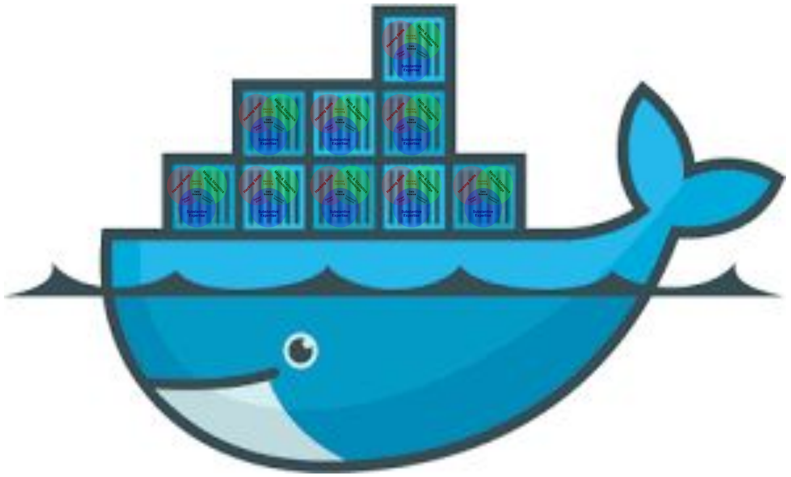
Docker for Data Science

http://bit.ly/d4ds-tutorial

Aly Sivji
@CaiusSivjus

Joe Jasinski
@JoeJJasinski

Tathagata Dasgupta
@Tathagata



#d4ds

ChiPy @ PYCON CLEVELAND 2018

About Us

Aly Sivji @CaiusSivjus

Mathematician / Software Engineer @ divvyDOSE

Medical Informatics Grad Student @ Northwestern

Joe Jasinski @JoeJJasinski

Senior Software Engineer @ Nielsen

MS in Computer Science

Tathagata Dasgupta (aka T) @Tathagata

Senior Software Engineer @ Here Technologies

MS in Information Systems

Agenda

- Rules of the Road
- Data Science Overview
- Introduction to Docker
    - Hands-on Labs
- Data Science Workflows using Docker Containers
    - Hands-on Lab
- Break (3:00 pm)
- Docker Compose Overview
- PyCon Talk Recommender Application
    - Hands-on Labs

Rules of the Road

- Format: Lecture + Lab to reinforce concepts
- Main Github Repo: http://bit.ly/d4ds-tutorial
  - Setup instructions
  - Link to Slides: http://bit.ly/d4ds-slides
- Asking for Help
  - Raise your hand during lab sessions
  - Question session at the end of (most) labs
- Beyond the scope
  - Specific questions about how to fit Docker into YOUR workflow
    - Let’s discuss offline!

Docker for Data Science

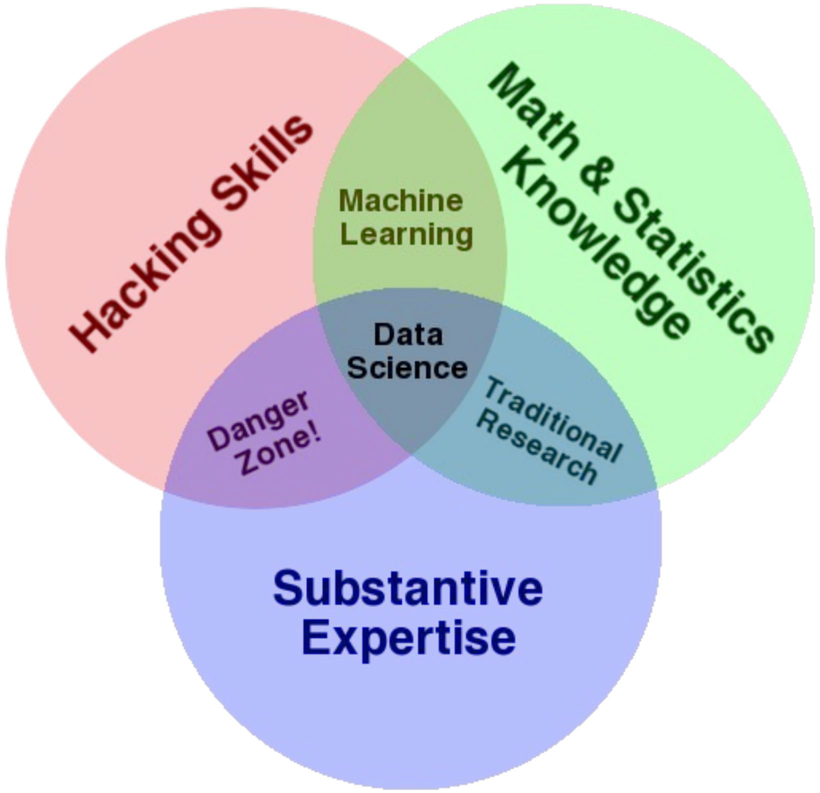
build passing

Materials for "Docker for Data Science" tutorial presented at PyCon 2018 in Cleveland, OH.

Slides

Data Science

What is Data Science?



Source: Drew Conway

Data Science Use Cases

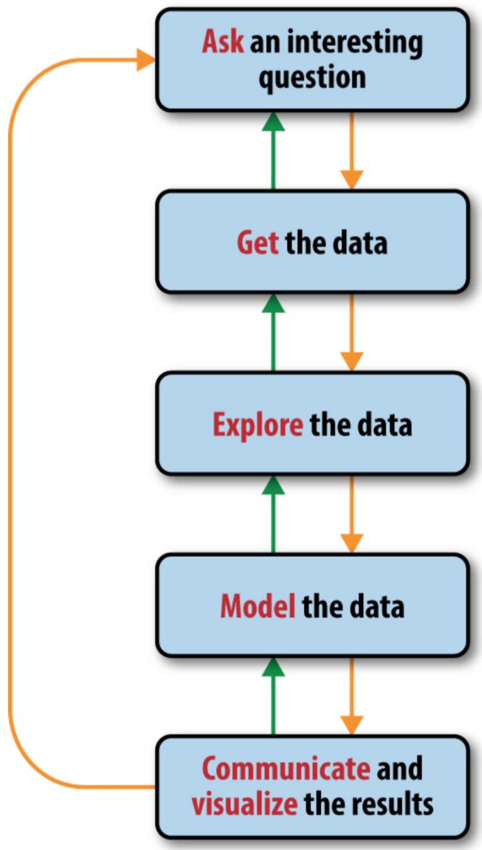
Data Science is Science

- Have a question
- Output is **findings + methodology**
- Reproducibility matters

Data Science Reproducibility

- Communicate results
- Defend decision making
- Auditable workflow

Data Science Process



Ask an interesting question
Get the data
Explore the data
Model the data
Communicate and visualize the results

What is the scientific goal?
What would you do if you had all the data?
What do you want to predict or estimate?

How was the data sampled?
Which data is relevant?
Are there privacy issues?

Plot the data.
Are there anomalies?
Are there patterns?

Build a model.
Fit the model.
Validate the model.

What did we learn?
Do the results make sense?
Can we tell a story?

Source: Harvard CS 109

Data Science and Python

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

scikit learn

IP[y]:

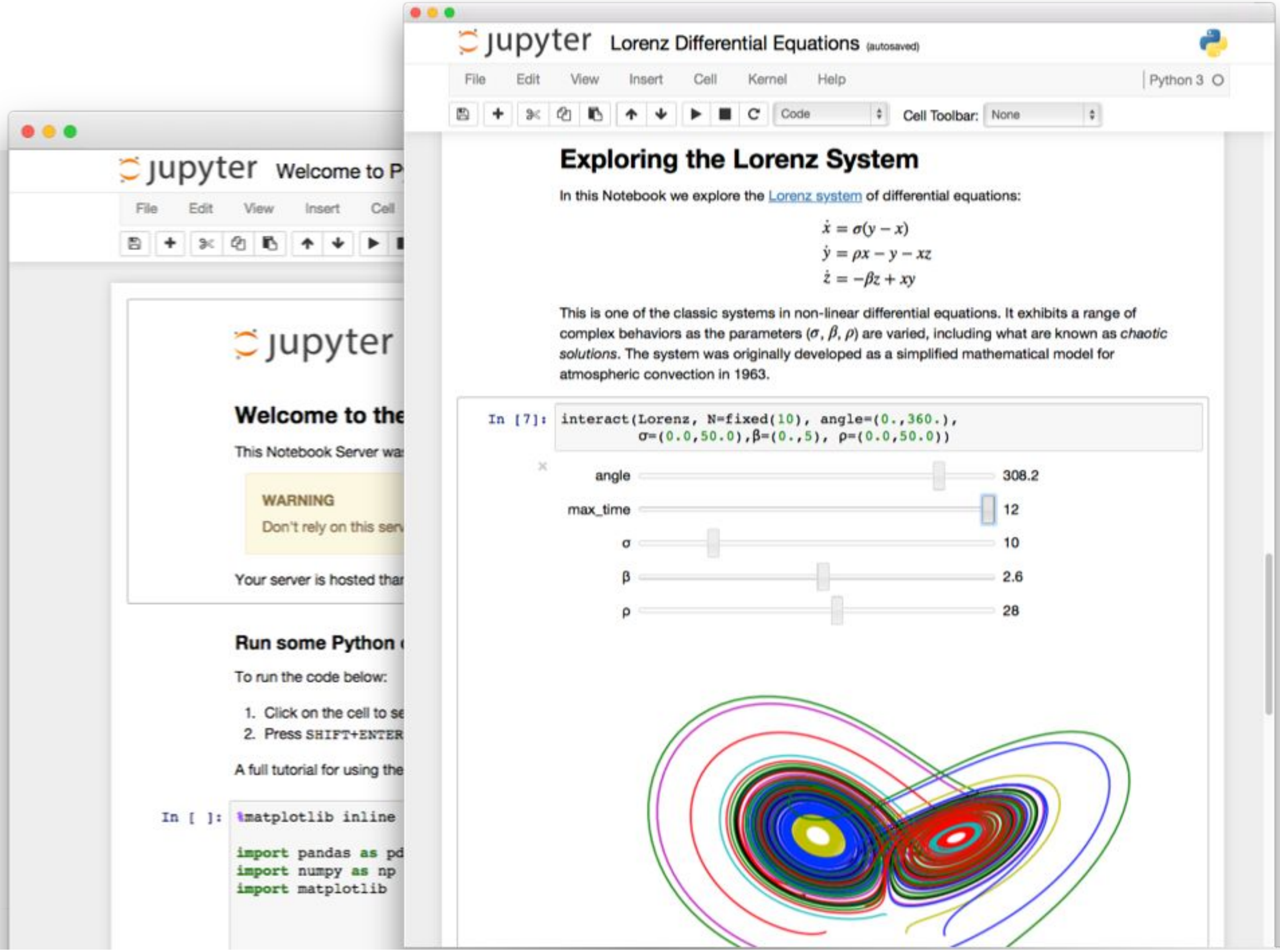
IPython

jupyter

Jupyter Notebooks

- Create / Share documents containing:
  - Live code
  - Equations
  - Visualizations
  - Explanatory Text

