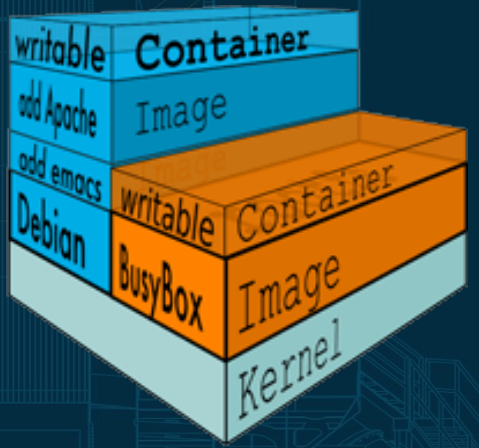
LSST software stack and deployment on other architectures

William O'Mullane for Andy Connolly with material from Owen Boberg

Containers and Docker

- Packaged piece of software with complete file system it needs to run!
- Portable unit - standard shape and connectors
- Boots in fraction of a second
- Lightweight - 10-100 times containers per machine w.r.t to VMs.

https://docker.com



Docker leading the pack now for containers - others exist ..
Solaris Zones FreeBSD Jails

Docker Images

- Docker Image:
  - Built from an easily readable/writeable Dockerfile
    - Install commands you would run in your terminal (apt-get, yum,pip,conda) can almost be directly copied into a Dockerfile.
    - This includes the instructions for LSST pipelines.
  - Or pulled from the Docker hub repository
    - Images on Docker hub can, and often do, serve as the base image for applications you build.
- Docker allows you to run standalone applications in a sandbox environment
  - Docker Container:
    - A running instance of a Docker image
    - You can think of it as a lightweight VM
    - Only uses the resources you give it

Docker - Pros

- Completely isolated environment
  - Need more than one version of python, just make different images/containers.
  - Truly isolate the LSST stack from your native environment.
- Get access to software not available on your host OS.
  - Example: Macs can not currently run OpSim
  - A Mac running OpSim through Docker can, however.
- Reproducibility:
  - Easily share the exact environment used to produce a result
  - Avoid the statement, “Well it worked on my machine”.
- Collaboration:
  - Docker Hub makes it easy to share images
  - Quickly get people using LSST software while at workshops
  - No time lost with people trying to update their version of the stack
  - Just install Docker and pull the image for the software

Docker Cons

- There is a small learning curve to Docker
  - Have to learn how to mount volumes so you do not lose data when a container is removed
- Lose software changes to a container if they are not committed
- Security:
  - Careful steps need to be taken when running Docker on a shared machine
  - Only privileged users should have access to Docker commands
  - See: https://github.com/docker/labs/tree/master/security
- Some high performance computers centers (HPCs) will not install Docker.
- Alternatives: shifter (NCSA), singularity

For LSST DM Stack .. .

