Proyecto: Self hosting

Tabla de Contenido

Los alumnos realizarán hosting en su propio ordenador de casa, instalando un servidor web.

Durante la actividad se usará `example.com` o `www.example.com` como ejemplo de sitio web. Deberás sustituirlo por tu propio dominio.

1. Introducción

El objetivo de esta práctica es realizar hosting en servidores propios colocados en la vivienda del alumno. Esta tarea es llamada *self-hosting* y tiene ciertas complicaciones que son muy interesantes

de aprender para los futuros administradores.

2. Requerimientos

El proceso de self-hosting requiere de los siguientes pasos:

1. Registro de dominio y DynDNS
2. Redirección de puertos en el router residencial.
3. Instalar, desplegar y provisionar el servidor.
4. Probar.

2.1. Registro de dominio y DynDNS

El alumno deberá registrar un dominio y conectar el nombre con la dirección IP de su servidor.

2.1.1. Compra del dominio

El alumno comprará un nombre de dominio a su elección. Este proceso no incluirá el *hosting* es decir el alojamiento del servidor, solo su nombre.

Se ofrece un ejemplo de compra de dominio con el agente de registro Ionos en otro documento, a disposición del alumno.

Alternativamente se podría realizar mediante una empresa como http://noip.com la cual nos cede un nombre de dominio y nos permite asociarlo con la dirección IP del ordenador de casa. Sin embargo, en este caso preferimos hacerlo comprando un dominio para poder explorar todos la funcionalidad que nos ofrecen y aprender más.

2.1.2. DynDNS

Asignaremos nuestra dirección IP dinámica al nombre de dominio. Esto se puede realizar de distintas formas.

Algunos router permiten configurar automáticamente la actualización de la IP introduciendo las credenciales de la cuenta de http://noip.com u otro proveedor parecido.

En el caso de Ionos tenemos dos formas de hacerlo:

1. Instalando mediante python un programa en algún ordenador de casa, el cual al encenderse se pondrá en contacto con Ionos y actualizará la dirección IP que nos ha asignado nuestro ISP (*Internet Service Provider*). Este proceso se describe en https://www.ionos.es/ayuda/dominios/configurar-la-direccion-ip/ conectar-el-dominio-a-una-red-con-ip-dinamica-utilizando-dns-dinamico-linux/? source=helpandlearni
2. Accediendo al panel de desarrollador, obtendremos un token de autorización. Con el token autorizaremos el uso del API de DNS. Esta autorización nos devolverá una URL de actualización, la cual introduciremos como una tarea programada. Este proceso se explica en el documento de Ionos a disposición de los alumnos.

2.2. Redirección de puertos en el router

Dado que el número de direcciones IPs es limitada, nuestro router realizará traducción de direcciones (NAT) entre la IP pública (WAN) y las direcciones privadas. Debemos configurar nuestro router para que redirecciones los puertos `TCP/80` y `TCP/443` a la dirección privada de nuestro servidor.

Este proceso depende tanto del router como de nuestro ISP. En algunos casos no se puede realizar a no ser que paguemos (aproximadamente un euro). En otros casos no podemos utilizar algunos de esos puertos ya que la compañía los reserva para acceder remótamente a nuestro router y configurarlo. Deberás averiguar cómo es tu router y cual es la política de tu ISP.

2.3. Servidor web

A continuación comentamos los requerimientos del proyecto.

2.3.1. Software

Ser un servidor Apache 2.4+. El sistema operativo es indiferente.

2.3.2. Puertos

Tener conexión HTTP por el puerto 80 y HTTPS por el puerto 443.

2.3.3. Certificados

Se conectará a páginas seguras mediante:

- Certificado autofirmado
- Certificado proporcionado por IONOS.
- Certificado de [Let'sEncrypt](https://letsencrypt.org/). Realizar este apartado tiene más nota que el anterior. Podemos seguir las instrucciones de [Digital Ocean](https://www.digitalocean.com/community/tutorials/how-to-secure-apache-with-let-s-encrypt-on-debian-11)

Para usar Let's Encrypt sigue las instrucciones de la siguiente página https://letsencrypt.org/es/getting-started/

2.3.4. Página de inicio

El servidor tendrá una página de inicio `index.html` que nos dará la bienvenida al servidor.

2.3.5. Páginas de error personalizadas

Si accedemos a una página no disponible, nos mostrará una página de error 404 personalizada. Por ejemplo, http://example.com/xyz nos devolverá un error 404 si el recurso `xyz` no existe.

2.3.6. Descargar una imagen

Accediendo a `/logo.png` descargaremos con un tamaño de alrededor de un Megabyte (abstenerse fotos contra el decoro). Utilizaremos esta imagen para hacer pruebas.

2.3.7. Administración

Si accedemos a la ruta `/admin` del servidor nos pedirá un usuario y contraseña. El usuario `admin` podrá acceder con la contraseña `asir`.

En caso de autenticarse correctamente, nos aparecerá una página web con el título *Administración*. Dentro de esta página aparecerá una imagen de al menos un Megabyte de tamaño.

2.3.8. Status

Al acceder a una ruta `/status` se nos pedirá un usuario y contraseña. El usuario es `sysadmin` y la contraseña `risa`.

En caso de autenticarse correctamente, nos mostrará información sobre el status del servidor, peticiones, uso de memoria, etc.

Para mostrar la información de status usaremos, por ejemplo:

`mod_status`
: más sencillo, al estar incluido en Apache

[uptime kuma](https://uptime.kuma.pet/)
: Una herramienta fácil de usar y hospedar para monitorizar el sitio web.

