# Using Jupyter Notebooks As An Analysis Platform for LSST

## *Work Package 2 (Data Access Centre)*

| | |
|---|---|
| **Project Acronym** | LUSC-A |
| **Project Title** | UK Involvement in the Large Synoptic Survey Telescope |
| **Document Number** | LUSC-A-10 |

| | |
|---|---|
| **Submission date** | 20/AUG/2019 |
| **Version** | 1.0 |
| **Status** | Published |
| **Author(s) inc. institutional affiliation** | Gareth Francis (Edinburgh) |
| **Reviewer(s)** | Bob Mann (Edinburgh)<br>George Beckett (Edinburgh) |

| **Dissemination level** |
|---|
| Public |

# Version History

| Version | Date | Comments, Changes, Status | Authors, contributors, reviewers |
|---|---|---|---|
| 0.1 | 25/06/19 | First draft | Gareth Francis |
| 0.2 | 28/06/19 | | George Beckett |
| 0.3 | 1/07/19 | Added section 4 and executive summary, minor changes elsewhere. | Gareth Francis |
| 0.4 | 3/7/19 | Reformatted references. Added glossary. | Gareth Francis |
| 0.5 | 22/7/19 | | George Beckett |
| 0.6 | 20/8/19 | Added architecture diagram, requirements list as annex, rewrote section 5.2.2. | Gareth Francis |
| 1.0 | 20/8/19 | First version released | George Beckett |

# Table of Contents



# Index of Tables

# 1 Executive Summary

The LSST:UK Project is investigating Jupyter Notebooks[1] as a convenient and cost-effective way to provide an environment through which astronomers can access and analyse LSST data, as part of a wider suite of services called the Data Access Centre (DAC). In order to manage this environment we are deploying a JupyterHub [2] service providing pre-configured Jupyter environments to users on an on-demand basis.

An initial deployment has been made using the SFTC Cloud infrastructure [3] provided through IRIS [4]. The service is operational and in use by a small number of early access users.

JupyterHub is deployed to a Kubernetes [5] cluster, which we currently self-manage, but hope in future to be able to provision as a native OpenStack resource once this functionality is made available by the cloud provider.

AAI functionality is provided through the EGI Check-in service [6]. This means that users can log in using their own institutional credentials and allows DAC administrators to manage them using group management tools available in EGI Check-in.

In addition to the JupyterHub service itself, we also provision a number of supporting services such as remote logging, accounting and a local Docker repository. We provide a customised Jupyter environment – in the future likely a choice of environments – that is pre-configured with various standard astronomical libraries and software and database access settings. Using this environment a user can easily access the ZTF database in the current prototype DAC.

At present most of what we identify as the essential requirements for the service are met and the environment is usable. There is significant future work to be done in a number of areas, in particular around integration with other DAC services such as user-writable database access, access to additional data resources and the ability to offload processing to external clusters.

# 2 Introduction

## 2.1 Background

LSST data will be made available to users through several different routes, as part of a suite of services called the Data Access Centre (DAC). At one extreme, a web portal will provide easy, interactive access to standard (and lightweight) astronomy analysis functions for occasionally users. While, at the other extreme, data-intensive campaigns will be supported with bespoke workflows running in batch-processing mode on HPC resources, developed and implemented by groups of researchers—e.g. from one of the LSST Science Collaborations.

However, there is a significant demand for a level of access somewhere in between these two extremes, where an astronomer can interactively develop bespoke analysis workflows (analogous of scripts), with significant flexibility though only modest computing demands (e.g. a few hundred core hours). The LSST Project is investigating Jupyter Notebooks [1] as a convenient and cost-effective way to provide such an environment, with relevant software, to meet a significant portion of the envisaged use cases for data analysis.

Jupyter Notebooks are interactive documents that can contain code, equations, visualisations and text. They are becoming very popular amongst astronomers (and others) as a convenient way to explore and analyse data. They are easy to get started and yet scalable to large computations; Jupyter notebooks can be documented and shared, used as tutorials, and promote reproducible science. Once a researcher has learned a technique by reading the documentation and code contained in an existing notebook, that code can be executed and the results visualised, directly in the notebook. The code can then be modified and re-run with different data, different algorithms, different purposes, opening up a wide scope for experimentation with minimal start-up cost.

