PyData

Data Science with Docker

Gabriel Miretti

Who am I?

- Unsuccessful football player
- Computer Scientist
  - Licenciado @ FaMAF - Universidad de Córdoba
- Previously: Quality Assurance Engineer
  - Mostly on integration on large systems (Intel, CONAE)
- Now: Data Science/Engineering Freelancer
  - Working on Social network analysis projects

PyData

Data Science

- Machine Learning
- Deep Learning
- NLP
- Big Data
- …

- TOOLS
- TOOLS
- TOOLS
- TOOLS
- TOOLS
- …

Software Engineering

- Microservices
- Cloud
- Orchestration
- …

- CONTAINERS
- CONTAINERS
- CONTAINERS
- CONTAINERS
- CONTAINERS
- …

PyData

Data Science

- Machine Learning
- Deep Learning
- NLP
- Big D
- ...

- TOO
- TO
- TOOLS
  - ...

Softw e Engineering

ces

RS

RS

ERS

NTAINERS

- CONTAINERS
  - ...

Data Science in Containers

Why use containers?

Data scientist goals:

- Focus on data
  - not software tools
- Take advantage of latest software
  - To improve performance
- Make science
  - Reproducible experiments
- Share results
  - Also materials and methods

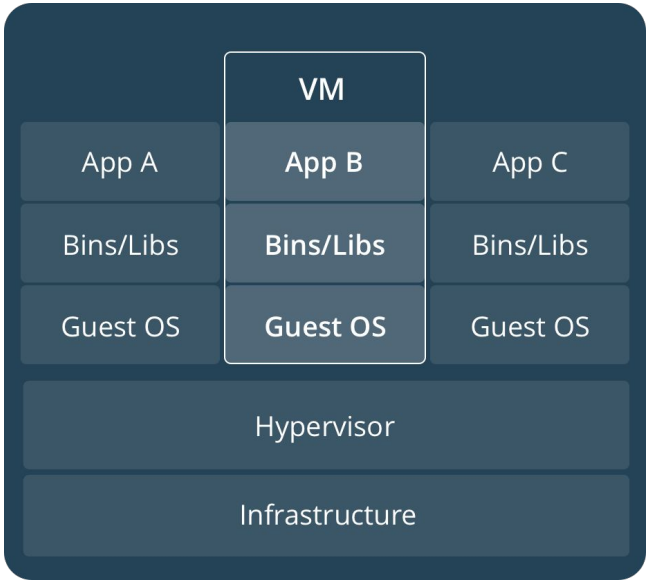
Container promises:

- Focus on services
  - Simplify how to deploy or install
- Easier updates
  - Continuous deployment
- Portable applications
  - Hardware and cloud agnostic
- Open platform
  - Vibrant ecosystem

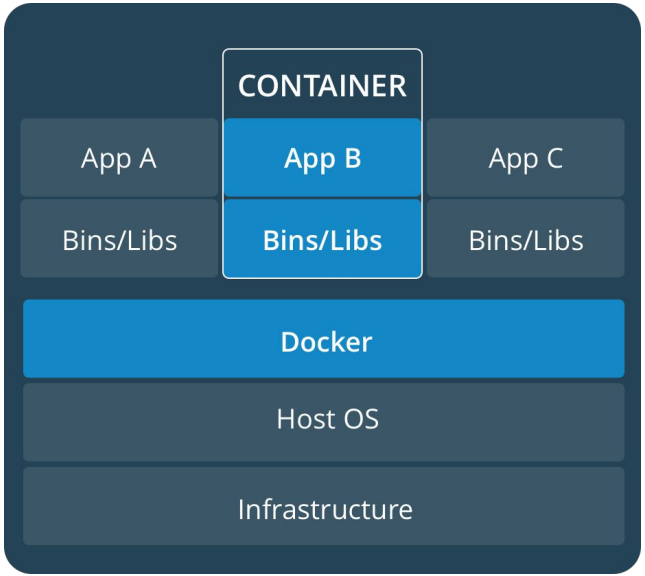
So, what is a container?

"A container image is a lightweight, stand-alone, executable package of a piece of software" .- Docker

"Containers are like ultralight virtual machines" .- Kaggle



vs.



