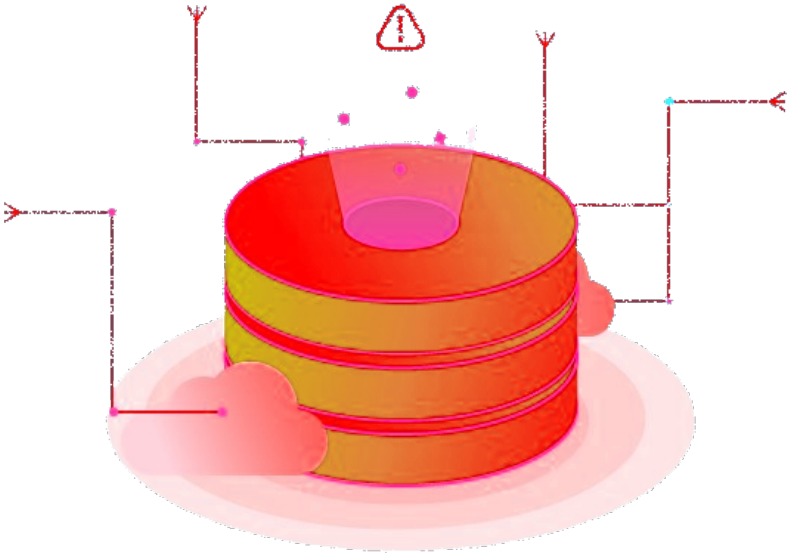
Universidad Nacional Autónoma De México
Facultad De Ciencias, 2024-I
Fundamentos De Bases De Datos



Práctica 01:
Instalación de Docker y del SMBD, PostgreSQL

Profesor:
Gerardo Avilés Rosas

Ayudantes de teoría:
Gerardo Uriel Soto Miranda
Valeria Fernanda Manjarrez Angeles

Ayudantes de laboratorio:
Ricardo Badillo Macías
Jerónimo Almeida Rodríguez

Introducción.

Docker

Docker es un sistema operativo para contenedores, el cual empaqueta todo el código y las dependencias de una aplicación en un formato estándar que permite su ejecución rapida y fiable.
De manera similar a las máquinas virtuales *Docker* virtualiza el hardware del servidor, los contenedores virtualizan el sistema operativo de un servidor.

Un **contenedor** proporciona virtualización ligera a nivel de sistema operativo mediante la abstracción del "espacio del usuario". Los **contenedores** comparten el núcleo del sistema host con otros contenedores. Un contenedor, que se ejecuta en el sistema operativo host, es una unidad de software estándar que empaqueta código y todas sus dependencias, para que las aplicaciones se pueden ejecutar de forma rápida y fiable de un entorno a otro. Los contenedores no son persistentes y se activan desde imágenes.

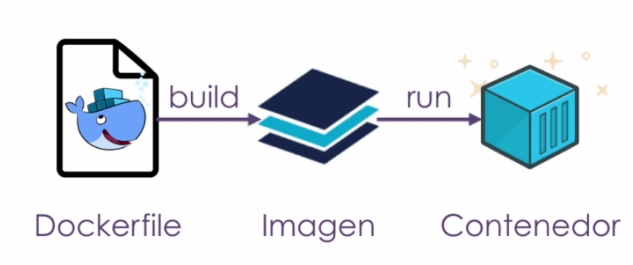


Figura 1: *Proceso General Docker.*

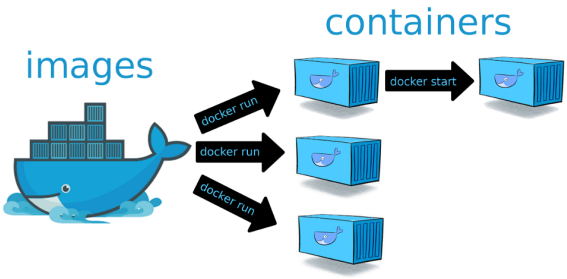


Figura 2: *Proceso ejecución Docker.*

Conceptos básicos de Docker

Los conceptos centrales de *Docker* son las imágenes y los contenedores. Una **imagen de Docker** contiene todo lo necesario para ejecutar software: el código, tiempo de ejecución, controladores, herramientas, scripts, bibliotecas, implementaciones, etc.

Un **contenedor de Docker** es una instancia en ejecución de una imagen de Docker, el cual se ejecuta en el núcleo del sistema operativo host.