There are several potential options for managing notebooks in a multiuser environment, on-demand. JupyterHub is a popular option, at the time of writing, used in peer activities within the academic community (such as University of Edinburgh Noteable Service), which in turn are able to advise us on setup and confirmation. As the project website [2] describes it:

> *JupyterHub brings the power of notebooks to groups of users. It gives users access to computational environments and resources without burdening the users with installation and maintenance tasks. Users - including students, researchers, and data scientists - can get their work done in their own workspaces on shared resources which can be managed efficiently by system administrators.*
>
> *JupyterHub runs in the cloud or on your own hardware, and makes it possible to serve a pre-configured data science environment to any user in the world. It is customizable and scalable, and is suitable for small and large teams, academic courses, and large-scale infrastructure.*

Given this, we have elected to adopt JupyterHub for the experiments described herein.

## 2.2 Glossary of Acronyms

| | |
|---|---|
| AAI | Authentication and Authorisation Infrastructure |
| DAC | Data Access Centre |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| K8s | Kubernetes |
| LSST | Large Synoptic Survey Telescope |
| STFC | Science and Technology Facilities Council |
| VM | Virtual Machine |
| ZTF | Zwicky Transient Facility |

# 3 Deployment

An initial deployment of the JupyterHub service has been made to the SFTC Cloud [3]. This is an OpenStack Cloud resource operated by the Rutherford Appleton Laboratory as part of the IRIS e-Infrastructure [4].

## 3.1 Required Resources

The following summarises the OpenStack resources used at present:

| Purpose | CPU/Memory | Disk | Public IP |
|---|---|---|---|
| Admin | 1/1GB | 50GB | Yes |
| K8s Head node | 2/4GB | 50GB | No |
| K8s Worker nodes | 16/32GB | 50GB | No |
| HTTP Proxy | 1/1GB | 12GB | Yes |
| Accounting | 1/1GB | 25GB | No |
| Logging | 2/4GB | 50GB | No |
| Development K8s | 2/4GB | 25GB | No |

**Table 1: VM instance requirements**

The purpose of each VM is described in more detail in section 3.2 below, but briefly: the "admin" node runs internal services such as Rancher and the Docker registry; the head and worker nodes comprise the Kubernetes cluster; the roles of the HTTP proxy, accounting and logging VMs are self-explanatory; and the development VM provides a minimal additional Kubernetes cluster for development and testing purposes. The architecture is shown in outline in Figure 1 below.

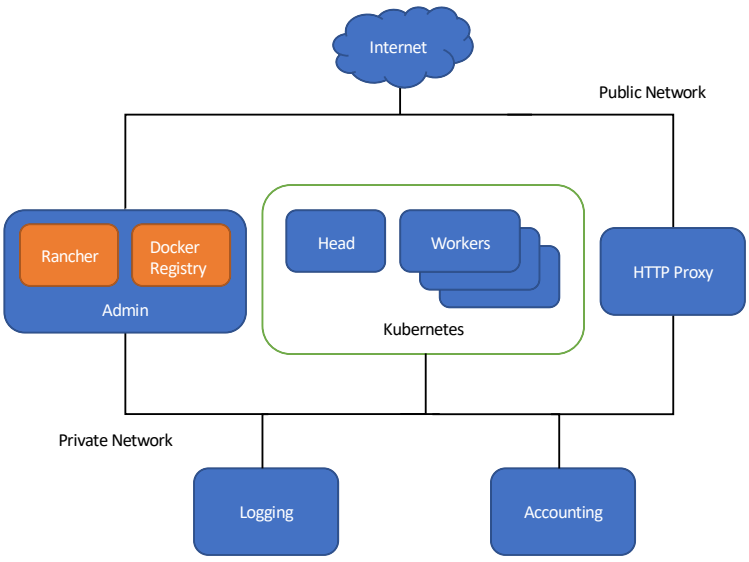


**Figure 1: Outline Architecture**

The number of worker nodes (currently one) is expected to grow, and potentially shrink, with demand. The sizing of the worker nodes is somewhat flexible depending on the underlying resources available, although larger instances here are helpful, both in terms of efficiency and by enabling us to offer larger, more capable notebook servers if required. The other VMs are simply the minimum size given their memory requirements.

