Projecte 2

Departament d'Informàtica
i Comunicació
LA GUINEUETA
Institut

Sergi Puerta y Pau Oliver
Tutors: Oriol, Víctor, Michael y Sergio
Projecte 2 de segon d'asix

INDEX:

Introducción

Este proyecto está centrado en Docker. Con Docker, puedes crear contenedores que tienen lo que requieres para ejecutar tu aplicación, facilitando la transferencia del producto en diferentes entornos. Ahorrando dinero y reduciendo el uso de la máquina virtual; y así una gran cantidad de memoria. Cada vez es más frecuente su implementación siendo una de las herramientas más útiles para la distribución de software a cualquier servidor a través de contenedores de software, permitiendo a nosotros como desarrolladores o administradores poder hacer simulaciones en entornos de prueba sin la necesidad de tener instalada algún tipo de dependencia.

A diferencia de las máquinas virtuales que utilizamos en el curso, los contenedores tienen una huella de recursos significativamente menor, requieren menos gastos generales para su administración. Las máquinas virtuales también deben encapsular un sistema operativo completamente independiente y otros recursos, mientras que los contenedores comparten el mismo núcleo del sistema operativo y usan un sistema proxy para conectarse a los recursos que necesitan, dependiendo de dónde se encuentren estos recursos. Como podemos ver en la siguiente imagen las máquinas virtuales incluyen la aplicación, las bibliotecas o los archivos binarios necesarios y un sistema operativo invitado completo. La virtualización completa requiere más recursos que la inclusión en contenedores. En cambio los contenedores incluyen la aplicación y todas sus dependencias. Sin embargo, comparten el kernel del sistema operativo con otros contenedores, que se ejecutan como procesos aislados en el espacio de usuario en el sistema operativo host. Todo esto se traduce en una reducción considerable de costos y un retorno de la inversión mucho más rápido al reducir los recursos materiales necesarios.

¿Por qué nos interesa este proyecto?

Este proyecto nos ha llamado la atención porque gracias a ello hemos podido aprender y conocer mejor Docker, cómo funciona, las funciones que tienen, sus capacidades y servicios y también las salidas profesionales que este tiene. Durante estos 2 últimos meses hemos podido emprender un nuevo reto, al principio nos costó plantear el proyecto e imaginar como lo organizaremos pero sesión a sesión fuimos haciendo. Cabe aclarar que el modelo del Trello nos ayudó bastante en la planificación y desarrollo del proyecto. Durante las clases fuimos comprendiendo cómo funciona esta nueva tecnología, y sus virtudes. La siguiente parte que veremos a continuación ya es la parte práctica del proyecto, en la cual enseñó todos los pasos que he llevado a cabo además de todos los servicios y programas utilizados como apache2, python, mySQL o el SSH.

Arquitectura de Docker

Docker usa una arquitectura cliente-servidor. El cliente de Docker habla con el demonio de Docker que hace el trabajo de crear, correr y distribuir los contenedores. Ambos pueden ejecutarse en el mismo sistema, o se puede conectar un cliente a un demonio Docker remoto. El cliente Docker y el demonio se comunican vía sockets o a través de una imagen. Explicamos un poco por orden la arquitectura o funcionamiento de Docker. El cliente de Docker (Docker Client) es la principal interfaz de usuario para Docker. Élacepta comandos del usuario y se comunica con el demonio Docker. El Docker Engine (Demonio del Docker) corre en una máquina anfitriona (host). El usuario no interactúa directamente con el demonio, en su lugar lo hace a través del cliente Docker. El demonio Docker levanta los contenedores haciendo uso de las imágenes, que pueden estar en local o en el Docker Registry y cada contenedor se crea a partir de una imagen en un entorno aislado y seguro dónde se pueda ejecutar nuestra aplicación.

Propuesta Técnica

La propuesta técnica es un documento en el que plasmamos los pasos que hemos seguido para instalar los servicios en un entorno de Docker. En primer lugar, para crear este proyecto, hemos utilizado la tecnología de contenedores Docker. Hemos utilizado una estructura de directorios en la que hemos instalado varios servicios teniendo un Dockerfile y los archivos de configuración. En el Dockerfile hemos definido la imagen base que queríamos utilizar y las dependencias necesarias para el servicio.

Para el servicio de base de datos hemos utilizado MySQL y un archivo schema.sql para definir la estructura de la base de datos y otro archivo data/ para almacenar los datos de las tareas. Luego hemos definido el Dockerfile con las dependencias necesarias para ejecutar MySQL y hemos copiado el archivo schema.sql al contenedor para que se cree la estructura de la base de datos automáticamente al arrancar el contenedor. Para el servicio de administración de la base de datos has utilizado phpMyAdmin y hemos definido el Dockerfile con las dependencias necesarias para ejecutar phpMyAdmin, a parte hemos configurado el archivo de configuración de Nginx para que actúe como un proxy inverso y redireccione las solicitudes del navegador a la interfaz web de phpMyAdmin.