- Perfect for Data Science Workflows



Welcome to the

This Notebook Server wa

WARNING

Don't rely on this serv

Your server is hosted thar

Run some Python

To run the code below:

1. Click on the cell to se
2. Press SHIFT+ENTER

A full tutorial for using the

```
In [ ]: %matplotlib inline

        import pandas as pd
        import numpy as np
        import matplotlib
```

Exploring the Lorenz System

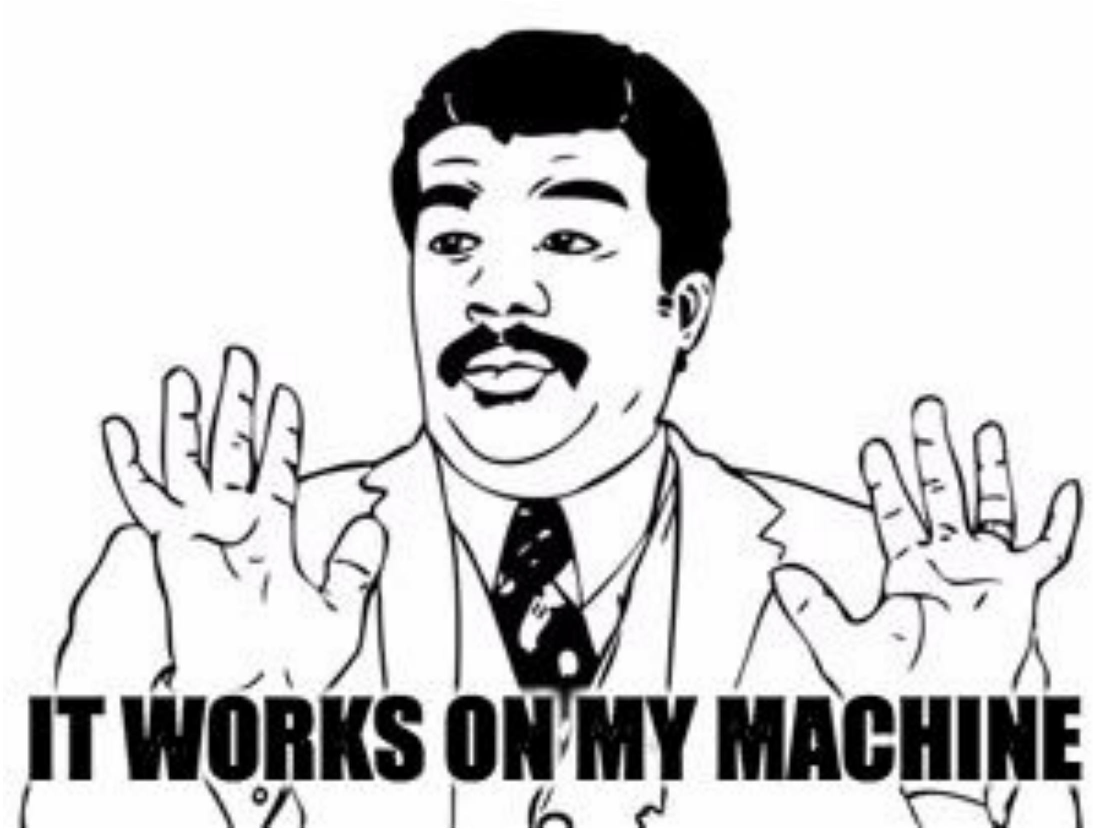
In this Notebook we explore the Lorenz system of differential equations:

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

This is one of the classic systems in non-linear differential equations. It exhibits a range of complex behaviors as the parameters (σ, β, ρ) are varied, including what are known as *chaotic solutions*. The system was originally developed as a simplified mathematical model for atmospheric convection in 1963.

```
In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.),
                 σ=(0.0,50.0),β=(0.,5), ρ=(0.0,50.0))
```

Jupyter Limitations



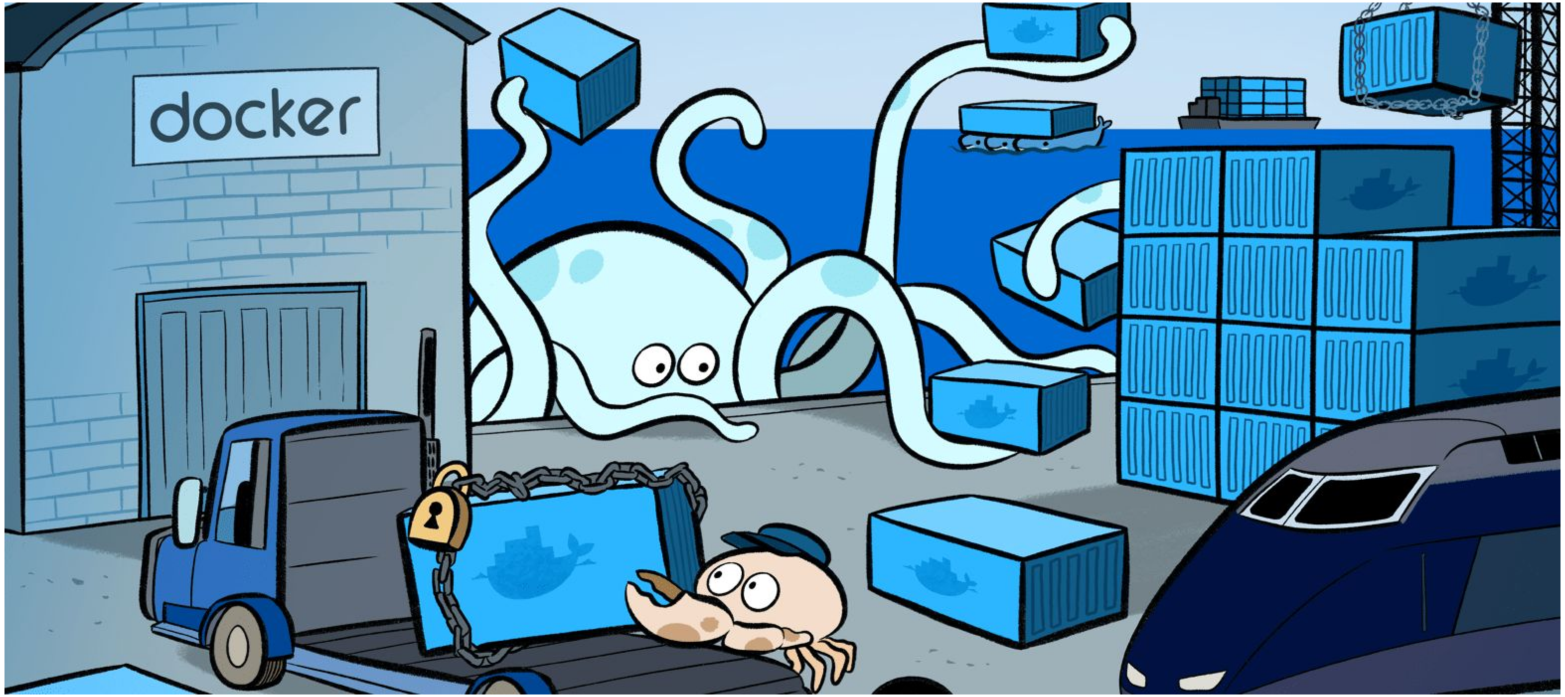
Docker

Introduction to Docker

- Docker allows us to **package** and **run** applications in an **isolated environment**