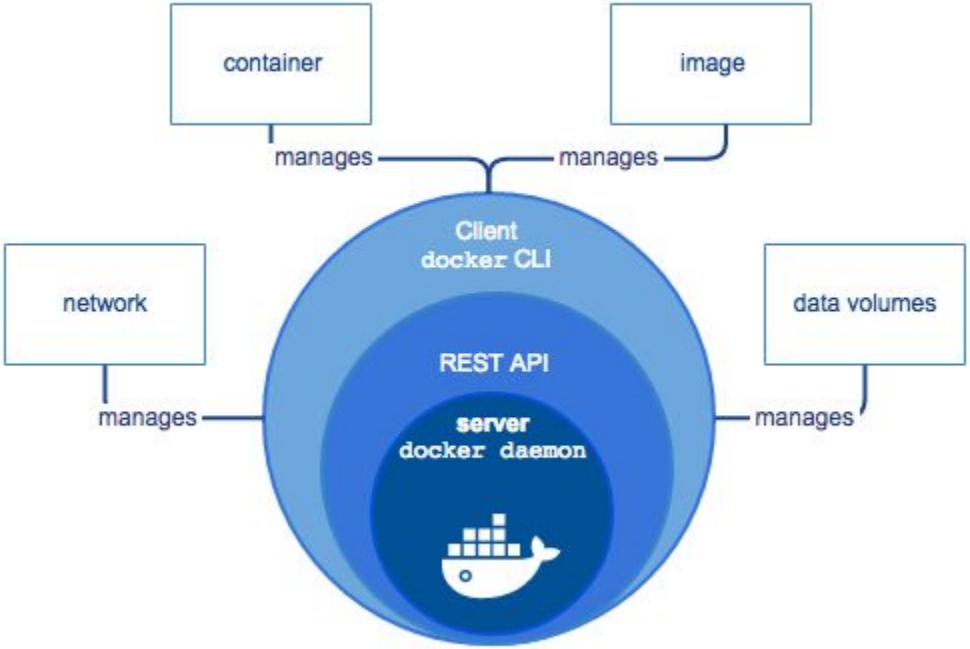
Why Docker?

- Made containers popular again:
  - Old idea: in Unix, chroot 1982. In IBM mainframes 1960s
- Multi-platform
  - Works on Linux, Mac and Windows
- Community and Enterprise support
- Leader ecosystem
  - Lot of vendors using docker (open and commercial)

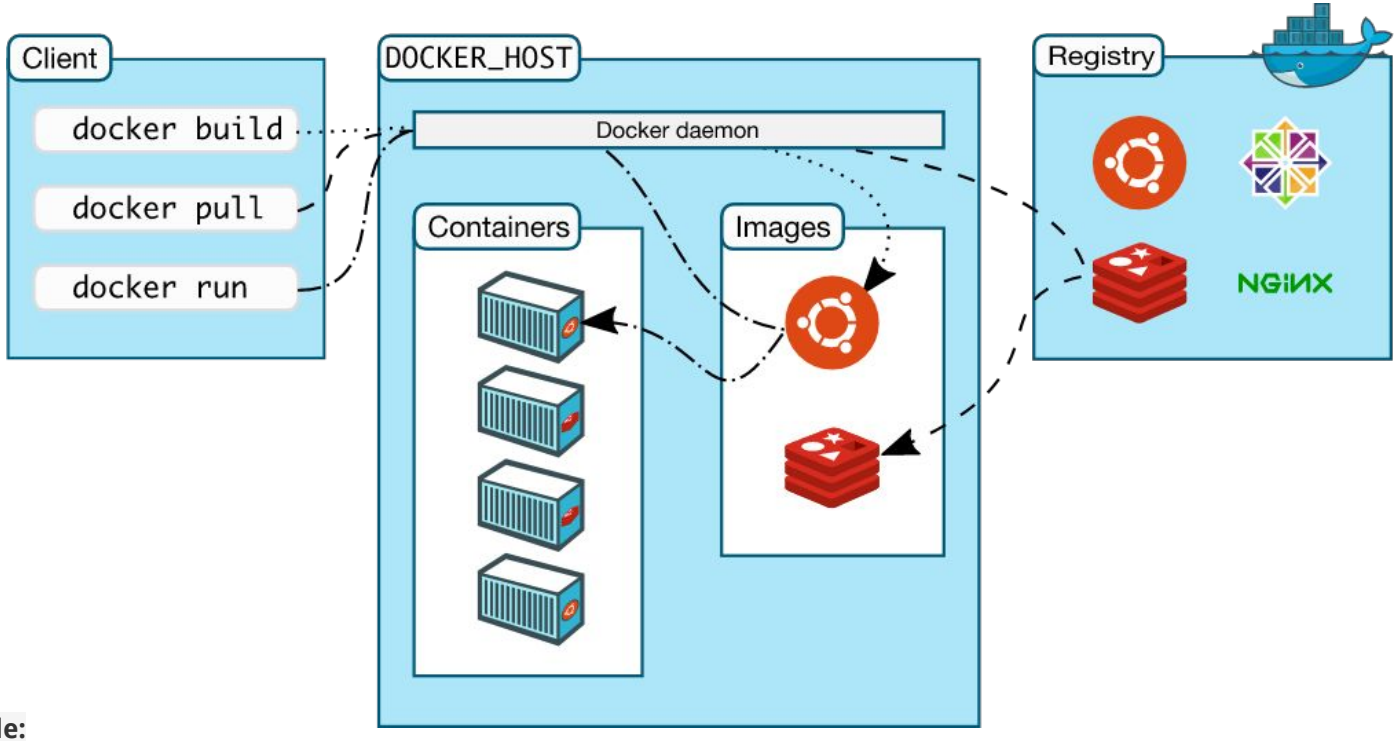
Other options:

- on Linux: snapcraft, AppImage
- on Cloud: Amazon AWS

How Docker works



Docker lifecycle



Example:

```
docker run -i -t ubuntu /bin/bash
```

Example Data Science stack: Jupyter

jupyter.org

python

ANACONDA®

NumPy

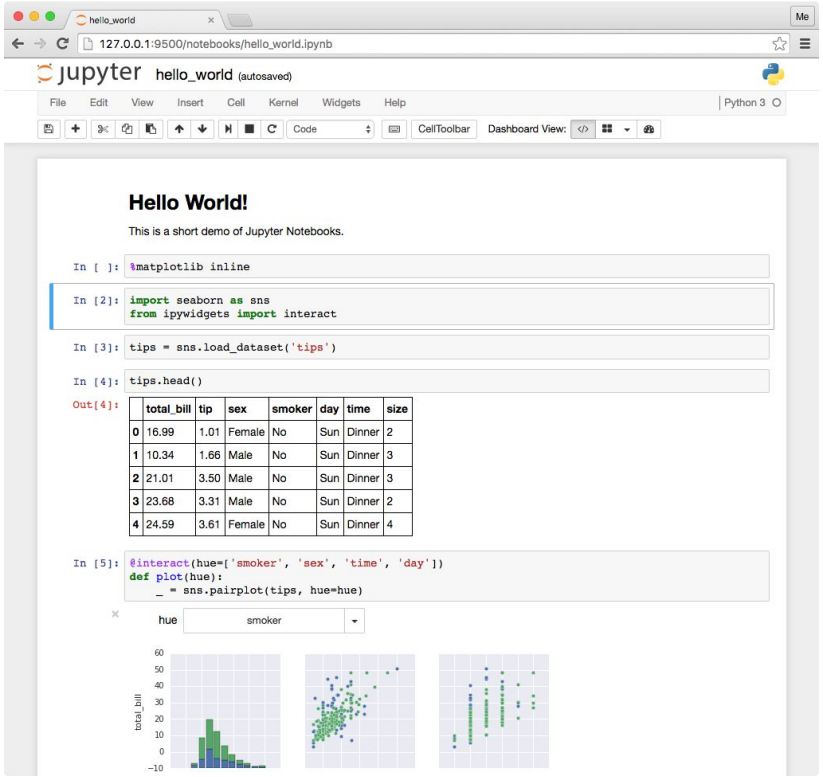
SciPy

matplotlib

IP[y]: IPython
Interactive Computing

pandas

scikit learn



Simpler: run Jupyter + Scipy stack

```
$ docker run -it --rm -p 8888:8888 jupyter/scipy-notebook
```

- Just wait until image is downloaded
  - Open browser using the url showed
  - Create notebook and run using:
    - Jupyter Notebook 5.2.x
    - Conda Python 3.x environment
    - Scipy Stack pre-installed: pandas, matplotlib, scipy, seaborn, scikit-learn, scikit-image, sympy, cython, patsy, statsmodel, cloudpickle, dill, numba, bokeh, vincent, beautifulsoup, xlrd

