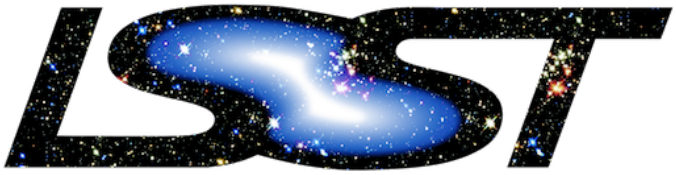

LARGE SYNOPTIC SURVEY TELESCOPE

Large Synoptic Survey Telescope (LSST)

LSST Level 1 System Test Specification

Eric C. Bellm, John D. Swinbank

LDM-533

Latest Revision: 2018-07-16

Abstract

This document describes the detailed test specification for the LSST Level 1 System.

Change Record

Version	Date	Description	Owner name
1.0	2018-01-11	Initial release of draft version.	Bellm, Swinbank
1.1	2018-01-11	Implementation of RFC-429	T. Jenness
	2018-07-13	Update document with latest test cases defined in Jira	G. Comoretto

Document curator: Eric C. Bellm

Document source location: `https://github.com/lsst/ldm-533`

Version from source repository: 5bbf2ca

Contents

LSST Level 1 System Test Specification

1 Introduction

This document specifies the test procedure for the LSST Level 1 System.

The LSST Level 1 System is the component of the LSST system which is responsible for scientific processing leading to:

- Single frame processing and measurement;
- Alert generation from difference image analysis;
- Alert distribution to community brokers;
- Simple filtering of alerts;
- Precovery and forced photometry measurements on new and previously-known sources found in difference imaging;
- Identification of moving objects.
- Generating QC metrics based on pipeline execution and post-processing of scientific data products.

A full description of this product is provided in §6 (which describes the Data Facility-provided execution services) and §13.1 (the science payloads) of LDM-148.

1.1 Objectives

This document builds on the description of LSST Data Management's approach to testing as described in LDM-503 to describe the detailed tests that will be performed on the LSST Level 1 System as part of the verification of the DM system.

It identifies test designs, test cases and procedures for the tests, and the pass/fail criteria for each test.

1.2 Scope

This document describes the test procedures for the following components of the LSST system (as described in LDM-148):

- Services provided by the LSST Data Facility:
  - Prompt Processing Execution
  - Batch and Offline Processing Execution
  - Level 1 Quality Control
  - Alert Distribution Execution
  - Alert Filtering Execution
- Science payloads:
  - Single frame processing Payload
  - Alert Generation Payload
  - Precovery and Forced Photometry Payload
  - MOPS Payload

1.3 Applicable Documents

LDM-148	LSST DM System Architecture
LDM-151	LSST DM Science Pipelines Design
LDM-294	LSST DM Organization & Management
LDM-502	The Measurement and Verification of DM Key Performance Metrics
LDM-503	LSST DM Test Plan
LSE-61	LSST DM Subsystem Requirements
LSE-163	LSST Data Products Definition Document

1.4 References

[1] **[LSE-61]**, Dubois-Felsmann, G., Jenness, T., 2017, *LSST Data Management Subsystem Requirements*, LSE-61, URL `https://ls.st/LSE-61`

[2] **[LPM-17]**, Ivezić, Ž., The LSST Science Collaboration, 2011, *LSST Science Requirements Document*, LPM-17, URL https://ls.st/LPM-17

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

[4] **[LDM-148]**, Lim, K.T., Bosch, J., Dubois-Felsmann, G., et al., 2017, *Data Management System Design*, LDM-148, URL https://ls.st/LDM-148

[5] **[LDM-502]**, Nidever, D., Economou, F., 2016, *The Measurement and Verification of DM Key Performance Metrics*, LDM-502, URL https://ls.st/LDM-502

[6] **[LDM-503]**, O'Mullane, W., Jurić, M., Economou, F., 2017, *Data Management Test Plan*, LDM-503, URL https://ls.st/LDM-503