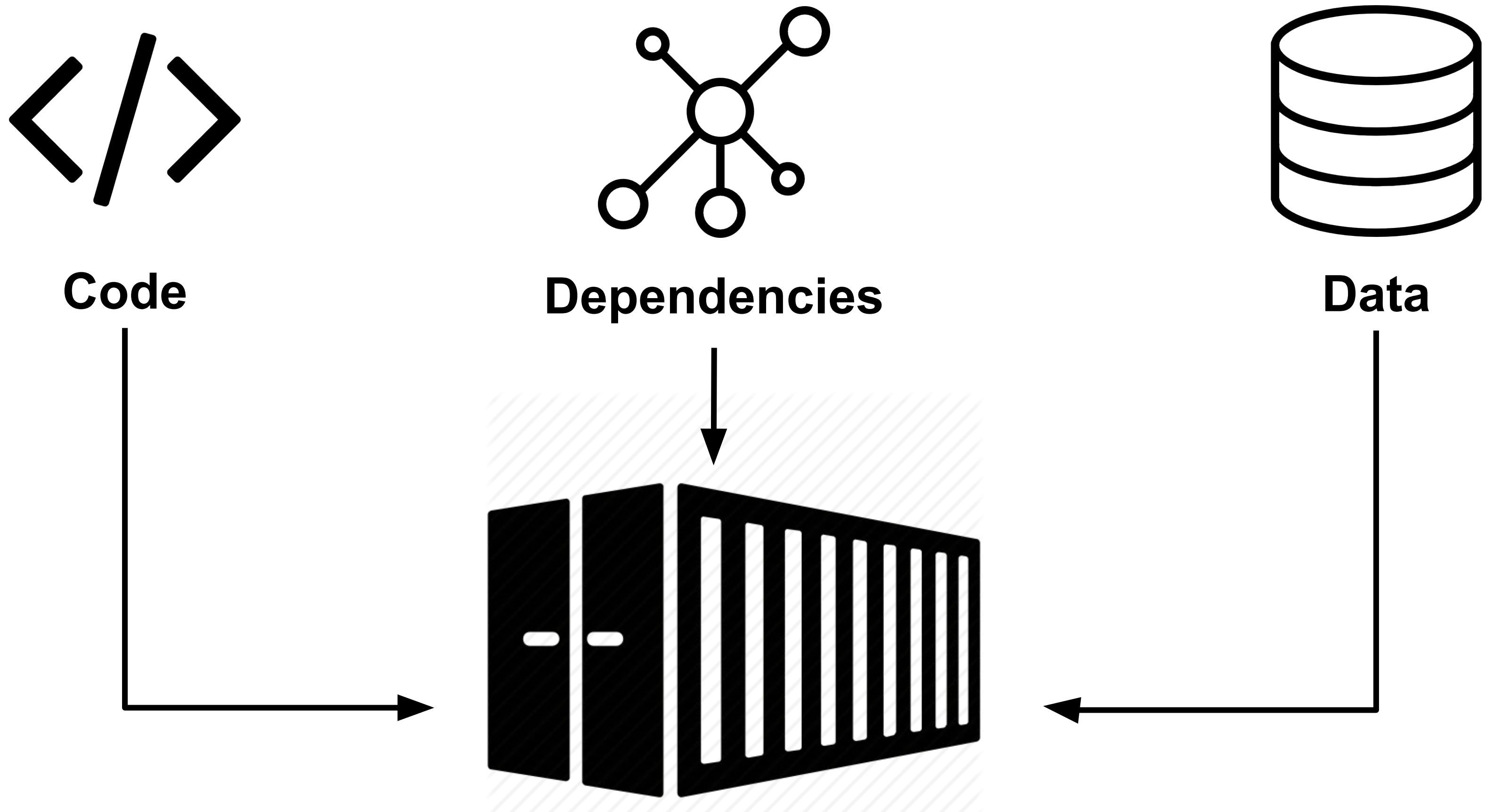
Source: Docker Docs

Shipping Container Analogy

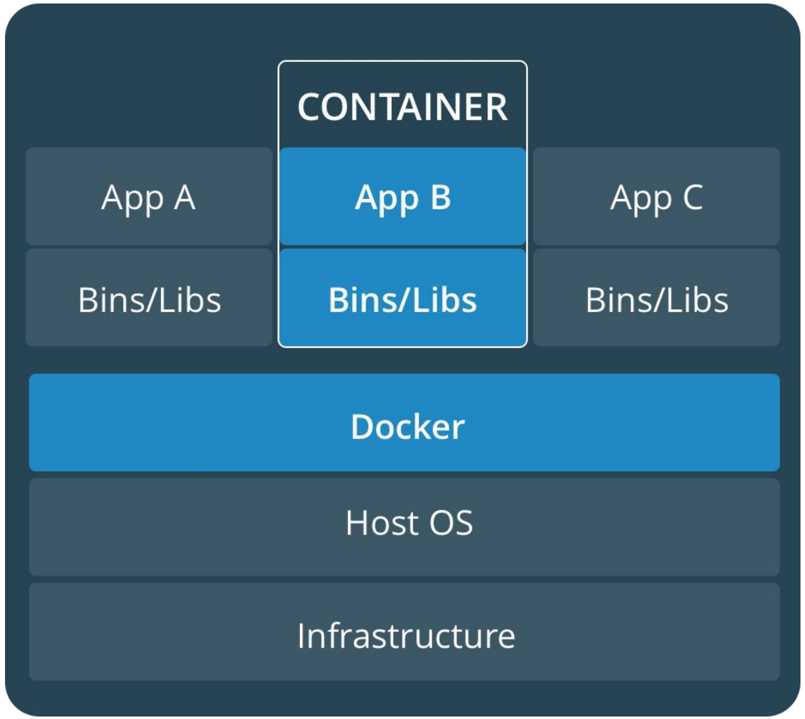


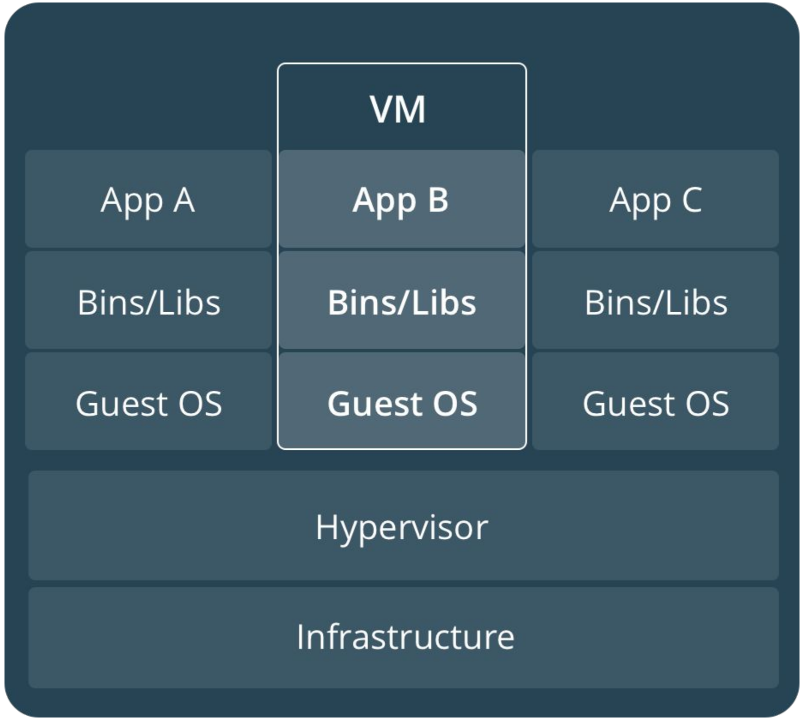
Source: Docker

Software Containers



Docker Containers vs Virtual Machines





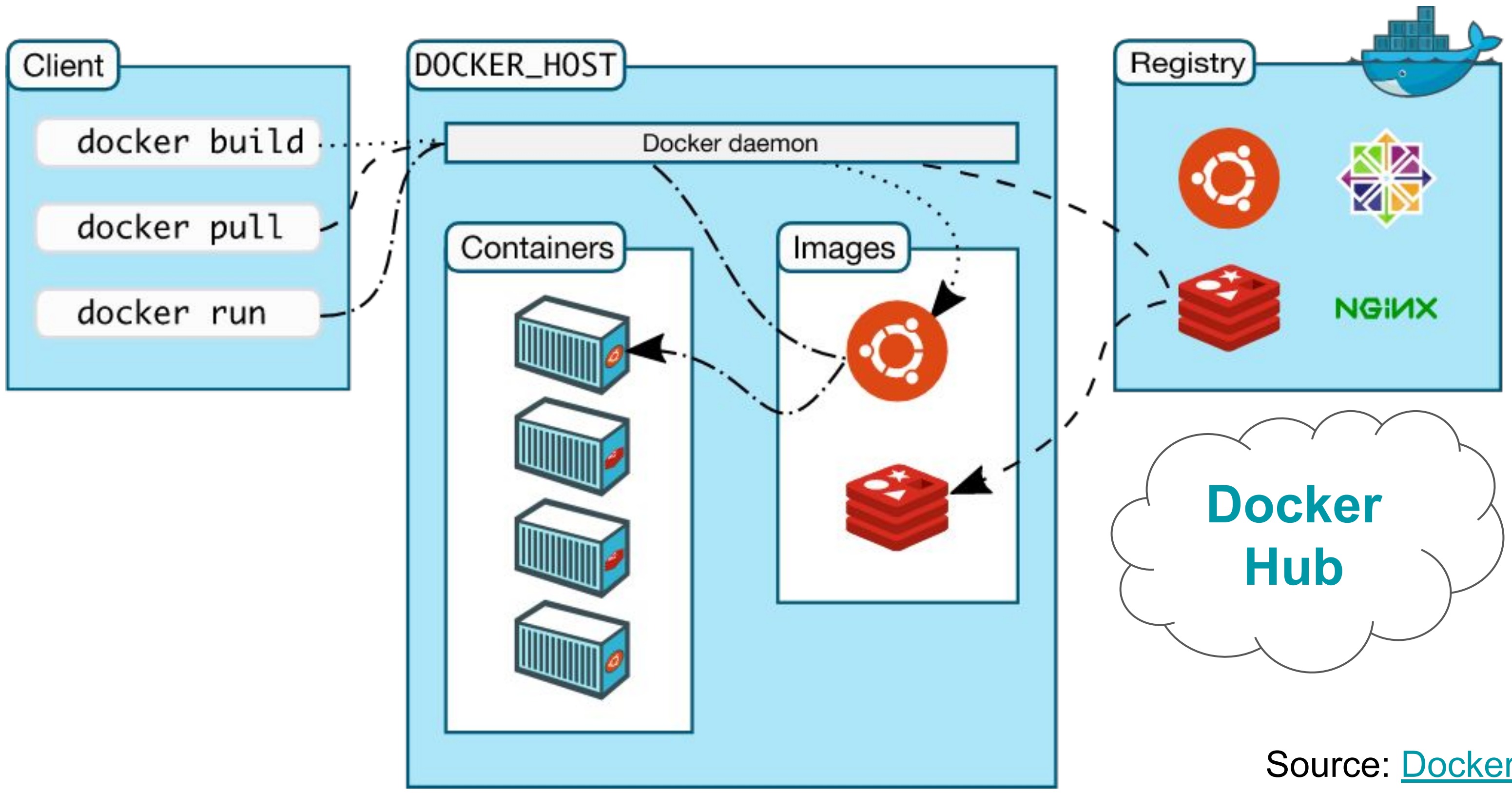
Source: Docker Docs

Docker Use Cases

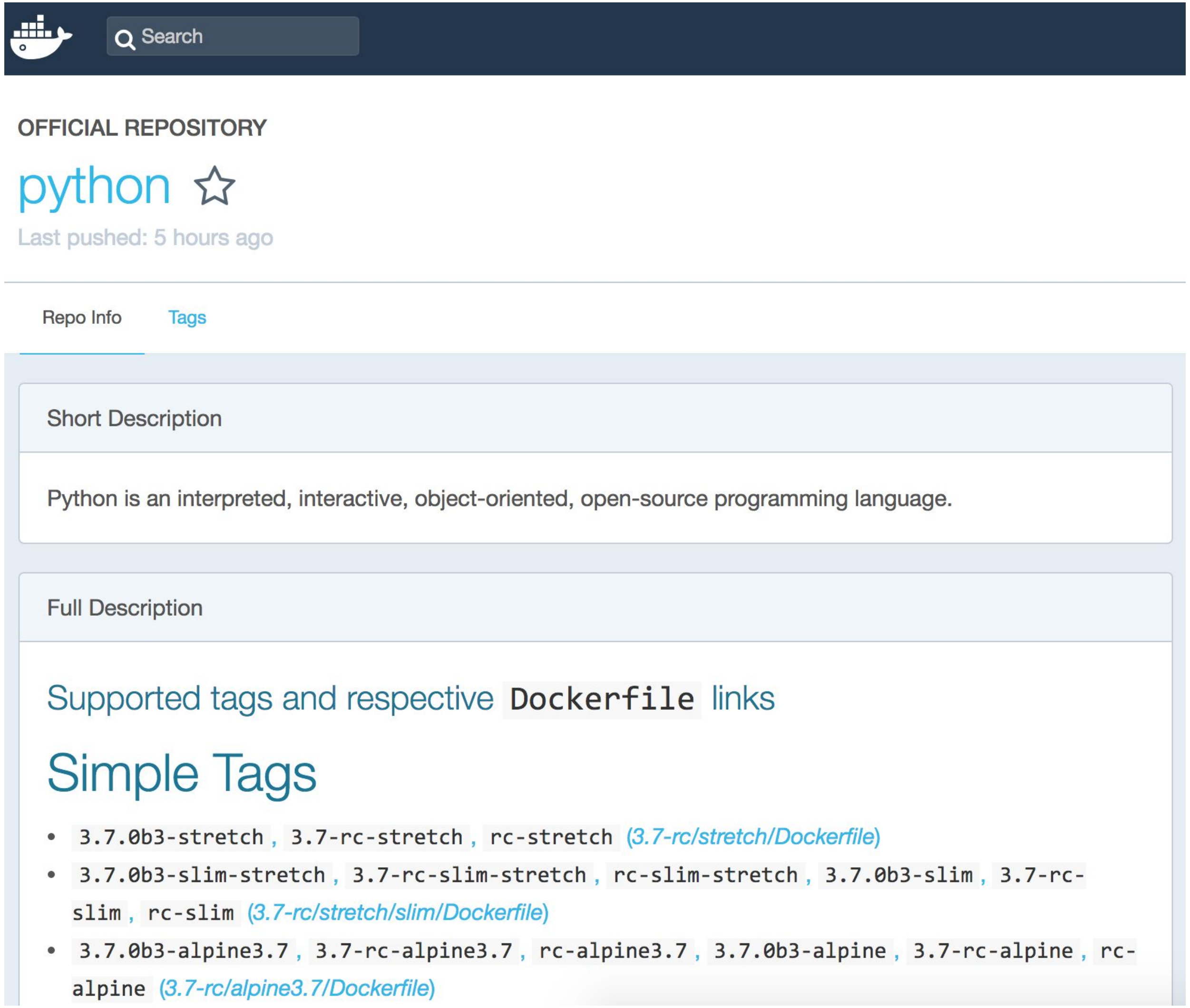
- Streamline development workflows
  - Continuous Integration and Deployment (CI/CD)
- Microservices
  - But remember, monolith first
- Reproducible Data Science

Source: Docker

Docker Architecture: Overview



Source: Docker Docs



OFFICIAL REPOSITORY

python

Last pushed: 5 hours ago

Short Description

Python is an interpreted, interactive, object-oriented, open-source programming language.

Full Description

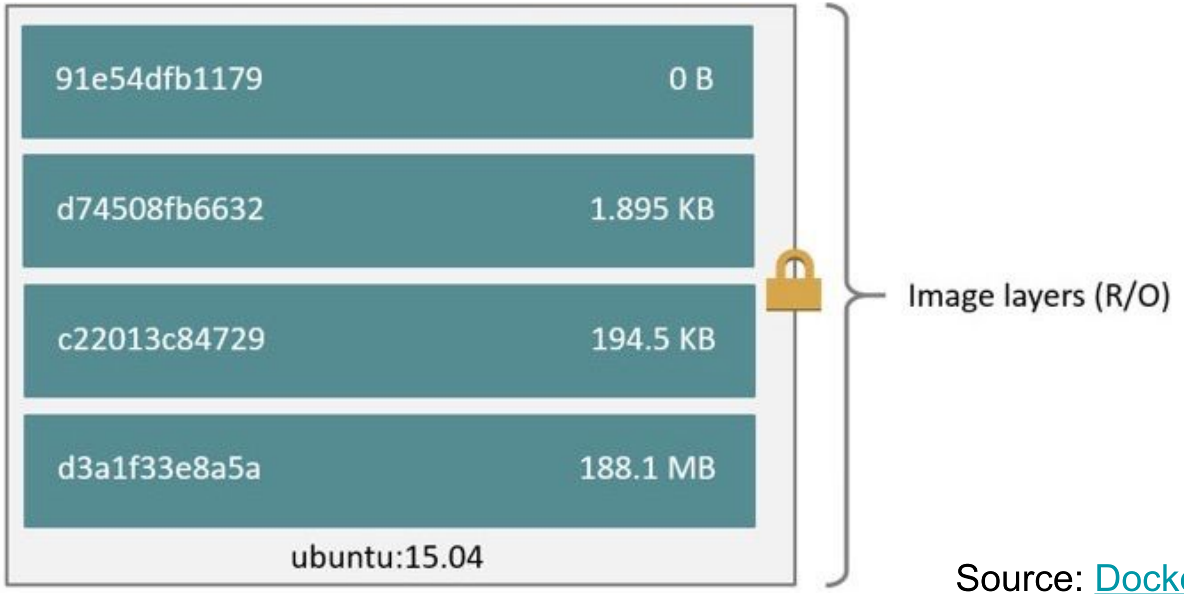
Supported tags and respective `Dockerfile` links

Simple Tags

- `3.7.0b3-stretch`, `3.7-rc-stretch`, `rc-stretch` (3.7-rc/stretch/Dockerfile)
- `3.7.0b3-slim-stretch`, `3.7-rc-slim-stretch`, `rc-slim-stretch`, `3.7.0b3-slim`, `3.7-rc-slim`, `rc-slim` (3.7-rc/stretch/slim/Dockerfile)
- `3.7.0b3-alpine3.7`, `3.7-rc-alpine3.7`, `rc-alpine3.7`, `3.7.0b3-alpine`, `3.7-rc-alpine`, `rc-alpine` (3.7-rc/alpine3.7/Dockerfile)

Docker Image

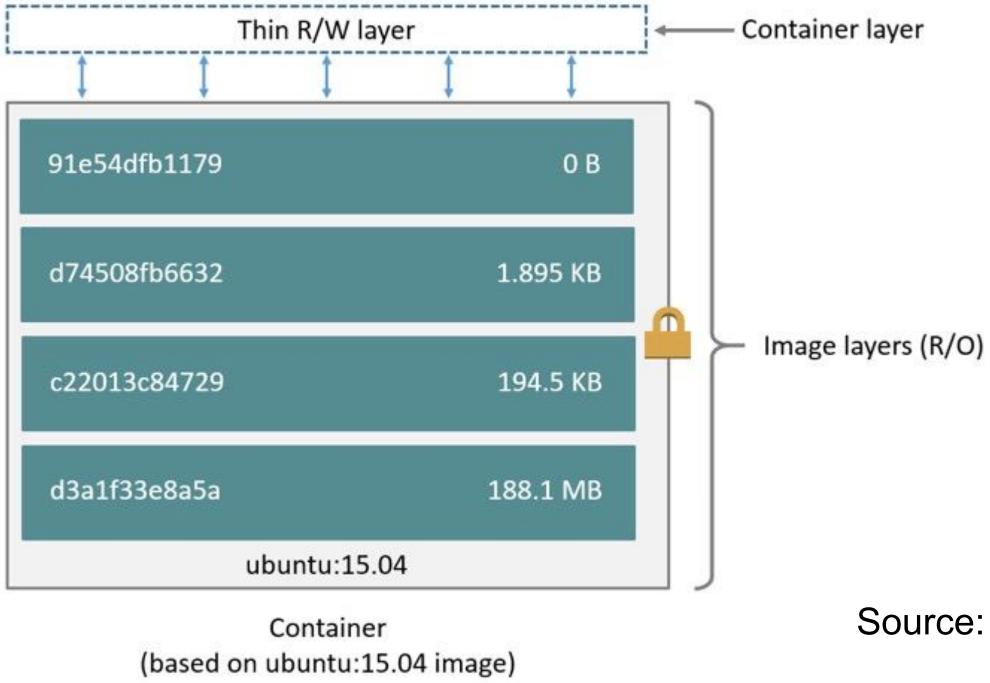
- A frozen snapshot of a container



Source: [Docker Docs]()

Docker Containers

- Runtime instance: `docker run [image]`



Source: [Docker Docs]()

Object-Oriented Programming Analogy

- Images : Classes
- Layers : Inheritance
- Containers : Objects

