CLOUD OPERATIONS GUIDE

<REDACTED> 5.0.1

December 2020

Table of contents

Document revision history 5

About this publication 6

Intended audience 6

Related publications 6

Terminology 6

1 Getting started 7

2 Prerequisites 7

3 Deployments 7

4 System startup, scaling, and shutdown 9

5 Troubleshooting 9

5.1 Log messages 9

5.1.1 Access log messages 9

5.1.2 Common log query fields 11

5.1.3 Index Lifecycle Management 13

5.1.4 Discover Data 14

5.1.5 Log Dashboards 15

6 Platform monitoring (metrics) 16

6.1 Metric descriptions 16

7 REST management API 20

8 Transaction traces 21

9 Application tools 23

9.1 <REDACTED> toolkit 23

10 Alerts 24

11 Data Offloads 25

12 Rejected Messages Log 25

13 Distributed Traces **25**

13.1 Review distributed traces 26

13.2 Identify services 27

13.3 Adjust the sampling rate 28

14 Security **28**

14.1 Overview 28

14.2 <REDACTED> Role-Based Security 28

14.3 <REDACTED> Cloud Ingress and Egress 28

14.4 Securing the Cluster Communication using Istio Authentication 29

About this publication

This document describes the operations topics for the Cloud-native variant of the <REDACTED> platform.

Intended audience

The audience for this document is administrative and operational staff responsible for provisioning, configuring, managing, and operating the <REDACTED> when it is hosted in a Kubernetes cloud.

This guide assumes knowledge in the following areas:

- The <REDACTED>
- Software operations and maintenance
- Host operating system (Windows, or Linux)
- Host container orchestration (Kubernetes)

Related publications

For more information, see the following publications:

- <REDACTED>

Terminology

Terms used in the publication:

- <REDACTED>

1 Getting started

Many aspects of the *Operations Guide* are not applicable because the operation of the cloud offering of the <REDACTED> platform is substantially different from the traditional deployment model. This document offers specific guidance for the unique aspects of a cloud-native deployment on Kubernetes.

After using the *Getting Started with* <REDACTED> *Cloud* document to successfully complete the installation, consult this document to understand how to monitor, adjust, and operate an installed environment.

The intent is to not repeat information and direction supplied by other widely available guides on the third-party technologies used in a <REDACTED> cloud deployment. <REDACTED> intent is to give context to how typical operations are applied in the <REDACTED> solution.

2 Prerequisites

- A high degree of familiarity with Kubernetes cloud orchestration and administration of Kubernetes installations.

- General familiarity with Istio networking as it is applied to Kubernetes-based deployments.
- General familiarity with Prometheus and its associated query languages (if working with custom Platform Monitoring dashboards).
- General familiarity with Grafana (if working with custom Platform Monitoring dashboards).
- General familiarity with Kibana and the Kibana Query language.
- Familiarity with Elastic Index Management and Index Lifecycle Management.
- Familiarity with the Elastic Common Schema.
- Familiarity with <REDACTED> Entitlements (if your integration solution requires integration with Entitlements APIs).

3 Deployments

The Kubernetes deployments to be managed in the <REDACTED> cloud are the following:

<REDACTED table>

The namespace for these deployments is established at installation. See the *Getting Started with <REDACTED> Cloud* for additional details.

In addition, there are deployments that are customary as part of third-party products. These are shown as multiple deployments in the following namespaces:

- istio-system
- <namespace>-jaeger
- <namespace>-operator
- fluent-bit
- <namespace>-elasticsearch
- <namespace>-kibana

4 System startup, scaling, and shutdown

Generally, after completing the steps in the *Getting Started with UPF Cloud Guide* (specifically the helm install of the <REDACTED> chart), the system is considered to be started. After this initial startup, Kubernetes assumes responsibility for starting instances of the <REDACTED> platform deployments.

