Universidad Autónoma Gabriel René Moreno
Facultad de Ingeniería en Ciencias de la Computación y Telecomunicaciones

Configuración de un Server en la Nube

GRUPO 03-SC

Integrantes:

Choque Coca Liz Dara	217011810
Choque Severiche Diego Ilich	215057376
Delgadillo García Mauricio Elian	217015689

Docente: M.Sc. Ing. Evans Balcazar Veizaga

Asignatura: Tecnología Web - INF513 SC

Semestre: 2 - 2020

SANTA CRUZ - BOLIVIA

Índice General

1. Servicio DNS

El servicio de nombres de dominio (DNS) es un servicio de Internet que asigna direcciones IP y nombres de dominio completos (FQDN) entre sí. De esta forma, DNS alivia la necesidad de recordar direcciones IP. Las computadoras que ejecutan DNS se denominan servidores de nombres. Ubuntu viene con BIND (Berkley Internet Naming Daemon), el programa más común utilizado para mantener un servidor de nombres en Linux.

1.1. Instalación

En un indicador de terminal, ingresamos el siguiente comando para instalar dns:

$ sudo apt install bind9

Un paquete muy útil para probar y solucionar problemas de DNS es el paquete dnsutils. Muy a menudo, estas herramientas ya estarán instaladas, pero para verificar y/o instalar dnsutils ingresamos lo siguiente:

$ sudo apt install dnsutils

1.2. Configuración

Hay muchas formas de configurar BIND9. Algunas de las configuraciones más comunes son un servidor de nombres de almacenamiento en caché, un servidor primario y un servidor secundario.

- Cuando se configura como un servidor de nombres de almacenamiento en caché, BIND9 encontrará la respuesta a las consultas de nombres y recordará la respuesta cuando se vuelva a consultar el dominio.
- Como servidor primario, BIND9 lee los datos de una zona desde un archivo en su host y tiene autoridad para esa zona.
- Como servidor secundario, BIND9 obtiene los datos de la zona de otro servidor de nombres autorizado para la zona.

1.2.1. Visión general

Los archivos de configuración de DNS se almacenan en el directorio **/etc/bind**. El archivo de configuración principal es **/etc/bind/named.conf**, que en el diseño proporcionado por el paquete solo incluye estos archivos.

- **/etc/bind/named.conf.options**: opciones de DNS global
- **/etc/bind/named.conf.local**: para nuestras zonas
- **/etc/bind/named.conf.default-zones**: zonas predeterminadas como localhost, su reverso y sugerencias de root.

1.2.2. Almacenamiento en caché del servidor de nombres

La configuración predeterminada actúa como un servidor de almacenamiento en caché. Simplemente descomentamos y editamos /etc/bind/named.conf.options para configurar las direcciones IP de los servidores DNS nuestro ISP, en este caso usaremos los servidores DNS públicos de Google.

```
forwarders {
    8.8.8.8;
    8.8.4.4;
};
```

Para habilitar la nueva configuración, reiniciamos el servidor DNS. Desde un indicador de terminal:

$ sudo systemctl restart bind9.service

1.2.3. Servidor Primario

En esta sección, BIND9 se configurará como servidor primario para el dominio **testing06.com**.

1.2.3.1. Archivo de reenvío de zona

Para agregar una zona DNS a BIND9, convirtiendo BIND9 en un servidor primario, primero edite **/etc/bind/named.conf.local**:

```
zone "testing06.com" {
  type master;
  file "/etc/bind/db.testing06.com";
};
```

Ahora usamos un archivo de zona existente como plantilla para crear el archivo **/etc/bind/db.testing06.com**:

$ sudo cp /etc/bind/db.local /etc/bind/db.testing06.com

Editamos el nuevo archivo de zona **/etc/bind/db.testing06.com** y cambiamos localhost. al FQDN de nuestro servidor, dejando el adicional **.** al final. Cambiamos 127.0.0.1 por la dirección IP del servidor de nombres y root.localhost por una dirección de correo electrónico válida, pero con la extensión **.** en lugar del símbolo @ habitual, dejando nuevamente el **.** al final. Cambiamos el comentario para indicar el dominio al que pertenece este archivo.

