**Parte 2: Ejemplos usando el comando Docker**

En esta segunda guía aprenderá a instalar docker y a ejecutar el comando docker para usar imágenes existentes, crear, detener, arrancar e inspeccionar contenedores.

**Instalar y Configurar:** Descargue el repositorio del taller desde Github. El repositorio tiene todos los documentos y comandos requeridos, para facilitar y agilizar el taller. Para descargarlo use el siguiente comando:

git clone https://github.com/ingsistemascloud/seiiis2019.git

Espere a que descargue y ubíquese en el directorio seiiis2019. Use el siguiente comando:

cd seiiis2019/

Encontrará un SCRIPT llamado install-docker.sh, que permitirá instalar docker y docker-compose, las dos herramientas a usar en el taller. Ejecute ese SCRIPT con el siguiente comando:

./install-docker.sh

Espere a que instale y ubíquese en el directorio ejemplosDockerCommand. Use el siguiente comando:

cd ejemplosDockerCommand/

En ese directorio encontrará seis (6) ejemplos. Por cada ejemplo hay un fichero de texto plano (TXT) que contiene los comandos. Puede copiarlos y pegarlos, de manera que no se presenten errores mientras prueba y aprende. Si usa putty, para copiar simplemente seleccione el texto y para pegar haga click derecho. Para ver el contenido de los ficheros pruebe a usar el comando more y el nombre del fichero, por ejemplo:

more ejemplo0.txt

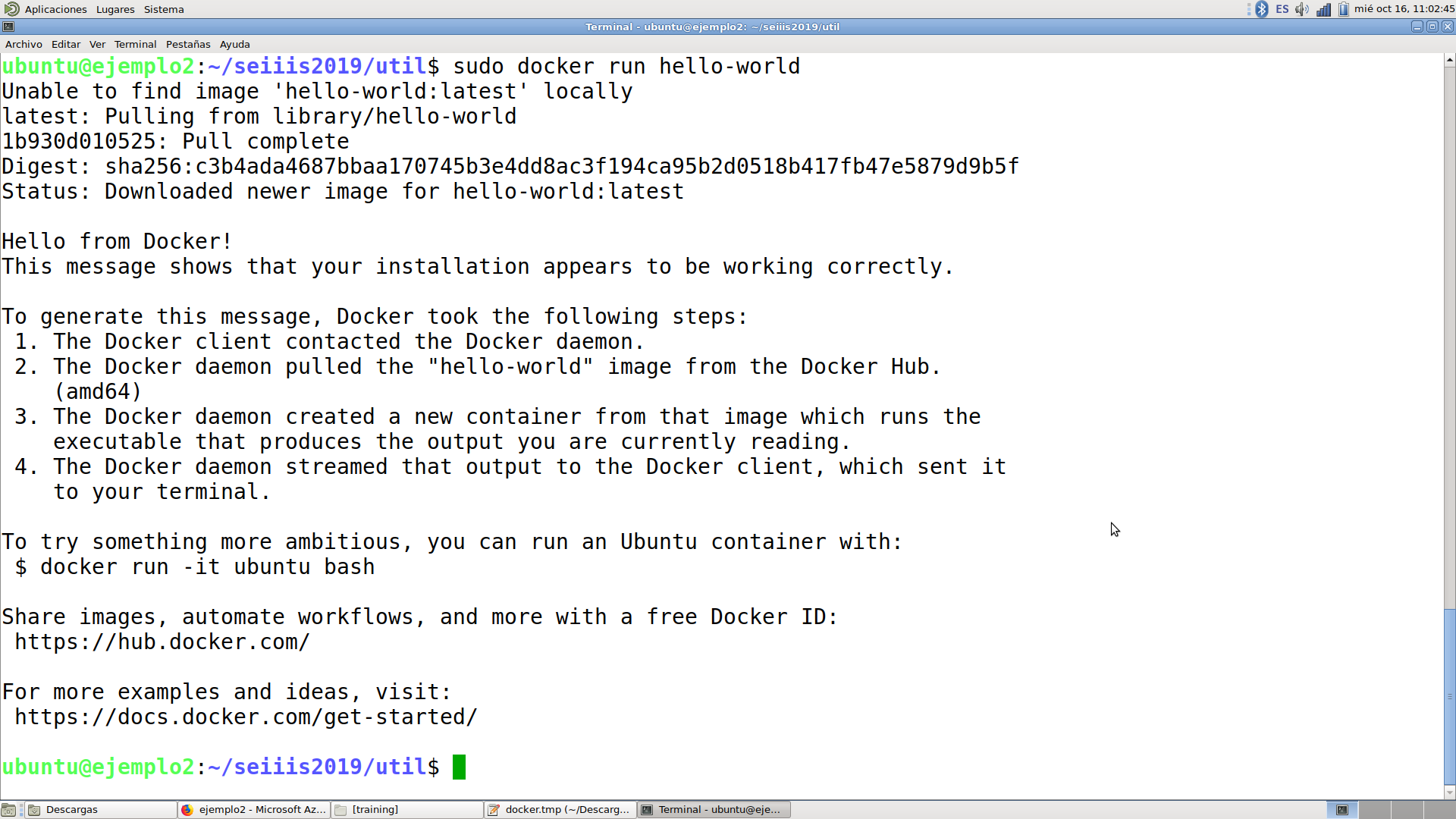
**0. Docker Hola Mundo**. Para crear un docker se usa el comando “sudo docker run IMAGEN”, donde IMAGEN corresponde al nombre de una imagen existente. Si la imagen no existe en el servidor, se descarga desde dockerhub. Para realizar la prueba típica de Hola Mundo, ejecute los siguientes comandos:

sudo docker images

Si todo funciona correctamente, no aparecerá ninguna imagen, pues acaba de instalar docker y es la primera vez que lo usa. Ahora ejecute el siguiente comando:

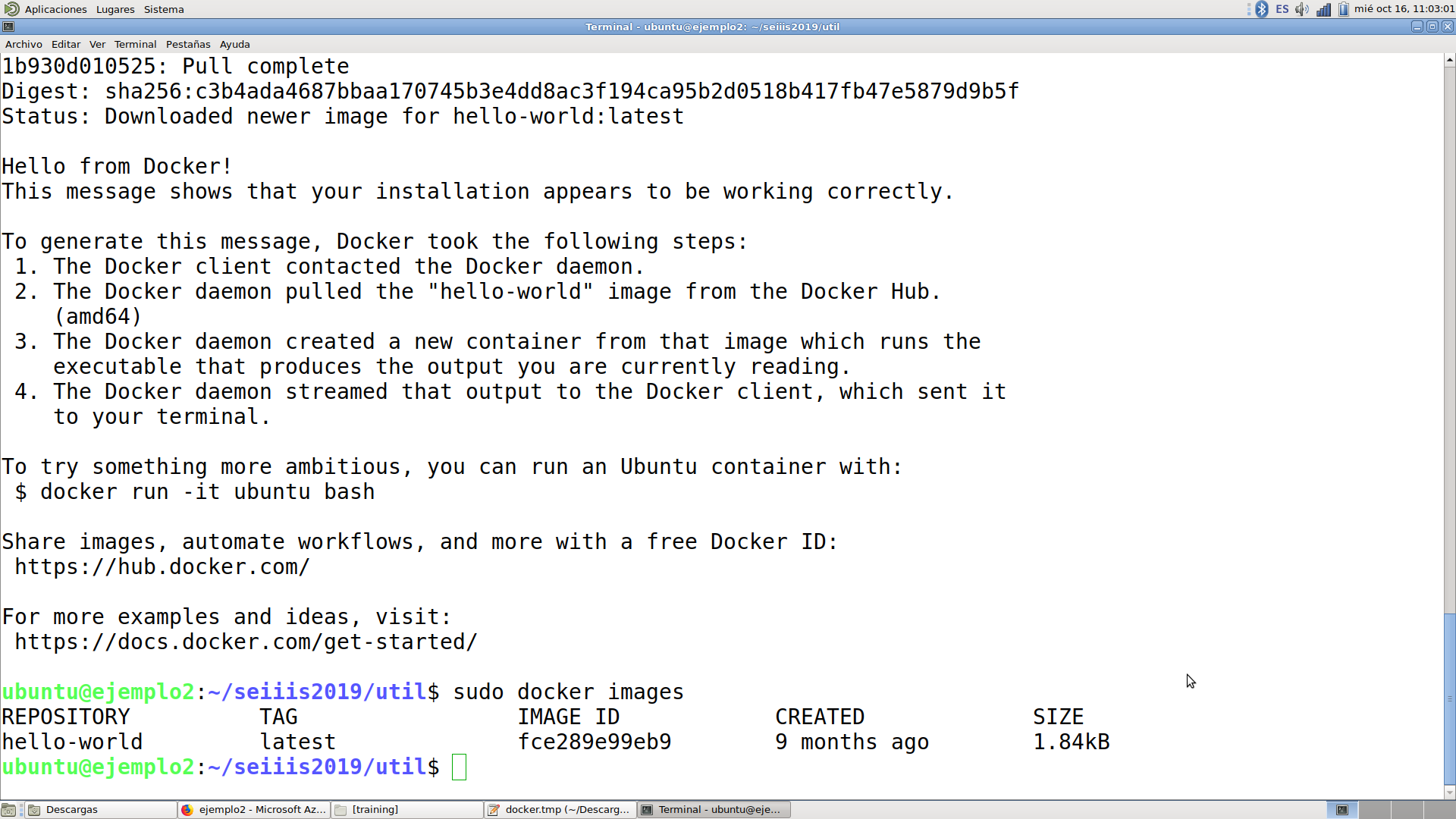
sudo docker run hello-world

Si todo funciona correctamente, aparecerá una pantalla como la siguiente:



Lo que sucedió es que se descargó la imagen del docker hello-world y se ejecutó en el servidor. Pruebe de nuevo el primer comando y ahora aparecerá la imagen que se descargó.

sudo docker images

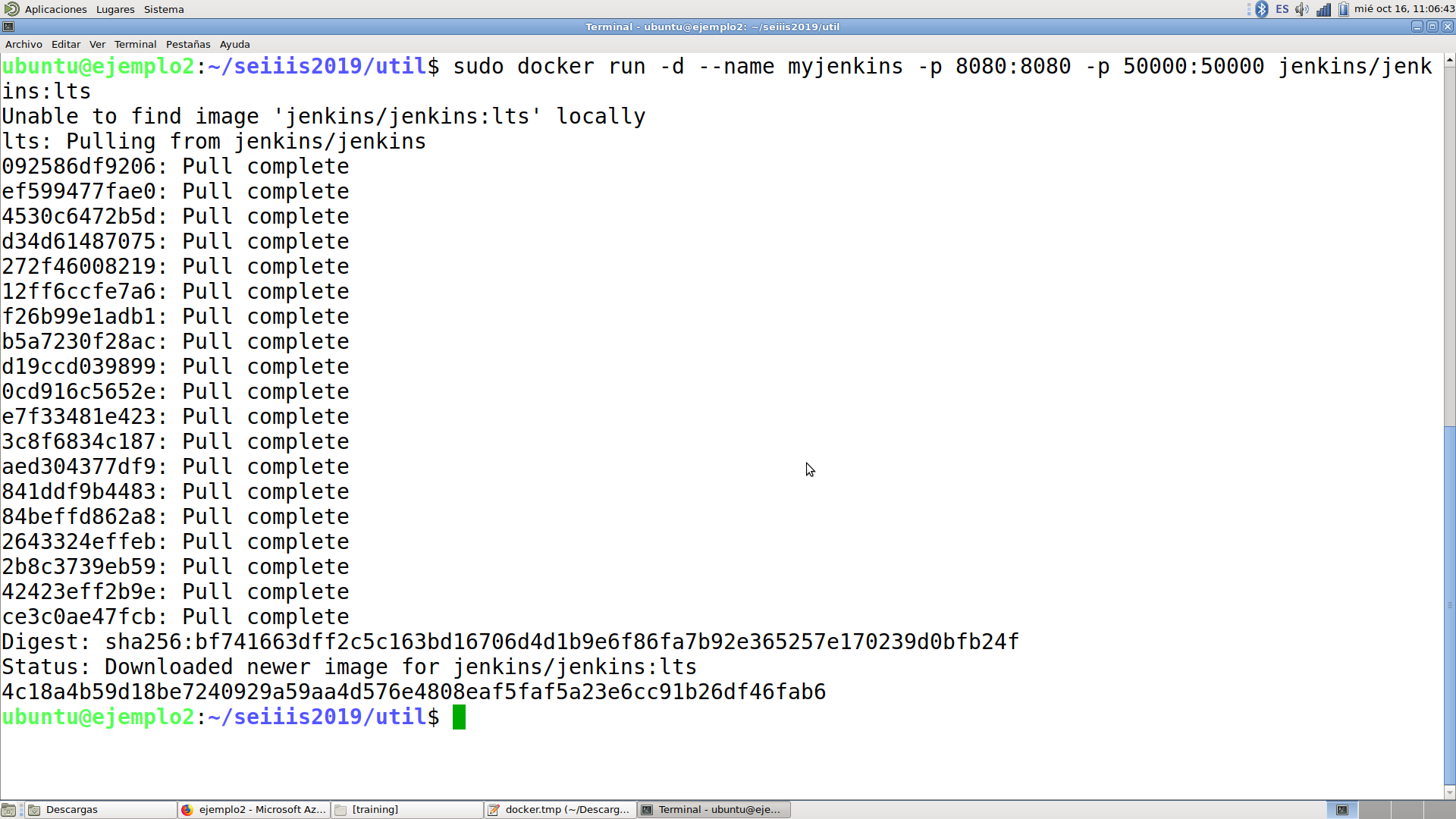


**1. Docker jenkins**. Ahora se va a crear un docker usando una imagen de la herramienta jenkins, usada para automatizar la integración contínua (devops). Use el siguiente comando (como el comando no cabe en una sóla línea, las lineas se separan usando el carácter backslash \):

sudo docker run -d --name myjenkins \

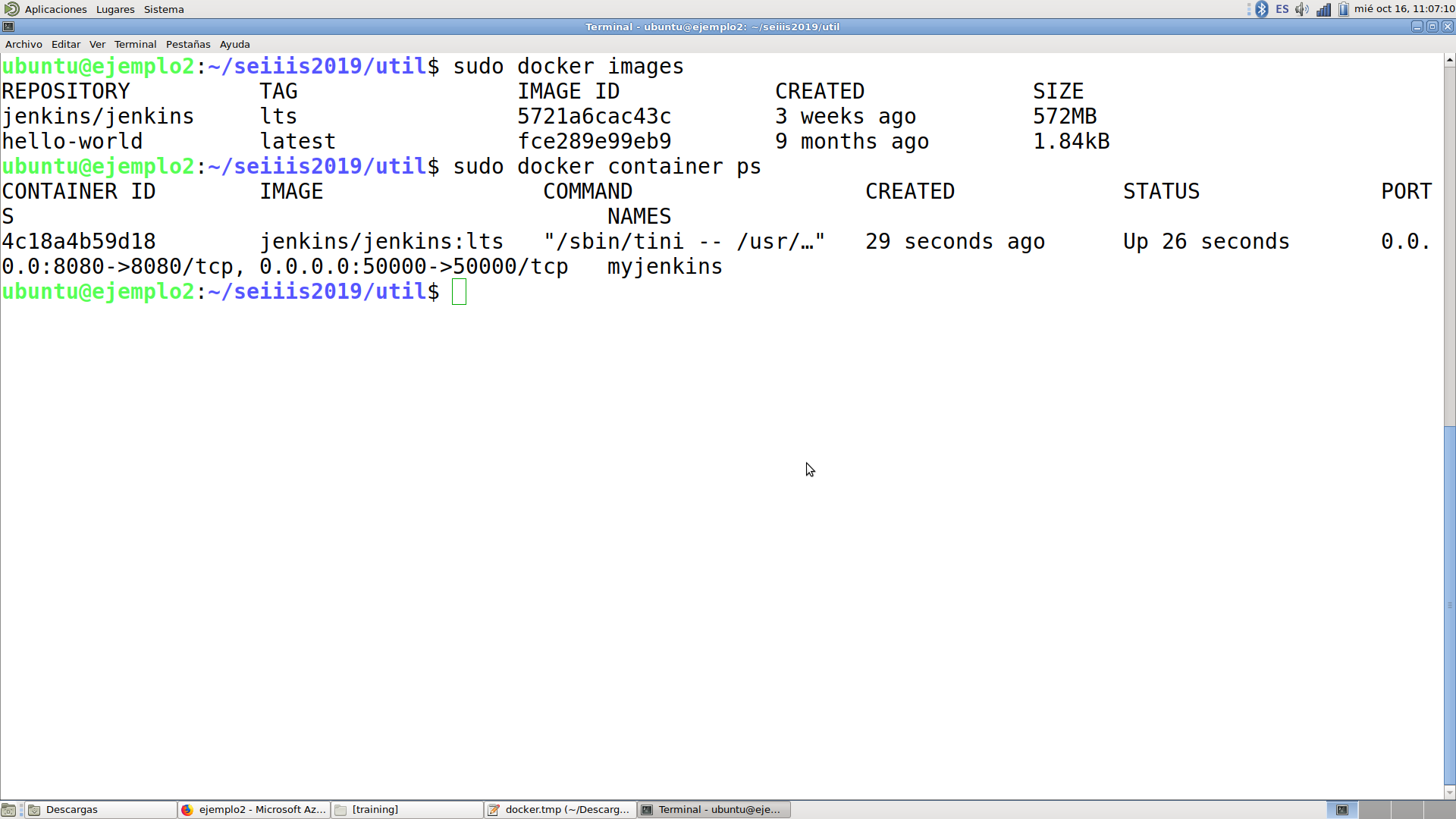
-p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