[7] **[LDM-294]**, O'Mullane, W., Swinbank, J., Jurić, M., DMLT, 2017, *Data Management Organization and Management*, LDM-294, URL https://ls.st/LDM-294

[8] **[LDM-151]**, Swinbank, J.D., et al., 2017, *Data Management Science Pipelines Design*, LDM-151, URL https://ls.st/LDM-151

[9] **[LSE-63]**, Tyson, T., DQA Team, Science Collaboration, 2017, *Data quality Assurance Plan: Requirements for the LSST Data Quality Assessment Framework*, LSE-63, URL https://ls.st/LSE-63

2 Approach

The major activities to be performed are to:

- Compare the design of the Alert Production payload as implemented to the requirements on the outputs of the DM Subsystem as defined in LSE-63 and LSE-163 to demonstrate that all data products required by the scientific community will be delivered by the system as built.
- Ensure that all data products included in the AP payload design are correctly produced and persisted appropriately to the LSST Data Backbone, Alert Distribution System, and/or Alert Filtering service as appropriate.
- Ensure that all data products required by the Precovery and Forced Photometry payload are correctly produced and persisted appropriately to the LSST Data Backbone.

- Ensure that all data products required by the MOPS system are correctly produced and persisted appropriately to the LSST Data Backbone.
- Demonstrate that QC metrics are properly calculated and transmitted during the execution all L1 production types.
- Demonstrate that post-processing QC analysis of data products can be used to identify and report on failures or anomalies in the processing.

2.1 Tasks and criteria

The following are the major items under test:

- The science payload capable of prompt processing of single visit images;
- The Alert Generation payload that detects variable sources through difference image analysis;
- The Alert Distribution System that packages alerts and forwards them to community brokers;
- The filtering system that allows science users to apply simple filters to the alert stream;
- The Precovery and Forced Photometry payloads that measure flux levels for new and previously-known sources found in difference images;
- The Moving Object Processing System payload that identifies solar system bodies from difference image sources;
- Services capable of scheduling and managing the execution of all of the above payloads, marshalling their results, and making them available to other parts of the system for analysis or further distribution.

2.2 Features to be tested

- Code for the payloads described in §2.1 can be made available on systems managed by the LSST Data Facility;
- The payloads described in §2.1 can be executed under the control of appropriate LDF services;

- All required data products are persisted;
- Data products exhibit scientific fidelity, satisfying the requirements described in LSE-61.

2.3 Features not to be tested

This test specification does not extend to demonstrating the detailed compliance of LSST data products with all Science Requirements Document level requirements: such a demonstration would require carefully curated LSST-like datasets (or simulated data), a detailed understanding of the precursor observing system at the level required by LSST, LSST-like calibration products, etc., which are assumed not to be available for all test cases.

This document does not describe facilities for periodically generating or collecting key performance metrics (KPMs), except insofar as those KPMs are incidentally measured as part of executing the documented test cases. The KPMs and the system being used to track KPMs and to ensure compliance with documented requirements is described in LDM-502.

2.4 Pass/fail criteria

The results of all tests will be assessed using the criteria described in LDM-503 §4.

Note that, when executing pipelines, tasks or individual algorithms, any unexplained or unexpected errors or warnings appearing in the associated log or on screen output must be described in the documentation for the system under test. Any warning or error for which this is not the case must be filed as a software problem report and filed with the DMCCB.

2.5 Suspension criteria and resumption requirements

Refer to individual test cases where applicable.

2.6 Naming convention

With the introduction of the Jira Test Management, the following definitions have to be considered:

LVV : Is the label for the "LSST Verification and Validation" project in Jira where all information regarding tests are managed.

LVV-XXX : Are Verification Elements, where XXX is the Verification Element identifier. Each Verification Element is derived from a requirement and has at least one Test Case associated. There can be multiple Verification Elements associated with a requirement.

LVV-TYYY : Are Test Cases. Each Test Case is associated with a Verification Element, where YYY is the Test Case identifier. There can be multiple test cases associated with a Verification Element.