Lab: Docker Essentials

http://bit.ly/d4ds-lab1

Creating Docker Images

1. Freeze container using `docker commit`

2. Dockerfile and `docker build` * **Preferred** *
    - File containing all commands used to assemble image
    - Automated build

Dockerfile Commands

- FROM - sets base image
- LABEL - adds metadata to image
  - MAINTAINER is deprecated
  - LABEL maintainer=”Aly Sivji <alysivji@gmail.com>”
- COPY - copies files / directories into image
  - .dockerignore
- ENV - sets environment variable
- WORKDIR - sets working directory

Source: Docker Docs

Dockerfile Commands

- RUN - executes shell commands in a new layer

```
RUN pip install jupyter

RUN pip install pandas
```

2 layers

```
RUN pip install jupyter && \
    pip install pandas
```

1 layer

Dockerfile - Configuring Runtime

- ENTRYPOINT - configures container to run as executable
- CMD - provides default for executing container
  - CMD and ENTRYPOINT interaction

- Two forms:
  - Shell `CMD python hello-world.py`
  - Exec (preferred) `CMD [“python”, “hello-world.py”]`

- Additional Information

Hello World Dockerfile

```
# Use latest Python runtime as base image
FROM python:3.6.3-alpine3.6

# Set the working directory to /app and copy current dir
WORKDIR /app
COPY . /app

# Run hello_world.py when the container launches
CMD ["python", "hello_world.py"]
```

Building Image

```
$ docker build -t hello-world .

Sending build context to Docker daemon  3.072kB

Step 1/4 : FROM python:3.6.3-alpine3.6

…

Successfully built f4e5a0ccfcd5

Successfully tagged hello-world:latest
```

