Práctica 3 - Documento Teórico

Administración de Servidores

Ángel Sánchez Corchado
Manuel Pérez Ruiz
Pablo Vilar Bustillo

Índice

1. Introducción

En este documento hablaremos, desde una perspectiva teórica, de los distintos conceptos y aspectos que hemos manejado tanto en las sesiones prácticas como teóricas de la asignatura, como son servidores web, Vagrant, el uso del fichero *hosts*, Apache2 (qué es, cómo funciona, cómo se estructura...) o el archivo de configuración de dominio. Estos conceptos nos ayudarán en la correcta realización de la Práctica 3 de la asignatura. Esto es un documento teórico como se ha especificado anteriormente, por lo que algunos conceptos más complejos serían convenientes aplicarlos de manera práctica antes de la realización de la Práctica 3, para así garantizar una correcta realización de la misma.

1.1. Vagrant

- **Introducción**

Vagrant es una herramienta de código abierto que permite la creación y gestión de entornos de desarrollo virtualizados de manera fácil y reproducible. Proporciona una capa de abstracción sobre diferentes tecnologías de virtualización, como *VirtualBox*, *VMware*, *Hyper-V*, entre otras, permitiendo a los desarrolladores y administradores de sistemas configurar y compartir entornos de desarrollo de manera consistente, independientemente del sistema operativo anfitrión.

Con *Vagrant*, es posible definir y compartir configuraciones de máquinas virtuales a través de archivos de configuración, llamados *Vagrantfiles*, lo que facilita la creación y aprovisionamiento automático de máquinas virtuales con software y configuraciones específicas. Esto proporciona un entorno de desarrollo uniforme y reproducible, lo que resulta beneficioso para la colaboración en equipos de desarrollo y la creación de ambientes de pruebas consistentes.

- **Creación de un *Vagrantfile***

Un *Vagrantfile* es un archivo de configuración en formato *Ruby* que define la configuración de una máquina virtual. Un ejemplo de un fichero *Vagrantfile* podría ser el siguiente:

```
Vagrant.configure("2") do |config|
    config.vm.box = "hashicorp/bionic64"
    config.vm.define "vm1" do |vm1|
        vm1.vm.hostname = "vm1"
        vm1.vm.network "private_network", ip: "192.168.2.101"
        # Revisar la ruta del aprovisionamiento por si es diferente
        vm1.vm.provision :shell, path: "aprovisionamiento.sh"
    end

    config.vm.define "vm2" do |vm2|
        vm2.vm.hostname = "vm2"
        vm2.vm.network "private_network", ip: "192.168.2.102"
        # Revisar la ruta del aprovisionamiento por si es diferente
        vm2.vm.provision :shell, path: "aprovisionamiento.sh"
    end

    config.vm.define "vm3" do |vm3|
        vm3.vm.hostname = "vm3"
        vm3.vm.network "private_network", ip: "192.168.2.103"
        # Revisar la ruta del aprovisionamiento por si es diferente
        vm3.vm.provision :shell, path: "aprovisionamiento.sh"
    end
end
```

Figura 1: Vagrantfile

Para crear un Vagrantfile se pueden seguir los siguientes pasos:

1. **Inicializar el directorio:** En el directorio donde se desea crear el *Vagrantfile*, se ejecuta el comando ***vagrant init***. Esto inicializa un nuevo proyecto de *Vagrant* en ese directorio y crea un archivo de configuración básico llamado *Vagrantfile*.
2. **Configurar el Vagrantfile:** Puede abrirse con un editor de texto y configurarse según las necesidades del proyecto. Aquí se definen las características de la máquina virtual que se creará, así como las opciones de configuración adicionales, como en el ejemplo dado en la *Figura 1*.

Algunas de las configuraciones más comunes que se pueden establecer en un *Vagrantfile* son:

- **Configuración de la máquina virtual**: Se especifica el tipo de máquina virtual a utilizar (como *VirtualBox* o *VMware*), por ejemplo, *config.vm.provider*, la cantidad de memoria *RAM* asignada, el número de *CPUs*, el sistema operativo base, etc.