Creamos un registro **A** para el dominio base, **testing06.com**. Además, creamos un registro **A** para **ns.testing06.com**, y por último los registros que nos servirán para los servicios que brindaremos a través del servidor:

```
;
; BIND data file for testing06.com
;
$TTL    604800
@       IN      SOA     testing06.com. root.testing06.com. (
                              3         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.testing06.com.
@       IN      MX  1   mail.testing06.com.
@       IN      A       34.123.221.192
@       IN      AAAA    ::1
ns      IN      A       34.123.221.192
mail    IN      A       34.123.221.192
www     IN      A       34.123.221.192
```

Debemos incrementar el número de serie cada vez que realice cambios en el archivo de zona. Si realiza varios cambios antes de reiniciar BIND9, simplemente incremente el Serial una vez.

Una vez que hayamos realizado cambios en el archivo de zona, es necesario reiniciar BIND9 para que los cambios surtan efecto:

$ sudo systemctl restart bind9.service

1.2.3.2. Archivo de zona reversa

Ahora que la zona está configurada y resolviendo nombres en direcciones IP, es necesario agregar una zona inversa para permitir que DNS resuelva una dirección en un nombre.

Editamos **/etc/bind/named.conf.local** y agregamos lo siguiente:

```
zone "221.123.34.in-addr.arpa" {
  type master;
  file "/etc/bind/db.34"
}
```

Ahora creamos el archivo **/etc/bind/db.34**:

$ sudo cp /etc/bind/db.127 /etc/bind/db.34

A continuación, editamos **/etc/bind/db.34** cambiando las mismas opciones que **/etc/bind/db.testing06.com**:

```
;
; BIND reverse data file for 34.123.221.192
;
$TTL    604800
@       IN      SOA     ns.testing06.com. diego_severiche.hotmail.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns.
192     IN      PTR     ns.testing06.com.
192     IN      PTR     mail.testing06.com.
192     IN      PTR     www.testing06.com.
```

El número de serie en la zona inversa también debe incrementarse en cada cambio. Para cada registro A que configuremos en /etc/bind/db.testing06.com, es decir, para una dirección diferente, debemos crear un registro PTR en /etc/bind/db.34.

Después de crear el archivo de zona inversa, reiniciamos BIND9:

$ sudo systemctl restart bind9.service

2. Servidor de correo electrónico

El proceso de obtener un correo electrónico de una persona a otra a través de una red o Internet implica que muchos sistemas trabajen juntos. Cada uno de estos sistemas debe estar configurado correctamente para que el proceso funcione. El remitente utiliza un Agente de usuario de correo (MUA), o cliente de correo electrónico, para enviar el mensaje a través de uno o más Agentes de transferencia de correo (MTA), el último de los cuales lo entregará a un Agente de entrega de correo (MDA) para su entrega al buzón del destinatario, desde el cual será recuperado por el cliente de correo electrónico del destinatario, generalmente a través de un servidor POP3 o IMAP.

2.1. Dovecot

Dovecot es un agente de entrega de correo, escrito con la seguridad principalmente en mente. Admite los principales formatos de buzón de correo: mbox o Maildir. Esta sección explica cómo configurarlo como servidor POP3.

2.1.1. Instalación

Para instalar un servidor Dovecot básico con funciones POP3 e IMAP comunes, ejecute el siguiente comando:

$ sudo apt install dovecot-imapd dovecot-pop3d

Hay varios otros módulos de Dovecot, incluidos dovecot-sieve (filtrado de correo), dovecot-solr (búsqueda de texto completo), dovecot-antispam (entrenamiento de filtro de spam), dovecot-ldap (directorio de usuarios).

2.1.2. Configuración

Para configurar Dovecot, editaremos el archivo **/etc/dovecot/dovecot.conf** y sus archivos de configuración incluidos en **/etc/dovecot/conf.d/**. De forma predeterminada, todos los protocolos instalados se habilitarán mediante una directiva de inclusión en **/etc/dovecot/dovecot.conf**.

