Open Location Platform - Logs, Monitoring, and Alerts

User Guide

Version 1.5.2

Legal Notices

Trademark Acknowledgements

Disclaimer

Document Information

Product	
Name:	Open Location Platform - Logs, Monitoring, and Alerts
Version:	Version 1.5.2

Document	
Name:	Open Location Platform - Logs, Monitoring, and Alerts User’s Guide
ID:	548cbb8-1536627726-41db9917
Status:	FINAL
Date:	2018-09-11T01:02:33.317Z

Table of Contents

Introduction

Why Use Logs, Metrics, and Alerts

Your usage of OLP generates operational logs and metrics that you can use to make sense of what's happening with your processes. OLP provides a managed instance of the open source Grafana tool for managing your metrics dashboard and alerts. In addition, you can use the managed instance of the Splunk log analytics environment to search your applciation logs and save the results as reports. Dashboards, alerts, and logs are visible to all users in your account.

Use Pipeline Logging

Pipeline Logging Basics

OLP Pipelines use logging to provide more details during their operation. Different levels of logging are available for different purposes. OLP pipelines support the following levels of logging:

- **Debug** — Includes fine-grained informational events that are most useful to troubleshoot a pipeline.
- **Info** — Includes informational messages that highlight the progress of the pipeline at a coarse-grained level.
- **Warn** — Includes information on potentially harmful situations; including other run-time situations that are undesirable or unexpected, but not necessarily "wrong".
- **Error** — Includes other run-time errors or unexpected conditions such as error events that might still allow the pipeline to continue running.

Note

Pipelines have a default logging level of `warn`.

Tools

Event Logging is handled by an embedded version of Splunk. In general, you will not need to be concerned because Splunk runs in the background. However, you can create a new Splunk dashboard or run reports on the Portal.

See also

- Pipeline API Reference
- Pipeline Logging via CLI
- Grafana User Documentation
- Splunk Enterprise 6.5.2 User Documentation

Change the Logging Level

The Logging Level is set for a Pipeline Version. All the Jobs use the logging level associated with the corresponding Pipeline Version.

The Logging Level can be set at the **root** level for the entire pipeline and/or at the individual **logger** level for a pipeline class. Due to operational latency, it takes a few minutes for the changes to take effect. This may delay the availability of the logs at the new level in Splunk.

Note

The displayed Pipeline Version details include either the last logging level modified by the user or the default logging level (i.e. `warn`).

The logging level for a Pipeline Version can be changed only when it is in one of the following states:

- **Running** — For a Pipeline Version in a Running state, when the logging-level is changed, the system will change the logging-level of the **currently running job**.
- **Ready or Scheduled** — For a Pipeline Version in Ready or Scheduled state, when the logging-level is changed, the system will run the **future** jobs using the new logging-level.
- **Paused** — For a Pipeline Version in the Paused state, when the logging-level is changed and Pipeline Version is resumed, the system will run the **future** job using the new logging-level.

API Solution

To change a Pipeline Version's log-level to DEBUG at the root-level, use the following command from the REST API:

```
PUT /v2/pipelines/{pipelineId}/versions/{versionId}/logging-configuration
```

with the following body:

```
{
 "configuration": {
 "loggers": {
  "root": {
  "level": "debug"
 }
  }
 }
}
```

For more information, see the Pipeline API Reference.

CLI Solution

To change a Pipeline Version's log-level to DEBUG at the root-level, use the following command from the OLP CLI:

```
olp pipeline version log set-level <pipeline-id> <pipeline-version-id> [command parameters]
```

Example

To change the logging level of a pipeline version to DEBUG at the root-level:

```
olp pipeline version log set-level f2fc50c4-a0ac-4c8a-9637-0d9b3a0d4a96 d77f288e-2c89-4c94-b4ba-79fbd1e26e79 --
root DEBUG --json
```

On success, the command returns something like the following.

```
{"configuration": {"loggers": {
    "root": {"level": "debug"}
}}}
```

In this case the pipeline service confirms that the log-level has been set to `debug`.

