Rubin Observatory

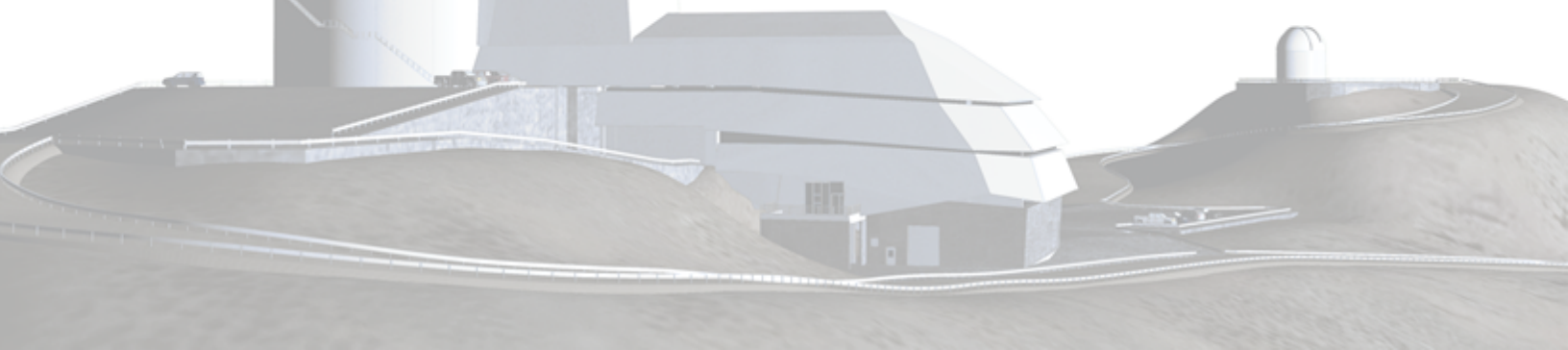
Vera C. Rubin Observatory
Data Management

Offline Batch Production Services Use Cases

Mikolaj Kowalik, Michelle Gower, and Rob Kooper

LDM-633

Latest Revision: 2020-10-12



Rubin Observatory

Abstract

This document describes use cases for Offline Batch Production Services.

Rubin Observatory

Change Record

Version	Date	Description	Owner name
1.0	2019-07-16	First release, approved in RFC-602	Mikolaj Kowalik

Rubin Observatory

Contents

Rubin Observatory

Offline Batch Production Services Use Cases

1 Introduction

This document describes use cases for the LSST Offline Batch Production Services (BPS). These use cases are driven by offline production processing, but may also apply to other user needs, e.g. pipeline developers.

Offline Batch Production Service includes both the Campaign Management and Workload-/Workflow software products. The service executes science payloads as "campaigns" consisting of a defined sequence of pipelines, a defined configuration, and defined inputs and outputs. Many different payloads may be executed on many different campaign cadences. The service is able to handle massively distributed computing allocating work across multiple enclaves, in particular between NCSA and the Satellite Facility at CC-IN2P3, which will have capacity for half of the DRP processing. As the way science pipelines are executed in the BPS differs significantly from the way it is done by most science pipelines developers, the service needs to provide additional set of features for offline production processing.

This service use case document does not cover portions of what would be covered in complete Operations use cases, e.g., having change board approval (or pre-approved rules) prior to changing a configuration.

Many operations such as submissions, restarts, pauses, terminations, monitoring, etc., apply to single pipelines or groups of pipelines. To avoid repetition, this use case document groups them together unless explicitly specified otherwise.

2 Definitions

Pipeline step definition Data independent description of algorithmic work (in Gen3 Middleware nomenclature it is called a PipelineTask)

Pipeline A sequence of individual pipeline step definitions that performs particular data analysis and product generation (see LDM-151 for science definitions of pipelines).

Payload A sequence of pipelines to achieve an LSST objective, e.g. a sequence of pipelines

to be used for DRP (sometimes referred to as a “production”).

Campaign A payload with defined configuration and inputs to achieve a specific LSST objective, e.g., the DRP payload, specific configuration and specific set of inputs. A campaign describes *what* work is required to achieve that objective. Depending on its configuration, it corresponds to a set of workflows (see the definition below) which determine *how* the required work will be done.

