SYBEROFFENSE

Lab - Installing NESSUS Using Docker

Overview

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

Using Docker, we will install NESSUS and all its dependencies without calling on or using any dependencies installed 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 quickly 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 it runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable software package 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:

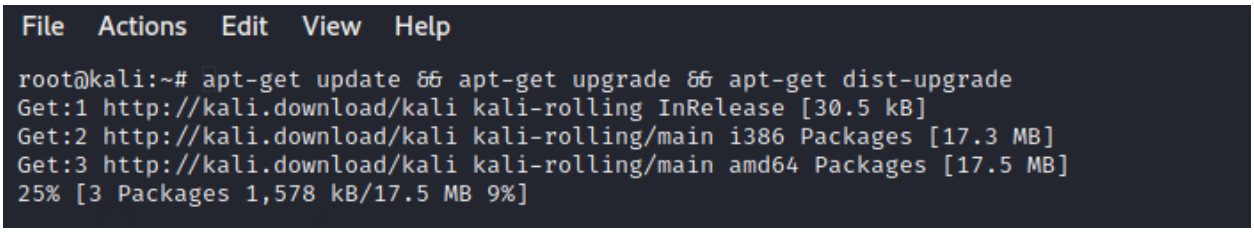
https://docs.docker.com/get-started/

Requirements

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

Begin the lab

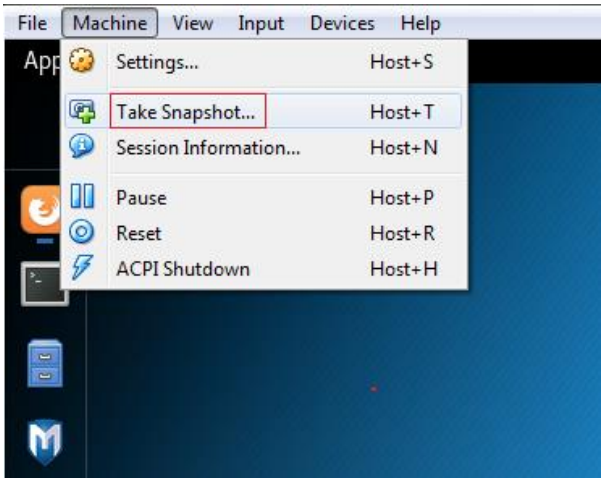
```
apt-get update && apt-get upgrade && apt-get dist-upgrade
```



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.

With your virtual install of Kali running, from the VirtualBox taskbar, click on the machine, and from the context menu, select, Take Snapshot.



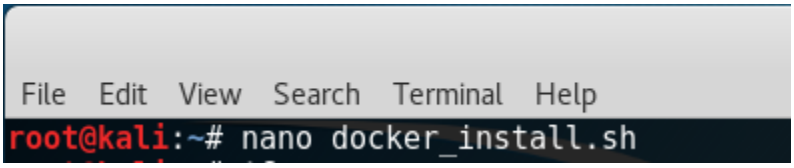
Installing the Docker Program

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. I will be using Nano for this demonstration, but you are free to use your choice of text editor.

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
```



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

The script we will be using is available at

https://gist.github.com/decidedlygray/1288c0265457e5f2426d4c3b768dfcef

Thanks to decidelygray for creating and sharing this script!

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

Copy Only Text Inside the Box

```
#!/bin/bash

#
# Kali Docker Setup Script
# @decidedlygray 20180902
# LICENSE: MIT
#
# Steps taken from: https://docs.docker.com/install/linux/docker-
ce/debian/
# And: https://medium.com/@airman604/installing-docker-in-kali-linux-
2017-1-fbaa4d1447fe
# Install uses the repository, so we can get updates in the future

# Remove any existing docker packages and update package list
sudo apt remove docker docker-engine docker.io -y
sudo apt update

# Install apt HTTPS packages
sudo apt install apt-transport-https ca-certificates curl gnupg2
software-properties-common -y

# Add Docker GPG key
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key
add -
# Verify key exists
# sudo apt-key fingerprint 0EBFCD88

# Add stable repo
# sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/debian jessie stable"
echo 'deb https://download.docker.com/linux/debian stretch stable' >
/etc/apt/sources.list.d/docker.list