Cada **contenedor de Docker** tiene su propio sistema de archivos, su propia pila de red, su propia dirección IP, su espacio de proceso propio y limitaciones de recursos definidas para la CPU y la memoria. **El contenedor de Docker** se inicia de forma instantánea, sin tener que arrancar un sistema operativo. Docker se basa en el **aislamiento**, en otras palabras se basa en separar los recursos de un sistema operativo host. Mientras que la **virtualización** suministra un sistema operativo invitado sobre el sistema operativo host.

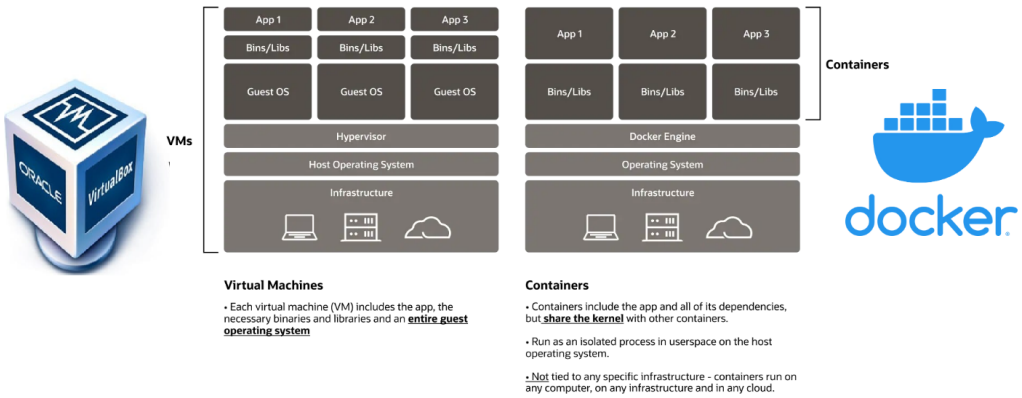


Figura 3: *Maquinas Virtuales vs Contenedores.*

PostgreSQL:

PostgreSQL es un Sistema Manejador de Bases de Datos (*SMBD*) *open source*, que incorpora el modelo relacional a sus bases de datos y soporta el estándar *SQL* como lenguaje de consulta. Es posible ejecutar *PostgreSQL* en la mayoría de los sistemas operativos modernos, como *Windows*, *Mac* y diversas distribuciones de *Linux*.

PostgreSQL puede ser utilizado para desarrollar las siguientes aplicaciones:

- **Online Transactional Processing OLTP**: Aplicaciones con un gran uso de operaciones `SELECT`, `INSERT`, `UPDATE` y `DELETE`, gran procesamiento de operaciones y mantenimiento de integridad de datos en ambientes multiacceso.
- **Online analytical processing OLAP:** Aplicaciones con una escasa cantidad de peticiones, consultas complejas que involucran agregación de datos, *data mining* y análisis de datos.

Arquitectura.

PostgreSQL utiliza un modelo cliente/servidor, donde el cliente y los programas de servidor pueden estar en diferentes *hosts* y comunicarse entre ellos a través del protocolo *TCP/IP* o vía *Linux sockets*. Es posible controlar múltiples conexiones de un cliente. *PostgreSQL* consta de los siguiente procesos:

- **Proceso del cliente o programa (frontend):** Realizan acciones a la base de datos. *PostgreSQL* provee algunas herramientas *frontend* tales como `psql`, `createdb`, `dropdb` y `createuser`.
- **Procesos del servidor (backend):** Administran los archivos de la base de datos, aceptan conexiones de clientes y realizan acciones solicitadas por el cliente.
  El nombre del proceso principal del servidor es *postgress/postmaster*. El proceso principal de *PostgreSQL* genera un nuevo proceso por cada conexión, así, cliente y servidor se pueden comunicar sin la intervención del proceso principal, cada proceso tiene un tiempo establecido de vida para aceptar y terminar la conexión.