The old naming convention described below is now obsolete, but existing test cases can be encountered in old documents, and will keep the original identifier at the beginning of the name.

Tests were named according to the pattern PROD-XX-YY where:

PROD The product under test. Relevant entries for this document are:
AG The Alert Generation payload and associated service

XX Test specification number (in increments of 10)

YY Test case number (in increments of 5)

3 Test Cases Summary

Jira Id	Test Name
LVV-T17	AG-00-00: Installation of the Alert Generation science payload.
LVV-T18	AG-00-05: Alert Generation Produces Required Data Products
LVV-T19	AG-00-10: Scientific Verification of Processed Visit Images
LVV-T20	AG-00-15: Scientific Verification of Difference Images
LVV-T21	AG-00-20: Scientific Verification of DIASource Catalog
LVV-T22	AG-00-25: Scientific Verification of DIAObject Catalog
LVV-T216	Installation of the Alert Distribution payloads.
LVV-T217	Full Stream Alert Distribution
LVV-T218	Simple Filtering of the LSST Alert Stream

4 Test Cases

4.1 LVV-T17 - AG-00-00: Installation of the Alert Generation science payload.

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

4.1.1 Test Items

This test will check:

- That the Alert Generation science payload is available for distribution from documented channels;
- That the Alert Generation science payload can be installed on LSST Data Facility-managed systems.

4.1.2 Requirements

- LVV-139 - DMS-REQ-0308-V-01: Software Architecture to Enable Community Re-Use

4.1.3 Precondition

- Input specification

No input data is required for this test case.

4.1.4 Test Script

Step 1
Release 16.0 of the LSST Science Pipelines will be installed into the GPFS filesystem accessible at /software on lsst-dev01 following the instructions at `https://pipelines.lsst.io/install/newinstall.html` .

Step 2

The lsst_distrib top level package will be enabled:

```
source /software/lsstsw/stack3/loadLSST.bash
setup lsst_distrib
```

Step 3

The "LSST Stack Demo" package will be downloaded onto the test system from https://github.com/lsst/lsst_dr
and uncompressed.

Step 4

The demo package will be executed by following the instructions in its "README" file. The string "Ok." should be returned. Specifically, we execute:

```
setup obs_sdss
./bin/demo.sh
python bin/compare expected/Linux64/detected-sources.txt
```

Step 5

A shell on an LSST-VC compute node will now be obtained by executing:

```
$ srun -I –pty bash
```

Step 6

The demo package will be executed on the compute node and the same result obtained.

Step 7

The Alert Production datasets and packages are not yet part of lsst_distrib and so must be installed separately. They will be installed as follows on the GPFS filesystem:

```
setup git_lfs
```

git clone https://github.com/lsst/ap_verify_hits2015.git

export AP_VERIFY_HITS2015_DIR=$PWD/ap_verify_hits2015 cd $AP_VERIFY_HITS2015_DIR
setup -r .
cd-

setup obs_decam
git clone https://github.com/lsst-dm/ap_association
cd ap_association
setup -k -r .
scons
cd-

git clone https://github.com/lsst-dm/ap_pipe
cd ap_pipe
setup -k -r .
scons
cd-

git clone https://github.com/lsst-dm/ap_verify
cd ap_verify
setup -k -r .
scons
cd-

and any errors or failures reported.

4.2 LVV-T18 - AG-00-05: Alert Generation Produces Required Data Products

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

Version	Status	Priority	Verification Type	Critical Event	Owner

4.2.1 Test Items

This test will check that the basic data products produced by Alert Generation are generated by execution of the science payload.
These products will include:

- Processed visit images (PVIs; DMS-REQ-0069); • Difference Exposures (DMS-REQ-0010);
- DIASource catalogs (DMS-REQ-0269);
- DIAObject catalogs (DMS-REQ-0271);

4.2.2 Requirements

- LVV-29 - DMS-REQ-0069-V-01: Processed Visit Images
- LVV-7 - DMS-REQ-0010-V-01: Difference Exposures
- LVV-100 - DMS-REQ-0269-V-01: DIASource Catalog
- LVV-102 - DMS-REQ-0271-V-01: DIAObject Catalog

4.2.3 Precondition

- Input specification

A complete processing of the DECam “HiTS” dataset, as defined at https://dmtn-039.lsst.io/ and https://github.com/lsst/ap_verify_hits2015, through the Alert Generation science payload. This dataset shall be made available in a standard LSST data repository, accessible via the “Data Butler”.
It is not required that all combinations of visit and CCD have been processed successfully: a number of failures are expected. However, documentation to describe processing failures should be provided.