After the initial install, there will be one replica of each deployment (each pod type). Increase the replica count of all <REDACTED> deployments for production environments to at least two. Also, increase the replica count of other critical components like the Istio ingress gateway.

When a restart of an individual deployment is required, you can scale an individual deployment to 0 replicas. Then, after all pods terminate, the deployment can be scaled to a higher value to create a new set of pods.

Use `kubectl drain <node name>` to stop running any pods on a given node. After the command completes successfully, no pods will execute on the node and you can conduct maintenance (including restarts). After maintenance activities have been completed, use `kubectl uncordon <node name>` to start scheduling pods on the node again.

5 Troubleshooting

5.1 Log messages

When you select the centralized logging installation options according to the *Getting Started with <REDACTED> Cloud Guide*, <REDACTED> leverages the Elastic ELK Observability stack to provide observability in cloud deployments. **Note:** Typically, the traditional logging and tracing facilities as described in the *Logging and Tracing Guide* or the *Operation Guide* are not applicable.

Instead, log messages (server log, Safeguard, diagnostics, and transaction trace) from the <REDACTED> and <REDACTED> framework tiers are collected in Elasticsearch and viewed through Kibana.

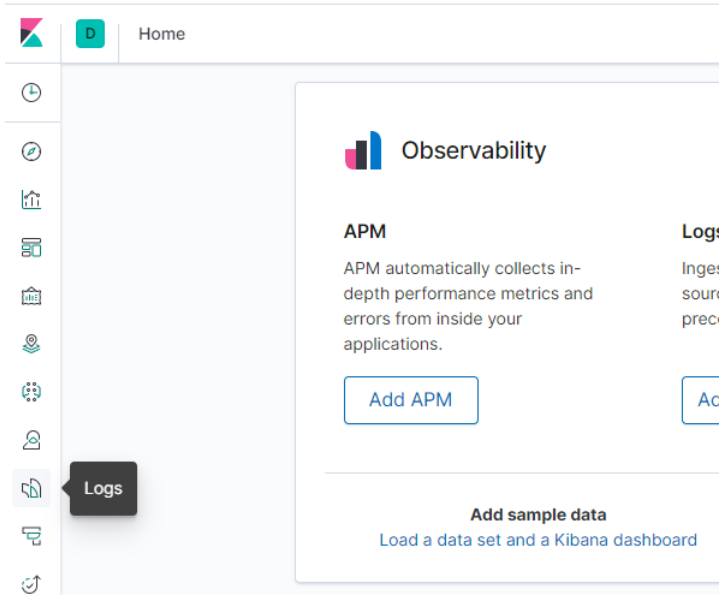
A secondary stream of log messages is stored as flat files for assistance in HELP24 support cases.

5.1.1 Access log messages

1. In the <REDACTED>, go to **Operations >** <REDACTED **> Log Messages** to launch Kibana, which has options related to logging in the **Visualize and Explore Data** sections like **Dashboard**, **Discover**, and **Logs**.

   The log message content from the <REDACTED> platform is formatted in the Elastic Common Schema. Use the *Elastic Common Schema Field Reference* to understand the content of each field in the log. Only the most relevant fields for which log data is available will be populated by the <REDACTED> platform.

2. Click the Logs  tab to get familiar with the logs through the Kibana logs feature.



5.1.2 Common log query fields

This section describes important fields in the log data schema. They will be often used to select and visualize log data.

5.1.2.1 event.dataset

This is the highest level of classification of any logged event. The datasets fall in line with how log lines were separated into human-readable files in a log directory. Currently, the datasets are defined as follows:

<REDACTED table>

5.1.2.2 log.logger

Identifying name of the object that was responsible for announcing the event. For example, in the <REDACTED> tier, it's typically the class name given to the **ADFLogger** instance. In <REDACTED>, the name given to the instance of **Diag**.

This field is present only if a logging framework object like **ADFLogger** or **Diag** was invoked to log the event. For example, it will not be present for events where the instrumenting code wrote a message directly to stdout.

