**ROOTLESS DOCKER**

STEP-1: CREATE UBUNTU USER

The environment is currently running Ubuntu 16.04 with the user logged in as root. The first step is to create a new user without these root privileges, meaning they will be running with increased security and not be able to make critical changes to the system.

The useraddcommand will create a user with the default permissions. Run the command in the terminal to add a new user called lowprivuser. This user can be called anything.

useradd -m -d /home/lowprivuser -p $(openssl passwd -1 password) lowprivuser

Using`sudo su, it's possible to switch to running as this new, low privileged user.

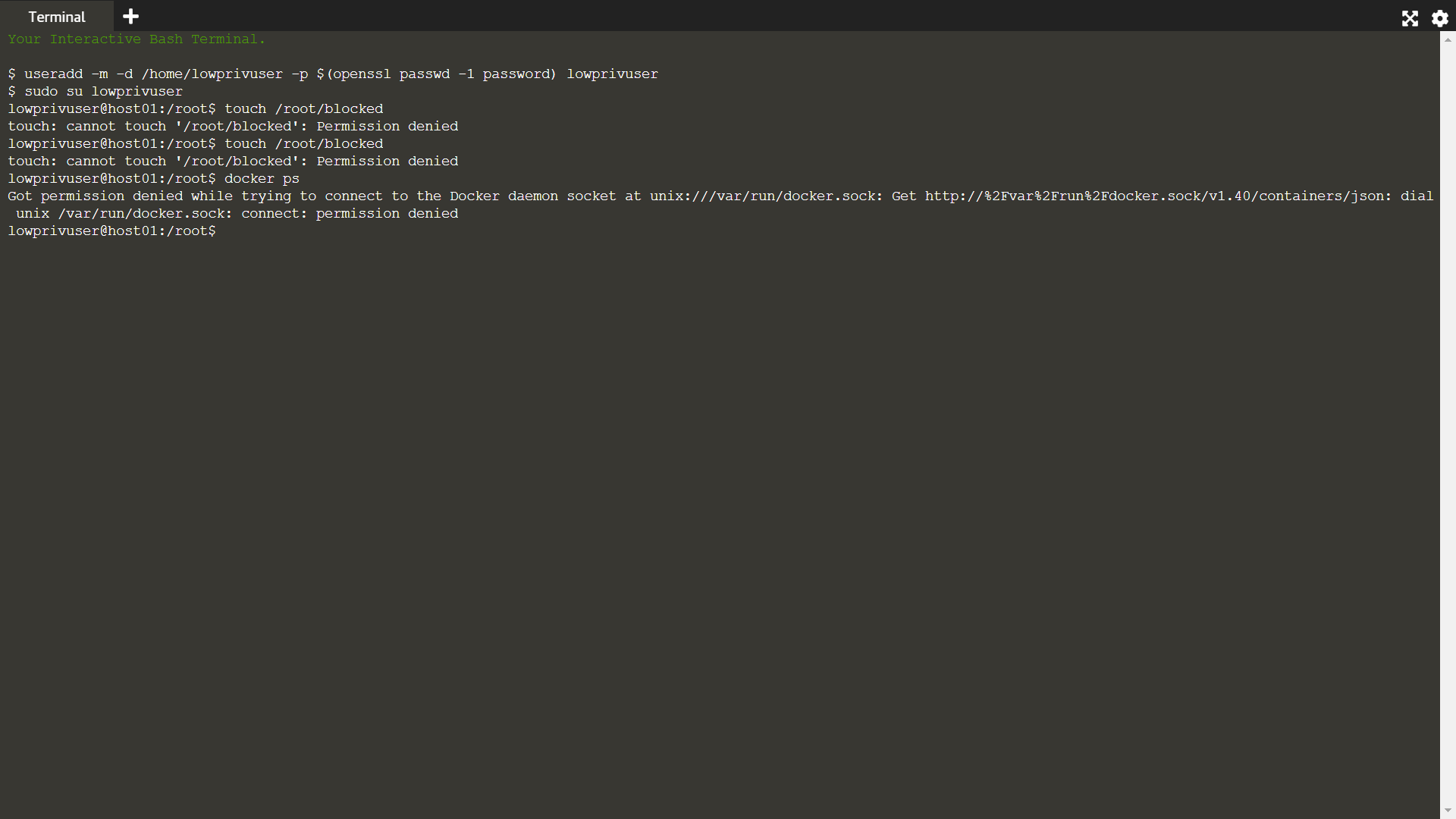
sudo su lowprivuser

When running as this user, a couple of items change. For example, the user is not able to create or change files in certain locations such as the root directory, touch /root/blocked.

