Lab - Installing NESSUS Using Docker

Overview

In this lab, you will install the docker program onto your Kali machine. Once that has been completed, you will move on to the second part of the lab and download and install the Docker container for NESSUS.

Using Docker, we will be able to install NESSUS and all its dependencies without having to call on or use any dependencies on our Kali machine.

About docker

Docker is a platform for developers and sysadmins to develop, deploy, and run applications with containers. The use of Linux containers to deploy applications is called containerization. Containers are not new, but their use for easily deploying applications is.

Containerization is increasingly popular because containers are:

- Flexible: Even the most complex applications can be containerized.
- Lightweight: Containers leverage and share the host kernel.
- Interchangeable: You can deploy updates and upgrades on-the-fly.
- Portable: You can build locally, deploy to the cloud, and run anywhere.
- Scalable: You can increase and automatically distribute container replicas.
- Stackable: You can stack services vertically and on-the-fly.
- Containers are portable

Images and containers

A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Container images become containers at runtime and in the case of Docker containers - images become containers when they run on Docker Engine.

Containers and virtual machines

A container runs natively on Linux and shares the kernel of the host machine with other containers. It runs a discrete process, taking no more memory than any other executable, making it lightweight.

By contrast, a virtual machine (VM) runs a full-blown "guest" operating system with virtual access to host resources through a hypervisor. In general, VMs provide an environment with more resources than most applications need.

Reference:

https://docs.docker.com/get-started/

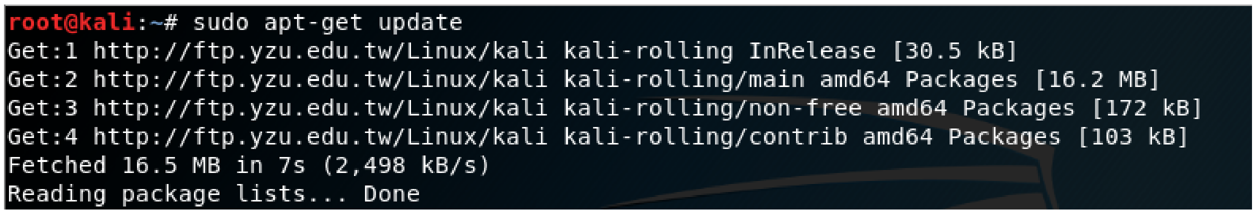
Requirements

- One virtual install of Kali Linux.
- Kali has been recently updated and upgraded with the latest packages.
- Internet connection

Begin the lab

Ensure Kali has been updated.

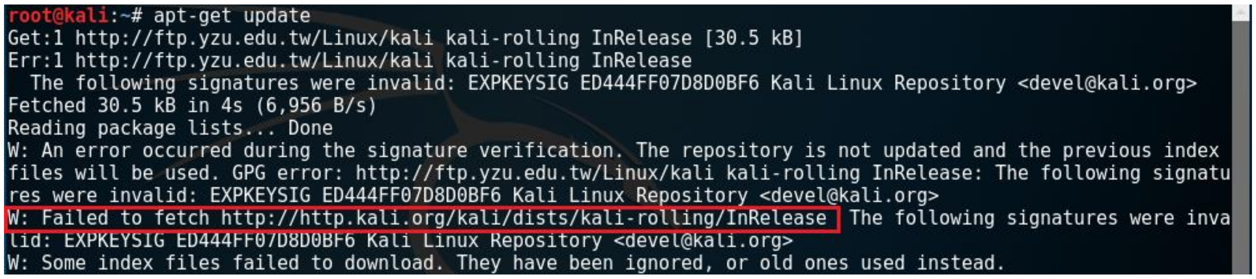
```
apt-get update
```



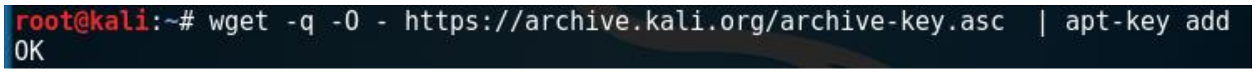
```
root@kali:~# sudo apt-get update
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease [30.5 kB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main amd64 Packages [16.2 MB]
Get:3 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/non-free amd64 Packages [172 kB]
Get:4 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/contrib amd64 Packages [103 kB]
Fetched 16.5 MB in 7s (2,498 kB/s)
Reading package lists... Done
```

If you get an error referencing an invalid key signature, you need to update your key signature using the following command:

```
wget -q -O - https://archive.kali.org/archive-key.asc  | apt-key add
```

