Glosario

Capítulo que va a contener varias definiciones de conceptos teóricos que pueden ser necesarias de aclarar para las distintas cosas a llevar a cabo

Direcciones IP

> Para este artículo, y por simplicidad del tema, vamos a estar hablando de IPv4, y no de IPv6, ya que esta última no desplazó a IPv4 todavía, y es la que se nos asigna en la mayoría de los casos a quienes usamos computadoras conectadas a internet.

> Si no sabés de qué habla lo de arriba, no te preocupes :) Seguí leyendo

Qué son las direcciones IP?

Cuando las computadoras se conectan a internet, o a cualquier red, utilizan una "dirección", como la dirección de tu casa, única que les permite identificarse ante el mundo y compartir información con otras computadoras. Desde ya que las otras computadoras también tienen direcciones como la tuya. Estas direcciones se llaman **direcciones IP**. Su formato son cuatro números del 0 al 255 separados por puntos, por ejemplo: `155.42.3.23`.

El problema con las IPs

Sin embargo, las IP tienen una limitación. Al sólo poder ser estos cuatro números, y al sólo poder ir del 0 al 255, tienen cómo límite 4.300 *millones* de combinaciones. Esto parece un montón, pero recordemos: Cada "computadora", cada aparato electrónico que se pueda conectar a internet, debería tener una. Esto incluye:

- Celulares
- Relojes "inteligentes"
- Lamparitas "inteligentes"
- Heladeras
- Sensores
- Y muchas otras cosas

Por esto, y sobre todo desde que se empezaron a crear cada vez más tipo de dispositivos conectados a interent, empezó a crecer la preocupación por que se acaben las direcciones de IP. Para solucionar este problema, se decidió separar una cantidad de rangos de IP que no podrían ser utilizados en el internet, y se reservarían para redes locales.

IPs locales

Como solución, estos rangos de IP sólo podrían ser usados en las redes de cada casa, empresa, o lo que sea, y cuando se conectasen a internet, lo harían *con la IP del router*. De esta forma, una red de casa que conteniese, por ejemplo, 200 dispositivos conectados, sólo tendría una sóla IP a través de la cual se conectaría a internet. Y el dispositivo encargado de hacerlo(el router), tendría la responsabilidad de, cada vez que una computadora externa se conecte con la IP de la casa, "derivar" a la computadora externa hacia la IP local que le corresponda.

Cómo sabemos si una IP es local o de internet

Como mencioné más arriba, las IPs locales pertenecen a un rango específico. Estos son:

- 10.0.0.0 - 10.255.255.255
- 172.16.0.0 - 172.31.255.255
- 192.168.0.0 - 192.168.255.255 <- Estas son las que más vi

Por lo tanto, por ejemplo, la dirección `192.168.10.46`sería una dirección local, igual que `172.20.0.157`, mientras que direcciones como `172.40.54.23` o `192.170.23.4` **no lo serían**.

Puertos

Los puertos, o *ports*, en inglés, son esencialmente puntos de conexión de nuestras computadoras hacia otras computadoras. **Todas las computadoras se conectan entre sí mediante puertos**. Es decir, cualquier conexión que ocurra de una computadora a otra va a ocurrir mediante puertos.

Rango de puertos

Los puertos están numerados: Pueden ser desde el puerto 0 al puerto 65535. Sin embargo, no todos estos puertos son iguales:

- 1-1023 Puertos conocidos: Son puertos reservados para servicios fundamentales, como e-mail, http o https(los protocolos de internet)
- 1024-49151 Puertos registrados: Los puertos "de uso común" por varios programas. No tienen un uso específico como los anteriores, y son los puertos que normalmente usan las varias aplicaciones que podemos correr en nuestra computadora.
- 49152-65535 Puertos dinámicos: Son puertos que utilizan varios servicios de nuestra computadora automáticamente. No conviene que los usemos nosotrxs manualmente.

Ejemplos de puertos conocidos y usados

Puerto	Servicio	Descripción
80	http	Protocolo de internet
443	https	Protocolo de internet(seguro, el candadito
22	ssh	Protocolo de "terminal remota" secure shell
21	ftp	Protocolo de FTP(File transfer protocol)
25	smtp	Protocolo de e-mail
7878	radarr	Puerto por defecto utilizado por el programa "radarr"
8096	jellyfin	Puerto por defecto de la aplicación "Jellyfin"

Vale destacar que los últimos dos puertos del ejemplo, al ser puertos de la sección de "puertos registrados", pueden llegar a ser usados por otros progamas también. Cuando se instala, por ejemplo, Jellyfin, si el puerto ya está en uso, se puede aclarar al programa que debería utilizar otro.