Por defecto, el formato mbox está configurado; si es necesario, también puede utilizar Maildir. Se puede encontrar más sobre eso en los comentarios en **/etc/dovecot/conf.d/10-mail.conf**.

2.2. Postfix

Postfix es el Agente de transferencia de correo (MTA) predeterminado en Ubuntu. Intenta ser rápido y seguro, con flexibilidad en la administración. Es compatible con el sendmail de MTA. Esta sección explicará la instalación, incluido cómo configurar SMTP para comunicaciones seguras.

2.2.1. Instalación

Para instalar Postfix, ejecutamos el siguiente comando:

$ sudo apt install postfix

Por ahora, está bien simplemente aceptar los valores predeterminados presionando return para cada pregunta. Algunas de las opciones de configuración se investigarán con mayor detalle en la siguiente etapa.

2.2.2. Configuración básica

Hay cuatro cosas que debe decidir antes de iniciar la configuración:

- El <Domain> para el que aceptará el correo electrónico (usaremos mail.testing06.com en nuestro ejemplo)
- La red y el rango de clases de su servidor de correo (usaremos 192.168.0.0/24)
- El nombre de usuario (usamos docente)
- Tipo de formato de buzón (mbox es el predeterminado, usaremos la alternativa, Maildir)

Para configurar postfix, ejecute el siguiente comando:

$ sudo dpkg-reconfigure postfix

Se mostrará la interfaz de usuario. En cada pantalla, seleccione los siguientes valores:

- Internet Site
- mail.testing06.com
- docente
- testing06.com, localhost.localdomain, localhost, etc
- No
- 0.0.0.0/0 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128 192.168.0.0/24
- 0
- +
- Ipv4

Para configurar el formato del buzón, editamos el archivo de configuración directamente o usar el comando postconf. En cualquier caso, los parámetros de configuración se almacenarán en el archivo **/etc/postfix/main.cf**. Más adelante, si deseamos volver a configurar un parámetro en particular, podemos ejecutar el comando o cambiarlo manualmente en el archivo.

2.2.3. Autenticación SMTP

SMTP-AUTH permite que un cliente se identifique a sí mismo a través del mecanismo de autenticación SASL, utilizando Transport Layer Security (TLS) para cifrar el proceso de autenticación. Una vez autenticado, el servidor SMTP permitirá al cliente retransmitir correo.

Para configurar Postfix para SMTP-AUTH usando SASL (Dovecot SASL), ejecutamos estos comandos en un indicador de terminal:

$ sudo postconf -e 'smtpd_sasl_type = dovecot'

$ sudo postconf -e 'smtpd_sasl_path = private/auth'

$ sudo postconf -e 'smtpd_sasl_local_domain ='

$ sudo postconf -e 'smtpd_sasl_security_options = noanonymous,noplaintext'

$ sudo postconf -e 'smtpd_sasl_tls_security_options = noanonymous'

$ sudo postconf -e 'broken_sasl_auth_clients = yes'

$ sudo postconf -e 'smtpd_sasl_auth_enable = yes'

**$ sudo postconf -e 'smtpd_recipient_restrictions = **

$ permit_sasl_authenticated,permit_mynetworks,reject_unauth_destination'

Hay varias propiedades del mecanismo SASL que vale la pena evaluar para mejorar la seguridad de su implementación. Las opciones “noanonymous, noplaintext” impiden el uso de mecanismos que permiten la autenticación anónima o que transmiten credenciales sin cifrar.

Configuraremos la encriptación TLS más adelante.

