TICS565 - Técnicas para Big Data

Manejo de Servidores e Instalación de PSQL

1. Introducción

En esta guía vamos a repasar brevemente los conceptos que necesitamos para manejar un servidor remoto (e.g. Un servidor de Digital Ocean o EC2 de AWS). El objetivo es que después sea más fácil seguir el tutorial para distribuir una base de datos Postgres. Primero vamos a partir por un resumen de los comandos básicos, para luego repasar la instalación de una base de datos PSQL. **Importante**: esta guía se realizó considerando un servidor Ubuntu 20.04.

2. Comandos básicos

A continuación presentamos un resumen de los comándos más importantes al usar una terminal.

- `cd`: este comando nos permite cambiarnos de directorio. Por ejemplo al correr:

  ```
  cd /Users/juan/Desktop
  ```

  Nos estamos moviendo a la carpeta `Desktop` ubicada en `/Users/juan/`. Aquí estamos entregando en una ruta absoluta. Pero también podemos entregar una ruta relativa:

  ```
  cd miCarpeta
  ```

  Esto es, movernos a la carpeta `miCarpeta` ubicada en el directorio en que nos encontramos. También para volver a la carpeta padre podemos usar:

  ```
  cd ..
  ```

- `ls`: este comando muestra todas las carpetas en el directorio en que nos encontramos.
- `pwd`: este comando nos muestra la ruta absoluta del directorio en el que nos encontramos.
- `mv`: este comando nos sirve para renombrar archivos o para cambiarlos de ubicación. Por ejemplo al escribir:

```
mv archivo1.txt archivo2.txt
```

Estamos cambiando el nombre del archivo `archivo1.txt` a `archivo2.txt`, en la carpeta donde estamos posicionados. Esto también sirve para carpetas, y con rutas absolutas.

- `mkdir`: este comando nos sirve para crear una carpeta. Por ejemplo al escribir:

  ```
  mkdir miCarpeta
  ```

  Estamos creando una carpeta llamada `miCarpeta` en el directorio en el que nos encontramos.

- `head`: este comando nos sirve para ver las primeras líneas de un archivo. Por ejemplo al correr:

  ```
  head -20 archivo.txt
  ```

  Estamos mostrando las primeras 20 líneas del archivo `archivo.txt`.

- `cp`: este comando nos sirve para copiar un archivo. Funciona de forma similar al comando `mv`:

  ```
  cp archivo1.txt archivo2.txt
  ```

  Aquí estamos copiando el archivo `archivo1.txt` a un archivo `archivo2.txt`. Funciona también con rutas absolutas.

- `rm`: este comando nos sirve para eliminar archivos. Por ejemplo al escribir:

  ```
  rm archivo1.txt
  ```

  Estamos eliminando el archivo con el nombre `archivo1.txt`. Si queremos eliminar una carpeta tenemos que entregarle otras opciones, por ejemplo:

  ```
  rm -rf miCarpeta
  ```

  Elimina la carpeta `miCarpeta`, del directorio en el que me encuentro. El `-rf` indica que queremos que sea recursivo (y elimine sub-carpetas y archivos dentro de estas sub-carpetas) y que fuerce a eliminar los archivos (esto es, que no nos pregunte si estamos seguros). Hay que tener cuidado de no eliminar algo que no queremos, ya que no es reversible. Funciona también con rutas absolutas.

2.1. Otros comandos importantes

Otro comando importante que vamos a utilizar es `nano`. Este comando nos abre un editor de texto en terminal. Es relativamente intuitivo de usar, salvo que para salir de él debemos apretar `ctrl+X` y además decirle si queremos o no guardar el archivo. También hay otros comandos de este estilo en este editor.

Otro comando util es `tmux`. Este comando nos permite abrir sub-ventanas en la terminal, además de poder dejar corriendo programas en *background*. Este programa debemos instalarlo con el comando:

```
sudo apt install tmux
```

Finalmente, otros comandos a destacar son `htop` y `micro`. El primero es una forma de monitorear la actividad del computador, mientras que el segundo es un editor de texto que permite interacción con el *mouse*.