4.2.4 Test Script

Step 1
The DM Stack and Alert Processing packaged shall be initialized as described in LVT-T17 (AG-00-00).

Step 2
The alert generation processing will be executed using the verification cluster:

“‘bash
python ap_verify/bin/prepare_demo_slurm_files.py
At present we must run a single ccd+visit to handle ingestion before
parallel processing can begin
./ap_verify/bin/exec_demo_run_1ccd.sh 410915 25
ln -s ap_verify/bin/demo_run.sl
ln -s ap_verify/bin/demo_cmds.conf
sbatch demo_run.sl
“‘

and any errors or failures reported.

Step 3
A “Data Butler” will be initialized to access the repository.

Step 4
For each of the expected data products types (listed in §4.2.2) and each of the expected units (PVIs, catalogs, etc.), the data product will be retrieved from the Butler and verified to be non-empty.

Step 5

DIAObjects are currently only stored in a database, without shims to the Butler, so the existence of the database table and its non-empty contents will be verified by directly accessing it using sqlite3 and executing appropriate SQL queries.

4.3 LVV-T19 - AG-00-10: Scientific Verification of Processed Visit Images

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

4.3.1 Test Items

This test will check that the Processed Visit Images (PVIs) delivered by the alert generation science payload meet the requirements laid down by LSE-61.
Specifically, this will demonstrate that:

- Processed visit images have been generated and persisted during payload execution;
- Each PVI includes a science pixel array, a mask array, and a variance array. (DMS-REQ-0072).
- Each PVI includes a background model (DMS-REQ-0327), photometric zero-point (DMS-REQ-0029), spatially-varying PSF (DMS-REQ-0070) and WCS (DMS-REQ-0030).
- Saturated pixels are correctly masked.
- Pixels affected by cosmic rays are correctly masked.
- The background is not oversubtracted around bright objects.

This test does not include quantitative targets for the science quality criteria.

4.3.2 Requirements

- LVV-29 - DMS-REQ-0069-V-01: Processed Visit Images
- LVV-158 - DMS-REQ-0327-V-01: Background Model Calculation
- LVV-12 - DMS-REQ-0029-V-01: Generate Photometric Zeropoint for Visit Image
- LVV-30 - DMS-REQ-0070-V-01: Generate PSF for Visit Images

- LVV-13 - DMS-REQ-0030-V-01: Generate WCS for Visit Images
- LVV-31 - DMS-REQ-0072-V-01: Processed Visit Image Content

4.3.3 Precondition

Input specification

A complete processing of the DECam "HiTS" dataset, as defined at https://dmtn-039.lsst.io/ and https://github.com/lsst/ap_verify_hits2015, through the Alert Generation science payload. This dataset shall be made available in a standard LSST data repository, accessible via the "Data Butler".
It is not required that all combinations of visit and CCD have been processed successfully: a number of failures are expected. However, documentation to describe processing failures should be provided.

4.3.4 Test Script

Step 1
The DM Stack shall be initialized using the loadLSST script (as described in LVV-T17 - AG-00-00).

Step 2
A "Data Butler" will be initialized to access the repository.

Step 3
For each processed CCD, the PVI will be retrieved from the Butler, and the existence of all components described in §4.3.2 will be verified.

Step 4
Five sensors will be chosen at random from each of two visits and inspected by eye for unmasked artifacts.