The user is also not able to access Docker as previously this required them to have root permissions.

docker ps

In the next step, we'll deploy the new Rootless version and allow users launch their own containers.



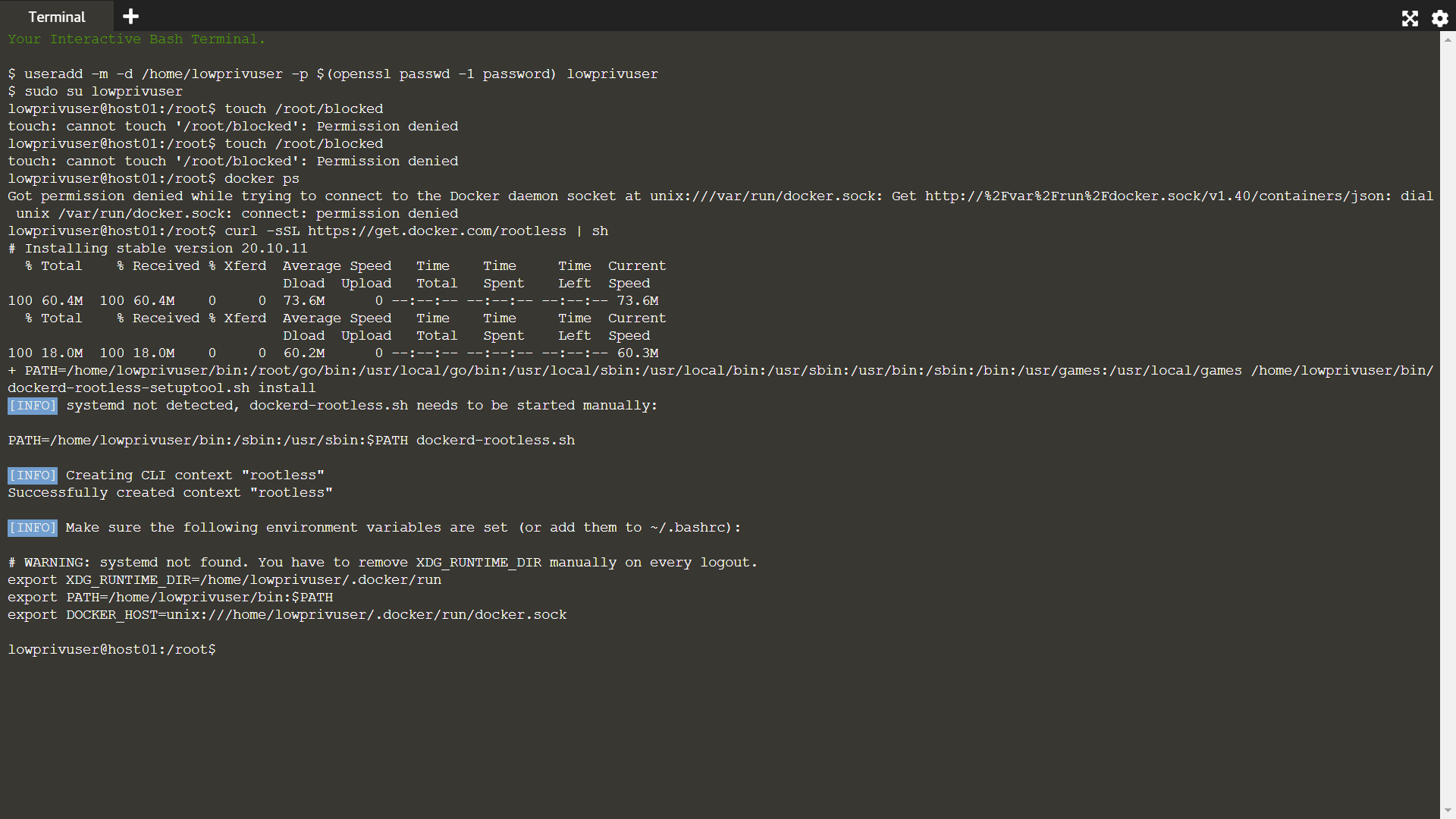
STEP-2: INSTALL ROOTLESS DOCKERS

Docker have made available a script which will deploy the required components for the new Rootless version.

Run the following command as lowprivuser to execute the script and install the components.

curl -sSL https://get.docker.com/rootless | sh

After this has finished, proceed to the next step to setup the user environment and start launching containers.



