

Laboratorio 09.- Servidor remoto de Google (Computer Engine)

Contenido:

1.	INTRODUCCIÓN	2
2.	CLAVES SSH	2
3.	CREAR MÁQUINA VIRTUAL EN COMPUTER ENGINE.....	3
4.	DOCKER	4
4.1.	Instalación y gestión de imágenes	4
4.2.	Ejecutar containers.....	5
4.3.	Ejecutar servicios	6

Objetivos:

1. Aprender a configurar un servidor remoto en el servicio Google Cloud.
2. Aprender a conectarse a un servidor remoto mediante SSH, sin contraseña.
3. Aprender a dar acceso a terceros a nuestro servidor mediante SSH.

Desarrollo Avanzado de Software - Curso 2023/ 2024

1. Introducción

En esta práctica vamos a crear una máquina virtual como servidor remoto mediante el servicio Computer Engine de Google Cloud. Para utilizar este servicio, es indispensable tener una cuenta de Google y crédito en Google Cloud. Podéis crearos una cuenta con finalidades académicas o bien usar una que ya tengáis. Por otro lado, se os proporcionará un crédito de 50 dólares con fines educativos mediante una clave asociada a vuestro correo de la universidad.

Nota importante: es responsabilidad del alumno configurar correctamente el servidor en Google Cloud y tenerlo listo para las entregas. Para ello, hay que detener el servidor una vez terminado su uso, si no, se corre el riesgo de no llegar con suficiente crédito al periodo de evaluación.

2. Claves SSH

Para generar el par de claves SSH podéis seguir las indicaciones dadas [aquí](#).

En Windows, abre el símbolo del sistema y usa el comando `ssh-keygen` con la marca `-C` para crear un nuevo par de claves SSH.

```
ssh-keygen -t rsa -f C:\Users\WINDOWS_USER\.ssh\KEY_FILENAME -C USERNAME -b 2048
```

Reemplaza lo siguiente:

- `WINDOWS_USER`: Es tu nombre de usuario en la máquina de Windows.
- `KEY_FILENAME`: el nombre de tu archivo de claves SSH.

Por ejemplo, un nombre de archivo de `my-ssh-key` genera un archivo de clave privada llamado `my-ssh-key` y un archivo de clave pública llamado `my-ssh-key.pub`.

- `USERNAME`: tu nombre de usuario en la VM.

Para conectarte a una VM mediante SSH desde un cliente de OpenSSH, haz lo siguiente:

1. [Agrega una clave SSH a la VM](#). Si lo haces mediante consola de la VM, Google Cloud creará el usuario automáticamente en la instancia si el usuario de la clave SSH no se corresponde con el usuario de la VM.
2. En la consola de Google Cloud, ve a la página **Instancias de VM** y busca la dirección IP externa de la VM a la que te quieres conectar.
3. Abre una terminal en tu estación de trabajo.
4. Conéctate a la VM mediante la ejecución del siguiente comando:

```
ssh -i PATH_TO_PRIVATE_KEY USERNAME@EXTERNAL_IP
```

Reemplaza lo siguiente:

- `PATH_TO_PRIVATE_KEY`: la ruta al archivo de clave SSH privada que corresponde a la clave pública que agregaste a la VM.
- `USERNAME`: el nombre de usuario es el que especificaste cuando [creaste la clave SSH](#).

Desarrollo Avanzado de Software - Curso 2023/ 2024

- `EXTERNAL_IP`: la dirección IP externa de la VM.

3. Crear máquina virtual en Compute Engine

Una vez en la página principal, pincha en Compute Engine y luego Instancias de VM (Habilitar Compute Engine si fuera necesario). Pincha en “Crear Instancia” y aparecerá esta pantalla:

Para crear una instancia de VM, selecciona una de estas opciones:

- Nueva instancia de VM**
Crea una instancia de VM única desde cero
- Nueva instancia de VM a partir de una plantilla**
Crea una instancia de VM única a partir de una plantilla existente
- Instancia nueva de VM a partir de una imagen de máquina**
Crea una instancia de VM única a partir de una imagen de máquina existente
- Marketplace**
Implementa una solución lista para usar en una instancia de VM

Nombre *
sgssi-servidor-e2-micro

ADMINISTRAR ETIQUETAS DE INSTANCIA Y ETIQUETAS DE RECURSO

Región *
us-central1 (Iowa)
La región es permanente

Zona *
us-central1-a
La zona es permanente

Configuración de la máquina

Try the new H3 machine series, optimized for HPC. **TRY NOW**

☒ De uso general ☐ Optimizado para procesamiento ☐ NUEVO ☐ Con optimización de memoria ☐ GPU