Retrieve the Logging Level

To check the current log-level configuration, you need to retrieve the log-level setting for a specific Pipeline Version.

API Solution

To retrieve Current Logging Level via API, use the following command from the REST API:

```
GET /v2/pipelines/{pipelineId}/versions/{versionId}/logging-configuration
```

For more information, see the Pipeline API Reference.

CLI Solution

The retrieve Current Logging Level using the OLP CLI, use the following command:

```
olp pipeline version log get-level <pipeline-id> <pipeline-version-id> [command parameters]
```

Example

To retrieve the current logging level.

```
olp pipeline version log get-level f2fc50c4-a0ac-4c8a-9637-0d9b3a0d4a96 d77f288e-2c89-4c94-b4ba-79fbd1e26e79 --
json
```

On success, this command returns something like the following:

```
{"configuration": {"loggers": {
    "logger": [
        {
            "level": "info",
            "name": "com.example.someLoggingClass"
        },
        {
            "level": "info",
            "name": "com.example.examplepkg.otherClass"
        }
    ],
    "root": {"level": "error"}
}}}
```

Find Pipeline Logs

Each running Pipeline Version has a unique URL where the logs for that pipeline are stored. When using the CLI, this URL information is supplied by the pipeline service whenever a pipeline version is activated, upgraded, or when a status request is submitted to the service. But if you are running your pipeline by interfacing directly with the Pipeline Service's REST API, you will need to make a status request to discover the logging URL.

Example

The CLI `show` function is one way you can check status. In the case of a specific job, you can use the following CLI command.

```
olp pipeline version job show f2fc50c4-a0ac-4c8a-9637-0d9b3a0d4a96 d77f288e-2c89-4c94-b4ba-79fbd1e26e79 e8e3107
0-4f4b-4e4b-a8e0-40425844cb75
```

Successful output returns:

```
{
    "catalogVersions": null,
    "created": "2018-03-01T15:23:04.618Z",
    "id": "e8e31070-4f4b-4e4b-a8e0-40425844cb75",
    "state": "failed",
    "updated": "2018-03-01T15:23:32.497Z",
    "loggingUrl": null
}
```

In this case, the `loggingUrl` parameter has a null value. This is what you would expect to find for a job that is not yet running. Once the job runs, the logging URL would be shown.

Note

Until a scheduled Pipeline runs, the logging URL will remain `null`.

Search Application Logs

Application logs can be accessed through Splunk. To start a new search, click **Tools** > **Logs (Powered by Splunk)**. Then from the Splunk Homepage interface, click **Search**.

For more information on using Splunk, see the tutorials in the official documentation.

Pipeline Log Index

Pipeline logs are stored in the `<realm>_common` index. For example, if your account is in the `olp-example` realm, your index would be `olp-example_common` . You can search for this by adding the following string to your Splunk search query:

```
index="olp-example_common"
```

Troubleshoot Pipeline Issues

You can use application logs to debug and troubleshoot pipeline issues.

During the pipeline setup (before it is submitted to be run), the pipeline service does not produce any logs. Any feedback is provided only through the Pipeline service APIs.

Once a pipeline job is submitted, the errors can happen in two different scnearios: before and after the pipeline has started running.

Before the Pipeline Starts Running

There are a number of steps that the pipeline goes through before it actually starts running. Any errors at this point are not attached to a job. These errors are first captured internally by OLP, and then pushed out to your appropriate Splunk index. Only error logs related to your pipeline can appear in your specific Splunk index.

To search your Splunk index for these logs, use the pipeline or pipeline version UUID.

Once the Pipeline Starts Running

Once the pipeline starts running, there is a corresponding job reported in **Pipeline Version Details** on the OLP web portal and Pipelines API responses. Each pipeline job is assigned a link to the logs for that job run.