The major factors are reliability and reproducibility

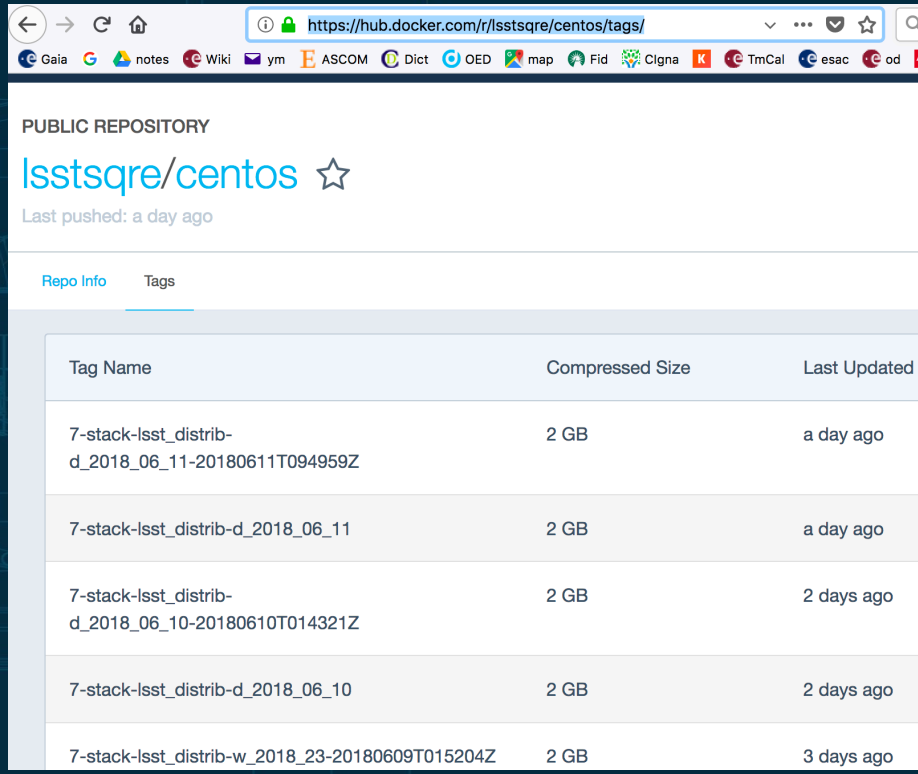
- Code from build and test environment
- Directly to deployment
- Including in the JupyterHub (used in the tutorial)
- With some work other systems like Spark/Dask can be accommodated