5.1.2.3 log.level

The nature of the event such as general information, notification of an error, or debug output. The possible values are particular to the logging application but might have some commonality. For example, FATAL, DEBUG, INFO, and WARN.

5.1.2.4 message

The portion of the event that conveys human-readable content to assist you in interpreting the event. This is simply an expression of "what happened." Elasticsearch wildcard searches are useful in this field. For example, message: "<REDACTED> *Ready*".

5.1.2.5 kubernetes.container_name

The name of the container based on the Docker image name. Use this field when qualifying a search that applied across pods, but in a type of pod. Example values are <REDACTED>, switch, responsive, and entitlements.

5.1.2.6 kubernetes.pod_name

The name of the pod from which the event was announced. Use this field when isolating activity to a specific node or pod.

<REDACTED>

5.1.3 Index Lifecycle Management

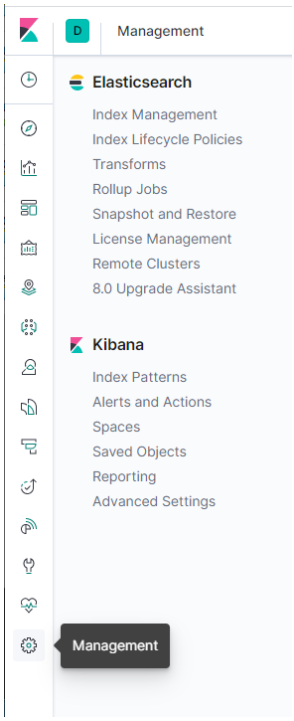
Log data from the <REDACTED> platform is kept in Elasticsearch in indexes conforming to the logstash-* pattern. In the centralized logging installation, a default Index Lifecycle Management policy is applied to indexes on this pattern to ensure that the data is well organized and the performance of the Elasticsearch cluster is maintained. By default, log data is:

- Moved from hot to warm status after 25 hours

- Moved from warm to cold/frozen status after 5 days
- Deleted after 120 days

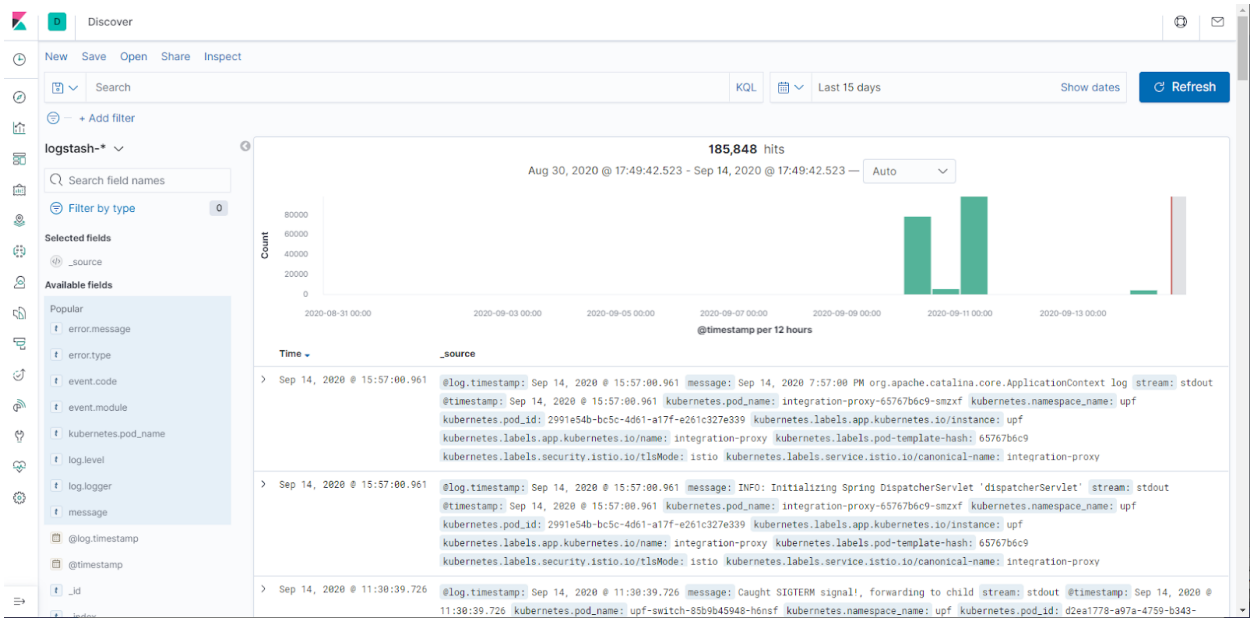
By default, snapshots backups are scheduled nightly with the default installation. The retention period is configured with the `snapshot_retention_period` ansible variable at installation.