Espere a que finalice y aparecerá una imagen como la siguiente:

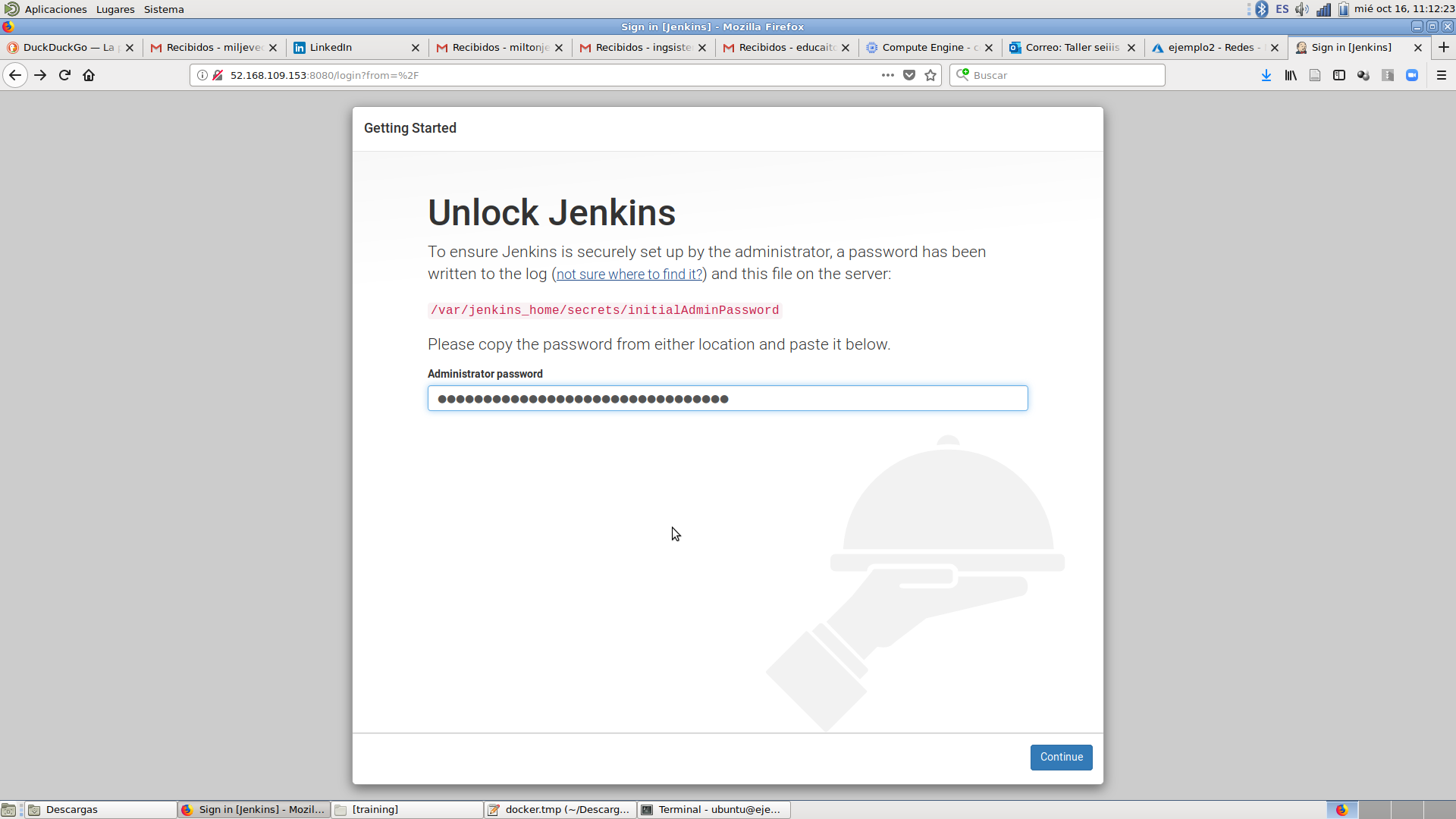


Ahora pruebe el comandos docker container ps, que lista los contenedores en ejecución. Aparecerá su docker jenkins ejecutándose y recibiendo peticiones HTTP en el puerto 8080. Observe que el docker se llama myjenkins, según el comando usado para crearlo.

sudo docker container ps



Abra el navegador e ingrese a la URL <http://DIRECCION_IP_PUBLICA:8080/> donde la DIRECCION\_IP\_PUBLICA corresponde a la dirección de su máquina virtual, la cual debe revisar en la consola de AWS. Aparecerá una ventana como la siguiente, pidiéndole una clave e informándole la ruta donde se encuentra esa clave.



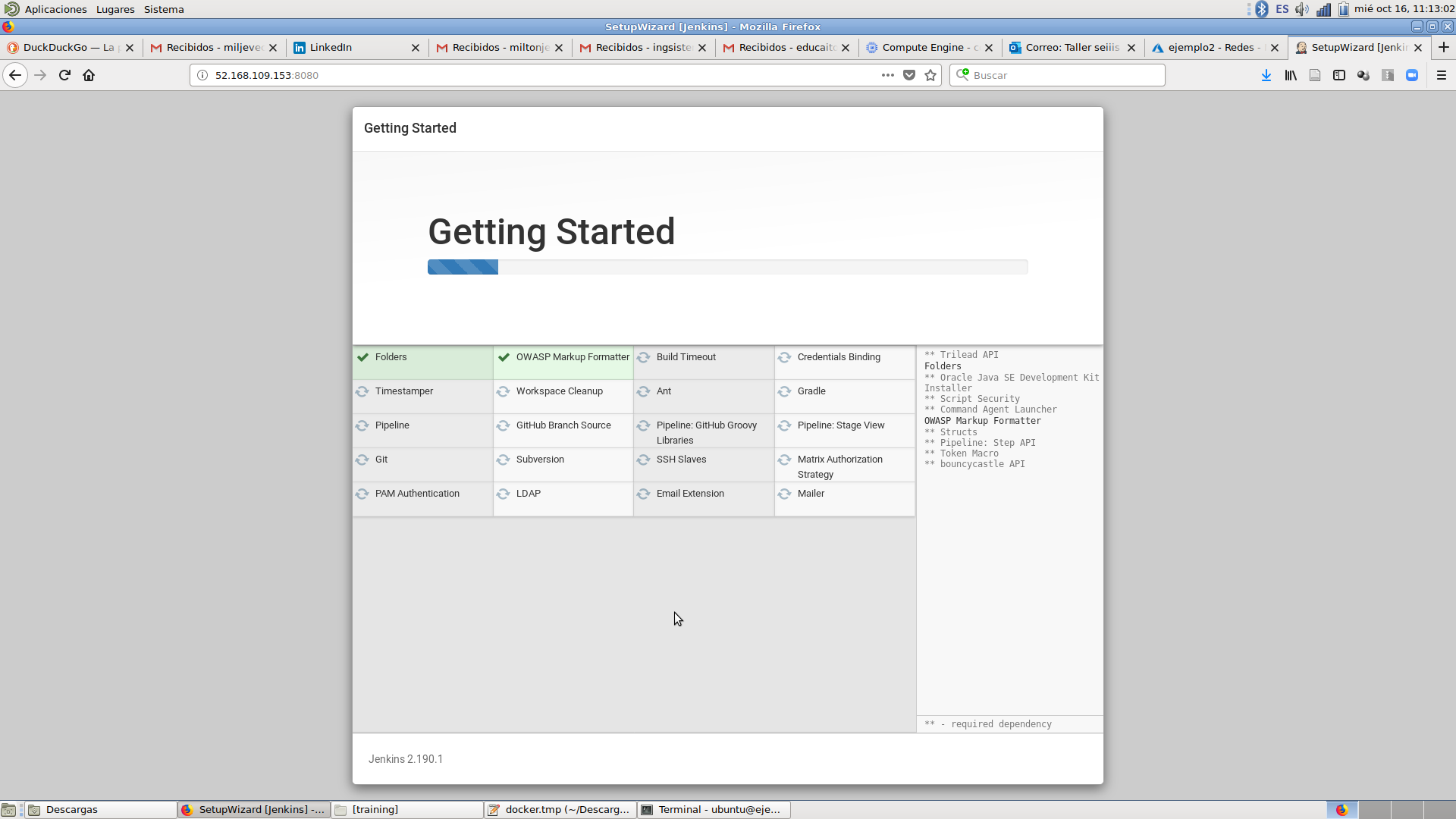
Ejecute el siguiente comando para acceder a la clave, copie la clave y péguela en la ventana del navegador.

