# –Docker

Contenido

[Docker 1](#_Toc44955117)

[Que es Docker 1](#_Toc44955118)

[Instalación 2](#_Toc44955119)

[Imágenes en Docker 7](#_Toc44955120)

[Descarga de una imagen de Docker Hub 7](#_Toc44955121)

[Ejemplo: 13](#_Toc44955122)

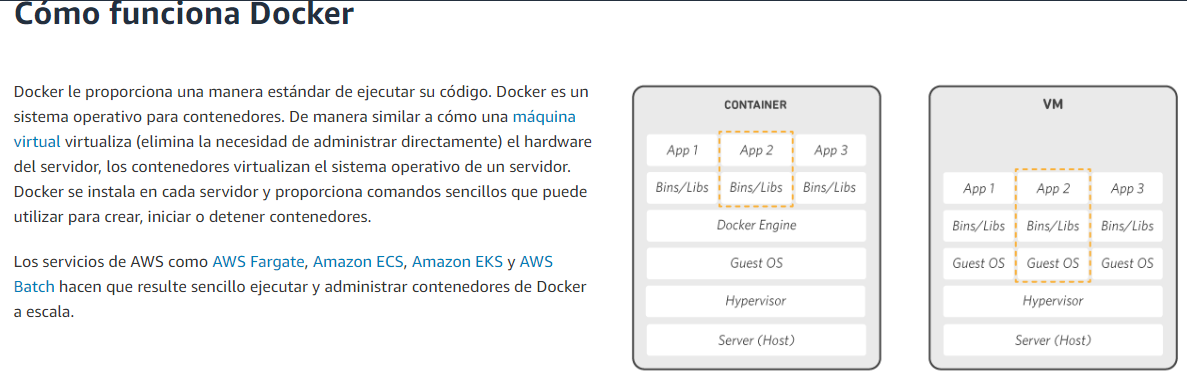
[Contenedor con apache desde cero con la imagen de Ubuntu. 16](#_Toc44955123)

[DockerFile 18](#_Toc44955124)

[Uso 18](#_Toc44955125)

## Que es Docker

Docker es una plataforma de software que le permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas [contenedores](https://aws.amazon.com/es/containers/) que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con Docker, puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

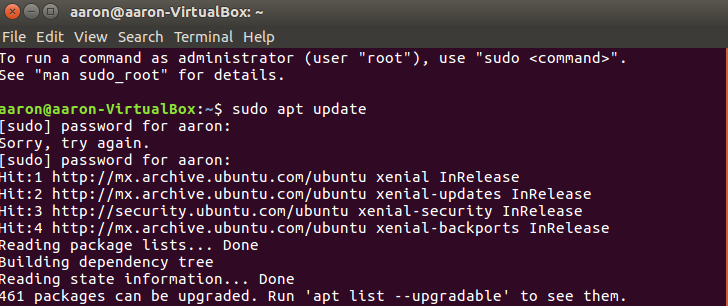


Fuente: <https://aws.amazon.com/es/docker/>

## Instalación

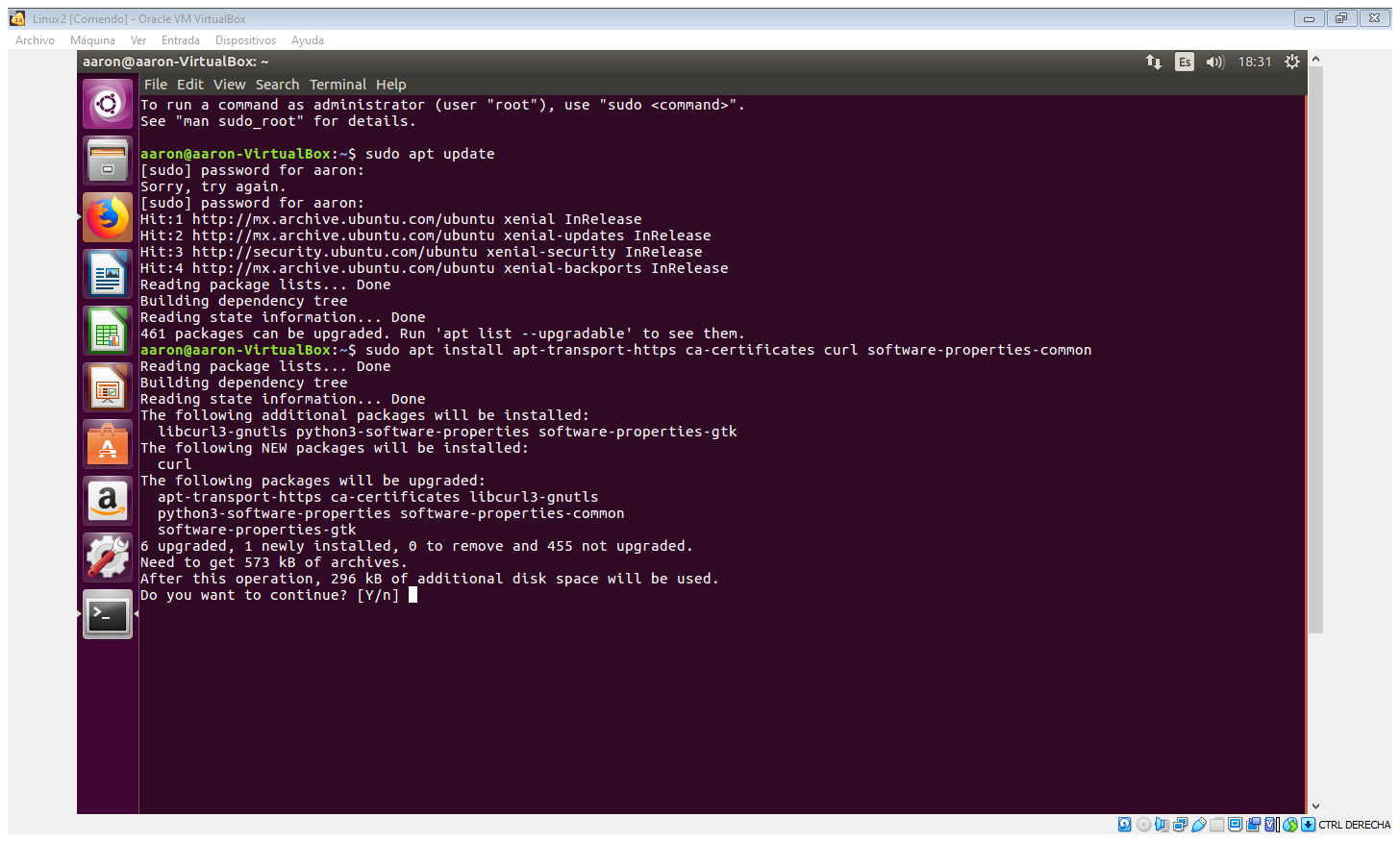
Primero, actualice su lista de paquetes existente:

* sudo apt update



A continuación, instale algunos paquetes de requisitos previos que le permiten a apt usar paquetes mediante HTTPS:

* sudo apt install apt-transport-https ca-certificates curl software-roperties-common



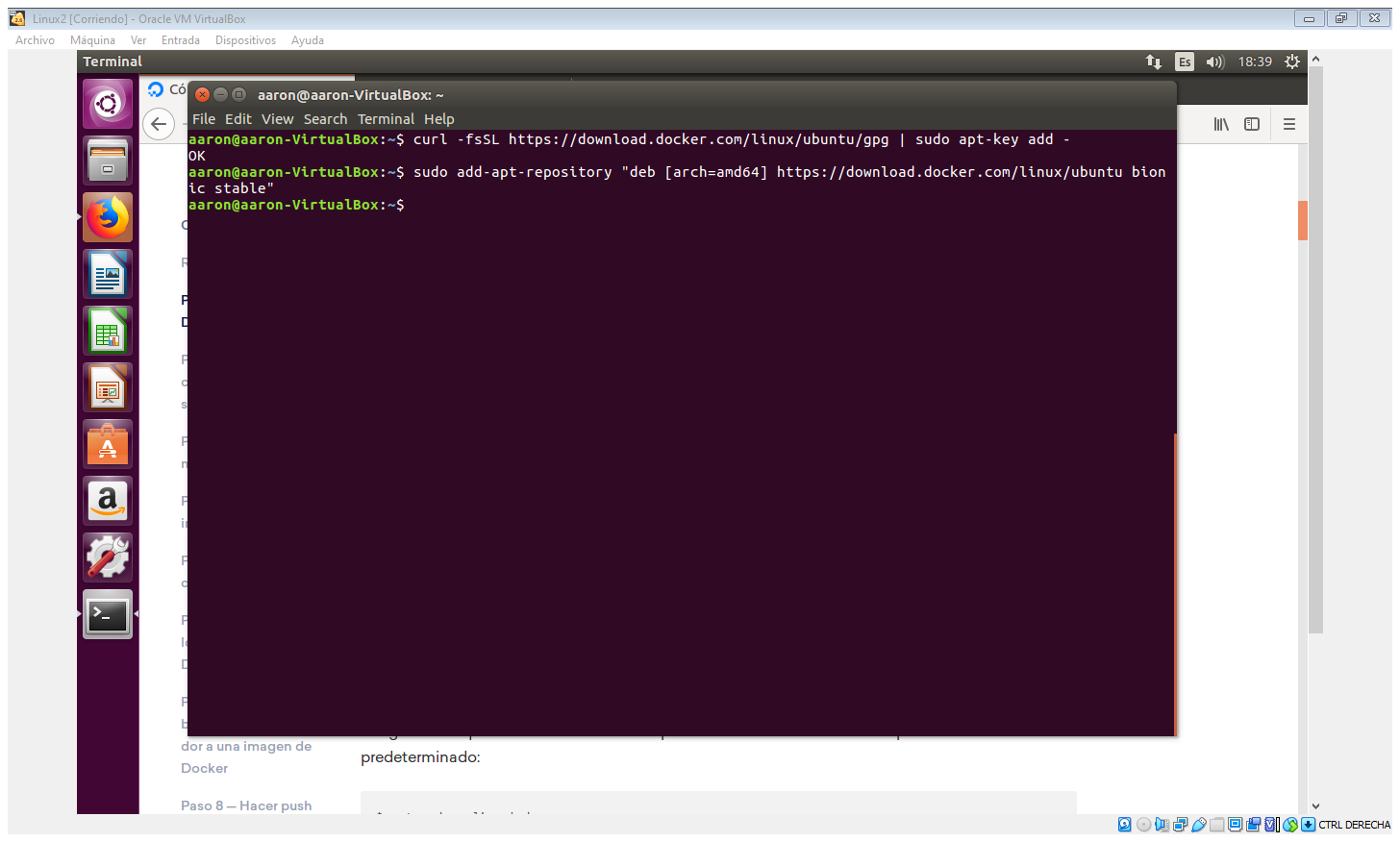
Luego, agregue la clave GPG para el repositorio oficial de Docker a su sistema:

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



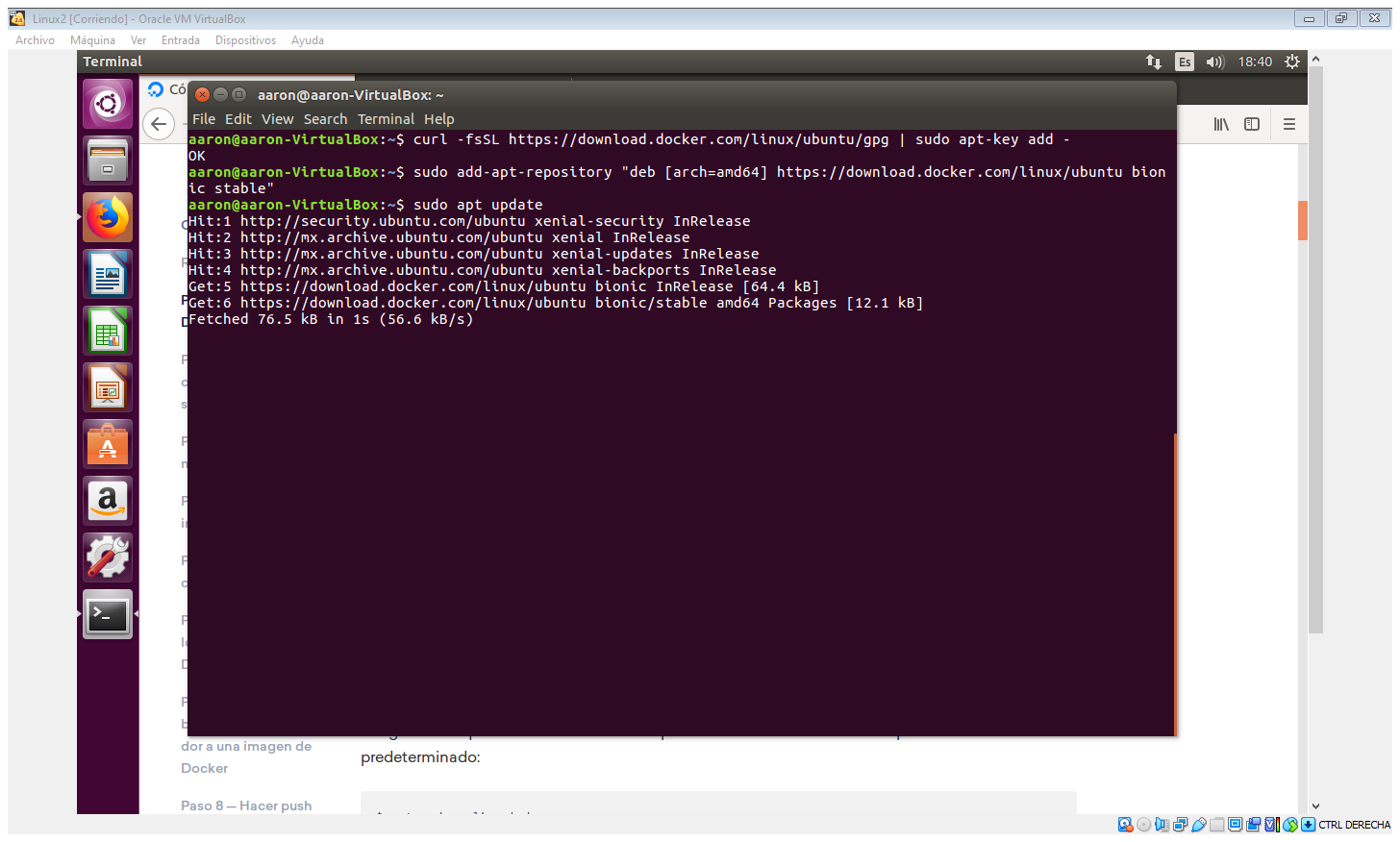
Agregue el repositorio de Docker a las fuentes de APT:

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



Posteriormente, actualice la base de datos de paquetes usando los paquetes de Docker del repositorio que acaba de agregar:

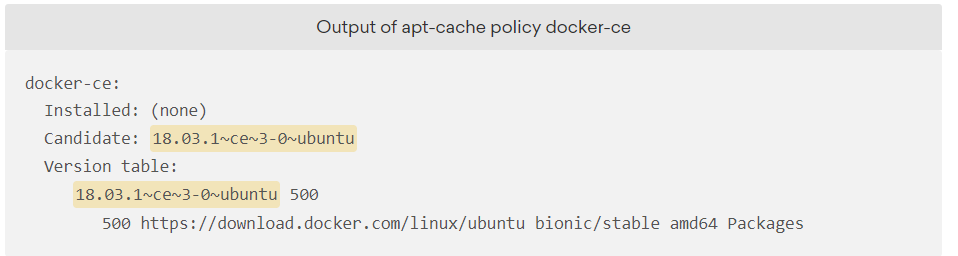
sudo apt update

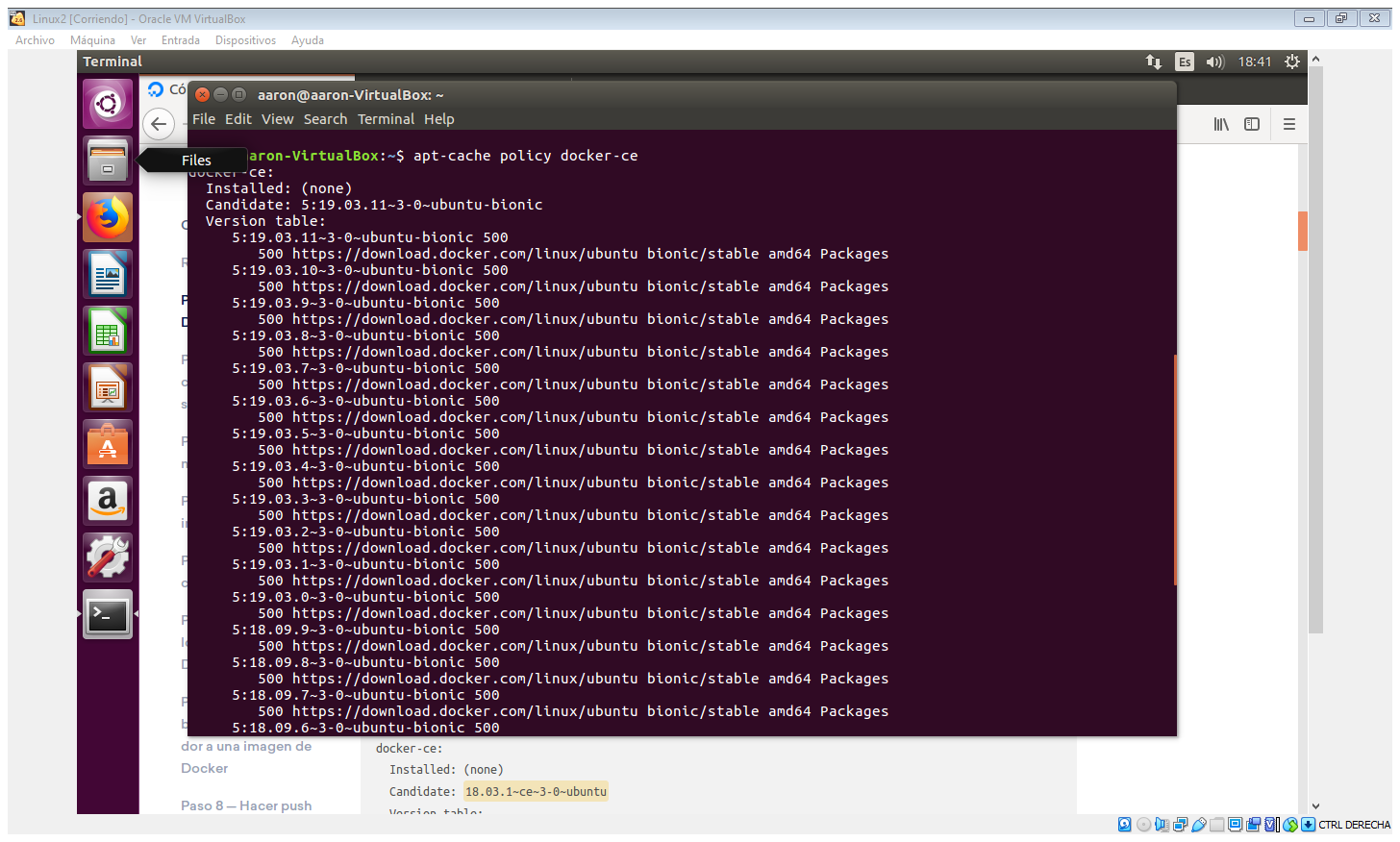


Asegúrese de que va a instalar desde el repositorio de Docker en vez del repositorio de Ubuntu predeterminado:

apt-cache policy docker-ce

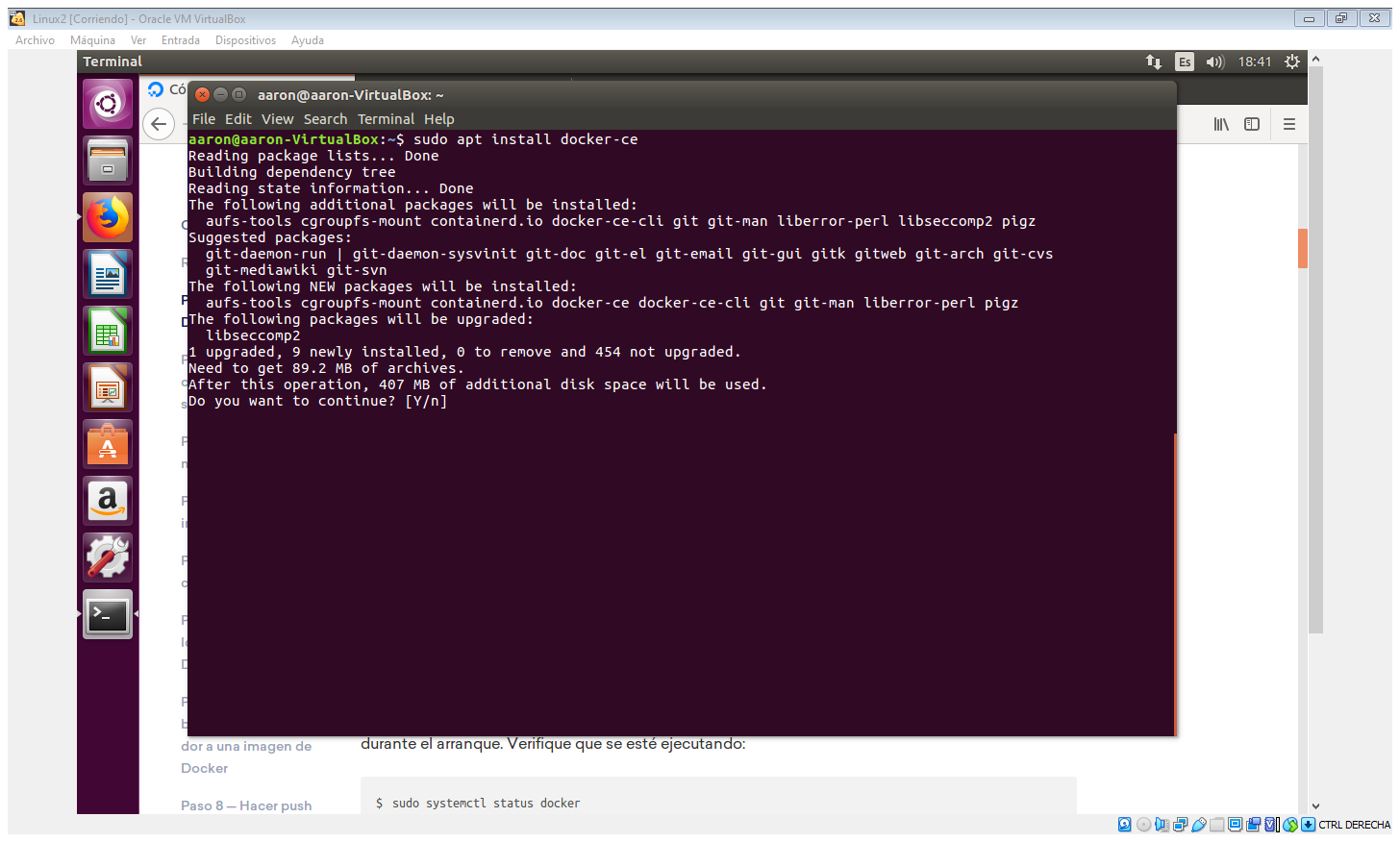
Verá un resultado como este, aunque el número de versión de Docker puede variar:

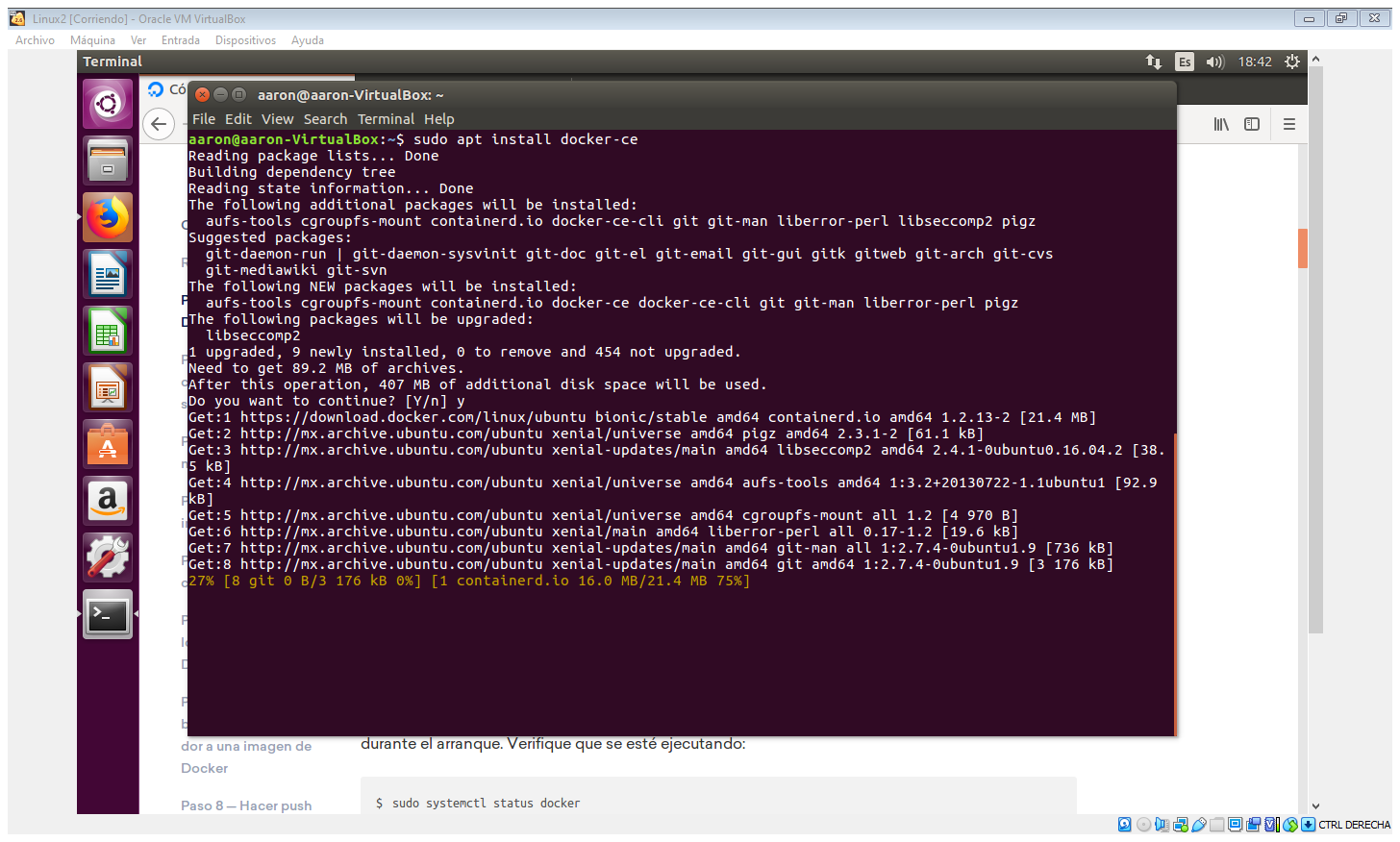




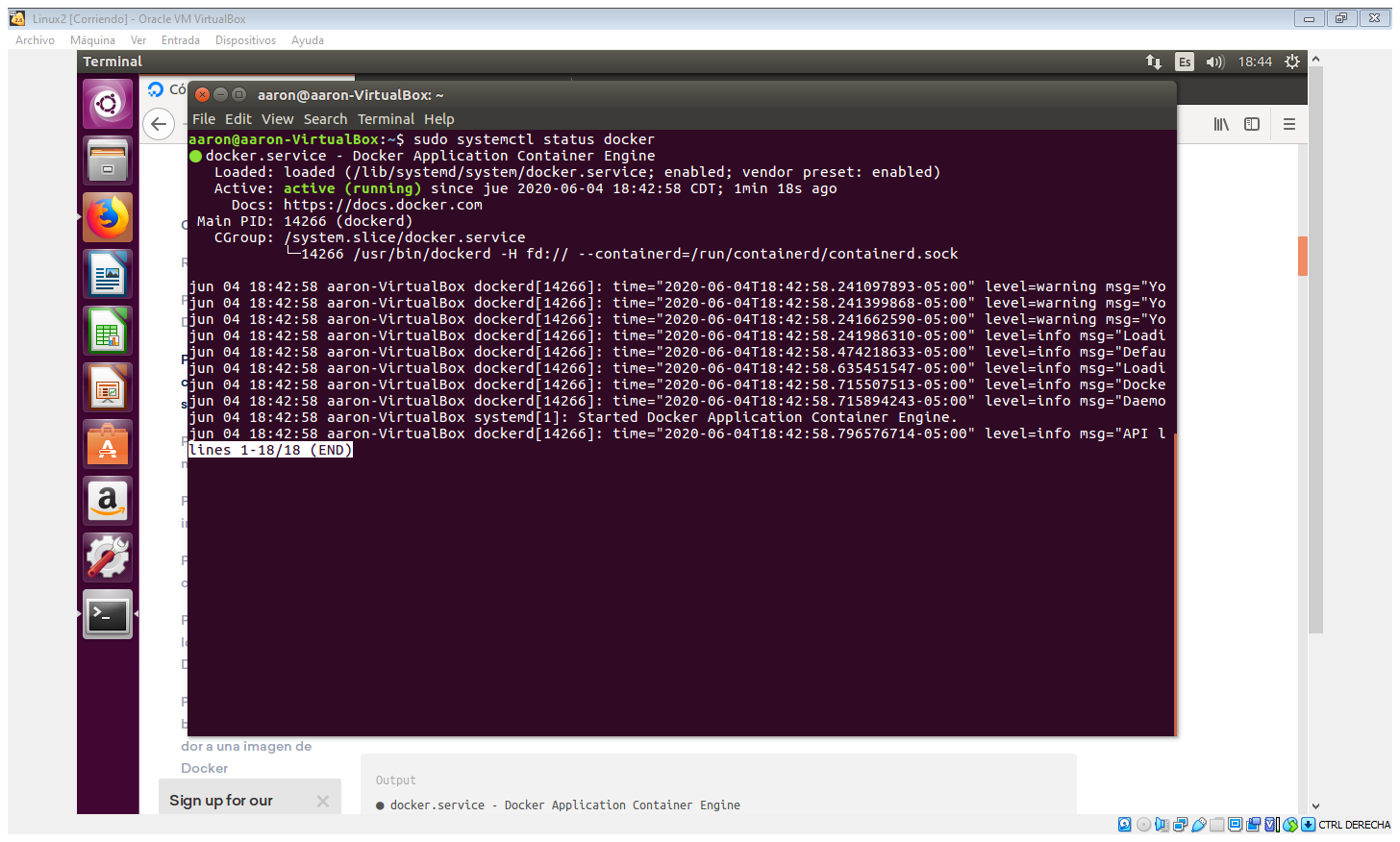
Por último, instale Docker:

* sudo apt install docker-ce

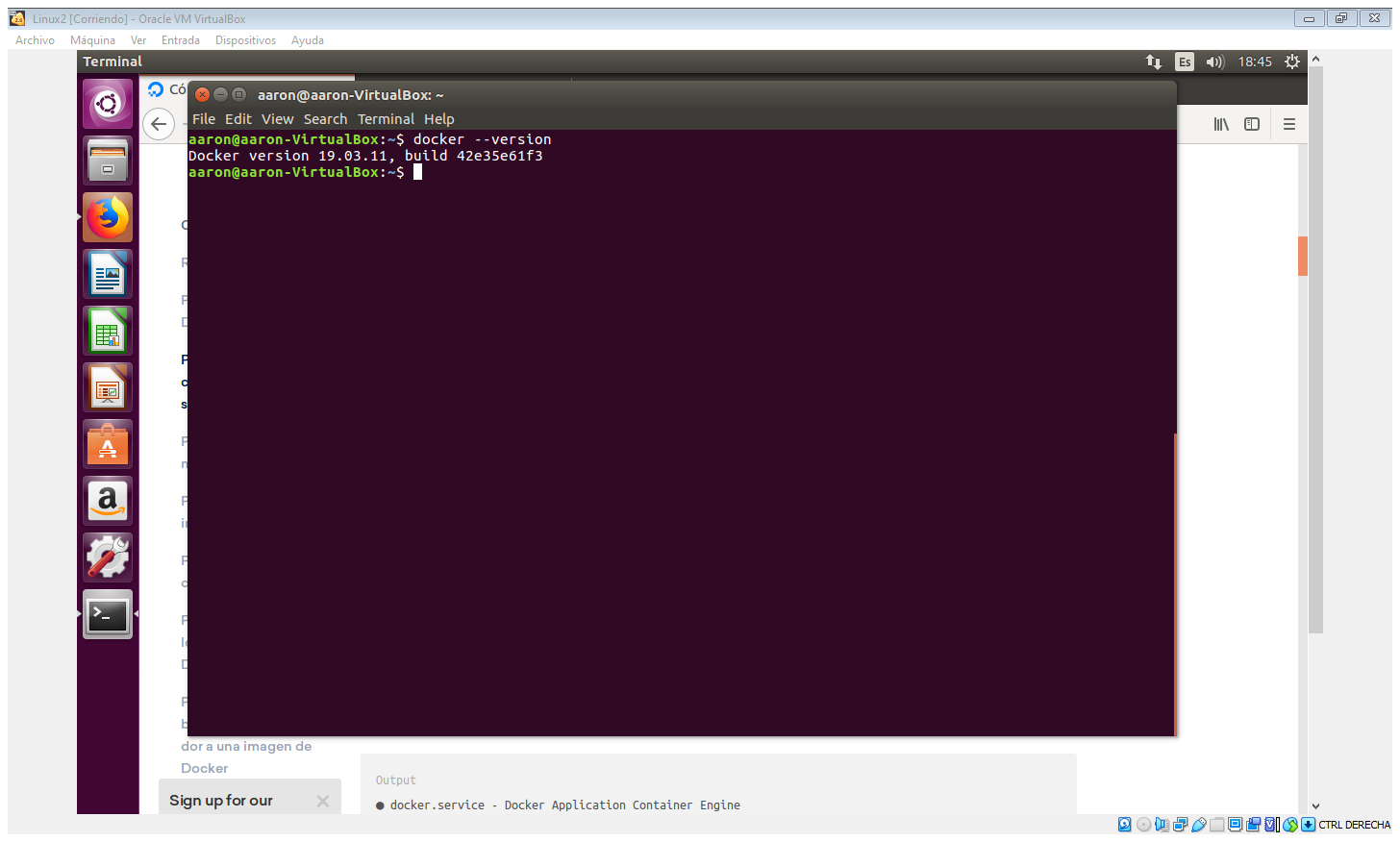




Ahora debería tener Docker instalado, el daemon iniciado, y el proceso habilitado para iniciar durante el arranque. Verifique que se esté ejecutando:

* sudo systemctl status Docker
* 

Ya se encuentra corriendo



## Imágenes en Docker

<https://hub.docker.com/>

Ya se encuentran diferentes imágenes creadas para nosotros poder crear nuestros contenedores.

***Se recomienda crear una cuenta***

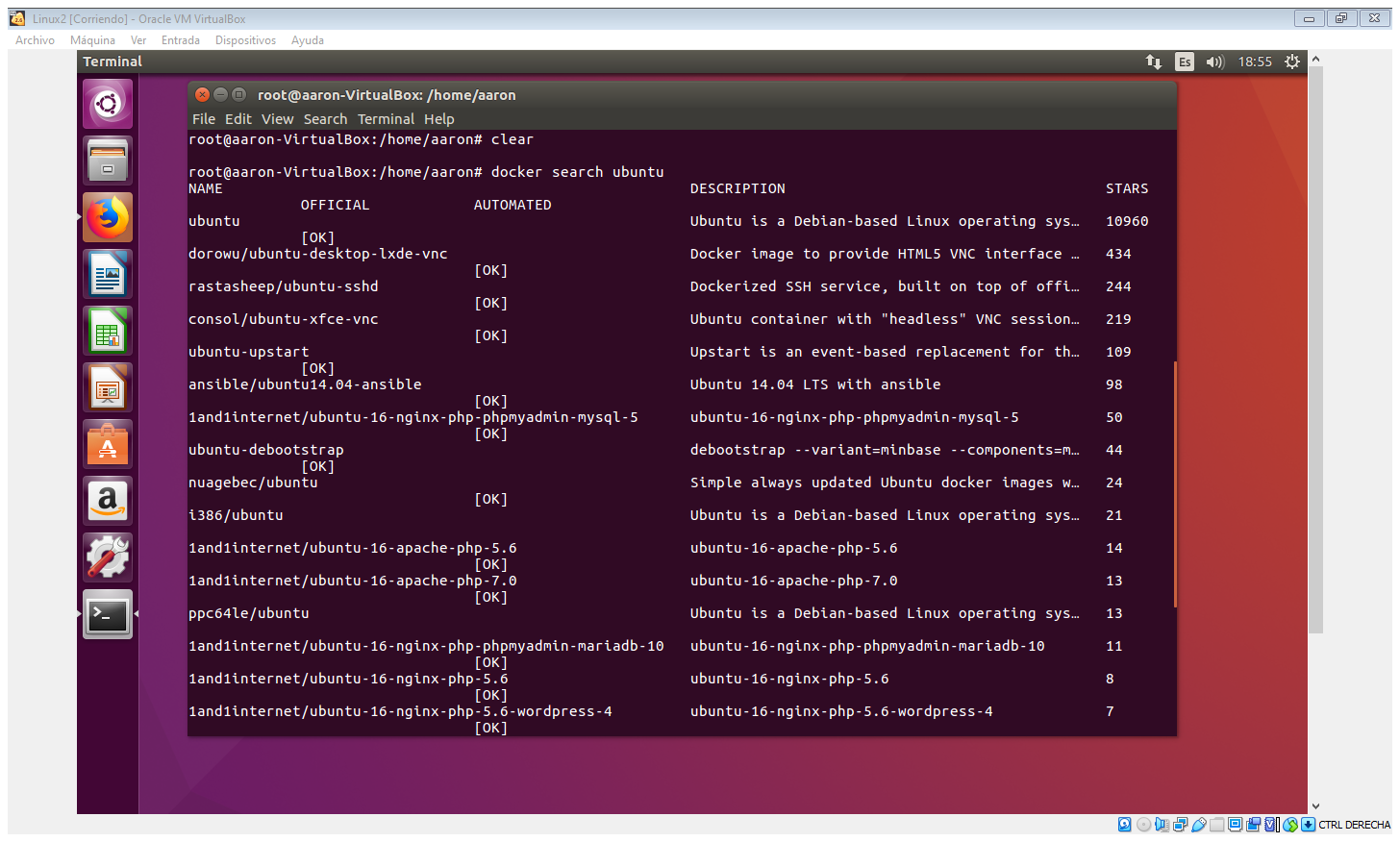
## Descarga de una imagen de Docker Hub

Comandos básicos

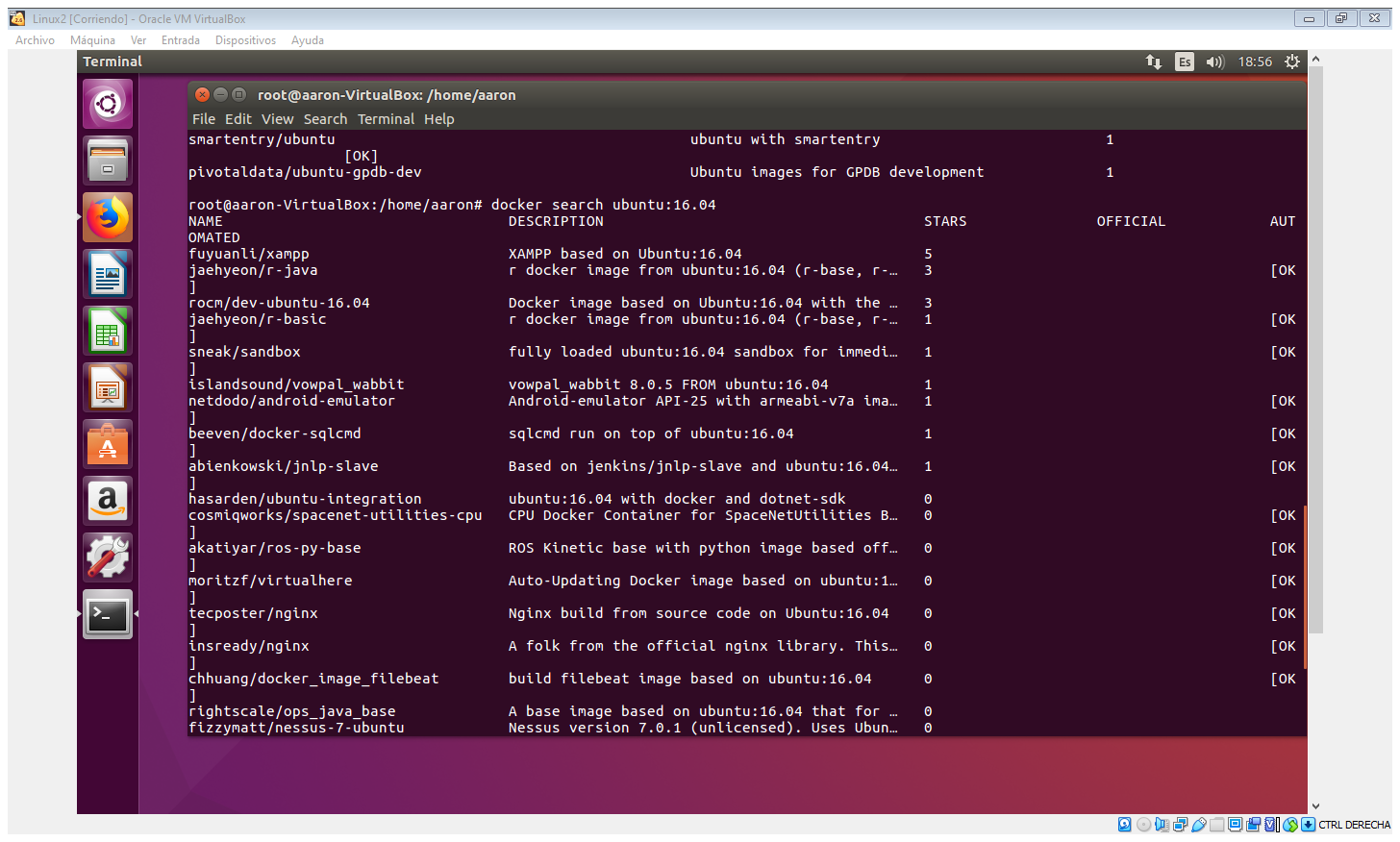
**Docker search [name:tag]**

Ejemplos:

#### docker search Ubuntu

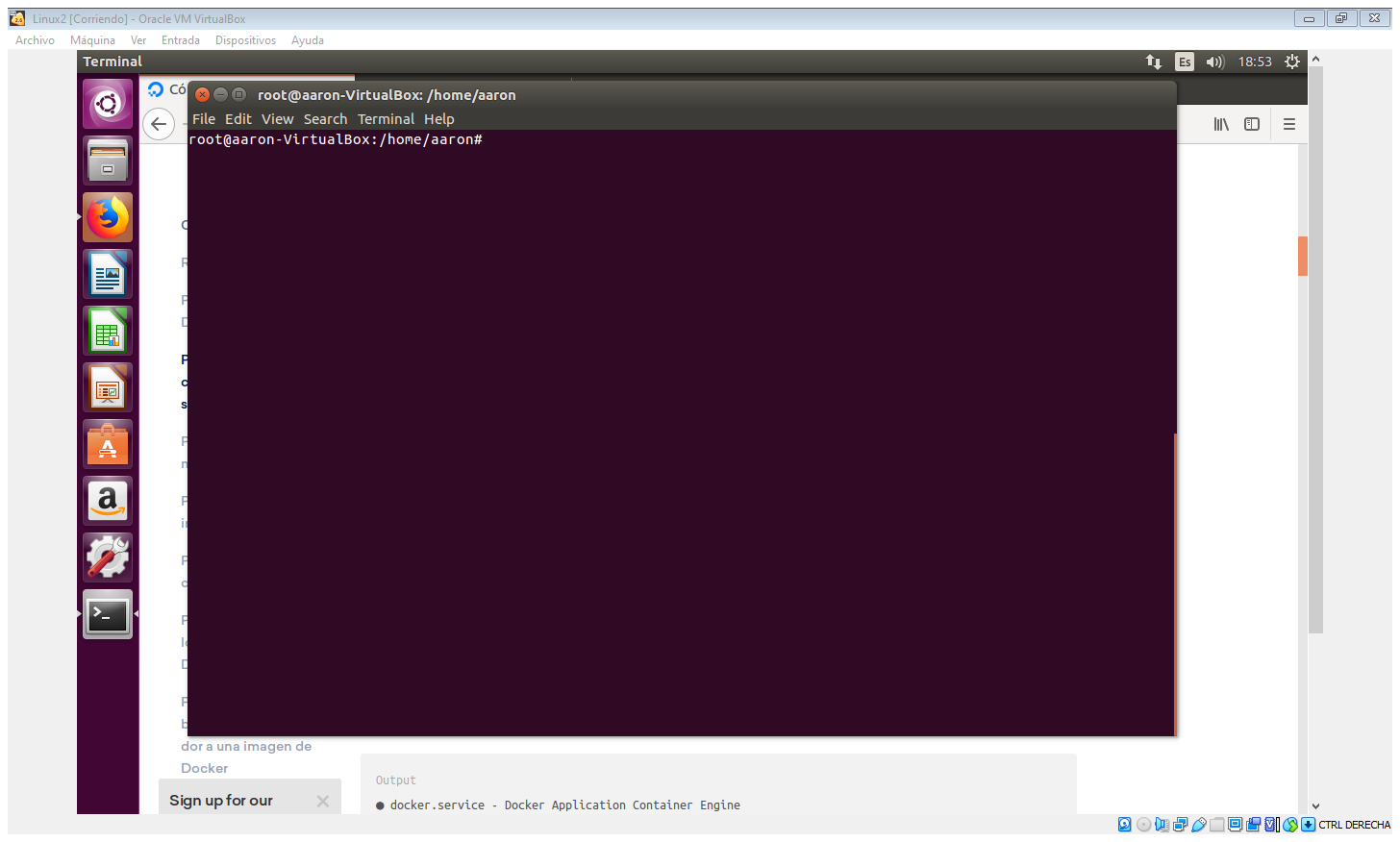


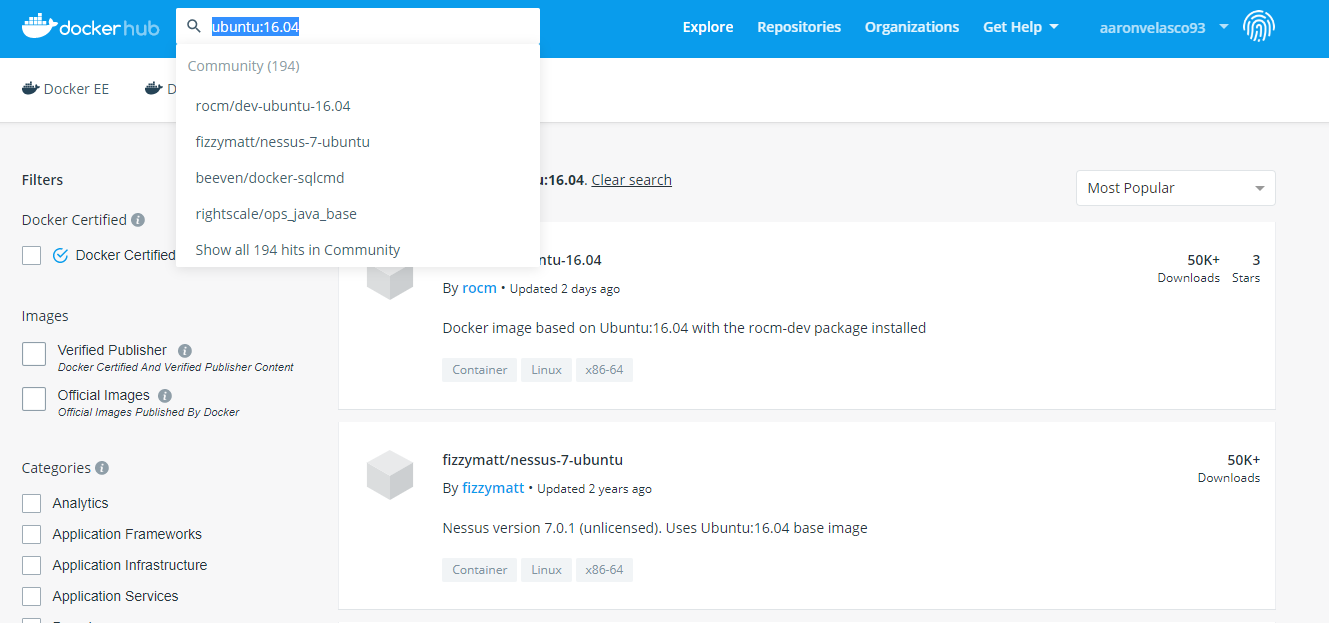
#### docker search Ubuntu:16.04



Sirve para buscar imágenes, así como se puede poner un tag para hacer la búsqueda mas exacta

Para poder usar comando referente a Docker es recomendable estar como super usuario.





#### Docker pull [name:tag]

Comando sirve para poder hacer una descarga de una imagen ya construida.

O también se puede visualizar desde Docker hub

#### docker images

Sirve para listar imágenes

#### docker ps

Ver que contenedores e están ejecutando

#### docker ps -a

Lista los contenedores que se habían ejecutado

#### docker run [name image]

Sirve para ejecutar una imagen

#### docker run -i -t [name] bash

Sirve para ejecutar una maquina con acceso a la terminal

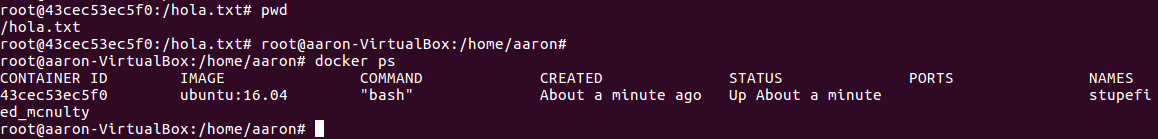
-i //interativo

-t //acceso a una terminal

bash // tenerminal

Para poder salir de un contenedor sin pararlo usar la combinación

ctrl + p + q



En las imagenes se puede ver como estamos dentro de un contenedor pero sigue ejecutando el contenedor eso lo corroboramos con “docker ps”

Docker stop [id de contenedor]

Sirve para detener un contenedor



#### docker start [idContenedor]

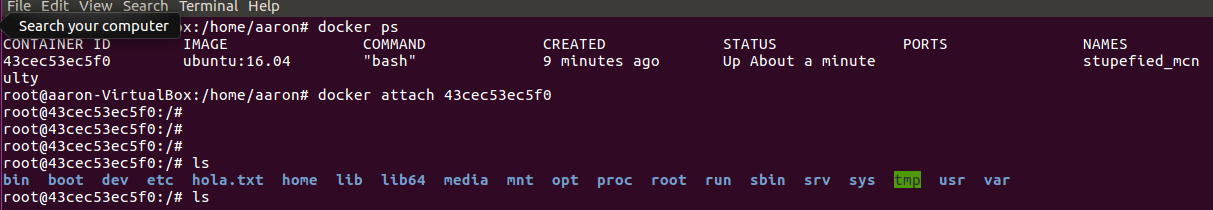
Sirve para ejecutar un contendor ya creado solamente ejecutarlo.

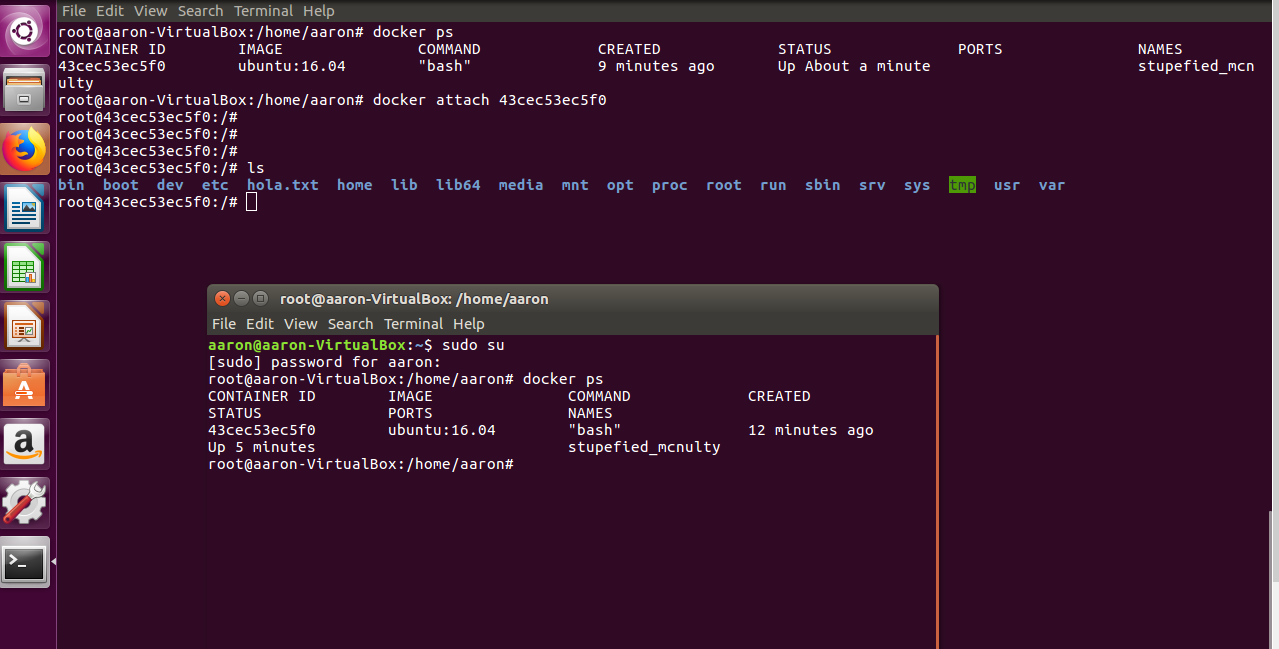


Se listaron los contenedores ya creados, después se ejecuto con contenedor ya creado con “Docker ps” se corrobora que ya este encendido

#### docker attach [id del contenedor]

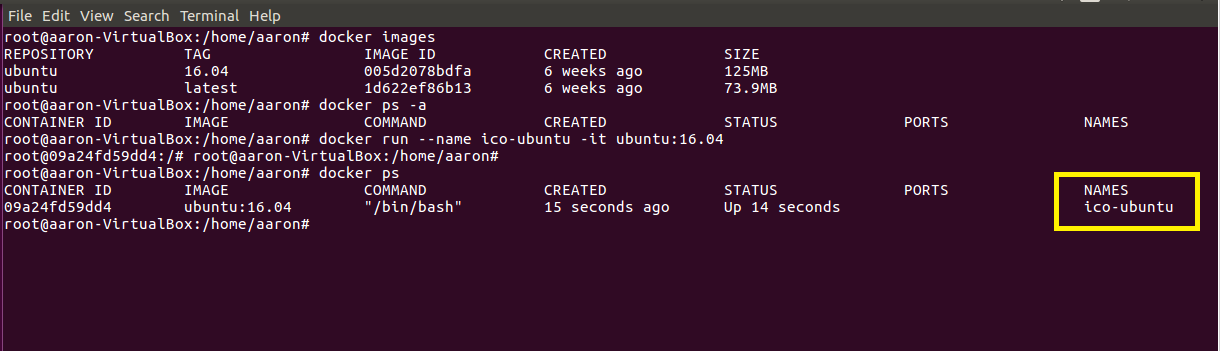
Con este comando puedo entrar al contenedor para poder trabajar sobre el