4.4 LVV-T20 - AG-00-15: Scientific Verification of Difference Images

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

4.4.1 Test Items

This test will check that the difference images delivered by the Alert Generation science payload meet the requirements laid down by LSE-61.
Specifically, this will demonstrate that:

- Difference images have been generated and persisted during payload execution;
- Each difference image includes information about the identity of the input exposures, and metadata such as a representation of the PSF matching kernel (DMS-REQ-0074);
- Masks are correctly propagated from the input images.

This test does not include quantitative targets for the science quality criteria.

4.4.2 Requirements

- LVV-7 - DMS-REQ-0010-V-01: Difference Exposures
- LVV-32 - DMS-REQ-0074-V-01: Difference Exposure Attributes

4.4.3 Precondition

- Input specification

A complete processing of the DECam “HiTS” dataset, as defined at https://dmtn-039.lsst. io/ and https://github.com/lsst/ap_verify_hits2015, through the Alert Generation science payload. This dataset shall be made available in a standard LSST data repository, accessible via the “Data Butler”.

It is not required that all combinations of visit and CCD have been processed successfully: a number of failures are expected. However, documentation to describe processing failures should be provided.

4.4.4 Test Script

Step 1
The DM Stack shall be initialized using the loadLSST script (as described in LVV-T-17 AG-00-00).

Step 2
A "Data Butler" will be initialized to access the repository.

Step 3
For each processed CCD, the difference image will be retrieved from the Butler, and the existence of all components described in §4.4.2 will be verified.

Step 4
Five sensors will be chosen at random from each of two visits and the masks of the input and difference images compared by eye.

4.5 LVV-T21 - AG-00-20: Scientific Verification of DIASource Catalog

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

4.5.1 Test Items

This test will check that the difference image source catalogs delivered by the Alert Generation science payload meet the requirements laid down by LSE-61.

- Specifically, this will demonstrate that:
- Measurements in the catalog are presented in flux units (DMS-REQ-0347);
- Each DIASource record contains an appropriate subset of the attributes required by DMS-REQ-0269. In particular, the LDM-503-3-era pipeline is expected to provide DIASource positions (sky and focal plane), fluxes, and flags indicative of issues encountered during processing.
- Faint DIASources satisfying additional criteria are stored (DMS-REQ-0270).
- Derived quantities are provided in pre-computed columns (DMS-REQ-0331);

This test does not include quantitative targets for the science quality criteria.

4.5.2 Requirements

- LVV-100 - DMS-REQ-0269-V-01: DIASource Catalog
- LVV-101 - DMS-REQ-0270-V-01: Faint DIASource Measurements
- LVV-178 - DMS-REQ-0347-V-01: Measurements in catalogs
- LVV-162 - DMS-REQ-0331-V-01: Computing Derived Quantities

4.5.3 Precondition

- Input specification

A complete processing of the DECam “HiTS” dataset, as defined at https://dmtn-039.lsst. io/ and https://github.com/lsst/ap_verify_hits2015, through the Alert Generation science payload. This dataset shall be made available in a standard LSST data repository, accessible via the “Data Butler”.
It is not required that all combinations of visit and CCD have been processed successfully: a number of failures are expected. However, documentation to describe processing failures should be provided.

4.5.4 Test Script

Step 1
The DM Stack shall be initialized using the loadLSST script (as described in LVV-T17 - AG-00-00).

Step 2
A "Data Butler" will be initialized to access the repository.

Step 3
DIASource records will be accessed by querying the Butler, then examined interactively at a Python prompt.

4.6 LVV-T22 - AG-00-25: Scientific Verification of DIAObject Catalog

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Approved	Normal	Test	False	Eric Bellm

4.6.1 Test Items

This test will check that the DIAObject catalogs delivered by the Alert Generation science payload meet the requirements laid down by LSE-61.
Specifically, this will demonstrate that:

- DIAObjects are recorded with unique identifiers (DMS-REQ-0271);
- Measurements in the catalog are presented in flux units (DMS-REQ-0347);
- EachDIAObjectrecordcontainscontainsanappropriatesetofsummaryattributes(DMS- REQ-0271 and DMS-REQ-0272). Note:
  - This test is executed independently of the Data Release Production system. Hence, DIAObjects are not associated to Objects, and the association metadata specified

by DMS-REQ-0271 is not expected to be available.
  - TheLDM-503-3erapipelineisnotexpectedtocalculateorpersistallattributesspec- ified by DMS-REQ-0272 requirement.
- Relevant derived quantities are provided in pre-computed columns (DMS-REQ-0331);

This test does not include quantitative targets for the science quality criteria.