- **Configuración de la red:** Se puede asignar una dirección IP privada a la máquina virtual, abrir puertos para acceder a servicios específicos, configurar la red interna o externa, entre otras opciones. Por ejemplo en la *Figura 1* mediante **vm1.vm.network: "private_network", *ip: "192.168.2.101"***

- **Aprovisionamiento:** Permite definir scripts o comandos de configuración que se ejecutarán en la máquina virtual durante su creación y configuración. Esto incluye la instalación de software adicional, la configuración de servicios, la creación de usuarios, etc. Un ejemplo sería el siguiente:

```
#!/bin/bash

# Actualiza los repositorios
sudo apt-get update

# Instala nmap
sudo apt-get install nmap -y

# Instala iptables (en caso de que no esté instalado por defecto)
sudo apt-get install iptables -y
```

Figura 2: Script de aprovisionamiento

- **Compartición de carpetas:** Permite definir carpetas compartidas entre el sistema anfitrión y la máquina virtual, facilitando el intercambio de archivos y permitiendo el acceso a recursos locales desde la máquina virtual.

- ## Comandos en *Vagrant*

Una vez configurado a nuestro gusto el Vagrantfile, se pueden utilizar los comandos de Vagrant para administrar la máquina virtual. Algunos comandos comunes son los siguientes:

- ***vagrant up*:** Inicia la máquina virtual basada en el Vagrantfile. Si la máquina virtual no existe, se crea utilizando la configuración especificada.

- ***vagrant ssh*:** Permite acceder a la máquina virtual mediante una conexión SSH. Esto facilita la interacción con la máquina virtual y la ejecución de comandos dentro de ella.

- ***vagrant halt*:** Detiene la máquina virtual. Los recursos asignados a la máquina se liberan, pero se mantiene su estado para poder reiniciarla posteriormente.

- ***vagrant destroy*:** Elimina completamente la máquina virtual y todos sus recursos asociados. Esta acción es irreversible, por lo que se debe tener precaución al utilizar este comando.

- ***vagrant reload*:** Reinicia la máquina virtual. Se aplicarán los cambios realizados en el Vagrantfile y se reiniciará la máquina con la configuración actualizada.

1.2. Fichero *Hosts*

Este fichero es importante en la realización de la práctica para poder acceder al servidor web de *Apache* que configuraremos en la misma. Este archivo lo que hace es mapear las direcciones *IP* a los nombres de dominio, para que el sistema sepa a qué dominio pertenece la dirección *IP* que, por ejemplo, introducimos en el navegador, como por ejemplo la dirección *as.uca.es* usada en la práctica.

Dependiendo del sistema operativo utilizado, este fichero estará en un lugar diferente:

- **Windows**: `C:/Windows/System32/drivers/etc/hosts`
- **Linux**: `/etc/hosts`
- **Mac**: `/private/etc/hosts`

De esta forma tenemos la máquina virtual y el entorno listos para la realización de la configuración sobre servidores web.

La sintaxis a usar para añadir una pareja de IP-Dominio es **<IP><Dominio>**. En la práctica por ejemplo sería **192.168.2.101 as.uca.es**, siendo la ip la ip privada de la máquina virtual que hemos creado y configurado en el Vagrantfile, en el apartado *ip:* (Figura 1)

1.3. Apache2

1.3.1. ¿Qué es *Apache*?

Apache2 es un servidor web de código abierto utilizado para alojar y servir sitios web de manera segura y confiable. Compatible con varios sistemas operativos, utiliza el protocolo HTTP y se puede personalizar mediante módulos adicionales. Apache2 ofrece soporte para múltiples lenguajes de programación como PHP, Python o Perl, así como algunas características entre las que podemos encontrar compresión de datos, autenticación, encriptación, manejo de sesiones...

Además, permite la configuración de virtualhosts, que se utilizan para alojar varios sitios en un solo servidor físico. Su arquitectura modular proporciona una gran flexibilidad y escalabilidad. Sin embargo, al tener tantas opciones de configuración, puede dar problema a brechas de seguridad y exploits si no se manejan con cuidado.

En líneas generales, podríamos decir que en Apache2 encontramos un servidor web confiable y popular dentro de la comunidad de desarrollo web.