| Purpose | Description | Current Usage |
|---|---|---|
| Fixed overhead | Not expected to grow significantly | 212GB |
| Variable overhead | Scales with number of worker nodes | 50GB |
| User storage | 10GB per user | 130GB |

**Table 2: Storage requirements**

## 3.2 Deployment Process

### 3.2.1 Kubernetes

We wish to deploy the JupyterHub service to a Kubernetes (K8s) cluster, but this is not yet widely available in IRIS as a native resource. We therefore provision the required VMs – one for administration, one as a K8s head node and as many worker nodes as we require based on the anticipated workload – then use the Rancher tool[7] to deploy the K8s cluster.

This is currently done using the GUI interface. Although this is not ideal in terms of replicability compared with a more scripted approach, it is considered acceptable since the only major configuration consists of setting the OpenStack cloud parameters. Furthermore, as we do not anticipate continuing to use Rancher once K8s clusters are available as native OpenStack resources (though e.g. Magnum) this is not considered a useful place to invest further effort.

Once the K8s cluster is deployed, a storage driver must be configured in order to allow K8s to create storage volumes on demand. We can then install Helm (an application deployment and management system that runs on top of K8s) in order to manage the deployment of JupyterHub itself.

### 3.2.2 JupyterHub

The JupyterHub service is deployed using Helm (a more detailed description of the process can be found at https://zero-to-jupyterhub.readthedocs.io/en/latest/setup-jupyterhub.html). The Helm charts have not required customisation (other than that which can be achieved using the standard configuration file); they can be found at https://jupyterhub.github.io/helm-chart/.

The configuration used can be found in the project GitHub repository (https://github.com/lsst-uk/jhub-test) under helm/jhub.

### 3.2.3 Docker Images

Containers (specifically Docker containers) are used both for the management services that make up the Jupyter Notebook offering, and for the user-spawned notebooks that run within this.

Two slightly customised Docker images are used, one for the hub itself and one for the notebook server. More custom notebook servers will likely be added in future. The Docker files are available in the git repository.

A local Docker registry running on the admin VM is used to serve these.

### 3.2.4 HTTP Proxy

While in principle a Kubernetes cluster can manage its own proxy server(s), our experience so far is that getting this working in an OpenStack environment is non-trivial and the resulting configuration fragile and not readily portable across different infrastructures. We therefore use an external proxy server (running Nginx) to manage this.

### 3.2.5 AAI

Identity and authentication is provided using EGI Check-in [6]. This allows users to log in using their own institutional credentials. We handle authorization using groups that we can define as required.

Technically the JupyterHub service acts as an OpenID Connect client. The configuration for this is all contained in the Helm configuration file.

### 3.2.6 Accounting

Accounting (at the user level) is handled by two components.

A custom lightweight service tracks users and running notebook servers. It does this using a REST interface which is called by the JupyterHub service when relevant events occur (user log in, spawning a new server, shutting down a server). Details of running servers are stored in a database and periodically a log file of completed sessions and their resource usage is written.

This log is ingested by Open XDMoD[8]. It can then be viewed and queried in much the same way as any other HPC resource.

Currently only CPU allocation is tracked in this way, although memory and storage could potentially be added if required.

The groups defined in EGI Check-in can be used by the accounting system. Where a user is a member of multiple groups they must select which group a session should be accounted to when spawning their server.

### 3.2.7 Logging

In order to facilitate log retention and analysis we use Elastic Stack[9]. This runs in a separate VM instance.

Since logs in Docker containers are necessarily somewhat ephemeral, FileBeat is used to export (a subset of) logs to the ElasticSearch database via Logstash, along with logs from the HTTP proxy. These can be visualised and queried using Kibana.