STEP-3:ACCESS DOCKER

Rootless Docker has now been installed. The daemon can be started using the following script:

export XDG\_RUNTIME\_DIR=/tmp/docker-1001

export PATH=/home/lowprivuser/bin:$PATH

export DOCKER\_HOST=unix:///tmp/docker-1001/docker.sock

mkdir -p $XDG\_RUNTIME\_DIR

/home/lowprivuser/bin/dockerd-rootless.sh --experimental --storage-driver vfs

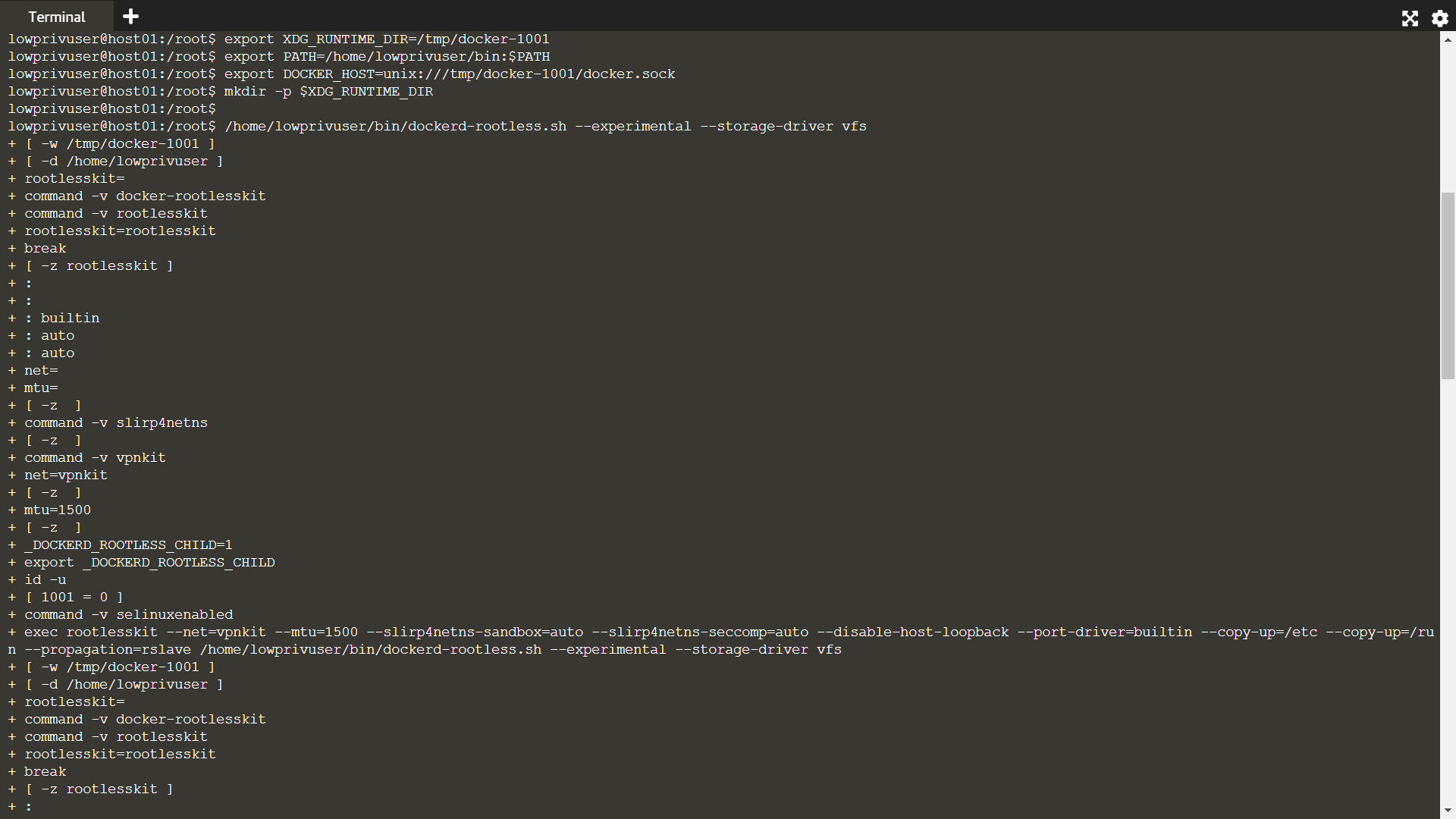
This will run in the foreground and allow you to see the debug output from the Rootless Docker Daemon.