# Update repo list
sudo apt update

# Finally, install Docker Community Edition
sudo apt install docker-ce -y
# Verify install by running the docker hello world
systemctl start docker
sudo Docker run hello-world
```

Save the Script

Save the file by pressing CTRL+x.

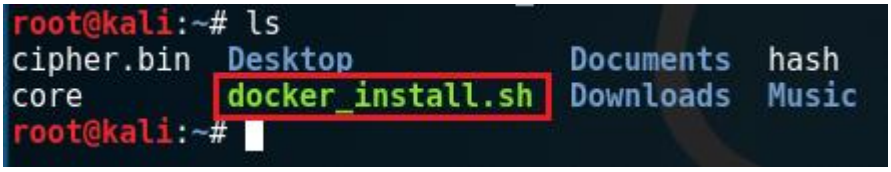
Type in 'y' to save the changes and then press enter to exit.

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

Type the following to make the script executable:

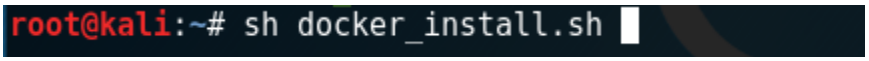
```
chmod +x docker_install.sh
```

Type in `ls` to find the script you just created.



To run the script, at the terminal type:

```
sh docker_install.sh
```



Hit enter

Allow the script to run, and do not interrupt!

If after the script completes and you receive the following error when Docker attempts to run docker run hello-world command, run the two following commands at the terminal prompt, one at a time.

```
sudo mkdir /sys/fs/cgroup/systemd
sudo mount -t cgroup -o none,name=systemd cgroup /sys/fs/cgroup/systemd
```

Run the script one more time and this time you should see the following return from the Docker server.

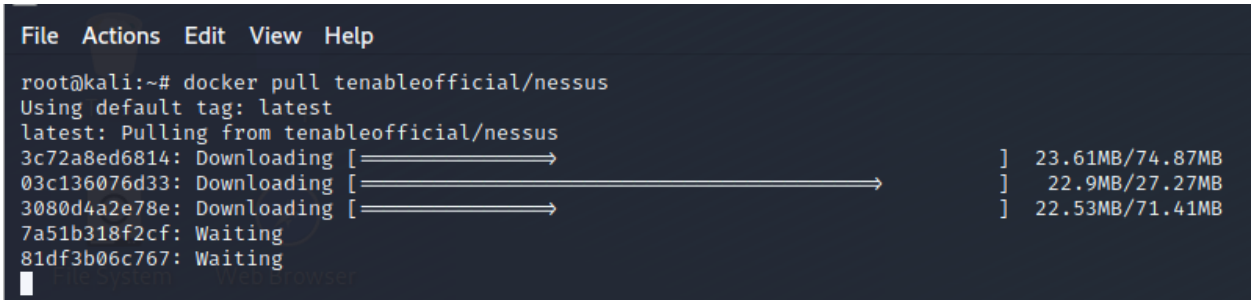
```
Hello from Docker!
This message shows that your installation appears to be working correctly.

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

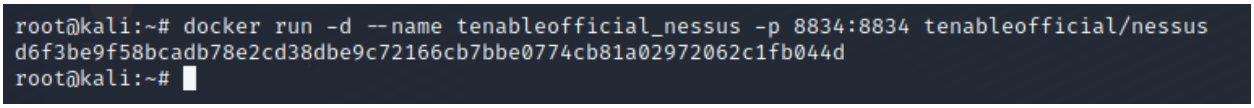
We can pull down the NESSUS container from the Docker repository site by running the following command at the terminal:

```
docker pull tenableofficial/nessus
```