Source: [Docker Docs]

Container Commands

- Create Container

```
$ docker run hello-world

Hello World!
```

- Restart Container

```
$ docker start -ia [CONTAINER]
```

Source: [Docker Docs]

`$ docker run [OPTIONS] IMAGE [COMMAND]`

- `[Options]`

`-d`	Detached (runs in background)
`-a`	Attach to STDOUT/STDERR
`-i`	Interactive (keeps STDIN open)
`-t`	Allocates pseudo-TTY
`--name [NAME]`	Set the container name

- `[Command]`
    - Can pass in parameters or **`/bin/sh`** to get into container’s shell

Source: Docker Docs

Managing Data Inside Containers

- Data disappears when we delete a container
- `docker cp` to copy files in/out of containers
- Mount data volume inside container

Adding Data Volume to Container

```
$ docker run -v /full/local/path:/mounted_dir
```

Host Path **Container Path**

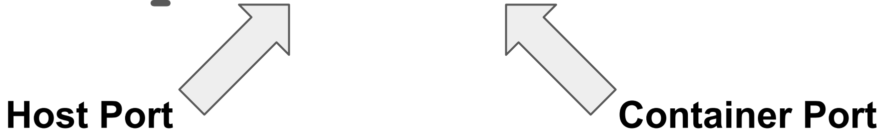
- Best Practice: Add VOLUME command to Dockerfile

```
# Create mount point for external volumes
VOLUME /mounted_dir
```

Binding Ports

- Setup port forwarding to connect to containers

```
$ docker run -p 9999:8888
```



- Best Practice: Add EXPOSE command to Dockerfile

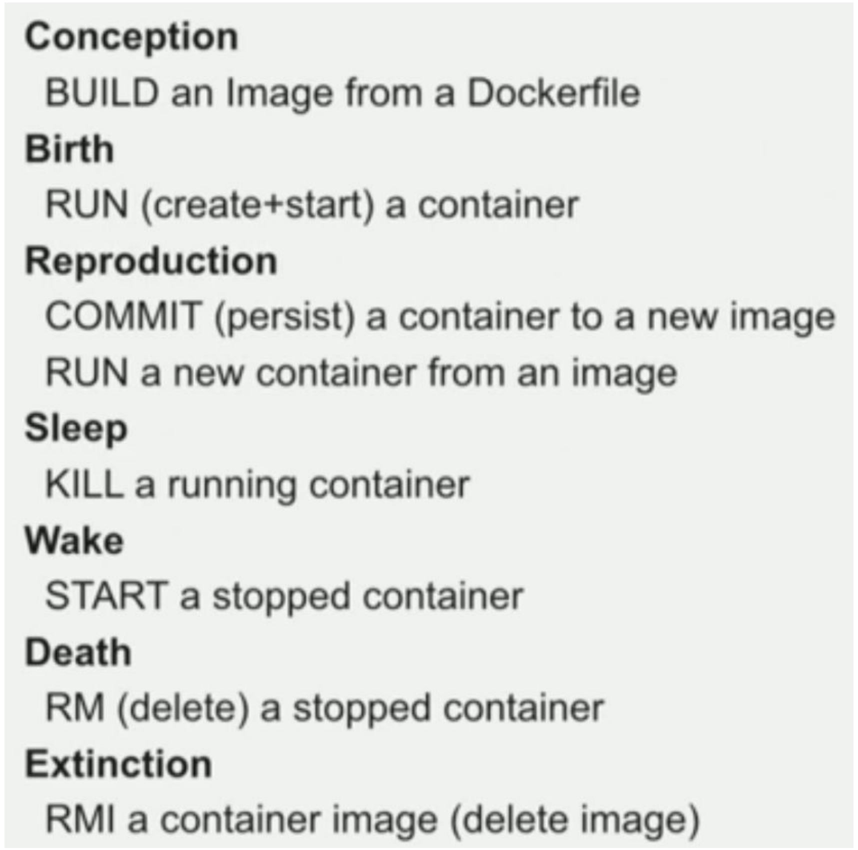
```
# Make port 8888 available to outside world
EXPOSE 8888
```

Dockerfile - Best Practices

- Be explicit about build process
- Containers should be stateless
- Use **`.dockerignore`** file
- Avoid installing unnecessary packages
  - Clean cache after installation
- Each container should have only one concern / purpose
- Minimize the number of layers
  - Multi-line arguments, sort alphabetically
- CMD should be used to run processes inside container
  - Advanced users should use it in conjunction with ENTRYPOINT
- MAINTAINER is deprecated; use LABEL

Source: Docker Docs

Docker Container Lifecycle



Conception

BUILD an Image from a Dockerfile

Birth

RUN (create+start) a container

Reproduction

COMMIT (persist) a container to a new image

RUN a new container from an image

Sleep

KILL a running container

Wake

START a stopped container

Death

RM (delete) a stopped container

Extinction

RMI a container image (delete image)

Source: Docker 101

Docker Commands: Containers

Lifecycle

- docker create
- docker rename
- ☆ docker run
- ☆ docker rm
- docker update

Misc

- docker cp
- docker export
- ☆ docker exec

Start/Stop

- ☆ docker start
- docker stop
- docker restart
- docker pause
- docker unpause
- docker wait
- ☆ docker kill
- ☆ docker attach

Info

- ☆ docker ps
- docker logs
- ☆ docker inspect
- docker events
- docker port
- ☆ docker top
- docker stats
- docker diff

Source: Docker Cheat Sheet

Docker Commands: Images

Lifecycle

- ☆ `docker images`
- `docker import`
- ☆ `docker build`
- `docker commit`
- ☆ `docker rmi`
- `docker load`
- `docker save`

Info

- `docker history`
- `docker tag`

Registry

- `docker login`
- `docker logout`
- `docker search`
- ☆ `docker pull`
- ☆ `docker push`

Source: Docker Cheat Sheet

Tips and Tricks

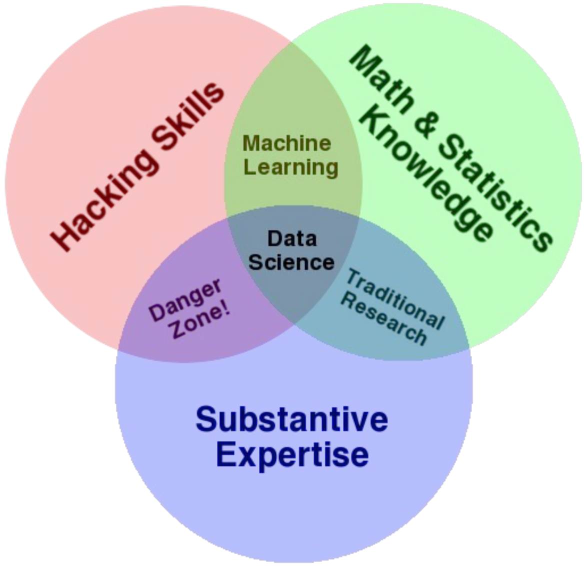
- Smaller images are better. Install only the packages you need.
  - Look into different Linux distributions (Alpine Linux... only 5MB!)
  - Clear cache after installing or use no-cache flags!
- Link bash_history and keep track of commands typed inside container
- dockviz command line app to visualize docker data
- Ctrl + P + Q to detach from container while inside shell
- Instructions on mounting symbolic links
- Always set IP address for apps running inside container to `0.0.0.0`

Lab: Dockerfile Essentials

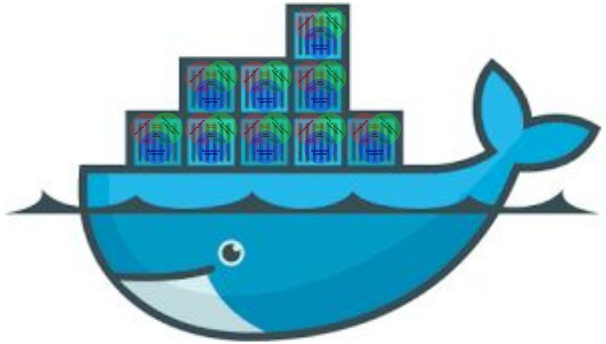
http://bit.ly/d4ds-lab2

Questions?

Data Science Workflows with Docker



+



Self-Contained Container (Workflow #1)

- Problem: Sharing results (Jupyter notebook)

- Workflow:
  - Create Docker image with libraries, data and notebook
  - Push image to DockerHub

Self-Contained Container: Dockerfile

```
FROM python:3.6.3-slim

LABEL maintainer="Aly Sivji <alysivji@gmail.com>"

WORKDIR /app

COPY . /app

RUN pip --no-cache-dir install numpy pandas seaborn sklearn jupyter

EXPOSE 8888


# Run app.py when the container launches

CMD ["jupyter", "notebook", "--ip='*'", "--port=8888",

"--no-browser", "--allow-root"]
```

Self-Contained Container: Commands

- Build Image

```
$ docker build -t alysivji/workflow1-self-contained .
```

- Initialize Container

```
$ docker run -p 9999:8888
alysivji/workflow1-self-contained
```

- Restart Container

```
$ docker start -ia [CONTAINER]
```

Self-Contained Container: Docker Hub

- Upload to Docker Hub

```
$ docker login
$ docker push [full-image-name]
```

- Download Image

```
$ docker pull [full-image-name]
```

- Instructions from previous slide for lifecycle

Data Science Project (Workflow #2)

- Problem:
  - Need to standardize team development environment
  - Project based workflows

- Workflow:
  - Create team / project image with dev environment
  - Mount volume containing notebooks and data

Data Science Project: Benefits

- Separate out projects
- Create container to onboard new employees
- Easy to upgrade dependencies
  - Build automated testing pipeline

Data Science Project: Dockerfile

```
FROM continuumio/miniconda3

LABEL maintainer="alysivji@gmail.com"

WORKDIR /app

RUN conda install jupyter -y && \

    conda clean -y -all

EXPOSE 8888

VOLUME /app

CMD ["jupyter", "notebook", "--ip='*'", "--port=8888",

"--no-browser", "--allow-root"]
```

Data Science Project: Commands

- Build Image

```
$ docker build -t
alysivji/workflow2-data-science-project .
```

- Initialize Container

```
$ docker run -p 9999:8888 -v
/Users/alysivji/siv-dev/datasci:/app
alysivji/workflow2-data-science-project
```

- Restart Container

```
$ docker start -ia [CONTAINER]
```

Data Driven App (Workflow #3)

- Problem: Distributing application

- Workflow:
  - Package app in image and deploy using Docker

- Further Reading
  - Docker Compose

Data Driven App: Dashboard

- Data stored on local machine

- Create & package dashboard inside container
  - Dash Tutorial

- Container is an executable on top of data
  - Start container to view dashboard

Data Driven App: Dockerfile

```
FROM python:3.6.3-alpine3.6

LABEL maintainer="alysivji@gmail.com"

WORKDIR /app

COPY . /app

RUN pip --no-cache-dir install -r /app/requirements.txt

EXPOSE 8050

VOLUME /app/data

ENTRYPOINT ["python"]

CMD ["plot_timeseries.py"]
```

Data Driven App: Commands

- Build Image

```
$ docker build -t alysivji/workflow3-data-driven-app .
```

- Initialize Container

```
$ docker run -p 8050:8050 -v
/Users/alysivji/siv-dev/docker-example:/app/data
--name dashboard alysivji/workflow3-data-driven-app
```

- Restart Container

```
$ docker start -ia dashboard
```

Lab: Data Science Workflows using Docker

http://bit.ly/d4ds-lab3

Questions?

