LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

Science Platform Design

Gregory Dubois-Felsmann, Frossie Economou, and Kian-Tat Lim

LDM-542

Latest Revision: 2017-07-07

Abstract

This document describes the design of the LSST Science Platform, the primary user-facing interface of the LSST Data Management System.

Change Record

Version	Date	Description	Owner name
1	2017-07-07	First Draft.	Gregory Dubois-Felsmann, K-T Lim, Frossie Economou

Document source location: `https://github.com/lsst/LDM-542`

Contents

Science Platform Design

1 Introduction

The "LSST Science Platform" (LSP; LSE-319) refers to a set of software and services that is used to provide data rights holders and project team members access to the LSST data and support for its scientific analysis, and is frequently used to refer in particular to the user-facing presentation of these components of the LSST system. The requirements for the Science Platform are described in LDM-554.

The Science Platform provides access to both catalog and image data through a range of cooperating interfaces (Python APIs, Web APIs, and a graphical user interface), as well as providing dedicated computing and storage resources to users at the LSST Data Access Centers.

The Science Platform's principal components are delivered by the LSST construction project groups at SLAC (Database and Data Access), NCSA (middleware and infrastructure), LSST-Tucson (SQuaRE), and IPAC (SUIT). These components make extensive use of the Science Pipelines software developed by the UW (Alert Production) and Princeton (Data Release Production) groups to provide data analysis software frameworks and algorithms.

The Science Platform is visible to users through three "Aspects" giving access to a unified system.

1.1 The Portal Aspect

The Portal Aspect, a highly evolved version of a traditional astronomical-archive GUI, provides Web-based query and visualization tools for all the LSST data products, designed to facilitate the user's discovery and query of the wide variety of catalog and image data products, illuminate the connections between them, enable exploratory data analysis, and provide contextual access to relevant documentation. The Portal Aspect's UI is extensible with additional computation and visualization tools developed by users and collaborations. It provides access not only to the Project's released data products but to storage for file- and database-oriented user data, and facilitates the sharing of such data between users and within collaborations.

1.2 The Notebook Aspect

The Notebook Aspect provides access to a Python-oriented computational environment hosted at the LSST Data Access Centers through the "Jupyter Notebook" paradigm that has become extremely influential in astronomy and in the wider scientific computing community in recent years.[1] Through a Web-based notebook interface, users are able to run Python code in close proximity to the LSST data archive, accessing and analyzing the data and generating derived data products. Curated releases of the LSST software stack are made available to all users of the Notebook Aspect, including all the framework and algorithmic software used in the Project's own data reduction pipelines. Sufficient configuration and provenance data are made available to permit users to reproduce the production processing steps and then to customize them to their own ends or to assemble entirely new analyses. Batch computing services and storage are available to users, with larger allocations requiring resource utilization proposals, enabling them to perform analyses of a substantial scale. Users are enabled to install additional software at their own discretion, enabling them to bring existing analysis code to the LSST data, and take advantage of the wide variety of computational and visualization tools available in the scientific Python ecosystem.

1.3 The API Aspect

The API Aspect provides remote access to the LSST data, user data, and user computing resources, through a set of Web APIs (many based on VO standards). The Web APIs will deliver data in community-standard formats, including, e.g., VOTable, CSV, and FITS. The same Web APIs are used internally in the Portal Aspect, and are also available in the Notebook Aspect.

Notebook Aspect users, running their personal Python code within the Data Access Centers, will normally access the LSST data and resources through the DM Python APIs, whose implementation will generally bypass the Web APIs.

1.4 Architecture

Underlying all three Aspects are a common set of services hosting the LSST catalog and image data and user data products, a user storage "workspace", and providing access to flexible

[1] The LSST project is closely following current developments in the Jupyter project and expects to use the upcoming "JupyterLab" evolution of the notebook concept. We are already using an alpha release of JupyterLab in our development activities.

computing resources. Catalog data are stored in database systems providing highly scalable parallel access to the science data and Observatory metadata. The database systems and the Web and Python data access APIs through which they are accessed support flexible, user-defined ADQL queries. Image data is provided through a combination of image metadata search facilities and pixel data retrieval services that support cutouts and mosaicing. Additional service provide access to a wide range of metadata describing the available data products, their structure and their provenance.

Further services support the creation of user image and catalog data products and facilitate the federation of user data products with the Project's (e.g., permitting joins between the Project's catalogs and catalogs of additional, user-derived attributes). User data products created within the Data Access Centers are remotely accessible through the API Aspect. The authentication and authorization services provided allow users to restrict access to the datasets they create to specific other users and groups of users, supporting the work of scientific collaborations.

By relying on common underlying services, exchange of data and seamless workflows crossing Aspects are enabled. For example, the data resulting from a graphically-driven query in the Portal Aspect can be immediately accessed in the Notebook Aspect for further analysis. Similarly, data retrieved or created in the Python environment in the Notebook Aspect can be displayed or accessed in the Portal aspect.

Both the Portal and Notebook Aspects of the Science Platform, as Web applications, are designed to be usable by remote users without the installation of any software on users' own systems, with all computing and data access hosted at a Data Access Center. Nevertheless, all the software required to host the Science Platform is part of the LSST open-source code base, installable on mainstream Unix-like platforms, and we anticipate its usability both by non-Project data access centers and by individuals and groups on their own systems.

2 The Architecture of the Science Platform

This section presents the overall architecture of the LSST Science Platform (LSP, not to be confused with the Science Pipelines "stack" code), including its major components and interfaces and design principles.

2.1 Design Overview

The LSP is a multi-tier architecture with distinct but inter-related user-facing "Aspects" over an underlying layer of services that in turn rests on computational infrastructure.

2.1.1 Functional Architecture

The Portal Aspect, JupyterLab Aspect, and Web APIs Aspects provide three distinct user interfaces. Internally, the LSP interfaces with the Data Backbone via the Data Butler client interface and direct "native" Data Backbone interfaces to provide access to the Science Data Archive and (for some instances) other data products. The LSP also has its own data storage for Portal configurations, user notebooks, and other user files and databases. It interfaces with compute provisioning services, authentication and authorization services, and resource management, including a proposal management system to handle requests for resources. Networking connects the LSP with internal systems and the external world and provides a measure of cybersecurity.

The Portal Aspect is implemented using the language-agnostic Web APIs to retrieve data. The JupyterLab Aspect can also use the Web APIs if the user desires, either through a VO client or directly as a Web service.

2.1.2 Deployment Architecture

The LSP is designed to be deployable in multiple locations within the LSST DMS. The production instances are the US Data Access Center, the Chilean Data Access Center, the Science Validation instance in the NCSA Enclave, and the Commissioning Cluster instance in the Base Enclave. Additional instances are used for integration testing in the Integration environment and for science pipelines developer usage in the Developer environment. LSP developers will of course instantiate components of the LSP during their own development.

The LSP is composed of multiple services that are containerized and deployed using Kubernetes. The underlying provisioning of compute resources is handled by the Provisioning and Deployment Service component of the LSST DMS. The technology to be used for provisioning in this component is being evaluated and may vary between instances. vSphere, HTCondor, SLURM, and OpenStack are all potential alternatives. Provisioning is elastic: JupyterLab dy-

namically instantiates notebooks on demand, and the Portal and Web API services can be expanded or contracted as needed. This elasticity and fault tolerance considerations require that the services hold minimal non-persistent state so that loss of a single service instance has correspondingly minimal impact on users.

2.2 Data Access and Storage

Since the goal of the LSP is to retrieve and analyze LSST data products, data access and the storage of user data are key aspects of the platform.