2.3.9. Prueba de rendimiento

Realizar una prueba de rendimiento con Apache `ab` (https://httpd.apache.org/docs/2.4/programs/ab.html). La prueba debe realizarse **desde otro ordenador** distinto al que contiene el servidor web. Se puede pedir la colaboración de un compañero/a para realizar las pruebas.

Se deberá probar la carga del servidor con 100 y 1000 usuarios **concurrentes**. El número de peticiones a probar será de 1000 y 10000.

Parámetros a utilizar además de otros que fuesen necesarios:

`-k`
: Envía la cabecera `KeepAlive`, la cual pide al servidor que no cierre la conexión después de que se realice la petición, reusándola en su lugar (recomendada).

Generaremos la prueba, primero con 100 usuarios y 1000 peticiones y luego con 1000 usuarios y 1000 peticiones, para los siguientes recursos:

- `https://example.com/`, la misma página con seguridad SSL/TLS.
- `https://example.com/logo.png` accediendo a un recurso.

- `https://example.com/admin/` usando el parámetro apropiado para la autenticación básica.

¿Qué efecto tendría en el rendimiento enviar la siguiente cabecera? ¿Por qué?

`-H`

Enviar una cabecera `-H "Accept-Encoding: gzip, deflate"` la cual que se comprima la salida de datos del 25% a un 75%.

Documenta en la práctica los resultados obtenidos y comenta tus impresiones acerca del rendimiento de tu sitio web.

Interpretar los resultados

Ejemplo de resultados de una prueba de rendimiento

```
Time taken for tests:   60.1234 seconds         ①
Complete requests:  24538                       ②
Failed requests:    58                          ③
(Connect: 0, Length:    58, Exceptions: 0)
Requests per second:    408.96 [#/sec] (mean)   ④
Time per request:   48.905 [ms] (mean)          ⑤
Time per request:   2.445 [ms] (mean, across all concurrent requests) ⑥
```

① Tiempo total que ha durado el test (menor es mejor)

② Número de pruebas completadas con éxito (mayor es mejor)

③ Número de pruebas que han fallado (menor es mejor)

④ Número de peticiones por segundo (mayor es mejor)

⑤ Tiempo medio que lleva cada petición (menor es mejor)

⑥ Tiempo medio que llevan las peticiones teniendo en cuenta todas las peticiones que se hacen a la vez.

2.3.10. Despliegue

La creación del servidor web se puede realizar de distintas formas:

1. Mediante una máquina virtual desplegada con Vagrant con provisión mediante scripts.
2. Mediante contenedores desplagados por Docker y Docker Compose.
3. Mediante una combinación de Vagrant (crea la máquina virtual) y Ansible (hace la provisión).
4. Sólo mediante Ansible. Por ejemplo en una Raspberry Pi.

3. Evaluación

A continuación se resume la puntuación por apartados:

Puntos	Apartado
[0-3]	Conexión segura
[0-2]	Autenticación básica
[0-1]	Páginas de error personalizadas
[0-3]	Información sobre el servidor
[0-2]	Pruebas de rendimiento
[0-2]	Despliegue automatizado
[0-3]	Documentación
[0-16]	**Total**

3.1. Se conecta al servidor de forma segura

Puntos	Descripción
0	No se puede conectar o el acceso no es seguro.
1	El certificado es autofirmado.
2	El certificado es proporcionado por IONOS
3	El certificado es proporcionado por Let's Encrypt

3.2. Autenticación básica

Al acceder a `/admin` se controla el acceso mediante seguridad, usuario `admin` y contraseña `asir`

Puntos	Descripción
0	No se puede conectar o el acceso falla (no se puede entrar o se puede entrar sin contraseña)
1	Se puede acceder con autenticación pero hay algún problema (usuario o contraseña equivocados)
2	Se accede con usuario y contraseña especificada.

3.3. Páginas de error personalizadas

Cuando se accede a un recurso que no existe, se muestra una página de error `404` personalizada.

Puntos	Descripción
0	No funciona o no es una página personalizada.
1	Todo funciona como se espera.

3.4. Status del servidor

Accediendo a `/status` tendrá autenticación básica con usuario `admin` y contraseña `asir`. Nos mostrará información sobre el servidor.

Puntos	Descripción
0	No hay autenticación o no hay una página para mostrar datos del servidor.
1	Muestra estadísticas mediante el módulo `mod_status`
2	Muestra estadísticas mediante una herramienta con `uptime kuma` o similar.
3	Utiliza otra herramienta con mayor dificultad de instalación

3.5. Pruebas de rendimiento

Realiza las pruebas de rendimiento.

Puntos	Descripción
0	No hace pruebas de rendimiento.
1	Realiza las pruebas para 100 usuarios y 1000 peticiones y con 1000 usuarios y 1000 peticiones.
2	Realiza las pruebas y documenta los resultados obtenidos y expresa su opinión al respecto.

3.6. Despliegue automatizado

Puntos	Descripción
0	No se puede automatizar el despliegue.
1	Realiza el despliegue mediante Vagrant y máquinas virtuales.
2	Realiza el despliegue mediante contenedores y/o Vagrant + Ansible

3.7. Documentación

Puntos	Descripción
0	Sin documentación
1	Documenta algo, pero es poco o de poca calidad
2	Documenta abundantemente.
3	Documenta con calidad explicado y razonado