Planificación del proyecto

- **Temporal**

Desarrollo del proyecto	Días
Propuesta técnica	11 dias
Definición de objetivos y alcance del sistema	2
Definición de requisitos	2
Estimación y planificación	3
Estimación de costes	2
Diagrama de contexto	2
Análisis del sistema	9 dias
Modelado de datos	4
Modelo de procesos	5
Diseño	22 días
Revisión del análisis	10
Diseño de BD y ficheros	3
Diseño de módulos	4
Diseños de interfaces de usuario	4
Diseño de procesos	1
Construcción y puesta en marcha	6 días
Desarrollo del plan de instalación	4
Preparación del entorno de desarrollo y pruebas	2
Desarrollo de los componentes de software	26 días
Diseño detallado	5

Programación	16
Diseño y realización de pruebas	5
Memoria final	10 días
Puesta en marcha	4
Redacción del proyecto	4
Entrega del producto	2
Fin del proyecto	84 días

- **Costes**

Recursos	**Unidades**	**Distribuidor**
Personal		
Analistas	**1**	
Programadores	**3**	
Especialista en estudios de mercado	**1**	
Administrador de sistemas	**3**	
Material		
Ordenadores personales	**8**	**Dell**
Software edición de video	**1**	**Kodac**
Software Web	**3**	**Adobe**
Software Ofimática	**8**	**Libre distribución**
Máquina fotográfica	**1**	**Microsoft**

RECURSOS HUMANOS	
RECURSO	CANTIDAD
Especialista en fotografia	1
Especialista en diseño	1
Analistas	1
Programadores	3
Especialista en estudios de mercado	1
Administrador de sistemas	3

RECURSOS MATERIALES	
RECURSO	CANTIDAD
Servidor	2
Pantalla TFT	2
Software	8
Dominio Web	1
Conexión banda ancha	2
Ordenadores personales	8
Máquina fotográfica	1
Software edición de video	1

Software Web	2
Software ofimática	8

RECURSOS HUMANOS	
RECURSO	COSTE/HORA
Especialista en fotografia	15 €
Especialista en diseño	20 €
Analistas	20 €
Programadores	30 €
Especialista en estudios de mercado	25 €
Administrador de sistemas	30€

RECURSOS MATERIALES	
RECURSO	PRECIOS
Servidor	3200 €
Pantalla TFT	110 €
Software	50 €
Dominio Web	13€/MES
Conexión banda ancha	30€/MES
Ordenadores personales	Propiedad de la empresa
Máquina fotográfica	Propiedad de la empresa

Software edición de video	Propiedad de la empresa
Software Web	Propiedad de la empresa
Software ofimática	Propiedad de la empresa

Gasto en recursos: 3360€ + 43€ (al més) = 3403€
Gasto en mano de obra: 260€/hora x 240/horas al més= 62.000€

Desarrollo del Proyecto:

Docker-compose.yml

Este archivo docker-compose define tres servicios que serán ejecutados en contenedores Docker. Cada servicio tiene su propia configuración y dependencias.

El primer servicio llamado "db" define una instancia de MySQL que se ejecutará en un contenedor Docker. Las líneas que comienzan con "-" bajo la clave "volumes" indican la vinculación de los directorios del host con los del contenedor. En particular, la segunda línea monta el archivo de inicialización SQL del host en el directorio de inicialización de la base de datos del contenedor. La tercera línea vincula el socket de MySQL del host con el del contenedor. Las variables de entorno especifican las credenciales y la base de datos que serán creadas. Además, la sección "ports" vincula el puerto 3306 del contenedor con el mismo puerto en el host. Finalmente, se define un comando de "healthcheck" para comprobar periódicamente si el servicio MySQL está en funcionamiento.

```yaml
version: '3'
services:
  db:
    image: mysql:latest
    volumes:
     - db:/var/lib/mysql
     - ./resources/sql/setup.sql:/docker-entrypoint-initdb.d
     - /var/run/mysqld/mysqld.sock:/var/run/mysqld/mysqld.sock
    environment:
      MYSQL_ROOT_PASSWORD: dev
      MYSQL_USER: dev
      MYSQL_PASSWORD: dev
      MYSQL_DATABASE: bbdduniversitat
    ports:
      - "3306:3306"
    command: --default-authentication-plugin=mysql_native_password
    healthcheck:
      test: "mysqladmin ping -h localhost -u dev -p dev"
      interval: 3s
      timeout: 2s
      retries: 10
```