La configuración inicial del sufijo está completa. Ejecutamos el siguiente comando para reiniciar el servicio postfix:

$ sudo systemctl restart postfix.service

2.2.4. Configuración SASL

Postfix admite dos implementaciones SASL: Cyrus SASL y Dovecot SASL. Para habilitar Dovecot SASL, será necesario instalar el paquete dovecot-core:

$ sudo apt install dovecot-core

A continuación, editamos **/etc/dovecot/conf.d/10-master.conf** y cambiamos lo siguiente:

```
service auth {
  # auth_socket_path points to this userdb socket by default. It's typically
  # used by dovecot-lda, doveadm, possibly imap process, etc. Users that have
  # full permissions to this socket are able to get a list of all usernames and
  # get the results of everyone's userdb lookups.
  #
  # The default 0666 mode allows anyone to connect to the socket, but the
  # userdb lookups will succeed only if the userdb returns an "uid" field that
  # matches the caller process's UID. Also if caller's uid or gid matches the
  # socket's uid or gid the lookup succeeds. Anything else causes a failure.
  #
  # To give the caller full permissions to lookup all users, set the mode to
  # something else than 0666 and Dovecot lets the kernel enforce the
  # permissions (e.g. 0777 allows everyone full permissions).
  unix_listener auth-userdb {
    #mode = 0666
    #user =
    #group =
  }

  # Postfix smtp-auth
  unix_listener /var/spool/postfix/private/auth {
    mode = 0660
    user = postfix
    group = postfix
  }

  # Auth process is run as this user.
  #user = $default_internal_user
}
```

Para permitir el uso de SMTP-AUTH por clientes de Outlook, cambie la siguiente línea en la sección de mecanismos de autenticación de **/etc/dovecot/conf.d/10-auth.conf** de:

auth_mechanisms = plain

a esto:

auth_mechanisms = plain login

Una vez que haya configurado Dovecot, reiniciamos con:

$ sudo systemctl restart dovecot.service

2.3. SendGrid

De forma predeterminada, el servicio que utilizamos en la nube permite conexiones de salida en todos los puertos excepto en el 25, que está bloqueado debido al riesgo de abuso. Todos los demás puertos están abiertos, incluidos los puertos 587 y 465. Es por eso que usaremos SendGrid el cual es un servicio de correo electrónico de terceros que ofrece a los usuarios de Compute Engine (el servicio que utilizamos) una prueba gratuita con 12,000 correos electrónicos transaccionales gratis cada mes.

2.3.1. Configuración

Ejecutamos los siguientes comandos en la terminal de SSH para usar SendGrid como retransmisión de SMTP con Postfix.

Nos convertimos en superusuario:

$ sudo su –

Establecemos una máscara de seguridad:

$ umask 077

Modificamos las opciones de configuración de Postfix. Abrimos **/etc/postfix/main.cf** para edición. Por ejemplo, para usar el editor de texto vim, ingresamos el comando siguiente:

$ vim /etc/postfix/main.cf

Actualizamos el archivo agregando las siguientes líneas:

```
relayhost = [smtp.sendgrid.net]:2525
smtp_tls_security_level = encrypt
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
header_size_limit = 4096000
smtp_sasl_security_options = noanonymous
```

Generamos el mapa de contraseña SASL mediante la clave de API que generamos en nuestro perfil de SendGrid. Reemplazamos nuestra-api-key por la clave de API que generamos.

$ echo [smtp.sendgrid.net]:2525 apikey:nuestra-api-key >> /etc/postfix/sasl_passwd

Usamos la utilidad postmap para generar un archivo .db:

$ postmap /etc/postfix/sasl_passwd

Quitamos el archivo que contiene nuestras credenciales, ya que no lo necesitaremos, con el comando siguiente:

$ rm /etc/postfix/sasl_passwd

Configuramos los permisos en nuestro archivo .db:

$ chmod 600 /etc/postfix/sasl_passwd.db

Ejecutamos el siguiente comando para reiniciar el servicio postfix:

$ sudo systemctl restart postfix.service

3. Servidor de base de datos

3.1. PostgreSQL

PostgreSQL es un sistema de base de datos relacional de objetos que tiene las características de los sistemas de bases de datos comerciales tradicionales con mejoras que se encuentran en los sistemas DBMS de próxima generación.

3.1.1. Instalación

Para instalar PostgreSQL, ejecutamos el siguiente comando en el símbolo del sistema:

$ sudo apt install postgresql

El servicio de base de datos se configura automáticamente con valores predeterminados viables, pero se puede personalizar en función de nuestras necesidades especializadas.