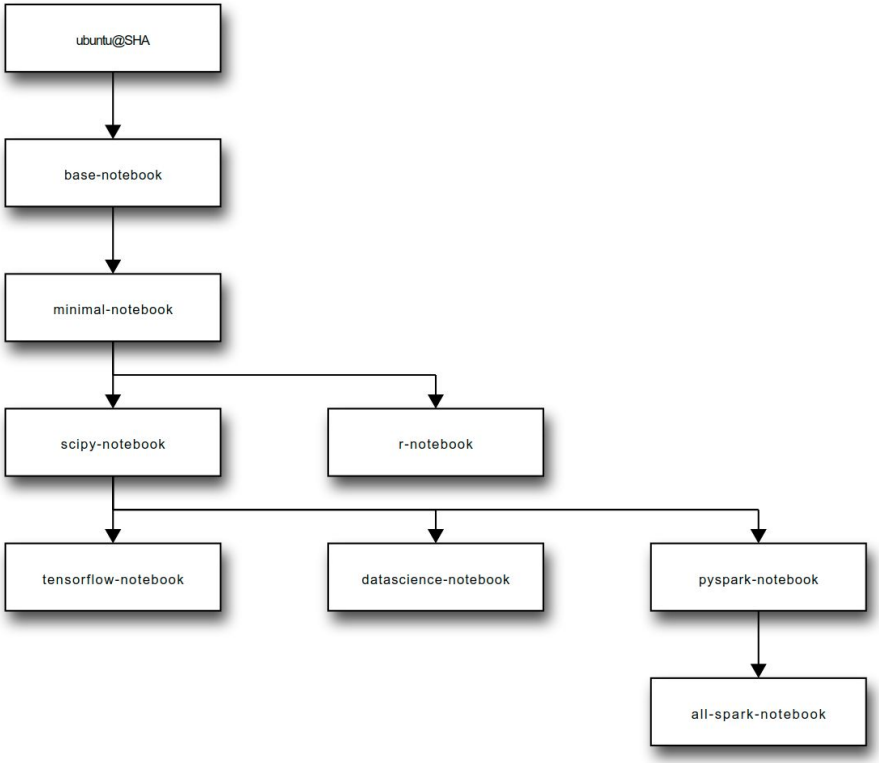
Simpler: add Tensorflow + Keras

```
$ docker run -it --rm -p 8888:8888 jupyter/tensorflow-notebook
```

- Just wait until differential image is downloaded
  - Open browser using the url showed
  - Create notebook and run using:
    - Everything on scipy-notebook
    - Tensorflow
    - Keras
- How to update the image? Just pull and then run

```
$ docker pull jupyter/tensorflow-notebook
```

Simpler: more options for Jupyter



jupyter/docker-stacks

- r-notebook
  - R language with commonly used packages
- datascience-notebook
  - combo with Python, R and Julia
- pyspark-notebook
  - includes a Spark on local mode
  - enable connection with a Mesos cluster
- all-spark-notebook
  - adds R and Scala language and spark packages

Going deeper

- --p 8888:8888 :
  - open on localhost:8888 port 8888 of container
  - so you can run 2 containers using the same local port, ex. -p 8889:8888
- --rm :
  - everything created by container is erased when it stops
  - you have to upload and download as an desktop app, or ...
- -v $PWD:/home/jovyan/work -e NB_UID=`id -u`
  - mounts current directory on work folder
  - everything on that folder is saved after execution
  - Container directory specific on images, ie. this exactly works only on jupyter images
- `docker run -it --rm jupyter/scipy-notebook start.sh ipython`
  - You can also open ipython instead of jupyter (also jupyter image specific)

Personalize images

Interactive way:

- Run container:
  - `$ docker run -it --rm -e GRANT_SUDO=yes --user root --name pydatasl jupyter/scipy-notebook start.sh bash`
- Modify container: Install packages as usual, using conda or pip
  - `jovyan@2faa89dbf8d3:~$ conda install pymc3`
- Save new image
  - `$ docker commit pydatasl gmiretti/pydatasl:handmade`
  - `sha256:924e5971fe97379b1951fdf9f2cb48c2f5a9461ae5fd12ea22a475d9127109e1`
  - `$ docker images`
  - `REPOSITORY  TAG  ID  CREATED  SIZE`
  - `gmiretti/pydatasl  handmade  924e5971fe97  20 seconds ago  4.5GB`
- Close initial container
- Open container from new image
  - `$ docker run -it --rm gmiretti/pydatasl:handmade start.sh bash`
  - `jovyan@a8b3a5a985f0:~$ python -c "import pymc3; print(pymc3.__version__)"`
    `3.2`

¡Not recommended!