#### docker run --name [nombre del contenedor] -it [imagen]

Crea un contenedor con nombre y que sea iterativo

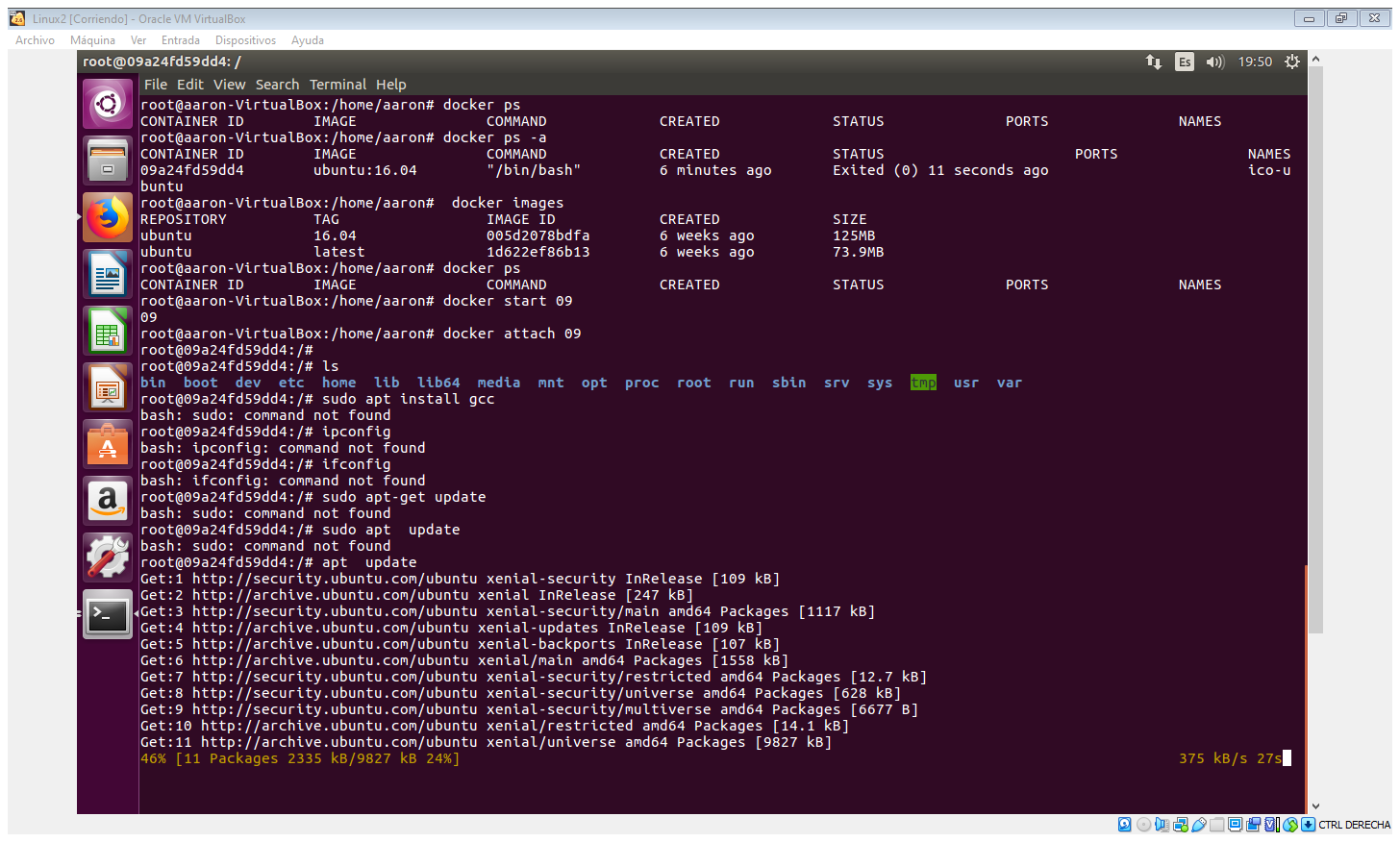


Comando

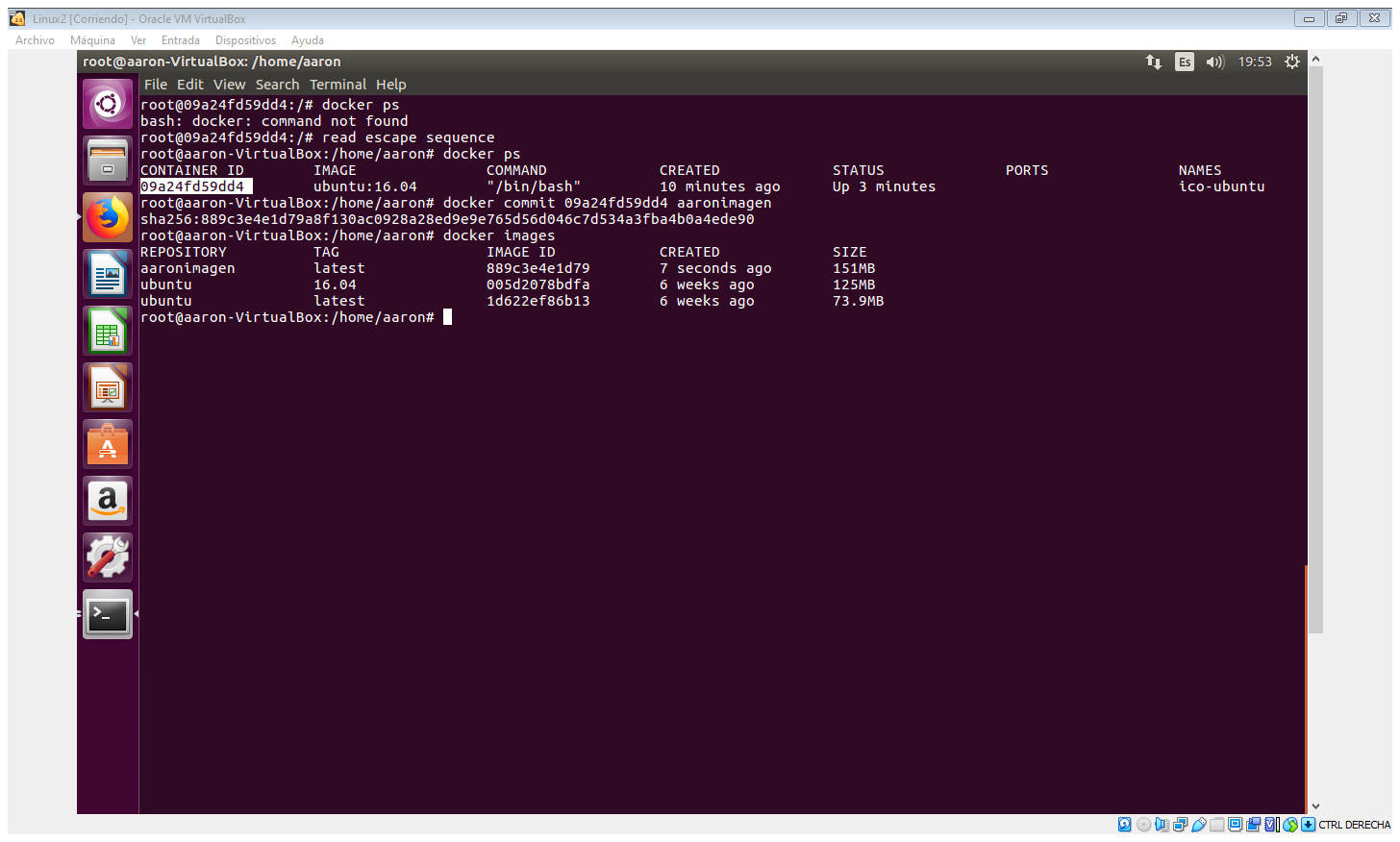
* Docker --name ico-ubuntu -it Ubuntu:16.04

#### docker commit [id del contenedor que se quiere hacer su imagen] [nombre de la imagen]

Nos permite crear imágenes a partir de un contenedor ya modificado.



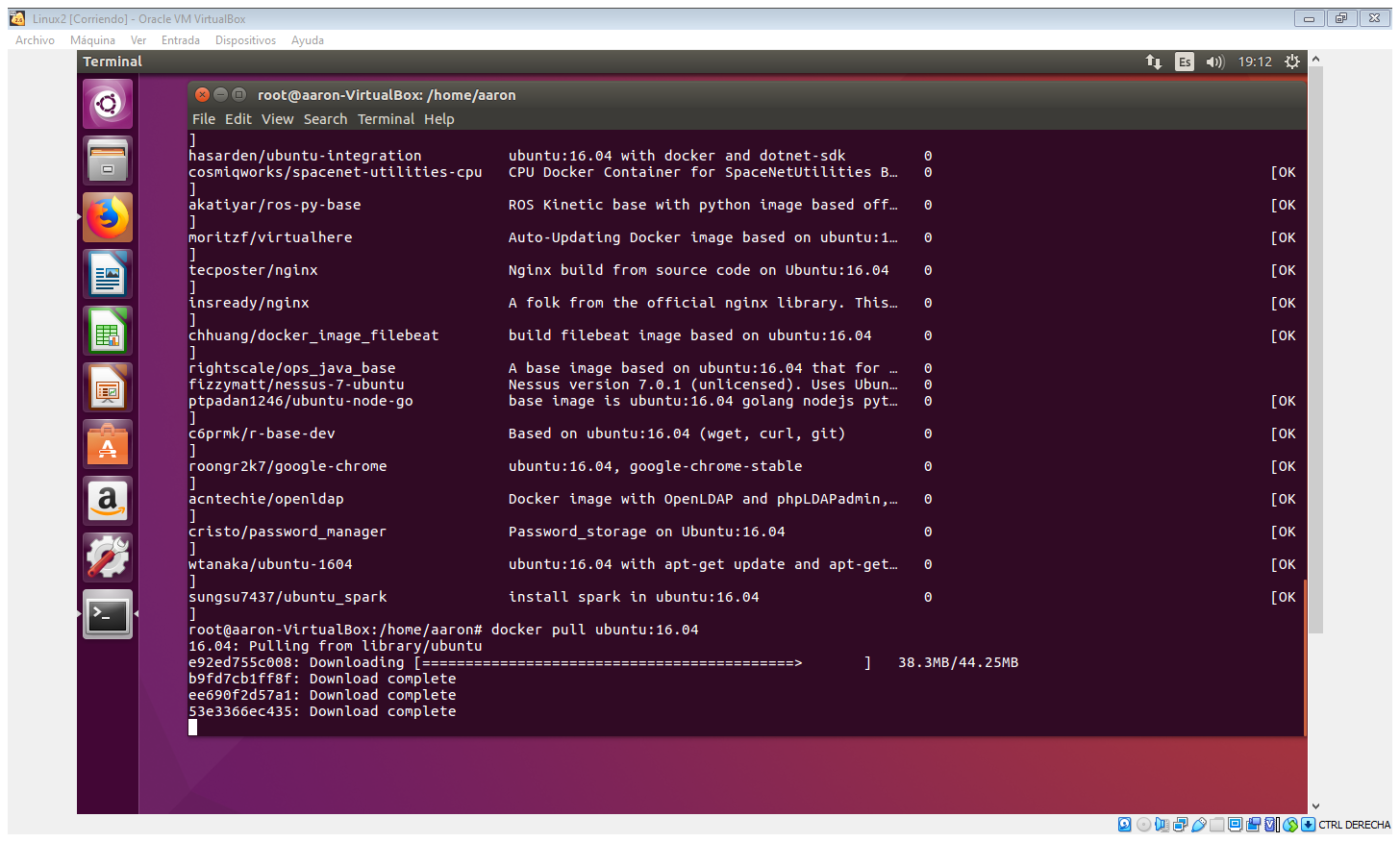
Actualiza el contenedor y se instalan las aplicaciones que se quieran implementar