Dominios

Los dominios son las "direcciones legibles por humanos". Son, por ejemplo `google.com`, que nos dirige a la IP de google. Los dominios están distribuídos a distintas empresas que los alquilan, o a distintos países que hacen igual. Los dominios de máximo nivel (Top Level Domain, o TLD), por ejemplo, ".com.ar" y ".ar", le corresponden al Estado Argentino, y se consiguen en la página "nic.ar ". Hay muchos dominios que son vendidos por entidades privadas, como los dominios ".xyz", ".lat", o incluso los ".com". Páginas que normalmente venden dominios incluyen:

- Namecheap
- Porkbun
- Cloudflare

Permisos en Linux(y UNIX)

> Esta página aún no fue escrita

Sinopsis

Permisos

Root

SUDO

La Terminal

La terminal (a veces llamada tambíen "consola" o "command line interface"(cli)) es la principal forma de interactuar que vamos a tener con nuestro servidor, especialmente si éste no tiene una interfaz gráfica.

En realidad, todo corre a través de la consola; la interfaz gráfica simplemente nos genera un entorno más "amigable" y, bueno, gráfico, para interactuar con ella, en lugar de "comandos de texto". (Envía dichos comandos de texto por nosotrxs)

Tutorial: conceptos básicos de la consola

Si estás acá es porque ya estamos manejándonos usando la consola como principal forma de interactuar con el server. No paniquees! Es mucho más simple de lo que parece. Ante todo, si querés un tutorial un poco más completo, aunque en inglés, recomiendo este de acá. Si preferís algo más superficial, sigamos!

Qué es la consola?

La consola, simplemente, es un programa que nos permite interactuar con la computadora a través de comandos de texto. Para abrir una, en Linux o Mac, simplemente tenés que buscar el programa "Terminal". En caso de que tengas Windows, la "consola" que tienen ellos se va a llamar la Command Prompt (en español, si no me equivoco, "Símbolo del Sistema", y en ambos idiomas se encuentra como "CMD", también).

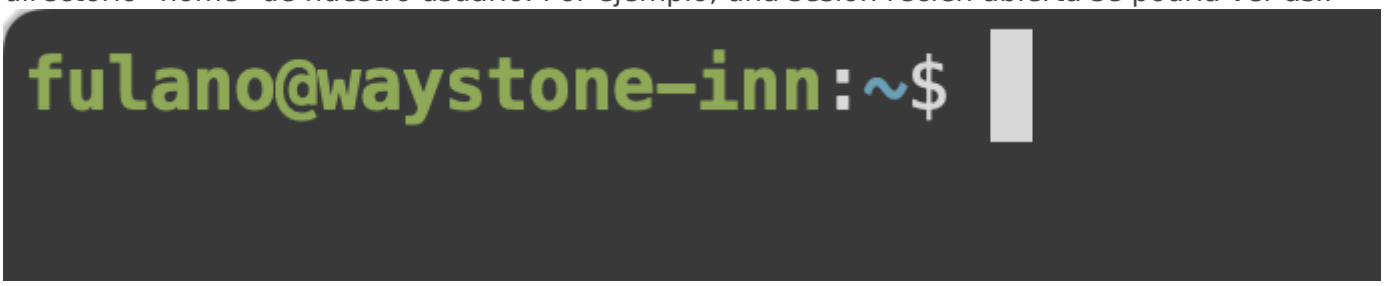
> **IMPORTANTE!** En este tutorial, todo lo que veamos va a aplicar a consolas de Linux o macOS (sistemas operativos basados en "UNIX", un sistema operativo abuelito suyo). Si querés seguir el tutorial desde Windows, te recomiendo que busques el "WSL" (Windows Subsystem for Linux), o que lo hagas ya desde una conexión "ssh" con tu servidor.

Si te conectaste por [ssh], felicitaciones! Ya estás usando una terminal.