This system is somewhat experimental at present as the requirements regarding what should be logged, how long logs should be retained for and what analysis, if any, is required are all still in need of clarification.

### 3.2.8 Development Instance

In addition to the above, we have a second (minimal) K8s cluster configured as a test and development environment.

# 4 Current Status

## 4.1 Development Progress

The requirements for the Jupyter service are documented at https://lsst-uk.atlassian.net/wiki/spaces/LUSC/pages/662863877/Jupyter+Notebook+Service+Requirements+for. A summary of these requirements at the time of writing is provided in Annex A.

We do not view this list as being in any way complete since we expect to add new requirements and elaborate upon existing ones in future phases and as a result of testing by working astronomers, nevertheless it gives an overview of progress in the main areas. This can be summarised as follows:

- All high-level requirements are met, at least to some degree, showing that the system is in fact usable.
- All essential AAI and accounting requirements are met; a number of items are not implemented either due to being insufficiently well-defined at present or too low a priority.
- Requirements relating to collaboration and group working can be mostly met using version control systems such as Git provided that users already have access to such systems.
- Working storage and (writable) database access is at an early stage of development.
- Requirements relating to use of external resources, such as HPC or Spark clusters for large or long running processing tasks, are not yet well defined.

## 4.2 Operational Status

At present the service is operational at https://jupyter.lsst.ac.uk. It is currently open to a small number of invited users. Accounting data shows low, but sustained, usage by a subset of users.

A set of example notebooks (in various states of maturity) has been created at https://github.com/lsst-uk/jupyter_notebooks.

# 5 Future Developments

## 5.1 Deployment

### 5.1.1 Magnum

As mentioned above, as soon as OpenStack Magnum becomes available on the IRIS resources that we are using, we intend to replace our current system for deploying and managing the Kubernetes cluster. We hope that this will simplify our deployment process, improve the overall robustness of the system and support further work on elasticity and automation.

### 5.1.2 Elasticity

At present, any change to the size of the Kubernetes cluster requires a certain amount of work from the system administer to implement. Once Magnum is in place we expect to invest some effort in automating this process. Automated scale-out is expected to be straightforward; safely reducing the size of the cluster is likely to be more challenging since we need to be sure that running (or starting) notebook servers are not interrupted.

### 5.1.3 Automation

The deployment and configuration of JupyterHub itself is already highly automated using Helm. However, the various supporting services are deployed in a somewhat idiosyncratic manner. Ideally we will both automate as much of the deployment and configuration management as possible and minimise the number of discrete systems as far as possible, although there is likely to be a certain amount of tension between this and the desire to isolate critical systems such as logging and accounting.

## 5.2 Features

### 5.2.1 Integration with DAC services

As LSST:UK DAC services become available the Jupyter service will need to integrate with them. At present the only such service is the ZTF database, which is accessible (read-only) from Jupyter notebooks. Other services that we anticipate becoming available during the course of LSST commissioning (2019—2021) include:

- Access to other survey catalogues (that is, in addition to ZTF)
- Access to images from surveys
- Ability to create and manage short-lived databases to support user analysis
- Access to, in real-time, the transient-event stream from (for example, ZTF)
- Access to specialised computing platforms for offloading analysis. For example, a batch-processing cluster or a Spark cluster.

### 5.2.2 Additional supporting services

#### Version Control

The natural way to store, manage and share notebooks themselves is using a version control system (the project uses GitHub for example). Git is available to the user in the Jupyter environment if using our customised image, including GUI access to some functions if they are using JupyterLab (https://jupyterlab.readthedocs.io/en/latest/).

We do not currently provide a managed repository service for users. There could be some advantages of doing so, such as the ability to integrate with our AAI in order to provide single sign-on and integrated group management, for example. However, we would also imagine that many, if not most, user groups already have a version control solution that they are happy with and would not wish to incur the costs of changing for relatively minor benefit. There is also the consideration that providing a repository service

represents a substantial commitment to long term secure storage of user data that should not be undertaken lightly. Our intention is therefore to prioritise interoperation with existing solutions, whilst letting our direction for future development be guided to some extent by user demand.