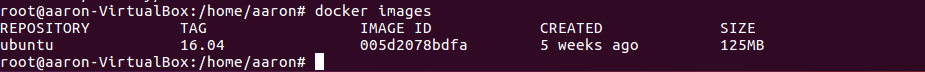
Nueva imagen

## Ejemplo:

Descarga una imagen de Ubuntu 16.04



Se listan las imágenes



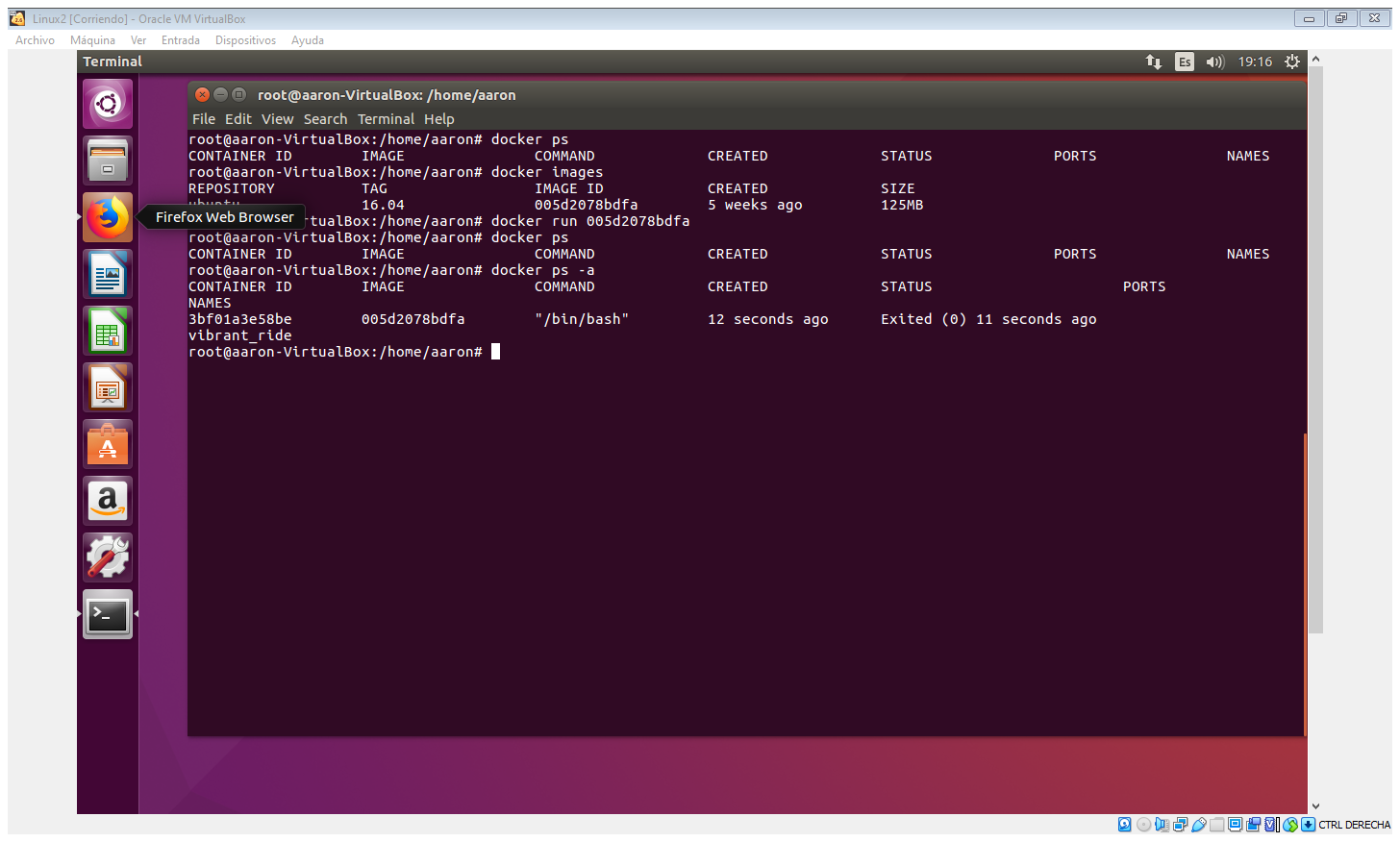
Solo existe una

Se ve si ay un contenedor iniciado

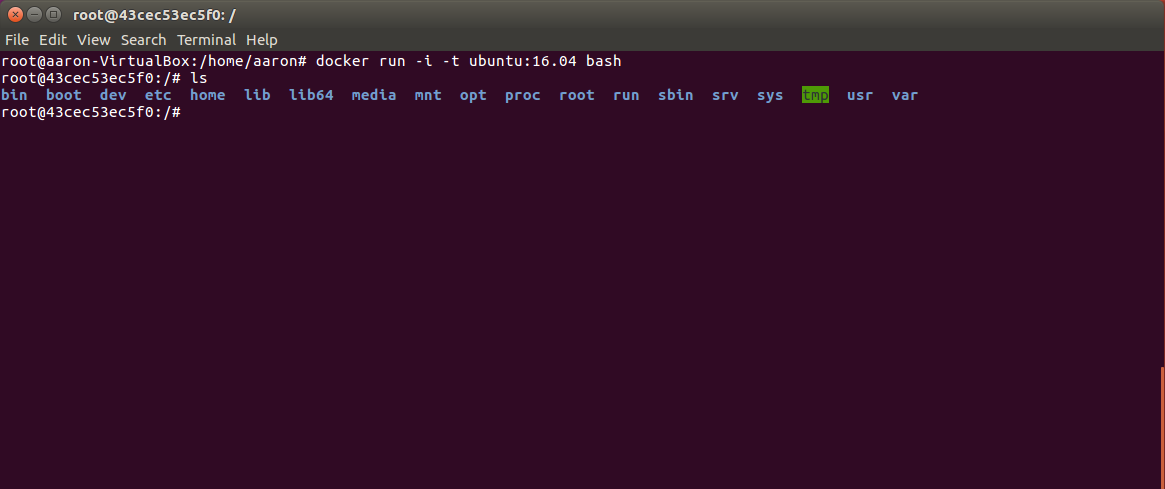


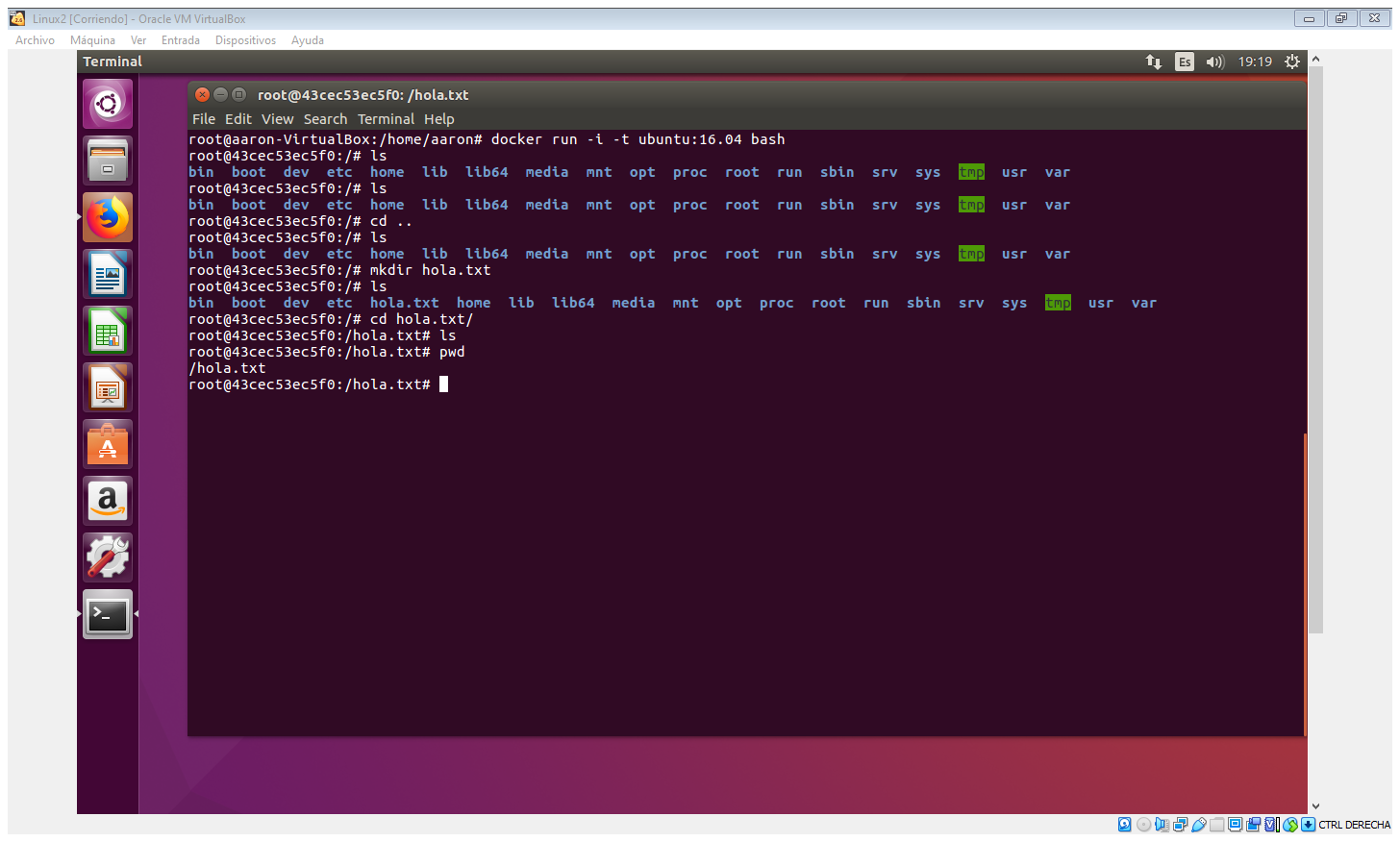
Ninguno

se corre un nuevo contenedor a partir de una imagen, ese contenedor solo mantuvo un lapso de vida ya que no tenía ninguna acción programada.