Subcampaign A part of a campaign obtained via subsetting a payload and/or inputs.

Pipeline step instances A single instance of a pipeline step definition with specific inputs and outputs (in Gen3 Middleware, it is called a Quantum)

Job A unit of execution submitted via platform’s batch computing service. It may consist of one or more pipeline step instances.

Workflow A collection of jobs ordered by data dependencies represented as a direct, acyclic graph (in Gen3 Middleware, it is a result of applying an execution configuration to a QuantumGraph, grouping pipeline step instances into jobs).

3 Actors

Operator (OP) A person responsible for offline processing (e.g. DRP).

Campaign Manager (CM) A system or person managing processing campaigns.

Workflow Management System (WMS) An infrastructure for the set-up, execution and monitoring of a defined workflow

4 Pre-runtime

4.1 *PRE1*: Override default configuration options

The Operator wants to override a subset of configuration options for a campaign/pipeline at any configuration level (i.e., global, site, campaign, payload, pipeline step definition). These overrides include but are not limited to:

- execution configuration of a pipeline step definition (e.g., expected amount of memory needed),
- science configuration of a pipeline step definition (e.g., update particular threshold),
- globally override certain configurations (e.g., whether to bring all output datasets home from job including intermediates),
- specify to process campaign/payload/pipeline step definitions on particular computational platform(s) or providing a list of specific machine(s) to avoid (e.g., run on cluster at CC-IN2P3 but avoid specific machines cc3000 and cc4003).
- excluding certain data from a campaign.

As it was mentioned earlier, which changes of configuration are pre-approved and which require change board approval is beyond scope of this document. Regardless of the LSST operation change protocol, the Operator needs an ability to make approved configuration changes.

4.2 *PRE2*: Modify pipeline

The Operator modifies sequence of pipeline step definitions. Modification may include:

- deleting pipeline step definitions (e.g. only running half of a pipeline for debugging),
- adding new pipeline step definitions,
- reordering existing sequence of pipeline step definitions for science reasons.

4.3 *PRE3*: Select specific software version(s)

The Operator requests to run a campaign with specific versions of the LSST packages (e.g. wants to use a particular release of LSST Scientific Pipelines or specific versions of individual packages).

4.4 *PRE4*: Execute workflow in specific order

The Operator specifies the order in which the workflow should be executed including, but not limited to, breadth-first search and depth-first search.

4.5 *PRE5*: Support dynamic workflows

The workflow of certain pipelines may not be determinable at submit time, but require discovering what outputs were actually created at runtime for certain steps for determining number of instances of subsequent steps in the workflow. In this case, the Operator need to indicate where in the pipeline the dynamic generation is required. The dynamic generation is related to data discovery only. It does not include a support for conditional execution or rerunning jobs or parts of the workflow based on some quality metrics.

4.6 *PRE6*: Reduce number of compute jobs

The Operator configures the BPS to group portions of the workflow into smaller number of compute jobs to reduce the time overhead related to job startup costs.

4.7 *PRE7*: Halt or continue on failures

The Operator configures the BPS whether to stop or continue workflow execution based on failures of specific pipeline step instances.

4.8 *PRE8*: Save intermediate datasets

The Operator turns on/off the saving of intermediate datasets during submit time (e.g. for debugging purposes).

4.9 *PRE9*: Deal with unknown/defective data

The Operator configures the BPS to stage out any files that are not expected (i.e. not listed, are in a wrong place, have wrong name) or are missing some of required metadata.

4.10 *PRE10*: Prioritize (sub)campaigns

The Operator assigns different processing priorities for (sub)campaigns at time of submission.

4.11 *PRE11*: Initiate a campaign

The Operator specifies a campaign and submits it for processing.

5 Runtime

5.1 *RUN1*: Terminate a (sub)campaign

The Operator declares a running (sub)campaign to be a failure. The BPS should terminate its processing immediately.

5.2 *RUN2*: Pause a (sub)campaign

The Operator orders the BPS to stop dispatching new workflows/jobs associated with a selected running (sub)campaign either to free resources for a campaign with higher priority or due to site maintenance.

5.3 *RUN3*: Alter compute resources

The Operator assigns new compute resources to a (sub)campaign put on hold and orders the BPS to start dispatching related workflows/jobs. This does not include the ability to change other aspects of configuration for the (sub)campaign.

