Pivotal

PRODUCT DOCUMENTATION

PCF Metrics®

Version 1.4

User's Guide

Table of Contents

PCF Metrics

Pivotal Cloud Foundry (PCF) Metrics stores logs, metrics data, and event data from apps running on PCF for the past two weeks. It graphically presents this data to help operators and developers better understand the health and performance of their apps.

What's New in PCF Metrics v1.4

PCF Metrics v1.4 supports out-of-the-box Spring Boot Actuator metrics, custom app metrics, and instance-level metrics visualization. It also adds significant changes to the UI.

For more information...	See...
Spring Boot Actuator metrics	Custom Metrics
Custom app metrics	Custom Metrics
Instance-level metrics visualization	View Metrics at App-Instance Level
Installation steps	Installing PCF Metrics

Product Snapshot

The following table provides version and version-support information about PCF Metrics.

Element	Details
Tile version	v1.4.7
Release date	September 12, 2018
Compatible Ops Manager version(s)	v1.12 and later
Compatible Elastic Runtime version(s)	v1.12 and later
Compatible Pivotal Application Service (PAS)* version(s)	v2.0 and later
IaaS support	AWS, Azure, GCP, OpenStack, and vSphere
IPsec support	Yes

* As of PCF v2.0, *Elastic Runtime* is renamed *Pivotal Application Service (PAS)*.

PCF Metrics User Guide

See the following topics for details about PCF Metrics:

- Installing PCF Metrics
- Sizing PCF Metrics For Your System
- Monitoring and Troubleshooting Apps with PCF Metrics
- Troubleshooting PCF Metrics
- PCF Metrics Product Architecture
- Release Notes and Known Issues
- Monitoring PCF Metrics

Installing PCF Metrics

This document describes how to install and configure Pivotal Cloud Foundry (PCF) Metrics.

To use the custom metrics feature of the PCF Metrics tile, you must have the Metrics Forwarder tile. However, the PCF Metrics tile can be installed without the Metrics Forwarder tile.

For information about the components deployed as part of this install procedure, see PCF Metrics Product Architecture.

Add the PCF Metrics Tile to Ops Manager

1. Download the PCF Metrics file from Pivotal Network.
2. Upload the PCF Metrics file to your Ops Manager installation.
3. Click **Add** next to the uploaded product description in the **Available Products** view to add PCF Metrics to your **Installation Dashboard**.

Configure the PCF Metrics Tile

Note: The following procedures offer a standard configuration. To customize PCF Metrics for high capacity, see the Sizing PCF Metrics for Your System.

1. From the **Installation Dashboard**, click the **PCF Metrics** tile.
2. Follow the procedures below to configure the PCF Metrics tile:
    - Assign Availability Zones (AZs) and Networks
    - Configure Metrics Components
    - Configure Errands
    - Configure Resources
    - Import Stemcell

Assign Availability Zones (AZs) and Networks

1. Click **Assign AZs and Networks**.
2. Select an Availability Zone under **Place singleton jobs**.
   Ops Manager runs Metrics jobs with a single instance in this Availability Zone.
3. Select one or more Availability Zones under **Balance other jobs**.
   Ops Manager balances instances of Metrics jobs with more than one instance across the Availability Zones that you specify. However this setting does not balance Elastic Search jobs across multiple AZs as merely balancing ES jobs across multiple AZs does not ensure Highly available ES because ES auto-shards and distributes replicas across ES data nodes in real time. So instead, we have added a redis instance between the firehose and ES (https://docs.pivotal.io/pcf-metrics/1-4/architecture.html) which buffers logs before they are written to ES. In case of unavailable ES (during deployment, ES red indices or AZ failure), redis will hold logs in buffer and write it back once ES is stable again. You can create multiple redis jobs and balance them across AZs for ensuring HA redis in case of AZ failure.
4. Use the drop-down menu to select a network.
   PCF Metrics should be installed on the same network as the Elastic Runtime tile uses. However, if you do select a different subnet network, then refer to the table below when you configure the subnet network in the Elastic Runtime tile to make sure that you have the ports needed for each of the PCF Metrics data services.

Service	Port
Elasticsearch	9200
MySQL	3306
Redis	6379

5. Click **Save**.

Configure Metrics Components

1. Click **Metrics Components Config**.
2. Review the **MySQL Logqueue Count** value. You can increase this instance count at any time to accommodate higher levels of inbound metrics traffic.
3. Review the **Elasticsearch Logqueue Count** value. You can increase this instance count at any time to accommodate higher levels of inbound log traffic.
4. Review the **Ingestor Count** value. You can increase this instance count at any time to accommodate higher levels of Loggregator Firehose traffic.
5. Review the **Metrics API server instance count** value. You can increase this instance count at any time to accommodate deployments with more apps, metrics, and logs.
6. Click **Save**.

Configure Errands

To properly configure the **Errands** pane, follow the procedure that corresponds to your Ops Manager version.

- Configure Errands for Ops Manager v1.9
- Configure Errands for Ops Manager v1.10

Configure Errands for Ops Manager v1.9

1. Click **Errands**.

   Note: The PCF Metrics tile selects all **Post-Deploy Errands** by default. Pivotal recommends that you do not deselect any errands as doing so can cause issues with the deployment of the tile. However, you can deselect the **Remove PCF Metrics 1.3 CF Resources Errand** and **Migrate Old Data to 1.4 Errand** after deploying v1.4 of the tile.

2. Review the **Post-Deploy Errands** and **Pre-Delete Errands**: If you are deploying the tile for the first time, select all **Post-Deploy Errands**. The following list describes what the **Smoke Tests** errand does. See Smoke Test Errors in *Troubleshooting PCF Metrics* for information on resolving errors discovered by this errand:

   - Confirms that MySQL ingests metrics
   - Confirms that Elasticsearch ingests logs
   - Confirms that the Metrics API returns metrics and logs
   - Confirms that any custom metrics coming out of the Firehose are ingested and stored

   Note: If you deselect **Remove PCF Metrics 1.4 CF Resources Errand** under **Pre-Delete Errands**, artifacts might remain after the PCF Metrics tile uninstalls.

Configure Errands for Ops Manager v1.10

1. Click **Errands**.
2. Review the **Post-Deploy Errands** and **Pre-Delete Errands**:

   a. If you are deploying the tile for the first time, set all **Post-Deploy Errands** to **On**.
   b. For deployments after the initial install, configure the **Post-Deploy Errands** as follows:

      i. Set **Remove PCF Metrics 1.3 CF Resources Errand** to **When Changed**
      ii. Set **Push PCF Metrics components Errand** to **On**
      iii. Set **Migrate Old Data to 1.4 Errand** to **When Changed**
      iv. Set **Smoke Tests Errand** to **On**
      The following list describes what the **Smoke tests** errand does. See Smoke Test Errors in *Troubleshooting PCF Metrics* for information on resolving errors discovered by this errand:

      - Confirms that MySQL ingests metrics

- Confirms that Elasticsearch ingests logs
- Confirms that the Metrics API returns metrics and logs
- Confirms that any custom metrics coming out of the Firehose are ingested and stored

Note: If you set **Remove PCF Metrics 1.4 CF Resources** under **Pre-Delete Errands** to **Off**, artifacts might remain after the PCF Metrics tile uninstalls.

Configure Resources

1. Click **Resource Config**.
2. Review the resource configurations. By default, the settings match the instance types that are best suited for each job. For reference, the following table shows the default resource and IP requirements for installing the PCF Metrics tile:

Resource	Instances	Persistent	CPU	RAM	Ephemeral	Static IP	Dynamic IP
Elasticsearch Master	1	10 GB	2	8 GB	16 GB	1	0
Elasticsearch Data	4	100 GB	2	16 GB	32 GB	4	0
Redis	1	10 GB	4	4 GB	8 GB	1	0
MySQL Server	1 (Not configurable)	500 GB	4	16 GB	32 GB	1	0

If you expect a high level of use, you might need to increase the disk resources available to your instances. For more information, see Sizing PCF Metrics for Your System.

Note: For Ops Manager v1.9, if you configure a job with persistent disk larger than 2 TB, you might experience disk partitioning issues with the BOSH Director.

3. Click **Save**.

Import Stemcell

For Ops Manager v2.0 and earlier:

1. Navigate to Pivotal Network and click Stemcells for PCF.
2. Download the appropriate stemcell version for your IaaS.

Note: On AWS, make sure to use a Hardware Virtual Machine (HVM) stemcell if you are using the default instance sizes.

3. Navigate to the **Stemcell** pane found under the **Settings** tab of your tile.
4. Click **Import Stemcell** and select the stemcell file you downloaded.

For Ops Manager v2.1 and later:

1. Follow the procedure outlined in Importing and Managing Stemcells.

Deploy PCF Metrics

1. If you are using Ops Manager v2.3 or later, click **Review Pending Changes**. For more information about this Ops Manager page, see Reviewing Pending Product Changes.
2. Click **Apply Changes** to install the service. If the smoke tests fail, see Smoke Test Errors in *Troubleshooting PCF Metrics*.

For more information on how to log in, use, and interpret data from PCF Metrics, see Monitoring and Troubleshooting Apps with PCF Metrics.

Sizing PCF Metrics for Your System

This topic describes how operators configure Pivotal Cloud Foundry (PCF) Metrics depending on their deployment size. Operators can use these procedures to optimize PCF Metrics for high capacity or to reduce resource usage for smaller deployment sizes.

After your deployment has been running for a while, use the information in this topic to scale your running deployment.

If you are not familiar with the PCF Metrics components, review PCF Metrics Product Architecture before reading this topic.

For how to configure resources for a running deployment, see the procedures below:

- Procedure for Scaling the Metrics Datastore
- Procedure for Scaling the Log Datastore
- Procedure for Scaling the Temporary Datastore
- Procedure for Scaling the Ingestor, Logqueues, and Metrics API

Suggested Sizing by Deployment Size

Use the following tables as a guide for configuring resources for your deployment.

Estimate the size of your deployment according to how many apps are expected to be deployed.

Size	Purpose	Approximate number of apps
Small	Test use	100
Medium	Production use	5,000
Large	Production use	15,000

If you are using Metrics Forwarder and custom metrics, you might need to scale up the MySQL Server instance more than indicated in the tables below. Pivotal recommends you start with the one of the following configurations and scale up as necessary by following the steps in Configuring the Metrics Datastore.

Deployment Resources for a Small Deployment

This table lists the resources you need to configure for a small deployment, about 100 apps.

Job	Instances	Persistent Disk Type	VM Type
Elasticsearch Master	1	10 GB	small (cpu: 1, ram: 4 GB, disk: 8 GB)
Elasticsearch Data	2 or more	32 GB	small (cpu: 1, ram: 4 GB, disk: 8 GB)
Redis	1	32 GB	micro (cpu: 1, ram: 4 GB, disk: 8 GB)
MySQL Server	1 (not configurable)	32 GB	small (cpu: 1, ram: 4 GB, disk: 8 GB)

Deployment Resources for a Medium Deployment

This table lists the resources you need to configure for a medium deployment, about 5000 apps.

Job	Instances	Persistent Disk Type	VM Type
Elasticsearch Master	1	10 GB	small (cpu: 1, ram: 2 GB, disk: 8 GB)
Elasticsearch Data	5	200 GB	small.disk (cpu: 1, ram: 2 GB, disk: 16 GB)
Redis	1	10 GB	small.disk (cpu: 1, ram: 2 GB, disk: 16 GB)
MySQL Server	1 (not configurable)	500 GB	medium (cpu: 2, ram: 4 GB, disk: 8 GB)

Deployment Resources for a Large Deployment

This table lists the resources you need to configure for a large deployment, about 15,000 apps.

Job	Instances	Persistent Disk Type	VM Type
Elasticsearch Master	1	10 GB	small (cpu: 1, ram: 2 GB, disk: 8 GB)
Elasticsearch Data	10	500 GB	large (cpu: 2, ram: 8 GB, disk: 16 GB)
Redis	1	10 GB	large (cpu: 2, ram: 8 GB, disk: 16 GB)
MySQL Server	1 (not configurable)	2 TB	large (cpu: 2, ram: 8 GB, disk: 16 GB)

Scale the Metrics Datastore

PCF Metrics stores metrics in a single MySQL node. For PCF deployments with high app logs load, you can add memory and persistent disk to the MySQL server node.