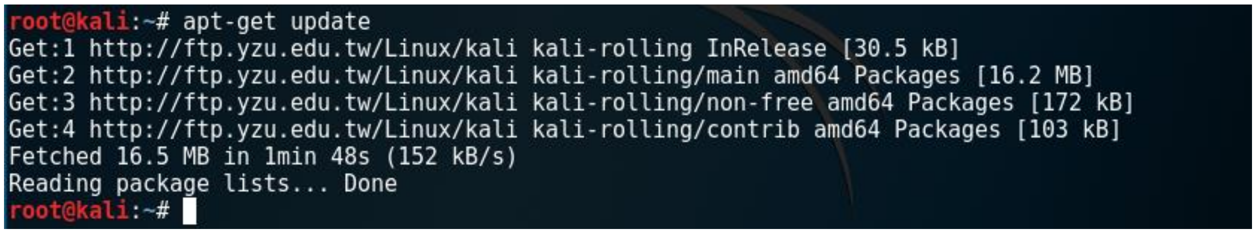


```
root@kali:~# apt-get update
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease [30.5 kB]
Err:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease
  The following signatures were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
Fetched 30.5 kB in 4s (6,956 B/s)
Reading package lists... Done
W: An error occurred during the signature verification. The repository is not updated and the previous index
files will be used. GPG error: http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease: The following signatu
res were invalid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
W: Failed to fetch http://http.kali.org/kali/dists/kali-rolling/InRelease  The following signatures were inva
lid: EXPKEYSIG ED444FF07D8D0BF6 Kali Linux Repository <devel@kali.org>
W: Some index files failed to download. They have been ignored, or old ones used instead.
```



```
root@kali:~# wget -q -O - https://archive.kali.org/archive-key.asc  | apt-key add
OK
```

Run the update command once again.



```
root@kali:~# apt-get update
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling InRelease [30.5 kB]
Get:2 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main amd64 Packages [16.2 MB]
Get:3 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/non-free amd64 Packages [172 kB]
Get:4 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/contrib amd64 Packages [103 kB]
Fetched 16.5 MB in 1min 48s (152 kB/s)
Reading package lists... Done
root@kali:~# 
```

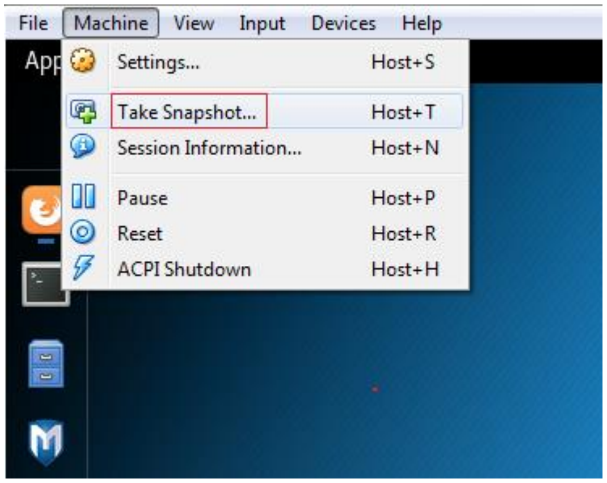
Once you're done with the apt-get update, continue with the apt-get upgrade command.

Creating a Snapshot of Your Current Kali Configuration

Before making any changes to your Kali install, you can take a snapshot of the current configuration so that if needed, you can rollback you, Kali, before the changes were made.

For VirtualBox, with your Kali running, from the VirtualBox taskbar, click on the machine and from the context menu, select, Take Snapshot.

To create a snapshot using VMWare, you will need the Workstation Pro version. Creating a snapshot using the VMWare Free Player is not an option.



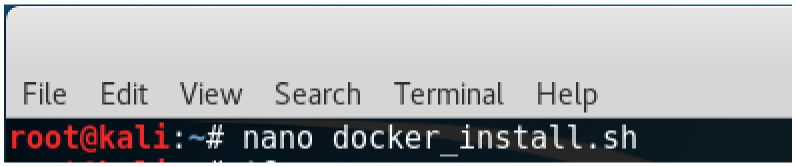
Installing the Docker Program

To install the Docker program, we create a script that will automate the entire process. To build the script, we can use any text editor Kali provides. For this demonstration, we will be using Nano, but you are free to use the text editor of your choice.

Building the Docker Installation Install Script

Once Kali has been updated, and your Snapshot has been completed, at the terminal, type the following:

```
nano docker_install.sh
```



```
root@kali:~# nano docker_install.sh
```

This opens a blank text file using the nano text editor.