4.6.2 Requirements

- LVV-116 - DMS-REQ-0285-V-01: Level 1 Source Association
- LVV-102 - DMS-REQ-0271-V-01: DIAObject Catalog
- LVV-103 - DMS-REQ-0272-V-01: DIAObject Attributes
- LVV-178 - DMS-REQ-0347-V-01: Measurements in catalogs
- LVV-162 - DMS-REQ-0331-V-01: Computing Derived Quantities

4.6.3 Precondition

Input specification

A complete processing of the DECam "HiTS" dataset, as defined at https://dmtn-039.lsst. io/ and https://github.com/lsst/ap_verify_hits2015, through the Alert Generation science payload. This dataset shall be made available in a standard LSST data repository, accessible via the "Data Butler".
It is not required that all combinations of visit and CCD have been processed successfully: a number of failures are expected. However, documentation to describe processing failures should be provided.

4.6.4 Test Script

Step 1
The DM Stack shall be initialized using the loadLSST script (as described in LVV-T17 - AG-00-00).

Step 2

sqlite3 or Python's sqlalchemy module will be used to access the Level 1 database.

4.7 LVV-T216 - Installation of the Alert Distribution payloads.

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Draft	Normal	Test	False	Eric Bellm

4.7.1 Test Items

This test will check:

- That the Alert Distribution payloads are available from documented channels.
- That the Alert Distribution payloads can be installed on LSST Data Facility-managed systems.
- That the Alert Distribution payloads can be executed by LSST Data Facility-managed systems.

4.7.2 Requirements

- LVV-139 - DMS-REQ-0308-V-01: Software Architecture to Enable Community Re-Use

4.7.3 Test Script

Step 1
Download Kafka Docker image from https://github.com/lsst-dm/alert_stream.

Step 2

Change to the alert_stream directory and build the docker image.

```
docker build -t "lsst-kub001:5000/alert_stream" .
```

Step 3
Register it with Kubernetes

docker push lsst-kub001:5000/alert_stream

Step 4
From the alert_stream/kubernetes directory, start Kafka and Zookeeper:

```
kubectl create -f zookeeper-service.yaml
kubectl create -f zookeeper-deployment.yaml
kubectl create -f kafka-deployment.yaml
kubectl create -f kafka-service.yaml
```

(use kubectl get pods/services between each command to check status; wait until each is "Running" before starting the next command)

Step 5
Confirm Kafka and Zookeeper are listed when running

kubectl get pods

and

kubectl get services

4.8 LVV-T217 - Full Stream Alert Distribution

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Draft	Normal	Test	False	Eric Bellm

4.8.1 Test Items

This test will check that the full stream of LSST alerts can be distributed to end users.

Specifically, this will demonstrate that:

- Serialized alert packets can be loaded into the alert distribution system at LSST-relevant scales (10,000 alerts every 39 seconds);
- Alert packets can be retrieved from the queue system at LSST-relevant scales.

4.8.2 Requirements

- LVV-3 - DMS-REQ-0002-V-01: Transient Alert Distribution

4.8.3 Precondition

Input data: A sample of Avro-formatted alert packets.

4.8.4 Test Script

Step 1
Download Kafka Docker image from https://github.com/lsst-dm/alert_stream.