Lo fundamental

Cuando trabajamos en una terminal (sea con una [conexión remota mediante SSH], o abriendo un "emulador de terminal" (a veces llamado simplemente terminal) o en la consola misma de la

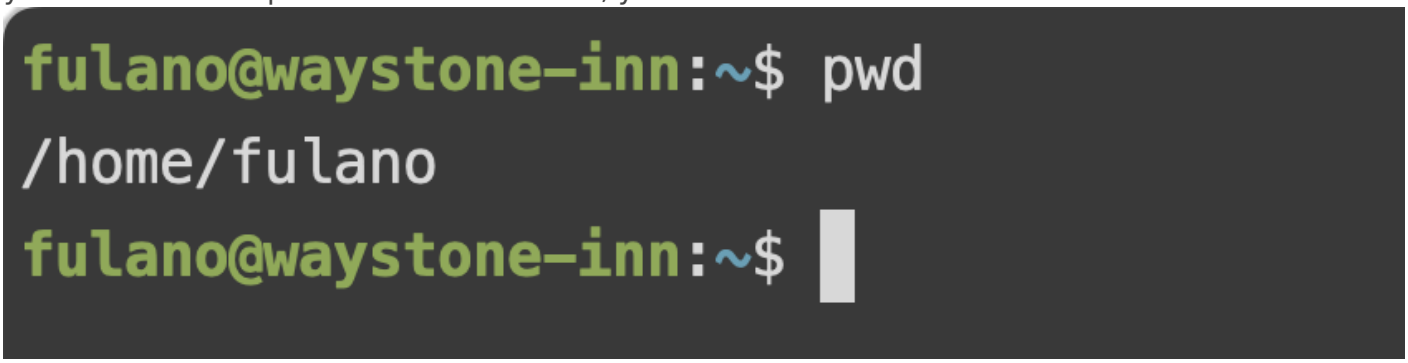
computadora, estamos esencialmente usando un programa que espera que corramos "comandos" indicándole qué hacer. Este programa funciona de forma similar al navegador "explorer" de Windows o el "finder" de mac, en un sentido: Los comandos que corramos, va a correrlos en el contexto de una ubicación específica. Cuando recién lo abrimos, esa ubicación suele ser el directorio "home" de nuestro usuario. Por ejemplo, una sesión recién abierta se podría ver así:



Parecen cosas al azar, pero en realidad, la consola nos muestra toda información importante para tener en cuenta:

- Al comienzo, y la izquierda de la `@`, podemos ver qué usuario está usando la consola (con quién nos logueamos). Este usuario, por ejemplo, se llama `fulano`.
- Del otro lado de la `@` (que en inglés significia "at", es decir, "en"), tenemos el nombre de la computadora a la que estamos conectadxs, en este caso `waystone-inn`. Si leemos ambas secciones juntas, se podrían traducir a "`fulano` en la computadora `waystone-inn`".
- Después de los dos puntos, tenemos la dirección en la que estamos trabajando. El símbolo `~` simboliza nuestro "home", o carpeta "hogar" de nuestro usuario. Esta carpeta es sólo nuestra, y ningún otro usuario (salvo lxs administradorxs) puede acceder a ella a menos que decidamos permitírselo.
- Finalmente, el símbolo `$` representa que esta sesión es de un usuario común, que no es un administrador (en realidad, que no estamos actuando a través del usuario especial `root`, que es un usuario cuyos todos comandos son corridos como de administración. En ese caso, veríamos un `#` en su lugar).
- Finalmente finalmente, vemos el cursor de texto, esperando que escribamos algo.

Qué pasaría si corremos un comando? Probemos con "pwd", que como su nombre indica (**p**ath to **w**orking **d**irectory), nos muestra cúal es el "camino" hacia el directorio de la carpeta en la que estamos trabajando siguiendo la [[estructura jerárquica de directorios]] de Linux. Ingresamos `pwd`, y tocamos "enter" para enviar el comando, y obtenemos esto:

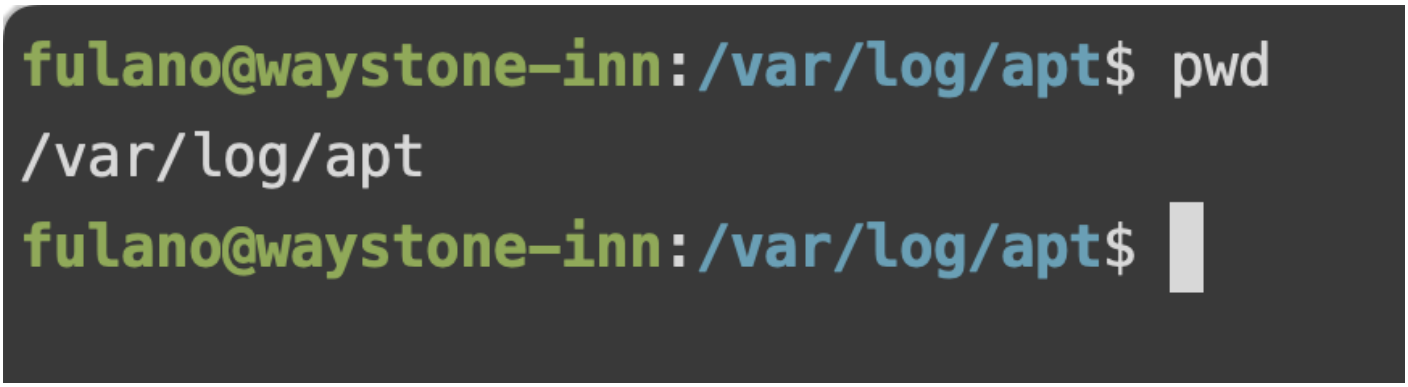


Después de tocar "enter", la computadora nos responde con la dirección al directorio actual (`/home/fulano`, es decir que desde la carpeta base, conocida como `/`, entraríamos a la carpeta `home`, y dentro de ésta estaría la carpeta `fulano`), y en otra línea nos figura la misma info que antes, simbolizando que nuevamente tenemos control sobre la computadora (y nuevamente está

esperando una indicación o comando).

Es importante notar que cuando ingresamos "pwd", no tuvimos que aclarar a dónde era esto, ya que la consola dio por sentado que nos referíamos al directorio en el que nos encontrábamos. Veamos un ejemplo distinto. Si la consola nos mostrase, después de los dos punto, esto:

Nos estaría indicando que no estamos trabajando el el directorio "home" de nuestro usuario, sino en el directorio que se encuentra en la ruta `/var/log/apt`. Si corriésemos el comando `pwd`, por más redundante que sea, nos mostraría como resultado que, en efecto, ese es el "camino" hacia la carpeta en la que estamos trabajando, y que la consola da por sentado que estamos corriendo el comando *desde esa ubicación en los directorios del sistema*:



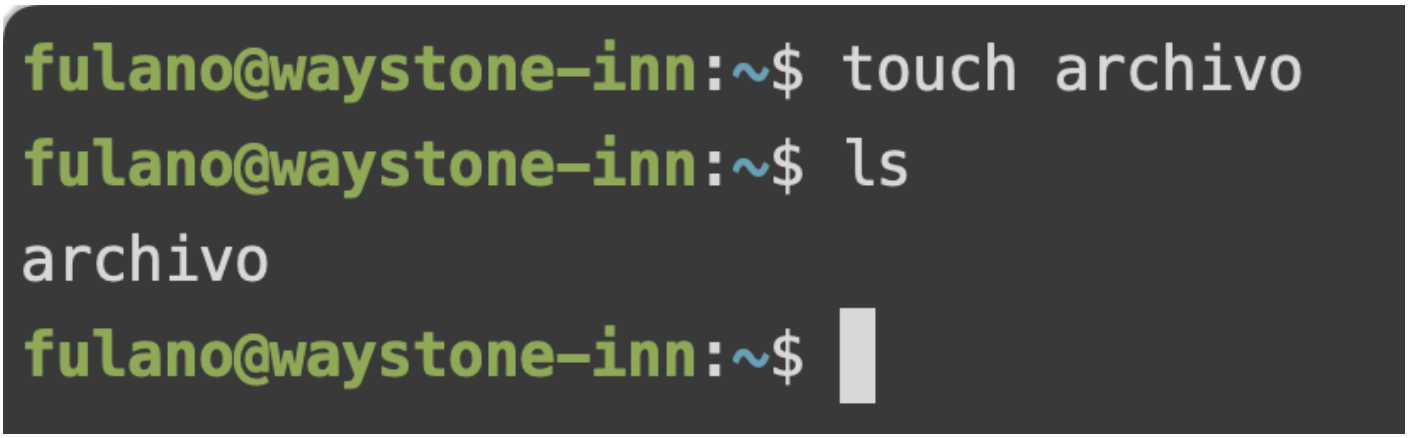
.

De la misma forma, y volviendo al directorio "home" (que figura como un `~` en nuestra consola, para abreviar), si usásemos el comando `touch`, que sirve para crear un archivo vacío, y se utiliza escribiendo `touch <nombre del archivo>`, el archivo creado lo encontraríamos en el actual directorio:

Un par de cosas para notar:

- Los archivos en Linux no *necesitan* llevar una extensión. Por lo general, si no indican nada, suelen ser archivos de texto, aunque a veces son binarios también. No nos importa mucho esa distinción, sí nos importa saber que **muchos programas usan la extensión para saber cómo tratar al archivo**, así que, aunque no sea necesaria (como en el ejemplo), es mejor ponerla por las dudas (en este caso, podría haber agregado `.txt` al final, por ejemplo).
- En Linux, si un comando no te dice nada cuando lo corrés, y asumiendo que es un comando que no nos da información específicamente (como `pwd`), *significa que el comando se corrió con éxito`.

