

• DESPLIEGUE

Instalación
Activación
Actualización
Eliminación

Componentes \Rightarrow Software preparado para su uso

• TENER EN CUENTA

Componentes heterogeneos en red

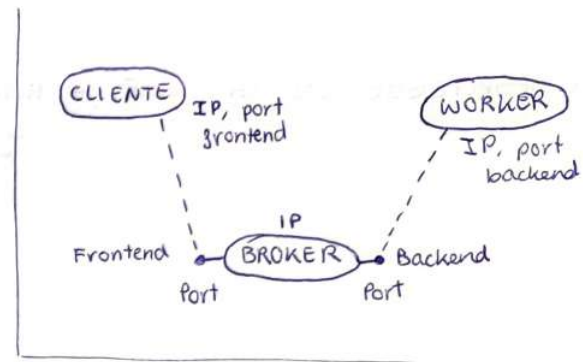
Deben cooperar \rightarrow dependencias

Nodos heterogeneos \rightarrow requisitos

Seguridad

creados por desarrolladores distintos

cambiantes



• EJEMPLO \rightarrow PATRÓN BROKER

Cliente - Broker - Worker \Rightarrow autónomos

Varias instancias de cada uno \Rightarrow independientes

CLIENTE \leftarrow ubicación del broker
IP y puerto frontend

WORKER \leftarrow ubicación del broker
IP y puerto backend

Posee una id única
cada instancia

TEMA 4

• PROBLEMA \rightarrow

Despliegue manual no es viable para un servicio grande real

• DESPLIEGUE

Solución

AUTOMATIZADO

PLAN DE
DESPLIEGUE

Aplicación + Despliegue = SERVICIO

Establece un SLA (Service Legal Agreement)

Función

Rendimiento

Disponibilidad

Desplegar un servicio \Rightarrow INSTALACIÓN, ACTIVACIÓN, ACTUALIZACIÓN, ADAPTACIÓN

① CONFIGURACIÓN PARA
CADA COMPONENTE

Fichero con la configuración + descripción de dependencias

La herramienta genera una config. específica para cada instancia del componente

(worker, broker...)

② PLAN DE CONFIGURACIÓN
GLOBAL

conexión entre componentes

Donde colocar cada instancia

Enlace (bindings) de dependencias, endpoints

¿Cómo se
resuelven?

1. Mediante el código

Bajo nivel

Propenso a errores

2. Inyección de dependencias

La aplicación define qué recursos
necesita pero no cómo obtenerlos

El contenedor se encargará

o DESPLIEGUE EN LA NUBE → Hemos creado y seleccionado componentes → ¿Proveedor? IaaS

IaaS

- Infraestructura (Hardware)
- Virtualización (Máquinas Virtuales) → Flexibilidad asignación recursos
- Limitaciones → Decisiones asignación bajo nivel → específica 100% todo
 - No permite elegir características
 - Baja recuperación tras fallo

PaaS

- Plataforma
- SLA como elemento central → Idemneo → Automatización del Despliegue
- Automatización / Configuración solucionadas
- Establece automáticamente los nodos necesarios

o CONTENEDORES

- Simplifica el enlace entre dependencias y endpoints
- Aprovisionamiento → Reserva infraestructura para una apl. distribuida
- Inconvenientes
 - Menor flexibilidad → Compatible con el SO anfitrión
 - Menor protección
- Ventajas
 - Más ligero → Menos recursos
 - Más rápido
 - Más fácil de instalar
- Recursos de intercomunicación
- Recursos para cada instancia
 - Máquina Virtual
 - SO + Bibliotecas
 - Contenedor
 - El SO del anfitrión

DOCKER

→ Automatiza el despliegue de cada instancia Inicio
Parada
Define un sistema de ficheros nativo, para compatir cosas entre contenedores Eliminación ...