Step 2
Change to the alert_stream directory and build the docker image.

```
docker build -t "lsst-kub001:5000/alert_stream" .
```

Step 3
From the alert_stream/kubernetes directory, start Kafka and Zookeeper:

```
kubectl create -f zookeeper-service.yaml
kubectl create -f zookeeper-deployment.yaml
kubectl create -f kafka-deployment.yaml
kubectl create -f kafka-service.yaml
```

(use kubectl get pods/services between each command to check status; wait until each is "Running" before starting the next command)

Step 4
Confirm Kafka and Zookeeper are listed when running

kubectl get pods

and

kubectl get services

Step 5
Register it with Kubernetes

docker push lsst-kub001:5000/alert_stream

Step 6
Start a consumer that monitors the full stream and logs a deserialized version of every Nth packet:

```
kubectl create -f consumerall-deployment.yaml
```

Step 7

```
Start a producer that reads alert packets from disk and loads them into the Kafka queue:
```

```
kubectl create -f sender-deployment.yaml
```

Step 8
Determine the name of the alert sender pod with

kubectl get pods

Examine output log files.

kubectl logs <pod name>

Verify that alerts are being sent within 40 seconds by subtracting the timing measurements.

Step 9
Determine the name of the consumer pod with

kubectl get pods

Examine output log files.

kubectl logs <pod name>

The packet log should show deserialized alert packets with contents matching the input packets.

4.9 LVV-T218 - Simple Filtering of the LSST Alert Stream

Version	Status	Priority	Verification Type	Critical Event	Owner
1	Draft	Normal	Test	False	Eric Bellm

4.9.1 Test Items

This test will demonstrate the “mini-broker” filtering service that returns a subset of alerts from the full stream identified by user-provided filters.

Specifically, this will demonstrate that:

- The filtering service can retrieve alerts from the full alert stream and filter them according to their contents;
- The filtered subset can be delivered to science users.

4.9.2 Requirements

- LVV-173 - DMS-REQ-0342-V-01: Alert Filtering Service
- LVV-179 - DMS-REQ-0348-V-01: Pre-defined alert filters
- LVV-174 - DMS-REQ-0343-V-01: Performance Requirements for LSST Alert Filtering Service

4.9.3 Precondition

Input data: A sample of Avro-formatted alert packets derived from LSST simulations corresponding to one night of simulated LSST observing.

4.9.4 Test Script

Step 1
Download Kafka Docker image from https://github.com/lsst-dm/alert_stream.

Step 2
Change to the alert_stream directory and build the docker image.

```
docker build -t "lsst-kub001:5000/alert_stream" .
```

Step 3

From the alert_stream/kubernetes directory, start Kafka and Zookeeper:

```
kubectl create -f zookeeper-service.yaml
kubectl create -f zookeeper-deployment.yaml
kubectl create -f kafka-deployment.yaml
kubectl create -f kafka-service.yaml
```

(use kubectl get pods/services between each command to check status; wait until each is "Running" before starting the next command)

Step 4

Confirm Kafka and Zookeeper are listed when running

kubectl get pods

and

kubectl get services

Step 5

Register it with Kubernetes

docker push lsst-kub001:5000/alert_stream

Step 6

Start 100 consumers that consume the filtered streams and logs a deserialized version of every Nth packet:

```
kubectl create -f consumer1-deployment.yaml
kubectl create -f consumer2-deployment.yaml
kubectl create -f consumer3-deployment.yaml
kubectl create -f consumer4-deployment.yaml
kubectl create -f consumer5-deployment.yaml
kubectl create -f consumer6-deployment.yaml
kubectl create -f consumer7-deployment.yaml
kubectl create -f consumer8-deployment.yaml
kubectl create -f consumer9-deployment.yaml
kubectl create -f consumer10-deployment.yaml
```

Step 7

Start 5 filter groups:

```
kubectl create -f filterer1-deployment.yaml
kubectl create -f filterer2-deployment.yaml
kubectl create -f filterer3-deployment.yaml
kubectl create -f filterer4-deployment.yaml
kubectl create -f filterer5-deployment.yaml
```

Step 8

Start a producer that reads alert packets from disk and loads them into the Kafka queue:

```
kubectl create -f sender-deployment.yaml
```

Step 9
Determine the name of the alert sender pod with

kubectl get pods

Examine output log files.

kubectl logs <pod name>

Verify that alerts are being sent within 40 seconds by subtracting the timing measurements.

