Proposed SUIT and related release processes

SUPERSEDED

This page is superseded by DMTN-136.

See also https://docs.google.com/document/d/15QSbahJKfq616JM0IrC6Xb7ywMJmlPggNk5iAP15ltY/edit#heading=h.utsrnzvuqwsv (SUIT Release Procedure Tailoring).

Firefly core

The Firefly package comprises both a library of visualization components (for images, tables, and table-based scientific plots) and a set of basic Web applications built on those components. The system combines a Java-based server side and a JavaScript-based client side. Firefly is designed to be deployed on a Tomcat server. In practice most deployments include multiple servers with a load-balancing front end (typically based on NGINX or Apache HTTP Server). Firefly is open-source and maintained on Github at Caltech-IPAC/firefly.

Firefly was developed originally to support the astronomical archives at IRSA, was substantially revised and extended to meet the needs of the LSST project, and benefits from contributions from other IPAC projects.

Currently "vanilla Firefly" deployments are used on the `lsst-demo` visualization server, as well as to serve `/firefly` endpoints in the LSP Kubernetes configurations at NCSA and in the cloud. It is expected that in the operations era these applications will be phased out in favor of endpoints in the LSST-specific Portal deployment, but for the time we would like to preserve the capability to deploy "vanilla Firefly".

Firefly core release process

The IPAC build processes for Firefly typically produce both a .war file archive (which includes the Firefly server Java code and the JavaScript code to be served at run time) and a Docker-format container image containing a Java runtime, Tomcat server, and the Firefly client and server code. LSST deployments and IPAC-internal development and test deployments are all based on the container images. (A previous "standalone deployment" build option was to produce a .war file archive that also contained a version of Tomcat, in order to allow Firefly to be started with only a JRE pre-installed, but as of mid-2019 this option is no longer being used and the equivalent functionality is offered by providing the Docker image.)

Projects needing additional features, search screen customizations, etc. generally combine Firefly with additional code in a project-specific repository in order to produce deployment artifacts. The current Firefly design uses the build scripts from the `firefly` package itself to build the project-specific artifacts. As a result, builds of these downstream artifacts require the build-time presence of the Firefly source code. (In other words, they are *not* built just as a layer on top of the build artifacts of the core Firefly package.)

The IPAC release process for Firefly is being revised to better support the increasing number of projects outside IRSA that deploy Firefly-based applications.

IPAC controls the development and release process through a Change Control Board on which LSST is represented. IPAC performs builds (using a highly automated in-house Jenkins configuration) and generates Docker images, which are deployed on an in-house Kubernetes cluster for feature-branch and release-candidate tests. Releases are made periodically, under CCB control, following in-house testing. They will be made available via Github, based on tags of the form `release-2019.2.0`, as well as by releasing Docker-format container images corresponding to these releases, on Dockerhub, with Dockerhub tags matching the release name string. These "vanilla" Firefly release images are available at ipac/firefly on Dockerhub.

LSST deployment configuration for "vanilla" Firefly core releases

As noted above, currently "vanilla Firefly" container builds such as this are used on the `lsst-demo` visualization server (as a bare Docker container), as well as to serve `/firefly` endpoints in the LSP Kubernetes configurations at NCSA and in the cloud. (These usages are expected to be phased out by the operations era.)

Until now, the policy has been that IPAC periodically updates the Dockerhub tag `ipac/firefly:lsst-dev` to indicate which build has been "blessed" for deployment on LSST services. We will continue to do this until a new policy is agreed with LSST.

What is required is a mechanism for determining, from the available set of `release-yyyy.m.n` release images, which will be used at any given time in the various LSST deployments.

It is proposed that, for now, we from time to time update tags `lsst-lsp-int` and `lsst-lsp-stable` to signal which builds are recommended for LSST deployment, with the former controlling which image is used on the `lsst-lsp-int` cluster, and the latter which image is used in all production and otherwise widely-used contexts, including the bare-Docker `lsst-demo` deployment and the production-like LSP instances on `lsst-lsp-stable` and the future Commissioning Cluster.

As an alternative, IPAC can create only the unchanging release-specific tags (as well as a `latest` tag we will maintain for casual outside users), and leave LSST to maintain the mapping of releases to deployments internally. (This could be done in a data file or by cloning the images into LSST-maintained image repositories.) We ask LSST to provide input on a preferred mechanism.

Portal Aspect (SUIT)

The LSST Portal Aspect is implemented as a Firefly application. As mentioned above, this involves combining Portal-specific code, primarily HTML and Javascript, (in the LSST-Github `suit` package) with Firefly core code from Caltech-IPAC/firefly and, as above, building a container image which contains a JRE, Tomcat server, compiled Java code, and ready-for-service JavaScript code. Roughly speaking, the resulting set of Web applications are a superset of those from the Firefly core builds, with some LSST-specific relabeling, icon changes, etc. applied to the core applications.

Currently the Portal Aspect applications are run in LSP instances by deploying containers in Kubernetes based on the resulting container images.

SUIT release process (proposed)

Until now, IPAC has performed the builds of the `suit` container images, in Jenkins, and has released the containers at ipac/suit on Dockerhub, again periodically updating the `lsst-dev` Dockerhub tag to indicate which version is "blessed" for deployment.

We are proposing to evolve the build process for the Portal Aspect (i.e., for the `suit` package) to follow more closely the software release mechanisms being defined by Gabriele.

Releases of the `suit` package will be based on semantic-version tags, e.g., v1.0.2, instantiated in Github. Each release will include a version of a file that names the Firefly release to be used in the recommended build of the `suit` package. (There is often some flexibility in cross-release compatibility, but for LSST software release purposes we will identify a specific Firefly release that should be used in building the release artifacts associated with each `suit` pa ckage release.)

IPAC will provide (**DM-20931** - Getting issue details... STATUS) a build script that LSST can use in an automated build infrastructure (e.g., Jenkins or Travis) that, given a tag of the `suit` package, will produce a Docker-format container image that includes the `suit` code, Firefly code, the IPAC-specified Tomcat server version, and a JRE (which will settle down on JRE 11).

LSST plans to release these images on a public Dockerhub account. The SQuaRE group will make a recommendation for the appropriate account and container image name to be used for the Portal application, ideally in the context of a wider scheme for organizing all the LSST-created container images needed for the deployment of the LSP. (Possible patterns: lsst-lsp/(aspect)-(component), e.g., lsst-lsp/portal-app; or lsst-lsp-(aspect)/(component), e.g., lsst-lsp-portal/app .) In the case of the Portal Aspect, the container images will be tagged on Dockerhub with the same version string used for the `suit` pac kage release name.

In addition to the public Dockerhub release, LSST may also run an internal Docker registry on which the same images are available.

As a bridge to this mechanism, IPAC will continue for a transition period to perform builds of the `suit` package, using the new build script, and release the resulting images at ipac/suit on Dockerhub.

Again, for now, we will continue to use a revisable `lsst-dev` Dockerhub tag to indicate the "blessed" versions, until a new policy has been agreed with LSST.

Portal Aspect online help

A similar mechanism will be used to generate container images for the Portal Aspect online help. The details are still being worked out.

Other IPAC-maintained packages

firefly_client (Python API for communication with a Firefly session)

firefly-api-access

jupyter_firefly_extensions

firefly_widgets

Other Firefly-related LSST-maintained packages

display_firefly (Firefly-specific back end for afw.display)

This package is maintained as a pure LSST Python package, using all the standard tooling as it applies to other LSST Stack packages. It is included in `lsst-distrib`.