Note: Time Window Expiration

The default time window built into the link expires after some time. For example, the default link may be pulling logs from the last 15 minutes. This link does not work on the next day.

To search for these logs again, select a time window using Splunk. You can do this through a drop-down list option at the top right of the Splunk UI.

Monitor Your Metrics

OLP provides a managed instance of the open source Grafana tool for managing your metrics dashboards and alerts. To access these dashboards, click on the **Monitoring and Alerts** link under the **Tools** menu. Then in the Grafana tool, click on the **Home** button on top left to view the metric dashboards offered by OLP.

Time Ranges and Intervals

You can select the dashboard time range by using the dropdown menu in the upper-right corner of a Grafana dashboard. This controls the range of time over which your dashboard widgets and graphs query for their data.

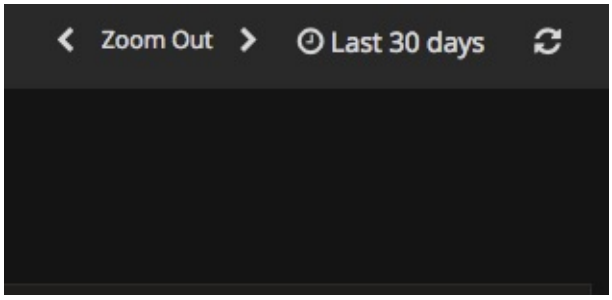

Figure: Grafana menu to select a time range

When a new time range is selected, Grafana automatically calculates the optimal display time interval (such as 1 day or 8 hours) for you. This is because when there is more data than can be shown, Grafana can make a more efficient display by grouping data points in larger intervals.

This auto-calculated interval may not be desired for all dashboards. For dashboards that are labeled "w/ Custom Time Interval", you can use the **Custom Time Interval** dropdown menu in the upper-left corner to select your own.

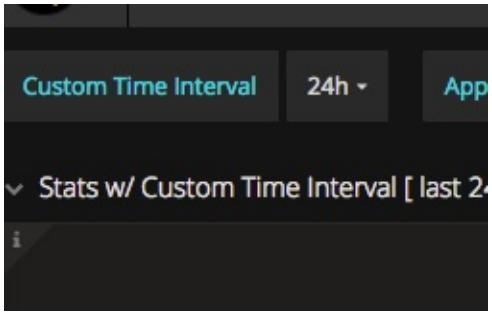

Figure: Grafana menu to select a time interval

Data, Catalog, and Layer Metrics

Data Metrics

METRIC	DESCRIPTION
Total Number Of Catalogs	Sum Of Current Existing Catalogs
Total Number of Layers	Sum Of Current Existing Layers (All Layer Types) Across all Catalogs
Total Number of Versioned Layers	Sum Of Current Versioned Layers Across all Catalogs
Total Number of Volatile Layers	Sum Of Current Volatile Layers Across all Catalogs
Total Number of Stream Layers	Sum Of Current Volatile Layers Across all Catalogs
Versioned Layers - Total Storage Volume (Usage)	Sum Of Data Volume Across all Versioned Layers (all Catalogs)
Volatile Layers - Total Capacity	Sum Of Reserved Capacity Across All Volatile Layers (all Catalogs)
Volatile Layers - Used Capacity	Sum Of Used Capacity Across All Volatile Layers (all Catalogs)
Stream Layers - Total Throughput Capacity	Sum Of Throughput Capacity Across all Stream Layers (all Catalogs)

Catalog Metrics

METRIC	DESCRIPTION
Layer Count Per Catalog	Total Layer Count Per Catalog
Versioned Layer Storage Volume Per Catalog	Total Volume (in GB) Of all Versioned Layers in this Catalog
Volatile Layer Unique Capacity Per Catalog	Total Unique Capacity (Amount of Stored Data Without Replication) Of all Volatile Layers in this Catalog
Attempted Commits Per Catalog (Versioned Storage Only)	Any change to a versioned catalog which triggers a version increment. Attempted means at least started.