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 tenableofficial_nessus -p 8834:8834
tenableofficial/nessus
```



Breaking Down the Command

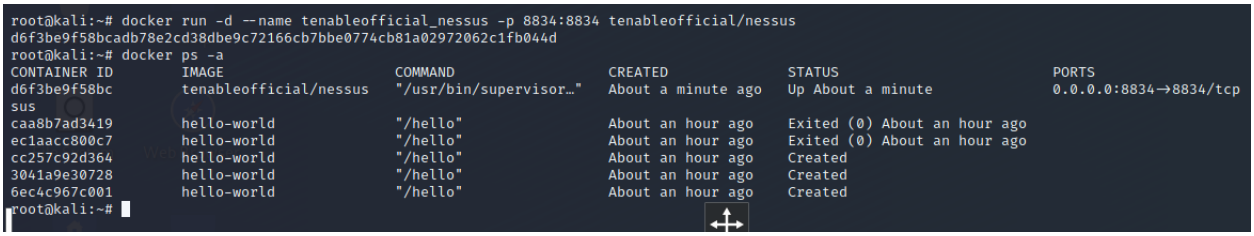
The "-d" (detached) argument tells Docker that we want to run the container without attaching a terminal.

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 (tenableofficial/nessus) is a reference to the image that we're using to create the container.

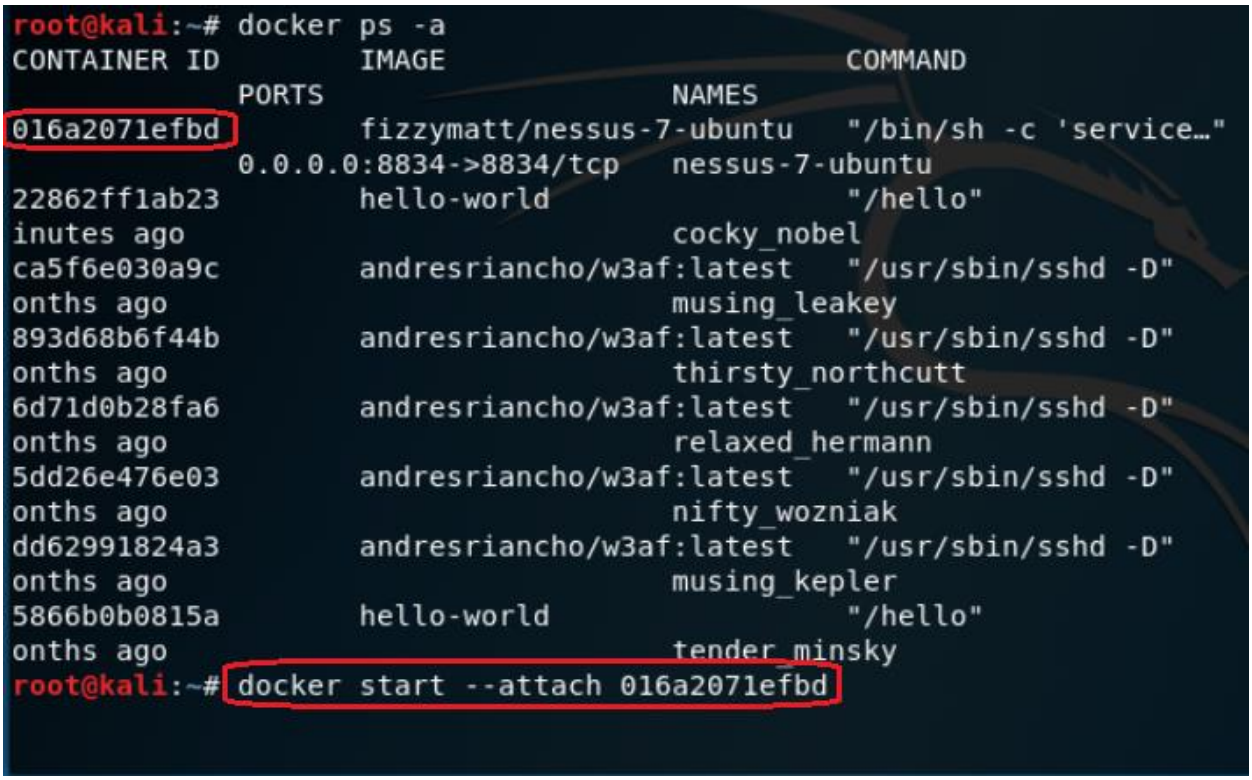
It takes but a moment for the image to build itself. If we do a **`docker ps -a`** command, we can confirm the image is up and running, for how long, and which port the image is configured to use.



To restart a container, we use the **`docker ps -a`** command to show all containers available inside 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.



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

End of the lab!