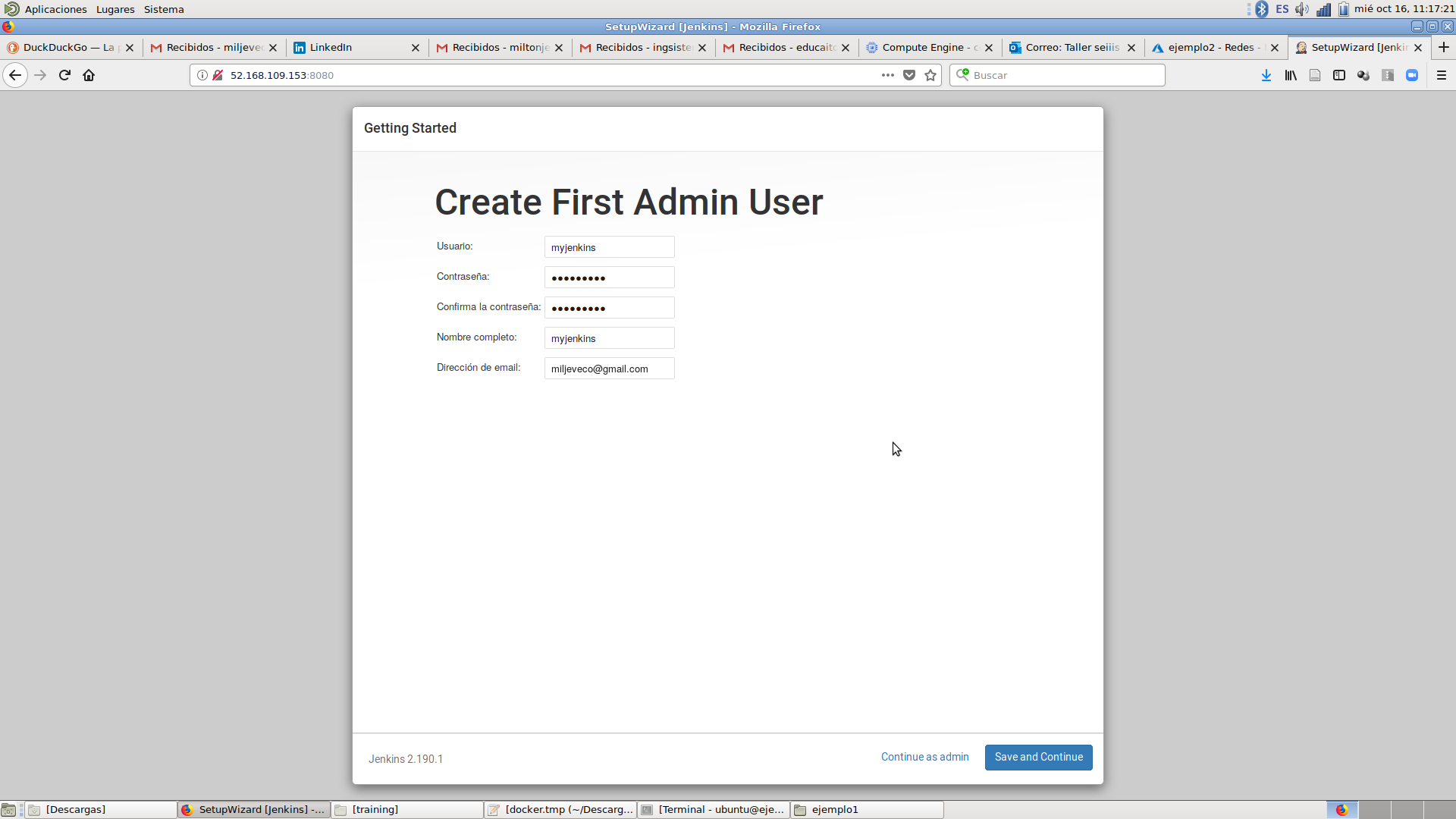
sudo docker exec -it myjenkins \

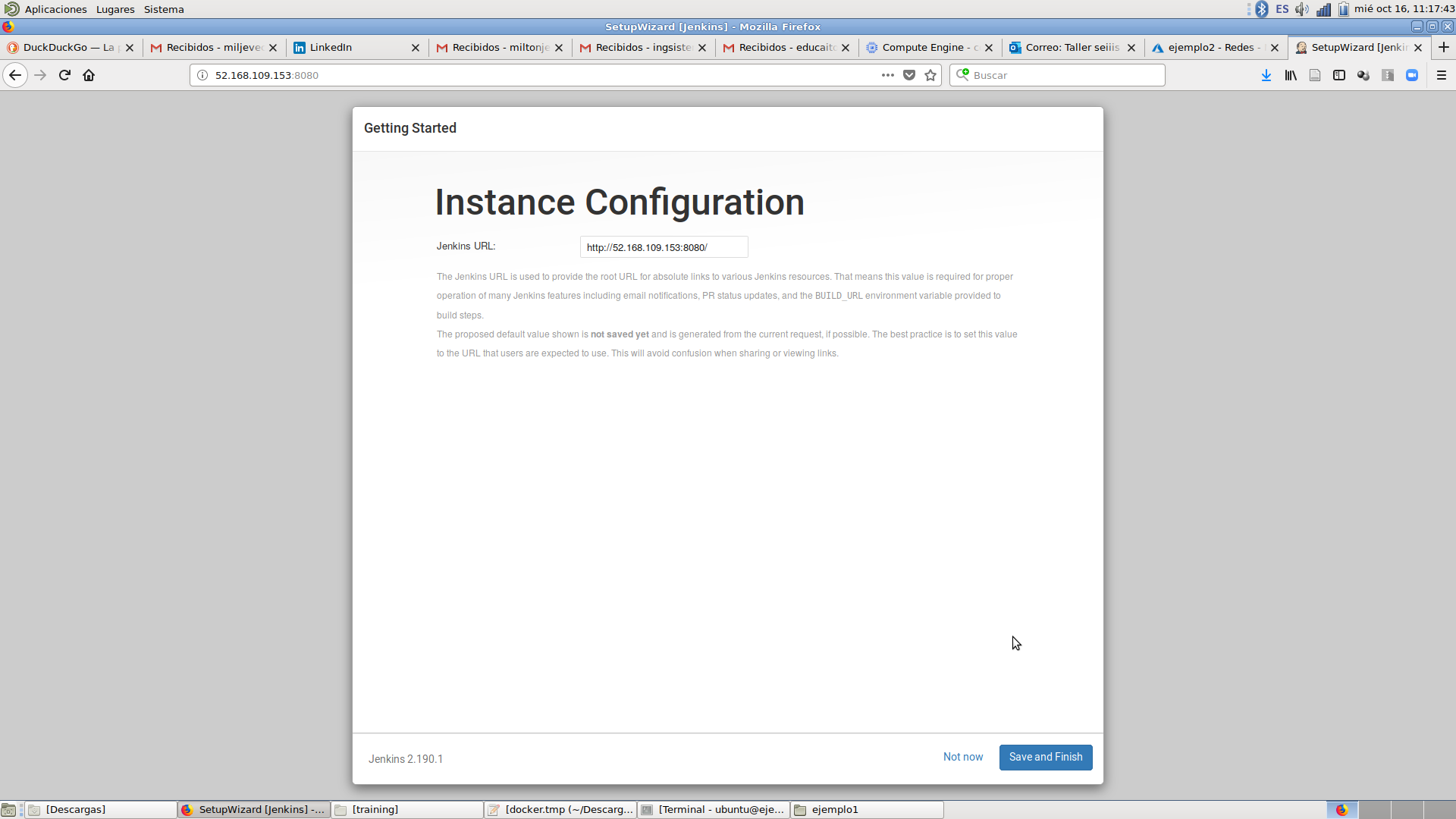
cat /var/jenkins\_home/secrets/initialAdminPassword

