

**Proyecto final  
Sistemas operativos**



**Ambiente de computación distribuida**

**presentado por**

**Deisy Catalina Melo (2041790)  
Natalia Riaños Horta (2042568)  
Carlos Andres Borja (2040507)**

## **CONTENIDO:**

**Introducción**

**máquina virtual 1 (Apache)**

**máquina virtual 2 (Docker)**

**máquina virtual 3 (Nginx)**

**Dificultades**

**Repositorio Github**

**video**

**referencias**

## **INTRODUCCIÓN**

En este proyecto se llevó a cabo el despliegue de un ambiente computacional distribuido compuesto por 3 máquinas virtuales. En la primera se instaló el servidor web de Apache, en la segunda se ejecuta una aplicación basada en contenedores de Docker usando la herramienta 'docker-compose', en la tercera se realizó la instalación y configuración de Nginx como 'proxy inverso'.

Para la creación de las máquinas virtuales se utilizó la herramienta Virtualbox e instalamos el sistema operativo Linux ubuntu.

## Maquina virtual 1 (Apache):

Los pasos que seguimos para la instalación y configuración de apache fueron los siguientes:

- Instalar apache con `$ sudo apt-get install apache2`
- Cargar la página por defecto de apache en el navegador de la máquina virtual para así confirmar que quedó instalado y corre correctamente; en el navegador digitamos 'localhost' o la dirección IP de la máquina virtual.
- Crear una página nueva :

```
$ cd /var/www/  
$ nano index.html
```

- Editar ruta de la página por defecto

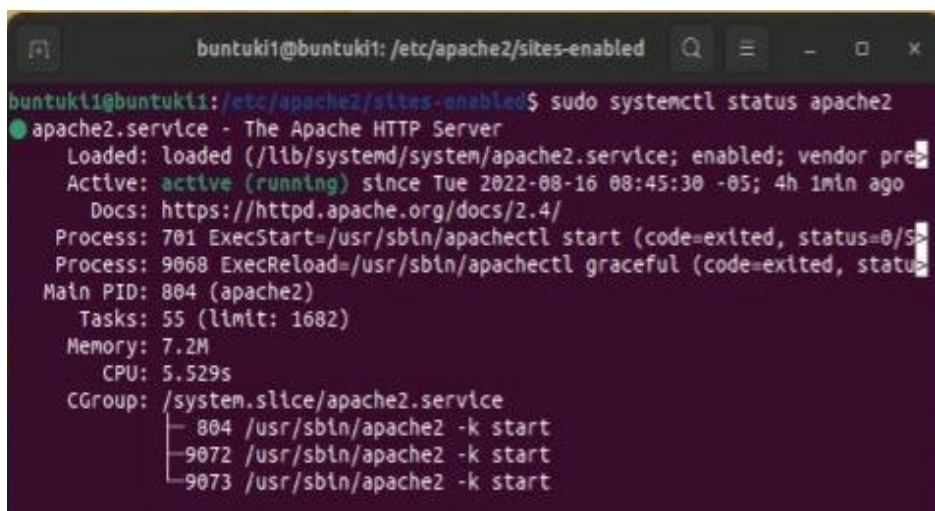
```
$ cd /etc/apache2/sites-enabled  
$ nano 000-default.conf
```

- Recargamos el servicio para actualizar la configuración.

```
$ systemctl reload apache2
```