Click the following command to launch a second terminal window and change the context to run as roootlessuser.

sudo su lowprivuser

: "Second Terminal running as lowprivuser

Id



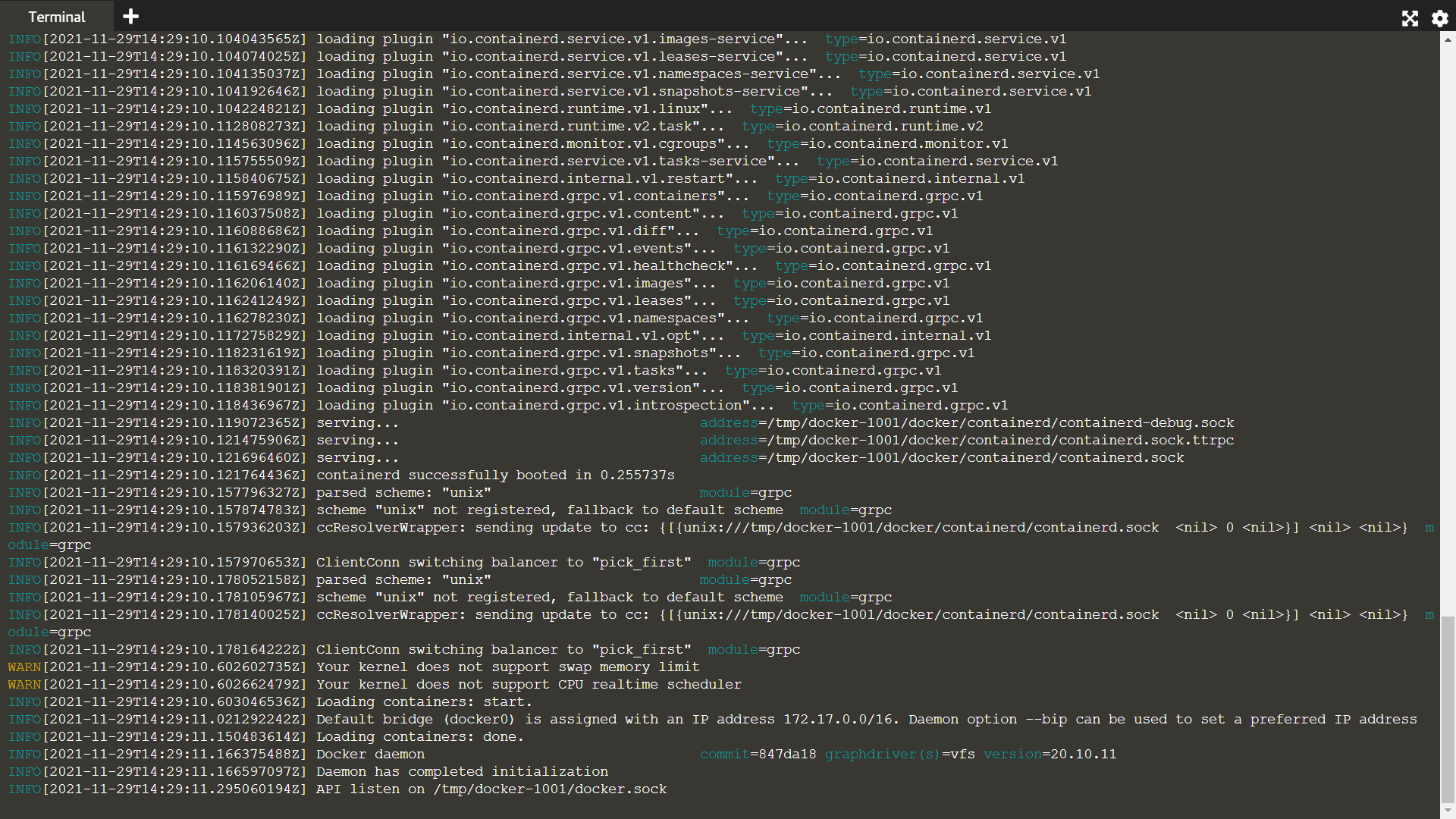
To access Docker, set the following environment variables. This specifies connecting to the Docker instance running for the user with id *1001*, which should match the id of *lowprivuser*.

export XDG\_RUNTIME\_DIR=/tmp/docker-1001

export PATH=/home/lowprivuser/bin:$PATH

export DOCKER\_HOST=unix:///tmp/docker-1001/docker.sock

In the next step we can start launching containers.



STEP-4: RUN CONTAINERS

It's now possible to access the Docker Daemon running for user 1001.

The standard Docker CLI commands work in the same way. The following command lists all the containers running for the user, currently it should return an empty list.

docker ps

It's possible to inspect details of the Daemon running:

docker info

Containers can be in the same way.

docker run -it ubuntu bash

Users within the container will still be reported as root. They will be able to install packages and modify parts of the system running inside of the Docker. However, if they managed to break out they wouldn't be able to interfer with the host.

id

In a separate terminal window, as root it's possible to explore which processes are running and which user started them. Using ps aux you can verify that our new container instance is managed and owned by our low privileged user.

id; ps aux | grep lowprivuser

The system is now running Docker Containers without requiring any additionals permissions, allowing our systems to operatoe with increased security.

