -L https://nvidia.github.io/libnvidia-container/stable/$distribution/libnvidia-container.list | \
    sed 's#deb https://#deb [signed-by=/usr/share/keyrings/nvidia-container-toolkit-keyring.gpg] https://#g' | \
    sudo tee /etc/apt/sources.list.d/nvidia-container-toolkit.list
```

Esto añade el repositorio y la clave GPG necesaria.

2. Actualizar repositorios

```
sudo apt-get update
```

3. Instalar el toolkit

```
sudo apt-get install -y nvidia-container-toolkit
```

4. Configurar Docker para usar el runtime de NVIDIA

```
sudo nvidia-ctk runtime configure --runtime=docker
```

Este comando edita automáticamente `/etc/docker/daemon.json`.

5. Reiniciar Docker

```
sudo systemctl restart docker
```

Verificar la instalación

Contenedor de prueba

```
sudo docker run --rm --gpus all nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

Debe mostrar el estado y detalles de tu GPU.

Solución de problemas

Problemas comunes

- `nvidia-smi` **no funciona en el host:** Revisa los controladores.
- **Docker no detecta GPU:** Revisa configuración del runtime.
- **Errores en el contenedor:** Asegúrate de que el toolkit esté instalado correctamente.

Referencias

- Guía oficial de instalación de NVIDIA Container Toolkit 1.17.0

Instalación y primeros pasos con Docker

Docker es una herramienta potente para crear, desplegar y gestionar contenedores. Esta guía resume los pasos para instalarlo desde repositorio oficial en Ubuntu.

1. Actualizar el sistema

```
sudo apt update && sudo apt upgrade -y
```

2. Instalar dependencias necesarias

```
sudo apt install -y apt-transport-https ca-certificates curl software-properties-common
```

3. Agregar la clave GPG de Docker

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Agregar el repositorio de Docker

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Instalar Docker

```
sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io
```

---

6. Verificar la instalación

```
docker --version
```

---

7. Habilitar e iniciar el servicio

```
sudo systemctl enable docker
sudo systemctl start docker
```

---

8. Ejecutar Docker sin sudo

```
sudo usermod -aG docker $USER
```

> ℹ Cierra sesión y vuelve a entrar para aplicar el cambio de grupo.

---

9. Probar instalación

```
docker run hello-world
```

Este comando descarga una imagen de prueba y muestra un mensaje si todo está funcionando.

Más información

- Documentación oficial de Docker