Step 10
Determine the name of the consumer pods with

kubectl get pods

Examine output log files.

kubectl logs <pod name>

The packet log should show deserialized alert packets with contents matching the input packets.

A Requirements Traceability

Requirements	Test Cases
LVV-3 - DMS-REQ-0002-V-01: Transient Alert Distribution	LVV-T217
LVV-7 - DMS-REQ-0010-V-01: Difference Exposures	LVV-T18, LVV-T20
LVV-12 - DMS-REQ-0029-V-01: Generate Photometric Zeropoint for Visit Image	LVV-T19
LVV-13 - DMS-REQ-0030-V-01: Generate WCS for Visit Images	LVV-T19
LVV-29 - DMS-REQ-0069-V-01: Processed Visit Images	LVV-T18, LVV-T19
LVV-30 - DMS-REQ-0070-V-01: Generate PSF for Visit Images	LVV-T19
LVV-31 - DMS-REQ-0072-V-01: Processed Visit Image Content	LVV-T19
LVV-32 - DMS-REQ-0074-V-01: Difference Exposure Attributes	LVV-T20
LVV-100 - DMS-REQ-0269-V-01: DIASource Catalog	LVV-T18, LVV-T21
LVV-101 - DMS-REQ-0270-V-01: Faint DIASource Measurements	LVV-T21
LVV-102 - DMS-REQ-0271-V-01: DIAObject Catalog	LVV-T18, LVV-T22
LVV-103 - DMS-REQ-0272-V-01: DIAObject Attributes	LVV-T22
LVV-116 - DMS-REQ-0285-V-01: Level 1 Source Association	LVV-T22
LVV-139 - DMS-REQ-0308-V-01: Software Architecture to Enable Community Re-Use	LVV-T17, LVV-T216
LVV-158 - DMS-REQ-0327-V-01: Background Model Calculation	LVV-T19
LVV-162 - DMS-REQ-0331-V-01: Computing Derived Quantities	LVV-T21, LVV-T22
LVV-173 - DMS-REQ-0342-V-01: Alert Filtering Service	LVV-T218
LVV-174 - DMS-REQ-0343-V-01: Performance Requirements for LSST Alert Filtering Service	LVV-T218
LVV-178 - DMS-REQ-0347-V-01: Measurements in catalogs	LVV-T21, LVV-T22
LVV-179 - DMS-REQ-0348-V-01: Pre-defined alert filters	LVV-T218

B The DECam "'HiTS" dataset

We use a subset of the DECam hits dataset, contained in the repository `https://github.com/lsst/ap_verify_hits2015.git`. As described in `https://dmtn-039.lsst.io/`, we select HiTS fields `Blind15A_26`, `Blind15A_40`, and `Blind15A_42`. We construct templates from the best-seeing observations of same region of sky using the previous year's observations, labelled `Blind14A_04`, `Blind14A_10`, and `Blind14A_09`.

The specific visits we use are:

410915, 410929, 410931, 410971, 410985, 410987, 411021, 411035, 411037, 411055, 411069, 411071, 411255, 411269, 411271 , 411305, 411319, 411321, 411355, 411369, 411371, 411406, 411420, 411422 , 411456, 411470, 411472, 411657, 411671, 411673, 411707, 411721, 411724, 411758, 411772, 411774, 411808, 411822, 411824, 411858, 411872, 411874 , 412060, 412074, 412076, 412250, 412264, 412266, 412307, 412321, 412324, 412504, 412518, 412520, 412554, 412568, 412570, 412604, 412618, 412620, 412654, 412668, 412670, 412704, 412718, 412720, 413635, 413649, 413651, 413680, 413694, 413696, 415314, 415328, 415330, 415364, 415378, 415380, 419791, 419802, 419804, 421590, 421604, 421606.

For each visit we exclude CCDs 1, 2, and 61, leaving CCDs 3-60 and 62. We use g-band only for these tests due to the need to build templates.