3.1.2. Configuración

PostgreSQL admite varios métodos de autenticación de clientes. De forma predeterminada, el método de autenticación IDENT se usa para postgres y usuarios locales.

La siguiente discusión asume que desea habilitar las conexiones TCP/IP y utilizar el método MD5 para la autenticación del cliente. Los archivos de configuración de PostgreSQL se almacenan en el directorio **/etc/postgresql/<version>/main**.

Para permitir que otras computadoras se conecten a nuestro servidor PostgreSQL, editamos el archivo **/etc/postgresql/12/main/postgresql.conf**.

Buscamos la línea **#listen_addresses = 'localhost'** y la cambiamos a:

listen_addresses = '*'

Ahora que podemos conectarnos a nuestro servidor PostgreSQL, el siguiente paso es establecer una contraseña para el usuario de Postgres. Ejecutamos el siguiente comando en un indicador de terminal para conectarse a la base de datos de plantilla de PostgreSQL predeterminada:

$ sudo -u postgres psql template1

El comando anterior se conecta a la plantilla1 de la base de datos de PostgreSQL como usuario postgres. Una vez que se conecte al servidor PostgreSQL, estará en un indicador SQL. Podemos ejecutar el siguiente comando SQL en el símbolo del sistema psql para configurar la contraseña para el usuario postgres.

ALTER USER postgres with encrypted password 'our_password';

Después de configurar la contraseña, editamos el archivo **/etc/postgresql/12/main/pg_hba.conf** para usar la autenticación MD5 con el usuario de postgres:

```
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                md5
```

Finalmente, debemos reiniciar el servicio PostgreSQL para inicializar la nueva configuración. Desde un indicador de terminal, ingresamos lo siguiente para reiniciar PostgreSQL:

$ sudo systemctl restart postgresql.service

Podemos probar las conexiones del servidor desde otras máquinas utilizando el cliente PostgreSQL.

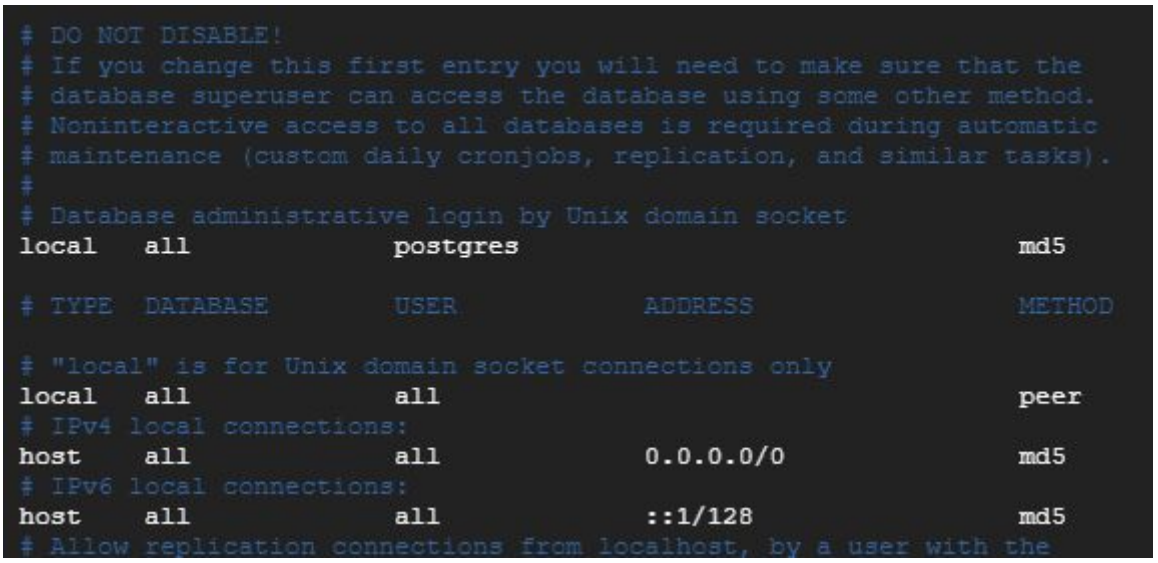
$ sudo apt install postgresql-client

$ psql -h testing06.com -U postgres -W

Creamos un usuario diferente al cual ya teníamos configurado pero el cuál no tendrá capacidad de crear base de datos, solo podrá leer y escribir datos en una base de datos ya creada.