Considerations for Scaling the Metrics Datastore

While the default configurations in Suggested Sizing by Deployment Size above are a good starting point for your MySQL server node, they do not take into account the additional load from custom metrics. Pivotal recommends evaluating performance over a period of time and scaling upwards as necessary. As long as persistent disk is scaled up, you won't not lose any data from scaling.

Procedure for Scaling

Do the following to scale up the MySQL server node:

To scale up the MySQL server node, do the following:

1. Determine how much memory and persistent disk are required for the MySQL server node.
2. Navigate to the Ops Manager Installation Dashboard and click the **Metrics** tile.
3. From the **Settings** tab of the **Metrics** tile, click **Resource Config**.
4. Enter the values for the **Persistent Disk Type** and **VM Type**.
5. Click **Save**.

warning! If you are using PCF v1.9.x and earlier, there might be issues Ops Manager BOSH Director using persistent disks larger than 2 TB.

Scale the Log Datastore

PCF Metrics uses Elasticsearch to store logs. Each Elasticsearch node contains multiple shards of log data, divided by time slice.

Considerations for Scaling

Pivotal suggests starting with the default configurations for your Elasticsearch Master and Data nodes, observing the disk usage of the Data nodes, and then scaling the resources accordingly.

The following calculation attempts to measure Elasticsearch resource requirements more precisely depending on your logs load. This formula is only an approximation, and Pivotal suggests rounding the numbers up as a safety measure against undersizing Elasticsearch:

1. Determine how many logs the apps in your deployment emit per hour (*R*) and the average size of each log (*S*).
2. Calculate the number of instances (*N*) and the persistent disk size for the instances (*D*) you need to scale to using the following formula:

$$R \times S \times 336 \times 2 = N \times D$$

The formula assumes that a log retention period is 336 hours (2 weeks), and the number of Elasticsearch replica shards is 1 (default).

For example:
200,000 logs/hr × 25 KB × 336 hr × 2 ≈ 7 instances × 500 GB
or
200,000 logs/hr × 25 KB × 336 hr × 2 ≈ 3 instances × 1 TB

As stated above, the default Elasticsearch configuration sets the number of replicas to 1, which means that every shard has 1 replica and Elasticsearch logs are HA by default.

Procedure for Scaling

warning! If you modify the number of Elasticsearch instances, the Elasticsearch cluster temporarily enters an unhealthy period during which it does not ingest any new logs data, due to shard allocation.

After determining the number of Elasticsearch nodes needed for your deployment, perform the following steps to scale your nodes:

1. Navigate to the Ops Manager Installation Dashboard and click the **Metrics** tile.
2. From the **Settings** tab of the **Metrics** tile, click **Resource Config**.
3. Locate the **Elasticsearch Data** job and select the dropdown menu under **Instances** to change the number of instances.
4. Click **Save**.

Scale the Temporary Datastore (Redis)

PCF Metrics uses Redis to temporarily store ingested data from the Loggregator Firehose as well as cache data queried by the Metrics API. The former use case is to prevent major metrics and logs loss when the data stores (Elasticsearch and MySQL) are unavailable. The latter is to potentially speed up front-end queries. See PCF Metrics Product Architecture for more information.

Considerations for Scaling

The default Redis configuration specified in Suggested Sizing by Deployment Size above that fits your deployment size should work for most cases. Redis stores all data in memory, so if your deployment size requires it, you can also consider scaling up the RAM for your Redis instance(s). You can additionally increase the number of Redis instances to 2 if you need HA behavior when Redis upgrades.

Procedure for Scaling

Follow these steps to configure the size of the Redis VM for the temporary datastore based on your calculations.

Note: In the case that the temporary datastore becomes full, Redis uses the `volatile-ttl` eviction policy to continue storing incoming logs. For more information, see *Eviction policies* in Using Redis as an LRU cache.

1. Navigate to the Ops Manager Installation Dashboard and click the **Metrics** tile.
2. From the **Settings** tab, click **Resource Config**.
3. Locate the **Redis** job and select the dropdown menu under **Instances** to scale Redis up or down.
4. Click **Save**.

Scale the Ingestor, Logqueues, and Metrics API

The procedures for scaling the Ingestor, Elasticsearch logqueue, MySQL logqueue, and Metrics API instances are similar.

- **Ingestor** — PCF Metrics deploys the Ingestor as an app, `metrics-ingestor`, within PCF. The Ingestor consumes logs and metrics from the Loggregator Firehose, sending metrics and logs to their respective Logqueue apps.
  To customize PCF Metrics for high capacity, you can scale the number of Ingestor app instances and increase the amount of memory per instance.
- **Logqueues** — PCF Metrics deploys a **MySQL Logqueue** and an **Elasticsearch Logqueue** as apps, `mysql-logqueue` and `elasticsearch-logqueue`, within PCF. The MySQL logqueue consumes metrics from the Ingestor and forwards them to MySQL. The Elasticsearch logqueue consumes logs from the Ingestor and forwards them to Elasticsearch.
  To customize PCF Metrics for high capacity, you can scale the number of Logqueue app instances and increase the amount of memory per instance. The number of MySQL and Elasticsearch logqueues needed is dependent on the frequency that logs and metrics are forwarded by the Ingestor. As a general rule:
  - For every 45,000 logs per minute, add 2 Elasticsearch logqueues.
  - For every 17,000 metrics per minute, add 1 MySQL logqueue.

  The above is a general estimate. You might need fewer instances depending on your deployment. To optimize resource allocation, provision fewer instances initially and increase instances until you achieve desired performance.
- **Metrics API** — PCF Metrics deploys the app, `metrics`, within PCF.

Refer to this table to determine how many instances you need for each component.

Item	Small	Medium	Large
Ingestor instance count	Number of Doppler servers	Number of Doppler servers	Number of Doppler servers
MySQL logqueue instance count	1	1	2
ES logqueue instance count	1	2	3
Metrics API instance count	1	2	2

Find the number of Doppler servers in the Resource Config pane of the Pivotal Elastic Runtime tile.

Considerations for Scaling

Pivotal recommends starting with the configuration in Suggested Sizing by Deployment Size above, for your deployment size and then evaluating performance over a period of time and scaling upwards if performance degrades.

Procedure for Scaling

warning! If you decrease the number of instances, you might lose data currently being processed on the instances you eliminate.

After determining the number of instances needed for your deployment, perform the following steps to scale:

1. Target your Cloud Controller with the Cloud Foundry Command Line Interface (cf CLI). If you have not installed the cf CLI, see Installing the cf CLI.

   ```
   $ cf api api.YOUR-SYSTEM-DOMAIN
   Setting api endpoint to api.YOUR-SYSTEM-DOMAIN...
   OK
   API endpoint:   https://api.YOUR-SYSTEM-DOMAIN (API version: 2.54.0)
   Not logged in. Use 'cf login' to log in.
   ```

2. Log in with your UAA administrator credentials. To retrieve these credentials, navigate to the **Pivotal Elastic Runtime** tile in the Ops Manager Installation Dashboard and click **Credentials**. Under **UAA**, click **Link to Credential** next to **Admin Credentials** and record the password.

   ```
   $ cf login
   API endpoint: https://api.YOUR-SYSTEM-DOMAIN


   Email> admin
   Password>
   Authenticating...
   OK
   ```

3. When prompted, target the `metrics` space.

```
Targeted org system


Select a space (or press enter to skip):
1. system
2. notifications-with-ui
3. autoscaling
4. metrics


Space> 4
Targeted space metrics


API endpoint:   https://api.YOUR-SYSTEM-DOMAIN (API version: 2.54.0)
User:           admin
Org:            system
Space:          metrics
```

4. List the apps that are running in the `metrics` space.

```
$ cf apps
Getting apps in org system / space metrics as admin...
OK

name                    requested state   instances   memory   disk   urls
elasticsearch-logqueue  started           1/1         256M     1G
metrics                 started           1/1         1G       2G     metrics.YOUR-SYSTEM-DOMAIN/api/v1
metrics-ingestor        started           1/1         256M     1G
metrics-ui              started           1/1         64G      1G     metrics.YOUR-SYSTEM-DOMAIN
mysql-logqueue          started           1/1         512M     1G
```

5. Scale the app to the desired number of instances:
   `cf scale APP-NAME -i INSTANCE-NUMBER`
   Where the `APP-NAME` is `elasticsearch-logqueue`, `metrics`, `metrics-ingestor`, or `mysql-logqueue`.
   For example, to scale all the apps:

```
$ cf scale elasticsearch-logqueue -i 2
$ cf scale metrics -i 2
$ cf scale metrics-ingestor -i 2
$ cf scale mysql-logqueue -i 2
```

6. Evaluate the CPU and memory load on the instances:
   `cf app APP-NAME`
   For example,

```
$ cf app metrics-ingestor
Showing health and status for app metrics-ingestor in org system / space metrics as admin...
OK


requested state: started
instances: 1/1
usage: 1G x 1 instances
urls:
last uploaded: Sat Apr 23 16:11:29 UTC 2016
stack: cflinuxfs2
buildpack: binary_buildpack



     state     since                  cpu    memory       disk         details
#0   running   2016-07-21 03:49:58 PM   2.9%   13.5M of 1G   12.9M of 1G
```

7. If your average memory usage exceeds 50% or your CPU consistently averages over 85%, add more instances with `cf scale APP-NAME -i INSTANCE-NUMBER`.

   In general, you should scale the app by adding additional instances. However, you can also scale the app by increasing the amount of memory per instance:

```
cf scale APP-NAME -m NEW-MEMORY-LIMIT
```

   For example,

```
$ cf scale metrics-ingestor -m 2G
```

For more information about scaling app instances, see Scaling an Application Using cf scale.

Monitoring and Troubleshooting Apps with PCF Metrics

This topic describes how developers can monitor and troubleshoot their apps using Pivotal Cloud Foundry (PCF) Metrics.

Overview

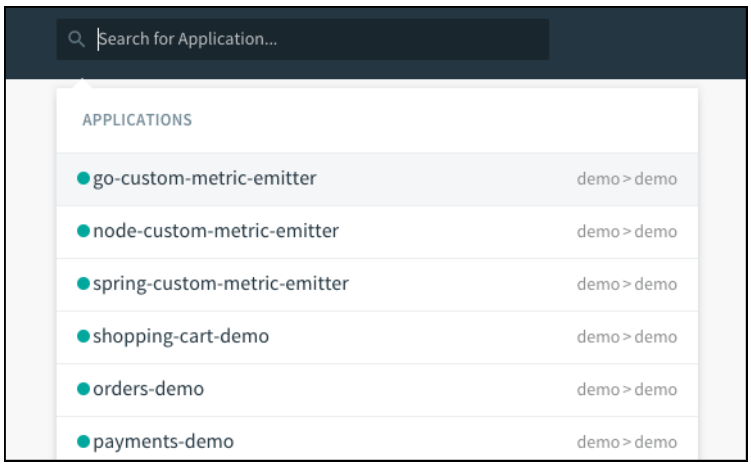
PCF Metrics helps you understand and troubleshoot the health and performance of your apps by displaying the following:

- Container Metrics: Three graphs measuring CPU, memory, and disk usage percentages
- Network Metrics: Three graphs measuring requests, HTTP errors, and response times
- Custom Metrics: User-customizable graphs for measuring app performance, such as Spring Boot Actuator metrics
- App Events: A graph of update, start, stop, crash, SSH, and staging failure events
- Logs: A list of app logs that you can search, filter, and download
- Trace Explorer: A dependency graph that traces a request as it flows through your apps and their endpoints, along with the corresponding logs

