Universidad de Oviedo

EPI GIJÓN

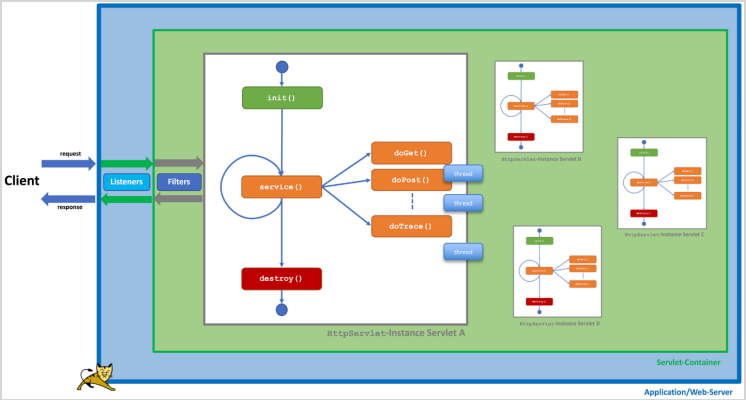
Universidad de Oviedo

Despliegue de aplicaciones web

Enol García González
Universidad de Oviedo
13 de diciembre de 2023

Contenidos

Servidor de aplicaciones



Modelos de servidor

- Servidor independiente del proyecto.
- Servidor incluido en el proyecto.

Modelos de servidor

- Servidor independiente del proyecto.
  - Este es el tipo que utilizamos en la asignatura.
  - El servidor es un producto software independiente del framework.
  - El proyecto con el framework se exporta de forma que se le puede dar como un fichero al servidor.
  - El servidor interpreta los ficheros y los sirve.
  - Principalmente utilizado para HTML+PHP (Apache, Nginx) y Java (JBoss, WildFly, Tomcat).

Modelos de servidor

- Servidor incluido en el proyecto.
  - El proyecto crea un archivo ejecutable que contiene todos los datos del proyecto y además tiene un servidor integrado.
  - El framework integra toda la gestión que tiene que hacer un servidor de aplicaciones.
  - En estos casos el ejecutable del proyecto se incluye como un servicio en la maquina para que se pueda programar su arranque automático y se gestione la ejecución en segundo plano.
  - Principalmente utilizado por NodeJS y Python.

Modelos de despliegue

- Barebone.
- Máquina virtual.
- Contenedor.

Modelos de despliegue

- Barebone.
  - Es el modelo más tradicional.
  - Se instala un sistema operativo y el servidor de aplicaciones directamente sobre la máquina física.
  - Este puede tomar todos sus recursos.
  - Problema: Normalmente los servidores web necesitan muy pocos recursos, pero los servidores físicos traen mucha más capacidad. Se desperdician recursos

Modelos de despliegue

- Máquina virtual.
  - Una evolución para resolver el problema del malgasto de recursos.
  - Sobre el servidor físico se instala un sistema operativo con hipervisor, que tiene la capacidad de albergar múltiples máquinas virtuales dentro.
  - La ventaja es que se pueden crear múltiples servidores destinados a diferentes cosas, incluso servidores que no sean web.
  - Algunos ejemplos de estos operativos con hipervisor son Citrix XenServer, XCP-ng, Proxmox.
  - La creación e instalación de las máquinas virtuales se puede automatizar con herramientas de software como Vagrant.

Modelos de despliegue

- Contenedor.
  - Los contenedores se caracterizan por ser de usar y tirar, así que el contenido que se quiere persistir hay que guardarlo en un volumen con un nombre fijo.
  - Cuando se cambia de contenedor se le puede incrustar al nuevo contenedor la información que tenía el anterior reutilizando el mismo volumen.
  - Poco a poco, esta solución esta ganando popularidad por la facilidad de automatizar despliegues.

Modelos de despliegue

- Contenedor.
  - Es la solución más novedosa.
  - Los contenedores son entornos definidos y desechables.
  - Podríamos entenderlos como máquinas virtuales muy ligeras, que se crean automáticamente y que, desde el punto de vista del diseño, están pensadas para ser borradas de forma fácil en cualquier momento.
  - Los contenedores tienden a segregar bastante el espacio en disco. Esta segregación se hace en volúmenes.

CI/CD

- La idea detras de CI/CD es automatizar toda la ingeniería del proceso del software para generar los productos de forma automática.
- Dentro de CI/CD se automatiza el análisis del código, el testing, la complilación, y el despliegue y la monitorización.

CI/CD

- La idea detras de CI/CD es automatizar toda la ingeniería del proceso del software para generar los productos de forma automática.
- Dentro de CI/CD se automatiza el análisis del código, el testing, la complilación, y el despliegue y la monitorización.

- El objetivo de CI/CD llega a ser que, con cada commit que llegue a main/master:
  1. Se analice la calidad del código.
  2. Se ejecuten todos los test.
  3. Se compile un ejecutable.
  4. Se haga disponible al cliente.

  Todo eso sin que una persona tenga que intervenir.

GitHub Actions

- Una de las herramientas que tenemos para poder hacer esto es GitHub Actions.
- Con GitHub Actions podemos programar flujos de trabajo de que acciones queremos que se desencadenen con cada acción.
- Los flujos se definen en ficheros en formato YAML y se colocan dentro del directorio *.github/workflows*
- Github buscara todos los flujos que encuentre en ese directorio y los tratará de procesar.

Un ejemplo

```
name: Build and push Docker images
on: [push]

jobs:
  unit-test-restapi:
      ...

  build:
    ...

  deploy:
    ...
```

Un ejemplo

```
build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Docker Buildx
        uses: elgohr/Publish-Docker-Github-Action@3.04
        with:
          name: user/my-project
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}
          registry: ghcr.io
```

Un ejemplo

```
deploy:
    name: Deploy over SSH
    runs-on: ubuntu-latest
    needs: [build]
    steps:
      - name: Deploy over SSH
        uses: fifsky/ssh-action@master
        with:
          host: ${{ secrets.DEPLOY_HOST }}
          user: ${{ secrets.DEPLOY_USER }}
          key: ${{ secrets.DEPLOY_KEY }}
          command: |
            docker stop my-project
            docker rm my-project
            docker pull ghcr.io/user/my-project:latest
            docker run -d -p 3000:3000 --net=host  --name my-project ghcr.io/user/
                my-proyect:latest
```