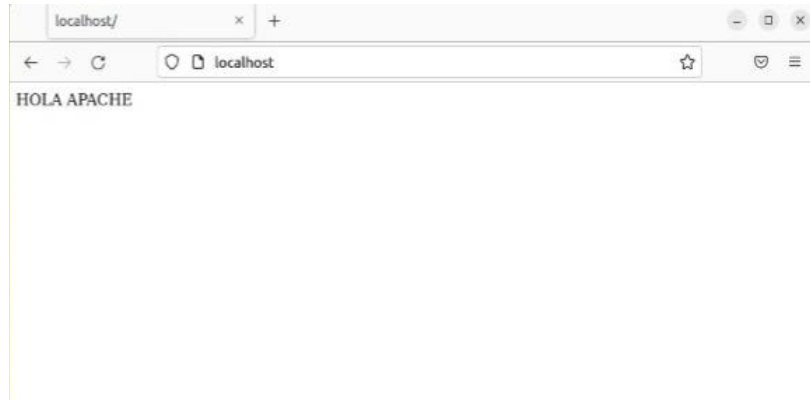
- Miramos el estado de apache, para confirmar que esté "activo-corriendo"

```
$ systemctl status apache2
```



```
buntuki1@buntuki1: /etc/apache2/sites-enabled  
buntuki1@buntuki1:/etc/apache2/sites-enabled$ sudo systemctl status apache2  
● apache2.service - The Apache HTTP Server  
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pre  
   Active: active (running) since Tue 2022-08-16 08:45:30 -05; 4h 1min ago  
     Docs: https://httpd.apache.org/docs/2.4/  
   Process: 701 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/S  
   Process: 9068 ExecReload=/usr/sbin/apachectl graceful (code=exited, statu  
 Main PID: 804 (apache2)  
    Tasks: 55 (limit: 1682)  
   Memory: 7.2M  
      CPU: 5.529s  
   CGroup: /system.slice/apache2.service  
           └─ 804 /usr/sbin/apache2 -k start  
             9072 /usr/sbin/apache2 -k start  
             9073 /usr/sbin/apache2 -k start
```

- Finalmente corremos nuestra página en el navegador digitando "localhost"



## Maquina virtual 2 (Docker)

Para asegurarnos de que obtuvimos la versión estable más reciente de Docker Compose, se hizo la descarga de su repositorio oficial de Github.

El siguiente comando descargó la versión 1.26.0 y guardó el archivo ejecutable en /usr/local/bin/docker-compose, para que estuviera globalmente accesible como docker-compose:

```
$ sudo curl -L  
"https://github.com/docker/compose/releases/download/1.26.0/do  
cker-compose-$(uname -s)-$(uname -m)" -o  
/usr/local/bin/docker-compose
```

A continuación, se procedió a establecer los permisos correctos para que el comando docker-compose fuera ejecutable:

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Verificamos su correcta instalación y su versión:

```
$ docker-compose --version
```

Se procedió a configurar un archivo docker-compose.yml

Para ello se creó un entorno de servidor web usando la imagen Nginx oficial de Docker Hub, el registro público de Docker. Este entorno en contenedor sirvió como archivo HTML estático único.

Comenzamos creando un nuevo directorio en su carpeta de inicio, y luego lo movimos a él:

```
$ mkdir ~/compose-demo  
$ cd ~/compose-demo
```

En este directorio, se configuró una carpeta de aplicaciones que se usó como la raíz del documento para el entorno Nginx:

```
$ mkdir app
```

Mediante el editor “nano” se creó un nuevo archivo ***index.html*** en la carpeta app:

```
$ nano app/index.html
```

Se editó con el siguiente contenido:

```
<!doctype html>  
<html lang="en">  
<head>  
  <meta charset="utf-8">  
  <title>Docker Compose Demo</title>  
  <link rel="stylesheet"  
href="https://cdn.jsdelivr.net/gh/kognise/water.css@latest/dist/dark.min.css">  
</head>  
<body>  
  
  <h1>This is a Docker Compose Demo Page.</h1>  
  <p>This content is being served by an Nginx  
container.</p>  
  
</body>  
</html>
```

Procedimos a crear el archivo **docker-compose.yml**:

```
$ nano docker-compose.yml
```

Dejamos el siguiente contenido en este archivo:

```
version: '3.7'
services:
  web:
    image: nginx:alpine

    ports:
      - "8000:80"

    volumes:
      - ./app:/usr/share/nginx/html
```

Explicando lo anterior mencionamos que el archivo `docker-compose.yml` normalmente comienza con la definición de la versión. Esto para poder indicarle a Docker Compose qué versión de la configuración estamos usando.