You can get the stack running on a machine with docker using

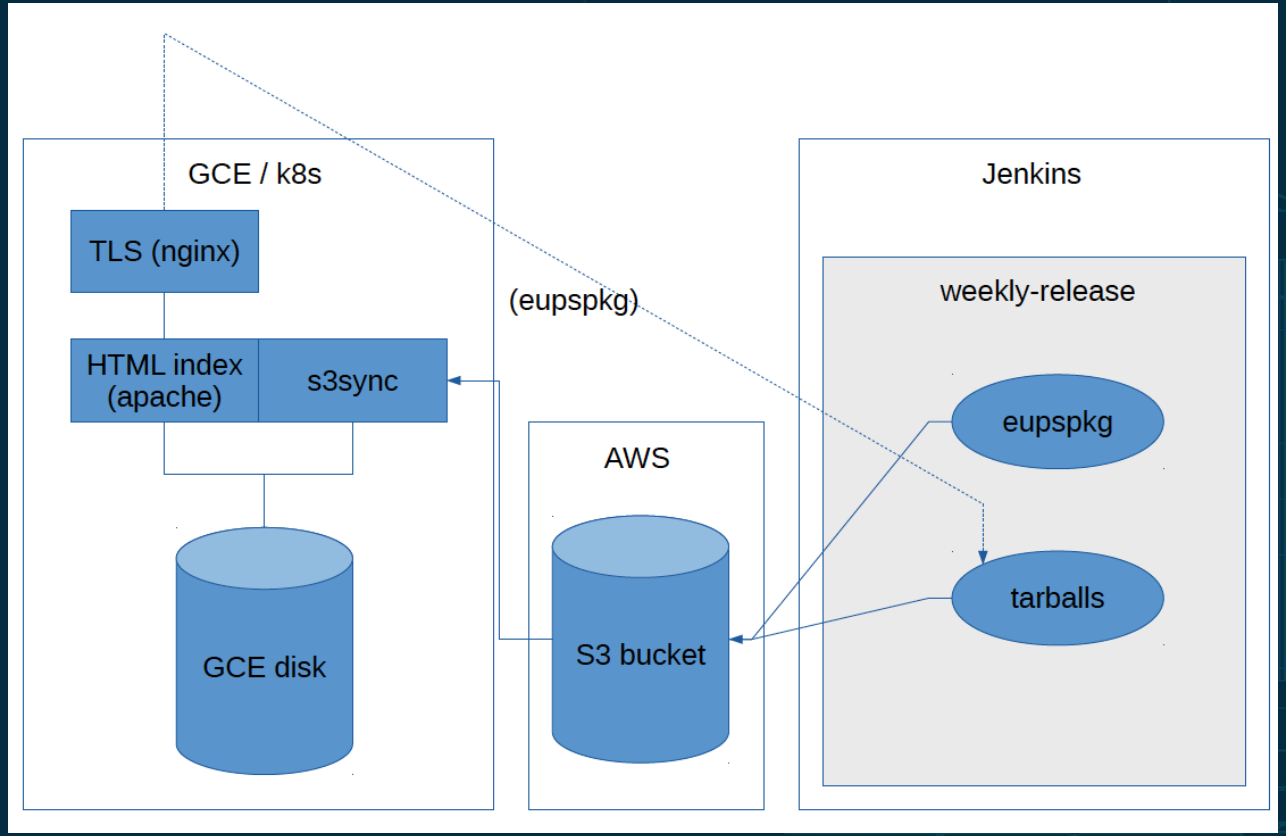
Any Stack release with Docker

https://hub.docker.com/r/lsstsqre/centos/tags/

```
>docker run -ti lsstsqre/centos:7-stack-lsst_distrib-v15_0