2.2.1 Databases

Since the LSST catalog data products represent the best measurements of astrophysical phenomena available to the project as well as metadata describing how data was taken and processed, it is anticipated that most uses of the LSP will involve these catalogs which are stored in databases.

2.2.1.1 Database content overview Several types of databases will be accessible to the LSP instances:

- Image and visit metadata
- Catalogs
- Composite data (binary data in catalogs)
- Observatory metadata (including EFD)
- Reference catalogs

Image and visit metadata includes information about the observation captured at the time. It also includes information derived later from analyzing the data and metadata, such as an accurate mapping from pixel coordinates in an image to sky coordinates or an estimate of the photometric zeropoint in an image. The metadata will follow the Common Archive Observation Model [6, 1] as its core. This metadata will be stored in relational databases and made available via ADQL queries.

Catalogs are described in detail in the Data Products Definition Document LSE-163. They include summary measurements of astrophysical Objects, plus measurements in images from each epoch of observation and in difference images. These will also be stored in relational databases and made available via ADQL queries. The catalog of all Alerts that have been issued may be stored in a different database technology, such as a document database, and may thus use a different query interface.

The above catalogs will include "BLOB" fields that provide large data items that are useful to retrieve along with other catalog data but are not suitable for searching. Examples include samples of the posterior probability surface for a model, photometric redshift probability distribution functions, or "heavy footprints" giving deblended pixel values for an Object. These are generally stored in separate tables within the relational database as they are used less frequently.

The Engineering and Facility Database, which contains a record of all commands, configuration, and telemetry from the Observatory systems, will be transformed to provide simplified query access to conditions during each observation. All data present in the original EFD will be available in the transformed EFD with the exception of Internal, Sensitive, or Highly Sensitive information, as defined in the Information Classification Policy LPM-122, which will not be available to users other than project staff. The EFD does not contain Protected User data. The Transformed EFD will be stored in a relational database, accessible via ADQL queries.

Reference catalogs used in the LSST Alert, Calibration Products, and Data Release Productions will be provided, also in relational databases. Additional catalogs, e.g. from precursor or concurrent surveys in a variety of wavelengths, will also be provided as relational databases depending on usefulness and available resources.

2.2.1.2 Database technologies While many of the above databases will be implemented in "conventional" relational database technology, the largest catalogs will require a distributed system to provide sufficient performance. The LSST-developed Qserv distributed database will be used to implement these. The two implementations may vary somewhat in their support of ADQL features. Users creating their own catalogs (see section 2.2.5) will need to specify whether they are to be distributed, thereby selecting the appropriate back-end implementation.

2.2.2 Images

LSST image data products include raw images, processed visit images, difference images, coadded images, and templates. These are part of the Science Image Archive in the Data Backbone. All of these are accessible to the LSP.

Some data products are currently specified to be "virtual": they are regenerated on demand from other persistently-stored data products, reflecting the estimates of the technology-dependent space/time cost trade-off. Such virtual data products are regenerated for the LSP by services that are part of the LSP Web APIs.

Images will be delivered in at least FITS format; other formats may be added. The LSST DM internal image data format is currently FITS as well, but that may change in the future. Any necessary translation between internal format and external format is performed by services that are part of the LSP Web APIs.

Image data products typically contain additional information besides image pixels. This information includes metadata describing the conditions under which the image was generated, its provenance and processing history, if any, and metadata derived from the image itself. Much of this metadata can be expressed as simple data types in "headers" or similar mechanisms, but some, such as mask and variance planes, pixel-to-sky mappings, and point spread function (PSF) models, may be too complex for such a representation. When this information is specific to the image, LSST DM generally stores this information in the same file as the image pixels, as additional pixel planes or extension tables. The structure and content of these will be documented for LSP users.

2.2.3 LSST-specific Objects

The Python objects used by the Science Pipelines code do not always correspond one-to-one with persisted catalog or image data products in the Data Backbone. For example, an image and its background model may be persisted in two distinct files. The PSF model associated with the detections in an image may be stored in a file while the information about the detections is stored in a database catalog.

The component files and catalogs are directly accessible via the LSP, but for convenience the Data Butler client library provides mechanisms to define composites that combine multiple

components into a single retrieved Python object. Since this usage is specific to the Python language, it is only directly visible to the user in the LSP in the JupyterLab Aspect.

2.2.4 Data Access Services

The LSP Web APIs Aspect provides services to retrieve catalog and image data products. Access to catalog data products is provided by a Web service that supports the VO TAP protocol [7] with queries specified in ADQL [13]. Access to image and file data products is provided by a separate Web service that supports the VO SIA [8], SODA, and VOSpace [10] protocols. The image and file service is also responsible for regenerating "virtual" data products and converting from internal to external data formats as necessary. Both of these Web services support asynchronous operations, as some requests will have long response latencies (e.g. large-scale catalog queries or data regeneration). Access to metadata about images and files is provided by TAP/ADQL queries on metadata tables, rather than a distinct metadata service. More detail on these services is provided in section 5.

Within the JupyterLab Aspect, and internal to the Web APIs Aspect, the Data Butler client interface may be used to retrieve Python objects, and a "native" Data Backbone API may be used for high-performance retrieval of datasets. The Portal Aspect uses the Web APIs Aspect exclusively for data access; it does not use the Data Butler nor the "native" Data Backbone API.

2.2.5 User Catalog Data Support

Users may create their own catalogs within the LSP, a capability that is patterned after SDSS SkyServer's "MyDB". Catalogs may be derived from queries to the LSST catalogs or loaded from computations or external data sources. As mentioned above in section 2.2.1.2, large distributed catalogs meant for joining with the LSST data products must be identified so that they can be stored appropriately in the Qserv database.

All three Aspects will provide mechanisms for creating user catalogs; the Portal's mechanism will be implemented via the Web APIs, as usual. User catalogs may be accessed in the same ways that LSST catalog data products are accessed.

User catalogs are backed up for disaster recovery purposes, but they are not fully protected

against user error.

2.2.6 User Workspace Storage

Users may create their own images and files within the LSP. These all live within a User Workspace that is common across all three Aspects. All three Aspects will provide mechanisms for creating and retrieving user files. In the JupyterLab Aspect, this is via a mounted filesystem. In the Web APIs Aspect, this is via VOSpace or WebDAV.

User file workspaces are backed up for disaster recovery purposes, but they are not fully protected against user error.

2.2.7 Data Access Permissions and Quotas

All LSST data products are available to all users with LSST data rights. Users who had LSST data rights but have since lost them may be granted access to the data products available as of the time of loss but not newer ones.

Users may grant other users or groups of users access to catalogs and files in their workspace. Such access may be read-only or read-write.

Per-user catalogs and files are a limited resource with quotas determined by resource management (see section 2.5). Some "user" catalogs and files may be considered to be owned by a group, with a distinct quota.

2.2.8 Support of Previous Releases

The LSST baseline currently anticipates that the data products contained in the last two Data Releases (as well as Level 1 Alert Production data products) will be available through the LSST Science Platform. Data products from other Data Releases would need to be retrieved via the Bulk Distribution service; they would not be visible within the LSP.

A request to have the catalogs from all Data Releases be available through the LSP is being evaluated.

2.3 Computing Resources

The LSP provides access to computing resources to enable users to analyze LSST data products, alone and in combination with user-provided data.

2.3.1 Basic user compute services

Users of the Portal Aspect share computing resources allocated to the Portal as a whole. The same strategy is used for the Web APIs. Within those shared resources, per-user accounting is performed for resources such as queries and temporary disk space so that resource management policies may be implemented if required.

