LSST Software Survey Report

Project Acronym	LUSC-A
Project Title	UK Involvement in the Large Synoptic Survey Telescope
Document Number	LUSC-A-04

Submission date	16/NOV/2017
Version	1.0
Status	Published
Author(s) inc. institutional affiliation	James Perry, University of Edinburgh
Reviewer(s)	George Beckett, University of Edinburgh; Bob Mann, University of Edinburgh

Dissemination level	
Public	*Public*

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	15/NOV/17	Initial draft	James Perry
1.0	16/NOV/17	Draft reviewed and published	James Perry, George Beckett

Table of Contents

VERSION HISTORY **2**

1 EXECUTIVE SUMMARY **4**

2 INTRODUCTION **5**

2.1 PURPOSE 5
2.2 GLOSSARY OF ACRONYMS 5

3 THE PHASE B PREPARATION WORKSHOP **6**

3.1 BACKGROUND 6
3.2 SOFTWARE INSTALLATION 6
3.3 RUNNING JUPYTER NOTEBOOKS WITHIN DOCKER 7
3.4 ADDITIONAL TUTORIAL NOTEBOOKS 8

4 THE LSST SIMULATION SOFTWARE **9**

4.1 BACKGROUND 9
4.2 INSTALLATION 9
4.3 THE MAF TUTORIAL NOTEBOOKS 9
4.4 THE LSST SIMULATIONS DOCKER IMAGE 9
4.5 DEPLOYING DATABASES LOCALLY 10

5 REFERENCES **12**

1 Executive Summary

This report describes my work on the LSST software stack for LSST:UK – surveying what is available, understanding how to deploy and use it, and providing useful resources for UK astronomers.

The work began in the lead up to the Phase B Preparation Workshop in late April 2017. We provided instructions and an easy to install Docker image of the necessary software for participating in the hands-on tutorial session at this workshop. After the workshop we provided useful related information online, including links to other relevant tutorial material.

I then moved onto looking at the LSST simulation software, as this (especially the Metrics Analysis Framework) was identified as being of particular interest to LSST:UK. To speed up and simplify deployment of this software, I created another Docker container, this time including a full installation of the LSST simulation software as well as the base LSST stack. This container has already proved useful to LSST:UK members.

Working with the MAF tutorials provided in the form of Jupyter notebooks, I liaised with one of the authors of these notebooks to get them working without errors on the latest versions of the LSST software. I also deployed some of the databases required by the notebooks on a testbed machine at the Royal Observatory of Edinburgh, avoiding the need for all users to download their own local copies of the databases in SQLite format.

2 Introduction

Although full operation of the LSST is still several years away, a multitude of different software packages have already been created for working with the data produced by the telescope. The aim of the software survey work was to understand the range of software available as well as what might be of particular interest to UK astronomers, and to gain experience of deploying and using the software locally.

2.1 Purpose

This document describes my work in surveying and setting up the LSST software for LSST:UK.

2.2 Glossary of Acronyms

EUPS – Extended Unix Product System

GUI – Graphical User Interface

LSST – Large Synoptic Survey Telescope

LSST:UK – The UK consortium of institutions interested in working with LSST data

MAF – Metrics Analysis Framework

PSF – Point Spread Function

VNC – Virtual Network Computing

3 The Phase B Preparation Workshop

3.1 Background

A workshop was held at the Royal Observatory of Edinburgh in late April 2017, aiming to start the preparation of the science cases for Level 3 deployment in the Phase B proposal (Phase B running from April 2019 to March 2023, spanning LSST commissioning and the start of survey operations). As an initial step towards understanding the LSST software, we decided to run a hands-on tutorial session at this workshop, giving participants a chance to run some of the software for themselves.

The session was based on a training session that had been run in Chile during 2016 [1]. It consisted of two hands-on exercises:

- A set of Jupyter notebooks for examining light curves sampled at the LSST cadence.
- Instructions for using the LSST software stack to process a DECam image to obtain a calibrated image and a catalogue of detected objects.

These exercises were lightweight enough to run easily on participants' laptops, so rather than providing a centralised installation of the software for people to use, we instead provided detailed instructions for installing it locally.

3.2 Software installation

In advance of the workshop, I installed the LSST pipeline software on a number of different platforms and wrote up detailed instructions explaining how to do this [6]. The platforms were:

- macOS Sierra (10.12)
- CentOS 7 (64-bit)
- CentOS 7 (32-bit)
- Linux Mint 17.1 (64-bit)
- Windows 8.1

Installation on the Mac and 64-bit Linux platforms was straightforward, using the 'conda' installation method provided by LSST:

```
conda update conda
conda config --add channels
http://conda.lsst.codes/stack/0.13.0
conda create --name lsst python=2
source activate lsst
conda install lsst-distrib
source eups-setups.sh
setup lsst_distrib
pip install sncosmo iminuit
conda install lsst-obs-decam
setup obs_decam
```

The majority of the software required could be obtained by installing the `lsst-distrib` via conda, though it was necessary to install a few additional packages via pip (`sncosmo`, `iminuit`) and conda (`lsst-obs-decam`, `jupyter`).