El segundo servicio llamado "phpmyadmin" define una instancia de PhpMyAdmin que se ejecutará en otro contenedor Docker. Las variables de entorno especifican la configuración necesaria para conectarse a la instancia de MySQL en el servicio "db". La sección "ports" vincula el puerto 8091 del contenedor con el puerto 80 en el host. El parámetro "depends_on" indica que el servicio "db" debe estar en funcionamiento antes de que se ejecute este servicio.

```yaml
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  environment:
    PMA_HOST: db
    PMA_PORT: 3306
    PMA_USER: root
    PMA_PASSWORD: dev
  ports:
    - "8091:80"
  depends_on:
    - db
```

Y por último El tercer servicio llamado "jenkins" define una instancia de Jenkins que se ejecutará en otro contenedor Docker. Las líneas que comienzan con "-" bajo la clave "volumes" montan los directorios del host con los del contenedor. La sección "ports" vincula los puertos 8011 y 50000 del contenedor con los mismos puertos en el host. Las variables de entorno configuran opciones de Java y Jenkins. El parámetro "user" establece que el contenedor se ejecutará con permisos de root. El parámetro "depends_on" indica que el servicio "db" debe estar en funcionamiento antes de que se ejecute este servicio. El comando especifica una serie de instrucciones de shell que se ejecutarán cuando el contenedor se inicie. En este caso, se descarga e instala Docker en el contenedor y se instalan algunas herramientas adicionales. Además, se instalan algunos plugins de Jenkins y se inicia el servicio de Jenkins.

```yaml
jenkins:
  image: jenkins/jenkins:lts
  ports:
    - "8011:8080"
    - "50000:50000"
  volumes:
    - "home:/var/jenkins_home"
    - "/var/run/docker.sock:/var/run/docker.sock"
    - "./jenkins:/var/jenkins_conf"
  environment:
    JAVA_OPTS: "-Djenkins.install.runSetupWizard=false"
    JENKINS_OPTS: "--prefix=/jenkins"
    TZ: "Europe/Madrid"
  user: root
  depends_on:
    - db
  command: >
    /bin/sh -c '
    curl -L https://get.docker.com | sh;
    usermod -aG docker jenkins;
    apt-get update;
    apt-get install -y maven gradle curl unzip;
    apt-get clean;
    rm -rf /var/lib/apt/lists/*;
    /usr/local/bin/install-plugins.sh docker-plugin workflow-aggregator github-branch-source;
    /sbin/tini -- /usr/local/bin/jenkins.sh;'
volumes:
  mysql_data:
  jenkins_data:
  home:
  db:
```

Dockerfile

Nuestro dockerfile crea una imagen de contenedor Docker que se utiliza para crear un entorno de desarrollo con varias herramientas instaladas. A continuación, se describen las diferentes instrucciones utilizadas en el Dockerfile y lo que hacen:

FROM orboan/dind: esta instrucción establece la imagen base que se utilizará para construir la imagen del contenedor.

```
FROM orboan/dind
```

ENV: esta instrucción establece varias variables de entorno para el contenedor.

```
ENV \
    USER=alumne \
    PASSWORD=alumne \
    LANG="${language}.UTF-8" \
    LC_CTYPE="${language}.UTF-8" \
    LC_ALL="${language}.UTF-8" \
    LANGUAGE="${language}:ca" \
    REMOVE_DASH_LINECOMMENT=true \
    SHELL=/bin/bash
ENV \
    HOME="/home/$USER" \
    DEBIAN_FRONTEND="noninteractive" \
    RESOURCES_PATH="/resources" \
    SSL_RESOURCES_PATH="/resources/ssl"
ENV \
    WORKSPACE_HOME="${HOME}" \
    MYSQL_ALLOW_EMPTY_PASSWORD=true \
    MYSQL_USER="$USER" \
    MYSQL_PASSWORD="$PASSWORD"
```

apt-get install: esta instrucción instala varias herramientas en el contenedor, como el servidor SSH, Apache, Mariadb, git, Node.js, etc.

```
#Program install
RUN \
  apt update -y && \
  DEBIAN_FRONTEND=noninteractive \
  apt-get install -y --no-install-recommends \
  apt-transport-https \
  ca-certificates \
  build-essential \
  software-properties-common \
  libcurl4 \
  curl \
  apt-utils \
  ssh \
  gradle \
  maven \
  nodejs \
  openssl \
  vim \
  bash-completion \
  iputils-ping \
  npm \
  wget \
  openssl \
  git \
  zip \
  gzip \
  unzip \
  bzip2 \
  lzop \
  sudo && \
  clean-layer.sh


RUN \
    apt update -y && \
    apt install -y supervisor openssh-server apache2 mariadb-server && \
    clean-layer.sh


#We install the services needed
RUN apt-get update && apt-get install -y \
    supervisor \
    openssh-server \
    python3 \
    python3-pip \
    nodejs \
    npm \
    #sdkman \
    docker.io \
    docker-compose \
    mysql-client \
    git \
    #github-cli \
    maven \
    gradle
```