Tipos de máquinas para cargas de trabajo comunes, optimizados en función del costo y la flexibilidad

Series	Descripción	vCPUs	Memory	Plataf
<input type="radio"/> C3	Rendimiento alto y coherente	4 - 176	8 - 1,408 GB	Intel S
<input type="radio"/> C3D	VISTA PREVIA	4 - 360	8 - 2,880 GB	AMD
<input checked="" type="radio"/> E2	Procesamiento diario de bajo costo	0.25 - 32	1 - 128 GB	Segu
<input type="radio"/> N2	Precio y rendimiento equilibrados	2 - 128	2 - 864 GB	Intel C
<input type="radio"/> N2D	Precio y rendimiento equilibrados	2 - 224	2 - 896 GB	AMD I
<input type="radio"/> T2A	Cargas de trabajo de escalamiento horizontal	1 - 48	4 - 192 GB	Altra c
<input type="radio"/> T2D	Cargas de trabajo de escalamiento horizontal	1 - 60	4 - 240 GB	AMD I
<input type="radio"/> N1	Precio y rendimiento equilibrados	0.25 - 96	0.6 - 624 GB	Intel S

Tipo de máquina
Elige un tipo de máquina con cantidades predeterminadas de CPU virtuales y memoria que se adapten a la mayoría de las cargas de trabajo. También puedes crear una máquina personalizada según las necesidades particulares de tu carga de trabajo. [Más información](#)

CONFIGURACIÓN PREDETERMINADA PERSONALIZADO

e2-micro (2 CPU virtuales, 1 núcleo, 1 GB de memoria)



vCPU
De 0.25 a 2 CPU virtuales (1 núcleo compartido)

Memory
1 GB

Estimación mensual
USD7.11
Equivalente a alrededor de USD0.01 por hora
Paga por lo que usas, con facturación por segundo y sin pagos por adelantado

Elemento	Estimación mensual
2 vCPU + 1 GB memory	USD6.11
Disco persistente balanceado de 10 GB	USD1.00
Total	USD7.11

[Precios de Compute Engine](#)
[LESS](#)

Opciones a configurar:

- Nombre: cualquiera pero debe ser fácilmente reconocible, por ejemplo “sgssi-labo”.
- Familia de máquinas uso general.
- Serie: E2.
- Tipo de máquina: e2-micro.

En la opción de Disco de arranque; cambiar a Ubuntu 22.04 LTS (x86-64, amd64).

En la opción de identidad y acceso a la API, elegir Compute engine default service account; elegir la opción permitir el acceso predeterminado y permitir tráfico HTTP, HTTPS.

Desarrollo Avanzado de Software - Curso 2023/ 2024

Identidad y acceso a la API ?

Cuentas de servicio ?

Cuenta de servicio

Compute Engine default service account

Requiere que se configure el rol de usuario de cuenta de servicio (roles/iam.serviceAccountUser) para los usuarios que desean acceder a las VMs con esta cuenta de servicio. [Más información](#)

Permisos de acceso ?

- ☒ Permitir el acceso predeterminado
- ☐ Permitir el acceso total a todas las API de Cloud
- ☐ Configurar acceso para cada API

Firewall ?

Agrega etiquetas y reglas de firewall para permitir determinados tipos de tráfico de red desde Internet

- ☒ Permitir tráfico HTTP
- ☒ Permitir tráfico HTTPS
- ☒ Permitir las verificaciones de estado del balanceador de cargas

Para añadir la clave pública SSH generada en el primer paso, en Opciones avanzadas; Seguridad; Acceso a la VM; Administrar acceso; Agrega claves SSH generadas de forma manual; Agregar elemento.

Por último, crear instancia.

Para parar el servidor, selecciona la instancia creada y pulsa DETENER.

4. Docker

4.1. Instalación y gestión de imágenes

Instalación de Docker Engine en Ubuntu siguiendo los pasos de Seguir los pasos "Install using the apt repository" [aquí](#).

Docker necesita privilegios de root. Para evitar el uso de sudo:

- Crear grupo docker: **\$ sudo groupadd docker**
- Añadir usuario actual al grupo docker: **\$ sudo usermod -aG docker \$USER**
- Reiniciar el sistema, volver a entrar, y ejecutar **\$ docker run hello-world**

Tras ejecutar el último comando, se tiene que leer el siguiente texto en consola:

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

Desarrollo Avanzado de Software - Curso 2023/ 2024

Docker tiene un repositorio local que contiene las imágenes que vamos a usar en nuestro ordenador.