Commit Metrics

METRIC	DESCRIPTION
Successfully Completed Commits Per Catalog (Versioned Storage Only)	Successful commits to the Catalog where a new version is published in the Metadata service
Canceled Commits Per Catalog (Versioned Storage Only)	User initiated canceled commits per Catalog

Failed Commits Per Catalog (Versioned Storage Only)	Incomplete commits to a Catalog - Change batch committed but job not finished. Failure reason examples: User Errors, 500 Errors, Etc. Does not include canceled commits.

Volatile Layer Metrics

METRIC	DESCRIPTION
Capacity Allocated for Volatile Storage	Total capacity configured in GB
Capacity Used for Volatile Storage	Total capacity used in GB
Unique Capacity Used for Volatile Storage	Amount of stored data without replication

Stream Layer Metrics

METRIC	DESCRIPTION
TTL per Stream Layer	"Time To Live (TTL)" user configuration setting per Stream Layer
Throughput per Stream Layer	IN/OUT throughput (Bytes/Second) per Stream Layer

Versioned Layer Metrics

METRIC	DESCRIPTION
Durable Storage Used for Versioned Layer	Total capacity used for Versioned Layer in the catalogs

Ingestion Metrics

METRIC	DESCRIPTION
Received Single Requests	Number of requests with a single message request body.
Failed Single Requests	Out of "Total Received Single Message Requests", the number of requests which returned a non-success status code.
Multi-Message Requests	Number of requests a multi message request body.
Messages successfully processed from Received Multi-Message Requests	Number of messages contained in multi-message requests which have been successfully received and processed.
Failed Multi-Message Requests	Out of "Total Received Multi-message Requests", the number of requests which returned a non-success status code.
Mean Ingestion Requests	Average rate of requests sent to the ingestion service per second.
Total Bytes IN per Provider (valid messages only)	Total GB Ingested (inclusive of all message types)

Flink Metrics

The following metrics are available for Flink pipelines. Check the official Flink documentation for more information about Flink metrics.

CPU/Memory Metrics

NAME	METRIC	DESCRIPTION
Max `Status_JVM_CPU_Load`	`cpu_load`	The recent CPU usage of the JVM
Max `Status_JVM_Memory_Heap_Used`	`heap_used`	The amount of heap memory currently used.
Min `Status_JVM_Memory_Heap_Committed`	`heap_committed`	The amount of heap memory guaranteed to be available to the JVM.
Min `Status_JVM_Memory_Heap_Max`	`heap_max`	The maximum amount of heap memory that can be used for memory management.
Max `Status_JVM_Memory_NonHeap_Used`	`nonheap_used`	The amount of non-heap memory currently used.
Max `Status_JVM_Memory_NonHeap_Committed`	`nonheap_committed`	The amount of non-heap memory guaranteed to be available to the JVM.
Max `Status_JVM_Memory_NonHeap_Max`	`nonheap_max`	The maximum amount of non-heap memory that can be used for memory management
Current `Status_JVM_Memory_Direct_Count`	`direct_count`	The number of buffers in the direct buffer pool.
Max `Status_JVM_Memory_Direct_MemoryUsed`	`direct_memoryused`	The amount of memory used by the JVM for the direct buffer pool.
Current `Status_JVM_Memory_Direct_TotalCapacity`	`direct_totalcapacity`	The total capacity of all buffers in the direct buffer pool.
Current `Status_JVM_Memory_Mapped_Count`	`mapped_count`	The number of buffers in the mapped buffer pool.
Max `Status_JVM_Memory_Mapped_MemoryUsed`	`mapped_memoryused`	The amount of memory used by the JVM for the mapped buffer pool.
Max `Status_JVM_Memory_Mapped_TotalCapacity`	`mapped_totalcapacity`	The number of buffers in the mapped buffer pool.