Each user of the JupyterLab Aspect is assigned a virtual machine per notebook. The virtual machine can be selected from a menu of preconfigured installations of LSST and other software, or users can customize a virtual machine and save it for future use. The resources allocated to the virtual machine are defined by policy.

2.3.2 Large-scale batch and parallel computing

Users of the LSP (any Aspect) may trigger large-scale, long-latency, asynchronous data processing. This processing is handled by submitting jobs to a batch system based on HTCondor. Mechanisms are provided in the three Aspects to monitor the status of batch jobs and retrieve their results. Instances of the workload and workflow management tools and services used by the LSST Data Facility to manage large-scale productions will be made available to each LSP instance, although these will be separate and distinct from the production instances, of course.

2.4 Authentication and Authorization

Identity management and associated roles and authorizations are provided by a unified LSST system as described in LSE-279. This system allows the use of external identity providers such as the Chilean COFRe federation [2] or the US InCommon [3] or GitHub; those verified external identities are then mapped to internal LSST identities. All uses of all instances of the LSP, including the Web APIs Aspect, are authenticated.

2.5 Resource Management

LSP users in the DACs will receive a certain quota of resources (compute, storage, query, bandwidth, etc.). Usage beyond that quota will be allocated by a committee. Outside the LSP, a proposal management system will track requests and allocations that will then be implemented by per-user resource management features within each LSP component.

2.6 Cybersecurity Considerations

The Data Access Center instances of the LSP are the only ones exposed directly to the public Internet. The Science Validation, Commissioning Cluster, Integration, and Developer instances are only reachable from LSST-internal networks, helping to reduce the attack surface. The DAC instances are isolated (on a different LAN) from the production Level 1 and Level 2 domains within the NCSA enclave. Although they may share underlying computing resources, such resources are provisioned for either production or DAC but not both at the same time. The LSST data products within the Data Backbone will be read-only to the DAC LSP instances. The Data Backbone interfaces are all authenticated, but they will be used for some types of user data, so the Backbone must be hardened against attack (or accident) from the LSP. Provisioning separate virtual machines per user in the JupyterLab Aspect helps to prevent unwanted sharing of information between users. The Portal Aspect and Web APIs Aspect need to be hardened in the same ways as any Web-enabled service, including sanitizing user inputs, avoiding cross-site scripting, etc.

2.7 Additional Support Services

System management and monitoring services for the LSP will be provided by the infrastructure layer supporting each instance. Monitoring at the system and application level will be exposed to LSP users where consistent with user privacy and system security.

3 The Portal Aspect

The Portal Aspect of the LSST Science Platform provides a Web user interface for the science community to discover, access, explore, analyze, and download the LSST data. It provides query and visualization services for all the LSST data products, designed to facilitate the

user's discovery and exploration of the wide variety of catalogs and image types available. It illuminates the connections among the data products, including the data flows that produced them, and provides contextual access to relevant documentation.

The Portal Aspect provides for both simple form-based queries and the construction of complex queries in ADQL. It provides a sophisticated set of data visualization capabilities for inspecting the results of queries, both for catalog and image data, and performing basic exploratory data analysis on the results.

It provides access not only to the Project's released data products but to storage for file- and database-oriented user data, and facilitates the sharing of such data between users and within collaborations.

The implementation of the Portal Aspect within the overall Science Platform, and its reliance on common underlying services with the other Aspects, allows users to identify and query data in the Portal and then access the results in the Notebook Aspect, where they can perform custom analyses and draw on the full variety of computational and visualization tools available in the Python language. Conversely, users can also query or create data in the Notebook Aspect, constructing complex ADQL queries and processing workflows, and then make the results available in the Portal for visualization.

The Portal Aspect is being designed to serve users with a wide range of levels of expertise and knowledge of the LSST survey and data products. It provides new and casual users with a guided introduction to, and discovery of, all the LSST data products, but also provides expert users with a source of quick reference information on the available data products and their structure, and the associated documentation.

The Portal Aspect is intended for users with data rights. It will, therefore, start with a login screen at which user may use the credentials they have registered with the LSST project to identify themselves. All interaction with the detailed survey data will require their user account to hold the appropriate rights.[2] How much broad summary information may be provided to unauthenticated users through the Portal itself — e.g., documentation on the LSST project, a summary of the progress and coverage of the survey — versus through other public faces of the Project is yet to be determined.

[2] Note that although the *broadcast* of alerts via VOEvent (or a successor) is world-public, the use of the *Science Platform* to browse or query the historical alert database is limited to data rights holders.

Since all users will be authenticated, the Portal will be able to provide identity-based persistent state that allows users to start their work in one browser and then continue it elsewhere. How much user state will be identity-based and how much will be session-based is still being considered and will be adjusted in response to user feedback.

The Portal Aspect also provides convenient access to external data relevant to the understanding and exploitation of the LSST data products: to reference catalogs maintained on LSST systems, to a selection of specific publicly available data archives, and to any data archives available through standard VO services.

3.1 Generic Data Browsing

A key requirement on the Portal is that it provides for discovery of all the LSST catalog and image data products as well as the Observatory metadata such as the contents of the Engineering and Facilities Database. By relying on the creation of extensive metadata on the structure of the data products in the Science Pipelines code and other data sources, and on its delivery via the deployment of the data for release through the API Aspect and the underlying databases and services, the Portal is able to provide content-sensitive documentation, query, exploration, and visualization of every released table.

Driven by the metadata available through the API aspect, the Portal will be able to display to the user the available data releases, and all the tables available in each release and in Level 1. The Portal will allow the user to select the the data product to search on, provide a search form for the selected data product, generate the search request based on user's input, submit the search and display the results. If sufficient metadata are available, a detailed search form will be constructed that allows the user to specify constraints on any column without having to know how to write an ADQL query. The search form will also always provide a means for the user to specify an ADQL query directly. In either case, the Portal will then pass the ADQL along to the API Aspect to do the search.

Given appropriate metadata (e.g., IVOA UCDs), the Portal will recognize specific columns as representing spatial data, particularly sky coordinates, and will provide contextually appropriate search forms supporting commonly-used queries such as cone and box searches.

If the selected data product is catalog or other tabular data, the search results will be displayed as a table in the Portal. The table viewer will provide a familiar set of behaviors for

manipulating the results, such as sorting by column, setting the page size, and customizing the column selection. In addition, it will provide basic data analysis features such as the ability to filter on a column or add a synthetic (calculated) column. For any row in a query result, the system will provide the ability to obtain a metadata-driven formatted view of all the data in the row — a "property sheet" for the row — which can be extended to the full width of the underlying table(s) (in effect performing a "SELECT *" on demand for a selected row in a narrower query result). By providing adequate metadata throughout the LSST data products, the Portal will be enabled to, for instance, display values together with their uncertainties in these property sheets.

The Portal will recognize columns that are keys to information in other tables and will facilitate navigation to those tables for the related information.

The Portal will provide for straightforward plotting of tabular data. It will provide for the creation of scatter (or line) charts for any selection of X and Y axes from the available columns or arithmetic expressions based on combinations of columns, with assistance from the UI in actions such as finding columns of interest to use or verifying the spelling of their names. The Portal will also have the ability to create histograms on demand for columnar data or expressions on columnar data.