- Puedes ver las imágenes disponibles en tu repositorio local mediante **\$ docker images**

El repositorio remoto más común se encuentra en Docker Hub¹, que es el repositorio configurado por defecto al instalar Docker.

- Explora las imágenes que se pueden encontrar en Docker Hub.

4.2. Ejecutar containers

En Docker, los containers se ejecutan a partir de una imagen.

- El siguiente comando ejecuta un container a partir de la imagen **hello-world**: **\$ docker run hello-world**
- Este otro comando ejecuta la imagen **Ubuntu** de manera interactiva y abre la consola dentro del contenedor: **\$ docker run -it ubuntu bash**

Con el comando **docker ps -a** podemos ver cuántos contenedores están en marcha.

Para parar los containers, necesitamos su nombre o id:

- Ejecuta **\$ docker kill nombre/id** (Después ejecuta **\$ docker ps -a**)

Aunque los containers no están funcionando, hay que eliminarlos:

- Ejecuta **\$ docker rm nombre/id** (Después ejecuta **\$ docker ps -a** y **\$ docker images**)

Si queremos eliminar una imagen usaremos **\$ docker rmi nombre_imagen**

Para ejecutar comandos dentro de un contenedor usaremos **docker exec**

- **\$ docker exec nombre_container ls**

Construir imágenes

Para construir una imagen Docker necesitamos un *Dockerfile*. Un Dockerfile es un archivo de texto plano que le dice a Docker cómo tiene que construir la imagen. Algunas instrucciones son:

- **FROM**: la imagen base a usar.
- **ADD**: añade archivos locales a la imagen.
- **RUN**: ejecuta comandos.
- **CMD**: el comando que se ejecutará al arrancar el container a partir de la imagen descrita en el Dockerfile.

Por ejemplo, un Dockerfile podría ser de la siguiente manera:

```
# Imagen base
FROM Ubuntu

# Descargar e instalar dependencias
RUN apt -qq update && apt -qq -y install fortune

# Comando de arranque
```

Desarrollo Avanzado de Software - Curso 2023/ 2024

```
CMD /usr/games/fortune
```

Ejecutando **\$ docker build -t="nombre_imagen"** . en el mismo directorio del Dockerfile, crearemos una imagen con el nombre seleccionado. El nombre puede ser cualquiera.

Puedes comprobar que la imagen ha sido construida y añadida al repositorio local mediante **\$ docker images**

Una vez creada la imagen puedes, ejecutar un container de la imagen construida mediante **\$ docker run nombre_imagen**

Docker nos permite montar directorios que son compartidos por el host y el container, es decir que ambos pueden leer y escribir en esos directorios. Los containers son por definición efímeros y de una existencia muy corta. Por lo tanto, los volúmenes en Docker son muy importantes ya que nos permiten mantener datos que de otra manera se perderían al borrar el container.

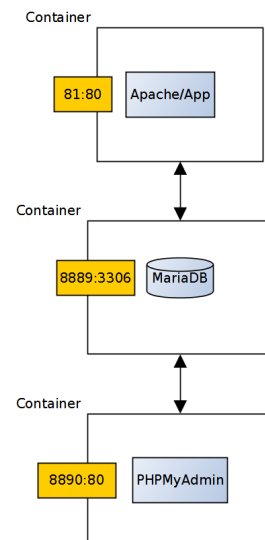
4.3. Ejecutar servicios

Mediante **docker-compose** podemos definir un grupo de servicios que se ejecuten a la vez de manera coordinada, basándose cada servicio en una imagen Docker. Teniendo en cuenta que hoy en día muchas aplicaciones se basan en combinaciones de servicios (Base de datos, servidor web, otros servidores, etc.) ésta es una característica muy importante.

Vamos a ver cómo funciona **docker-compose** explorando el proyecto que tenéis que usar como base para la entrega 2. El proyecto consta de tres servicios que conforman una aplicación web muy sencilla: un servidor Web con una aplicación PHP (la aplicación Web propiamente dicha) que accede a una Base de Datos Maria DB, la Base de Datos Maria DB, y un servidor Web con la aplicación PHPMyAdmin para gestionar la Base de Datos MariaDB.