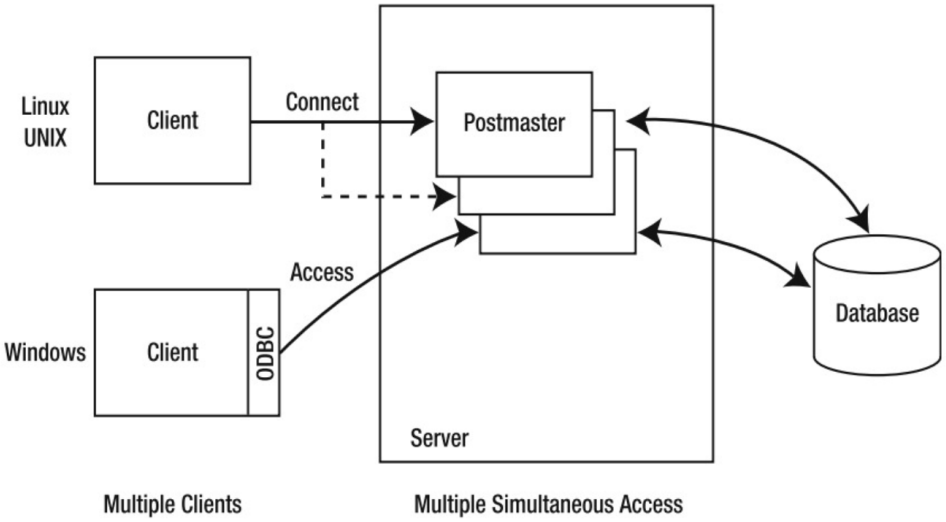


Figura 4: *Diagrama.*

En la Figura 2, se observa múltiples clientes conectándose a un servidor a través de la red, el proceso *postmaster* genera una conexión por cada cliente, sin importar que tipo de sistema operativo esté alojado el cliente.

Acceso de datos.

Con *PostgreSQL* es posible acceder a los datos de la siguiente manera:

- Utilizando la línea de comandos para ejecutar sentencias de *SQL*.
- Habilitando *SQL* directamente en la aplicación.
- A través de llamadas de funciones (*API's*).
- Accediendo a los datos de manera indirecta haciendo uso de *drivers* y bibliotecas como *ODBC* o *JDBC*.

Instalación Docker.

A continuación se enlistan diversos tipos de instalación en los sistemas operativos más comunes.

Linux, (Debian y Ubuntu).

i. Para instalar el repositorio oficial, primero actualizaremos la lista de paquetes existentes:

```
sudo apt update
```

ii. Instalaremos algunos paquetes de requisitos previos que permitan a apt a usar paquetes a través de HTTPS:

```
sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

iii. Añadimos la clave de GPC para el repositorio de Docker.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

iv. Agregamos el repositorio a las fuentes APT

```
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
```

v. Actualizamos el paquete de base de datos con los paquetes de Docker

```
sudo apt update
```

vi. Instalamos Docker

```
sudo apt install docker-ce
```

vii. Comprobamos el servicio:

```
sudo systemctl status docker
```

viii. Probamos si podemos trabajar con imágenes de Docker:

```
sudo docker run hello-world
```

Figura 5: *Linux.*

Windows y Mac

Para Windows y Mac instalaremos Docker Desktop.

Windows

i. Descargamos Docker de la siguiente dirección: `https://docs.docker.com/desktop/install/windows-install/`
ii. Una vez descargado el .exe, lo ejecutamos como administrador el cual nos abrira el siguiente instalador, donde seleccionaremos OK, para que comience la instalación.

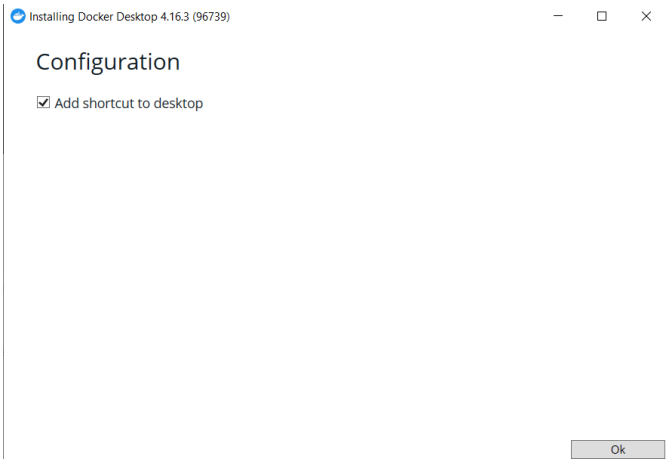


Figura 6: *Ejecutable*.

iii. Una vez que termine el proceso, nos pedira que reiniciemos el equipo

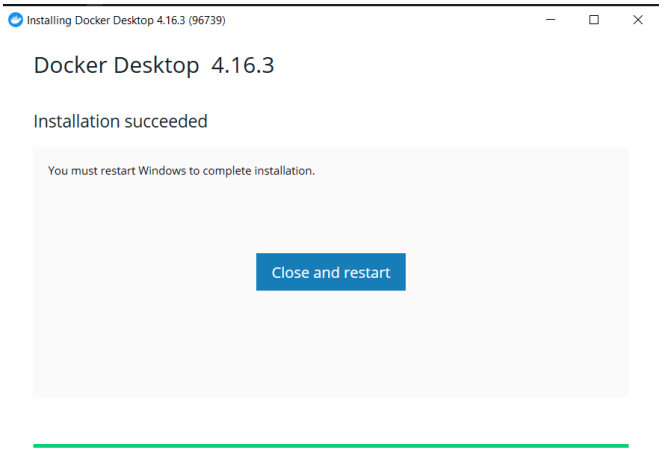


Figura 7: *Reiniciar*.

iv. Al reiniciar, si dejamos marcada la opción del acceso directo en el escritorio lo podremos ver. Si no aparecera si lo buscamos.

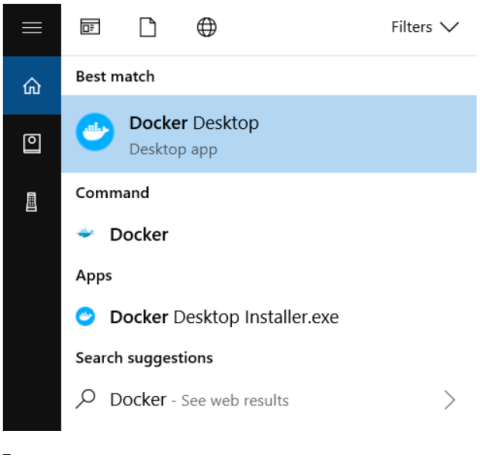


Figura 8: *Docker Instalado.*

v. Al ejecutar Docker, puede que salga el siguiente error:

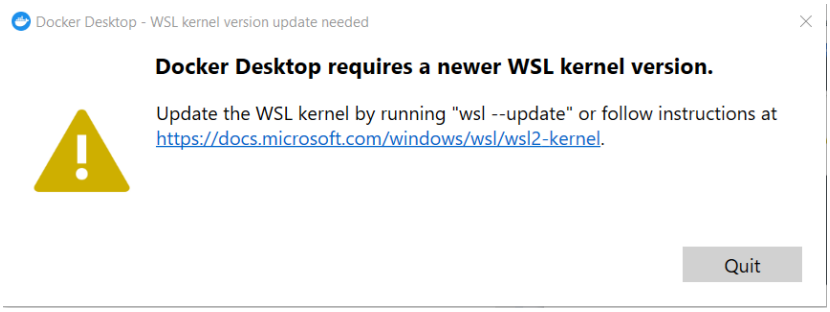


Figura 9: *Error.*

vi. Asi que en el cmd de Windows ejecutamos el siguiente comando, para que se actualice la dependencia necesaria.