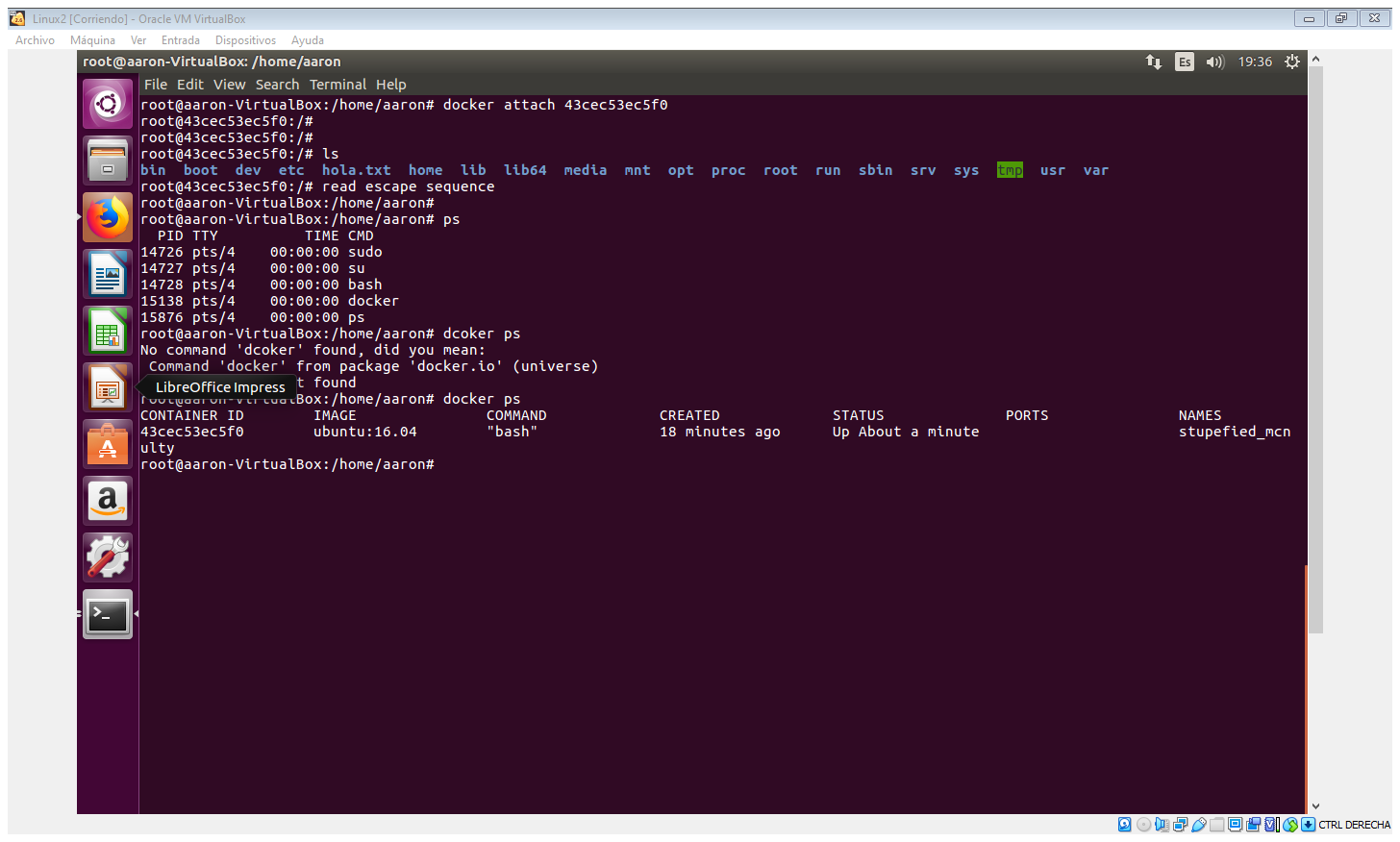


Ejecución de un contenedor con interacción

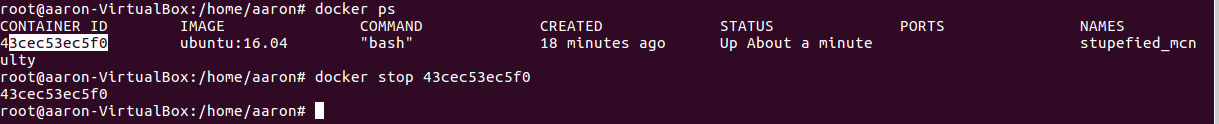
 estamos dentro del contenedor



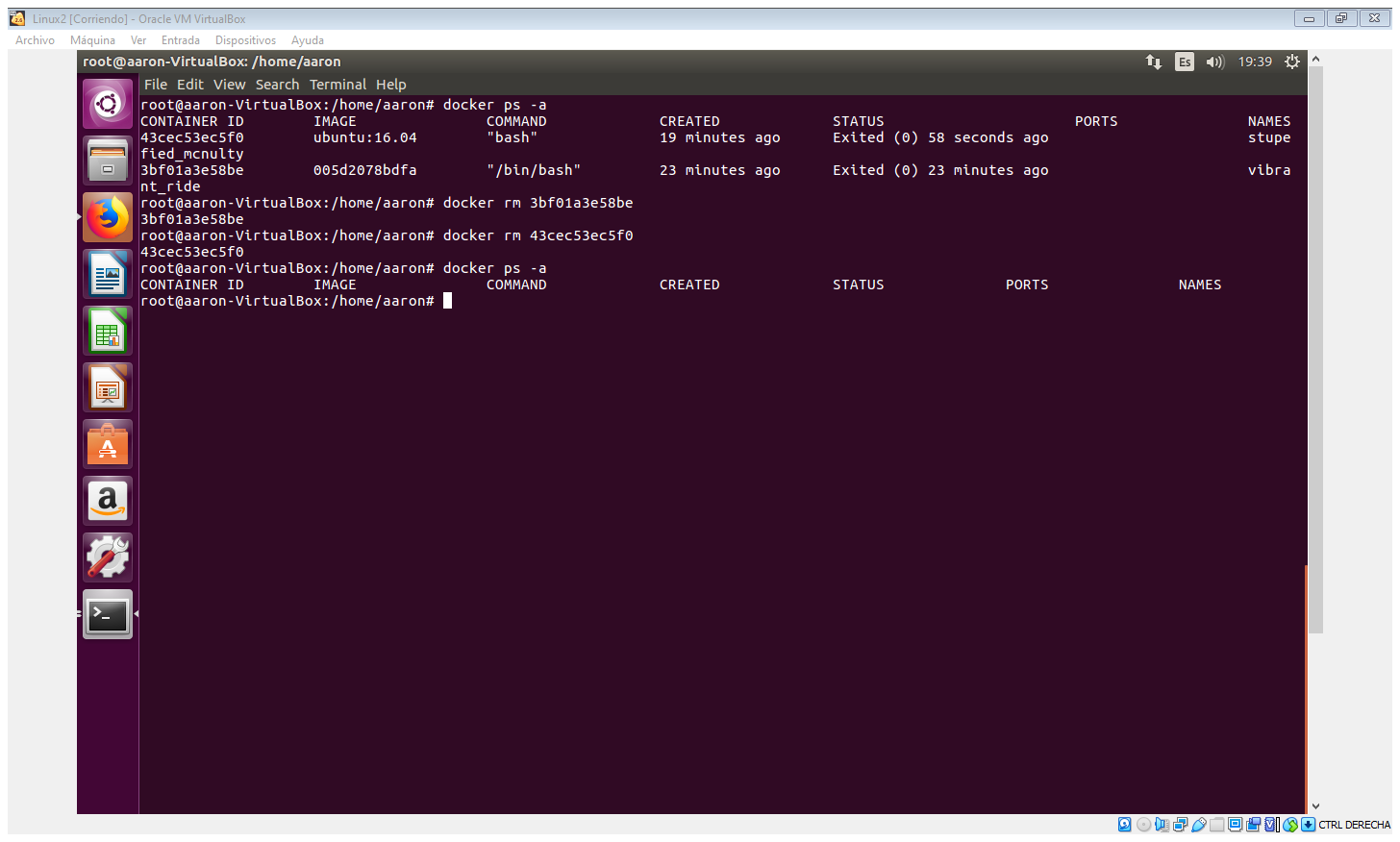
Nos salimos de contenedor, pero sin pararlo con Control + p + q



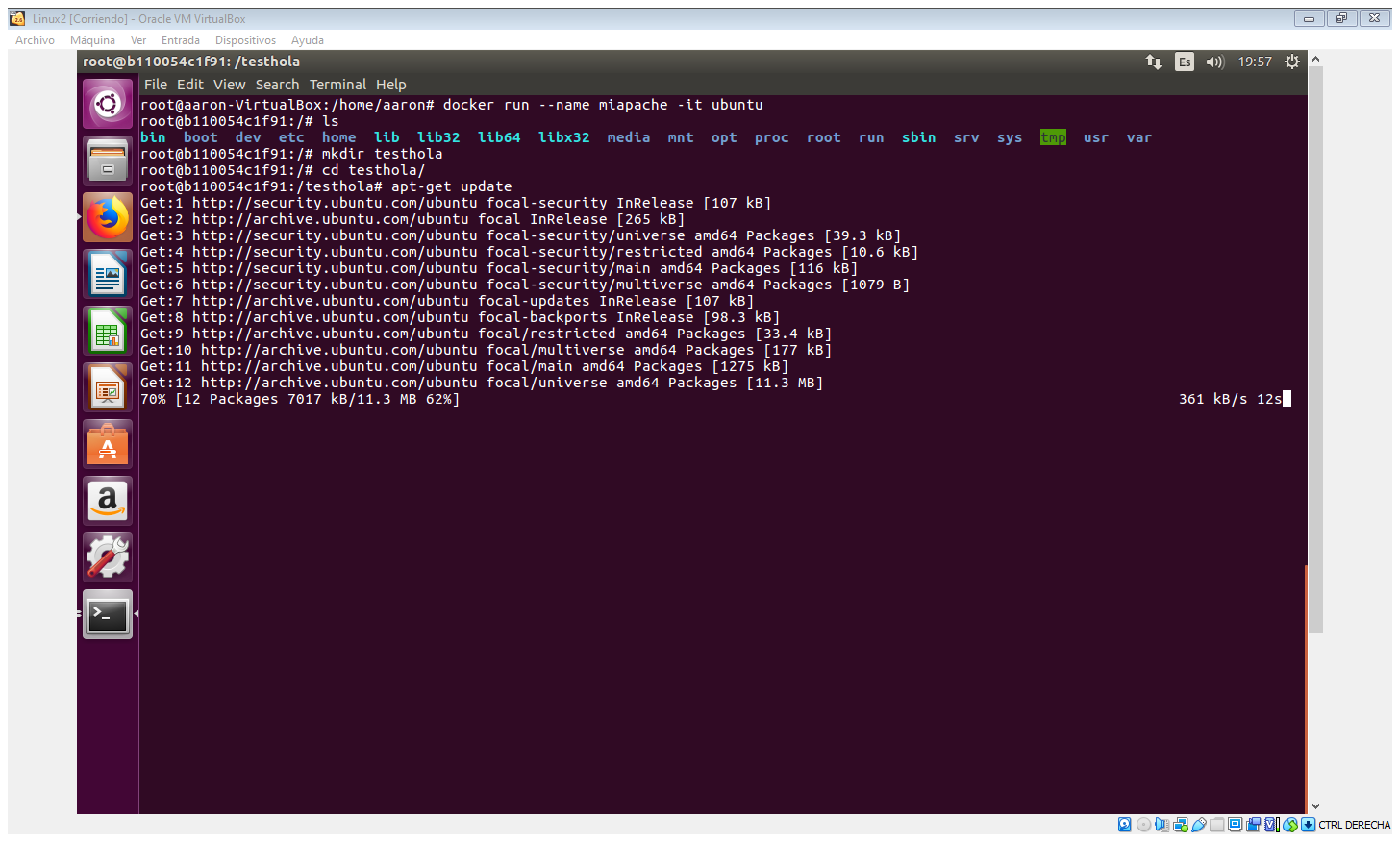
Detenemos el contenedor



Èliminamos con contenedores que hemos creado.

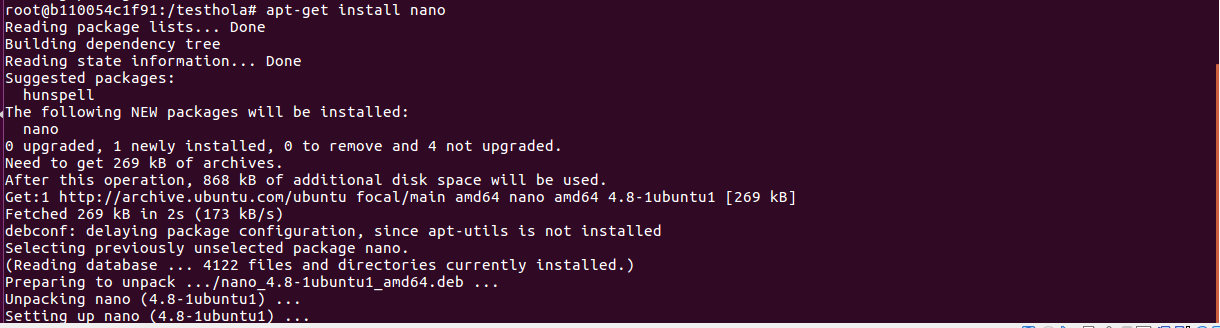


### Contenedor con apache desde cero con la imagen de Ubuntu.

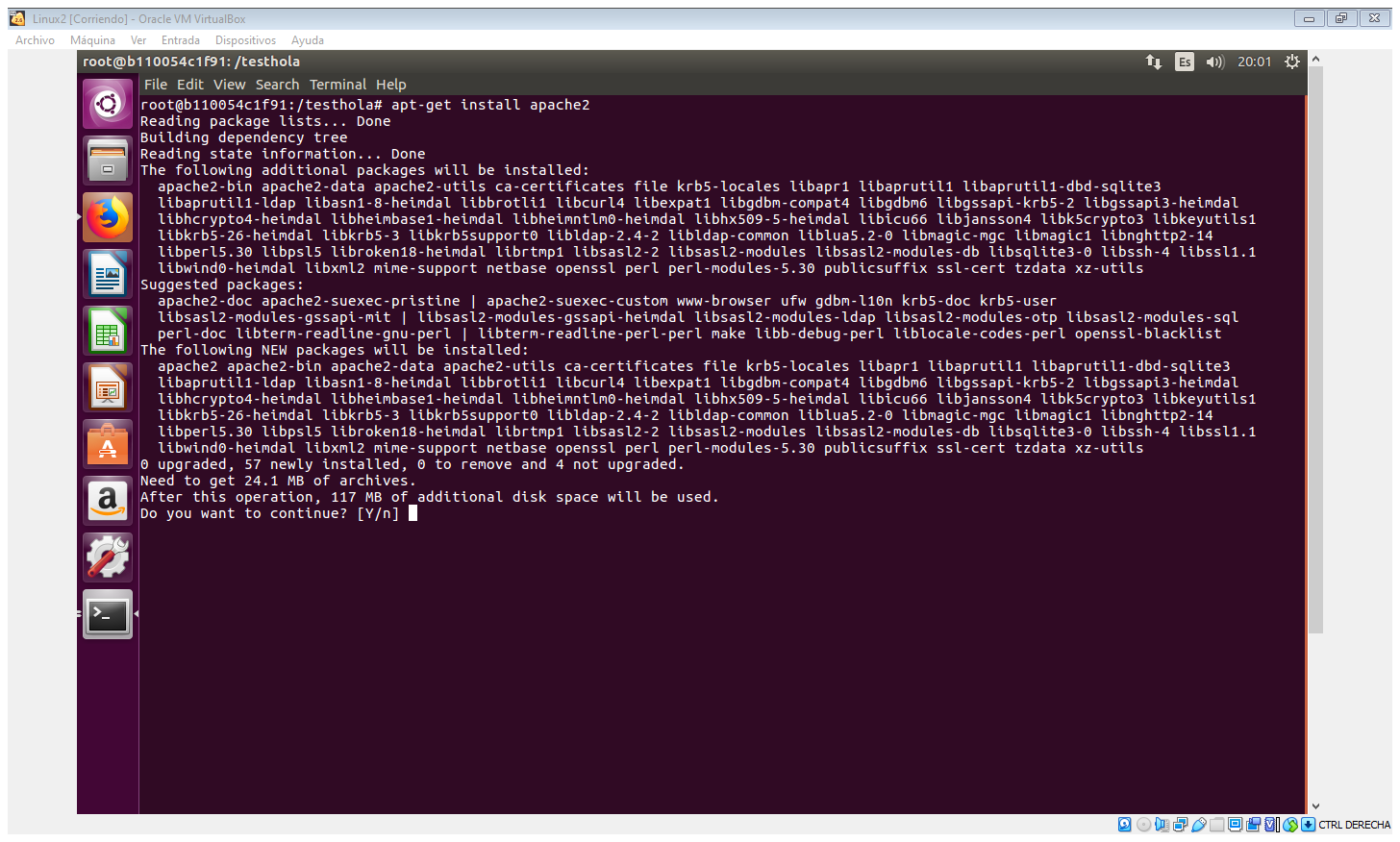


Actualización al contenedor

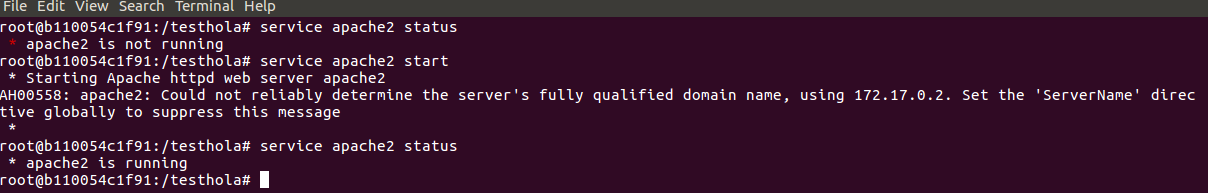
Instalación de nano



Instalación de apache



Una ves instalado el servicio de apache los iniciamos.

