Docker In Linux

Tested in Ubuntu.

Installation

This is a summarization of the full article found here.

I usually add a repo to the package manager for stuff like this, but I decided to use a standalone .deb file this time.

Since Linux Mint 19 is built on Ubuntu 18.04 (Bionic Beaver), I downloaded the latest .deb file from here.

After installation, the Docker daemon starts automatically.

If you don't want to require 'sudo' with every docker command, add your userid to the 'docker' group.

Verify the Docker installation with the following command:

```
sudo docker run hello-world
```

If the Docker installation is valid, the hello-world image will be downloaded from Docker Hub, and will run. It will print out an informational message similar to this:

```
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.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/
```

```
For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

...and then exit.

Manage Containers

You can display a list of all your installed images and their associated containers with this command:

```
sudo docker ps -a
```

The results list will look something like this:

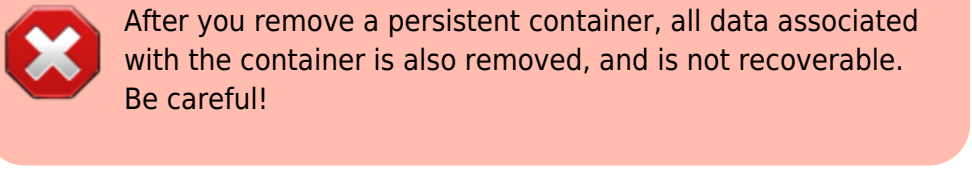
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3467bc449fe9	hello-world	"/hello"	24 hours ago	Exited (0) 24 hours ago		friendly_haibt
890e2533976a	hello-world	"/hello"	25 hours ago	Exited (0) 25 hours ago		angry_agnesi

If you'd like to remove a container, use the **docker rm** command. For example, if I wanted to remove the **friendly_haibt** container for the **hello-world** image, I'd do this:

```
sudo docker rm friendly_haibt
```

Then, when I run the docker ps command again, I'll see this:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
890e2533976a	hello-world	"/hello"	25 hours ago	Exited (0) 25 hours ago		angry_agnesi



Dump log file for a container:

```
sudo docker log <container-name>
```

Stop a running container:

```
sudo docker stop <container-name>
```

Restart a stopped container:

```
sudo docker restart <container-name>
```

Example: BusyBox container

A *very* simple example using BusyBox UNIX tools:

```
sudo docker run -it --rm busybox
```

This command drops you into a sh shell in a BusyBox environment, and the container is automatically removed when you exit.

Helper Script For Container Examples

dockit.py

```
    #!/usr/bin/python3

    import os
    import sys

    class CDockerMgr:
        dockerCmd = 'sudo docker'

        def __init__(self):
            pass

        def ExecuteBaseCommand(self, currentCommand):
            if currentCommand == 'status':
                self.ExecDocker("ps -a")
            else:
                self.ShowHelp()

        def ExecuteContainerCommand(self, currentContainer,
currentCommand):
            if currentContainer == 'sql1':
                if currentCommand == 'start':
                    self.ExecDocker("restart
{0}".format(currentContainer))
                elif currentCommand == 'stop':
                    self.ExecDocker("stop
{0}".format(currentContainer))
                elif currentCommand == 'bash':
                    self.ExecDocker("exec -it {0}
'bash'".format(currentContainer))
                else:
                    self.ShowHelp()
            elif currentContainer == 'busybox':
                if currentCommand == 'run':
                    self.ExecDocker("run -it --rm
```

```
{0}".format(currentContainer))
                else:
                    self.ShowHelp()
            else:
                self.ShowHelp()

        def ExecDocker(self, args):
            fullCmd = "{0} {1}".format(self.dockerCmd, args)
            os.system(fullCmd)
            exit(0)

        def ShowHelp(self):
            print("USAGE:")
            print("\tdockit <container> <command>")
            print("")
            print("Valid containers and commands:")
            print("\tstatus")
            print("\tsql1")
            print("\t\tstart")
            print("\t\tstop")
            print("\t\tbash")
            print("\tbusybox")
            print("\t\trun")

            exit(0)

    myDockerMgr = CDockerMgr()

    if len(sys.argv) == 2:
        myDockerMgr.ExecuteBaseCommand(sys.argv[1])
    elif len(sys.argv) == 3:
        myDockerMgr.ExecuteContainerCommand(sys.argv[1], sys.argv[2])
    else:
        myDockerMgr.ShowHelp()
```

docker