Guía Docker Swarm, Compose y Dev Containers

1. Iniciar un Swarm

- docker swarm init: Convierte el nodo actual en 'manager'.
- docker swarm join-token worker: Muestra el comando para que otro nodo se una.

2. Crear y gestionar servicios

- docker service create --name web --replicas 3 -p 80:80 nginx
- docker service scale web=5
- docker service ps web
- docker service rm web

Ejemplo: Desplegar un NGINX replicado 3 veces:

```
docker service create --name web --replicas 3 -p 80:80 nginx
```

3. Docker Compose en modo Swarm

- docker stack deploy -c docker-compose.yml mi_stack
- docker stack ls
- docker stack services mi_stack
- docker stack rm mi_stack

4. Monitoreo y logs

- docker service logs web
- docker container ls
- docker container logs <nombre | ID>

5. Conceptos clave

- Swarm: Orquestador de múltiples nodos.
- **Servicio:** Grupo de tareas gestionadas.
- Stack: Colección de servicios definidos en Compose.
- Task: Contenedor individual dentro de un servicio.

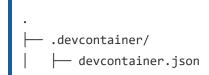
6. Buenas prácticas

- Usa nombres descriptivos.
- Evita docker-compose up en producción.
- Valida tus archivos con docker-compose config.

7. Dev Containers

Permiten definir entornos de desarrollo listos para trabajar con VS Code.

Estructura típica:



Dockerfile de ejemplo:

```
FROM ubuntu:20.04

RUN apt-get update && apt-get install -y \
    curl \
    git \
    vim \
    python3-pip
```

devcontainer.json de ejemplo:

```
"name": "Mi entorno Dev",
"build": {
    "dockerfile": "Dockerfile"
},
"settings": {
    "terminal.integrated.shell.linux": "/bin/bash"
},
"extensions": [
    "ms-python.python",
    "esbenp.prettier-vscode"
],
"forwardPorts": [8000],
"postCreateCommand": "pip3 install -r requirements.txt"
```

}

Cómo usarlo: Abre el proyecto en VS Code y selecciona *Reopen in Container*.