Docker Compose

Slides available at http://bit.ly/d4ds-compose-slides

Putting it Together

Talk Recommender

Agenda

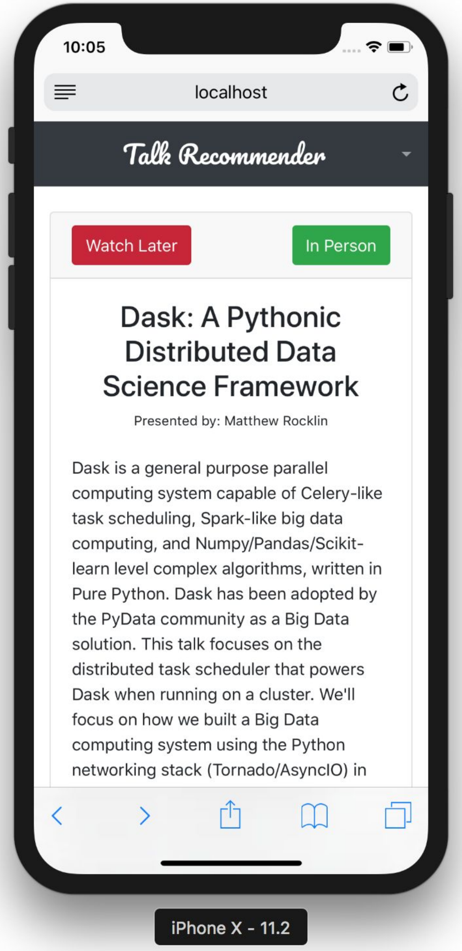
- Introduce problem
- Build solution in Jupyter Notebook
- Deploy solution as a service

PYCON
CLEVELAND 2018

32 Tutorials
12 Sponsor Workshops
16 Education Summit Talks
95 Conference Talks

PYCON
CLEVELAND 2018

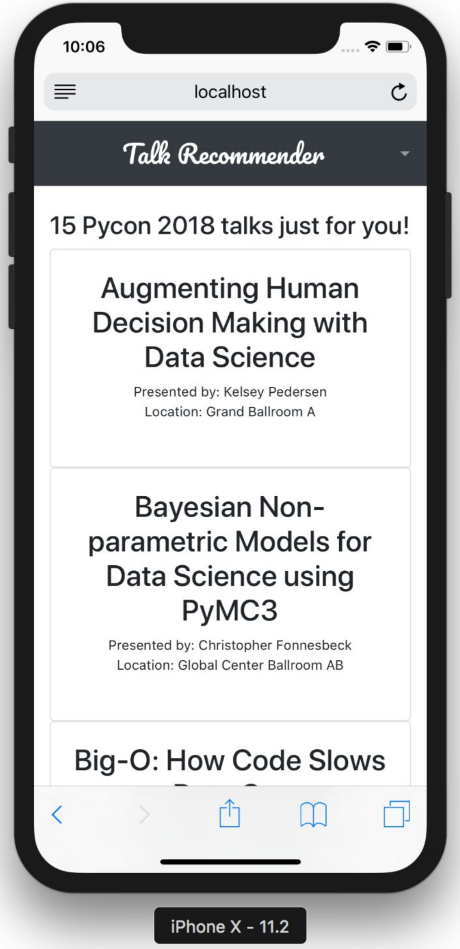
32 Tutorials
12 Sponsor Workshops
16 Education Summit Talks
95 Conference Talks



Dask: A Pythonic Distributed Data Science Framework

Presented by: Matthew Rocklin

Dask is a general purpose parallel computing system capable of Celery-like task scheduling, Spark-like big data computing, and Numpy/Pandas/Scikit-learn level complex algorithms, written in Pure Python. Dask has been adopted by the PyData community as a Big Data solution. This talk focuses on the distributed task scheduler that powers Dask when running on a cluster. We'll focus on how we built a Big Data computing system using the Python networking stack (Tornado/AsyncIO) in



15 Pycon 2018 talks just for you!

Augmenting Human Decision Making with Data Science

Presented by: Kelsey Pedersen
Location: Grand Ballroom A

Bayesian Non-parametric Models for Data Science using PyMC3

Presented by: Christopher Fonnesbeck
Location: Global Center Ballroom AB

Big-O: How Code Slows

Demo

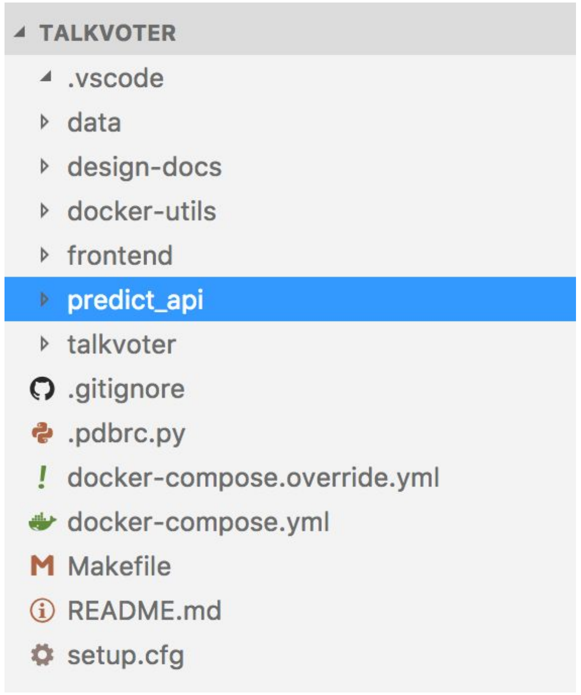
Lab: Data Science Essentials

http://bit.ly/d4ds-lab4

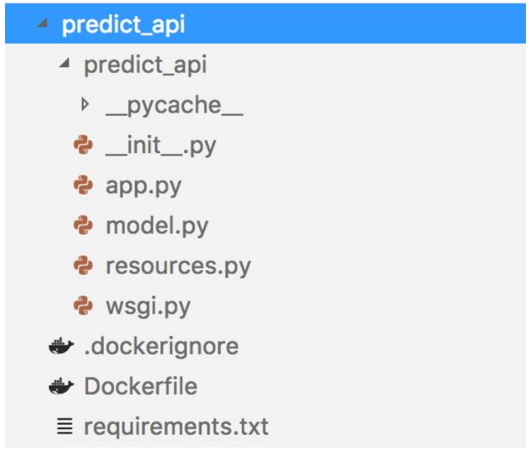
Talk Recommender: Code Walk Thru

https://github.com/docker-for-data-science/talkvoter

Talk Recommender: predict_api

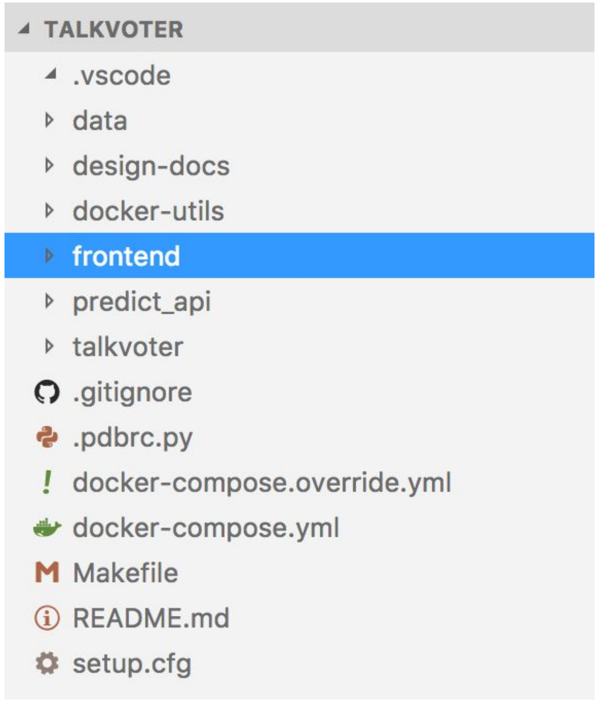


TALKVOTER
- .vscode
- data
- design-docs
- docker-utils
- frontend
- predict_api
- talkvoter
- .gitignore
- .pdbrc.py
- docker-compose.override.yml
- docker-compose.yml
- Makefile
- README.md
- setup.cfg

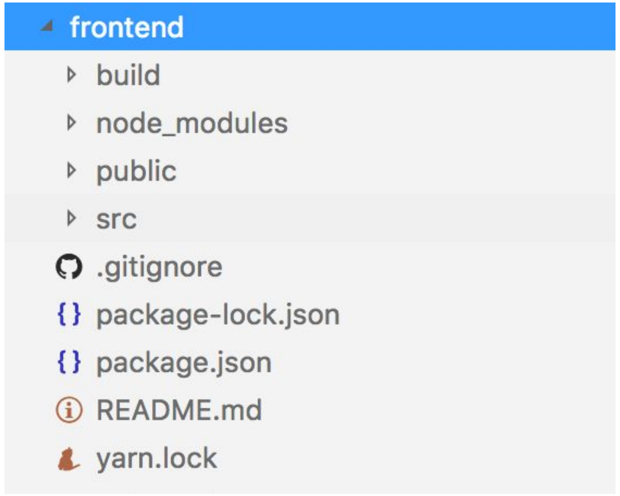


predict_api
- predict_api
  - __pycache__
  - __init__.py
  - app.py
  - model.py
  - resources.py
  - wsgi.py
- .dockerignore
- Dockerfile
- requirements.txt