5.4 *RUN4*: Dispatch workflows/jobs to specific computational platform(s)

During submission time, the Operator provides a list of available compute sites. The BPS dispatches workflows/jobs to specified computational platforms based on workflow/job requirements. This includes Operator's ability to enforce execution of selected workflows/jobs

on a specific platform.

5.5 *RUN5*: Alter priorities of (sub)campaigns

The Operator needs to alter priorities of a (sub)campaign (it does not imply providing a support for eviction of running jobs with lower priorities).

5.6 *RUN6*: Use Pipeline Middleware

The Operator submits a (sub)campaign which requires current, stable version of the Pipeline Middleware (e.g. to chain a set of PipelineTasks in memory).

5.7 *RUN7*: Support optional inputs

Certain steps in a workflow can proceed even when not all declared inputs are present. The Operator wants the BPS to continue with processing for such steps providing (a) all predecessors completed their execution and (b) minimal requirements are satisfied.

5.8 *RUN8*: Retry defective steps/jobs

Based on the submit time configuration, the WMS automatically retries defective steps/jobs providing certain criteria are met (e.g. type of failure). The preference is to retry the minimal amount of workflow necessary, but leaving flexibility to work around implementation difficulties.

5.9 *RUN9*: Interact with the Data Backbone

For production payloads, the BPS retrieves the inputs from the Data Backbone and store the results there.

5.10 *RUN10*: Monitor available resources

The Operator needs to be able to monitor the status of the BPS service as well as any of its underpinning components (including any suspicions by BPS that a node is not fit for use).

5.11 *RUN11*: Monitor workflow execution

The Operator or the CM tracks workflow execution, i.e., monitors in real time metrics such as:

- number of pending, running, finished, and failed jobs/pipeline step instances;
- amount of computer resources (e.g. CPUs, memory, disk space) in use;
- job runtime information (e.g. host name, memory, wall/CPU time, data input and output volume).
- what pipeline step instance or framework step is currently running (if possible seeing stdout/stderr from the step) for when pipelines seem to be taking too long.

5.12 *RUN12*: Verify results integrity

Based on submit time configuration, the BPS verifies that all the required outputs were generated and staged out correctly for each successful workflow.

Note: Validation that the payload configuration and execution produced the correct results is outside of the scope of this use case.

5.13 *RUN13*: Notify about runtime events

Based on submit time configuration, the BPS sends notifications about campaign/payload level events (e.g. its failure, completion, jobs taking longer than some operator-specified threshold, etc.)

6 Post-runtime

6.1 *POST1*: Restart a campaign/payload

The Operator requests to restart processing a failed campaign/payload from the point as close as possible to the failure (i.e. not from the very beginning). Restarting a campaign/pay-

load may include one or more of the following: changing software stack to be used, changing science configuration, changing execution configuration, etc.

6.2 *POST2*: Designate point of restart

A workflow execution may not hard fail at the correct step and hard fail later or may seemly complete the entire pipeline but produce bad science results. An Operator may choose to either submit a new campaign/payload or restart payload at a failure point the operator chooses.

6.3 *POST3*: Investigate output datasets

The Operator uses similar procedures and tools to investigate both successful and failed campaign/payloads.

6.4 *POST4*: View runtime metrics and provenance

For each workflow step (including pipeline step instances, scheduling, data transfer, data loading, workflow generation, etc.), the Operator views/queries information such as:

- machine where it was executed,
- when it was running and time it took for the process to complete,
- amount of memory it used during its execution,
- version of software used,
- job's environment,
- what input datasets were used (where applicable),
- what output datasets were produced (where applicable).

6.5 *POST5*: Browse stdout/stderr/logs

The Operator analyzes stderr/stdout/logs finished workflows/jobs, i.e., needs to be able to quickly find the log of a specific pipeline step instance, search logs for certain keyword-

s/phrases.

6.6 *POST6*: Compare pipeline executions

The Operator compares multiple payload executions with regard to data products and run-time metrics (e.g. to evaluate the impact of a software change).

6.7 *POST7*: Summarize a campaign

The Operator views a summary of a campaign, e.g., a breakdown showing times, resource usage for individual pipelines (if possible, indicating pipelines deviating from a norm).