1.3.2. Ubicaciones más importantes (Linux)

De entre las ubicaciones que podemos encontrar en *Apache2*, las más importantes que encontramos son:

- **Directorio raíz de la configuración *(/etc/apache2):*** El directorio raíz de la configuración contiene la mayoría de los archivos de configuración del servidor web *Apache2*, incluyendo el archivo principal de configuración *apache2.conf* y otros archivos de configuración para módulos y virtualhosts.

- **Fichero de configuración principal *(apache2.conf)*:** El archivo *apache2.conf* es el archivo de configuración principal de *Apache2*, donde se definen los ajustes globales del servidor web, como los puertos de escucha, los ajustes de seguridad y la configuración del directorio raíz.

- **Configuración de los módulos (*mods-available/, mods-enabled/*):** En *mods-available/* podemos encontrar todos los módulos disponibles, mientras que en *mods-enabled/* se encuentran los módulos que se encuentran activados actualmente.

- **VirtualHosts (sites-available/, sites-enabled/):** En estos directorios encontramos los archivos de configuración de VirtualHosts. Más concretamente, en *sites-available/* encontramos todos los hosts disponibles y en *sites-enabled/* encontramos los hosts que se encuentran activados.

- **Logs (/var/log/apache2):** En este directorio es donde podemos encontrar almacenados los registros de errores, los de acceso y otros registros del servidor web.

1.3.3. Módulos

Un módulo es una serie de funcionalidades que sirven para extender la funcionalidad de Apache. Esto deriva en una personalización y control total, ya que solo activas dichos módulos que vas a utilizar. Estos se cargan en la memoria del servidor web de Apache para proporcionar las funcionalidades extras que estos contienen.

Una ventaja de los módulos es que permiten una adaptación total a los requisitos de la web que estamos desarrollando, y un mayor rendimiento ya que se carga en memoria solo los módulos activados.

Hay una infinidad de módulos, los cuales se pueden consultar en la página de Apache: https://httpd.apache.org/docs/2.4/es/mod/

A continuación, se describirán algunos de los módulos más importantes:

- **Módulos de Autenticación:** Estos módulos permiten la autenticación de usuarios, además de un control a los recursos que queremos proteger de nuestro servidor web. El funcionamiento es sencillo, permiten acceso a ciertas áreas o funciones de la web a los usuarios con las credenciales adecuadas.

  Los más usados son: `mod_auth_basic`, `mod_auth_digest`, `mod_auth_form`, `mod_authnz_ldap`, `mod_auth_kerb` y `mod_auth_mellon`.

  Apache permite el uso de varios módulos de autenticación, así que puedes combinarlos entre ellos para un férreo control de autenticación en ciertas áreas, o un mayor rendimiento mediante una autenticación más liviana.

- **Módulos de Compresión:** La función principal de estos módulos es mejorar el rendimiento del servidor web, mediante una compresión de los datos enviados al servidor, lo cual permite un uso menor del ancho de banda. Esto se realiza gracias a que estos módulos reducen el tamaño de la respuesta HTTP. Es especialmente útil usarlos para dicho contenido web estático.

  Los más usados son 2: `mod_deflate` y `mod_gzip`

  Es mucho más recomendable usar el primer módulo, `mod_deflate` debido a que es una versión más nueva y más eficiente en términos de compresión. Sin embargo, `mod_gzip` es bastante usado también por motivos de compatibilidad.

- **Módulos de Balanceo de Carga**: Permiten distribuir la carga de trabajo entre varios servidores o nodos. La función principal de estos es mejorar el rendimiento, disponibilidad y escalabilidad de nuestro servidor web. Al dividir las peticiones de los usuarios, necesitan menos recursos en cada servidor, y si ocurre algún problema en un nodo, el resto pueden cumplir la labor sin afectar a la disponibilidad de la página.

  Los más usados son 2: `mod_proxy_balancer` y `mod_lbmethod_byrequests`.