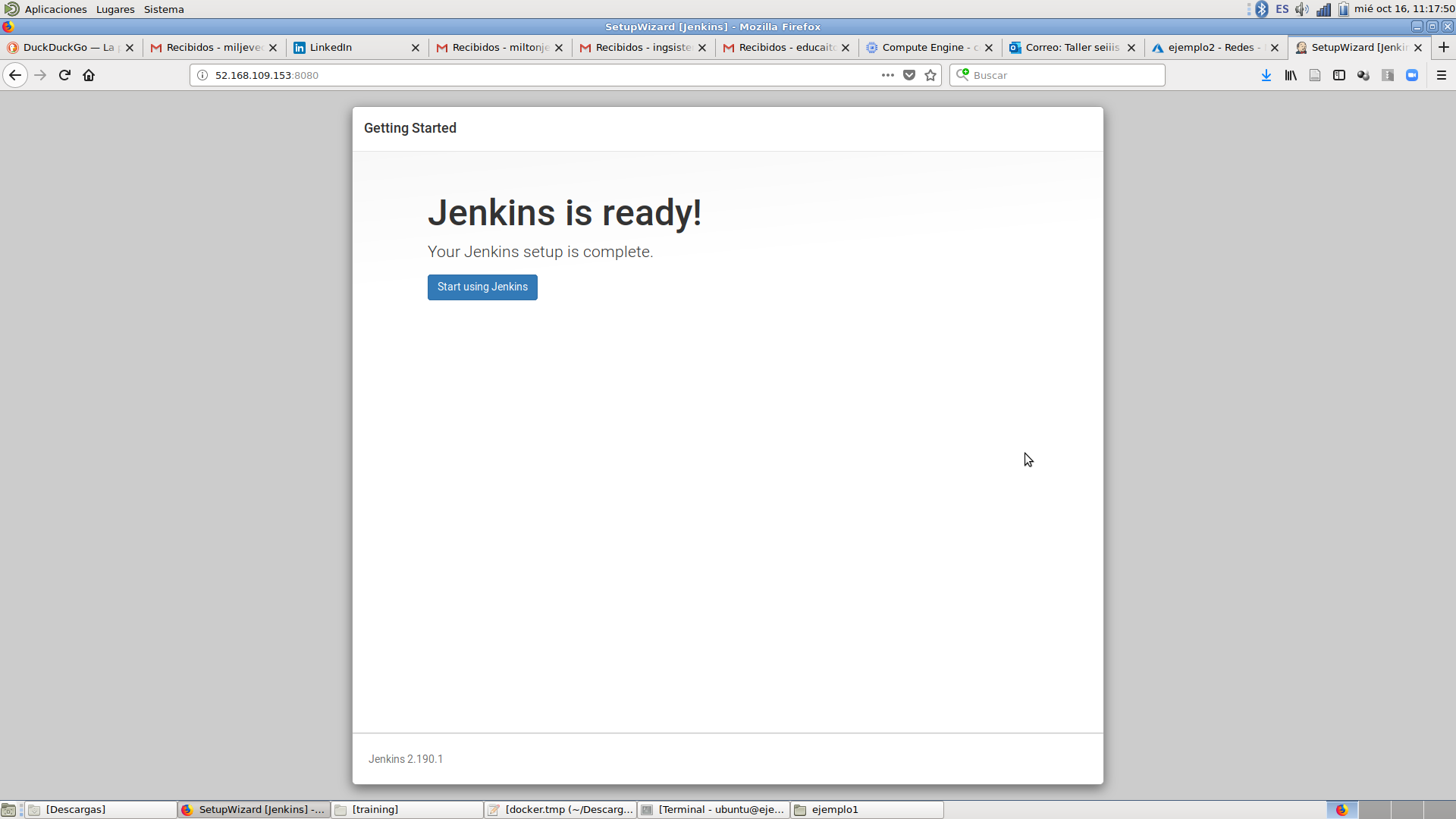
Felicitaciones. Acaba de desplegara una aplicación usando dockers, algo muy sencillo y ágil. Ahora siga el asistente de instalación web de la aplicación, paso a paso.