However, 32-bit Linux and Windows were not supported directly. We made the decision that 32-bit Linux is now rare enough to be disregarded. To allow installation on Windows,

we created a Docker image containing the LSST software and other required dependencies. We were able to build on the Docker image released by LSST containing version 13 of the software stack, and simply add the extra packages we required on top using this Dockerfile:

```
# Start with the LSST image as a base
FROM lsstsqre/centos:7-stack-lsst_distrib-v13_0

# Install Pip, which will be used to install other
software
RUN curl https://bootstrap.pypa.io/get-pip.py | sudo
python -

# Install python-devel, needed for some of the modules
RUN sudo yum install -y python-devel

# Install Jupyter itself
RUN sudo pip install jupyter

# Install Python modules needed by the notebooks in the
tutorial
RUN sudo pip install matplotlib
RUN sudo pip install astropy
RUN sudo pip install sncosmo
```

The resulting image is available in the Docker image repository with the tag `jamespepcc/lssttutorial:1.0`.

I experimented with running the tutorial exercises on Mac, Linux and Windows (via Docker). This was mostly straightforward. The image processing exercise is done from the command-line and involves obtaining the tutorial software from GitHub, downloading an example DECam image, "ingesting" the image into the LSST software system via the `ingestImagesDecam.py` script, and then running the `processCcd.py` script to actually process it.

The light-curves exercise is embedded in a set of Jupyter notebooks. These contain instructions that walk the user through each step of the process, so for the most part no further instructions were needed here. However, one of the notebooks required a little work (installing an additional package, `lsst-sims-sed-library`, and adding a missing `import re` statement) in order to make its optional section on debugging run correctly.

In addition to the notebooks for analysing the light curves, a notebook to generate them was also provided. This notebook required access to the "fatboy" database server at the University of Washington which was not generally available to us or our tutorial participants; however, pre-generated light curves were also provided as Python pickle files, making it unnecessary to run the generation notebook in order to complete the rest of the tutorial exercises.

3.3 Running Jupyter notebooks within Docker

Getting the Jupyter notebooks to work within the Docker environment was slightly more challenging. Jupyter works in a client-server configuration, where the server is started by running `jupyter notebook` and the client is typically a web browser running on the same machine. This model is not a good fit for Docker: the server must run within the container as it depends on the LSST software installed there, but the client web browser cannot as Docker containers cannot normally run GUI software.

To overcome this problem:

- the port on which the Jupyter notebook server listens for connections (normally port 8888) must be made visible outside the Docker container so that a local web browser can connect to it. This can be done by passing the `-p 8888:8888` switch to `docker run` when starting the container.
- the user must determine the IP address of the Docker container so that they can type it into their web browser. The procedure for doing this varies from platform to platform and we provided instructions for common operating systems.
- because from the server's point of view the client is now running on a different machine, a password must be set on the Jupyter Notebook server, using the `jupyter notebook password` command.
- when starting the Jupyter Notebook server, the `--ip 0.0.0.0` switch must be given to ensure it binds to a network address that will be visible outside the container environment.

3.4 Additional tutorial notebooks

In addition to the tutorial notebooks already mentioned, Robert Lupton also demonstrated a number of his own Jupyter notebooks [2] at the workshop. These were of interest to a number of participants so after the workshop, with Robert's permission, we made the notebooks, a data set for use with them, and detailed instructions for their use available to the LSST:UK community.

Some of these notebooks use the DS9 GUI application [3] to display their output. This is challenging to set up in Docker and requires configuring either X11 forwarding or VNC to allow GUI apps running in the container to be shown on a local display outside the container.

4 The LSST Simulation Software

4.1 Background

In addition to the base LSST pipeline software already discussed, a number of other software packages are provided by LSST. An area identified as interesting to UK astronomers was the LSST simulations software, a suite of software tools capable of running end-to-end simulations of the entire LSST system. One component of this, the Metrics Analysis Framework, was particularly interesting.

4.2 Installation

After the Phase B workshop preparation and related work, I turned my attention to the LSST simulation software. The first stage was learning how to install the software. The binary installation method using conda, which I had used to install the base LSST software, was not supported for the simulations software.