Unable to find image 'lsstsqre/centos:7-stack-lsst_distrib-v15_0' locally

>source /opt/lsst/software/stack/loadLSST.bash
>setup lsst_distrib …….
```



LSST DM Stack builds



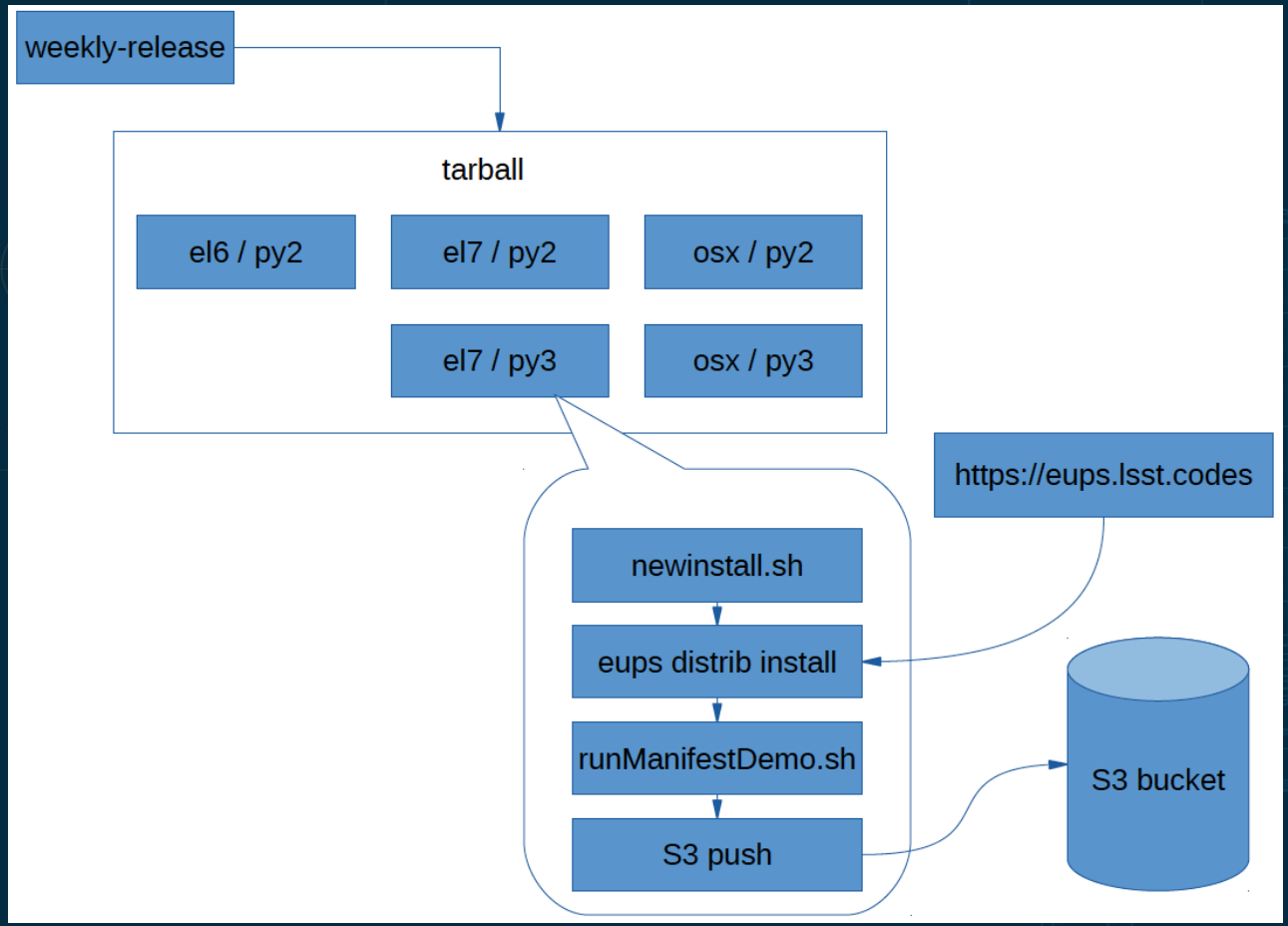
GCE Google Compute Engine -yes we pay for it.