```
wsl --update
```

vii. Si todo salio bien, al iniciar Docker Desktop debe salir lo siguiente

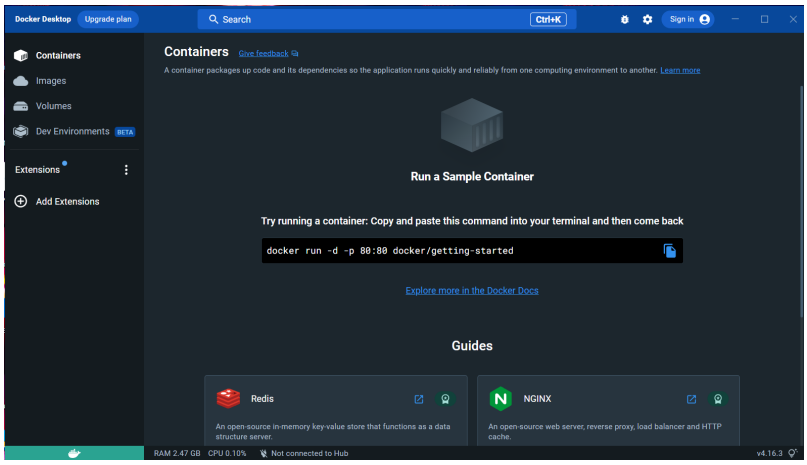


Figura 10: *Completado.*

Mac

i. Descargamos Docker de la siguiente dirección: `https://docs.docker.com/desktop/install/mac-install/`
ii. Una vez descargado el .dmg le damos doble click, para abrir el instalador.
iii. Al finalizar la instalación aparecera un icono llamado Docker.app el cual moveremos a la carpeta Aplicaciones.

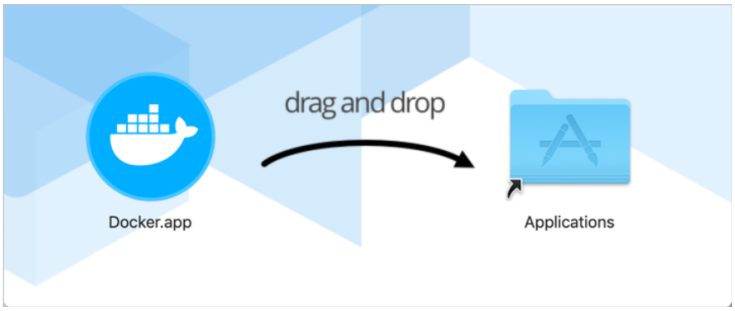


Figura 11: *App.*

iv. Para iniciar Docker, simplemente le damos doble click al Docker.app

Tanto para Mac, como para Windows para comprobar que todo este funcionando correctamente, en la terminal ejecutamos el siguiente comando

```
docker run hello-world
```

```
C:¥Users¥Shiri>docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:aa0cc8055b82dc2509bed2e19b275c8f463506616377219d9642221ab53cf9fe
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

Figura 12: *Correctamente Instalado.*

Figura 13: *Windows.*

Figura 14: *Mac.*

SMBD PostgreSQL

Nosotros al estar utilizando contenedores podemos utilizar el SMBD PostgreSQL sin tener que instalarlo. Entonces en una terminal hacemos lo siguiente:

i. Necesitamos descargar la imagen de postgres, el cual Docker utilizara al momento de crear el contenedor.

```
docker pull postgres
```

ii. Una vez descargado la imagen de postgres, utilizaremos el siguiente comando para crear el contenedor.

```
docker run -d --name postgres -e POSTGRES_PASSWORD=mysecretpassword -p 5432:5432 postgres
```

Donde:

- **docker run**: Es el comando que nos permite crear un contenedor a partir de una imagen Docker.
- **-d**: Nos permite ejecutar el contenedor en segundo plano.
- **–name**: Nos permite asignarle un nombre a nuestro contenedor. Si no se asigna Docker asignara uno automaticamente.
- **-e**: Es para pasarle al contenedor variables de entorno. En este caso le pasaremos la variable de entorno **POSTGRES_PASSWORD**, el cual tiene la contraseña que queremos en postgres
- **-p**: Nos permite mapear los puertos entre nuestra maquina local y el contenedor. En este caso Postgres escucha en el puerto 5432, asi que mapeamos el puerto 5432 de nuestra máquina local con el puerto 5432 del contenedor.
- **postgres**: Es el nombre de la imagen. Como no se especifica la version, esta utilizara la ultima version disponible.

iii. Una vez creado el contenedor, automaticamente se ejecutara, pero si necesitas iniciarlizarlo una vez ya creado y que no este activo tenemos que averiguar el **CONTAINER ID** de PostgreSQL, con el siguiente comando:

```
docker ps -a
```

iv. Supongamos que por el anterior comando nos da que el **CONTAINER ID** es *a1e321c26d33*, entonces para iniciar el conetenedor utilizamos :

```
docker start a1e321c26d33
```

v. Y para detener el contenedor que estamos utilizando, utilizamos el comando siguiente.

```
docker stop a1e321c26d33
```

vi. Si queremos conectarnos a PostgreSQL una vez creado el contenedor podemos utilizar psql(La consola de comandos de PostgreSQL) con:

```
docker run -it --link postgres:postgres postgres psql -h postgres -U postgres
```

Figura 15: *PostgreSQL*.

Database Tools.

Son *software* especializados en facilitar la interacciones entre los motores de bases de datos y los usuarios. Para no profundizar en la interfaz de línea de comando para la manipulación de sus bases de datos, puede elegir entre una gran variedad de herramientas de administración.
Las herramientas de administración de bases de datos proporcionan una interfaz *GUI/web* para automatizar las tareas de la base de datos, como buscar tablas, buscar y reemplazar, y cualquier otra tarea que desee ejecutar. Hay cientos para elegir, cada uno con sus propios pros y contras.