Instead, it was necessary to install it from source code using the `eups distrib install` command provided by the EUPS [4] version management system. Whilst installation via this method is normally straightforward, it can be time consuming, taking several hours to download and build the software and all its dependencies. (The underlying LSST pipeline software must also be installed via EUPS rather than via conda before the simulation software can be installed on top).

4.3 The MAF tutorial notebooks

A set of tutorial Jupyter notebooks are provided for the MAF software [5]. These show several examples of how to use the MAF libraries from Python. While testing these I encountered a number of errors. These were due to incompatibilities between the notebooks and the latest version of the LSST simulation software. With the help of Lynne Jones, one of the notebooks' authors, I was able to get all of these errors fixed, by modifying either the notebooks themselves or the underlying library software.

One of the MAF notebooks requires access to the "fatboy" database server in Washington, and so can only be used by people with an account on this machine. However, we are currently exploring ways to make this notebook usable by hosting a subset of the information from the required database locally.

Other notebooks in the set rely instead on local SQLite database files: `minion_1016`, `enigma_1189`, `ops2_1114` and `pontus_1074`. With the exception of `pontus_1074` all of these are available for download from LSST, though some of them are several gigabytes in size.

4.4 The LSST simulations Docker image

In order to mitigate the time consuming installation process for the LSST simulation software, we decided to update our Docker container to include this software in addition to the base LSST pipeline. This was fairly straightforward but took several attempts – any download failure during the build process resulted in a failed build that had to be restarted from scratch. However, we were able to build a working Docker image including the simulation software, using a very simple Docker file:

```
# Start with original tutorial image (LSST + Jupyter) as
a base
FROM jamespepcc/lssttutorial:1.0
# Install LSST simulation software
```

```
RUN sudo bash -c "source
/opt/lsst/software/stack/loadLSST.bash ; eups distrib
install lsst_sims -t sims"
```

Because the LSST container on which our original tutorial container was built contains the LSST software installed via the EUPS method compatible with the simulation software, we were able to use this container as a base rather than having to start from scratch.

The resulting container worked for simple command-line tasks using the simulation software, however it was unable to run the MAF tutorial notebooks described above because of a conflict between the version of Python used by the LSST software installation, and the version used by the Jupyter installation. To fix this, I created a new version of the container (v1.1), with Jupyter installed into the LSST environment via Conda:

```
# Start with version 1.0 of the simulations image
FROM jamespepcc/lsstsimulations:1.0
# Install usable version of Jupyter
RUN sudo bash -c "source
/opt/lsst/software/stack/loadLSST.bash ; conda install
jupyter"
```

This container is available with the tag `jamespepcc/lsstsimulations:1.1`.

The container has been tested by other members of LSST:UK, and is now being used by Joe Zuntz and his colleague for running PSF image simulations.

4.5 Deploying databases locally

As mentioned previously, some of the SQLite database files required by the MAF tutorial notebooks are fairly large, and when using them with Docker special care must be taken (either copying them in from permanent storage, or mapping a local directory within the Docker container) to avoid having to download them again every time the container is launched.

To remove the need for each user to download local copies of these files, as well as to gain an understanding of how to deploy MySQL databases and access them from Jupyter notebooks, we decided to deploy local copies of some of these database files and make them accessible to the tutorial notebooks.

I dumped the `minion_1016` database using the SQLite `.dump` command, converted it to MySQL format using a Python script, and then imported it into a MariaDB server running on `lsstuk2.roe.ac.uk`. This was reasonably straightforward although some manual work was required in order to get the indexing working correctly.

It was then possible to access the database from the MAF notebooks by:

- opening an SSH tunnel to the database server and forwarding the MySQL port (3306). (This step is not necessary when the notebook server is running on one of the `lsstuk` testbed machines).
- setting a username and password for accessing the database server in the local file `~/.lsst/db-auth.paf`.
- modifying the database constructor call within the notebook to connect to a database on the `lsstuk2.roe.ac.uk` server instead of a local SQLite file.

I later repeated the same process for the `ops2_1114` database. It is hoped that these database installations will be useful in conjunction with the Jupyter Hub currently being set up at the Royal Observatory of Edinburgh.

5 References

[1] https://github.com/lsst-dmsst/demo-lsst-uk-2017

[2] https://github.com/RobertLuptonTheGood/notebooks/tree/master/Demos

[3] http://ds9.si.edu/site/Home.html

[4] https://github.com/RobertLuptonTheGood/eups

[5] https://github.com/LSST-nonproject/sims_maf_contrib/tree/master/tutorials

[6] https://lsst-uk.atlassian.net/wiki/spaces/COL/pages/52161514/LSST+Software+Tutorial+Notes