If the selected data product is image data (i.e., if the search was on an image metadata table), the Portal will be able to use the associated metadata to construct queries for the images themselves and display them to the user. A variety of functions will be available to adjust and interact with the image displays: stretch, color change, zoom, rotate, flip, crop, measure distance between two points, overlay a variety of coordinate grids, add markers, upload DS9-style "region files", plot the telescope field of view, etc.

Tables of catalog data containing IDs linking to the images on which the measurements were made can be used in the same way to drive browsing over image displays.

The user experience when viewing image data will depend on the latency of availability of the image data from the underlying services. Currently only coadded images and a 30-day cache of calibrated single-visit images (and cutouts from such images) are expected to be available with low latency. The Portal will be able to deliver views such as "flip books" of cutouts from the single-epoch images for a given region of sky for the entire survey, but only upon submitting a request for the data, which may take of the order of several hours to

fulfill. The user workspace may be used to save the results of a limited number of such data requests, based on user quotas, for later re-use.

The table, charting, and image display components of the Portal will be linked through an underlying shared data model. As a result, the Portal can provide "brushing and linking" and similar capabilities, reflecting selections and operations on data in one component into the other components. For instance, the results of an Object table catalog query can be displayed simultaneously as a table, an overlay on deep coadd images, and a color-magnitude diagram. Selection of a region in the color-magnitude diagram will be immediately reflected as a highlighted selection of rows in the corresponding table and of object markers in the image overlay. Similarly, a column-based selection in the table will be reflected in the diagram and on the image.

All query results — tables and images — will be available for download in either their original or interactively modified forms. Data may be downloaded either to the user's remote system (i.e., the system running the browser) or to the user's Science Platform workspace. The Portal Aspect will also support the creation of user databases based on tabular query results.

In addition, all queries on LSST or LSST-hosted performed by the Portal through the API Aspect will be associated with the user and will be readily retrievable through the Notebook Aspect or directly through the API Aspect. This capability, further discussed below, is the key to enabling seamless scientific workflows that cross the Aspects. It will also be possible for a user to browse their query history and select individual queries for re-execution or for download of the previous results (if they are still in the results cache).

The Portal Aspect will provide for access to user data, both in the Science Platform's user workspace (which accommodates file-oriented data) and in user-created databases. It will provide a traditional hierarchy-browser capability for the workspace, with the ability to inspect and visualize the contents of image or data table files of recognized types (e.g., FITS, VOTable, CSV). The workspace browser will also provide basic management functions for the organization of the workspace hierarchy, renaming and deletion of files, and the like. The Portal Aspect will provide for discovery, query, and visualization of user databases using the same generic data browsing tools as for the LSST data, though the sophistication of this interface will depend on the availability of adequate metadata for the user databases.

The generic data browsing capabilities of the Portal will also be available for selected non-LSST

reference catalogs that will be hosted at the LSST Data Access Centers, for purposes of calibration, data quality analysis or user convenience. The semantic knowledge of the LSST data that the Portal is planned to exploit may be more limited for these hosted external datasets, but all the basic functions of the Portal will still be available. In addition, the Portal will be able to access any publicly available dataset that has a VO-standard interface, particularly a TAP/ADQL interface. Semantic features of the interface will be enhanced when external data (hosted or remotely accessed) are organized according to the CAOM model (in the case of observational metadata) and when queries return well-curated UCDs. Query results will be capable of being saved to the user workspace.

3.2 Documentation Delivery

A key function of the Portal is to connect the user to documentation on the Portal itself, the Science Platform more generally, the data structure and quality, the data processing, the survey, and the LSST itself. Much of this documentation will be created by other parts of the project, but the Portal will provide a starting point for discovering relevant documents.

3.2.1 Context-Dependent Help

The Portal will provide online documentation in a variety of ways: implicitly through the naming and labeling of UI elements and screens, via “tool tips”, and via dedicated “Help” buttons or icons.

UI elements such as the label and title for a window, an input form, an input field, a column in a result display, a function name, etc. are the first points of entry for users to understand and interact with the LSST data. Such labels and titles have to be well thought out, as precise and unambiguous as possible, but also human-readable. This may mean that many entities may need to have both an “internal” name, such as the true name of a table in a database, and a “common name” that facilitates human understanding and speech about the entity. In such cases, if the common name is displayed by default in the UI, the UI must allow the user to discover *and copy and paste* the true name so that it can be used in contexts, such as typing ADQL queries, where it is required.

Due to the limitations on screen real estate and resolution, and the desire to design “efficient” layouts that devote as much space to the data as possible, labels and titles cannot generally

contain all the useful information available. A second level of context-sensitive documentation is, therefore, provided in the form of "tool tips" or hover-over text. The design will allow for descriptive text to be associated with all data product entities, notably table and column names, and displayed on mouse-over. (The actual provision of this documentation will be a collaborative effort across Data Management.) In addition, the availability of metadata such as units and IVOA UCDs will permit the delivery of context-dependent help in the interpretation of displayed values and in the use of search form fields.

The final category of context-dependent online documentation will be associated with explicit "Help" icons in various places in the interface. These will take the user to detailed documentation in another window, with a strong preference for "deep-linking" that takes the user first to the most relevant point in a document. It is by this means that we expect to provide access to the larger world of documentation on the data processing, data quality reports, and documentation on the survey and the Observatory itself.

3.2.2 User Guide

While the aim of the Portal design will be to make its use, and the features it provides, as evident as possible through the UI elements themselves, we nevertheless expect to provide a user guide to the Portal. The user guide will describe the major capabilities of the Portal, the options available associated with each major screen and function, and design specifics such as the use of user state vs. session state. It will grow, with user feedback, to contain additional information of a "FAQ" nature that has been found to be useful. The user guide will contain examples that alert users to the breadth of capabilities they can expect to find in the Portal Aspect and motivate their use with scientific examples. Elements of the user guide will be linked to appropriate places in the UI as context-dependent help, but will also be available as part of the full document.

Some parts of the user guide will naturally cross Aspects of the Science Platform — for instance, a user-friendly explanation of the "user workspace" will naturally cover its availability and use from all three Aspects.

3.2.3 Reference and Deployment Manuals

Along with the user guide, the Portal development project will also produce reference and deployment manuals. The reference manual will include API documentation for both the client and server components of the Portal, as well as documentation on the interfaces the Portal relies on with the rest of the LSST system.

The deployment and maintenance manual will provide the information required to maintain the operation of the Portal in the context of the LSST infrastructure, including information on how to run and configure its components, and how to ensure that they are able to communicate successfully with the rest of the Science Platform components and infrastructure. It is intended primarily to be useful to the project's operations staff, but it will also be aimed at providing information useful to those who may wish to set up additional instances of the Science Platform outside the project.

3.3 Customized Workflows

In addition to the metadata-driven generic data exploration capability, the Portal Aspect will provide dedicated workflows aimed at improving the user experience for access to a small number of the most commonly accessed LSST data products.

This involves some extra effort to design the specific screens, UI actions, and connections from one to another. It involves a tradeoff between the improvement in user experience and a potentially increased exposure to the need to modify these elements of the Portal to account for changes in the structure of the data, e.g., from one data release to the next. It can also be seen as a tradeoff between providing certain functionality directly versus extending the capabilities and generality of the metadata system to unjustifiable levels.

Examples of Portal features in this category may include:

- In cases where the scientific information of the information in one table requires a more complex query involving, for example, calibration information from another table (e.g., the subtraction of photometric zero-points), the Portal will provide UI support for the single-click application of that calibration to queries generated from the standard screens. It will then also provide contextual help explaining what has been done,

so that the user can learn the underlying query language that is required and then be able to perform similar queries, e.g., in the Notebook aspect, without the UI support.