Personalize images

Recommended: Images as Code way

- Create Dockerfile
  - Simple, only 17 instructions available
- Build

```
$ docker build -t keras-notebook:2.0 Dockerfile
```

- Run

```
$ docker run --rm keras-notebook:2.0
```

```
# Copyright (c) Jupyter Development Team.
# Distributed under the terms of the Modified BSD License.
FROM jupyter/scipy-notebook

LABEL maintainer="Jupyter Project <jupyter@googlegroups.com>"

# Install Tensorflow
RUN conda install --quiet --yes \
   'tensorflow=1.3*' \
   'keras=2.0*' && \
   conda clean -tipsy && \
   fix-permissions $CONDA_DIR
```

Real example taken from Jupyter official stacks:

https://github.com/jupyter/docker-stacks/blob/master/tensorflow-notebook/Dockerfile

PyDataSL (unofficial) tutorial images

Just pull and run:

```
$ docker run -ti --rm -p 8888:8888 -v $PWD:/home/jovyan/work \
 -e NB_UID=`id -u` gmiretti/pydatasl-tutorials-cpu-notebook
```

Contents:

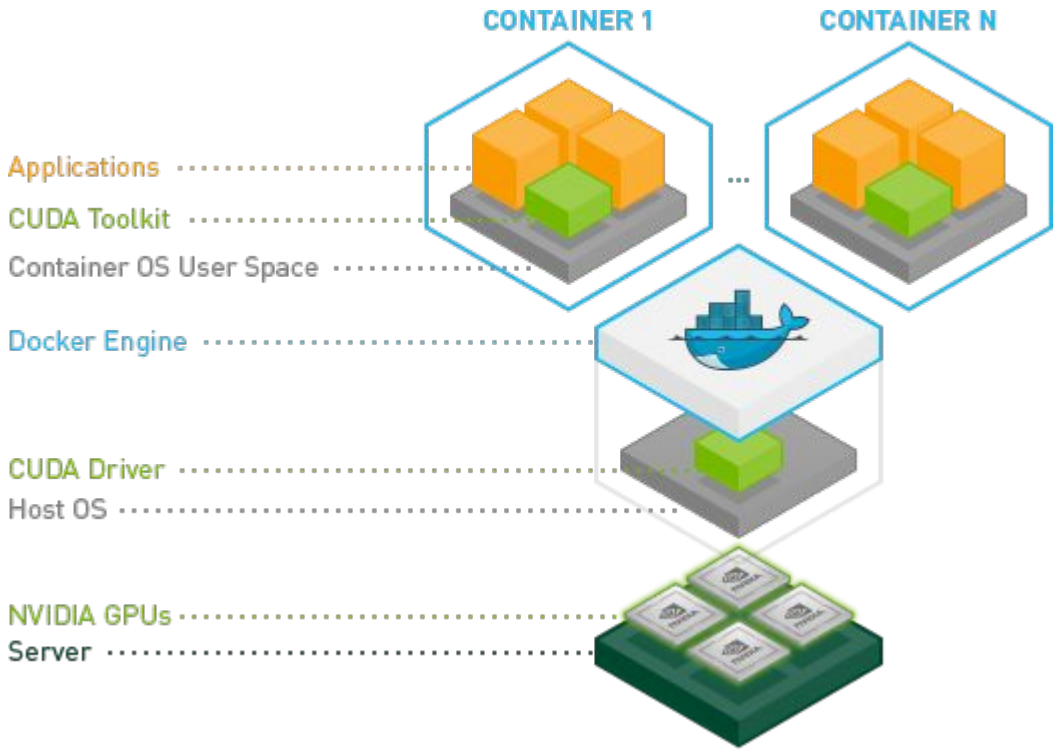
- Jupyter
- Python + Scipy stack
- Tensorflow 1.3 + Tensorboard
- Keras 2.0 + Quiver
- Pymc3 + Theano

```
FROM jupyter/tensorflow-notebook

# Install additional packages for PyDataSL 2017 tutorials
RUN conda install --quiet --yes 'pymc3' 'theano' 'mkl-service' && \
   conda install --quiet --yes -c anaconda 'quiver_engine' && \
   conda clean -tipsy && \
   fix-permissions $CONDA_DIR
```