Luego con el bloque **services**, donde se configuró los servicios que son parte de este entorno. En nuestro caso, tenemos un único servicio llamado `web`. Este servicio utiliza la imagen ***nginx:alpine*** y establece una redirección de puerto con la directiva `ports`. Así, todas las solicitudes en el **puerto 8000** del equipo host (el sistema desde el cual está ejecutando Docker Compose) serán redirigidas al contenedor `web` en el **puerto 80**, donde se ejecutará **Nginx**.

La directiva `volumes` crea un volumen compartido entre el equipo host y el contenedor. Esto compartirá la carpeta `app` local con el contenedor, y el volumen se ubicará en `/usr/share/nginx/html` dentro del contenedor, que luego sobrescribirá la raíz predeterminada del documento para Nginx.

Hasta acá ya hemos creado una página demo y un archivo `docker-compose.yml` para crear un entorno de servidor web en contenedor que lo presentará. Luego de esto, se abrió el entorno con Docker Compose.

Con el archivo `docker-compose.yml` implementado, ejecutamos Docker Compose para mostrar nuestro entorno. Con el siguiente comando se descargó la imagen Docker necesaria, creó un contenedor para el servicio web y ejecutó el entorno en contenedor en modo segundo plano:

```
docker-compose up -d
```

Luego en el navegador de esta máquina virtual ejecutamos *localhost:8000*



Máquina virtual con Docker compose corriendo correctamente.

### **Máquina virtual 3 (Nginx):**

Realizamos la instalación de nuestro nginx ejecutando el comando

```
$sudo apt-get install nginx
```





verificamos en el navegador de la MV que se esté ejecutando la página por defecto en la ruta 'localhost'

Luego configuramos nuestro proxy inverso modificando el archivo 'default', para ello primero nos ubicamos en el directorio 'sites-available'

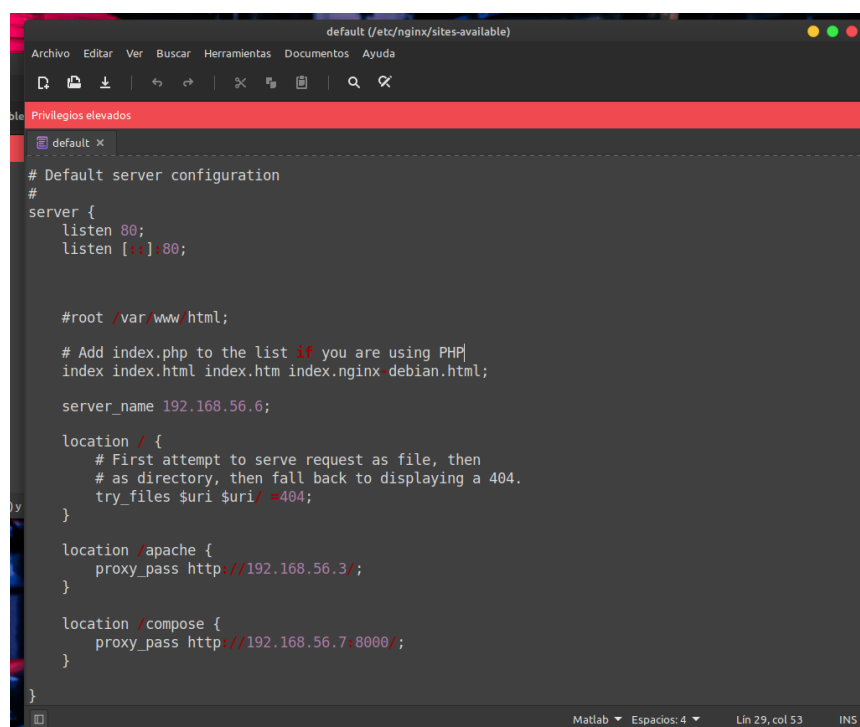
```
$cd etc/nginx/sites-available
```

procedemos a acceder al archivo para su edición:

```
$sudo nano default
```

comentamos la ruta especificada: 'root/var/www/html'