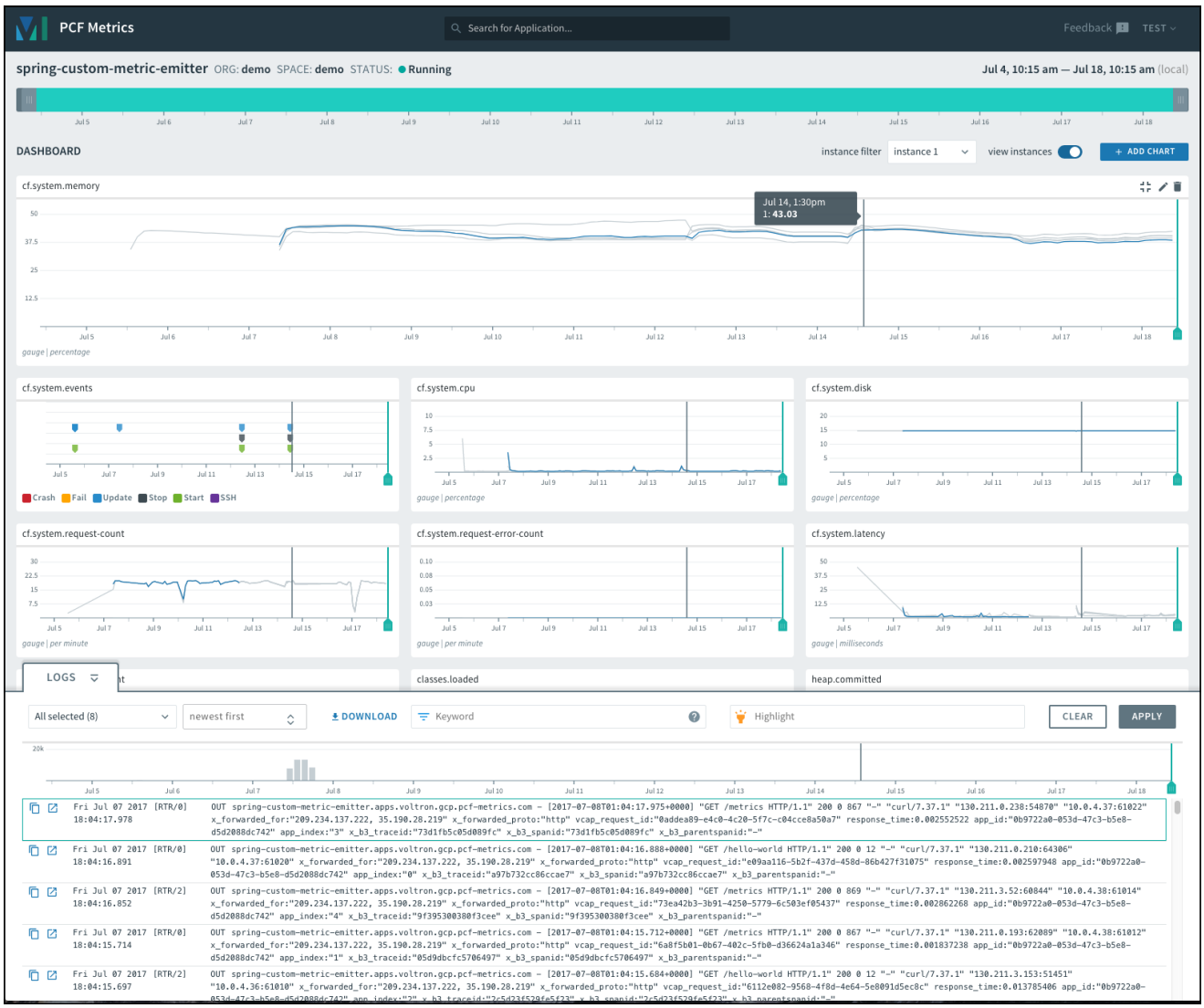
The following sections describe a standard workflow for using PCF Metrics to monitor or troubleshoot your apps.

View an App

In a browser, navigate to `metrics.YOUR-SYSTEM-DOMAIN` and log in with your User Account and Authentication (UAA) credentials. Choose an app for which you want to view metrics or logs. PCF Metrics respects UAA permissions such that you can view any app that runs in a space that you have access to.



PCF Metrics displays app data for a given time frame. See the sections below to Change the Time Frame for the dashboard, Interpret Metrics information on each graph, and Trace App Requests with the Trace Explorer.



Change the Time Frame

The graphs show time along the horizontal axis. You can change the time frame for all graphs and the logs by using the time selector at the top of the window. Adjust either end of the selector or click and drag.

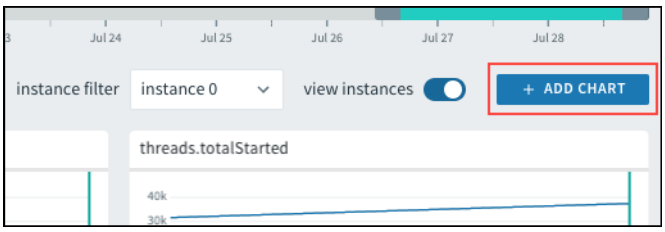
Zoom: From within any graph, click and drag to zoom in on areas of interest. This adjusts all of the graphs, and the logs, to show data from that time frame.

Add, Edit, and Delete Charts

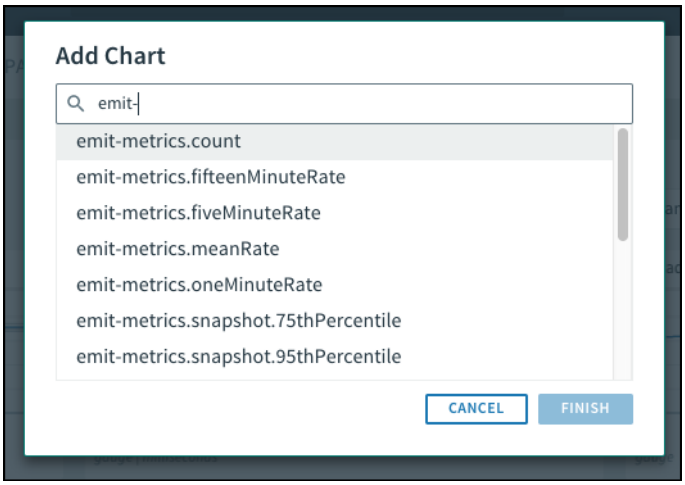
The PCF Metrics dashboard allows users to add, edit, and delete charts.

Add Chart: To add a new chart, follow the steps below.

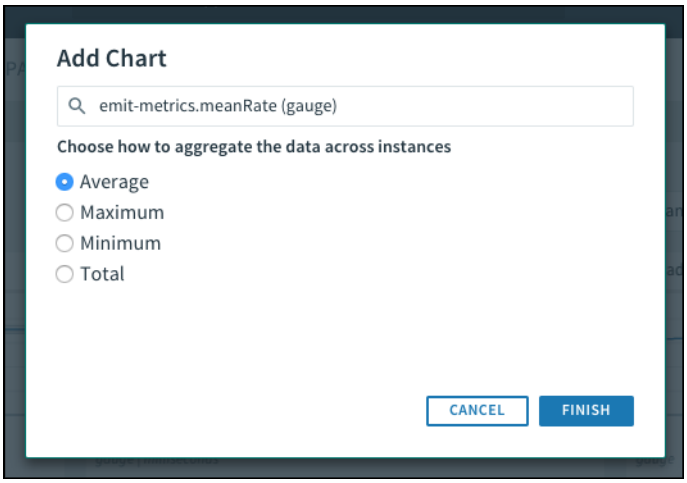
1. Click **+ ADD CHART** at the top right of the dashboard.



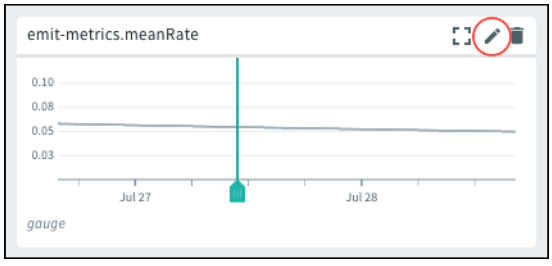
2. In the modal window, either select a metric from the dropdown menu or type the name of the metric into the search bar to filter results.



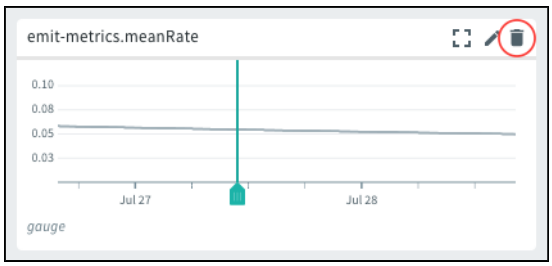
3. Select an aggregation type. This determines how to combine the data from multiple instances.



Edit Chart: To change how instances are aggregated for an existing metric, click the pencil icon on the header of the metric chart. When the **Edit Chart** modal window appears, you can choose the aggregation type and click **Save** to apply changes.

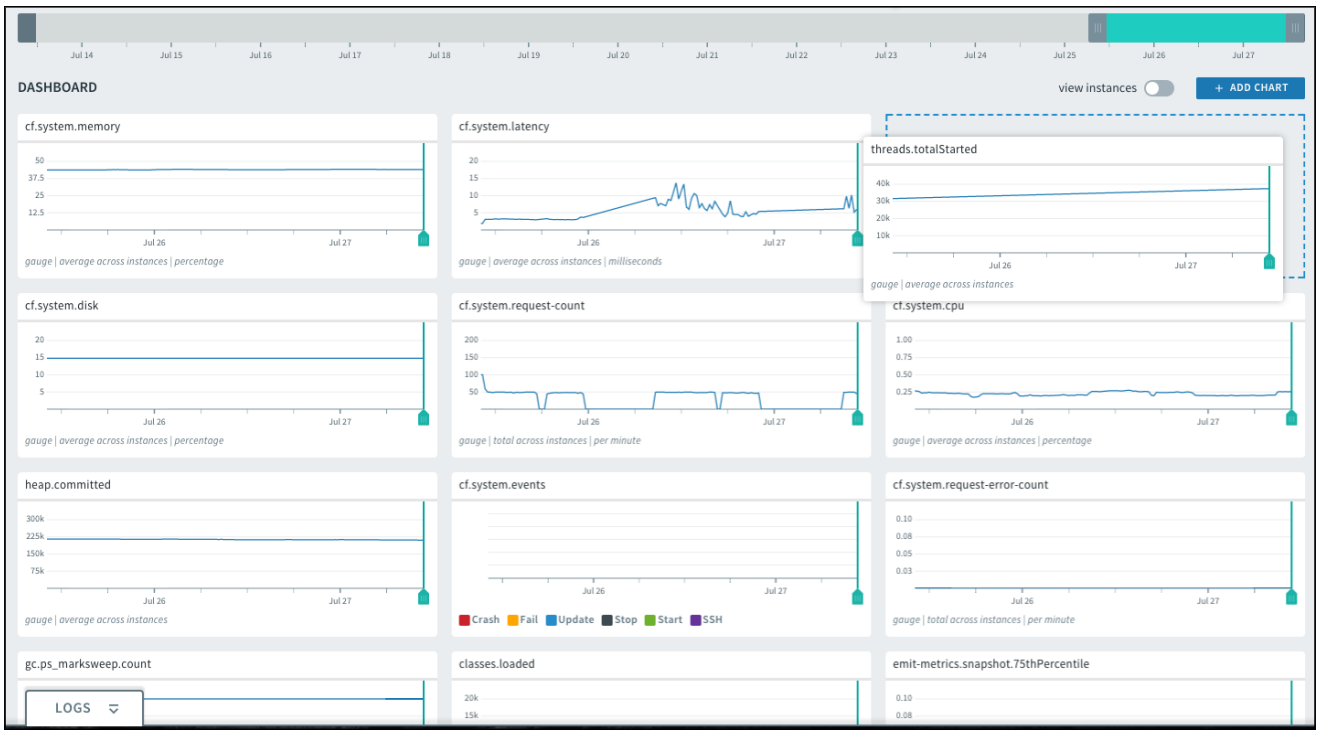


Delete Chart: To delete an existing chart on the dashboard, click the trash can icon on the header of the metric chart and then click **Delete**.