3. Configurando la terminal

Para trabajar de mejor forma con la terminal, es recomendable instalar `zsh` (es una terminal que remplaza a `bash`, la que viene por defecto). Para instalar `zsh` debemos escribir el comando:

```
sudo apt install zsh
```

Y luego, indicar que queremos que esta sea nuestra terminal por defecto:

```
chsh -s /usr/bin/zsh
```

Una vez con `zsh` instalado, recomendamos salir del servidor y volver a entrar (para salir usamos el comando `exit`). Luego, al volver a entrar tenemos que instalar *Oh My Zsh*, que es un *framework* para administrar `zsh`. Para esto, usamos el comando:

```
sh -c "$(curl -fsSL https://raw.github.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Ahora vamos a configurar `zsh`. Para esto, tenemos que abrir el archivo de configuración con el comando:

```
nano ~/.zshrc
```

Ahí editamos la variable `ZSH_THEME` según el tema que queramos usar. En lo personal, mi favorito es `fino`, así que esa parte se vería así:

```
ZSH_THEME="fino"
```

Ahora vamos a instalar dos *plugins* que facilitan nuestro trabajo.

3.1. Plugins (Opcional)

El primer *plugin* es **History Substring Search**. Este *plugin* sirve cuando buscamos comandos ejecutados anteriormente, para buscar solo los que contengan determinado *substring*. Las instrucciones de instalación están aquí:

`https://github.com/zsh-users/zsh-history-substring-search`

Ojo que tendremos que editar el archivo de configuración `/.zshrc` y agregar este *plugin*.

Nuestro segundo *plugin* es **Zsh Syntax Highlightning**, que nos sirve para resaltar ciertos comandos (y tener una guía de si estamos escribiendo el comando bien o no). Las instrucciones para la instalación están aquí:

`https://github.com/zsh-users/zsh-syntax-highlighting`

4. Instalación de PostgreSQL

Ahora, para instalar PostgreSQL debemos correr el siguiente comando (para actualizar el índice de los distintos *packages*):

```
sudo apt update
```

Y ahora para instalar Postgres:

```
sudo apt install postgresql postgresql-contrib
```

Ahora para ingresar a Postgres, debemos ingresar con el usuario `postgres`, que es el único que está creado. Para esto corremos el comando:

```
sudo -i -u postgres
```

Que nos sirve para *logearnos* en la terminal como el usuario `postgres`. Ahora si escribimos:

```
psql
```

Vamos a ingresar a Postgres[1]. El comando lo que hace es *logearse* como el usuario `postgres` a la base de datos llamada `postgres`. Ahora bien, es conveniente crear un usuario distinto, junto con una base de datos fuera a la por defecto. Para esto, en la terminal (una vez fuera de Postgres, y fuera del usuario `postgres`[2]) tenemos que correr el comando:

[1] Para salir de Postgres usamos el comando `\q`

[2] Para dejar de estar *logeado* como el usuario `postgres` debes escribir `exit` en la terminal.

```
sudo -u postgres createuser --interactive
```

Donde se abre un menú para crear un usuario. Te recomiendo que si tu usuario del servidor se llama `juan`, crees un usuario `juan` en la base de datos. Ahora si escribes `psql`, no dejará que te *logees*, porque "Falta una base de datos llamada `juan`". Esto pasa porque al correr el comando `psql`, Postgres intenta iniciar sesión con un usuario que tenga el mismo nombre que el usuario del servidor, a una base de datos con ese nombre.

Así, para crear una nueva base de datos, puedes volver a hacer *login* con el usuario postgres, y entrar con el comando `psql`. Una vez dentro puedes escribir:

```
CREATE DATABASE juan;
```

Este comando creará una base de datos llamada `juan`. Si ahora vuelves a intentar ejecutar el comando `psql` con el usuario `juan`, podrás ingresar sin problemas.