Talk Recommender: frontend

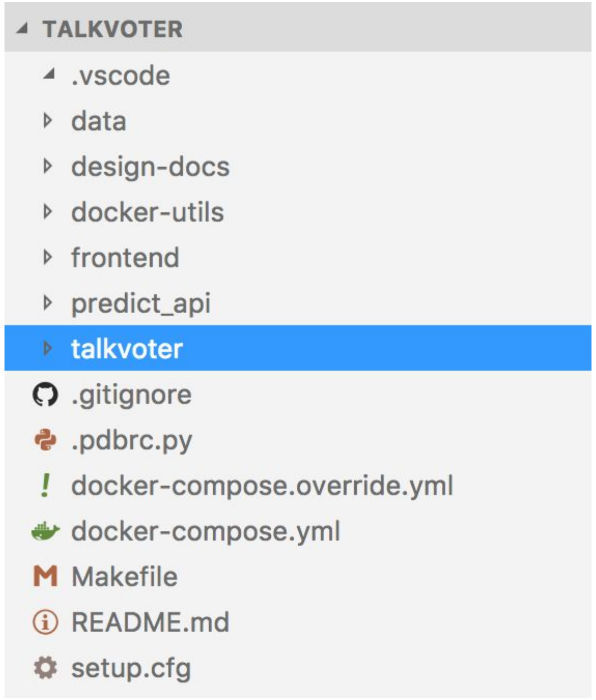


TALKVOTER
- .vscode
- data
- design-docs
- docker-utils
- frontend
- predict_api
- talkvoter
- .gitignore
- .pdbrc.py
- docker-compose.override.yml
- docker-compose.yml
- Makefile
- README.md
- setup.cfg

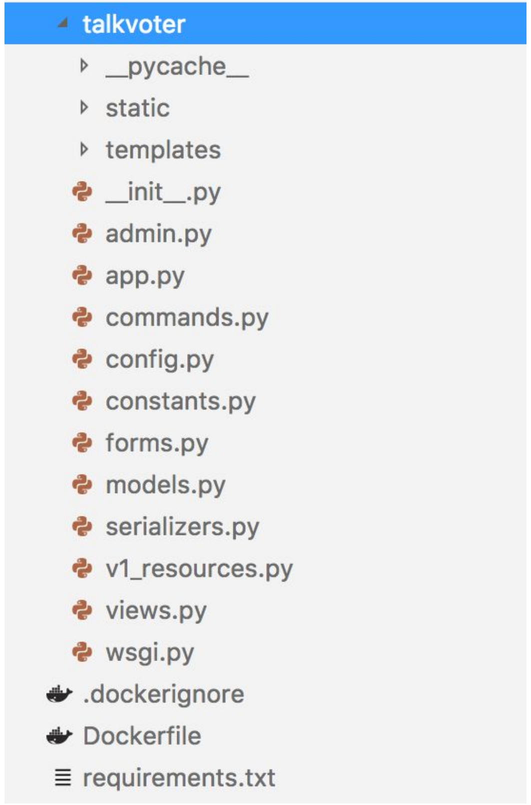


frontend
- build
- node_modules
- public
- src
- .gitignore
- package-lock.json
- package.json
- README.md
- yarn.lock

Talk Recommender: talkvoter

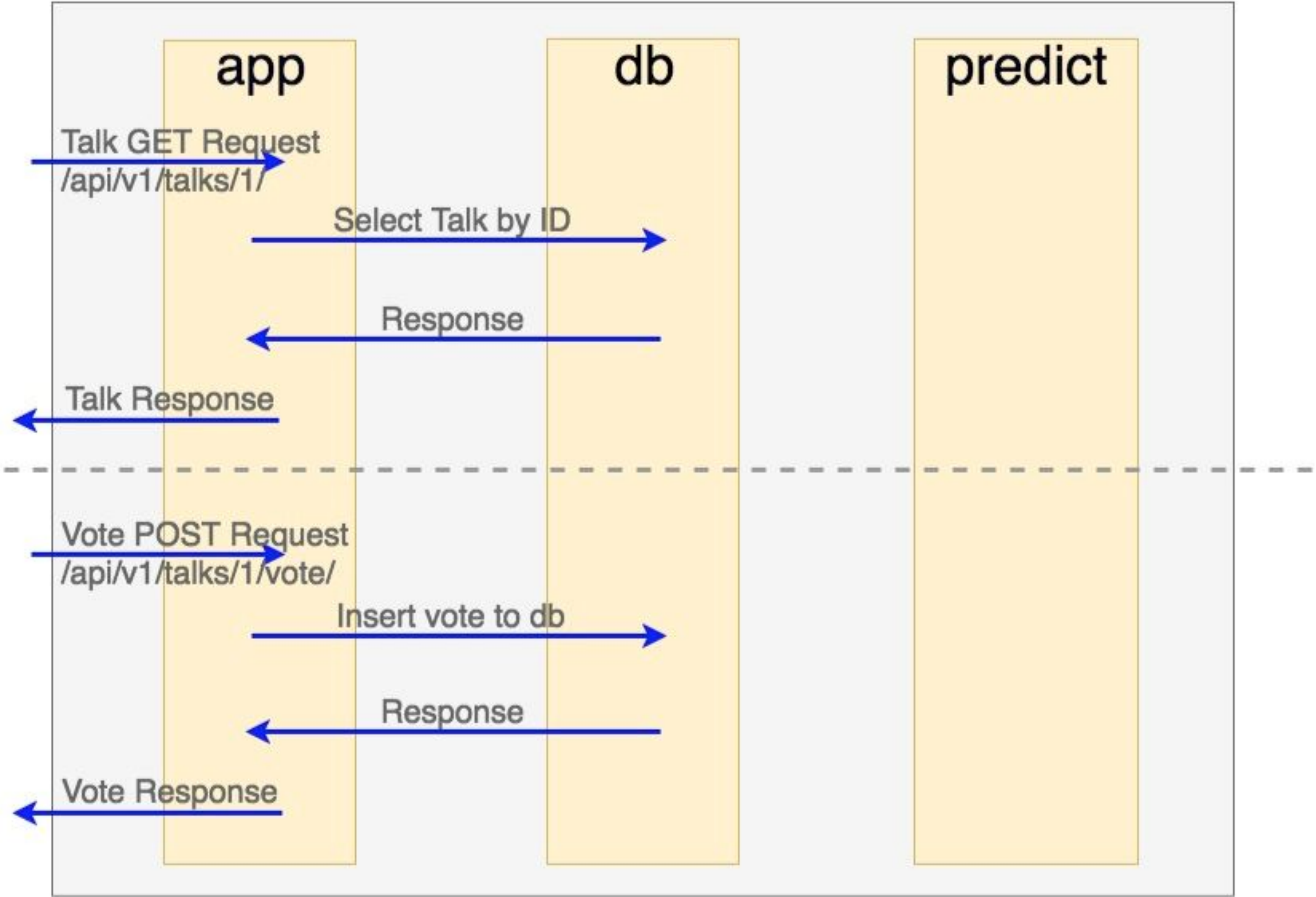


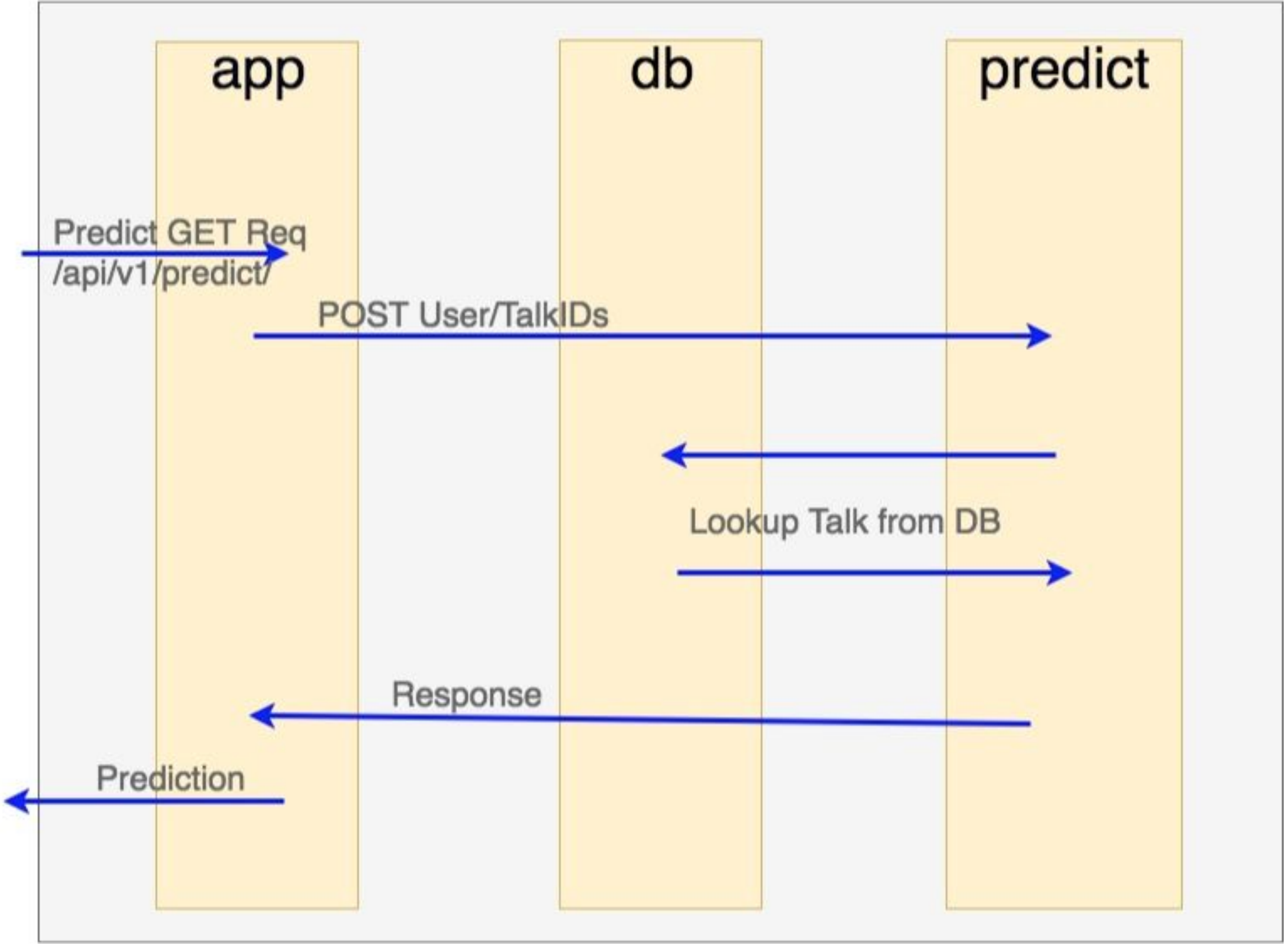
```
TALKVOTER
  .vscode
  data
  design-docs
  docker-utils
  frontend
  predict_api
  talkvoter
  .gitignore
  .pdbrc.py
  docker-compose.override.yml
  docker-compose.yml
  Makefile
  README.md
  setup.cfg
```