Code in https://github.com/gmiretti/pydatasl-stacks/

nvidia-docker



nvidia-docker

- Works only on Linux
  - https://github.com/NVIDIA/nvidia-docker#quick-start
  - but could be a server cluster
- Works as expected
  - `nvidia-docker run --rm nvidia/cuda nvidia-smi`
  - `nvidia-docker run --rm nvidia/cuda:6.5-devel nvidia-smi`
- Images available
  - https://hub.docker.com/r/nvidia/cuda/
  - https://github.com/NVIDIA/nvidia-docker/wiki/Third-party
    - Official: Tensorflow, Caffe, CNTK, Keras
    - Unofficial: Theano

Other official stacks

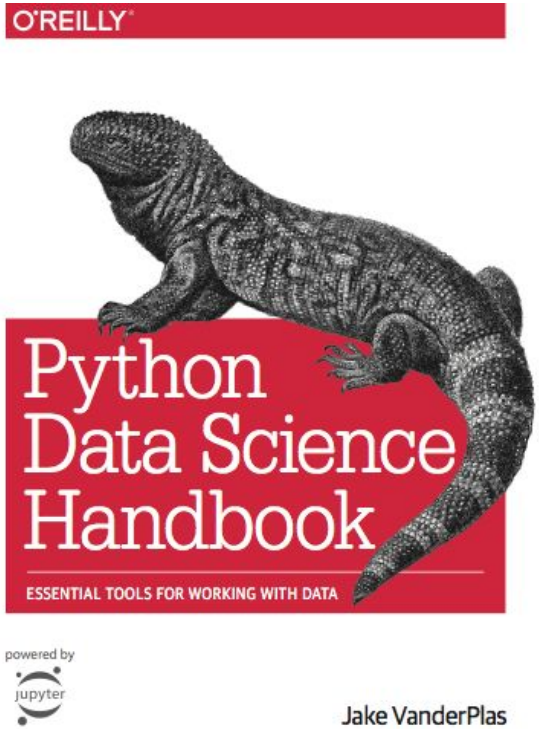
- Anaconda
- Apache Zeppelin: notebooks using Scala, R and Python with Spark
- Kaggle images for Python, R and Julia: run kernels on your machine
- Hortonworks Data Platfom and Data Flow
- Cloudera Data Science Workbench
- Apache Superset
- Metabase

Reproducible and Shareable

1. Build your own image
   with your code and data inside
2. Push your image to a registry
3. On any machine with docker, pull and run

Example: Python Data Science Handbook

- Great introductory and interactive book!
- Notebooks and data available on Github https://github.com/jakevdp/PythonDataScienceHandbook
- But installing requirements to run locally isn't so easy, also breaks frequently
- Docker can help us?



Example: Python Data Science Handbook

1. Download book

   Just fork and clone

2. Create Dockerfile

   ADD content

```
FROM jupyter/minimal-notebook

ADD requirements.txt /home/$NB_USER/
RUN conda install --yes --file /home/$NB_USER/requirements.txt && \
   conda clean -tipsy && \
   fix-permissions $CONDA_DIR

ADD notebooks /home/$NB_USER/
```

3. Build image

```
$ docker build -t gmiretti/python-data-science-handbook .
```

4. Push image

```
$ docker push gmiretti/python-data-science-handbook
```

5. Run image everywhere

```
$ docker run --rm -p 8888:8888 gmiretti/python-data-science-handbook
```

- code: github.com/gmiretti/PythonDataScienceHandbook
- image: hub.docker.com/r/gmiretti/python-data-science-handbook

Conclusions

- Docker keeps its promises
  - Data science stacks are **simpler** to use
    - Most of them in one command
    - Also simpler to update
  - Experiments can be **shared** and **reproduced**
    - Not only by me on different machines, also by others
    - Also they are controlled and isolated
- Docker is here to stay
  - It is the building block of Cloud computing
- Docker is a great addition to any data science toolkit

PyData

¿Questions?

http://about.me/gmiretti