DSC Capstone Sequence

Lecture 05
Environments & Containerization

Lecture Outline

- Intro to Containerization
- Working with Docker

Universal Problems

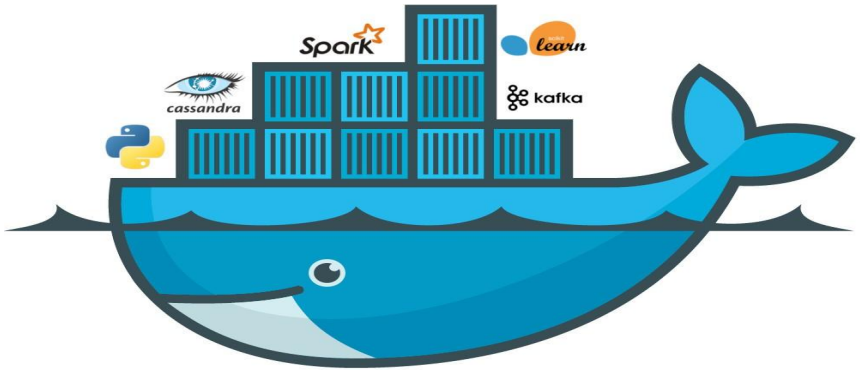
- "I don't know why it's not working on your computer... it worked on mine..."
- My code needs to run on both Windows and Unix, do I need to completely rewrite it?
- I need more RAM. I would use AWS, but my code is setup to run on my computer?
- I tried running your code, but I got "ImportError: No module named Numpy"

Containers to the Rescue

- Capture the state of (the necessary) software on a computer/server that your project runs.
- As containers can be version controlled, your data science project is replicable from the environment on up.
- Containers are lightweight blueprints or snapshots of a software environment.

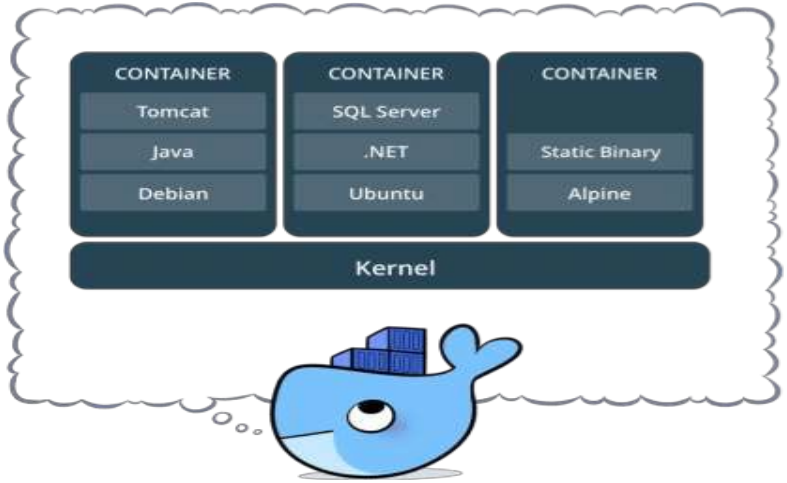
Docker

- What is Docker?



Docker Containers

- Standardized packaging of software and dependencies
- Isolate apps from each other.
- Shares the same OS kernel.