### Data Storage

We anticipate that there is likely to be some demand for a system that would allow users to store, and potentially share, bulk data (rather than notebooks). This could be either (or both) files and database tables. If at all possible this should integrate with the AAI system.

Ideally, we would like to integrate with an existing system rather than building one specifically to support the Jupyter service, especially whilst the requirements are still uncertain.

### Documentation

More extensive documentation is required, both for users and administrators. This will need to include a curated set of example/ tutorial notebooks.

### Custom Server Images

At present we have a single custom server image. As the number of users and customisations grows we will need to find a way of managing this.

# 6 References


[1] Project Jupyter, https://jupyter.org/

[2] Project Jupyter | JupyterHub, https://jupyter.org/hub

[3] STFC Cloud, https://openstack.stfc.ac.uk

[4] IRIS: A common eInfrastructure for STFC science, https://www.iris.ac.uk/

[5] Production-Grade Container Orchestration – Kubernetes, https://kubernetes.io/

[6] EGI | Check-in, https://www.egi.eu/services/check-in/

[7] Rancher, https://rancher.com/docs/rancher/v2.x/en/

[8] Open XDMoD 8.1, https://open.xdmod.org/8.1/

[9] Elastic Stack and Product Documentation, https://www.elastic.co/guide/

# Annex A. Requirements

This section summarises the known requirements for the Jupyter service at the time or writing as documented at https://lsst-uk.atlassian.net/wiki/spaces/LUSC/pages/662863877/Jupyter+Notebook+Service+Requirements+for.

## A.1 High Level

J.1.1: A user should be able to perform an interactive analysis of survey datasets held in the DAC, without the need to download those datasets to local system and without the need to install astronomy software on local system.

J1.1.2: A user should be able to make use of standard (astronomy and generic) analysis software–including: AstroPy, MatPlotLib, ...

J1.1.3: A user needs to be able to save scripted analyses

## A.2 AAAI

J.2.1: Access to the DAC Notebook Service should be limited to registered users.

J.2.2: A DAC Administrator should be able to check topical institutional affiliation of each User reliably and accurately

J.2.3: Each User's activity should be logged with sufficient detail to fulfil the DAC's accounting obligations to the IRIS Service Provider.

J.2.4: A DAC Administrator can monitor use of IRIS-awarded computing-time/ storage by the Notebook Service as a whole.

J.2.5: A DAC Administrator should be able to determine how much computing time has been used, in a period, by each User.

J.2.6: A DAC Administrator should be able to determine how much storage is being used (and has, historically, been used) by each User.

J.2.7: A DAC Administrator should be able to organise Users into Groups (N.B. based around scientific campaigns)

J.2.8: A DAC Administrator should be able to define a computing-time quota for each Group, reflecting the amount of computing time they are nominally expected to consume in an allocation period.

J.2.9: A DAC Administrator should be able to define a storage quota for each Group, reflecting the maximum amount of working storage the Group is expected to consume during the allocation period.

J.2.10: A DAC Administrator should be able to define one or more PIs for each Group.

J.2.11: A PI should be able to monitor computing time usage by their Group

J.2.12: A PI should be able to monitor storage usage by their Group.

## A.3 Collaboration and Group Working

J.3.1: Anonymously clone a publicly accessible repository. Read-only.

J.3.2: Read/write to an existing repo on e.g. GitHub.

J.3.3: As above with a GUI.

J.3.4: Automatically set name and email parameters based on login information.

J.3.5: Private git repository for users.

J.3.6: Give users ability to set authz on private repos using same identities and groups as for login.

## A.4 Working Storage and Local Databases

J.4.1: Create a database

J.4.2: Create a table in an existing database

J.4.3: Load data from a file into an existing database table

J.4.4: Write data from a notebook (e.g. from within a pandas DataFrame) into an existing database table

J.4.5: Read data from a database table into a notebook

J.4.6: Share an existing database table with a defined set of users

J.4.7: Run DB admin tasks (e.g. build indexes, etc) on existing database table

## A.5 Escalation to Cluster/HPC Resources

Not yet determined.