TLS Transport Layer Security – handles the SSL

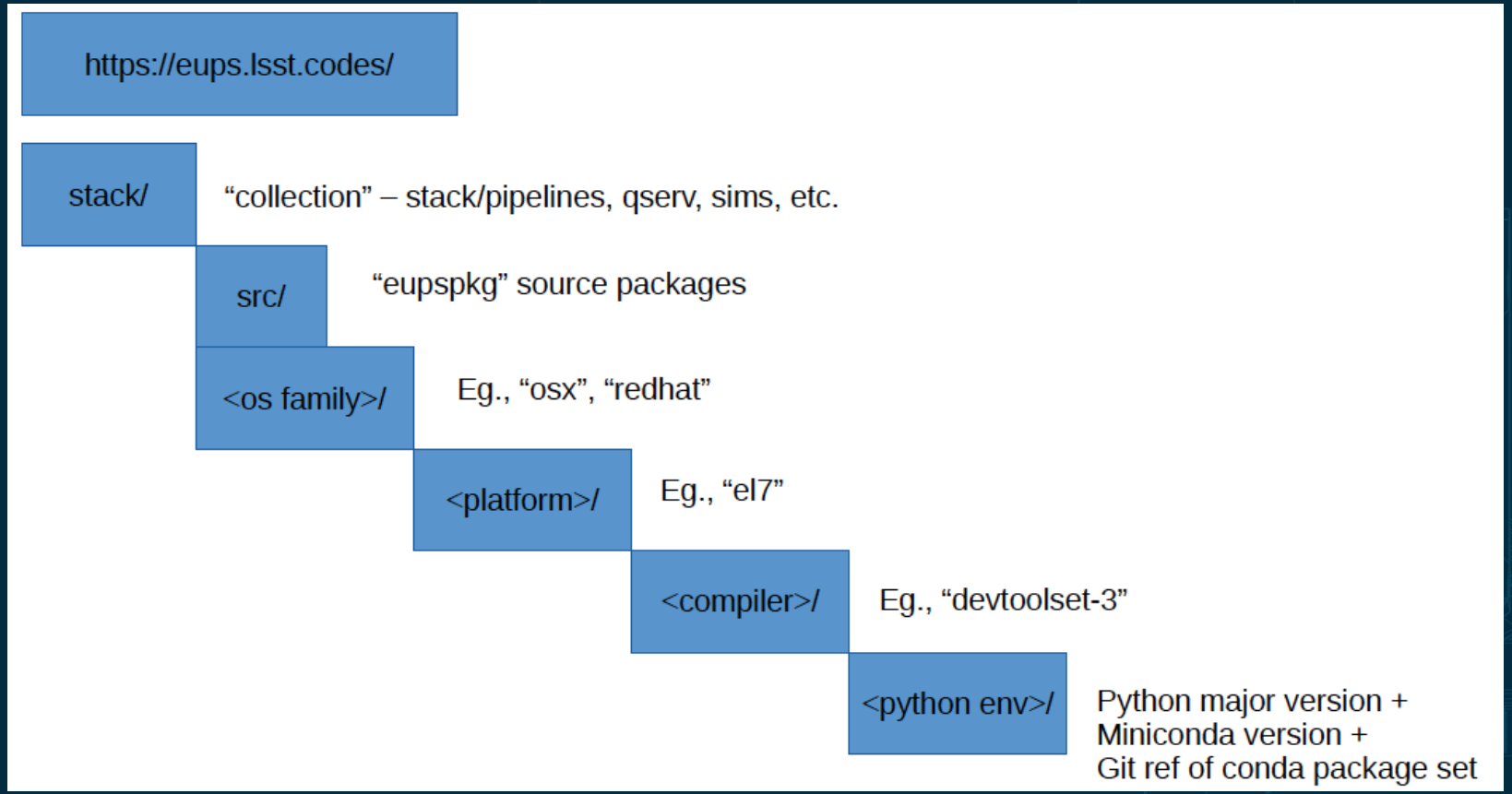
S3sync sync a filesystem to an S3 bucket