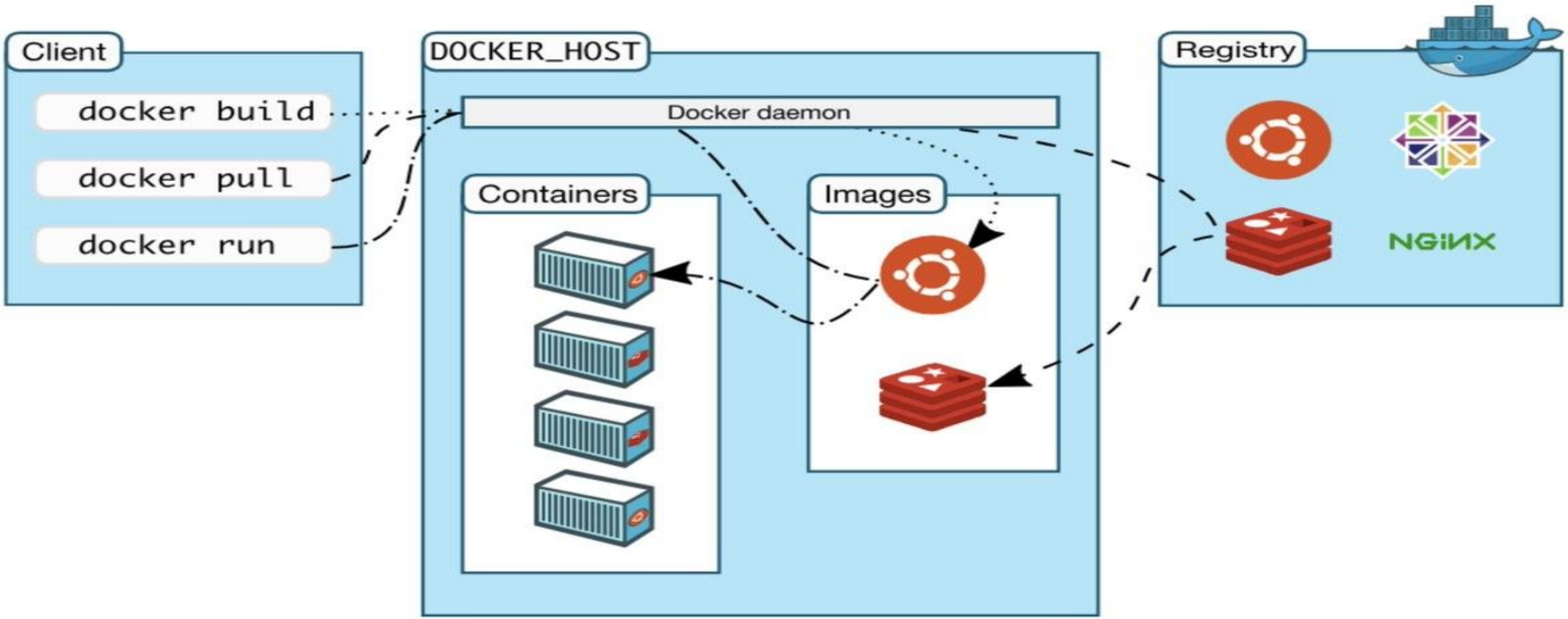


Terminology

- **Containers**: very small user-level virtualization that helps you build, install, and run your code.
- **Images**: a snapshot of your container.

- **Dockerfile**: a yaml-based file that's used to build the image; this is what is version controlled.
- **Dockerhub**: GitHub for your Docker images. You can automatically build images from Dockerfiles and pull images from Dockerhub to servers.

The Docker Landscape



Physical vs Virtual

- Working on a personal computer: you reserve all the resources, even when not in use

- DSMLP Servers: Create a container on login (claim memory); release upon logout.

Serverless Architectures

- Speed: Docker has no OS to boot; starts in seconds.
- Portability: few dependencies between processes
- Efficiency: Less OS overhead => more efficient use of infrastructure resources.

- Containerization powers serverless architectures like AWS Lambda.

Docker on DSMLP

- Launch scripts on the DSMLP server start containers with predefined docker images.
- Dockerfiles contain data-science environments: anaconda, scipy, r-studio, tensorflow, pytorch
- Supply your own dockerfile using the '-i' flag:
    - `launch-180.sh -i dockeruser/dockerimage`

```
/*---------------------
        Available remote ieng6-XXX hosts to connect:

        * ieng6-240, ieng6-241, ieng6-242, ieng6-243, ieng6-244, ieng6-245
        * ieng6-246, ieng6-247, ieng6-248, ieng6-249, ieng6-640, ieng6-700
        * ieng6-701, ieng6-702

*ECE* students, please check with your TA or instructor on which hosts
you can run your projects.

-------------------------*/
[ds180awi20ta1@dsmlp-login]:~:375$ launch-
launch-141.sh                launch-cuda9.sh              launch-py3torch-gpu-cuda9.sh   launch-rsm-jupyter.sh          launch-tf-gpu.sh
launch-148-gpu.sh~           launch-datascience.sh        launch-py3torch-gpu.sh         launch-rsm-rstudio.sh          launch-tf.sh
launch-148.sh~               launch-dev.sh                launch-py3torch.sh             launch-rstudio.sh              launch-torch7.sh
launch-allennlp.sh           launch-py3gym-gpu.sh         launch-pytorch-bg.sh           launch-scipy-ml-gpu-cuda10.sh
launch-caffe-gpu.sh          launch-py3gym.sh             launch-pytorch-gpu.sh          launch-scipy-ml-gpu.sh
launch-caffe.sh              launch-py3torch-cuda9.sh     launch-pytorch.sh              launch-scipy-ml.sh
[ds180awi20ta1@dsmlp-login]:~:375$ launch-
```

Creating your own Docker Images

- Build custom images from predefined images
- Create docker images from Dockerfiles using this tutorial:
    - https://github.com/ucsd-ets/datahub-example-notebook
    - Dockerfile is a 'blueprint' to build the Environment
- Create Docker images interactively using this tutorial:
    - https://docs.google.com/document/d/1LPfqHvk2Itm_ckafrxRVxXQdr5BSozjsv_TURQDj9x8/edit
    - Enter the image and interactively install (less reproducible)

