Docker Workflows

I will attempt to expand more on each of these as I learn the different phases of dockerization.

A quick-start guide to Docker can be found here: https://docs.docker.com/linux/step_one/

NOTE: CoreOS installations come with Docker pre-installed. You can skip that step if you already have Docker installed and scroll down to where they start running commands.

Read up on our Docker workflows here:

- Installing Docker
- Using Docker Images
- Building Custom Docker Images

Amazing Docker Cheat Sheet

https://github.com/wsargent/docker-cheat-sheet

Tips

CoreOS Docker Quirks

Default Docker IP Address

By default, Docker starts the daemon on a 172.17.X.X IP address. This can cause issues if 172.17.X.X is routable on the network from which you are trying to connect.

For example, the IllinoisNet WiFi uses a 172.X.X.X network, so the default IP address is routable on IllinoisNet. This results in traffic designated for the docker daemon to be sent over the network (since the network thinks it knows where you are trying to send it. **10.0.1.1/16** is unroutable on that network, so setting the docker daemon's IP to **10.0.1.1/16** allows our machine to properly send traffic to the docker daemon, instead of it being pushed out over the network where docker cannot see it.

To change Docker's default IP address, execute the following command, substituting for the CIDR notation of the new IP. It is imperative that you set this to a non-routable IP according to your source network's configuration.

```
sudo echo "[Service]

ExecStart=

ExecStart=/usr/lib/coreos/dockerd daemon --host=fd:// --bip=NEW_CIDR" > /etc/systemd/system/docker.service

sudo systemctl stop docker

sudo systemctl daemon-reload

sudo ip link set docker0 down

sudo brctl delbr docker0

sudo systemctl start docker
```

To verify:

ip addr should now list your newly specified ip instead of 172.17.x.x. A reboot may also be required in order for this change to take effect.

Command Completion: Unsupported

Bash completion is not available for CoreOS, making it difficult or impossible to install docker command completion, as described here: https://docs.docker.com/compose/completion/

Installing Docker Compose

Since the /usr/local/bin/ folder is read-only, and sudo -i did not seem to help, I was able to install docker-compose by using curl to place the executable in my home folder (or any writable folder). Then, simply add docker-compose to the PATH environment variable and give it permission to execute:

```
mkdir /home/core/docker-compose

curl -L https://github.com/docker/compose/releases/download/1.5.2/docker-compose-`uname -s`-`uname -m` > /home/core/docker-compose
```

```
export PATH=$PATH:/home/core/docker-compose

chmod +x /home/core/docker-compose/docker-compose
```

Debugging

General

- docker diff <name or id> - show what files differ from the source image (what changes you have made since running)

Building a Container

- docker stats <name or id> - shows memory usage / limits
- docker run <id> - can use the id of incremental build images to boot into the failing state

Running a Container

- docker events <id> - shows what docker daemon is doing in the background
- docker inspect <name or id> - inspect a particular container's configuration
- docker logs <id> - shows the logs of a particular container
- nsenter - enter a particular namespace

Inspecting a Running Container

- docker exec -it <name or id> /bin/bash