Jenkins master on AWS, the agents on Google, and we stuff the results in an AWS S3 bucket

How a “release” is made



File structure for releases



Dockerfile

The commands are easy to read and follow

Commands in PINK are Docker specific commands.

The base image is pulled FROM Docker hub

Installing additional software on top of the base image is done here using apt-get and pip.

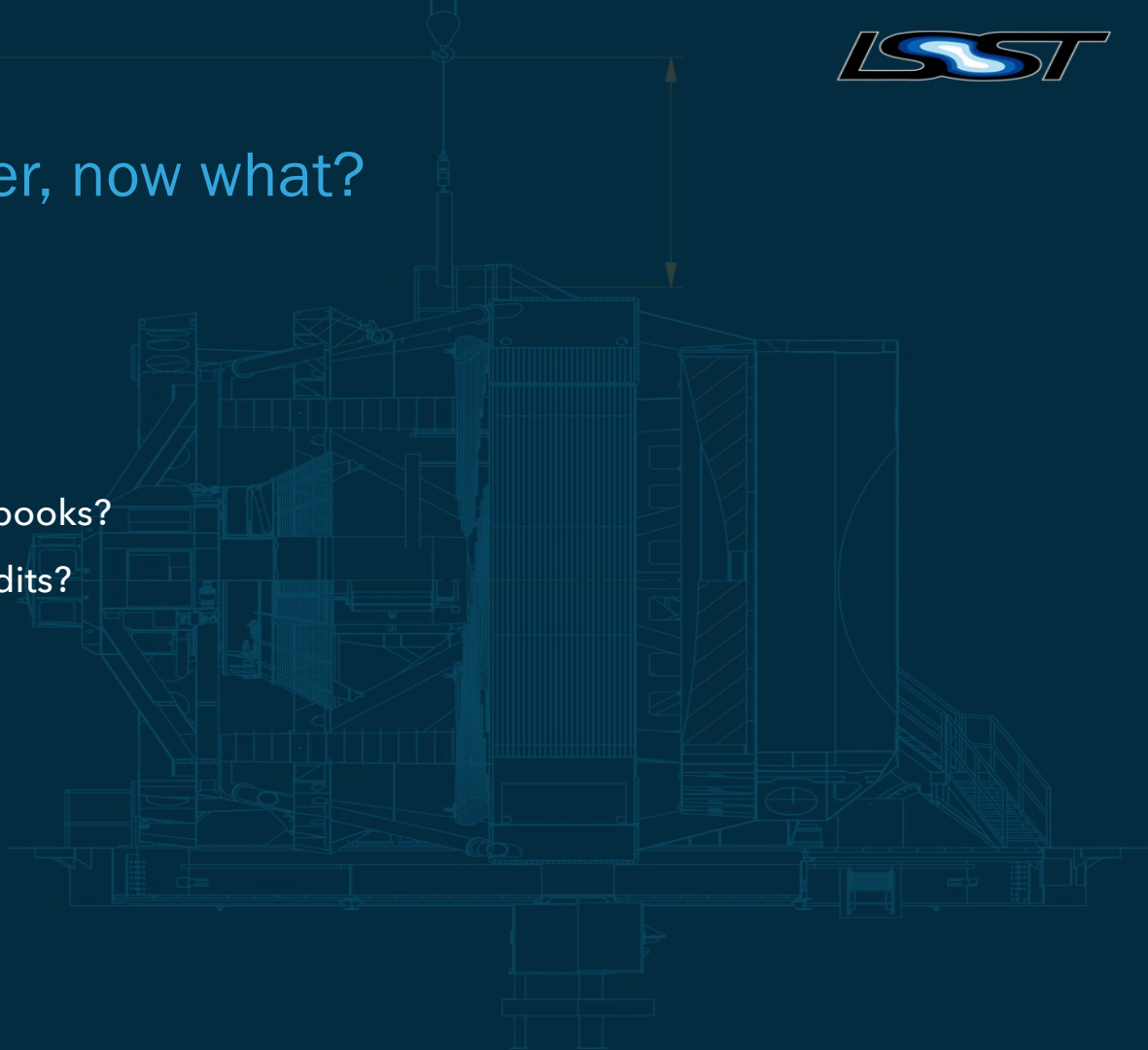
```
FROM jupyter/datascience-notebook
LABEL maintainer "boberg37@gmail.com"
ENV REFRESHED_AT 2018-01-23

USER root

RUN apt-get update \
    && apt-get install -y \
    texlive-full \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

USER jovyan

WORKDIR /home/jovyan/work
ADD requirements.txt .
RUN pip install -r requirements.txt
```

https://hub.docker.com/r/oboberg/astroml/

Docker container

Im in a Docker container, now what?

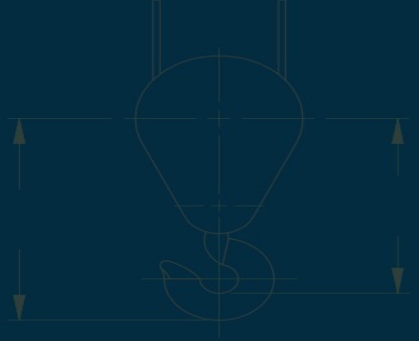
- You might have these questions:
  - How do I get to a command line?
  - Where do I write my code?
  - Where do I put my data?
  - Can I use jupyter lab and/or notebooks?
  - How do I save code output and edits?



Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
    -v ${PWD}/results:/home/maf/docmaf/maf_local \
    -v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
    -p 8888:8888 \
    oboberg/maf:180525
```



- How do I get to a command line?
- Where do I write my code?
- Where do I put my data?
- Can I use jupyter lab and/or notebooks?
- How do I save code output and edits?

Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
-v ${PWD}/results:/home/maf/docmaf/maf_local \
-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
-p 8888:8888 \
oboberg/maf:180525
```

- How do I get to a command line?
  - `docker run -it`
  - Starts the container in an interactive mode.
  - Gives you access to shell inside of the container.

Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
-v ${PWD}/results:/home/maf/docmaf/maf_local \
-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
-p 8888:8888 \
oboberg/maf:180525
```

- Where do I put my data?
  - `-v ${PWD}/results:/home/maf/docmaf/maf_local`
  - This option mounts ${PWD}/results on your computer inside the container at /home/maf/docmaf/maf_local
  - Any new data put in ${PWD}/results on your computer will be available to use in the Docker container

Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
-v ${PWD}/results:/home/maf/docmaf/maf_local \
-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
-p 8888:8888 \
oboberg/maf:180525
```

- Where do I write my code?
  - `-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos`
  - Mount `${PWD}/my_repos` on your computer inside the container at the path `/home/maf/docmaf/lsst_repos`
  - Edit any code in `${PWD}/my_repos` as you normally would on a local editor and the updated code will automatically available in the container
  - Switching git branches in `${PWD}/my_repos` will also switch branches in the container

Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

- A note about LSST software
  - -v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos
  - Once in the container you can eups declare and setup packages to run off of a GitHub repos.
  - Example:
    - sims_maf is cloned into ${PWD}/my_repos on my local host
      - Once in the container go to /home/maf/docmaf/lsst_repos/sims_maf
      - eups declare sims_maf -r . -t $USER ($USER is docmaf in this container)
      - setup sims_maf -t $USER
      - scons
  - The container will now be running MAF off of the GitHub repo that is stored (and edited) locally, but mounted in the container.
  - See https://pipelines.lsst.io/install/docker.html#how-to-develop-packages-inside-docker-containers for more info.

Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
-v ${PWD}/results:/home/maf/docmaf/maf_local \
-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
-p 8888:8888 \
oboberg/maf:180525
```

- Can I use jupyter lab and/or notebooks?
  - -p 8888:8888
  - Read as host_port:container_port
  - Maps port 8888 of your host to port 8888 in the container
  - Command in container:
    - jupyter lab —ip=0.0.0.0 —no-browser
  - Copy and paste the url you are given into a local browser and start making notebooks.
  - This does of course require jupyter to be installed in the container, which is the case for oboberg/maf:180525 image.

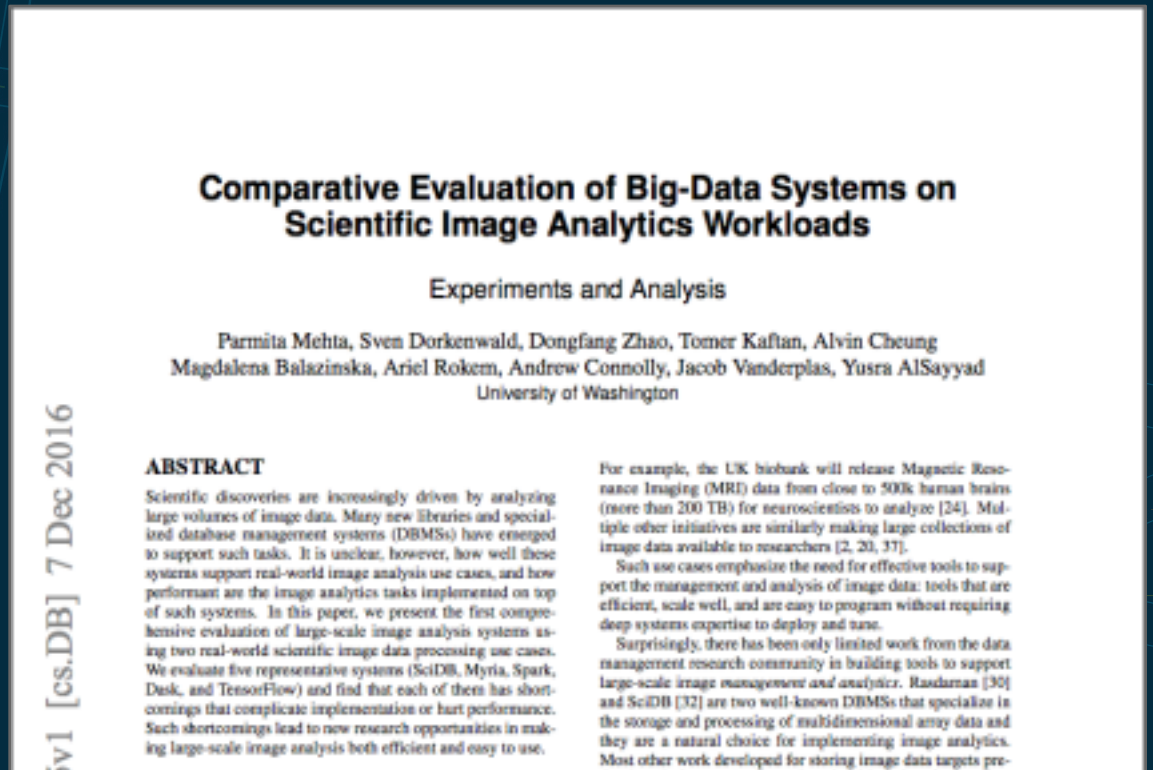
Docker container (MAF example)

Running the Metric Analysis Framework (MAF)