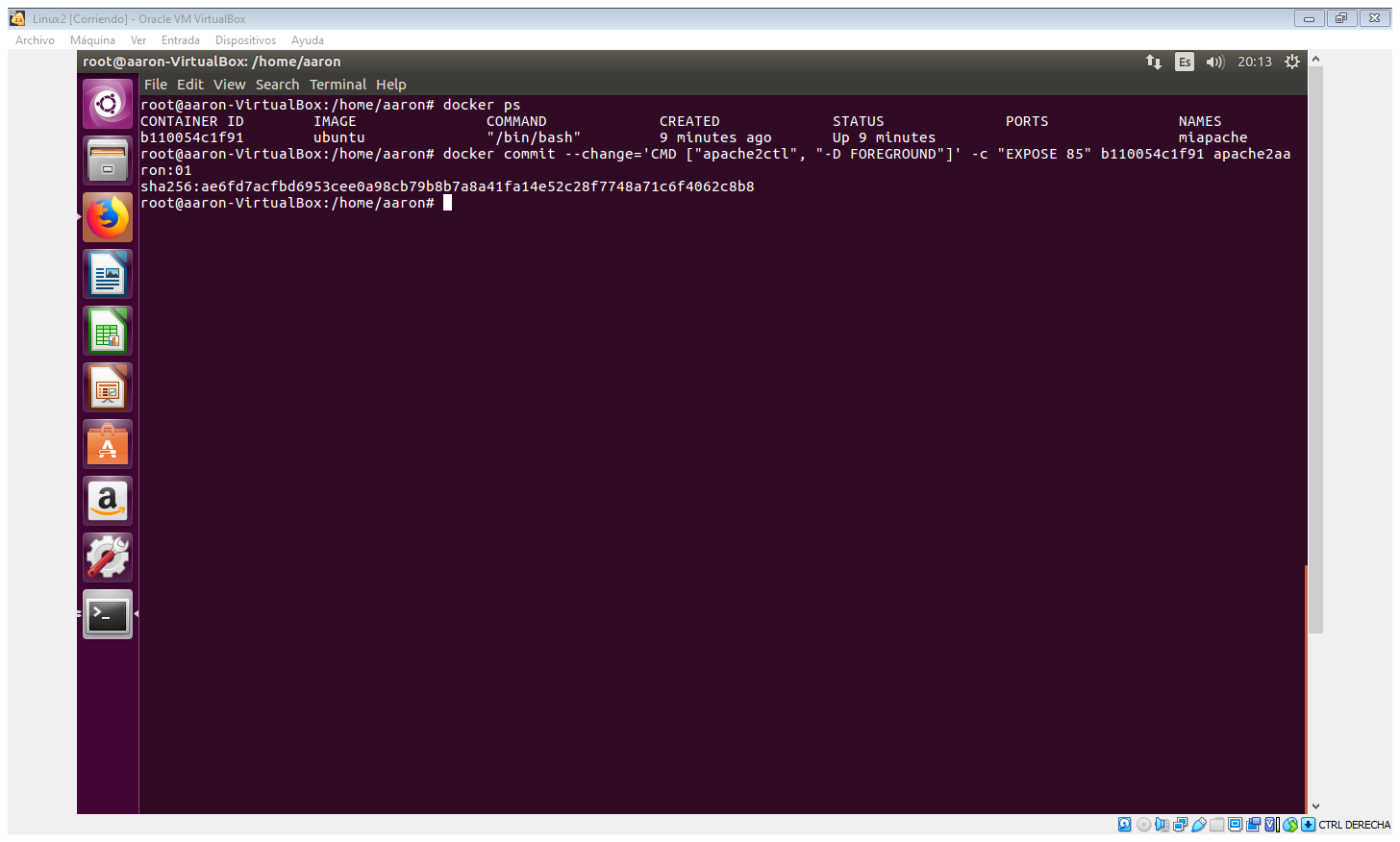


Cabe mencionar que todo se esta haciendo desde el contenedor

Se hace una imagen del contenedor creado

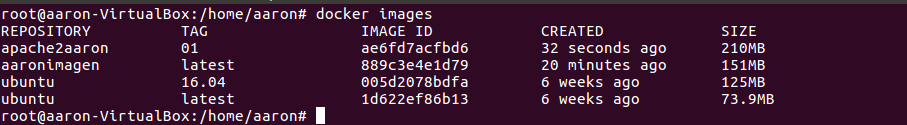
Comando:

docker commit –change=’CMD [“apache2ctl”, “-D FOREGROUND”] -c “EXPOSE 85”’ [contenedor base] [nombre]



explicación de los parámetros, cmd (comando), ejecuta el servicio apache2 ctl y se expone el puerto 85

Se tiene una nueva imagen



La de ***apache2aaron***

## DockerFile

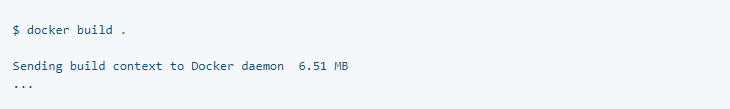
Docker puede construir imágenes automáticamente leyendo las instrucciones de a Dockerfile. A Dockerfilees un documento de texto que contiene todos los comandos que un usuario podría llamar en la línea de comandos para ensamblar una imagen. Los docker build usuarios pueden crear una compilación automatizada que ejecuta varias instrucciones de línea de comandos en sucesión.

Esta página describe los comandos que puede usar en a Dockerfile. Cuando haya terminado de leer esta página, consulte las [DockerfileMejores prácticas](https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices/) para obtener una guía orientada a consejos.

## Uso

El comando [docker build](https://docs.docker.com/engine/reference/commandline/build/) construye una imagen desde ayDockerfile un contexto . El contexto de la compilación es el conjunto de archivos en una ubicación específica PATHo URL. El PATHes un directorio en su sistema de archivos local. El URLes una ubicación de repositorio Git.

Un contexto se procesa de forma recursiva. Entonces, a PATHincluye cualquier subdirectorio y el URLincluye el repositorio y sus submódulos. Este ejemplo muestra un comando de compilación que usa el directorio actual como contexto:



La construcción es ejecutada por el demonio Docker, no por la CLI. Lo primero que hace un proceso de compilación es enviar todo el contexto (recursivamente) al demonio. En la mayoría de los casos, es mejor comenzar con un directorio vacío como contexto y mantener su Dockerfile en ese directorio. Agregue solo los archivos necesarios para construir el Dockerfile.



FROM ubuntu:14.04 //de donde va sacar la imagen

MAINTAINER AARONv2.1 [aaronvelasco1993@outlook.com](mailto:aaronvelasco1993@outlook.com) //quien lo creo

RUN apt-get update //ejecución de apache

RUN apt-get -y install apache2 //ejecucion

RUN apt-get -y install wget //ejecucion

RUN apt-get -y install unzip //ejecucion

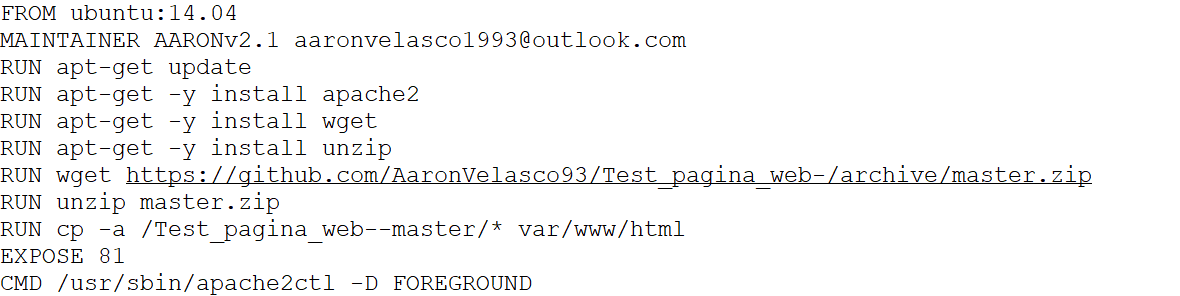
RUN wget <https://github.com/AaronVelasco93/Test_pagina_web-/archive/master.zip>

RUN unzip master.zip

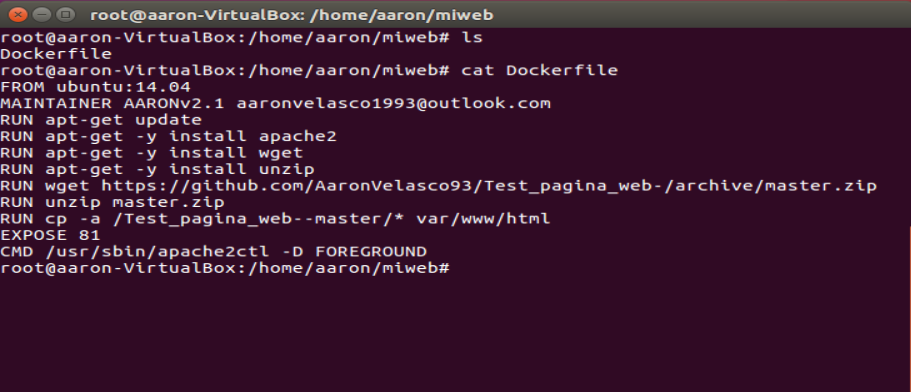
RUN cp -a /Test\_pagina\_web--master/\* var/www/html

EXPOSE 81 //puesto expuesto por defecto

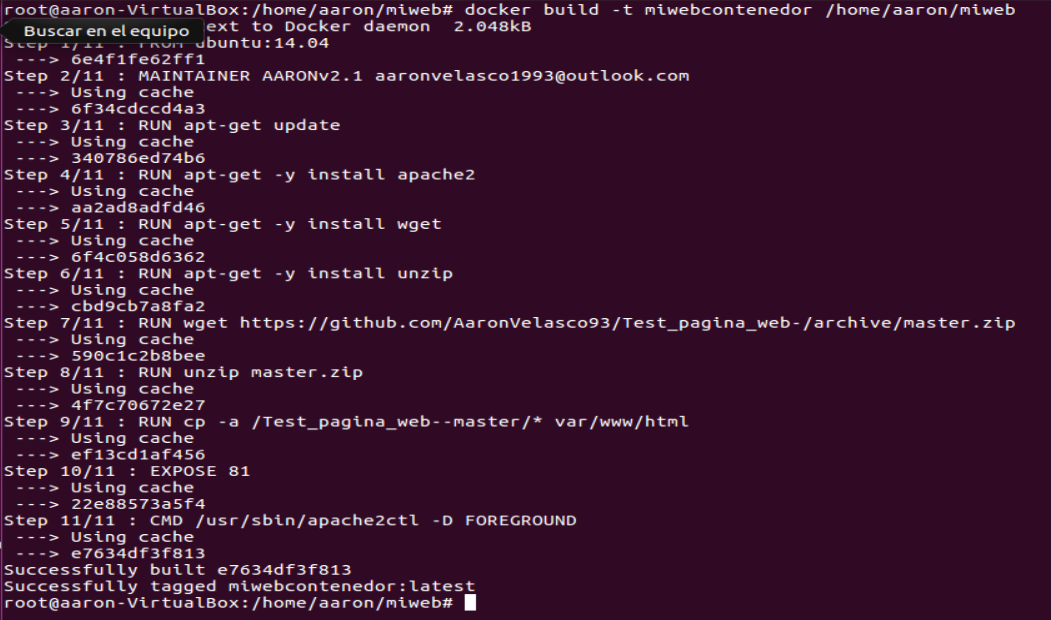
CMD /usr/sbin/apache2ctl -D FOREGROUND



Creación de mi dcokerfile



Creación de una nueva imagen a partir del dockerfile



Comando:

Docker build -t micontenedor /home/aaron/miweb

Donde:

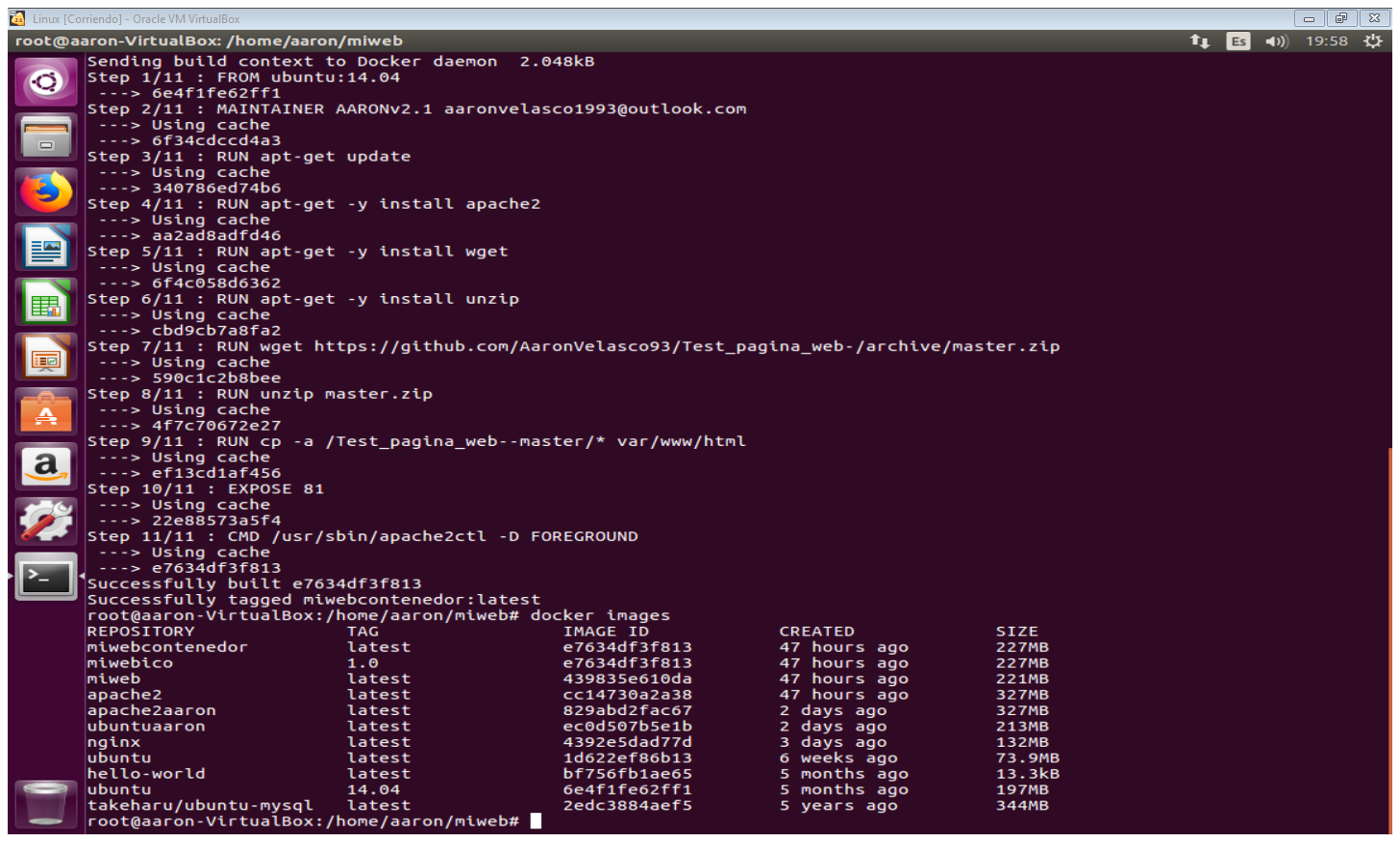
Build: va mostrar paso a paso lo que se va a realizar

-t

Miwebcontenedor : creación de un nuevo contenedor

/home/aaron/miweb : la ruta donde esta mi dockerfile

Se corrobora que tenemos un nueva imagen





Ejecutar la imagen



Comando para ejecutar

docker run -d -p 85:80 miwebcontenedor

escucharemos desde el puerto 85