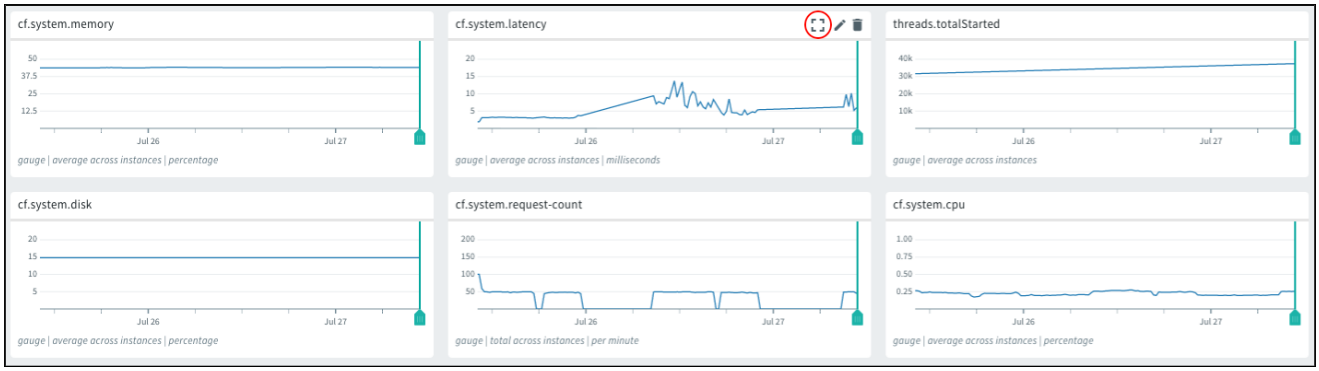


View and Reorder Metric Charts

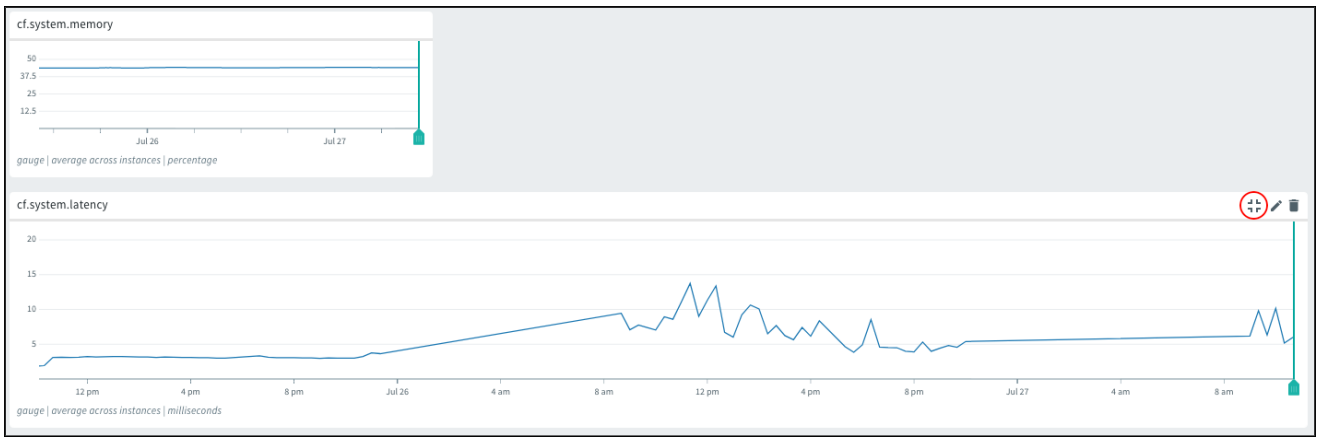
Reorder: Each metric has its own chart. You can click and drag the chart header to change the ordering of charts.



Expand: To see more details in complex graphs, you can expand a chart by clicking the icon in the chart header.



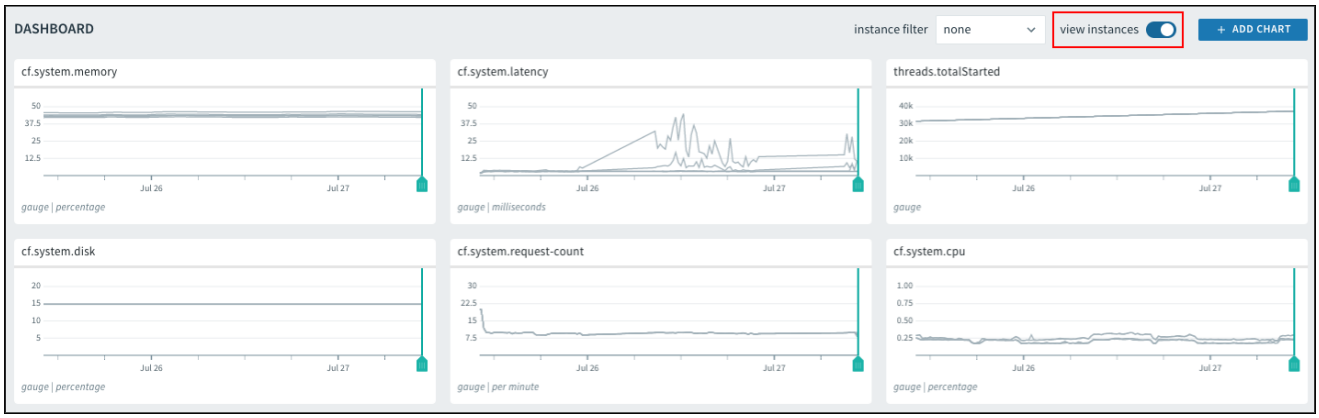
You can collapse the chart by clicking the icon again.



View Metrics at App-Instance Level

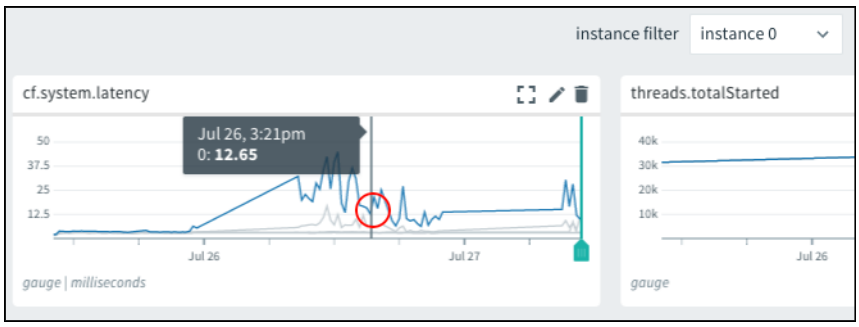
PCF Metrics relays metric data at the app-instance level to allow for an in-depth troubleshooting experience. Users are able to view the app metrics related to a specific instance index, which correlates directly with the app instance indices shown in Apps Manager.

To view metrics at the app-instance level, turn the **view instances** toggle on.



To select or deselect a specific app instance, select the desired instance from the **instance filter** dropdown menu.

Alternatively, click an instance line on the metric chart that interests you to select the instance.

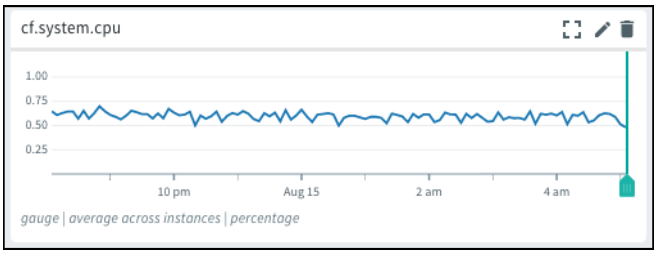


Interpret Metrics

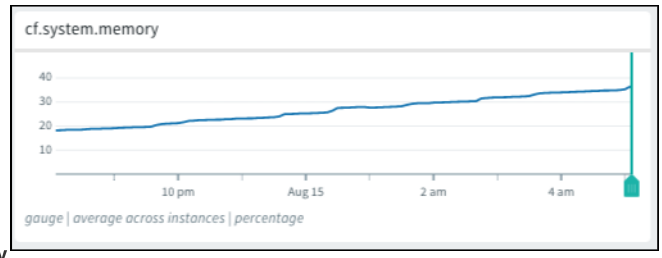
See the following sections to understand how to use each of the views on the dashboard to monitor and troubleshoot your app.

Container Metrics

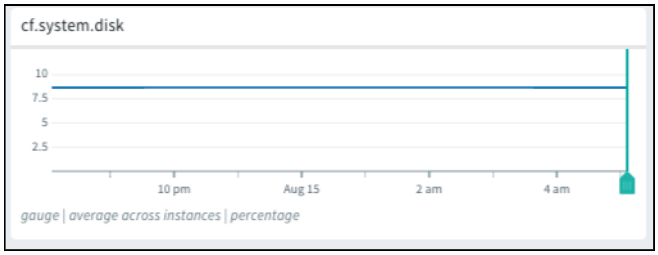
Three **Container Metrics** charts are available on the PCF Metrics dashboard:



- CPU usage percentage: **cf.system.cpu**
  A spike in CPU might point to a process that is computationally heavy. Scaling app instances can relieve the immediate pressure, but you need to investigate the app to better understand and fix the root cause.



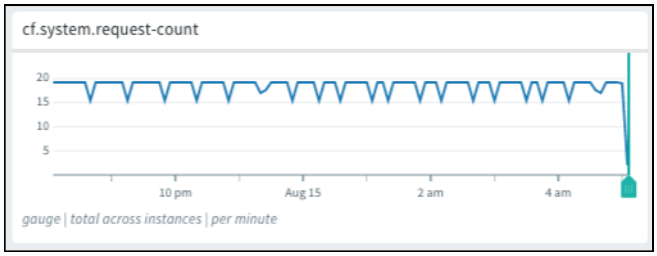
- Memory usage percentage: **cf.system.memory**
  A spike in memory might mean a resource leak in the code. Scaling app memory can relieve the immediate pressure, but you need to find and resolve the underlying issue so that it does not occur again.



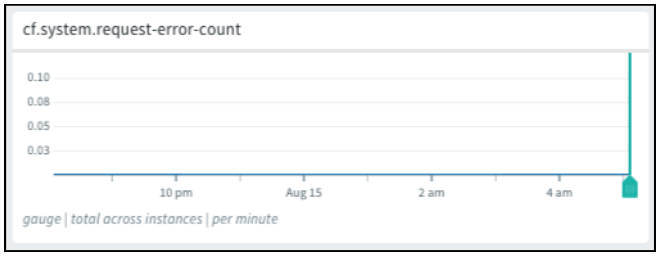
- Disk usage percentage: **cf.system.disk**
  A spike in disk might mean the app is writing logs to files instead of STDOUT, caching data to local disk, or serializing large sessions to disk.

Network Metrics

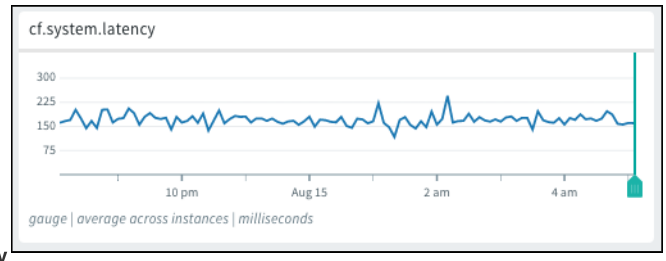
Three **Network Metrics** charts are available on the PCF Metrics dashboard:



- Number of network requests per minute: **cf.system.request-count**
  A spike in HTTP requests means more users are using your app. Scaling app instances can reduce the response time.
- Number of network request errors per minute: **cf.system.request-error-count**



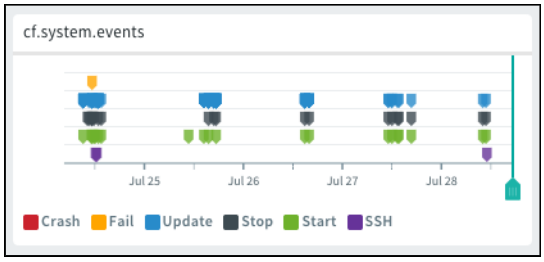
  A spike in HTTP errors means one or more 5xx errors have occurred. Check your app logs for more information.



