

Working with LSST Science Pipeline Containers

David Shupe, Caltech/IPAC

IPAC Containerization Workshop, 12 September 2019

LSST Science Pipelines are not installable via the usual methods

- Development started in 2004
 - C++ wrapped with SWIG (now pybind11)
 - 100,000+ lines of code
 - Custom package manager (eups)
- Build from source is difficult
 - 2.5 hours on my laptop
 - Library incompatibilities, compilers, etc
- => LSST Science Platform

I really need LSST Science Pipelines software on my laptop for development

`lsst.afw.display`: image display package with different backends

`lsst.display.firefly`: Firefly backend

`firefly_client`: Python interface to Firefly

Firefly – server and browser client (see Trey's talk)

`jupyter_firefly_extensions`: Jupyterlab extension for Firefly

LSST makes two flavors of builds, but neither is quite what I need

- Nightly builds at <https://hub.docker.com/u/lsstsqre>
 - lsstsqre/centos: Science Pipelines complete distribution
 - lsstsqre/sciplat-lab: add Jupyterlab, many customizations for Science Platform
- Solution: add Jupyterlab + Firefly-related packages to Science Pipelines

Containerized development workflow from <https://pipelines.lsst.io/install/docker.html>

- On the host system:
 - Run your own code editors
 - Run **git**
- In container shells:
 - Set up packages (**setup**)
 - Compile code (**scons**)
 - Run the Pipelines (**processCcd.py**)

Software and its configuration in container, all else on host / laptop

On host

- Code repositories
- Jupyter notebooks
- Sample data
- Web browser

Inside container only

- Science Pipelines (lsst-distrib)
- Conda package manager
- Jupyterlab, nodejs, Jupyterlab extensions

Some additional installations and configurations enable Firefly functionality

- See pipelines.isst.io and search “Docker” to find Firefly standalone doc
 - In container shell:
 - Set up the pipelines
 - *Install Jupyterlab and nodejs*
 - *Install and enable JLab extensions*
 - Set Firefly environment variables
 - Start Jupyterlab
- One-time only!**

On host: run a Science Pipelines container for the latest release v18.1.0

- cd to your base directory, then

```
docker run -itd -p 9888:9888 \  
-p 9889:9889 -p 9890:9890 \  
-v `pwd`:/home/lsst/mnt \  
--name lsst_v18_1_0 \  
lsstsgre/centos:7-stack-lsst_distrib-v18_1_0
```

Detached; 3 ports mapped; current directory is mounted; container is named

Start a container shell and set up LSST

- Start a container shell

```
docker exec -it lsst_v18_0_1 /bin/bash
```

- In this container shell, set up:

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

Now Python, conda are set up

One time only (per container), install Jupyterlab and extensions

- Still in the container shell, use conda

```
conda install jupyterlab nodejs ipywidgets
```

- In container, install extensions

```
jupyter labextension install \  
    @jupyter-widgets/jupyterlab-manager  
jupyter labextension install jupyter_firefly_extensions  
pip install jupyter_firefly_extensions  
jupyter serverextension enable --py jupyter_firefly_extensions
```

After `lsst_distrib` setup, set environment variables specifying Firefly server

- Can run Firefly container locally

```
docker run -p 8090:8080 -e "MAX_JVM_SIZE=8G" \
  --rm ipac/firefly:lsst-dev >& my.log &
```

- Special care is needed when referring to the IP address of the host (*localhost*)
- On Docker for Mac: *host.docker.internal*
- Better: use laptop's IP address

```
export FIREFLY_URL=http://10.8.20.113:8090/firefly
export FIREFLY_HTML=slate.html
```

Start Jupyterlab inside the container shell

- Start from mounted directory

```
cd /home/lsst/mnt
```

```
jupyter lab --ip 0.0.0.0 --port 9888
```

- Copy and paste Jupyterlab URL into a web browser on your laptop:

To access the notebook, open this file in a browser:

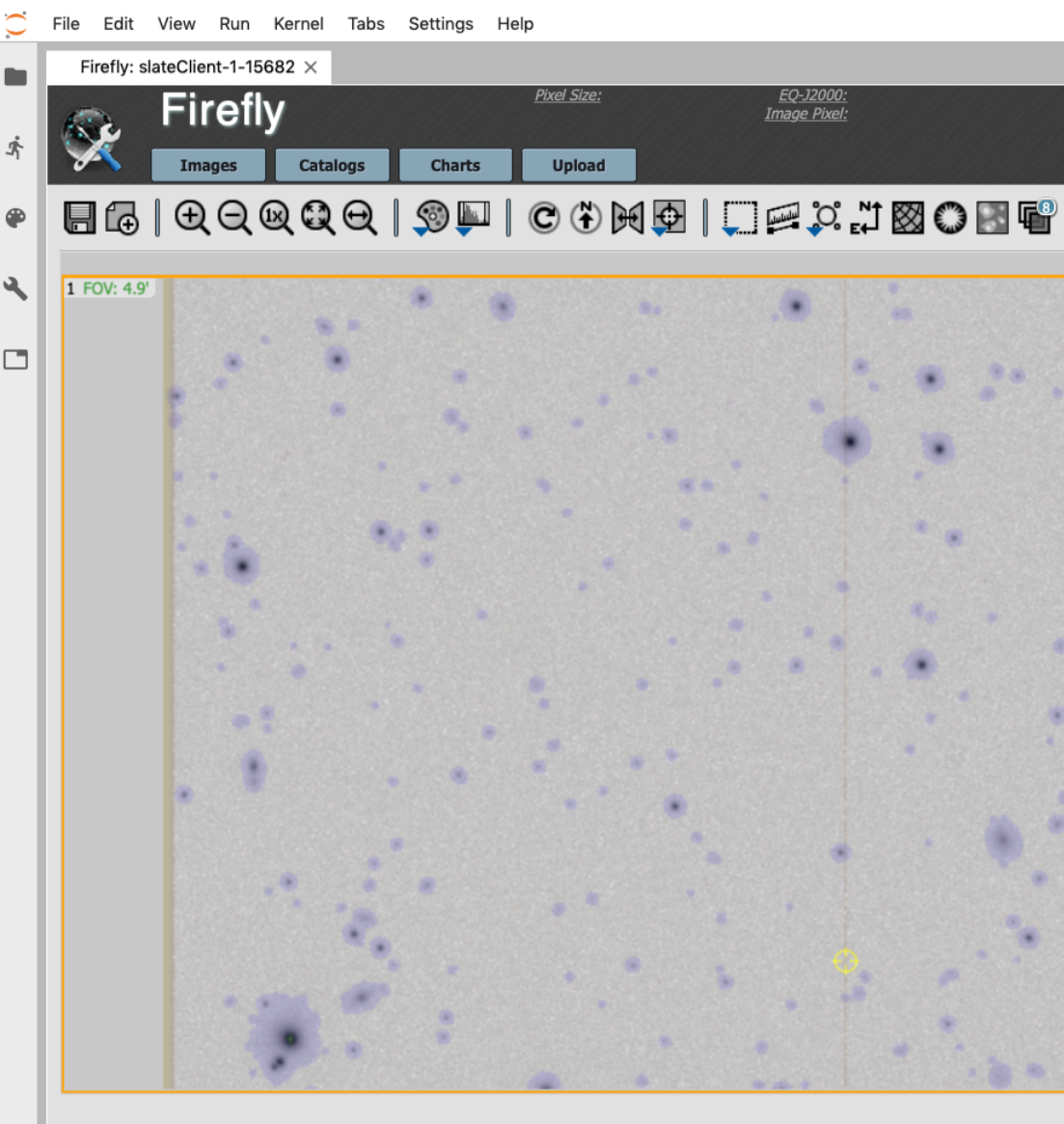
```
file:///home/lsst/.local/share/jupyter/runtime/nbserver-342-open.html
```

Or copy and paste one of these URLs:

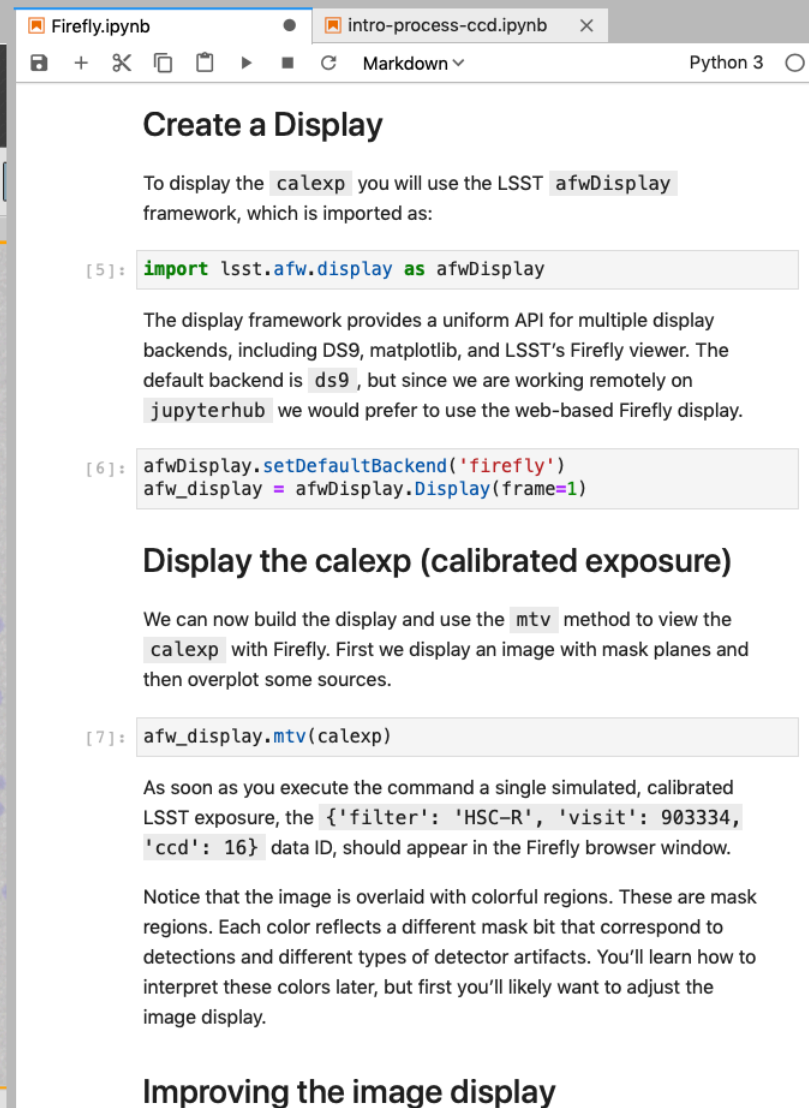
```
http://e191d2f91116:9888/?token=f6fcb67e2cad5c961771aa1d8543798e22de343fec9def3d
```

```
or http://127.0.0.1:9888/?token=f6fcb67e2cad5c961771aa1d8543798e22de343fec9def3d
```

Run notebooks and display images



The screenshot shows the Firefly web interface. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Tabs', 'Settings', and 'Help'. Below the menu, the browser tab is labeled 'Firefly: slateClient-1-15682 X'. The main interface has a dark header with the 'Firefly' logo and 'Pixel Size: EQ-J2000: Image Pixel:'. Below the header are four tabs: 'Images', 'Catalogs', 'Charts', and 'Upload'. A toolbar with various icons for zooming, panning, and other image manipulation functions is visible. The main content area displays a large astronomical image of a galaxy field. A vertical yellow line is drawn across the image, and a yellow circle highlights a specific source. In the top left corner of the image area, it says '1 FOV: 4.9''.



The screenshot shows a Jupyter Notebook with two cells. The first cell is titled 'Create a Display' and contains the following code:

```
[5]: import lsst.afw.display as afwDisplay
```

The second cell is titled 'Display the calexp (calibrated exposure)' and contains the following code:

```
[6]: afwDisplay.setDefaultBackend('firefly')
afw_display = afwDisplay.Display(frame=1)

[7]: afw_display.mtv(calexp)
```

The notebook interface shows the file names 'Firefly.ipynb' and 'intro-process-ccd.ipynb' in the tabs, and 'Python 3' in the bottom right corner.

Create a Display

To display the `calexp` you will use the LSST `afwDisplay` framework, which is imported as:

```
[5]: import lsst.afw.display as afwDisplay
```

The display framework provides a uniform API for multiple display backends, including DS9, matplotlib, and LSST's Firefly viewer. The default backend is `ds9`, but since we are working remotely on `jupyterhub` we would prefer to use the web-based Firefly display.

```
[6]: afwDisplay.setDefaultBackend('firefly')
afw_display = afwDisplay.Display(frame=1)
```

Display the calexp (calibrated exposure)

We can now build the display and use the `mtv` method to view the `calexp` with Firefly. First we display an image with mask planes and then overplot some sources.

```
[7]: afw_display.mtv(calexp)
```

As soon as you execute the command a single simulated, calibrated LSST exposure, the `{'filter': 'HSC-R', 'visit': 903334, 'ccd': 16}` data ID, should appear in the Firefly browser window.

Notice that the image is overlaid with colorful regions. These are mask regions. Each color reflects a different mask bit that correspond to detections and different types of detector artifacts. You'll learn how to interpret these colors later, but first you'll likely want to adjust the image display.

Improving the image display

Key points

- Let someone else (LSST, Dockerhub) handle the difficult software
- You can add, or upgrade, Jupyterlab and extensions, to a pre-existing image
- I could automate most of these steps by making my own Docker image

References and more information

- Documentation

<https://pipelines.lsst.io/install/docker.html>

<https://pipelines.lsst.io/modules/lsst.display.firefly/using-firefly-standalone.html>

- AAS 233 tutorial (ask me for the data):

https://github.com/lsst-sqre/notebook-demo/tree/master/AAS_2019_tutorial

- LSST Stack Club:

<https://github.com/LSSTScienceCollaborations/StackClub>