```
docker run -it \
-v ${PWD}/results:/home/maf/docmaf/maf_local \
-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos \
-p 8888:8888 \
oboberg/maf:180525
```

- How do I save code output and edits?
  - `-v ${PWD}/results:/home/maf/docmaf/maf_local`
  - `-v ${PWD}/my_repos:/home/maf/docmaf/lsst_repos`
  - When in the container, **anything created, edited, or removed** in /home/maf/docmaf/maf_local or /home/maf/docmaf/lsst_repos will have the same effect for content in **${PWD}/results** and **${PWD}/my_repos**, respectively.
  - Be careful what you mount!
  - There are security features to stop you from mounting your root directory

Paper Goal: evaluate existing Big Data systems on **real-world** scientific image analysis workflows & point the way forward for database & systems researchers.



Comparative Evaluation of Big-Data Systems on Scientific Image Analytics Workloads

Experiments and Analysis

Parmita Mehta, Sven Dorkenwald, Dongfang Zhao, Tomer Kaftan, Alvin Cheung
Magdalena Balazinska, Ariel Rokem, Andrew Connolly, Jacob Vanderplas, Yusra AlSayyad
University of Washington

ABSTRACT

Scientific discoveries are increasingly driven by analyzing large volumes of image data. Many new libraries and specialized database management systems (DBMSs) have emerged to support such tasks. It is unclear, however, how well these systems support real-world image analysis use cases, and how performant are the image analytics tasks implemented on top of such systems. In this paper, we present the first comprehensive evaluation of large-scale image analysis systems using two real-world scientific image data processing use cases. We evaluate five representative systems (SciDB, Myria, Spark, Dask, and TensorFlow) and find that each of them has shortcomings that complicate implementation or hurt performance. Such shortcomings lead to new research opportunities in making large-scale image analysis both efficient and easy to use.

For example, the UK biobank will release Magnetic Resonance Imaging (MRI) data from close to 500k human brains (more than 200 TB) for neuroscientists to analyze [24]. Multiple other initiatives are similarly making large collections of image data available to researchers [2, 20, 37].

Such use cases emphasize the need for effective tools to support the management and analysis of image data: tools that are efficient, scale well, and are easy to program without requiring deep systems expertise to deploy and tune.

Surprisingly, there has been only limited work from the data management research community in building tools to support large-scale image *management and analytics*. Rasdaman [30] and SciDB [32] are two well-known DBMSs that specialize in the storage and processing of multidimensional array data and they are a natural choice for implementing image analytics. Most other work developed for storing image data targets pre-

https://arxiv.org/abs/1612.02485