- Average latency of a request in milliseconds: **cf.system.latency**
  A spike in response time means your users are waiting longer. Scaling app instances can spread that workload over more resources and result in faster response times.

Events

The **Events** graph shows the following app events: **Crash**, **Fail** (staging failures), **Update**, **Stop**, **Start**, and **SSH**.



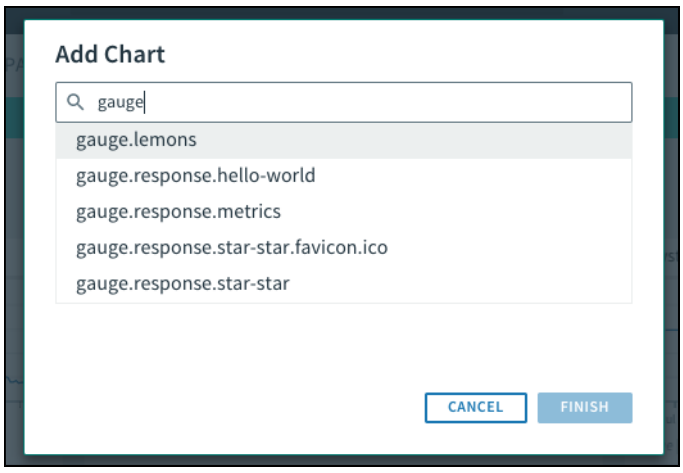
Note: The **SSH** event corresponds to someone successfully using SSH to access a container that runs an instance of the app.

See the following topics for more information about app events:

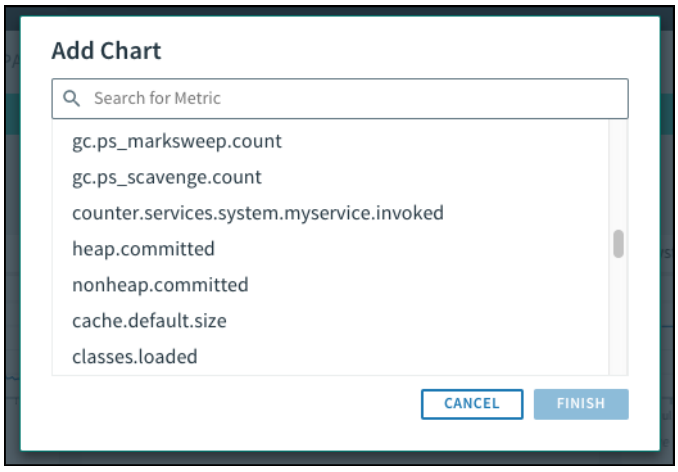
- About Starting Applications
- Troubleshooting Application Deployment and Health

Custom Metrics

Users can configure their apps to emit custom metrics out of the Loggregator Firehose and then view these metrics on the PCF Metrics dashboard. For steps on how to set up your apps to emit custom metrics, refer to the Metrics Forwarder Documentation. If you have configured the apps correctly, you should be able to automatically see custom metrics on the PCF Metrics dashboard when you add a chart.

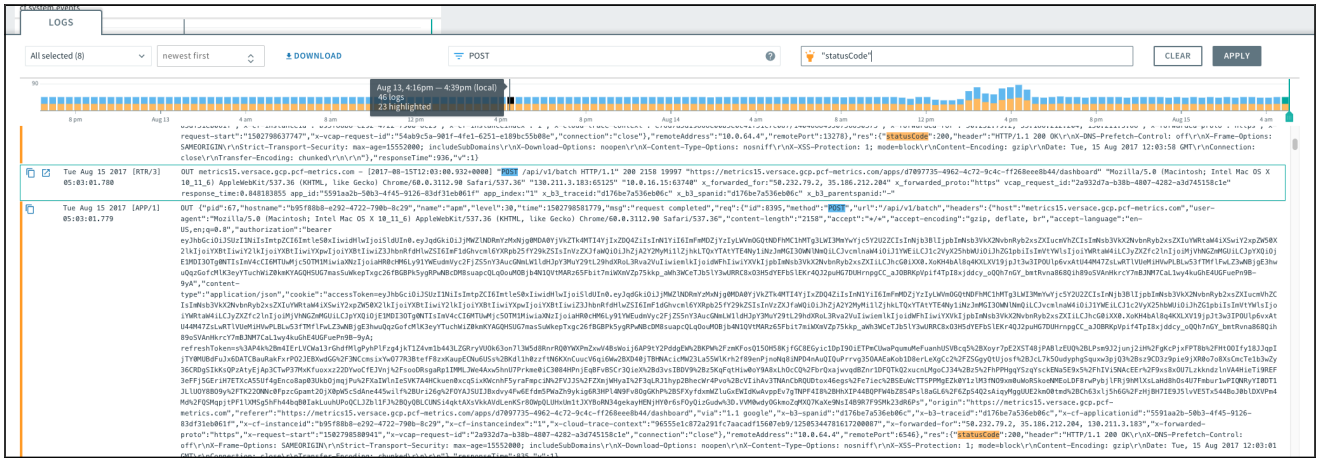


In addition, Spring Boot apps with actuators implemented emit Spring Boot Actuator metrics out of the box, without any changes to source code. In PCF Metrics, these metrics look similar to the following:



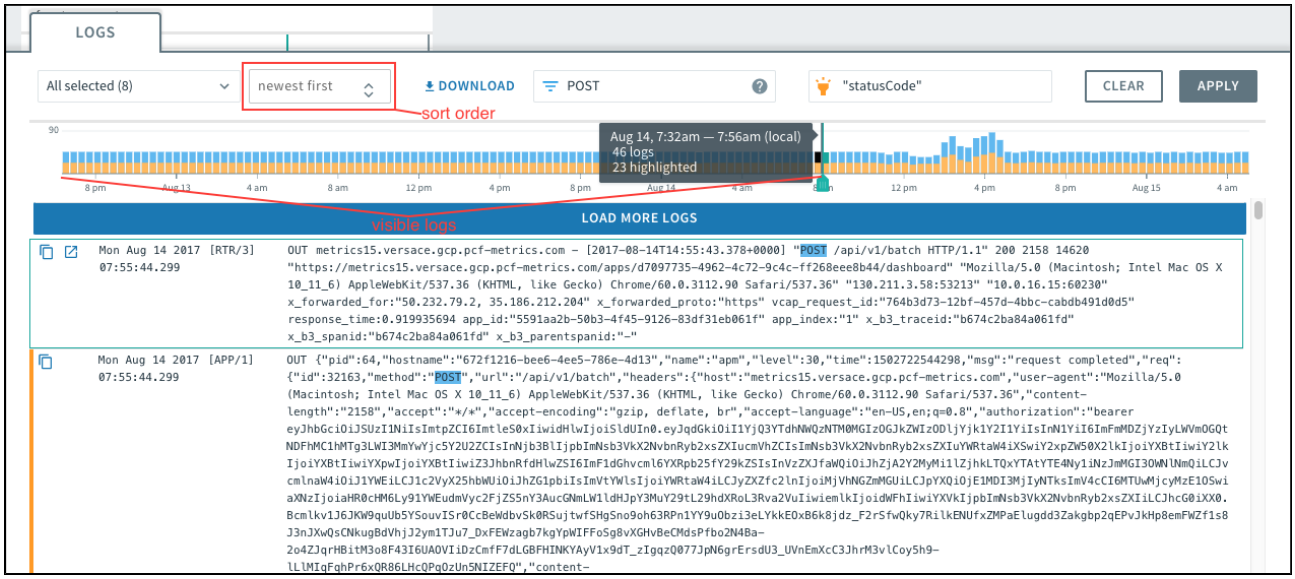
Logs

The **Logs** view displays app log data ingested from the Loggregator Firehose, including a histogram that shows log frequency for the current time frame:



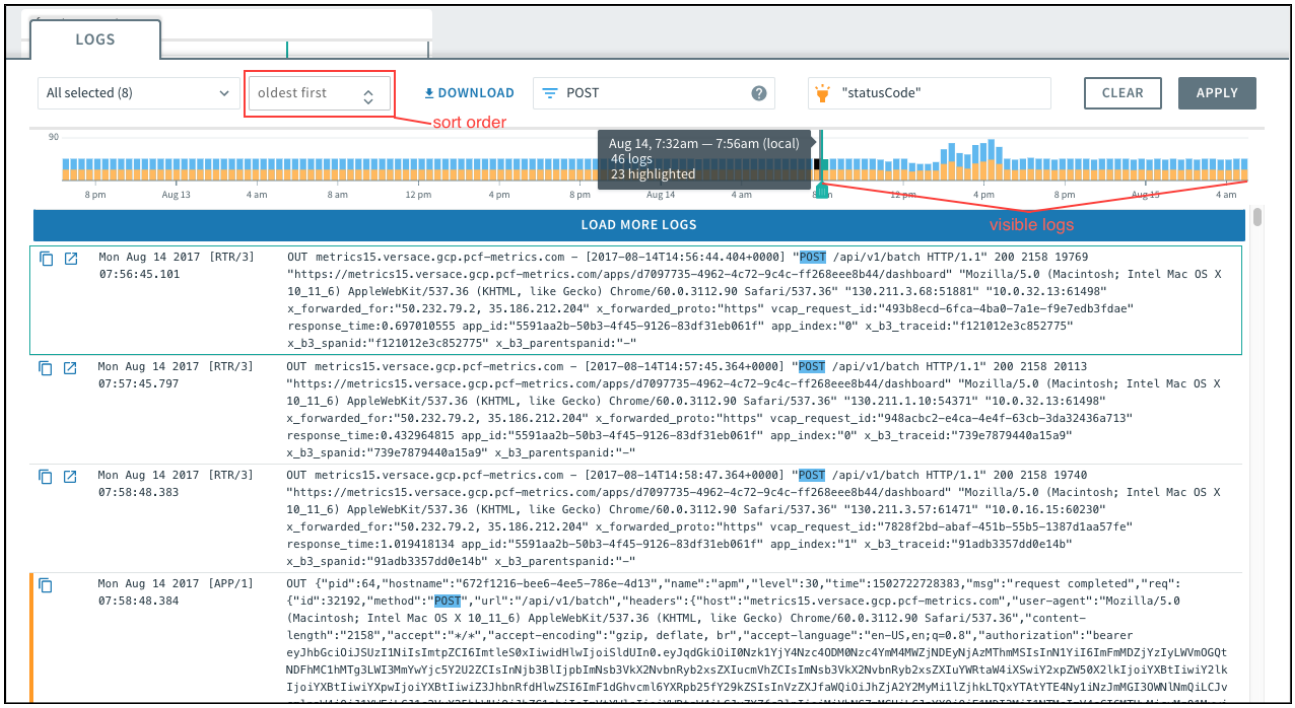
The green time needle visible on metrics charts and the logs histogram indicates the beginning of the logs. Depending on the sort order of your logs, you can see different results:

- Sort by **newest first (default)**:



The logs drawer retrieves all logs in the selected time frame that is *older than/to the left of* the needle. The log outlined in green is the newest log among the logs located to the left of the needle placement.

- Sort by **oldest first**:



The logs drawer retrieves all logs in the selected time frame that is *newer than/to the right of* the needle. The log outlined in green is the oldest log among the logs located to the right of the needle placement.

To adjust the placement of the time needle, click the handle at the bottom of the needle and drag to reposition it. Alternatively, you can click anywhere along the x-axis of a metric chart or the logs histogram to snap the needle to that position.

You can interact with the **Logs** view in the following ways:

- **Keyword**: Perform a keyword search. The histogram updates with blue bars based on what you enter. Hover over a histogram bar to view the number of logs for a specific time.
- **Highlight**: Enter a term to highlight within your search. The histogram updates with yellow bars based on the results. Hover over a histogram bar to view the number of logs for a specific time that contain the highlighted term.
- **Sources**: Choose which sources to display logs from. For more information, see Log Types and Their Messages.
- **Order**: Modify the order in which logs appear.
- **Download**: Download a file containing logs for the current search.
- **Copy**: Click the copy icon to copy the text of the log.