En el apartado del nombre del servidor le indicamos la dirección IP del proxy: **192.168.56.6** y en location, especificamos las rutas que deseamos **'/apache'** y **'/compose'** y las redirecciones a la IP de las respectivas máquinas:



**Crear red local:** Se realiza el paso a paso de [\[guia\]](#)

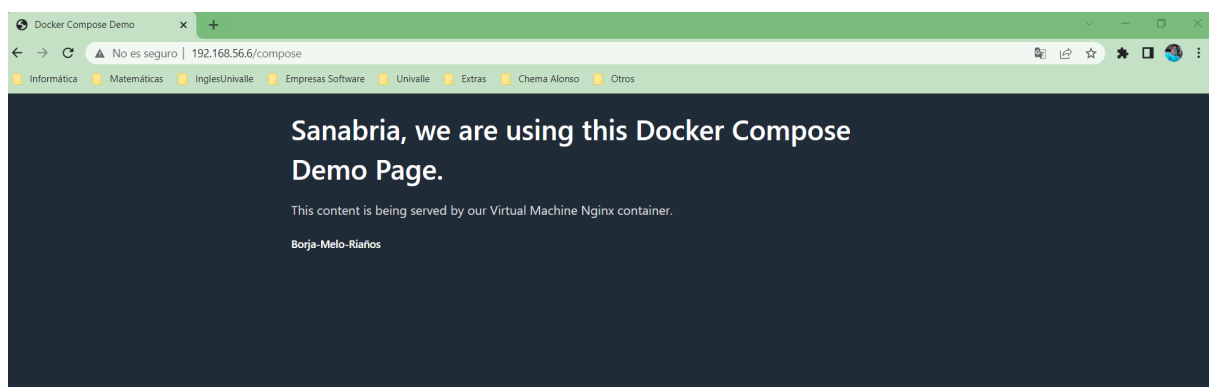
## Ejecución de la aplicación

Nos ubicamos en el navegador del anfitrión (la máquina física) y procedemos a realizar los solicitudes http,

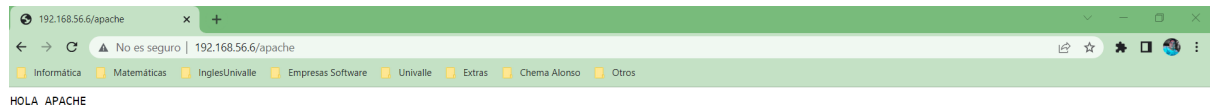
<http://192.168.56.6/>



<http://192.168.56.6/compose>



<http://192.168.56.6/apache>



## Dificultades:

- **IP docker-compose MV:** La mayor dificultad fue configurar las direcciones ip de las 2 máquinas anteriores. Puesto que inicialmente estábamos usando el ejemplo brindado por el docente en la sección de configuración de docker-compose en la nube de google, por ende la IP que arrojaba no nos permitía la conexión desde otra máquina virtual. Así que optamos por usar la configuración predeterminada para docker compose.
- **IP en el proxy:** Otro requerimiento que se nos hizo un poco complicado inicialmente fue el modificar el archivo del proxy para establecer las conexiones (redireccionamiento según la petición). Para solventar dudas acudimos a youtube y algunos sitios web, en donde explicaban cuáles eran las IP que debíamos usar realmente.

**Repositorio Github:** <https://github.com/Deisy05/proxyInversoNGINX.git>

**Video:** <https://youtu.be/ZVg4gDVVfk8>

## Fuentes de información de apoyo

- Crear redes tipo solo anfitrión <https://www.youtube.com/watch?v=1iFTkKFdL00>  
<https://carleton.ca/scs/2019/creating-a-new-host-only-adapter-in-virtualbox/>
- Docker compose <https://docs.docker.com/compose/gettingstarted/>
- Unit docker.service is masked:  
<https://forums.docker.com/t/failed-to-start-docker-service-unit-is-masked/67413>
- Sitio oficial de Docker  
<https://docs.docker.com/compose/install/>
- Configurar el proxy inverso  
<https://kinsta.com/es/blog/proxy-inverso/#como-configurar-nginx-como-un-proxy-inverso>