Instalación Database Tools, (DBeaver, pgAdmin, entre otros).

Algunas de las opciones recomendadas para la manipulación del *SMBD* para este curso son:

i. **pgAdmin4:** Es una herramienta de código abierto para la administración de bases de datos *PostgreSQL*.
Enlace de descarga: `https://www.pgadmin.org/download/`

ii. **DBeaver:** Es un cliente *SQL* y una herramienta de administración de bases de datos, utiliza la *API JDBC* para interactuar con ellas a través de un controlador *JDBC*. Para otras, como *NoSQL*, utiliza controladores propietarios.
Proporciona un editor que admite la creación de código y el resaltado de la sintaxis. Además de una arquitectura que permite modificar gran parte del comportamiento de la aplicación para proporcionar funciones específicas o que son independientes de la base de datos.
Esta es una aplicación de escritorio desarrollada en Java y basada en la plataforma Eclipse. DBeaver es gratuito como software de código abierto que se distribuye bajo la licencia de Apache.
Enlace de descarga: `https://dbeaver.io/download/`

Figura 16: *Dbeaver.*

Actividades.

En la sesión de laboratorio realizaremos la instalación de *Docker* y del SMBD *PostgreSQL* y explicamos de manera poco detallada los componentes en pantalla de *Dbeaver*.

i. Se debe de responder las siguientes preguntas en **equipo** y se entrega en un documento *PDF* con el nombre **Preguntas** seguido del **nombre** que le hayan dado a su equipo:

Número.	Pregunta.
1.	¿Qué otros *SMBD* existen actualmente en el mercado?
2.	¿Cuáles son las principales diferencias con *PostgreSQL*?
3.	¿Por qué una empresa debería escoger una base de datos *open source*?
4.	¿Cuáles son las ventajas, para un *DBA* el trabajar con un *SMBD, open source*?
5.	¿Qué son las bases de datos *NoSQL*? Menciona 3 ventajas y desventajas contra las bases relacionales.

ii. **Cada integrante** deberá de realizar un reporte en donde indiquen todos los pasos que realizaron para la instalación de *Docker* y de *PostgreSQL*.Ademas deberan incluir como se conectaron con *DBeaver* que es la herramienta que utilizaremos durante el curso. Cada bitácora debe ser un documento *PDF* que inicie con la palabra **Bitácora** seguido del **apellido paterno** de la persona que elaboró su documento.
Los puntos que se requiere en su reporte de instalación son:

Punto.	Descripción.
1.	Sistema operativo y versión.
2.	Distribución (solamente en el caso de *Linux*).
3.	Versión de la instalación.
4.	Tiempo requerido.
5.	Explicación del paso a paso que realizaste con sus respectivas capturas de pantallas. (Adicionalmente agrega las evidencias de los pasos que consideres esenciales de la instalación).
6.	Comentarios y los problemas a los que te enfrentaste en la instalación.

Figura 17: *Actividades.*

Entregables.

Deberán subir un archivo con formato *zip* a *Google Classroom*, de acuerdo a lo indicado en los lineamientos de entrega. Debe de estar organizado de la siguiente manera, (suponiendo que el nombre del equipo que está entregando es *Dream Team* y los integrantes del equipo son los profesores del curso).

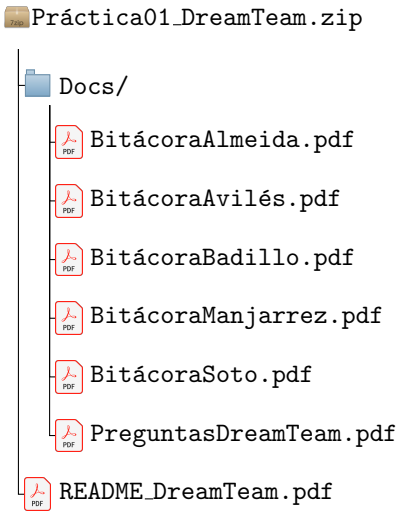


Figura 18: *Entregables.*

Nota.

Para cualquier duda o comentario que pudiera surgirles al hacer este trabajo, recuerden que cuentan con la asignación de este entregable en el grupo de *Classroom*, en donde seguramente encontrarás las respuestas que necesites.

Figura 19: *Nota.*