psql -> CREATE USER docente WITH PASSWORD 'tecnoweb';

Editamos el archivo **/etc/posgresql/12/main/pg_hba.conf** para permitir conexiones desde otros dispositivos.



```
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# Database administrative login by Unix domain socket
local   all             postgres                                md5

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     peer
# IPv4 local connections:
host    all             all             0.0.0.0/0               md5
# IPv6 local connections:
host    all             all             ::1/128                 md5
# Allow replication connections from localhost, by a user with the
```

Reiniciamos el servicio:

$ sudo systemctl restart postgresql.service

Ahora podemos ingresar con las credenciales de “docente” a una base de datos del servidor.

$ psql -h testing06.com -d test -U docente -W

4. Servidor web

La función principal de un servidor web es almacenar, procesar y entregar páginas web a los clientes. Los clientes se comunican con el servidor enviando solicitudes HTTP. Los clientes, en su mayoría a través de navegadores web, solicitan recursos específicos y el servidor responde con el contenido de ese recurso o con un mensaje de error. La respuesta suele ser una página web, como documentos HTML, que pueden incluir imágenes, hojas de estilo, scripts y el contenido en forma de texto.

Al acceder a un servidor web, cada solicitud HTTP que se recibe se responde con un contenido y un código de estado HTTP. Los códigos de estado HTTP son códigos de tres dígitos y se agrupan en cinco clases diferentes. La clase de un código de estado se puede identificar rápidamente por su primer dígito:

- **1xx**: Informativo: solicitud recibida, continuando el proceso
- **2xx**: Éxito: la acción se recibió, comprendió y aceptó correctamente
- **3xx**: redireccionamiento: se deben realizar más acciones para completar la solicitud
- **4xx**: Error del cliente: la solicitud contiene una sintaxis incorrecta o no se puede cumplir

- **5xx**: Error del servidor: el servidor no pudo cumplir con una solicitud aparentemente válida

4.1. Apache2

Apache es el servidor web más utilizado en sistemas Linux. Los servidores web se utilizan para servir las páginas web solicitadas por los equipos cliente. Los clientes suelen solicitar y ver páginas web mediante aplicaciones de navegador web como Firefox, Opera, Chromium o Internet Explorer.

Los usuarios ingresan un localizador uniforme de recursos (URL) para apuntar a un servidor web mediante su nombre de dominio completo (FQDN) y una ruta al recurso requerido.

El protocolo más común utilizado para transferir páginas web es el Protocolo de transferencia de hipertexto (HTTP). También se admiten protocolos como el Protocolo de transferencia de hipertexto sobre la capa de conexión segura (HTTPS) y el Protocolo de transferencia de archivos (FTP), un protocolo para cargar y descargar archivos.

Los servidores web Apache se utilizan a menudo en combinación con el motor de base de datos MySQL, el lenguaje de scripting HyperText Preprocessor (PHP) y otros lenguajes de scripting populares como Python y Perl. Esta configuración se denomina LAMP (Linux, Apache, MySQL y Perl

4.1.1. Instalación

El servidor web Apache2 está disponible en Ubuntu Linux. Para instalar Apache2: En un indicador de terminal, ingrese el siguiente comando:

Apache2 se configura colocando directivas en archivos de configuración de texto plano. Estas directivas están separadas entre los siguientes archivos y directorios:

$ sudo apt install apache2

4.1.2. Configuración

Apache2 se configura colocando directivas en archivos de configuración de texto sin formato. Estas directivas están separadas entre los siguientes archivos y directorios:

Además, se pueden agregar otros archivos de configuración usando la directiva Incluir, y se pueden usar comodines para incluir muchos archivos de configuración. Cualquier directiva puede colocarse en cualquiera de estos archivos de configuración. Apache2 solo reconoce los cambios en los archivos de configuración principales cuando se inicia o reinicia.