SSH conf: esta sección configura el servidor SSH.

```
# SSH conf
RUN mkdir /var/run/sshd \
RUN echo 'root:root' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config

# SSH login conf
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional pam_loginuid.so@g' -i /etc/pam.d/sshd

ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile
```

Install VS Code: esta sección instala el editor de código Visual Studio Code en el contenedor.

```
#Install VS Code
RUN apt-get update && apt-get install -y curl gpg
RUN curl https://code.visualstudio.com/docs/?dv=linux64_deb && \
    install -o root -g root -m 644 microsoft.gpg /etc/apt/trusted.gpg.d/ && \
    rm microsoft.gpg
RUN echo "deb [arch=amd64] https://packages.microsoft.com/repos/vscode stable main" > /etc/apt/sources.list.d/vscode.list
RUN apt-get update && apt-get install -y code
RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
RUN curl -fsSL https://code-server.dev/install.sh | sh
```

Install Github y Gitlab: esta sección instala el cliente de línea de comandos Hub de Github y Gitlab en el contenedor.

```
# install Github and Gitlab
RUN apt-get update && apt-get install -y git && \
    curl -LJO https://github.com/github/hub/releases/download/v2.14.2/hub-linux-amd64-2.14.2.tgz && \
    tar xvzf hub-linux-amd64-2.14.2.tgz && \
    cd hub-linux-amd64-2.14.2 && \
    ./install && \
    cd ../ && \
    rm -rf hub-linux-amd64-2.14.2 && \
    rm hub-linux-amd64-2.14.2.tgz
```

Install OpenSSL: esta sección instala OpenSSL en el contenedor.

```
# Install OpenSSL
RUN apt-get update && apt-get install -y openssl
```

Install Python 3 y pip: esta sección instala Python 3 y pip en el contenedor.

```
# Install Python 3 and pip
RUN apt-get update && apt-get install -y python3 python3-pip
RUN apt-get clean && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
```

Install Nodejs y npm: esta sección instala Node.js y npm en el contenedor.

```
# Install Nodejs and npm
RUN apt-get install -y nodejs && \
    apt-get install -y npm
```

Install Docker client: esta sección instala el cliente Docker en el contenedor.

```
# Install Docker client
RUN apt-get update && apt-get install -y apt-transport-https ca-certificates curl gnupg lsb-release
RUN curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add -
RUN add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
RUN apt-get update && apt-get install -y docker-ce-cli
```

Conclusión personal

Como conclusión personal, podemos decir que el proyecto de Docker que hemos realizado nos ha gustado, aunque al principio nos ha costado un poco entender y aplicar los conceptos de la tecnología de contenedores. Sin embargo, una vez que hemos empezado a comprender cómo funciona Docker y cómo se estructura un proyecto basado en contenedores, hemos sido capaces de tirar hacia adelante. Además, el proyecto también nos ha permitido mejorar nuestras habilidades técnicas y aprender nuevas tecnologías y herramientas, lo que nos ha dado la oportunidad de ampliar nuestro conocimiento y experiencia en el campo de la informática. Por último, cabe decir que este proyecto también nos ha enseñado la importancia de la perseverancia y el esfuerzo, ya que a pesar de las dificultades que hemos enfrentado, nunca hemos perdido el ánimo y siempre hemos seguido adelante con el objetivo de completar el proyecto de la mejor manera posible.

Webgrafia

https://josejuansanchez.org/bd/practica-05/index.html

https://aws.amazon.com/es/docker/

https://www.ionos.es/digitalguide/servidores/configuracion/tutorial-docker-instalacion-y-primeros-pasos/

https://iesgn.github.io/curso_docker_2021/sesion1/web.html

https://jclopex.medium.com/docker-b%C3%A1sico-primeros-pasos-c57144d18eea

https://www.arsys.es/blog/crea-un-servidor-cloud-con-docker-y-despliega-una-aplicacion-en-un-contenedor

https://www.freecodecamp.org/espanol/news/guia-de-docker-para-principiantes-como-crear-tu-primera-aplicacion-docker/

https://soka.gitlab.io/blog/post/2019-07-08-docker-imagenes-y-contenedores/

https://cloud.google.com/compute/docs/containers?hl=es-419

https://web.archive.org/web/20131229025044/http://www.openstack.org/blog/2013/06/openstack-community-weekly-newsletter-may-31-june-7/

https://web.archive.org/web/20160812093538/http://www8.hp.com/us/en/hp-news/press-release.html?wireId=2060626

https://dev.to/marlosrodriguez/crear-contenedores-de-bases-de-datos-en-docker-bdg