Flink Cluster Metrics

METRIC NAME	UNIT	DESCRIPTION
`numRegisteredTaskManagers`	Count	The number of registered taskmanagers.
`numRunningJobs`	Count	The number of running jobs.

`taskSlotsAvailable`	Count	The number of available task slots.
`taskSlotsTotal`	Count	The total number of task slots.

Flink I/O Metrics

METRIC	UNIT	DESCRIPTION
`currentLowWatermark`	Count	The lowest watermark this task has received.
`numSplitsProcessed`	Count	The total number of InputSplits this data source has processed (if the operator is a data source).
`numBytesInLocal`	Count	The total number of bytes this task has read from a local source.
`numBytesInLocalPerSecond`	Count	The number of bytes this task reads from a local source per second.
`numBytesInRemote`	Count	The total number of bytes this task has read from a remote source.
`numBytesInRemotePerSecond`	Count	The number of bytes this task reads from a remote source per second.
`numBytesOut`	Count	The total number of bytes this task has emitted.
`numBytesOutPerSecond`	Count	The number of bytes this task emits per second.
`numRecordsIn`	Count	The total number of records this operator/task has received.
`numRecordsInPerSecond`	Count	The number of records this operator/task receives per second.
`numRecordsOut`	Count	The total number of records this operator/task has emitted.
`numRecordsOutPerSecond`	Count	The number of records this operator/task sends per second.

Spark Metrics

The following metrics are available for Spark pipelines.

Check the Spark accumulator documentation for more information about these metrics.

Spark Metrics for Pipelines

METRIC	UNIT	DESCRIPTION
Pipeline Status	Count	
Number of Pipeline Executions	Count	
Custom Accumulators	Key:Value	

Spark Metrics for Notebooks

METRIC	DESCRIPTION
Average Memory per Executor	Average memory per executor and Spark driver
Average and Total Spark Memory Usage for All Units	Aggregate of average memory per executor and driver. Also aggregates all memory of the cluster
Active Cores	Number of active cores
Stages	Stages, such as running, pending and failed
Tasks by All Executors	Tasks by executors, active, and pool. This is another way to observe the active and available cores
Message Processing Time	Average message processing time
Completed Tasks by Each Executer	Completed tasks by executors and counters
File System Reads/Writes by Executors	File system read and writes in bytes (when the filesystem is used within jobs only)

Pipeline Metrics

You have access to the following five metrics to understand the status of your pipeline jobs.

For more information on monitoring and troubleshooting pipelines, see *Pipeline Monitoring* in the Pipeline Developer Guide.

METRIC	VALUE	DESCRIPTION
`pipeline_jobs_failed`	`0` or `1`	Value of `1` indicates that the pipeline job has failed.
`pipeline_jobs_canceled`	`0` or `1`	Value of `1` indicates that the pipeline job has been canceled.
`pipeline_jobs_running`	`0` or `1`	Value of `1` indicates that the pipeline job is running.
`pipeline_jobs_completed`	`0` or `1`	Value of `1` indicates that the pipeline job has completed.
`pipeline_jobs_submitted`	`0` or `1`	Value of `1` indicates that the pipeline job has been submitted to the service.

Current Usage Metrics

The following metrics are available for you to monitor your current usage of OLP. The metric in the single-stat panel to the right of the dashboard shows the current value within the last one hour, while the graph panel to the left shows the time series presentation of this value.

Storage Usage Metrics

METRIC	DESCRIPTION
Versioned	Storage of versioned controlled, Geo-Indexed data
Volatile	Storage in volatile memory for more performance with small writes and reads
Stream	Capacity allocated to queue Stream data
Stream TTL	Capacity allocated to store long lived streaming jobs. This covers the EBS volumes associated with the Stream Nodes
MetaData	Storage of meta-data about the data you store
Metrics Time Series	Storage of indexed metrics data for quick numeric time series dashboards. To know how much is allocated to Dashboards, and if more should be invested.