- apache2.conf: el archivo de configuración principal de Apache2. Contiene configuraciones que son globales para Apache2.
- httpd.conf: históricamente, el archivo de configuración principal de Apache2, llamado así por el demonio httpd. En otras distribuciones (o versiones anteriores de Ubuntu), el archivo puede estar presente. En Ubuntu, todas las opciones de configuración se han movido a apache2.conf y los directorios a los que se hace referencia a continuación, y este archivo ya no existe.
- conf-available: este directorio contiene archivos de configuración disponibles. Todos los archivos que estaban previamente en **/etc/apache2/conf.d** deben moverse a **/etc/apache2/conf-available**.

- conf-enabled: contiene enlaces simbólicos a los archivos en **/etc/apache2/conf-available**. Cuando un archivo de configuración tiene un enlace simbólico, se habilitará la próxima vez que se reinicie apache2.
- envvars: archivo donde se establecen las variables de entorno de Apache2.
- mods-available: este directorio contiene archivos de configuración para cargar módulos y configurarlos. Sin embargo, no todos los módulos tendrán archivos de configuración específicos.
- mods-enabled: contiene enlaces simbólicos a los archivos en **/etc/apache2/mods-available**. Cuando un archivo de configuración de módulo tiene un enlace simbólico, se habilitará la próxima vez que se reinicie apache2.
- ports.conf: aloja las directivas que determinan en qué puertos TCP está escuchando Apache2.
- sites-available: este directorio tiene archivos de configuración para los hosts virtuales Apache2. Los hosts virtuales permiten que Apache2 se configure para varios sitios que tienen configuraciones independientes.
- sites-enabled: al igual que los mods habilitados, los sitios habilitados contienen enlaces simbólicos al directorio / etc / apache2 / sites-available. De manera similar, cuando un archivo de configuración en sitios disponibles tiene un enlace simbólico, el sitio configurado por él estará activo una vez que se reinicie Apache2.
- magic: instrucciones para determinar el tipo MIME en función de los primeros bytes de un archivo.

Además, se pueden agregar otros archivos de configuración usando la directiva Incluir, y se pueden usar comodines para incluir muchos archivos de configuración. Cualquier directiva puede colocarse en cualquiera de estos archivos de configuración. Apache2 solo reconoce los cambios en los archivos de configuración principales cuando se inicia o reinicia.
Por razones de seguridad no es una buena idea difundir las versiones de software que se está ejecutando. Para eso, añadiremos la siguiente línea al archivo **/etc/apache2/apache2.conf**:

Por razones de seguridad no es una buena idea difundir las versiones de software que se está ejecutando. Para eso, añadiremos la siguiente línea al archivo **/etc/apache2/apache2.conf**:

ServerTokens Prod

Reiniciamos el servicio:

$ systemctl restart apache2.service

Si se tiene información en nuestra página web que sea información sensible o pensada para un grupo reducido de usuarios/personas, las técnicas que se describen en este manual, le servirán de ayuda para asegurarse de que las personas que ven esas páginas sean las personas que uno quiere.

Aquí está lo básico de cómo proteger con contraseña un directorio en nuestro servidor.

Primero, necesitaremos crear un fichero de contraseña. Dependiendo de que proveedor de autenticación se haya elegido, se hará de una forma u otra. Para empezar, usaremos un fichero de contraseña de tipo texto.

Este fichero deberá estar en un sitio que no se pueda tener acceso desde la web. Esto también implica que nadie pueda descargarse el fichero de contraseñas. Por ejemplo, si tus documentos están guardados fuera de **/usr/local/apache/htdocs**, querrás poner tu archivo de contraseñas en **/usr/local/apache/passwd**.

Para crear el fichero de contraseñas, usaremos la utilidad htpasswd que viene con Apache. Esta herramienta se encuentra en el directorio **/bin** en donde sea que se ha instalado el Apache. Si ha instalado Apache desde un paquete de terceros, puede ser que se encuentre en su ruta de ejecución.