  Al igual que los módulos de autenticación, estos pueden usarse en conjunto en el mismo servidor web. El primer módulo es mejor en casos cuando nuestros nodos están ubicados en distintos clústeres, ya que está basado en proxy. Sin embargo, cuando estos clusters son de una capacidad similar, es más efectivo el segundo, ya que este balancea la carga basado en el número de solicitudes.

- Módulos de Reescritura: Estos permiten manipular las URLs de las solicitudes antes de que lleguen al servidor web. Su principal función es de seguridad, debido a que, por ejemplo, nos permite eliminar de la URL posibles argumentos de un fichero PHP que puedan causar un exploit para atacantes. Además, las URLs serían más amigables y fáciles de recordar para los usuarios. Sin embargo, hay que tener en cuenta que una mayor manipulación de las URLs puede transformarse en un peor rendimiento, sobre todo si las reglas usadas son muy complejas.

  Los más usados son 2: `mod_rewrite` y `mod_alias`

  Es mucho mejor de usar `mod_alias`, debido a que este, al contrario que `mod_rewrite`, no es sólo un módulo de reescritura, por lo que con un solo módulo podemos tener acceso a las mismas funcionalidades que 4 o 5 módulos.

1.3.4. Herramientas

Apache contiene muchas herramientas y programas de soporte para tu servidor. Estos pueden ser utilizados de muchas maneras, por ejemplo, podemos medir el rendimiento del servidor, o para renderizar usando código java. A continuación describiré las más importantes:

- ***Apache Tomcat***: Es utilizado para ejecutar aplicaciones web desarrolladas en Java. Implementa tecnologías como *Java Servlets* o *JavaServer Pages (JSP)*. Además, proporciona una serie de características exclusivas, como la gestión de sesiones y la administración remota.
- ***Apache Maven***: Es una herramienta de gestión y construcción de proyectos ampliamente utilizada en el desarrollo de cualquier aplicación de software. Nos proporciona la gestión de dependencias (algo parecido a los módulos de *Apache HTTP*), un ciclo de vida, o un repositorio remoto para almacenar bibliotecas o artefactos. *Apache Maven* y *Apache HTTP* se complementan muy bien la una a la otra, ya que podemos construir una aplicación usando Apache *Maven* para desplegarla en *HTTP Server*.
- **Ab**: Es una herramienta para hacer pruebas de rendimiento (*benchmarks*) de tu servidor *HTTP*. Está totalmente orientado a *Apache HTTP Server*. Nos puede decir información muy útil, como el tiempo medio que tarda en responder nuestro servidor, o el tiempo total que tarda en gestionar un número concreto de peticiones.

1.4. Archivo de configuración del dominio

El archivo de configuración del dominio es aquel que define cómo se gestionan las solicitudes entrantes para nuestro dominio en cuestión. Dicho fichero, generalmente con la forma `midominio.com.conf` se ubica en `/etc/apache2/sites-available`.

Dicho fichero contiene 2 etiquetas principales:

- Etiqueta `<VirtualHost>`: Se utiliza para definir la configuración de un host virtual en Apache, el cual permite que un servidor web atienda a más de un dominio en la misma máquina. Las características más importantes de la etiqueta son las siguientes:
    - **ServerName**: Nombre del dominio principal asociado con el host virtual.
    - **ServerAlias:** Alias para el host virtual.
    - **DocumentRoot:** Ruta del directorio raíz (por defecto, `index.html`) para el host virtual.
    - **ErrorLog:** Ubicación de los registros de errores del host virtual.
    - **CustomLog:** Ubicación de los registros personalizados del host virtual.
- Etiqueta `<Directory>`: Se utiliza para aplicar configuraciones específicas a un directorio del host virtual. Las características más importantes de la etiqueta son las siguientes:
    - **Directory:** Ruta del directorio al que se le aplicarán las configuraciones específicas de la etiqueta.
    - **Options**: Permite habilitar ciertas opciones para el directorio. La más común es `Indexes`, que es para mostrar un listado de archivos si no se encuentra un archivo de índice.
    - **AllowOverride:** Define las directivas de configuración que se pueden sobreescribir mediante archivos `.htaccess` dentro del directorio.
    - **Require:** Establece los requisitos de acceso para el directorio, como restricciones de IP o autenticación.