- **View in Trace Explorer**: Open a window to see the trace of the request associated with the log. See Trace App Requests.

Trace App Requests

A request to one of your apps initiates a workflow within the app or system of apps. The record of this workflow is a *trace*, which you can use to troubleshoot app failures and latency issues. In the Trace Explorer view, PCF Metrics displays an interactive graph of a trace and its corresponding logs. See the sections below to understand how to use the Trace Explorer.

For more information about traces, see What is a Trace? in the *Open Tracing* documentation.

Prerequisites

PCF Metrics constructs the Trace Explorer view using trace IDs shared across app logs. Before you use the Trace Explorer, examine the following list to ensure PCF metrics can extract the necessary data from your app logs for your specific app type.

- **Spring**: Follow the steps below.

  1. Ensure you are using Spring Boot v1.4.3 or later.
  2. Ensure you are using Spring Cloud Sleuth v1.0.12 or later.
  3. Add the following to your app dependency file:

  ```
  dependencies { (2)
  compile "org.springframework.cloud:spring-cloud-starter-sleuth"
  }
  ```

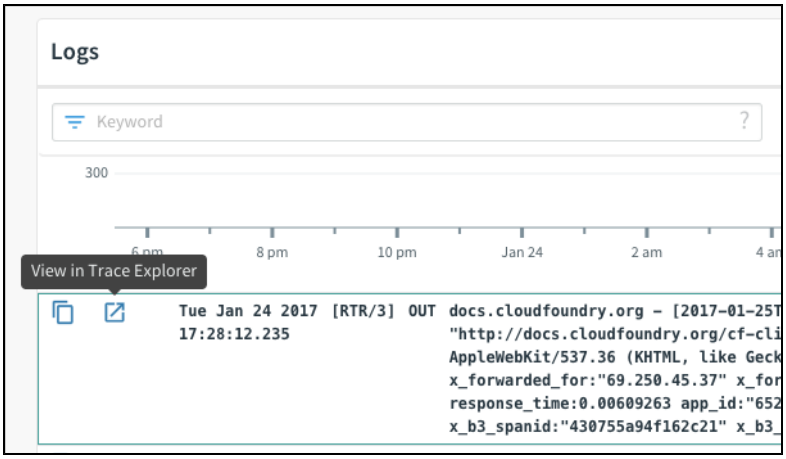
- **Node.js**, **Go**, and **Python**: Ensure that the servers associated with your app do not modify HTTP requests in a way that removes the `X-B3-TraceId`, `X-B3-SpanId`, and `X-B3-ParentSpan` headers from a request. You also need to add Trace ID, Span ID, and Parent Span ID to the SLF4J MDC in your app logs to correlate logs within the Trace Explorer.
- **Ruby**: Ruby servers that use a library depending on Rack modify HTTP request headers in a way that is incompatible with PCF Metrics. If you want to trace app requests for your Ruby apps, ensure that your framework does not rely on Rack. You may need to write a raw Ruby server that preserves the `X-B3-TraceId`, `X-B3-SpanId`, and `X-B3-ParentSpan` headers in the request. You also need to add Trace ID, Span ID, and Parent Span ID to the SLF4J MDC in your app logs to correlate logs within the Trace Explorer.

Use the Trace Explorer

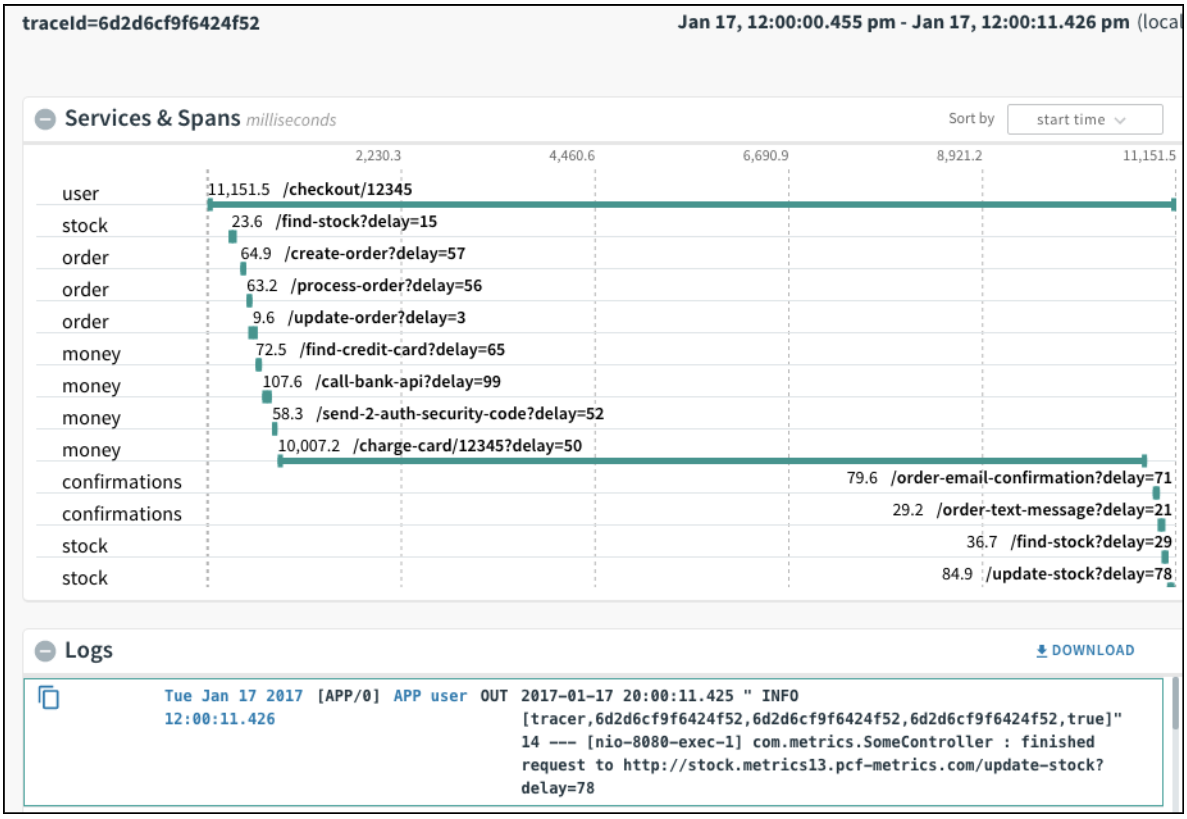
This section explains how to view the trace for a request received by your app and interact with the Trace Explorer.

1. Select an app on the PCF Metrics dashboard.

2. Click the Trace Explorer icon in a log for which you want to trace the request.



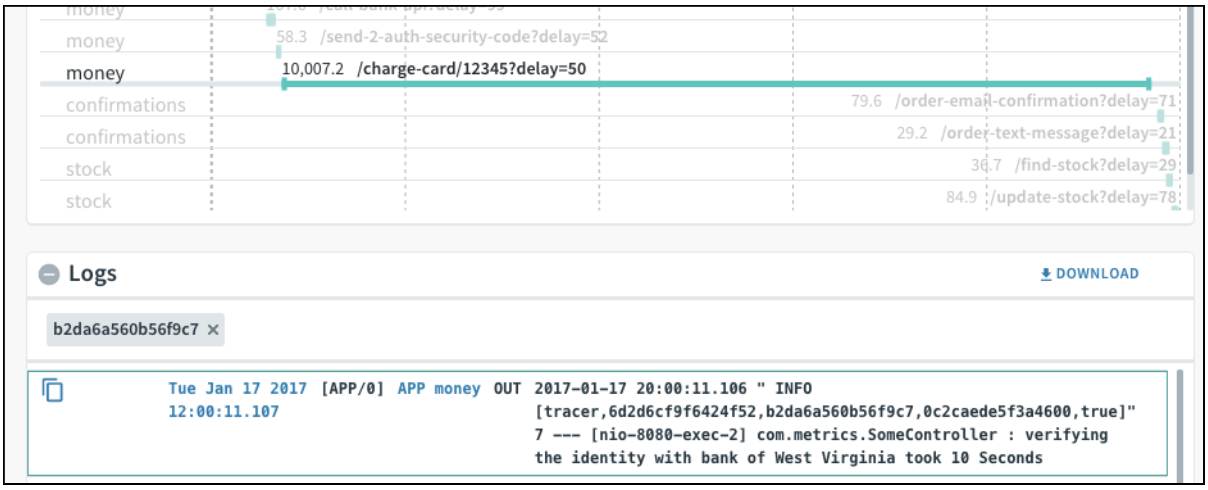
   - The Trace Explorer displays the apps and endpoints involved in completing a request, along with the corresponding logs:



A request corresponds to a single trace ID displayed in the top left corner. Each row includes an app in the left column and a *span* in the right column. A span is a particular endpoint within the app and the time it took to execute in milliseconds. By default, the graph lists each app and endpoint in the order they were called.

Note: If you do not have access to the space for an app involved in the request, you cannot see the spans or logs from that app.

- You can click a span to show only logs from that span or any number of spans to toggle which logs appear. Clicking a span also creates a box with that particular span ID in the **Logs** view:



- If you click `APP APP-NAME` within a log, PCF Metrics returns you to the dashboard view for that app, with the time frame focused on the time of the log that you clicked from.

Monitoring PCF Metrics

This topic explains how to monitor the health of the Pivotal Cloud Foundry (PCF) Metrics service using the logs, metrics, and Key Performance Indicators (KPIs) emitted by Cloud Foundry and the Metrics app itself.

Key Performance Indicators

KPIs for PCF Metrics are the metrics that operators find most useful for monitoring their PCF Metrics service. KPIs are high-signal-value metrics that can indicate emerging issues.

Pivotal provides the following KPIs as general alerting and response guidance for typical PCF Metrics installations. Pivotal recommends that operators continue to fine-tune the alert measures to their installation by observing historical trends. Pivotal also recommends that operators expand beyond this guidance and create new, installation-specific monitoring metrics, thresholds, and alerts based on learning from their own installations.

BOSH Metrics

All BOSH-deployed components generate the following metrics. Monitor them to ensure that they are not consuming excess resources.

system.mem.percent	
Description	Percentage used of the VM Memory for MySQL, Redis, ElasticSearch data, and ElasticSearch master. **Use**: Too much VM Memory usage will likely negatively impact data storage and access performance. **Origin**: JMX Bridge or BOSH HM **Type**: percent **Frequency**: 30 s (default), 10 s (configurable minimum)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > N/A **Red critical**: > 80%
Recommended response	Scale up as appropriate.

disk.persistent.percent	
Description	Percentage used of the VM persistent disk for MySQL, Redis, ElasticSearch data, and ElasticSearch master. **Use**: It is important to make sure that the system disks of data services do not fill up and cause data loss and performance degradation. **Origin**: JMX Bridge or BOSH HM **Type**: percent **Frequency**: 30 s (default), 10 s (configurable minimum)
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > N/A **Red critical**: > 80%
Recommended response	Scale up as appropriate.

Component Metrics

All apps pushed using Cloud Foundry automatically emit the following app component metrics. PCF Metrics is a collection of apps like any other CF apps, and thus can be monitored by PCF Metrics (among other monitoring services). The following KPIs can indicate problems with your installation.

system.mem.percent	
	Percentage used of the app container memory for PCF Metrics apps (metrics-ingestor, mysql-logqueue,

Description	elasticsearch-logqueue, metrics, metrics-ui). **Use**: PCF Metrics apps running out of memory will likely negatively impact performance. **Origin**: Firehose **Type**: percent **Frequency**: Every minute
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > N/A **Red critical**: > 80%
Recommended response	Scale up as appropriate.

disk.persistent.percent	
Description	Percentage used of the app container persistent disk for PCF Metrics apps (metrics-ingestor, mysql-logqueue, elasticsearch-logqueue, metrics, metrics-ui). **Use**: PCF Metrics apps running out of disk will likely negatively impact performance. **Origin**: Firehose **Type**: percent **Frequency**: Every minute
Recommended measurement	Average over last 10 minutes
Recommended alert thresholds	**Yellow warning**: > N/A **Red critical**: > 80%
Recommended response	Scale up as appropriate.