Para crear el fichero, escribiremos:

$ htpasswd -c /usr/local/apache/passwd/passwords docente

htpasswd nos preguntará por una contraseña, y después nos pedirá que la volvamos a escribir para confirmarla.

Lo próximo que necesitaremos, será configurar el servidor para que pida una contraseña y así decirle al servidor que usuarios están autorizados a acceder. Podemos hacer esto ya sea editando el fichero httpd.conf de configuración o usando in fichero .htaccess. Por ejemplo, si quieres proteger el directorio **/usr/local/apache/htdocs/secret**, puedes usar las siguientes directivas, ya sea en el fichero .htaccess localizado en following directives, either placed in the file **/usr/local/apache/htdocs/secret/.htaccess**, o en la configuración global del servidor httpd.conf dentro de la sección **<Directory "/usr/local/apache/htdocs/secret">** , como se muestra a continuación:

```
Alias /test /var/www/test
<Directory /var/www/test>
        Options Indexes FollowSymLinks
        AuthType Basic
        AuthName "Archivos restringidos"
        AuthUserFile "/usr/local/apache/passwd/passwords"
        Require user docente
</Directory>
```

Reiniciamos el servicio:

$ systemctl restart apache2.service

5. Configuración SSL/TLS

5.1. Apache

Para aceptar conexiones mediante HTTPS usaremos certbot. Debemos instalar snapd y asegurarnos de seguir las instrucciones para habilitar el soporte de snap clásico.

$ sudo apt install snapd

Ejecutamos las siguientes instrucciones en la línea de comando de la máquina para asegurarnos de tener la última versión de snapd.

$ sudo snap install --classic certbot

Ejecutamos la siguiente instrucción en la línea de comando de la máquina para asegurarse de que se pueda ejecutar el comando certbot.

$ sudo ln -s /snap/bin/certbot /usr/bin/certbot

Ejecutamos este comando para obtener un certificado y que Certbot edite nuestra configuración de Apache automáticamente para servirla, activando el acceso HTTPS en un solo paso.

$ sudo certbot –apache

Los paquetes de Certbot en nuestro sistema vienen con un trabajo cron o un temporizador systemd que renovará nuestros certificados automáticamente antes de que caduquen. No necesitaremos ejecutar Certbot nuevamente, a menos que cambiemos su configuración. Podemos probar la renovación automática de sus certificados ejecutando este comando:

$ sudo certbot renew --dry-run

Reiniciamos el servicio:

$ systemctl restart apache2.service

5.2. Dovecot

Editamos el archivo **/etc/dovecot/conf.d/10-ssl.conf** y editamos las siguientes líneas:

ssl_cert = </etc/letsencrypt/live/testing06.com/fullchain.pem

ssl_key = </etc/letsencrypt/live/testing06.com/privkey.pem

Reiniciamos el servicio:

$ systemctl restart dovecot.service

5.3. Postfix

Editamos el archivo **/etc/postfix/main.cf** y editamos las siguientes líneas:

smtpd_tls_cert_file = /etc/letsencrypt/live/testing06.com/fullchain.pem

smtpd_tls_key_file = /etc/letsencrypt/live/testing06.com/privkey.pem

Reiniciamos el servicio:

$ systemctl restart postfix.service

5.4. PostgreSQL

Copiamos los archivos generados por Certbot en una carpeta que pueda ser leída por el usuario postgres. Ejecutamos las siguientes líneas:

cp /etc/letsencrypt/live/testing06.com/privkey.pem /etc/postgresql/12/letsencrypt/fullchain.pem

cp /etc/letsencrypt/live/testing06.com/privkey.pem /etc/postgresql/12/letsencrypt/privkey.pem

Editamos el archivo **/etc/postgresql/12/main/postgresql.conf** y editamos las siguientes líneas:

ssl_cert_file = '/etc/postgresql/12/letsencrypt/fullchain.pem'

ssl_key_file = '/etc/postgresql/12/letsencrypt/privkey.pem'

Reiniciamos el servicio:

$ systemctl restart postgresql.service