The script we will be using is available at https://gist.github.com/apolloclark/f0e3974601346883c731

Thanks to appolloclark for creating and sharing this script!

apolloclark / **Kali 2016.1, Docker Install script**

Last active 6 days ago

Copy and paste the following text inside the box into the blank text editor.

Copy Only Text Inside the Box

```
#!/bin/bash

# update apt-get
export DEBIAN_FRONTEND="noninteractive"
sudo apt-get update

# remove previously installed Docker
sudo apt-get purge lxc-docker*
sudo apt-get purge docker.io*

# add Docker repo
sudo apt-get install -y apt-transport-https ca-certificates
sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-
keys 58118E89F3A912897C070ADBF76221572C52609D

cat > /etc/apt/sources.list.d/docker.list <<'EOF'
deb https://apt.dockerproject.org/repo debian-stretch main
EOF
sudo apt-get update

# install Docker
sudo apt-get install -y docker-engine
sudo service docker start
sudo docker run hello-world


# configure Docker user group permissions
sudo groupadd docker
sudo gpasswd -a ${USER} docker
sudo service docker restart

# set Docker to auto-launch on startup
sudo systemctl enable docker
```

Save the Script

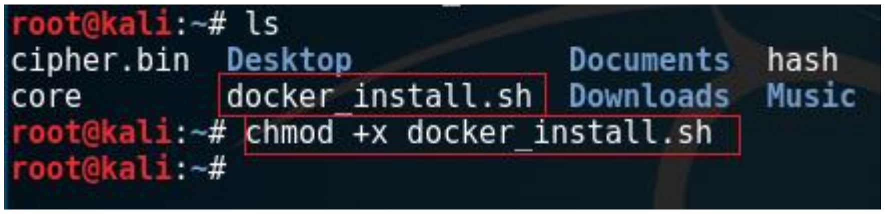
Save the file by pressing CTRL+x.

Type in ‘y’ to save the changes and then press enter to exit.

At the terminal, type **`ls`** to see the location your newly created script file.

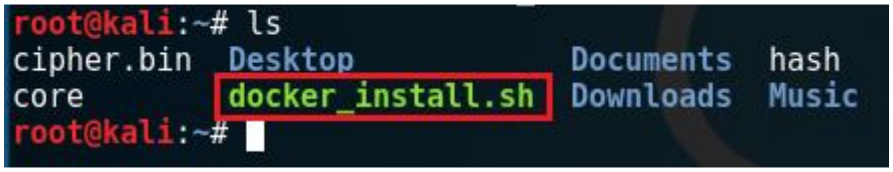
Type the following to make the script executable:

```
chmod +x docker_install.sh
```



```
root@kali:~# ls
cipher.bin  Desktop             Documents  hash
core        docker_install.sh   Downloads  Music
root@kali:~# chmod +x docker_install.sh
root@kali:~#
```

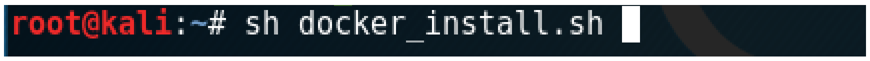
Type in `ls` and note the color of the file has changed to green annotating that the file is now an executable.



```
root@kali:~# ls
cipher.bin  Desktop             Documents  hash
core        docker_install.sh   Downloads  Music
root@kali:~#
```

To run the script, at the terminal type:

```
sh docker_install.sh
```



```
root@kali:~# sh docker_install.sh
```

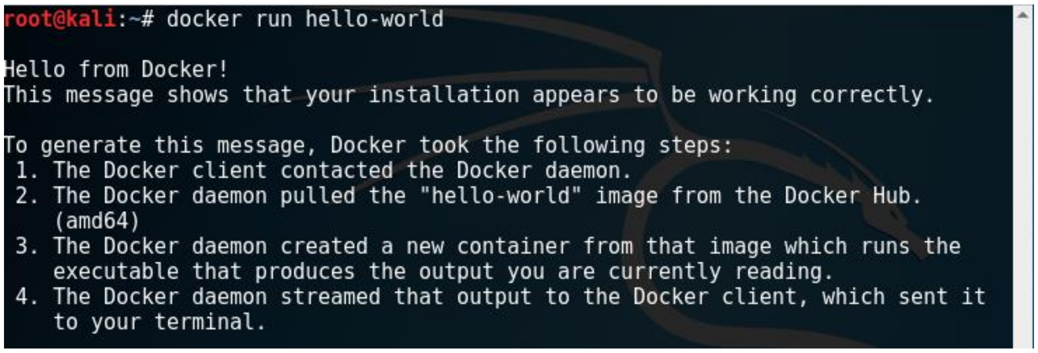
Hit enter

Allow the script to run and do not interrupt!

