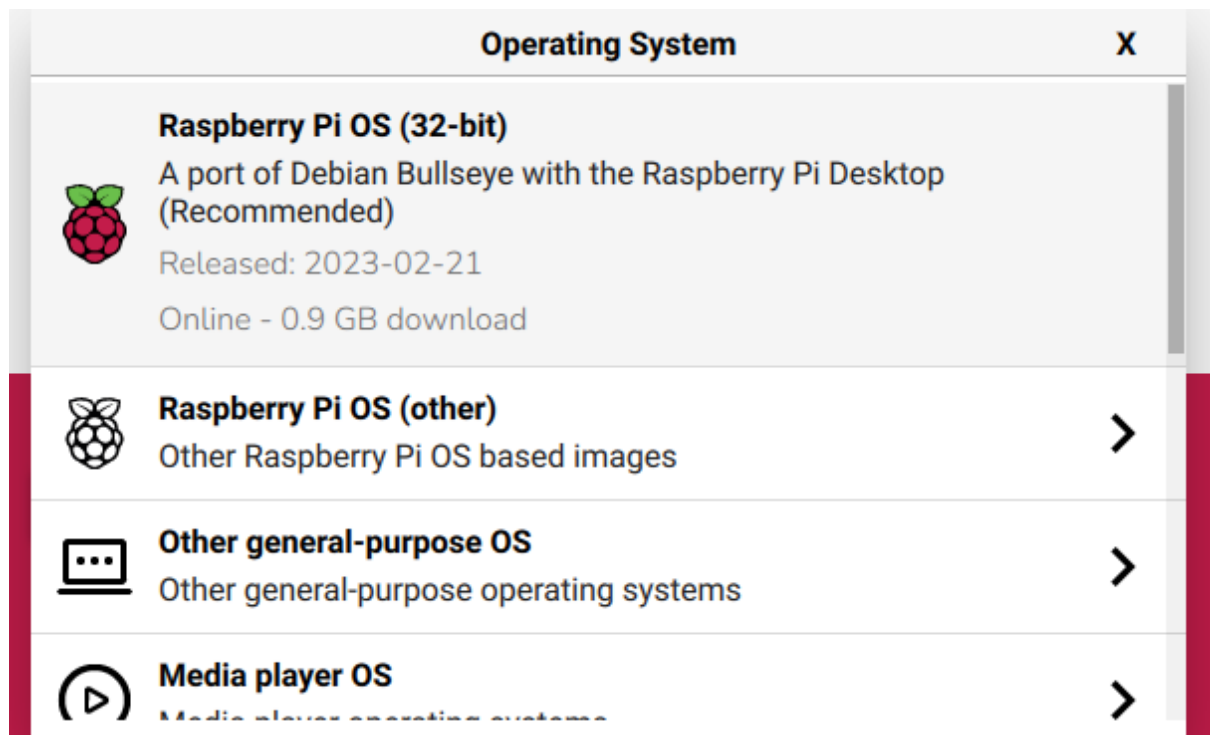


# Instalación de docker en Raspberry Pi

## Pasos previos

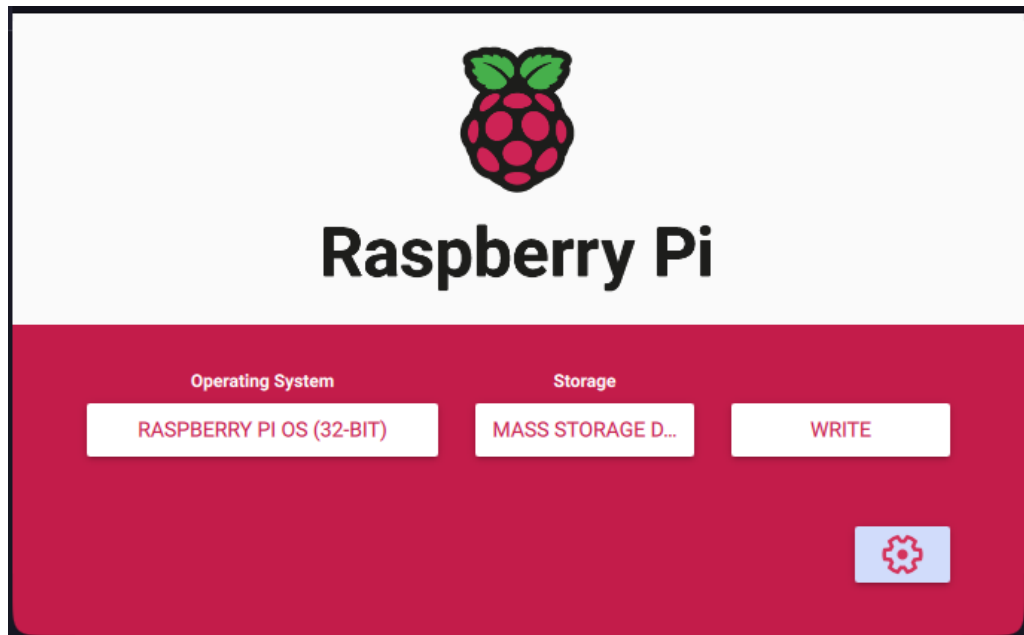
Instalar en tarjeta MicroSD (por lo menos 16GB) una imagen de Raspbian usando Raspberry Pi Imager <https://www.raspberrypi.com/software/>.

Choose OS:



Instala el sistema completo para Raspberry Pi completo con escritorio para usar con monitor, teclado y mouse. En caso de no usarlo, se puede descargar la versión Lite desde Raspberry Pi OS (other) que no incluye aplicaciones gráficas.

En choose storage se elige la tarjeta MicroSD. Antes de continuar con la instalación, ir a configuración en la parte inferior derecha



Allí habilitar:

- SSH con autenticación con contraseña
- Definir un usuario y contraseña
- Configure wireless LAN con SSID y contraseña. Para clase de laboratorio, usar Hands On IoT como SSID y handsoniot como contraseña.
- Opcionalmente configurar el locale para el país de origen.

Finalmente ir a write y esperar la descarga y grabado de imagen

## Acceso remoto por SSH

Una vez iniciada la Raspberry Pi por primera vez, se intentará conectar a la red elegida anteriormente. Si es posible, verificar desde la configuración del router la dirección IP asignada.

Al conectarse a la red WiFi, se puede acceder a al sistema de la Raspberry a través de red local con otro equipo utilizando Secure SHell (SSH). En sistemas Linux, Mac o afines, esta herramienta viene instalada por defecto y puede ejecutarse desde la terminal utilizando:

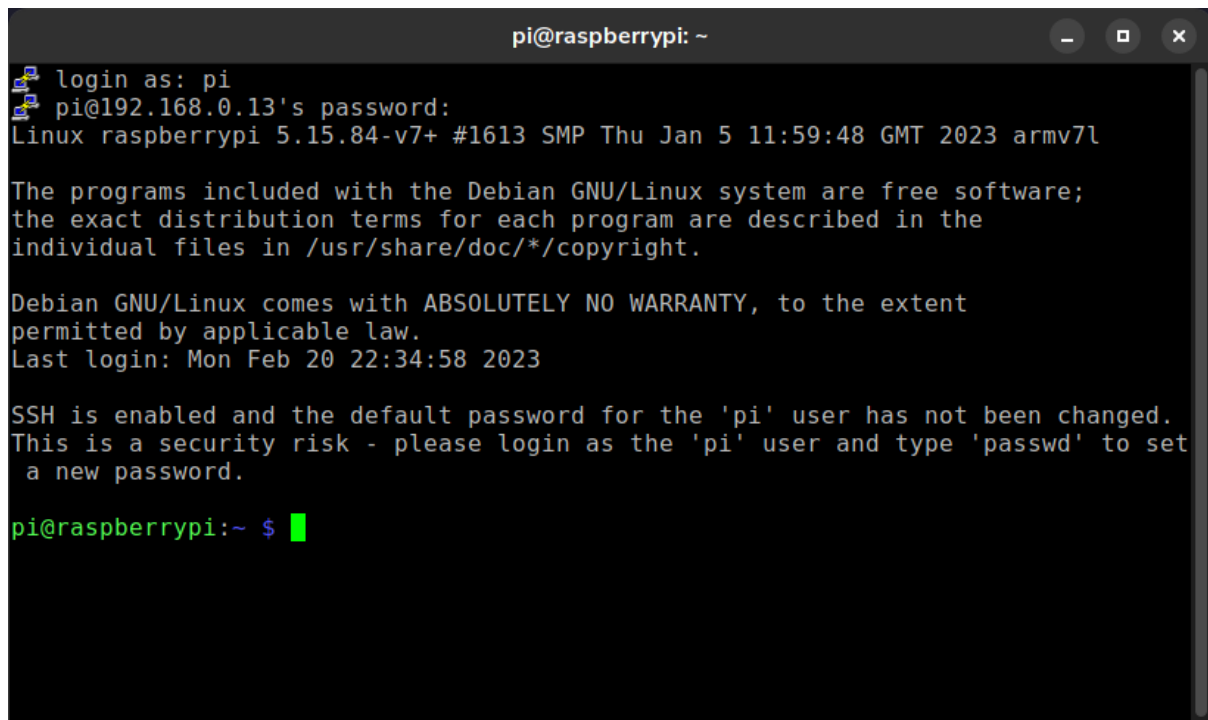
```
ssh <usuario>@<ip>
```

Para el caso que usuario haya sido pi e IP 192.168.0.10:

```
ssh pi@192.168.0.10
```

Luego de esto, nos pedirá si queremos agregarlo a dispositivos conocidos, a lo cual respondemos que sí y finalmente ingresar la contraseña para acceder.

En caso de no tener ssh por defecto (por ejemplo en Windows), se puede utilizar la herramienta [PuTTY](#). Para este caso, la conexión se realiza de forma similar. Primero nos pregunta si aceptamos la conexión, luego como quién queremos acceder (usuario) y luego contraseña.



```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.13's password:
Linux raspberrypi 5.15.84-v7+ #1613 SMP Thu Jan 5 11:59:48 GMT 2023 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 20 22:34:58 2023

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

En verde, nos muestra el nombre de usuario (pi) y el nombre del host (raspberrypi) seguido de ~ que representa que estamos en la carpeta personal del usuario actual (/home/pi) y el \$ nos representa como usuarios normales a diferencia de # que corresponde a superusuario.

Una vez que logramos conectarnos, accedemos a una línea de comandos de Linux. Algunos comandos útiles son:

- ls: lista los archivos en el directorio
- cd <directorio>: permite movernos a otro directorio relativo al actual. Nota: si usamos como directorio .. (dos puntos seguidos) podemos movernos al directorio anterior
- mkdir <directorio>: crea un directorio
- cp <origen> <destino>: copia un archivo. Debe agregarse la bandera -r para un directorio

- `mv <origen> <destino>`: mueve un archivo o carpeta
- `nano <archivo>`: editor de texto. Si el archivo no existe, lo crea
- `sudo <comando>`: nos permite ejecutar un comando con privilegios de superusuario. De esta manera, podemos modificar archivos del sistema (fuera de `/home/pi`).
- `sudo raspi-config`: herramienta que nos permite modificar configuraciones básicas y avanzadas de la Raspberry Pi y actualizar el sistema
- `sudo apt <install o search>`: gestor de paquetes del sistema. La opción `install` nos permite instalar paquetes nuevos y `search` nos permite buscar mediante palabras claves
- `<Ctrl-C>`: esta combinación de teclas permite interrumpir la ejecución de un comando.

## Instalación de docker

Para instalar docker, utilizamos el comando `apt` de la siguiente manera:

```
sudo apt install docker.io docker-compose
```

Con este comando, instalaremos tanto docker como una herramienta, `docker-compose`, que simplificará la gestión de contenedores. Finalmente, tenemos que generar permisos de usuario para ejecutar contenedores. Para esto ejecutamos:

```
sudo gpasswd -a <usuario> docker
```

Cerramos la sesión de `ssh` cerrando la ventana y volvemos a abrir. Finalmente, para probar que funciona correctamente, se ejecuta el siguiente comando:

```
docker run hello-world
```

Esto crea un contenedor de prueba y muestra un mensaje. Si todo funciona correctamente, deberíamos ver lo siguiente:

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
04341b189be6: Pull complete  
Digest: sha256:4e83453afed1b4fala3500525091dbfca6cele66903fd4c01ff015dbcb1ba33e  
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.  
   (arm32v7)  
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:
```

## Docker-compose

Es una herramienta que nos permite generar contenedores a partir de archivos de configuración en formato yaml. Este formato consiste en listar opciones en una jerarquía marcada por espacios. Un ejemplo de estos archivos sería:

```
version: "3"  
services:  
  nombre:  
    image: <nombre de imagen de docker>  
    ports:  
      - "<puerto local>:<puerto de contenedor>"  
    volumes:  
      - "./carpeta_local:/carpeta_en_contenedor"  
  otro:  
    image: <otra imagen>  
...
```

Estos archivos se tienen que llamar docker-compose.yaml y ser almacenados en carpetas distintas. Para simplificar el proceso, se proveerán los archivos necesarios para los servicios que se ejecutarán ya que el proceso de configuración excede el alcance del curso.

Para comenzar, desde ssh creamos una carpeta llamada docker, donde vamos a colocar las carpetas con los archivos de docker-compose. Para esto ejecutamos:

```
mkdir docker
cd docker
```

Ahora crearemos una carpeta iot donde quedará almacenado el archivo docker-compose.yaml correspondiente al servicio de mosquitto y NodeRED:

```
mkdir iot
cd iot
nano docker-compose.yaml
```

El contenido del archivo debe completarse con:

```
version: '3'

services:
  mosquitto:
    image: eclipse-mosquitto:2
    volumes:
      - "./mosquitto.conf:/mosquitto/config/mosquitto.conf"
      - "./mosquitto-data:/mosquitto/data"
      - "./mosquitto-log:/mosquitto/log"
    ports:
      - "1883:1883"
    restart: unless-stopped

  node-red:
    image: nodered/node-red:latest
    user: "1000"
    environment:
      - TZ=America/Buenos_Aires
    ports:
      - "1880:1880"
    volumes:
      - "./node-red-data:/data"
    restart: unless-stopped
```

Luego apretamos Ctrl-X e Y para cerrar y guardar los cambios. Nota: en PuTTY para pegar del portapapeles la combinación de teclas es Shift+Insert.

Antes de ejecutar el contenedor hay que crear la carpeta node-red-data y el archivo mosquitto.conf. El resto de las carpetas se generan de forma automática.

```
mkdir node-red-data
nano mosquitto.conf
```

Se sugiere utilizar la siguiente configuración de mosquitto que genera logs, persistencia entre sesiones y uso anónimo.

```
listener 1883
persistence true
persistence_location /mosquitto/data/
log_dest file /mosquitto/log/mosquitto.log
allow_anonymous true
```

Luego apretamos Ctrl-X e Y para cerrar y guardar los cambios.

Para ejecutar el contenedor utilizamos el comando `docker-compose up -d`.

La bandera `-d` indica que se ejecuta de fondo y no muestra logs sobre el funcionamiento actual. En caso de quitar la bandera, se puede ver en tiempo real qué está sucediendo con el servicio pero bloquea la línea de comandos. Para ver los logs mientras se está ejecutando el contenedor se puede usar el comando `docker-compose logs`. Para detener los contenedores se ejecuta el comando `docker-compose down`. Dada la forma en la que está configurado (`restart: unless-stopped`), los servicios continúan siendo ejecutados incluso al reiniciar la Raspberry Pi.

Una vez completado el comando se puede ir desde un navegador de un dispositivo conectado a la misma red que la Raspberry Pi a la dirección `http://<ip>:1880` para acceder a NodeRED. Para probar mosquitto, se puede utilizar MQTT-Explorer conectándose a la IP de la Raspberry Pi y puerto 1883.