After installation, review, adjust, and test these defaults according to the requirements and resources available in the deployment. Use the Kibana **Management** tab to perform this work.



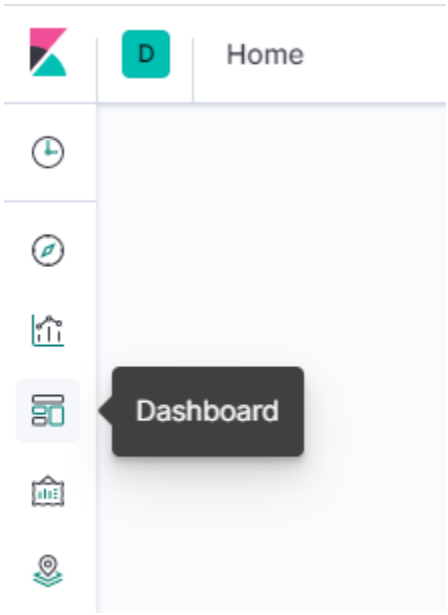
5.1.4 Discover Data

Use the **Discover** tab in Kibana to perform ad hoc queries and export CSV data. When using this view, you can use the logstash-* index pattern to access <REDACTED> platform log data.



5.1.5 Log Dashboards

Logging dashboards in Kibana provide visualizations of log data. Custom dashboards can also be introduced.



The set of default dashboards include:

<REDACTED>

6 Platform monitoring (metrics)

The Platform Monitoring feature of the <REDACTED> presents metrics dashboards through the Prometheus time-series database and Grafana visualization software. The cloud implementation of the <REDACTED> platform exposes several raw metrics involving the internal operation of the platform. The intent is to convey how well the system is operating internally. The areas of internal functionality are very broad. The following are examples of these metrics:

- Authentication success rate
- Cache hit rates
- Overall transactions per second
- Average time to execute a script

<REDACTED>

7 REST management API

The REST management API is the principal mechanism by which management commands can be issued to the platform. Wherein traditional deployments the commands to tune the platform, manage diagnostics, start traces, and set options were issued through the <REDACTED> or command-line utilities such as <REDACTED>, these operational management commands must be invoked through a REST API.

Typically, you must use a third-party utility or product (like curl or Postman) or software library to compose the appropriate JSON for the command and send it to the REST API address and port (as configured during installation) through HTTPS.

The HTTPS URL follows the format: **<host:port>/management/<target>/<command_name>**

<REDACTED>

8 Alerts

Alerts are error conditions that are escalated to operators for their attention. Alerts are recorded with details such as type, severity, tenants, autoHandled, and time when the alert occurred, and when the alert was cleared.

You can use the <REDACTED> framework to generate an alert based on predefined events. It uses a subscription mechanism that makes use of entitlements to determine the user eligibility for an alert. The Alert framework also provides lifecycle support for alerts so users can see alert states change from active when raised to non-active when cleared. Alert definitions control whether to present alerts with a Clear Alert button so users can clear an alert when it is no longer applicable.

<REDACTED>