Check to see if Docker is properly installed

To check if the docker program is installed and working you can use the following command:

```
docker run hello-world
```

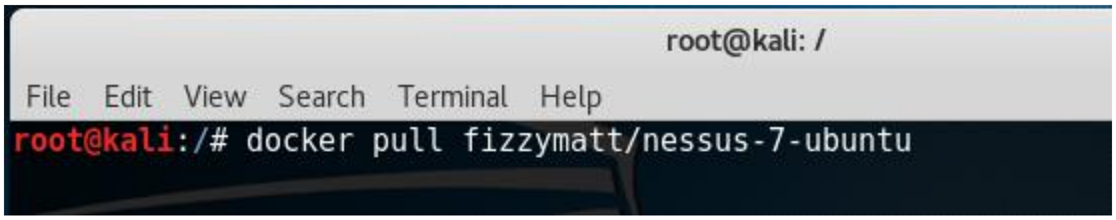


```
root@kali:~# docker run hello-world

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.
```

Downloading the Docker NESSUS image

We can pull down the NESSUS container from the Docker repository site by running the following command at the terminal:

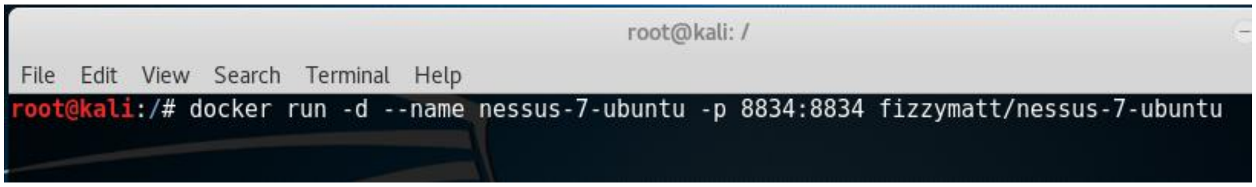
```
docker pull fizzymatt/nessus-7-ubuntu
```



```
root@kali:/# docker pull fizzymatt/nessus-7-ubuntu
```

Once the container for NESSUS has completed downloading, we can create a container to run our image of NESSUS using the following command:

```
docker run -d --name nessus-7-ubuntu -p 8834:8834 fizzymatt/nessus-7-ubuntu
```



```
root@kali:/# docker run -d --name nessus-7-ubuntu -p 8834:8834 fizzymatt/nessus-7-ubuntu
```

Breaking Down the Command

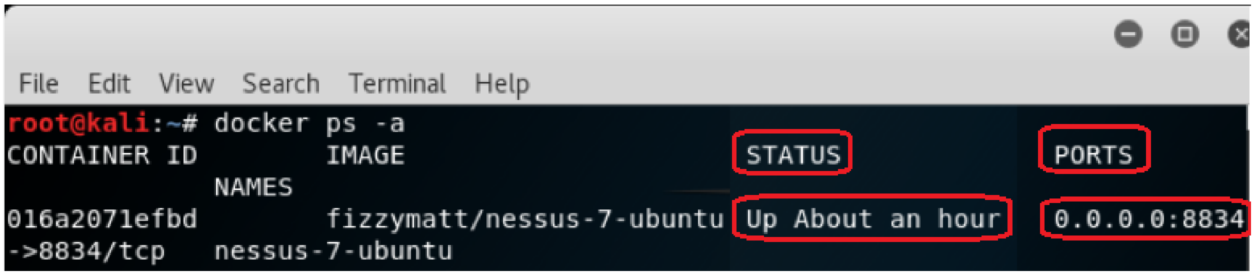
The "-d" (detached) argument tells Docker that we want to run the container without attaching a terminal.

The "--name" is just a handy way to reference your container.

The "-p" argument allows us to publish a specific port to the host. Here we've published port 8834 (the port that our containerized instance of Nessus listens on) to port 8834 on the host (which in my case is my install of Kali Linux.).

The final argument (fizzymatt/nessus-7-ubuntu) is a reference to the image that we're using to create the container.

It takes but a moment for the image to build itself. If we do a **`docker ps -a`** command, we can confirm the image is up and running, for how long and which port the image is configured to use.