Pero, bancá un segundo. Cómo sabemos que se creó el archivo? Bueno, es que el comando `touch` sólo crea el archivo, y siguiendo la filosofía Unix de *crear herramientas que sólo hagan una cosa y la hagan bien*, no hace más nada. Ni siquiera nos muestra el archivo. Para verlo, podemos usar el comando `ls`, que nos **ls**ta los archivos y directorios dentro del directorio en el que nos encontramos:



Podemos ver, ahora, que el archivo está en la carpeta en la que estamos (y, de paso, que es el único archivo presente).

Más abajo vamos a ver, entonces, varios comandos que podemos utilizar en la consola para navegarla y modificarla, creando archivos, copiándolos, cambiándoles el nombre y más.

Navegando

Comando `cd`

Comando `ls`

Creando archivos y directorios

Comando `mkdir`

Comando `touch`

Copiando, moviendo y eliminando

Comando `mv`

Comando `cp`

Comando `rm`

Editando archivos de texto

Comando `nano`

SSH

El SSH (Secure SHell) es un protocolo para acceder al "shell" (la consola o terminal, en pocas palabras) de una computadora de manera remota. Es la principal forma en la que vamos a interactuar con nuestro servidor, ya que es más cómodo por lo general acceder y configurar las cosas desde nuestra computadora de siempre y tener el server guardado en algún recoveco donde no moleste.

Acceder a un servidor a través de SSH

Usar SSH desde una Mac o desde una computadora con Linux es extremadamente simple, ya que el programa para hacerlo suele venir preinstalado en las mismas. En el caso de Windows, si usás Windows 10 u 11 (y los tenés al día), tambíen deberías poder hacer algo parecido. Si no, tendrías que instalar un cliente de SSH, como putty.

En caso de tener ssh instalado, el proceso es simple (y si no lo tenés instalado, dejo en vos googlear como instalar el cliente ssh en tu sistema operativo (el de tu compu de siempre, no el del server):

Abriendo la terminal (o símbolo de sistema/terminal en Windows), ingresamos el siguiente comando:

```
ssh <usuario>@<ip del servidor>
```

Al hacerlo, nos va a preguntar si queremos "guardar el fingerprint" del servidor al que accedemos. Indicamos que sí (esto va a pasar únicamente cuando accedemos por primera vez a una computadora nueva), y nos va a pedir la contraseña. Una vez que la ingresemos, vamos a haber accedido a una consola remota del servidor. Desde ahí, vamos a poder hacer todo lo que haríamos sentados frente a una terminal en la misma computadora.

Si no sabés qué hacer acá, y nunca usaste una terminal antes, podés ver los conceptos básicos acá, o leer la guía más completa (y muy buena y didáctica) que recomiendo en esa misma página, acá.

Gestor de paquetes (Package manager)

Gestor de paquetes

Qué es un gestor de paquetes?

En la mayoría de las distribuciones de Linux se incluye un "Package manager" o "Gestor de paquetes". Esto es un programa que, esencialmente, se encarga de bajar otros programas por nosotros. Sería el antecesor de lo que hoy son las "App Stores" en los celulares. Al igual que las app stores, además de bajar e instalar paquetes, este programa se utiliza también para actualizar, mantener y eliminar las cosas que instalamos con ellos. En ese sentido, nos simplifica un poco las cosas: En lugar de tener que bajar archivos, instalar drivers y dependencias, y demás, sólo tenemos que bajar un cierto paquete y el programa instala todo lo necesario para que funcione.

apt

`apt` es el gestor de paquetes instalado por defecto en Ubuntu, y en todos los distros de Linux basados en Debian (como Ubuntu lo es). Vamos a usarlo seguido en nuestro servidor, y frecuentemente, ya que los siguientes dos comandos, `apt update` y `apt upgrade`, son los que se encargan de actualizar todos los paquetes (programas) que instalemos a través de este método.

Ambos comandos deben ser ejecutados como administrador. El comando normalmente escrito para actualizar sería:

```
sudo apt update && sudo apt upgrade
```

(el && significa que, si el primer comando se corrió con éxito, se pueda correr el segundo)

Para instalar los paquetes sin que nos pregunte si aceptamos instalar cada uno, podemos correr en cambio:

```
sudo apt update && sudo apt upgrade -y
```

Docker

> Este artículo todavía no existe :3

Docker

Acá va a ir la definición de Docker

Docker compose

Acá voy a explicar cómo funciona docker compose