- UI guidance for the interpretation of parent-child relationships between entities related by deblending algorithms.
- Property sheet screens for the most important tables (e.g., Visit, Object) that organize and present the data in more user-friendly ways that depend on expert knowledge of, e.g., what the most scientifically interesting and most commonly used attributes are, or which attributes may be the most likely to be of interest to see close to each other on the screen, or that provide explanatory text or visual guidance beyond what can be readily represented by the metadata and general contextual help system.
- Customized UI actions available in property sheets or the table viewer for key tables, providing a greater richness and/or user-friendliness of follow-on searches. For example, we expect to provide a simple "Display light curve" UI action on the property sheet for an Object or a DIAObject. In contrast, in a generic-browsing version of the Object property sheet, the same functionality might be met by a "Search related tables" UI element that would let the user pick the "ForcedSource" table from a list of tables with foreign-key columns pointing back to Object, and then pick the appropriate flux column from the ForcedSource table. In cases where such simplified functionality is provided, the contextual help system will be used to ensure that the user understands what the actual search performed is, including the ability to review the ADQL used, so that the user can then decide to customize the search differently from the default.
- Prioritization of columns in tabular data displays for the most important tables, making — in the absence of user requests to the contrary — a consciously designed subset of the columns in these tables more easily accessed in the table viewer, in a column picker, etc.
- Custom designs of mouse-over inspectors for data points in plots generated from key tables.
- Pre-configured, UI-selectable actions for generating commonly used displays or calculated columns. For example, for photometric Object data where the actual database table contains per-band magnitudes, the UI may provide single-click actions for choosing to display colors, or even request standard displays like color-color or color-magnitude plots.

- Different default settings for parameters such as image stretch for different types of images (e.g., for coadds, difference images, and calibration flats).

The successive versions of the Prototype Data Access Center deployments of the Portal, and the use of the Portal during commissioning, will be used to gather experience and user recommendations to guide the development of the most useful such customizations as the operations era approaches.

The Portal will be designed to facilitate the continued development of additional customizations during the operations era, as user experience accumulates.

3.4 Use of the LSST Stack

The basic functionality of the Portal Aspect is based on the external persisted forms of the LSST data provided through the API Aspect's Web interfaces, which return "on the wire" data in community standard forms such as FITS image and table files and VOTable. In many cases the Portal implementation is able to work directly with these persisted forms, without dependencies on the LSST pipeline code base in order to interpret them. However, among the LSST data products there are also ones for which there are no semantically meaningful community standard forms, and for the interpretation of which the use of the LSST code is very useful, if not indispensable.[3] Likely examples of such data objects are source footprints and "heavy footprints" (bit maps and pixel flux data, respectively, for single sources and groups of sources), and image PSF models.

In order to offer visualizations of these data products, the Portal Aspect will invoke LSST stack code to transform them into useful visual forms such as image overlays, plots of PSF variation, and the like. The Portal Aspect will only attempt to provide a limited set of such standard visualizations, recognizing that the Notebook's Python environment and the range of visualization toolkits that will be available there will enable the user community to develop a rich set of specialized tools for these purposes.

Note that even in cases where there are LSST data products for which no dedicated visual-

[3] "Not indispensable" because ultimately all the persisted data object forms require complete documentation, of a quality and depth which would permit the independent development of code to interpret them. However, in some cases, due to the complexity of the data model, we anticipate that such a reimplementation would be costly to develop.

ization is provided, the Portal Aspect will still provide the ability to inspect the data object contents. For example, for data products represented by complex FITS table models, the tables themselves will be viewable even if the Portal does not directly provide a semantic interpretation of the data product they represent.

The extent to which this will be required is still being determined, because the full persisted data model for the LSST data products has not yet been specified. As above, development will prioritize the development of visualizations for the most commonly used or otherwise important data products.

3.5 API Interaction with the Portal

The Portal Aspect will provide APIs for interacting with its capabilities and with user sessions. It will be possible for a user to programmatically upload image and tabular data and open a visualization in a current Portal session — again, with the sophistication of the results dependent on the provision of metadata. It will also be possible to programmatically query the state of a session, in support of users' ability to begin a workflow in the Portal and then continue it programmatically outside the Portal Aspect – most notably, inside the Notebook Aspect.

3.6 Extensibility of Visualizations

The Portal software infrastructure is being developed to permit users to create visualizations not in the basic Portal software release. The Portal relies on the open-source scientific graphics library Plotly for the generation of a variety of types of plots. Plotly provides well-documented APIs and data structures for the creation of a wide variety of scientific and other graphics. While only a few of these will be used in the standard Portal configuration, the Portal will be designed to permit the use of the underlying interfaces to add custom plots under user control.

3.7 Extensibility of the UI

The Portal Aspect's UI will be extensible with custom actions associated with data types and selections. For example, in the Prototype Data Access Center the capability has already been demonstrated of creating a button in the image-viewing UI that allows the connection of a selected region on an image to user code that performs an action based on that selection.

3.8 Implementation

The planned implementation of the Portal Aspect is based on an evolution of Firefly, originally developed at Caltech/IPAC for the NASA Spitzer Heritage Archive and the IRSA archives. Firefly is a framework and a set of libraries for the development of Web-based client-server applications for accessing astronomical archives. It comes with a native set of capabilities: basic data-search functions, astronomical image visualization, and sophisticated tabular data visualization capabilities including a shared data model that permits the delivery of brushing-and-linking functionality across tabular, chart, and image displays. Firefly also provides a set of JavaScript, Web, and Python APIs, allowing the construction of customized Web applications, but also allowing users to access the visualization components in their own simple web pages (or Jupyter notebooks) without having to build a complete web application.

Many Firefly-based applications are in operations in IRSA-operated archives, such as the Spitzer Heritage Archive, the WISE Image Service, the Planck US Data Center visualization service, PTF Image Service, and the new IRSA Finder Chart application.

Conceptually, Firefly-based applications are organized into three tiers: Client, Server, and Data Services. The Client tier is responsible for rendering the display of the data and providing the UI. The Client is a native Javascript application based on the React framework. The Javascript APIs of the components of the application allow for the straightforward construction of a wide variety of portal UIs combining images, tables, and plots of the tabular data. Standard Scientific visualizations are generated in the Client via the Plotly libraries, while astronomical image visualizations are handled in a Google Maps-like way with tiled rendering performed on the Server.

The Server tier manages searches, the Firefly shared data model that enables data linkages across visualizations, and provides the interface to other supporting services such as authentication providers. The Server tier provides an abstraction layer for creating “search processors” usable from the Client. These are plugin interfaces to a variety of data sources, providing support both for executing queries via standard external protocols such as IVOA TAP and for working with data services with custom protocols. The Server handles search requests, dispatches them to the proper data service, obtains the results, and prepares them for rendering and display. The Server also provides exploratory data analysis features that can be applied to the data it has received.

The core of the Server is a Java application, hosted on a Tomcat server. Communication between the Client and Server tiers is through a combination of standard HTTPS requests and a WebSocket interface. The Server tier provides the Web APIs — and wrapper Python APIs for them — that enable driving and querying the server state from other system components or by users.

The Server tier includes the ability to extend the capabilities of Firefly through microservices. This facility will be used in the Portal Aspect to provide visualizations for LSST-specific "blob" data products for which the DM stack code will be used to convert the data product into an artifact suitable for display.

Finally, the Data Services tier actually performs searches and other data retrievals, and is generally composed of components provided by the underlying data archives. In the case of the Portal Aspect's Firefly application, the Data Services tier essentially comprises the data access services provided by the API Aspect.