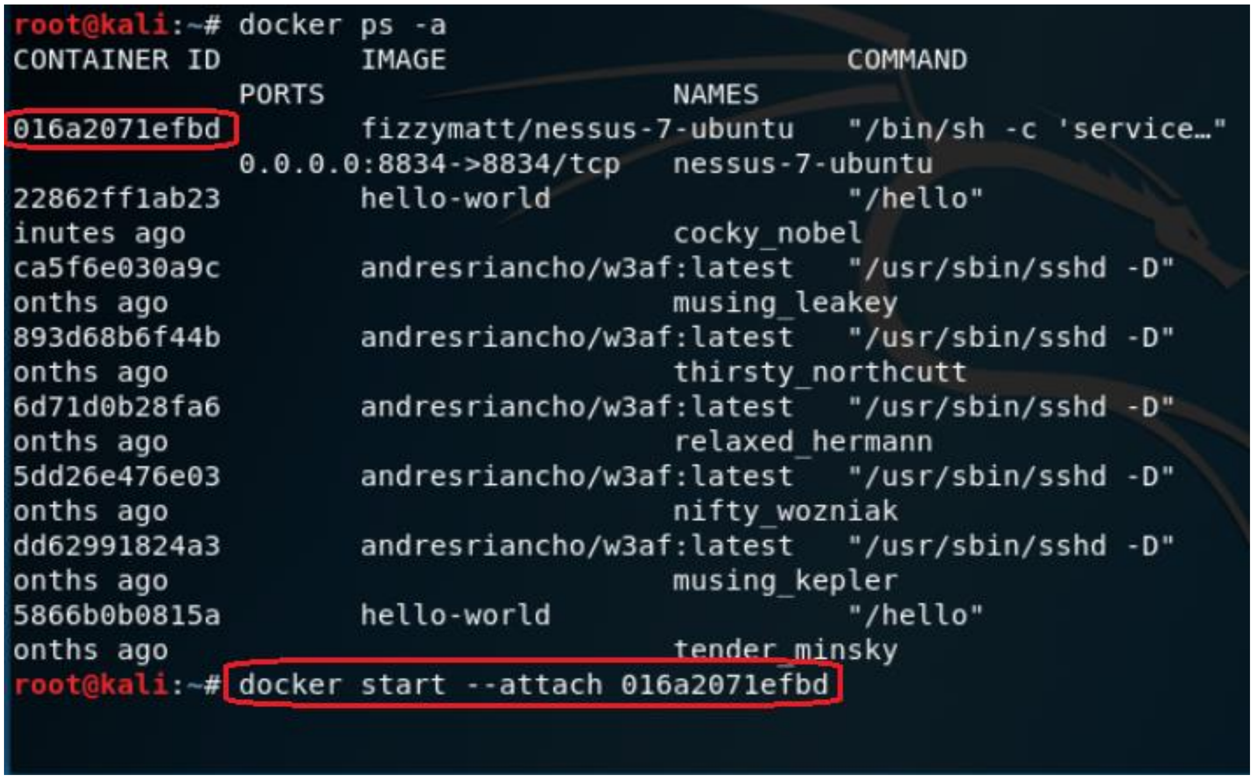


```
root@kali:~# docker ps -a
CONTAINER ID        IMAGE                     STATUS             PORTS
                    NAMES
016a2071efbd        fizzymatt/nessus-7-ubuntu Up About an hour   0.0.0.0:8834
->8834/tcp          nessus-7-ubuntu
```

To restart a container, we use the **`docker ps -a`** command to show all containers available inside of Docker. In the far-left column, we have the container ID's. Find the container ID for the image you would like to start and run, copy the long string of numbers and that the kali prompt, type the following command.

```
docker start --attach <container id>
```

In this example, I want to restart my previous NESSUS image.



```
root@kali:~# docker ps -a
CONTAINER ID        IMAGE                        COMMAND
             PORTS                     NAMES
016a2071efbd        fizzymatt/nessus-7-ubuntu    "/bin/sh -c 'service…"
             0.0.0.0:8834->8834/tcp    nessus-7-ubuntu
22862ff1ab23        hello-world                  "/hello"
inutes ago                             cocky_nobel
ca5f6e030a9c        andresriancho/w3af:latest    "/usr/sbin/sshd -D"
onths ago                              musing_leakey
893d68b6f44b        andresriancho/w3af:latest    "/usr/sbin/sshd -D"
onths ago                              thirsty_northcutt
6d71d0b28fa6        andresriancho/w3af:latest    "/usr/sbin/sshd -D"
onths ago                              relaxed_hermann
5dd26e476e03        andresriancho/w3af:latest    "/usr/sbin/sshd -D"
onths ago                              nifty_wozniak
dd62991824a3        andresriancho/w3af:latest    "/usr/sbin/sshd -D"
onths ago                              musing_kepler
5866b0b0815a        hello-world                  "/hello"
onths ago                              tender_minsky
root@kali:~# docker start --attach 016a2071efbd
```

You should now be able to launch Firefox and access the web interface for NESSUS using the URL of: `https://localhost:8834`

End of Lab!