• DOCKER

ORDENES
CONSOLA

IMAGEN { Plantilla solo lectura
Conjunto instrucciones para crear un contenedor } En el depósito existen imágenes predefinidas
Nueva imagen = imagen base + instrucciones

CONTENEDOR

Quando ejecutamos el software descrito en la imagen
Recursos que una instancia necesita para ejecutarse
Máquina independiente con su propio contexto
← ejecutables
bibliotecas
aplicaciones

DEPÓSITO { Donde se almacenan las imágenes (local / nube)
Para compartir imágenes

Estructura → docker acción opciones argumentos

CREAR CONTENEDOR → { docker run opciones imagen progInicial
docker run -i -t centos bash

-i -t → modo interactivo con la consola
Descarga la imagen 'centos', crea el contenedor, reserva el sistema de archivos ... y ejecuta bash

CREAR NUEVA IMAGEN → { con la consola podemos modificar el contenedor
* Guardar cambios docker commit nombreContenedor nombreImagen

• CREAR NUEVAS IMÁGENES

La nueva imagen debe incluir → bibliotecas + intérprete + programa a ejecutar
Formas: de imagen base en disco

- Interactivamente *

- DockerFile → Archivo de configuración

En la 1ª línea se especifica QUE IMAGEN base se va a modificar (FROM image)
Seguido de las mismas líneas de la consola
Preparado → docker build -t tsr-zmq
-Id del contenedor → docker ps -a

• FICHERO DOCKERFILE

FROM : Nombre de la imagen base a modificar, si no es ninguna → 'latest'

RUN orden : Ejecuta dicha orden en el shell

ADD origen destino : Copia ficheros del host al contenedor, add descomprime, sino se usa copy

EXPOSE puerto : Indica el puerto en el que el contenedor atenderá peticiones

ENV variable valor : Establece una variable de entorno accesible por los programas del contenedor

CMD orden arg1 arg2 : Valores por defecto para la ejecución del contenedor

• **ENTRYPOINT** orden arg1 arg2 : Ejecuta la orden al crear el contenedor

Solo habrá una orden **CMD** o **ENTRYPOINT** → sirven para lanzar el programa

WORKDIR path : Directorio de trabajo para las instrucciones **RUN**, **CMD**, **ENTRYPOINT** etc.

• MULTIPLES COMPONENTES

Queremos lanzar varios componentes (worker, broker, clientes) → hay que iniciar 1º broker
→ obtener ID y puertos → modificar Dockerfiles de w y c → COSTOSO

SOLUCIÓN → DOCKER COMPOSE → Crea un plan de trabajo que describe componentes y relaciones
Las dependencias se resuelven en ejecución

DOCKER COMPOSE

Resuelve las dependencias de componentes externos.

↓
DESPLIEGUE AUTOMATIZADO

↓
Docker-Files Configurables

BROKER

```
FROM ....
COPY ....
EXPOSE 9999 9998
CMD node broker
```

WORKER

```
FROM ...
COPY ...
CMD node worker $BROKER_URL
```

CLIENTE

```
FROM ...
COPY ...
CMD node client $BROKER_URL
```

↓
Variable de entorno, se le da valor en el DockerFile

DOCKER-FILE

→ se lanza con → **docker-compose up -d**
docker-compose up -d --scale X=n // para lanzar 'n' instancias
cli / wor

cli :

image : client → que imagen necesita

build : ./client/ → si la imagen NO existe se construye

links :

- bro → depende del broker, 1º se lanza bro, luego cli y wor, es igual a la IP

environment

- BROKER_URL = ...

↑
valor de la variable de entorno

bro :

image : broker

build : ./broker/

expose :

- "9999" } en que puertos
- "9998" } está escuchando

• MÚLTIPLES COMPONENTES EN DISTINTOS NODOS

Utilizan Pods o clústers

Kubernetes → distribuir instancias entre distintos nodos, nada que ver con Docker

Distribuidor de contenedores

Se compone de nodos virtuales o físicos + 'pods' ← pequeñas unidades para el despliegue

• A TENER EN CUENTA

EXPOSE port :

↑
SI + de uno igual
Indica que un contenedor escuchará en un puerto específico. Aunque varios contenedores se lancen en el mismo nodo, cada uno tiene sus puertos. Si varios lanzan 'EXPOSE 8000', cada uno tiene su propio puerto 8000.

PORTS port :

↑
NO + de uno igual
Conecta los puertos en los que un contenedor está escuchando a un puerto real de la máquina. Aquí NO puede usar dos veces el mismo puerto. Ya que solo hay un puerto real.