As part of the LSST project, Firefly will be extended with more sophisticated metadata models enabling the the implementation of the advanced generic-browsing behavior described above, support for all-sky visualizations based on HEALPix and the HiPS file format, and with the ability for users, via the Firefly APIs, to call on advanced visualizations provided by the Plotly library, beyond those used for Firefly's native functions.

In addition, in developing the Portal, the SUIT team is reviewing the existing heritage of UI elements provided by Firefly and revising the appearance and workflows to take advantage of contemporary developments in user experience design, and to provide a uniform experience across the tool.

Key design guidelines in this process are:

- the efficient use of display space, with as many pixels as possible devoted to rendering data in preference to static UI elements;
- in data-viewing screens, a corresponding preference for more context-sensitive "pop-up" UI elements versus static buttons;
- consistency of presentation of common entities (e.g., sky coordinates) across the Portal, with an emphasis on the ease of selecting an entity in one place in the UI and using it

(e.g., by copy and paste) in another without editing;

- consistency of layout and of the behavior of UI elements, including issues like the use of modal versus modeless dialogs and inspectors, and the positioning and meaning of controls such as “OK”, “Apply”, “Cancel”, and “Dismiss”.

3.9 Interface to the Alert Distribution and Filtering Services

Beyond providing the archive-access functionality described above, the Portal Aspect has the additional role of providing data-rights holders a simple UI for access to the alert distribution and filtering facilities provided by the Project. This is essentially a “mini-broker”; unlike the envisioned community alert brokers, the LSST-specific facilties will have access only to the information in the LSST alert messages themselves. The UI will allow users to establish and control subscriptions to the alert stream, and to select from a modest set of pre-defined filters, or submit simple user-defined filters, to be applied to the alert stream they receive.

The Portal UI for this will be closely tied with screens for the querying of the alert database and the visualization of its contents, to provide convenient workflows for users interested in the time-domain LSST data. We anticipate being able to provide data access workflows that enable users to tie the information actually present in alerts to additional data, though context-sensitive searches and the like.

4 The Notebook Aspect

The Science Platform offers in-house scientists and engineers as well as (in operations) general science users the opportunity to interactively explore the LSST dataset from a python notebook environment.

4.1 JupyterHub / JupyterLab service

The notebook service is a scalable kubernetes deployment with JupyterHub spawning individual JupyterLab pods for each user. The basic architecture is outlined in Figure 1. JupyterLab is a rich notebook architecture that offers terminal access and dashboarding in addition to the traditional Jupyter Notebook concept and the use of JupyterHub allows it to be scaled to demand as infrastructure allows.

A user of the JupyterLab sevice will hence be able to interact with LSST services in the same way they might have with python scripts in a shell environment. They can use our python interfaces to the web APIs and the Butler for data access, the interface to the workflow system for batch processing, visualisation via bokeh, Firefly, matplotlib etc.

The capabilities of the service are tied to major project milestones outlined in the table below in a series of phased releases. Each release is primarily targeted at a different audience - see Section 7 for a summary of how the capabilities of the Science Platform as a whole are going to be used by teams inside the project.

Version	Audience	Target L2 Milestone	Shims ok?
proto	SQuaRE and Arch	none	y
alpha	DM devs, Commissioning team training	LDM-503-1	y
beta	Commissioning Team	LDM-503-9	y
1.0	Commissioning (ComCam) Capabilities	LDM-503-12	n
2.0	Science Verification Capabilities	LDM-503-14	n
3.0	General Science User Capabilities	LDM-503-16	n

Note in particular that in addition to the originally planned “Level 3” science user capabilities, the JL service is the primary interface of the LSST System Scientist to Commissioning. This introduces scope not required by the general science user, eg. timely access to data in the Engineering Facilities Database and the ability to rapidly deploy instances of the service on the Commissioning Cluster.

For analyses requiring significant data processing, a process for transitioning a trial-and-error notebook into a batch job involving execution over a large dataset will be provided.

For detailed architectural discussions of the JupyterLab development work-in-progress, its interfaces to other components and the current schedule for major and internal milestones see SQR-018. As technical choices are baselined they will transition to this document.

4.2 Pre-installed LSST software

An important advantage of the JupyterLab environment is that it is already set up and optimised for running a variety of LSST data. We achieve this by building containers directly from the weekly releases created by our continuous integration system and then deploying them

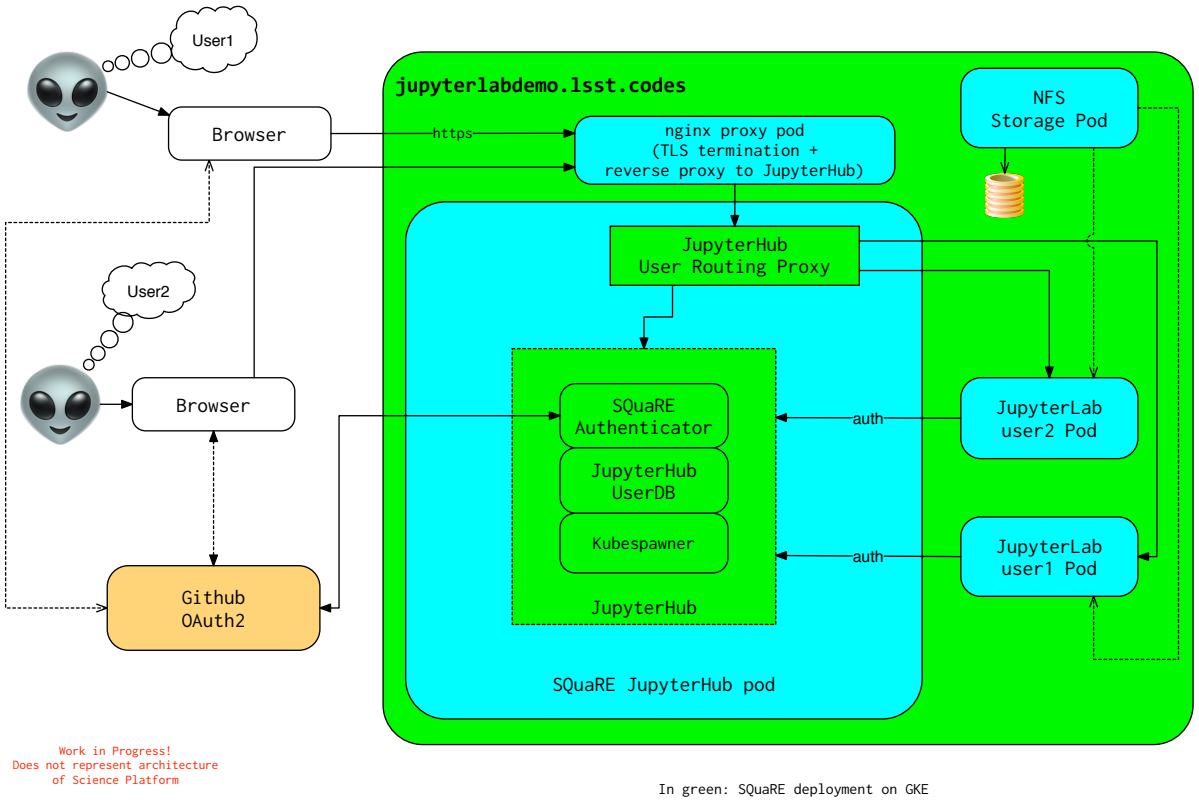

FIGURE 1: Architecture of the proof-of-concept (alpha) JupyterLab deployment

to JupyterHub which asks the user which version to use to spawn their pod. This means the users always have a choice of cadence (latest weekly or last stable) in their environment depending on their risk tolerance versus appetite for the latest feature. We plan to make other complementary third-party astronomical packages (eg. sextractor) available too.