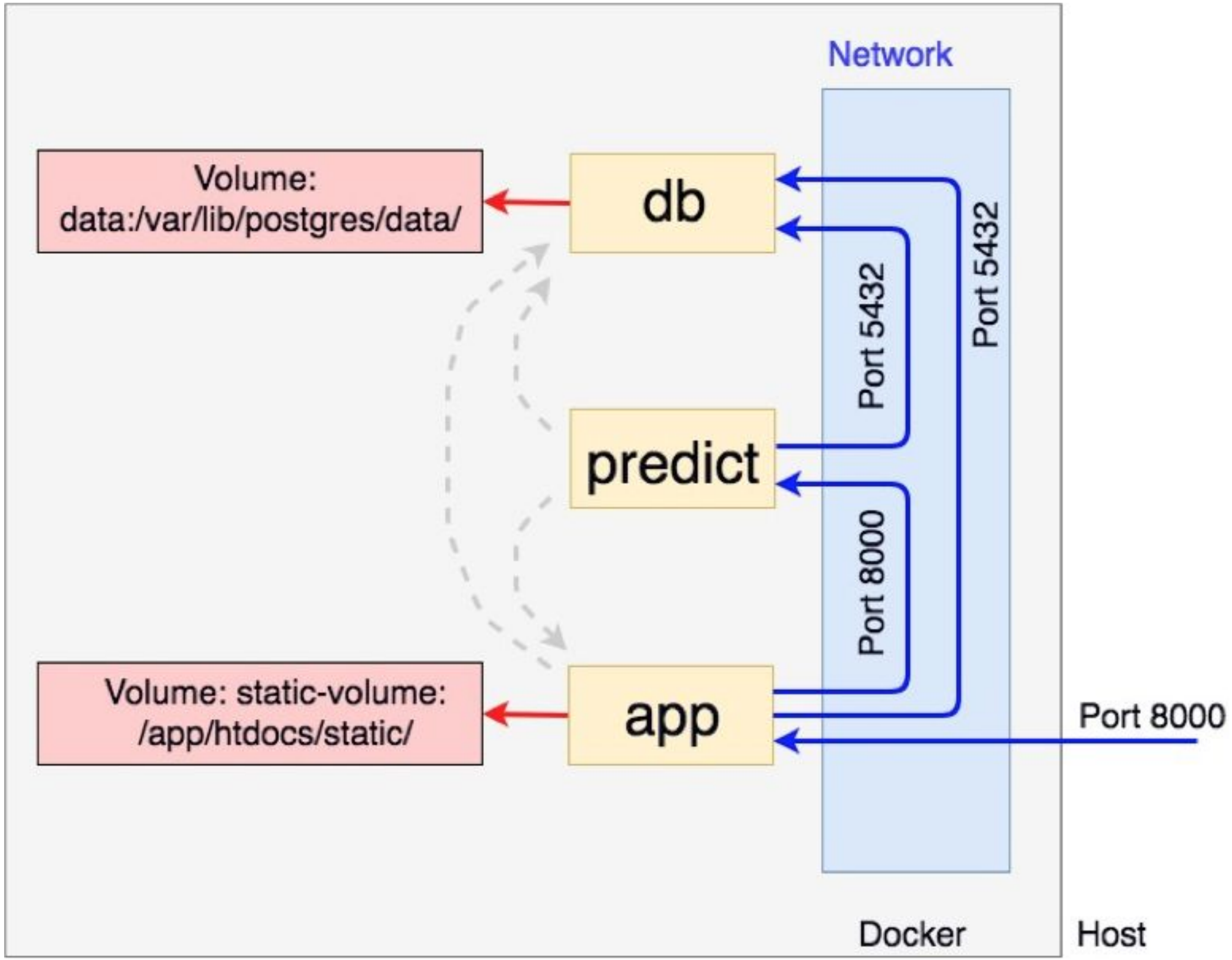


```
talkvoter
  __pycache__
  static
  templates
  __init__.py
  admin.py
  app.py
  commands.py
  config.py
  constants.py
  forms.py
  models.py
  serializers.py
  v1_resources.py
  views.py
  wsgi.py
.dockerignore
Dockerfile
requirements.txt
```

Talk Recommender Details


app
db
predict
Talk GET Request
/api/v1/talks/1/
Select Talk by ID
Response
Talk Response
Vote POST Request
/api/v1/talks/1/vote/
Insert vote to db
Response
Vote Response


app
db
predict
Predict GET Req
/api/v1/predict/
POST User/TalkIDs
Lookup Talk from DB
Response
Prediction


Network
Volume: data:/var/lib/postgres/data/
db
Port 5432
Port 5432
predict
Port 8000
Volume: static-volume: /app/htdocs/static/
app
Port 8000
Docker
Host

Lab: Docker-Compose Essentials

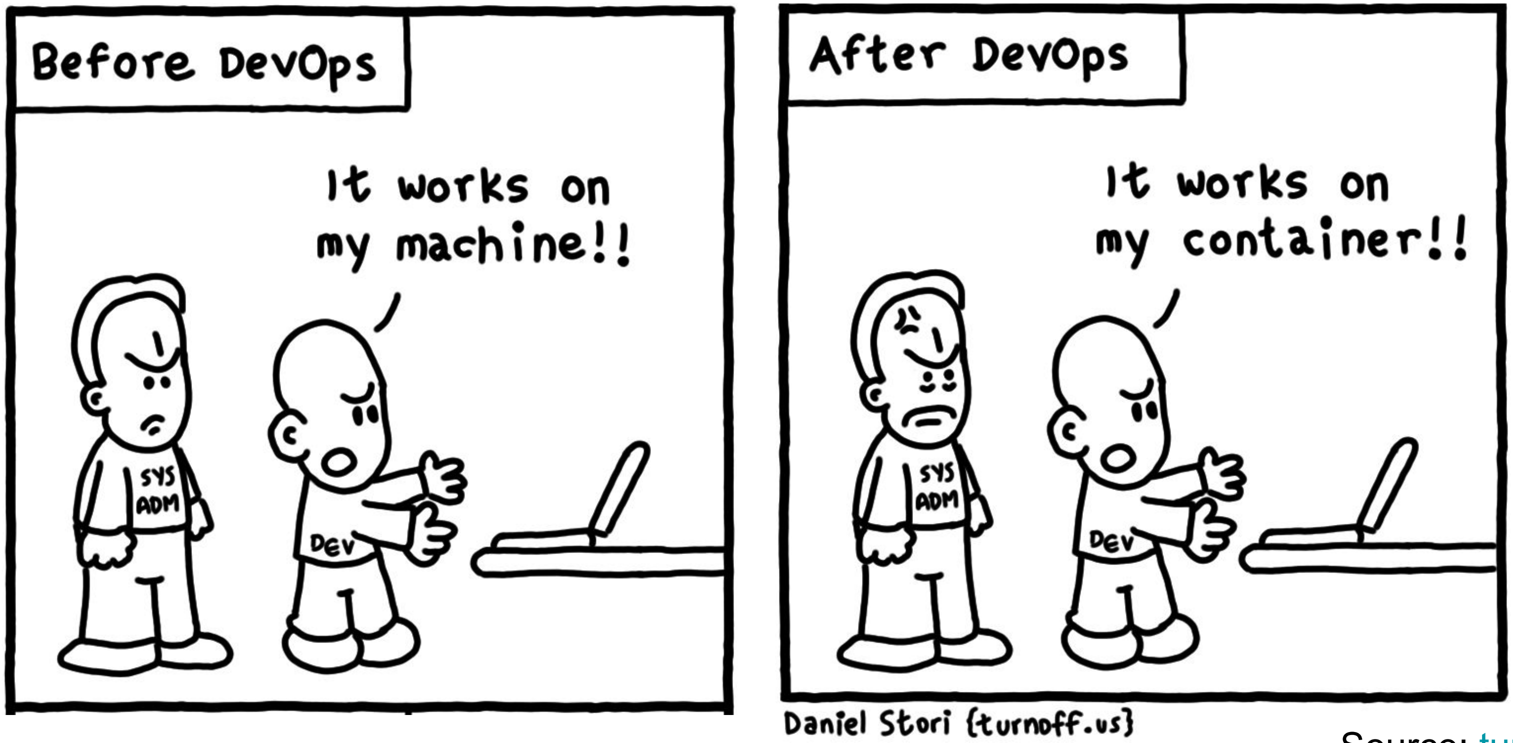
http://bit.ly/d4ds-lab5

Wrapping It Up

Container Workflow Best Practices

- Use official images as base when creating Dockerfile
- Version Docker images, don’t use `latest`
- Think of containers as immutable objects
  - To propagate changes to container, create new image
  - Use image to generate new container
- Use multi-stage builds to keep production image small
  - Copy build artifacts into final image from intermediate build image
- Do not run processes in container as root

Meet the New Excuse (Same as the Old Excuse)



Source: turnoff.us

Next Steps & Additional Resources

- How to Install Docker
- Docker Documentation: Getting Started Guide
- Nigel Poulton's Docker Deep Dive Course
- CenturyLink Developer Center
- Kubernetes

Thank You

Slides: http://bit.ly/d4ds-slides

Github: http://bit.ly/d4ds-tutorial

Twitter: @CaiusSivjus | @JoeJJasinski | @Tathagata

Acknowledgements (Easter Egg)

- ChiPy