Transfer Usage Metrics

METRIC	DESCRIPTION
Log Search IO	Storage of indexed log data for debugging search
Pipeline IO	Network I/O traffic generated by Pipelines

Compute Usage Metrics

METRIC	DESCRIPTION
Compute Core	CPU Core Hours consumed by Batch or Stream Pipelines
Compute RAM	RAM Hours consumed by Batch or Stream Pipelines

Zeppelin Notebook Metrics

METRIC	DESCRIPTION
HTTP Response Codes	HTTP metrics, response codes, and request/response time
Zeppelin Memory	Zeppelin server JVM memory usage
Zeppelin Threads	Zeppelin server thread states
Zeppelin notebooks status	Number of notebooks running and failing

Create Reports

In Splunk, you can develop your own report by creating a search and saving it as a report. Saved reports are available within the "Saved Reports" tab. For additional details, see the Splunk documentation on saving and sharing reports.

Create and Manage Alerts

You can set up alerts and request email notifications when a condition or threshold is met. Adding a metric can be summarized by the following two sets of steps.

Create a Notification Channel

An alert in Grafana has two components—a notification channel and an alert trigger. A notification channel is defined as a way that you can be notified by Grafana, such as email.

To create a notification channel, follow these steps:

1. Open the Grafana tool from https://platform.here.com/ by clicking on **Tools** > **Monitoring and alerts**.
2. In Grafana, click the dropdown menu in the upper left, select **Alerting** > **Notification Channels**.
3. On this screen, you can create a new channel and specify the notification method by selecting the type from the drop down menu. We only support email and webhook notification types.

Note: Grafana Documentation

For more details and options, see the the Grafana Documentation on Notifications.

Create the Alert

There are two important things to keep in mind before creating an alert:

- **Alerts cannot be created on the standard metrics dashboards that are supplied with OLP.**
- **Alerts can only be setup on a 'Graph' panel in a dashboard.**

To create the alert, follow these steps:

1. Create a new dashboard, use an existing one you've created, or duplicate an existing dashboard. For more information, see the Grafana Getting Started documentation
2. Select a new panel of type **Graph**. Alerts can only be setup on Graph Panels.
3. Edit the graph by clicking on the Panel Title and choosing **Edit**.
4. Add your metric to the Metrics tab. You can find more information on how to do this in the Grafana documentation on Alerting Rules.
5. Select the **Alert** tab to add values to the alert.
6. Select **Notifications** on the left side menu and add your previously created notification channel.

For more information, see the Grafana User Guide.

Duplicate a Dashboard

You can create an identical copy of the dashboard by performing the following:

1. From the Grafana tool, click on the Gear icon at the top.
2. Select 'Save As' and provide a new dashboard name.

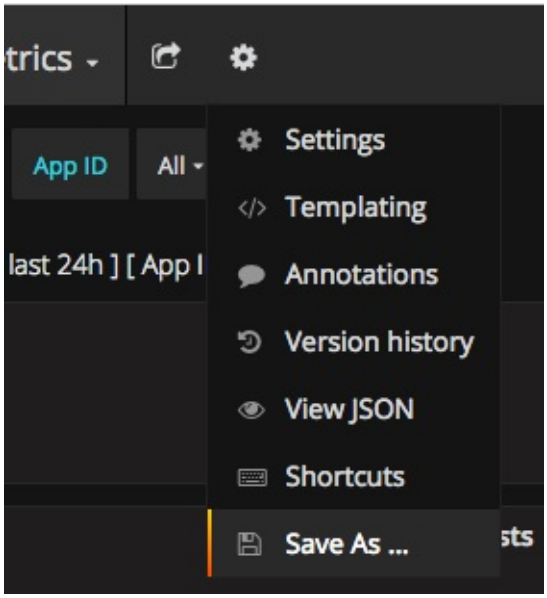

Figure: Grafana menu to duplicate a dashboard

Need Help?

If you need help with this or any other HERE Open Location Platform Services, visit platform.here.com for support.