4.3 Pre-configured access to LSST data

The JupyterLab service will make use of standard LSST stack features such as DAX services, the Butler, and their initialisation through the SuperTask Activator to allow python-level access to LSST data. After that users can use a mix of their own code and stack classes to manipulate and visualise the data. An example notebook running on our prototype deployment using classes from the LSST stack is shown in Figure 2.

5 The API Aspect

The Web APIs provide language-agnostic, location-agnostic authenticated access to LSST data. They are intended to be used by client tools and automated processes. They are not optimized or supported for direct human use via a browser, although they can be used that way. The available endpoints and the URL structures they accept will be documented.

The Web APIs are undergoing detailed design. Initial prototypes exist that provide non-VO-compliant access to catalogs, images, image mosaics, and image cutouts. This section describes the ultimate goals for the LSP Web APIs and documents design features where they are already known.

5.1 Overview - VO services and custom LSST services

The Web APIs support standard VO protocols where possible. They will also support extensions to the VO protocols and additional custom LSST services that are useful for the Portal Aspect or JupyterLab Aspect. To the extent that it makes sense, such extensions will be proposed to the IVOA for incorporation into the standard protocols.

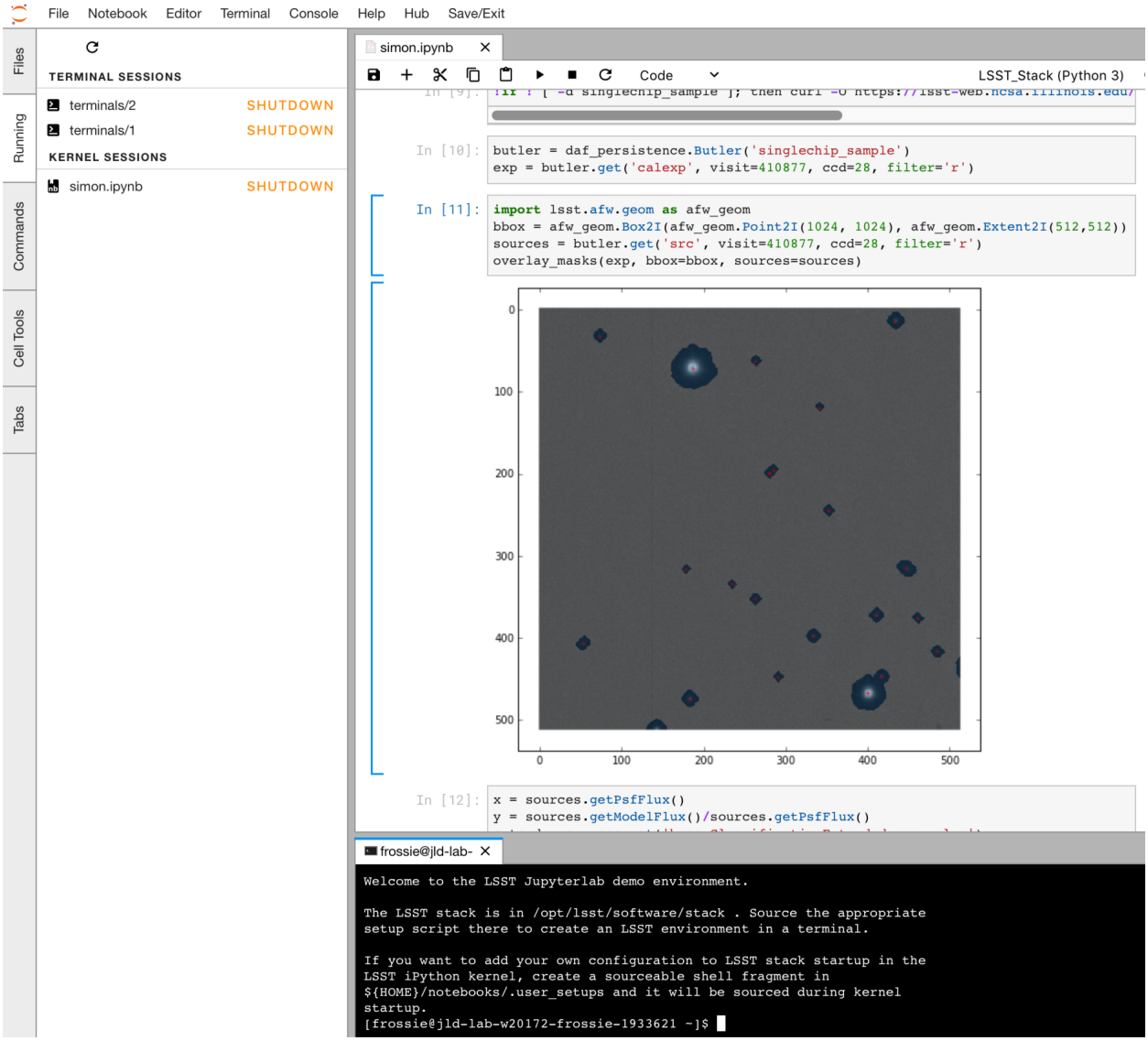

FIGURE 2: Example of star-galaxy separation analysis with the JupyterLab prototype

5.2 Catalog and other tabular data access

Catalogs, metadata, provenance, and other tabular data are accessed through the `dbserv` endpoint. For metadata and provenance, table schemas, when possible, will be reused from VO or community standards, such as CAOM2 [1] and others.

5.2.1 TAP service

`dbserv` is the LSST implementation of IVOA's TAP interface. `dbserv` will target compliance with TAP 1.1 once it is recommended by the IVOA.

5.2.2 ADQL implementation

Internal to dbserv is an ADQL parser. This parser will target compliance with ADQL 2.1 once it reaches recommended status. In addition to that, SQL language features of underlying databases will be made available, when possible. Some capabilities will be restricted if a database does not support them, such as subqueries which do not perform well in a distributed database system.

5.2.3 Return data formats

As `dbserv` will be TAP compliant, it will support the VOTable output specified by TAP 1.1, which should be version 1.3. Additional encoding protocols for VOTable output will be supported, including JSON and SQLite output, which were chosen for the ability to represent metadata inline with VOTable data. For all output formats, gzip compression through the HTTP protocol will be supported and encouraged.

5.3 Image data access

Image discovery, metadata, and data retrieval will be available through the `imgserv` endpoint.

5.3.1 Image finding

Image discovery will be facilitated through an implementation of the IVOA SIA V2 interface. Simple implementations of SIA allow direct access to files available via HTTP by adding a link to the SIA response; `imgserv` will use that attribute in the SIA response to link to another service. Image metadata will also be facilitated through TAP, as image metadata will be available according in the database through implementations of the IVOA ObsCore Data Model [15] and the CAOM2 data model, as well as LSST-native image metadata representations.

5.3.2 Image retrieval

Images may be retrieved via the URL returned in an SIA request, in the case images can be directly served over HTTP, or via a SODA implementation. The LSST SODA implementation will also offer enhancements to implement project-specific goals without presenting additional interfaces to science users. For example, key/value pairs as used by the Data Butler client may be provided.

5.4 Metadata access

Catalog and image metadata can be accessed via TAP query to provided metadata tables.