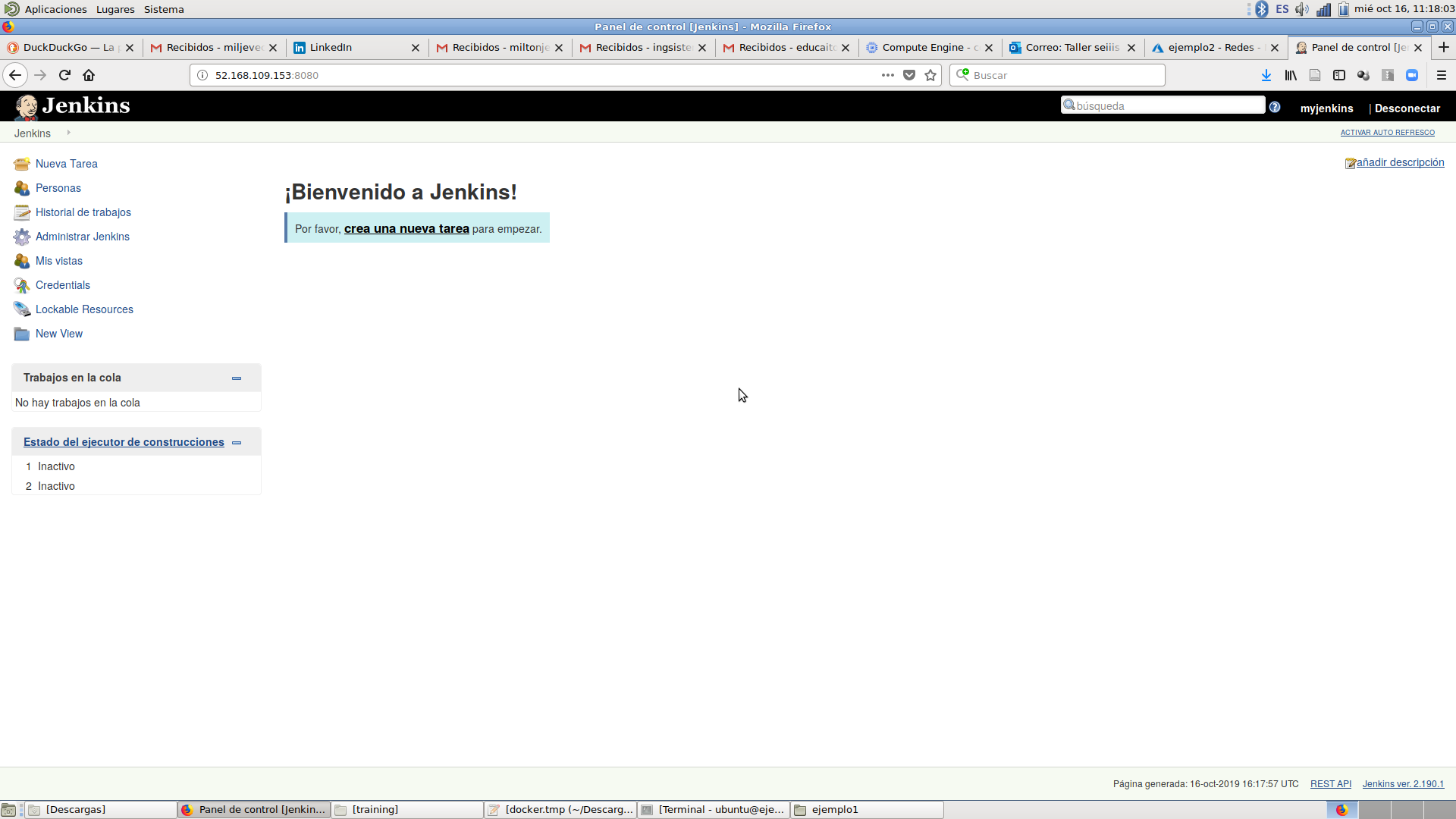








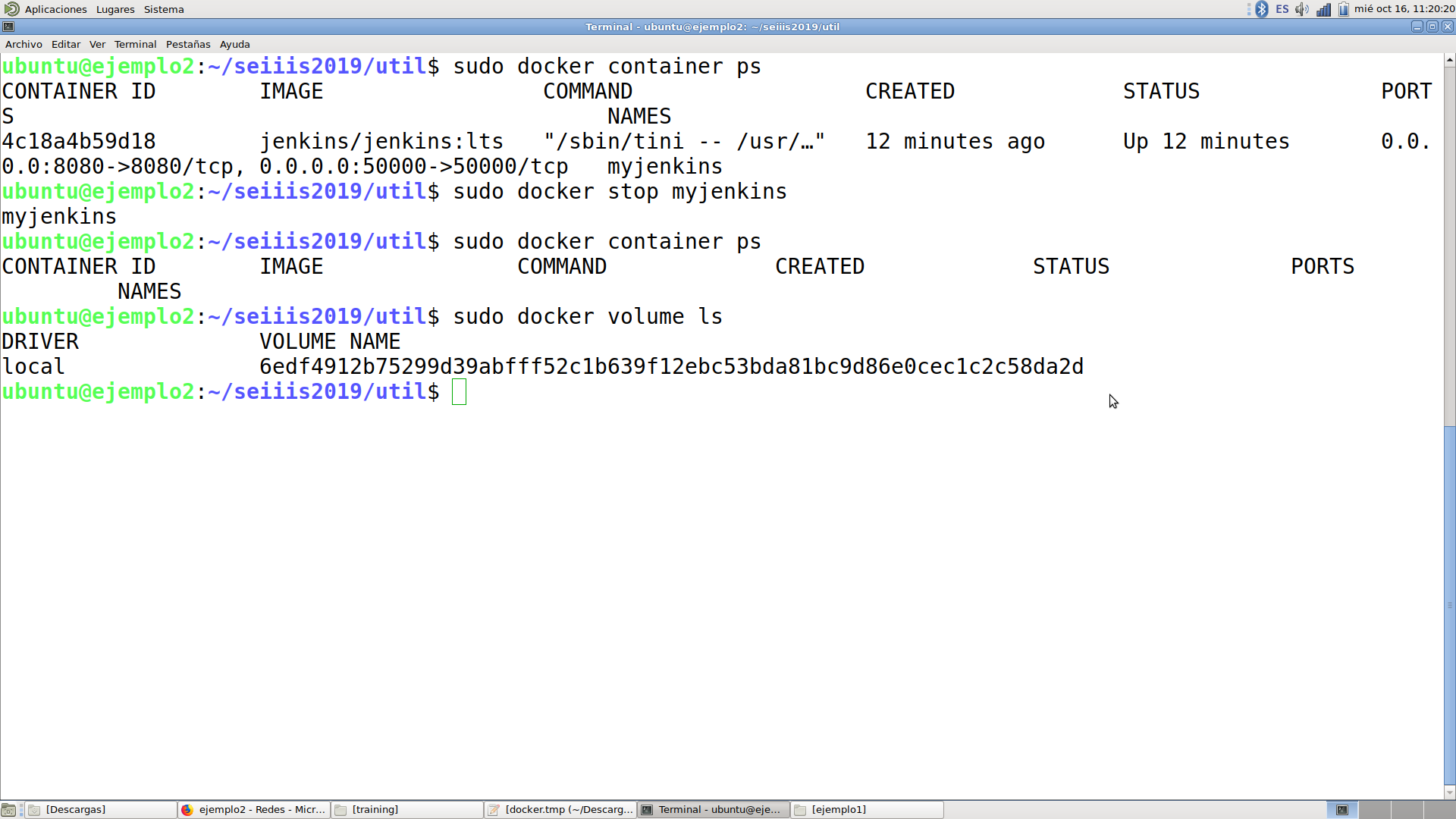




La aplicación lógicamente maneja datos, por lo cual el docker crea un volumen en el cual se almacenan esos datos. Pruebe el comando docker volume ls para ver ese volumen.

sudo docker volume ls

Aparecerá una imagen como la siguiente.



Por defecto, los volúmenes creados usan un indentificador alfanumérico hexadecimal, que docker gestiona internamente. Si se desea, se puede indicar la ruta donde se quiere crear un volumen, para poder realizar copias de respaldo o acceder de manera directa. En el siguiente ejemplo se indicará la ruta del volumen.

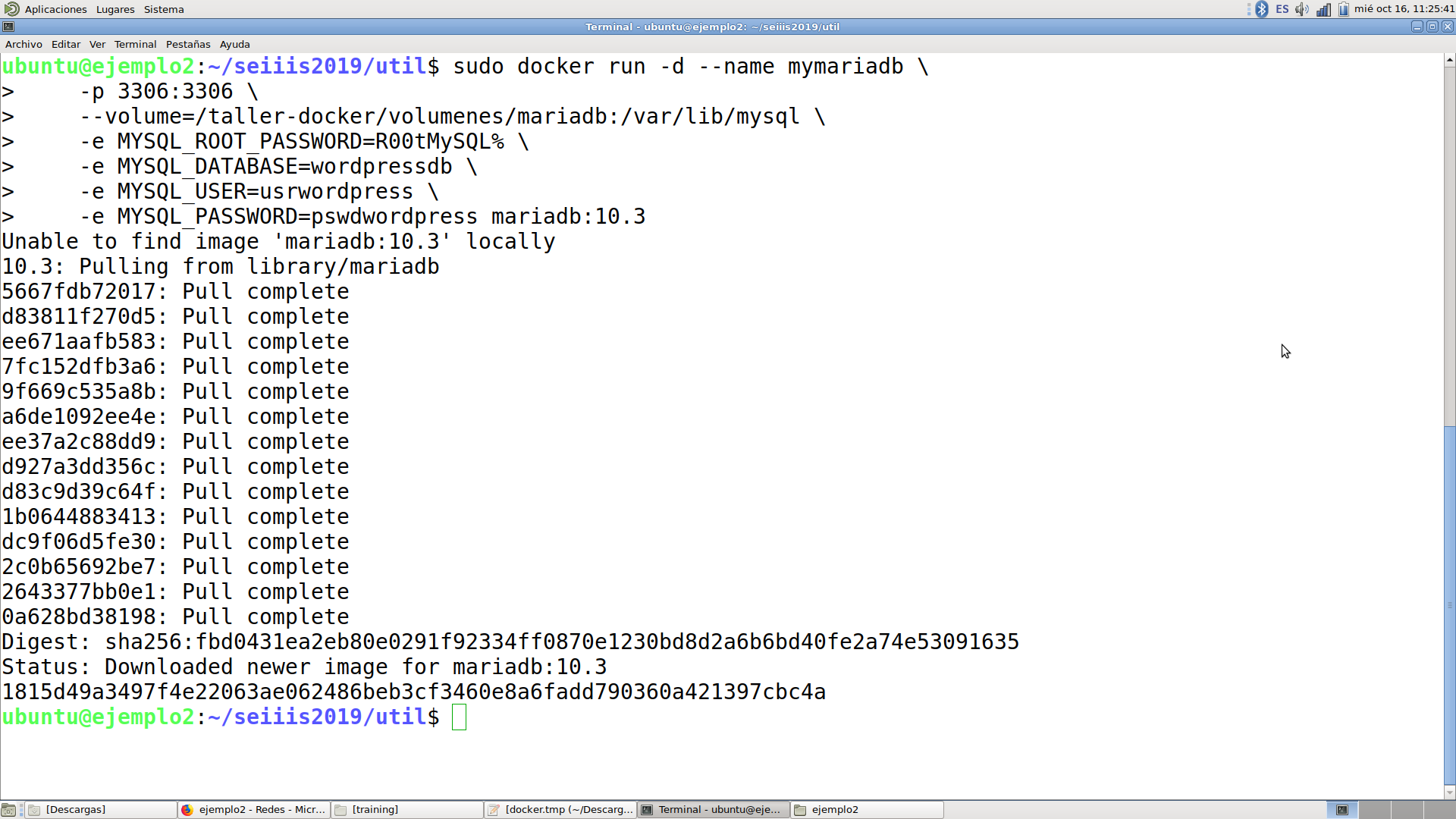
Detenga el docker de jenkins usando el siguiente comando:

sudo docker stop myjenkins

**2. Docker Base de Datos mariadb**. Ahora se va a crear un docker usando una imagen del motor de bases de datos mariadb. Use el siguiente comando

|  |
| --- |
| sudo docker run -d --name mymariadb \  -p 3306:3306 \  --volume=/taller-docker/volumenes/mariadb:/var/lib/mysql \  -e MYSQL\_ROOT\_PASSWORD=R00tMySQL% \  -e MYSQL\_DATABASE=wordpressdb \  -e MYSQL\_USER=usrwordpress \  -e MYSQL\_PASSWORD=pswdwordpress mariadb:10.3 |

Espere a que finalice, aparecerá una ventana como la siguiente:

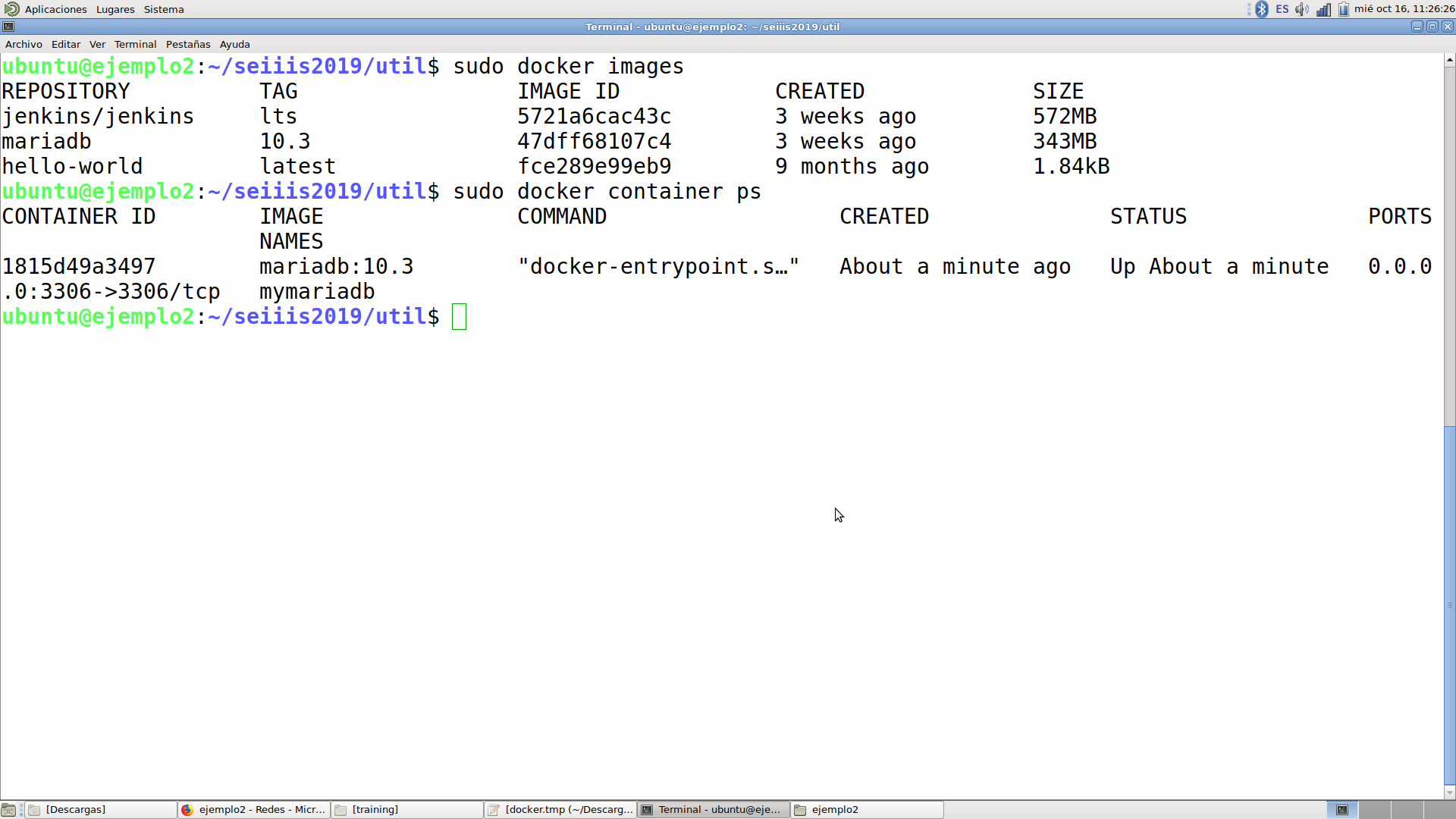


Ahora ejecute los comandos docker images y docker container ps y observe que hay tres imágenes: la imagen de Hola Mundo, la de jenkins y la de mariadb. Además, hay sólo un contenedor, el de mariadb, pues el de jenkins se detuvo previamente. Use los mismos comandos de los ejemplos anteriores:

sudo docker images

sudo docker container ps

Observará imágenes como la siguiente:



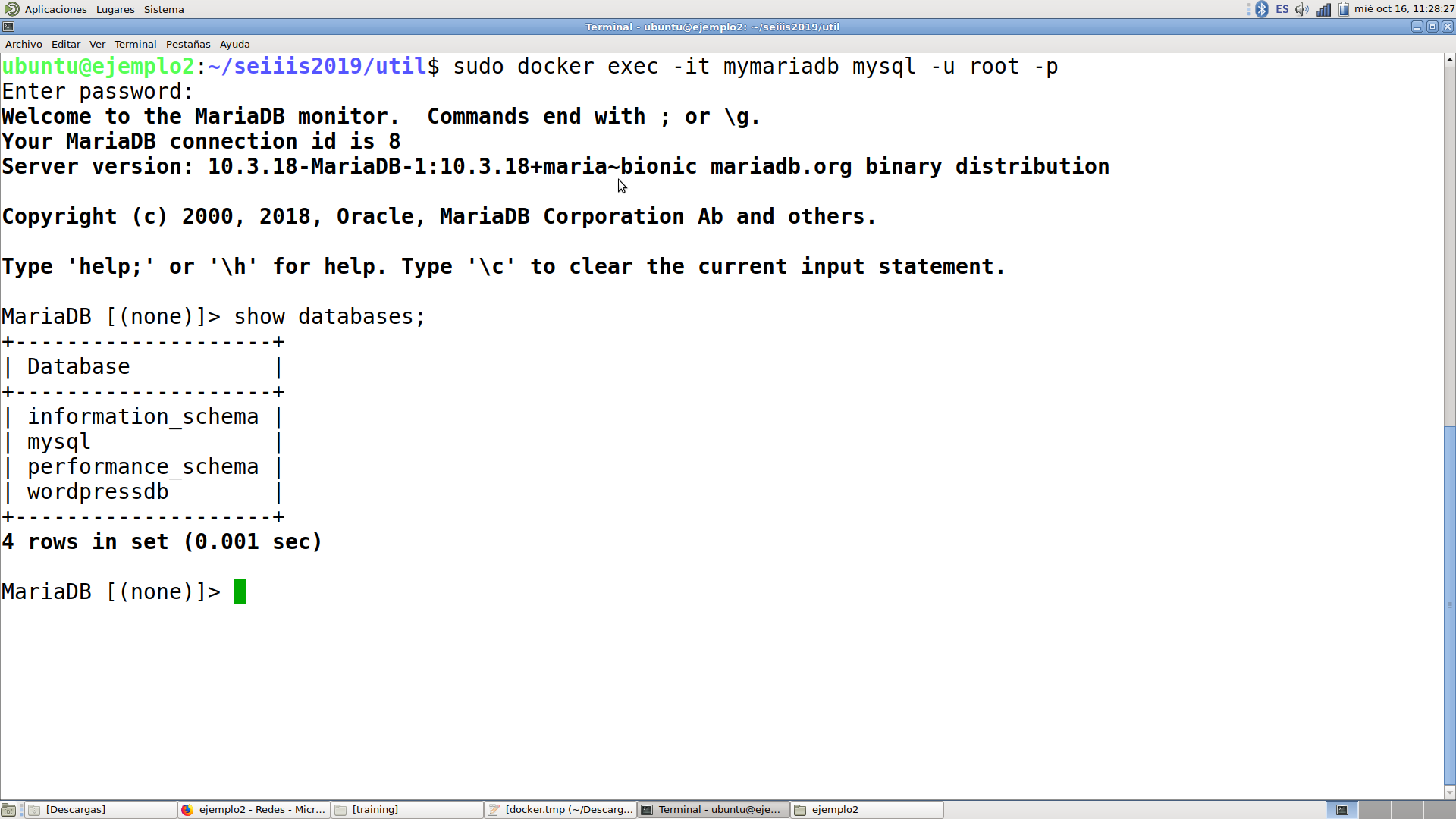
Verifique que se creó el volumen en la ruta indicada en el comando de creación de docker. Use el siguiente comando.

sudo ls /taller-docker/volumenes/mariadb/

Ahora pruebe el comando para conectarse a la base de datos de mariadb, usando el docker. Use la contraseña que aparece en el comando de creación del docker.

sudo docker exec -it mymariadb mysql -u root -p

Observará imágenes como la siguiente. Pruebe, según la imagen, listar las bases de datos del docker mariadb.



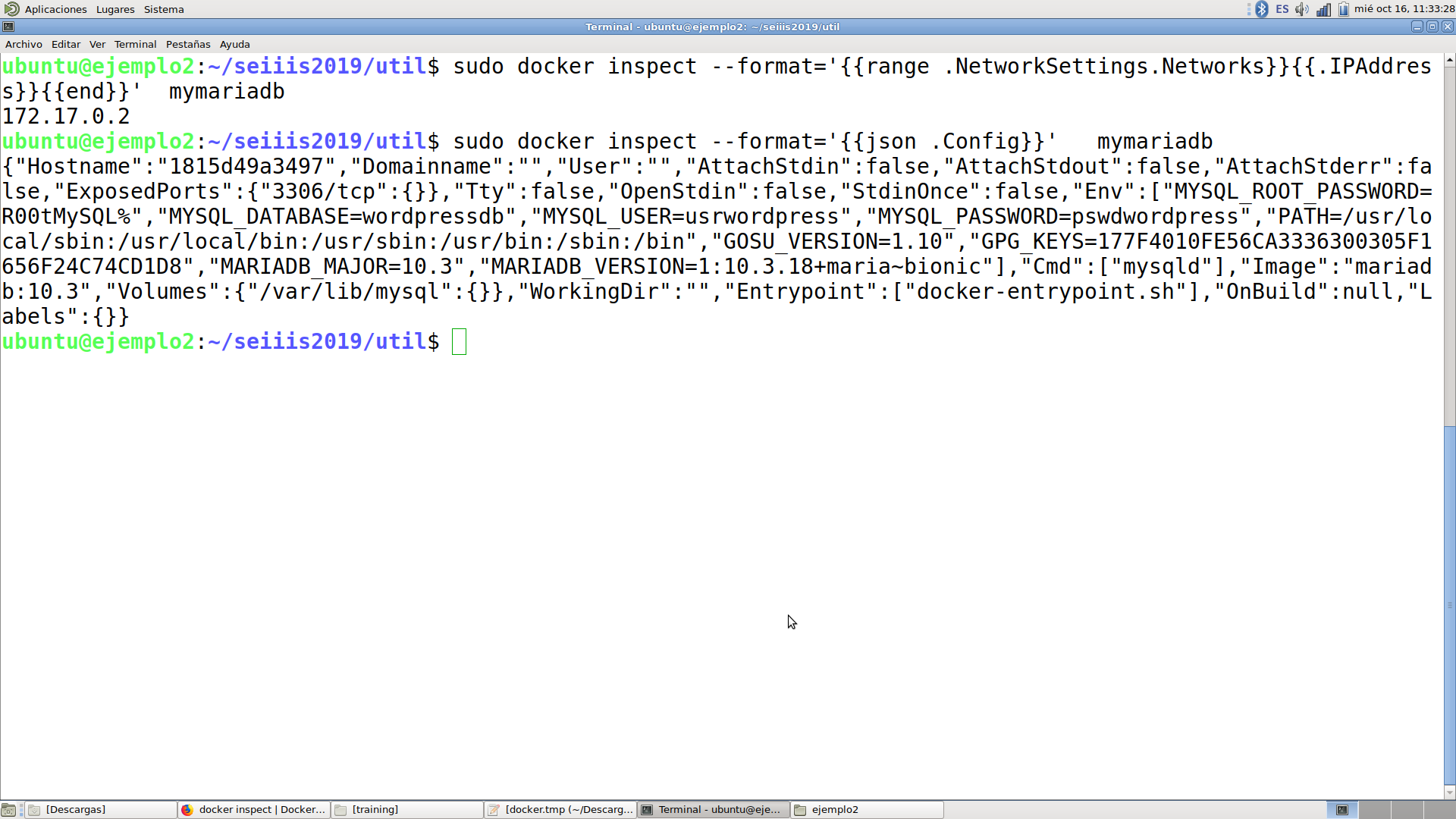
Al crear el docker se asigna una dirección IP en una red privada especial para dockers. Pruebe el comando docker inspect para ver la información del docker de mariadb. Los siguientes comandos le ayudarán a identificar fácilmente la dirección IP y otros detalles:

sudo docker inspect mymariadb

sudo docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mymariadb

sudo docker inspect --format='{{json .Config}}' mymariadb

Usualmente la dirección es 172.17.0.2, pues sólo hay un docker y la dirección 1 corresponde a la subred de dockers. Recuerde la dirección del docker para el siguiente ejemplo.



**3. Docker aplicación phpmyadmin**. Ahora se va a crear un docker usando una imagen de la aplicación web PHP phpmyadmin, que permite administrar a través de la web la base de datos de mariadb. Use el siguiente comando

|  |
| --- |
| sudo docker run -d --name myphpadmin \  -p 80:80 \  -e PMA\_HOST=172.17.0.2 \  --volume=/taller-docker/volumenes/phpmyadmin:/var/www/html \  phpmyadmin/phpmyadmin |

Espere a que finalice y pruebe todos los comandos de los ejemplos anteriores.

sudo docker images

sudo docker container ps

sudo docker inspect myphpadmin

sudo docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' myphpadmin

sudo docker inspect --format='{{json .Config}}' myphpadmin

sudo ls /taller-docker/volumenes/phpmyadmin

Luego ingrese al navegador y verifique que la aplicación funciona correctamente. Use la contraseña que aparece en el comando de creación del docker de mariadb.

Puesto que los docker de mariadb y de phpmyadmin se crearon usando volúmenes, los datos quedarán respaldados allí incluso cuando se eliminan los docker.

**4. Docker aplicación wordpress**. Ahora se va a crear un docker usando una imagen del popular Gestor de Contenidos (CMS) wordpress, que usa apache, PHP y mariadb. Use el siguiente comando

|  |
| --- |
| sudo docker run -d --name mywordpress \  --volume=/taller-docker/volumenes/wordpress:/var/www/html \  -e WORDPRESS\_DB\_HOST=172.17.0.2 \  -e WORDPRESS\_DB\_USER=usrwordpress \  -e WORDPRESS\_DB\_PASSWORD=pswdwordpress \  -e WORDPRESS\_DB\_NAME=wordpressdb \  -p 80:80 \  wordpress |

Espere a que finalice y pruebe todos los comandos de los ejemplos anteriores.

sudo docker images

sudo docker container ps

sudo docker inspect myphpadmin

sudo docker inspect --format='{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' myphpadmin

sudo docker inspect --format='{{json .Config}}' myphpadmin

sudo ls /taller-docker/volumenes/phpmyadmin

Luego ingrese al navegador y verifique que la aplicación funciona correctamente. Use la contraseña que aparece en el comando de creación del docker. Termine el proceso de instalación y configuración.

Felicitaciones, ya ha usado cinco (5) imágenes de contenedores y ha desplegado cinco (5) contenedores: hola mundo, mariadb, jenkins, phpmyadmin y wordpress. Además, los dockers de phpmyadmin y wordpress se comunican y usan el docker de mariadb.

**5. Docker SO Oracle linux**. Ahora se va a crear un docker usando una imagen del popular Sistema Operativo Oracle Linux, en el cual podría desplegar bases de datos y aplicaciones usando los marcos de trabajo y herramientas de Oracle (aunque sólo para aprendizaje por temas de licencias).

sudo docker run -it oraclelinux /bin/bash

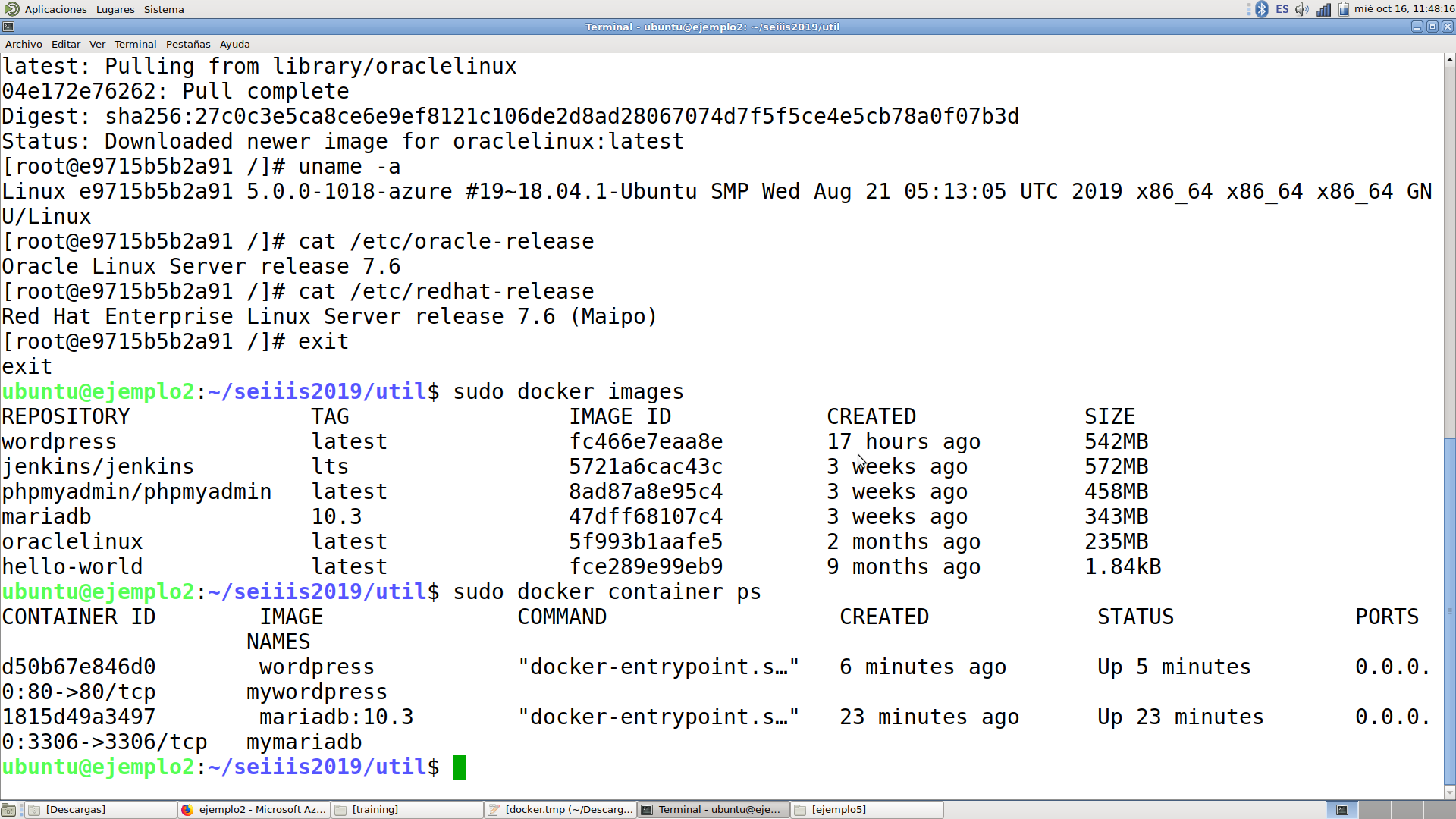
En este caso el docker no se ejecuta como un proceso persistente o demonio (dameon) sino que se ejedcuta de manera interactiva. Por lo tanto, ahora no está en su servidor sino en un servidor “dockerizado”, una abstracción o virtualización del sistema operativo que se maneja con docker. Pruebe dentro del contenedor los siguientes comandos:

uname -a

cat /etc/oracle-release

cat /etc/redhat-release

Pruebe todos los comandos de los ejemplos anteriores para revisar todas las imágenes y contenedores:



Detenga todos los contenedores, usando el comando docker stop, seguido de los nombres de todos los contenedores.