Custom Metrics

All PCF Metrics apps are already set up to emit certain custom metrics to indicate app health. As long as you make the appropriate configurations to export these metrics, they can be monitored by PCF Metrics (among other monitoring services). The following KPIs can indicate problems with your installation. Please refer to the Metrics Forwarder documentation for more information about how to set up custom metrics with the Metrics Forwarder.

metric_processor.envelopes_stored.rate.1_minute	
Description	The rate at which metrics processor stores metric envelopes to its persistent data store for mysql-logsqueue. **Use**: Zero-value rate indicates that no metrics have been stored, which is likely caused by some major metrics processing errors or failures. **Origin**: Firehose **Type**: count per minute **Frequency**: Every minute
Recommended measurement	At all times for the past 30 minutes
Recommended alert thresholds	Below 0 at all times for the past 30 minutes
Recommended response	Consult the troubleshooting document for further guidance.

log_processor.envelopes_stored.rate.1_minute	
Description	The rate at which log processor stores log envelopes to its persistent data store for PCF Metrics apps (metrics-ingestor, mysql-logqueue, elasticsearch-logqueue, metrics, metrics-ui). **Use**: Zero-value rate indicates that no logs have been stored, which is likely caused by some major logs processing errors or failures. **Origin**: Firehose

	Origin: Firehose **Type**: count per minute **Frequency**: Every minute
Recommended measurement	At all times for the past 30 minutes
Recommended alert thresholds	Below 0 at all times for the past 30 minutes
Recommended response	Consult the troubleshooting document for further guidance.

If you have any further questions regarding monitoring PCF Metrics, refer to the PCF Metrics troubleshooting guide.

Troubleshooting PCF Metrics

This topic describes how to resolve common issues experienced while operating or using Pivotal Cloud Foundry (PCF) Metrics.

Errors during Deployment

The following sections describe errors that cause failure during a PCF Metrics tile and how to troubleshoot them.

Smoke Test Errors

PCF Metrics runs a set of smoke tests during installation to confirm system health. If the smoke tests discover any errors, you can find a summary of those errors at the end of the errand log output, including detailed logs about where the failure occurred.

The following tables describe common failures and how to resolve them.

Insufficient Resources

Error	`Insufficient Resources`
Cause	Your PCF deployment has insufficient Diego resources to handle the apps pushed as part of a PCF Metrics installation. The PCF Metrics tile deploys the following apps: **App** / **Memory** / **Disk** `metrics-ingestor`* — 256 MB — 1 GB `mysql-logqueue`* — 512 MB — 1 GB `elasticsearch-logqueue`* — 256 MB — 1 GB `metrics` — 1 GB — 2 GB `metrics-ui` — 64 MB — 1 GB *You may have more than one instance of each of the Ingestor and Logqueue apps depending your sizing needs. You configure these instance counts as part of the Data Store pane of the tile.
Solution	Increase the number of Diego cells so that your PCF deployment can support the apps pushed as part of the PCF Metrics installation: 1. Navigate to the **Resource Config** section of the Elastic Runtime tile. 2. In the **Diego Cell** row, add another **Instance**.

Failed Querying MySQL

Error	`Failed querying mysql`
Cause	The tile deployed without the necessary errands selected to keep the internal database schema in sync with apps.
Solution	Re-deploy the tile with the following errands selected: • **Migrate Old Data to 1.4 Errand** • **Push PCF Metrics Components Errand**

Received No Results Back from MySQL - Failing

Error	`Received no results back from mysql - failing`
Cause	The Ingestor is not functioning properly.

Solution	1. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment:
	`$ cf target -o system -s metrics-v1-4`
	2. Run `cf apps` to see if these apps are running:
	◦ `metrics-ingestor`
	◦ `mysql-logqueue`
	3. If the apps are not running, run the following commands to start them:
	`$ cf start metrics-ingestor`
	`$ cf start mysql-logqueue`
	4. Run the following commands and search the app logs for `ERROR` messages containing additional information:
	`$ cf logs metrics-ingestor --recent`
	`$ cf logs mysql-logqueue --recent`
	Note: In some cases, the apps cannot communicate due to TLS certificate verification failure. If your deployment uses self-signed certs, ensure the **Disable SSL certificate verification for this environment** box is selected in the Elastic Runtime **Networking** pane.

Failed to Connect to MySQL

Error	`Failed to connect to mysql`
Cause	MySQL is not running properly.

Solution	1. Check the logs of the MySQL Server and MySQL Proxy jobs for errors.
	◦ You can download the logs from the PCF Metrics tile under the**Status** tab.
	2. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment:
	`$ cf target -o system -s metrics-v1-4`
	3. Run the following command and ensure the security group can access the MySQL jobs:
	Note: PCF Metrics creates a default security group to allow all traffic to its apps.
	`$ cf security-group metrics-api`

Failed to Start Elasticsearch Client

Error	`Failed to start elasticsearch client`
Cause	Elasticsearch is not running correctly.
Solution	1. Check the logs of the Elasticsearch Master, Elasticsearch Coordinator, and Elasticsearch Data jobs for errors. ◦ You can download the logs from the PCF Metrics tile under the**Status** tab. 2. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment: `$ cf target -o system -s metrics-v1-4` 3. Run the following command and ensure the security group can access the Elasticsearch jobs:

	Note: PCF Metrics creates a default security group to allow all traffic to its apps. `$ cf security-group metrics-api`

Never Received App Logs

Error	`Never received app logs - something in the firehose -> elasticsearch flow is broken`
Cause	Ingestor is not inserting logs correctly.

Solution	1. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment:
	
	`$ cf target -o system -s metrics-v1-4`
	
	2. Run `cf apps` to see if these apps are running:
	
	- `metrics-ingestor`
	- `elasticsearch-logqueue`
	
	3. If the apps are not running, run the following commands to start them:
	
	`$ cf start metrics-ingestor`
	`$ cf start elasticsearch-logqueue`
	
	4. Run the following commands and search the app logs for `ERROR` messages containing additional information:
	
	`$ cf logs metrics-ingestor --recent`
	`$ cf logs elasticsearch-logqueue --recent`
	
	Note: In some cases, you might discover a failure to communicate with Loggregator in the form of a bad handshake error.
	
	Ensure the **Loggregator Port** setting in the Elastic Runtime tile **Networking** pane is set to the correct value. For AWS, it is `4443`. For all other IaaSes, it is `443`.

Metrics and Events Not Available

Error	`Network metrics are not available.`
	`Container metrics are not available.`
	`App events are not available.`

Cause	PCF Metrics is misconfigured and the frontend API does not receive logs from MySQL.

Solution	1. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment:
	
	`$ cf target -o system -s metrics-v1-4`
	
	2. Run the following command to check the app logs and investigate the error:
	
	`$ cf logs metrics --recent`

Logs and Histograms Not Available

Error	`Logs are not available.` `Histograms are not available.`
Cause	PCF Metrics is misconfigured and the frontend API does not receive logs from Elasticsearch.

Solution	1. From the cf CLI, target the `system` org and `metrics-v1-4` space of your PCF deployment:
	`$ cf target -o system -s metrics-v1-4`
	2. Run the following command to check the app logs and investigate the error:
	`$ cf logs metrics --recent`

No Logs or Metrics in the UI

In some cases, the PCF Metrics UI might not display metrics and logs after successfully deploying.

Follow the steps in this section to help locate the app or component causing the problem.

Step 1: Check your Load Balancer Configuration

If you use a load balancer, the event-stream mechanism used by the Metrics UI might be blocked. See the table below to resolve this error.

If you do not use a load balancer, or this issue does not apply to your deployment, proceed to Step 2: Check the PCF Metrics Apps.

Error	In the case of a customer using an F5 load balancer, metrics and logs were not visible in the UI despite successful ingestion and no UI errors reported.
Cause	The root of the issue was the traffic of type text/event-stream was blocked by the F5 load balancer.
Solution	When F5 was configured to allow event-stream traffic, the issue was resolved.

Step 2: Check the PCF Metrics Apps

1. From Ops Manager, click the Elastic Runtime Tile.
    a. Click the **Credentials** tab.
    b. Under the **UAA** job, next to **Admin Credentials**, click **Link to Credential**.
    c. Record the username and password for use in the next step.
2. Log in to the Cloud Foundry Command Line Interface (cf CLI) using the credentials from the previous step.

    ```
    $ cf login -a https://api.YOUR-SYSTEM-DOMAIN -u admin -p PASSWORD
    ```

3. When prompted, select the `system` org and the `metrics-v1-4` space.
4. Ensure that the output displays the following apps, each in a `started` state:
    - `metrics-ingestor`
    - `mysql-logqueue`
    - `elasticsearch-logqueue`
    - `metrics-aggregator`
    - `metrics`
    - `metrics-ui`
5. Check the logs of each app for errors using the following command:

    ```
    $ cf logs APP-NAME --recent
    ```

    If you do not see any output, or if you did not find any errors, proceed to Step 3: Check the Elasticsearch Cluster.

Step 3: Check the Elasticsearch Cluster

1. From Ops Manager, select the PCF Metrics tile.
2. Under the **Status** tab, record the IP of an **Elasticsearch Master** node.
3. Use `bosh ssh` to access the VM from the previous step. For instructions, see Advanced Troubleshooting with the BOSH CLI.
4. Run the following command to list all the Elasticsearch indices:

```
$ curl ELASTICSEARCH-HOST-IP:9200/_cat/indices?v | sort


green  open   app_logs_1477512000   8   1  125459066          0    59.6gb       29.8gb
green  open   app_logs_1477526400   8   1  129356671          0    59.1gb       29.5gb
green  open   app_logs_1478174400   8   1  129747170          0    61.9gb       30.9gb
. . .
green  open   app_logs_1478707200   8   1  128392686          0    63.2gb       31.6gb
green  open   app_logs_1478721600   8   1  102005754          0    53.5gb       26.5gb
health status index              pri rep docs.count docs.deleted store.size pri.store.size
```

   a. If the `curl` command does not return a `success` response, Elasticsearch might not even be running correctly. Inspect the following logs for any failures or errors:

      - `/var/vcap/sys/log/elasticsearch/elasticsearch.stdout.log`
      - `/var/vcap/sys/log/elasticsearch/elasticsearch.stderr.log`

5. Examine the `status` column of the output.

   a. If any of the indices are `red`, delete them using the following command:

```
curl -X DELETE ELASTICSEARCH-HOST-IP:9200/INDEX
```

   b. Restart the `elasticsearch-logqueue` app:

```
$ cf restart elasticsearch-logqueue
```

   c. From each of the Elasticsearch VMs, run the following command:

```
$ monit restart all
```

   d. Check periodically to verify the indices gradually recover to a `green` status.

6. Run the `curl` command several more times and examine the most recent index to see if the number of stored documents periodically increases.

> **Note**: The last row of the output corresponds to the most recent index. The sixth column displays the number of documents for the index.

   a. If all indices show a `green` status, but the number of documents does not increase, there is likely a problem further up in ingestion. Proceed to to Step 4: Check the Elasticsearch Logqueue.

7. Check whether cluster-level shard allocation is enabled:

```
$ curl localhost:9200/_cluster/settings
```

   Examine the output:

   - `"all"` means the shard allocation is enabled.
   - `"none"` means the shard allocation is disabled.

8. Re-enable the shard allocation by running the following command:

```
$ curl -XPUT localhost:9200/_cluster/settings -d
{"transient" : {
  "cluster.routing.allocation.enable" : "all"
}}
```

9. Check whether a proxy is present in front of Elasticsearch and whether HTTP traffic is enabled:

a. Use `cf ssh` to SSH into any app in the metrics space:

```
$ cf ssh APP-NAME
```

b. Run the `curl` command with the IP address of the **Elasticsearch Master** node:

```
$ curl ELASTICSEARCH-MASTER-IP-ADDRESS
```

c. If the `curl` command fails, talk to your system administrator about removing the proxy or green-listing the PCF Metrics apps.