5.4.1 Data Releases

Each Data Release has its own set of tables. These are expected to differ in schema between Data Releases. In addition, the image and catalog metadata available for each Data Release will vary. Finally, identifiers within catalogs and images (except raw image identifiers) will vary between Data Releases.

Accordingly, the Data Release number is a key parameter that must be presented to the Web APIs. Collections of datasets that are not part of a Data Release (e.g. in-process, unreleased data or intermediate data products) are assigned labels that are used in place of the Data Release number.

The number of Data Releases available through the LSP and therefore the Web APIs aspect is discussed in section 2.2.8,

5.4.2 Tables

Available tables include those in LDM-153.

Catalog and image metadata tables will be implemented through IVOA and community standards, when possible, such as those laid out by IVOA's ObsCore DM, VOResource/RegTap, and the CAOM2 data model. Even if direct usage of those models is not feasible, a best-effort export to them will be performed. In such a case, the "native" metadta tables will be documented for direct access by users through TAP.

5.4.3 Table Structure

Each table will have associated metadata that records, for each column, a description, UCD when appropriate, unit when appropriate, and any relationship with other columns (e.g. so several columns making up a covariance matrix can be identified as such without resorting to pattern-matching of column names).

5.5 User Workspace access

WebDAV and/or VOSpace will be provided for user workspace access.

6 The Interconnectedness of the Science Platform

A key element in the Science Platform design is that by layering the three Aspects on top of the same underlying services, it enables scientific workflows that cross from one Aspect to another.

For example, work can begin with using the Portal Aspect to find data, continue in the Notebook Aspect to analyze it and create and store a derived data product, and the API aspect to access it remotely. Or, a complex query can be constructed in the Notebook Aspect and the results sent for visualization in the Portal Aspect.

The ability to work in this manner depends on three underlying capabilities of the Science Platform.

6.1 Sharing Data

All three Aspects expose the same LSST data products and provide access to the same user workspace and user database data. The user workspace, likely to be accessible via a VOSpace service, is available in any place in the Portal where the download or upload of data is semantically appropriate. For instance, a target list from another survey or mission can be uploaded to the workspace, accessed in the Portal to perform a cross-match query, and analyzed in the Notebook together with other LSST data to identify structure in the combined data.

6.2 Sharing Queries

Because the underlying DAX services allow for asynchronous queries, with a query submitted and assigned a name which is available for later re-use, and because the system is capable of recording the history of queries for a user, a query can be submitted from one Aspect and the results accessed from another.

Queries can be constructed using the Portal UI, with the Portal's visualization tools used for a quick look to ensure that the results are as expected, and then the query results can be retrieved from the Notebook - or the query itself can be programmatically repeated at a later date, e.g., by someone tracking the development of Level 1 data on a transient object.

Or complex queries can be programmatically constructed in ADQL from Python code in the Notebook, and the results discovered and browsed in the Portal.

6.3 Sharing State

Finally, the Portal exposes both read and write access to elements of its internal state. User code can push data at the Portal for visualization using its Python or Web APIs, or modify the state of a display (e.g., an image stretch). This capability can be used for overlays; for example, if the user is viewing an image in the Portal and has identified points or regions of interest in the image using code in the Notebook, those coordinates can be pushed to the Portal as an overlay. User code can also retrieve state from the Portal; this can be used to retrieve data from a user's session, but also to facilitate interactions with the data. For example, the position of a point or region selection on an image, or a row selection on a table, that the user has made in the Portal UI can be made available to user Python code in the Notebook.

7 Application of the Science Platform inside the Project

The Science Platform is not only for use by scientist end users through the Data Access Centers but it also forms an integral part of the toolchain within the Project. A phased release of capabilities focuses on the needs of the primary target audience for each major version. This approach allows us to deliver features of the platform to project staff to assist them in construction work while also providing an early user base whose feedback will allow us to refine our design.

7.1 Data Management Teams

The earliest releases of the Science Platform are aimed at Data Management teams. The capabilities supporting software development include interactively troubleshooting code and analysing metrics through the Notebook Aspect. The capabilities supporting algorithmic development include visualising the results of large processing runs of precursor and simulated data through the Portal Aspect.

7.2 Commissioning Team

Intermediate releases of the Science Platform are aimed at the commissioning team. The capabilities supporting commissioning include the interfaces needed to allow Notebooks to cross-correlate early ComCam data with telemetry captured in the Engineering and Facilities Database and a deployment of the Science Platform on the Commissioning Cluster.

7.3 Science Validation Team

Mature releases of the Science Platform are aimed at the Science Validation Team which is expected to involve a mixture of Project and Science Collaboration members. Science Validation requires many of the features of the operational Science Platform since it involves detailed interactive data analysis and generating and visualising statistics from large data runs.

7.4 Observatory Operations

In the Operations era, all of the above workflows will continue to be in heavy use by Observatory staff: software support will continue to characterise releases using the notebooks and reporting tools developed in Construction; facility checkout after engineering periods at the telescope will re-use many of the processes developed in commissioning; and facility scientists and calibration specialists will use workflows similar to those developed during science validation to enhance the understanding of the instrument. This will ensure that all Science Platform Aspects made available to the community remain vigorous and relevant.

References

[1] Common Archive Observation Model, URL `http://www.opencadc.org/caom2/`

[2] Comunidad Federada REUNA, URL `http://cofre.reuna.cl/index.php/en/`

[3] InCommon: Security, Privacy and Trust for the Research and Education Community, URL `https://www.incommon.org/`

[4] **[LDM-153]**, Becla, J., 2013, *LSST Database Baseline Schema*, LDM-153, URL `https://ls.st/LDM-153`

[5] **[LDM-554]**, Ciardi, D., Dubois-Felsmann, G., 2017, *Science Platform Requirements*, LDM-554, URL `https://ls.st/LDM-554`

[6] Dowler, P., 2012, In: Ballester, P., Egret, D., Lorente, N.P.F. (eds.) Astronomical Data Analysis Software and Systems XXI, vol. 461 of Astronomical Society of the Pacific Conference Series, 339, ADS Link

[7] Dowler, P., Rixon, G., Tody, D., 2010, Table Access Protocol Version 1.0, IVOA Recommendation 27 March 2010

[8] Dowler, P., Bonnarel, F., Tody, D., 2015, IVOA Simple Image Access Version 2.0, IVOA Recommendation 23 December 2015

[9] **[SQR-018]**, Economou, F., 2017, *Investigations into JupyterLab as a basis for the LSST Science Platform*, SQR-018, URL `https://sqr-018.lsst.io`

[10] Graham, M., Morris, D., Rixon, G., et al., 2013, VOSpace specification Version 2.0, IVOA Recommendation 29 March 2013

[11] **[LSE-319]**, Jurić, M., Ciardi, D., Dubois-Felsmann, G., 2017, *LSST Science Platform Vision Document*, LSE-319, URL https://ls.st/LSE-319

[12] **[LSE-163]**, Jurić, M., et al., 2017, *LSST Data Products Definition Document*, LSE-163, URL https://ls.st/LSE-163

[13] Osuna, P., Ortiz, I., Lusted, J., et al., 2008, IVOA Astronomical Data Query Language Version 2.00, IVOA Recommendation 30 October 2008

[14] **[LPM-122]**, Petravick, D., 2015, *LSST Information Classification Policy*, LPM-122, URL https://ls.st/LPM-122

[15] Tody, D., Micol, A., Durand, D., et al., 2011, Observation Data Model Core Components, its Implementation in the Table Access Protocol Version 1.0, IVOA Recommendation 28 October 2011