- Clona el repositorio de GitHub que contiene el proyecto (O descárgalo): **\$ git clone <https://github.com/mikel-egana-aranguren/docker-lamp.git>**

El archivo **docker-lamp/docker-compose.yml** contiene la definición de los servicios. En este caso hay tres servicios (el nombre del servicio y de la imagen Docker en el que se basa pueden ser diferentes):



Desarrollo Avanzado de Software - Curso 2023/ 2024

- **web:** este servicio se basa en la imagen "web" construida a partir del Dockerfile, que contiene un servidor web Apache y una aplicación PHP definida en /app (Esta imagen es simplemente una extensión de la imagen oficial de PHP). Se enlaza al servicio **db** y redirige el puerto 81 del host al puerto 80 del container (donde se ejecuta Apache).
- **db:** la imagen mariadb es la imagen oficial que provee la base de datos Maria DB. En este caso el servicio se ejecuta obteniendo los datos del volumen ./mysql (Es decir que si volvemos a ejecutar un container, los datos se cargarán de ese directorio y no se perderán, aunque el container haya desaparecido), con la configuración de la sección "environment" y redirigiendo el puerto del host 8889 al puerto del container 3306.
- **phpmyadmin:** este servicio se basa en la imagen oficial de PHPMyAdmin, que se conecta al servicio **db** y se usa para administrar la base de datos, redirigiendo el puerto del host 8890 al puerto del container 80.

```

web:
  image: web
  environment:
    - ALLOW_OVERRIDE=true
  ports:
    - "81:80"
  links:
    - db
  volumes:
    - ./app:/var/www/html/

db:
  image: mariadb:10.8.2
  restart: always
  volumes:
    - ./mysql:/var/lib/mysql
  environment:
    MYSQL_ROOT_PASSWORD: root
    MYSQL_USER: admin
    MYSQL_PASSWORD: test
    MYSQL_DATABASE: database
  ports:
    - "8889:3306"

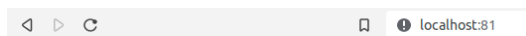
phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  links:
    - db
  ports:
    - 8890:80
  environment:
    MYSQL_USER: admin
    MYSQL_PASSWORD: test
    MYSQL_DATABASE: database
  
```

Los puertos 81 y 8890 del host no estarán abiertos. Por ello, habrá que crear una regla en el Firewall del servidor. Abre el menú de navegación de Google Cloud para acceder a VPC network > Firewall: Crear regla de Firewall. Hay que rellenar los siguientes campos:

- Nombre: http_dockerDAS
- Targets: Todas las instancias de la red
- Rangos IPv4: 0.0.0.0/0
- Seleccionar TCP y añadir puertos: 81, 8890

Para desplegar el proyecto:

- Sitúa la terminal dentro del repositorio /docker-lamp.
- Construye la imagen **web**: **\$ docker build -t="web"**.
- Despliega los servicios mediante **\$ docker-compose up**. Si fuese necesario, para instalar docker-compose ejecutar **\$ sudo apt-get install docker-compose**
- Visita la web desde consola haciendo **\$ curl http://localhost:81**

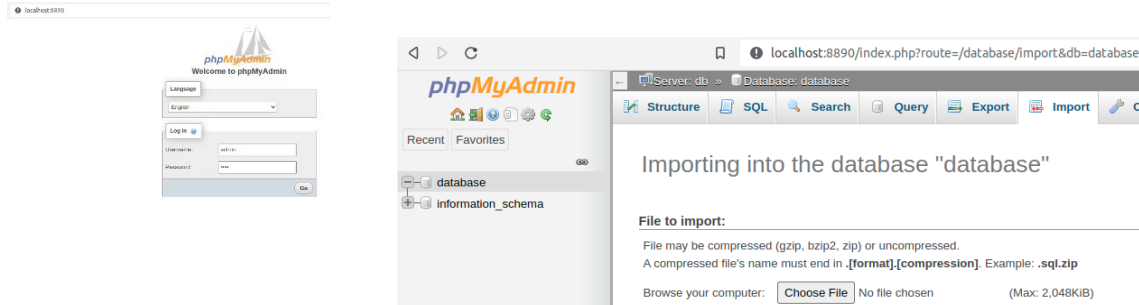


Yeah, it works!

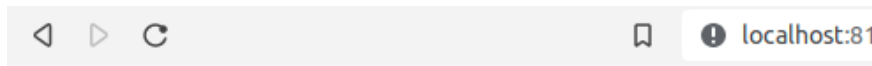
Table 'database.usuarios' doesn't exist

Desarrollo Avanzado de Software - Curso 2023/ 2024

- Para añadir los datos necesarios, visita http://ip_servidor:8890/ (Tal y como lo hemos definido en docker-compose.yml, usuario "admin", password "test"). Haz click en "database" y luego en "import", desde donde elegimos el archivo docker-lamp/database.sql.



- Vuelve a <http://localhost:81>, debería tener más información.



Yeah, it works!

1 mikel 2 aitor

- Para parar los servicios, en otra terminal, **\$ docker-compose down**

Antes de apagar el servidor, parar los servicios docker