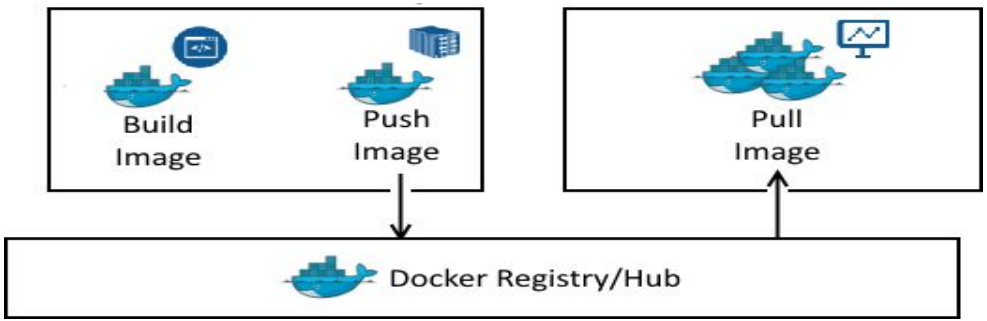
You will create and maintain your own project Dockerfile.

Creating Images Interactively

- Creating a docker image interactively
    - Pull a base image: docker pull ucsdets/scipy-ml-notebook
- Build and run the image interactively:
    - docker run -it ucsdets/scipy-ml-notebook /bin/bash
- Install software on the image as if your own computer.
    - Great for debugging Dockerfiles (later)
    - Push the Dockerfile to DockerHub to use elsewhere.

Interactive Image Creation

- Pros: easy to create an image and use it!
  - As easy as on your own computer...
- Cons: can't version control; can't easily tweak images.



Creation from Dockerfiles

- Better to create a blueprint for creating the image from a **Dockerfile**.
- A Dockerfile can be as simple as a few commands installing python libraries (requiring external dependencies).

Dockerfile

Base Image

Install software

```
FROM ucsdets/scipy-ml-notebook

USER root

RUN conda install --quiet --yes geopandas
```

Example Dockerfile

Base Image

Install Java w/pkg mngr

Set Variables (urls)

Download software and "Install"

Dockerfile

```
FROM ucsdets/scipy-ml-notebook

USER root

RUN apt-get update && \
    apt-get upgrade -y && \
    apt-get install -y default-jre && \
    apt-get install -y default-jdk


ENV APK_SCRIPT https://raw.githubusercontent.com/iBotPeaches/Apktool/master,
ENV APK_JAR https://bitbucket.org/iBotPeaches/apktool/downloads/apktool_2.4

RUN mkdir -p /usr/local/bin

RUN P=/tmp/$(basename $APK_SCRIPT) && \
    wget -q -O $P $APK_SCRIPT && \
    chmod +x $P && \
    mv $P /usr/local/bin

RUN P=/tmp/$(basename $APK_JAR) && \
    wget -q -O $P $APK_JAR && \
    chmod +x $P && \
    mv $P /usr/local/bin/apktool.jar
```

A Dockerfile from scratch (datascience)

```
# reference: https://hub.docker.com/_/ubuntu/
FROM ubuntu:16.04

# Adds metadata to the image as a key value pair example LABEL
version="1.0"
LABEL maintainer="Your Name <some_email@domain.com>"

# Set environment variables
ENV LANG=C.UTF-8 LC_ALL=C.UTF-8

# Create empty directory to attach volume
RUN mkdir ~/GitProjects

# Install Ubuntu packages
RUN apt-get update && apt-get install -y \
    wget \
    bzip2 \
    ca-certificates \
    build-essential \
    curl \
    git-core \
    htop \
    pkg-config \
    unzip \
    unrar \
    tree \
    freetds-dev
```

```
# Clean up
RUN apt-get clean && rm -rf /var/lib/apt/lists/*

# Install Jupyter config
RUN mkdir ~/.ssh && touch ~/.ssh/known_hosts
RUN ssh-keygen -F github.com || ssh-keyscan github.com >>
~/.ssh/known_hosts
RUN git clone https://github.com/bobbywlindsey/dotfiles.git
RUN mkdir ~/.jupyter
RUN cp /dotfiles/jupyter_configs/jupyter_notebook_config.py
~/.jupyter/
RUN rm -rf /dotfiles

# Install Anaconda
RUN echo 'export PATH=/opt/conda/bin:$PATH' >
/etc/profile.d/conda.sh
RUN wget --quiet https://repo.anaconda.com/archive/Anaconda3-5.2.0-
Linux-x86_64.sh -O ~/anaconda.sh
RUN /bin/bash ~/anaconda.sh -b -p /opt/conda
RUN rm ~/anaconda.sh

# Set path to conda
ENV PATH /opt/conda/bin:$PATH

# Update Anaconda
RUN conda update conda && conda update anaconda && conda update --
all

# Install Jupyter theme
RUN pip install msgpack jupyterthemes
RUN jt -t grade3
```

```
# Configure access to Jupyter
WORKDIR /root/GitProjects
EXPOSE 8888
CMD jupyter lab --no-browser --ip=0.0.0.0 --allow-root --
NotebookApp.token='data-science'
```

References

- Writing Dockerfile Tutorial (and links therein)
- Launching images on DSMLP
- Docker Commands (building/running images)
- Docker Syntax Reference