Step 4: Check the Elasticsearch Logqueue

1. Run `cf apps` to see if the `elasticsearch-logqueue` app instances are `started`.
2. If any instance of the app is `stopped`, run the following command to increase logging:

```
$ cf set-env elasticsearch-logqueue LOG_LEVEL DEBUG
```

a. Run the following command to stream logs:

```
$ cf logs elasticsearch-logqueue
```

b. In a different terminal window, run the following command:

```
$ cf restage elasticsearch-logqueue
```

c. Watch the logs emitted by the `elasticsearch-logqueue` app for errors.

- A common error is that the app cannot connect to Elasticsearch because a user deleted the application security group (ASG) that PCF Metrics creates to allow the Logqueue app to connect to the Elasticsearch VMs. You can run `cf security-group metrics-api` to see if the ASG exists. If not, see Creating Application Security Groups.

3. If the app is started and you do not find any errors, proceed to Step 5: Check the Metrics Ingestor.

Step 5: Check the Metrics Ingestor

1. Run `cf apps` to see if the `metrics-ingestor` app instances are `started`.
2. If any of the app instances are `stopped`, run the following command to increase logging:

```
$ cf set-env metrics-ingestor LOG_LEVEL DEBUG
```

a. Run the following command to stream logs:

```
$ cf logs metrics-ingestor
```

b. In a different terminal window, run the following command:

```
$ cf restage metrics-ingestor
```

c. Watch the logs emitted by the `metrics-ingestor` app for errors. See the list below for common errors:

- **Cannot connect to the firehose**: PCF Metrics creates a UAA user to authenticate the connection to the Firehose. This user must have the `doppler.firehose` authority.
- **Cannot connect to the logqueues**: There might be a problem with the UAA, or it could be throttling traffic.
- **WebSocket Disconnects**: If you see WebSocket disconnects logs in the Ingestor app, consider adding additional Ingestor instances. The Firehose might be dropping the Ingestor connection to avoid back pressure.

3. If the app is started and you do not find any errors, proceed to Step 6: Check MySQL.

Step 6: Check MySQL

1. From Ops Manager, select the PCF Metrics tile.
2. Under the **Status** tab, record the IP of a **MySQL Server** node.
3. Use `bosh ssh` to access the VM from the previous step. For instructions, see Advanced Troubleshooting with the BOSH CLI.
4. Log in to mysql by running `mysql -u USERNAME -p PASSWORD`

   Note: If you do not know the username and password, you can run `cf env mysql-logqueue` with the `system` org and the `metrics-v1-4` space targeted.

5. Verify that the database was bootstrapped correctly:
   a. Run `show databases` and check for a `metrics` database.
      i. If there is no `metrics` database, the `migrate_db` errand of the BOSH release might not have run or succeeded. Ensure the errand is selected in the tile configuration and update the tile.
6. Run `use metrics` to select the `metrics` database:

```
mysql> use metrics;
```

7. Run `show tables` and ensure you see the following tables:

```
mysql> show tables;
+------------------+
| Tables_in_metrics |
+------------------+
| app_event        |
| app_metric       |
| app_metric_rollup |
| schema_version   |
+------------------+
```

8. Enter the following query several times to verify that the value returned does not decrease over time:

```
mysql> select count(*) from metrics.app_metric_rollup where timestamp > ((UNIX_TIMESTAMP() - 60) * POW(10, 3));
```

   This command displays the rate at which metrics flow in over the last minute.
   a. If the command returns `0` or a consistently decreasing value, the problem is likely further up in ingestion. Proceed to Step 7: Check the MySQL Logqueue.

Step 7: Check the MySQL Logqueue

1. Run `cf apps` to see if the `mysql-logqueue` app instances are `started`.
2. If any instance of the app is `stopped`, run the following command to increase logging:

```
$ cf set-env mysql-logqueue LOG_LEVEL DEBUG
```

   a. Run the following command to stream logs:

```
$ cf logs mysql-logqueue
```

   b. In a different terminal window, run the following command:

```
$ cf restage mysql-logqueue
```

   c. Watch the logs emitted by the `mysql-logqueue` app for errors.
      - A common error is that the app cannot connect to MySQL because a user deleted the application security group (ASG) that PCF Metrics creates to allow the Logqueue app to connect to the MySQL VMs. You can run `cf security-group metrics-api` to see if the ASG exists. If not, see Creating Application Security Groups.
3. If the app is started and you do not find any errors, proceed to Step 8: Check the Metrics Aggregator.

MySQL Node Failure

In some cases, a MySQL server node might fail to restart. The following two sections describe the known conditions that cause this failure as well as steps for diagnosing and resolving them. If neither of the causes listed apply, the final section provides instructions for re-deploying BOSH as a last resort to resolve the issue.

Cause 1: Monit Timed Out

Diagnose

Follow these steps to see if a `monit` time-out caused the MySQL node restart to fail:

1. Use `bosh ssh` to access the failing node, using the IP address in the Ops Manager Director tile **Status** tab. For instructions, see Advanced Troubleshooting with the BOSH CLI.
2. Run `monit summary` and check the status of the `mariadb_ctrl` job.
3. If the status of the `mariadb_ctrl` job is `Execution Failed`, open the following file: `/var/vcap/sys/log/mysql/mariadb_ctrl.combined.log`.
    a. If the last line of the log indicates that MySQL started without issue, such as in the example below, `monit` likely timed out while waiting for the job to report healthy. Follow the steps below to resolve the issue.


```
{"timestamp":"1481149250.288255692","source":"/var/vcap/packages/
mariadb_ctrl/bin/mariadb_ctrl","message":"/var/vcap/packages/
mariadb_ctrl/bin/mariadb_ctrl.mariadb_ctrl
 started","log_level":1,"data":{}}
```


Resolve

Run the following commands to return the `mariadb_ctrl` job to a healthy state:

1. Run `monit unmonitor mariadb`.
2. Run `monit monitor mariadb`.
3. Run `monit summary` and confirm that the output lists `mariadb_ctrl` as `running`.

Cause 2: Bin Logs Filled up the Disk

Diagnose

1. Use `bosh ssh` to access the failing node. For instructions, see Advanced Troubleshooting with the BOSH CLI.
2. Open the following log file: `/var/vcap/sys/log/mysql/mysql.err.log`.
3. If you see log messages that indicate insufficient disk space, the persistent disk is likely storing too many bin logs. Confirm insufficient disk space by doing the following:
    a. Run `df -h`.
        i. Ensure that you see the `/var/vcap/store` folder is at or over `90%` usage.
    b. Navigate to `/var/vcap/store/mysql` and run `ls -al`.
        i. Ensure that you see many files named with the format `mysql-bin.########`.

In MySQL for PCF, the server node does not make use of these logs and you can remove all except the most recent bin log. Follow the steps below to resolve the issue.

Resolve

1. Log in to mysql by running `mysql -u USERNAME -p PASSWORD`

    Note: If you do not know the username and password, you can run `cf env mysql-logqueue` with the `system` org and the `metrics-v1-4` space targeted.

2. Run `use metrics;`.

3. Run the following command:

```
mysql> PURGE BINARY LOGS BEFORE 'YYYY-MM-DD HH:MM:SS';
```

Re-deploy BOSH to Restart the Node

If troubleshooting based on the causes mentioned above did not resolve the issue with your failing MySQL node, you can follow the steps below to recover it. Pivotal recommends only using this procedure as a if there are no other potential solutions available.

warning! This procedure is extremely costly in terms of time and network resources. The cluster takes a significant amount of time to put the data replicated to the rest of the cluster back into the rebuilt node. This procedure consumes considerable network bandwidth as potentially hundreds of gigabytes of data needs to transfer.

Stop the Ingestor App

1. From Ops Manager, click the Elastic Runtime Tile.

    a. Click the **Credentials** tab.
    b. Under the **UAA** job, next to **Admin Credentials**, click **Link to Credential**.
    c. Record the username and password for use in the next step.

2. Log in to the cf CLI using the credentials from the previous step.

```
$ cf login -a https://api.YOUR-SYSTEM-DOMAIN -u admin -p PASSWORD
```

3. Target the `system` org and `metrics-v1-4` space of your PCF deployment:

```
$ cf target -o system -s metrics-v1-4
```

4. Stop data flow into the Galera cluster:

```
$ cf stop metrics-ingestor
```

Edit Your Deployment Manifest

1. Follow the steps in Log in to BOSH in *Advanced Troubleshooting with the BOSH CLI* to target and log in to your BOSH Director. The steps vary slightly depending on whether your PCF deployment uses internal authentication or an external user store.

2. Download the manifest of your PCF deployment:

```
$ bosh download manifest YOUR-PCF-DEPLOYMENT YOUR-PCF-MANIFEST.yml
```

    Note: You must know the name of your PCF deployment to download the manifest. To retrieve it, run `bosh deployments` to list your deployments and locate the name of your PCF deployment.

3. Open the manifest and set the number of instances of the failed server node to `0`.

4. Run `bosh deployment YOUR-PCF-MANIFEST.yml` to specify your edited manifest.

5. Run `bosh deploy` to deploy with your manifest.
6. Run `bosh disks --orphaned` to see the persistent disk or disks associated with the failed node.
    a. Record the `CID` of each persistent disk.
    b. Contact Pivotal Support to walk through re-attaching the orphaned disks to new VMs to preserve their data.
7. Open the manifest and set the number of instances of the failed server node to `1`.
8. Run `bosh deploy` to deploy with your edited manifest.
9. Wait for BOSH to rebuild the node.

Log Errors

Error	The PCF Metrics UI does not show any new logs from Elasticsearch.
Cause	The tile deployed with the `Push PCF Metrics Data Components` errand deselected

Solution	Restart the Elasticsearch Logqueue using the cf CLI as follows:
	1. Target the `system` org and `metrics-v1-4` space of your PCF deployment:
	`$ cf target -o system -s metrics-v1-4`
	2. Run the following command to restart the Logqueue app:
	`$ cf restart elasticsearch-logqueue`
	Note: To avoid having to apply this fix in the future, select the checkbox to enable the `Push PCF Metrics Data Components` errand before your next tile update.

503 Errors

Error	You encounter `503` errors when accessing the PCF Metrics UI in your browser.
Cause	Your Elasticsearch nodes might have become unresponsive.
Solution	Check the Elasticsearch index health by following the procedure below, and consider adding additional Elasticsearch nodes. 1. Retrieve the IP address of your Elasticsearch master node by navigating to the **Metrics** tile in the Ops Manager Installation Dashboard, clicking the **Status** tab, and recording the IP address next to **ElasticSearchMaster**.  2. SSH into the Ops Manager VM by following the instructions in SSH into Ops Manager. 3. From the Ops Manager VM, use `curl` to target the IP address of your Elasticsearch master node. Follow the instructions in Cluster Health of the Elasticsearch documentation.

Failed to Fetch Apps

Error	Even though you entered the correct UAA credentials, the metrics app fails to fetch the list of apps.
Cause	The browser plugins or cookies inject extraneous content in requests to Cloud Controller API, causing it to reject the request.
Solution	Confirm the problem and clear the browser, as follows: 1. Try the browser's incognito mode to see if the metrics app is able to fetch the list of apps. If this works, the problem is likely cookies or plugins.

	2. Clear your browser cookies and plugins.

Redis Temporary Datastore Stops Accepting Metrics

Error	You see both these problems: • Metrics stop appearing on the UI. • When you run `cf metrics-ingestor logs`, you see the following entry in the Ingestor logs: `MISCONF Redis is configured to save RDB snapshots, but is currently not able to persist on disk. Commands that may modify the data set are disabled. Please check Redis logs for details about the error.`
Cause	The Redis datastore is full. The component is out of memory or persistent disk space.
Solution	Confirm the problem and scale up Redis, as follows: 1. On the **Metrics** tile, click the **Status** tab and look to see if the memory or persistent disk usage of the Redis job is over 80%. 2. Scale up the Redis component. For more information, see Scale the Temporary Datastore (Redis).

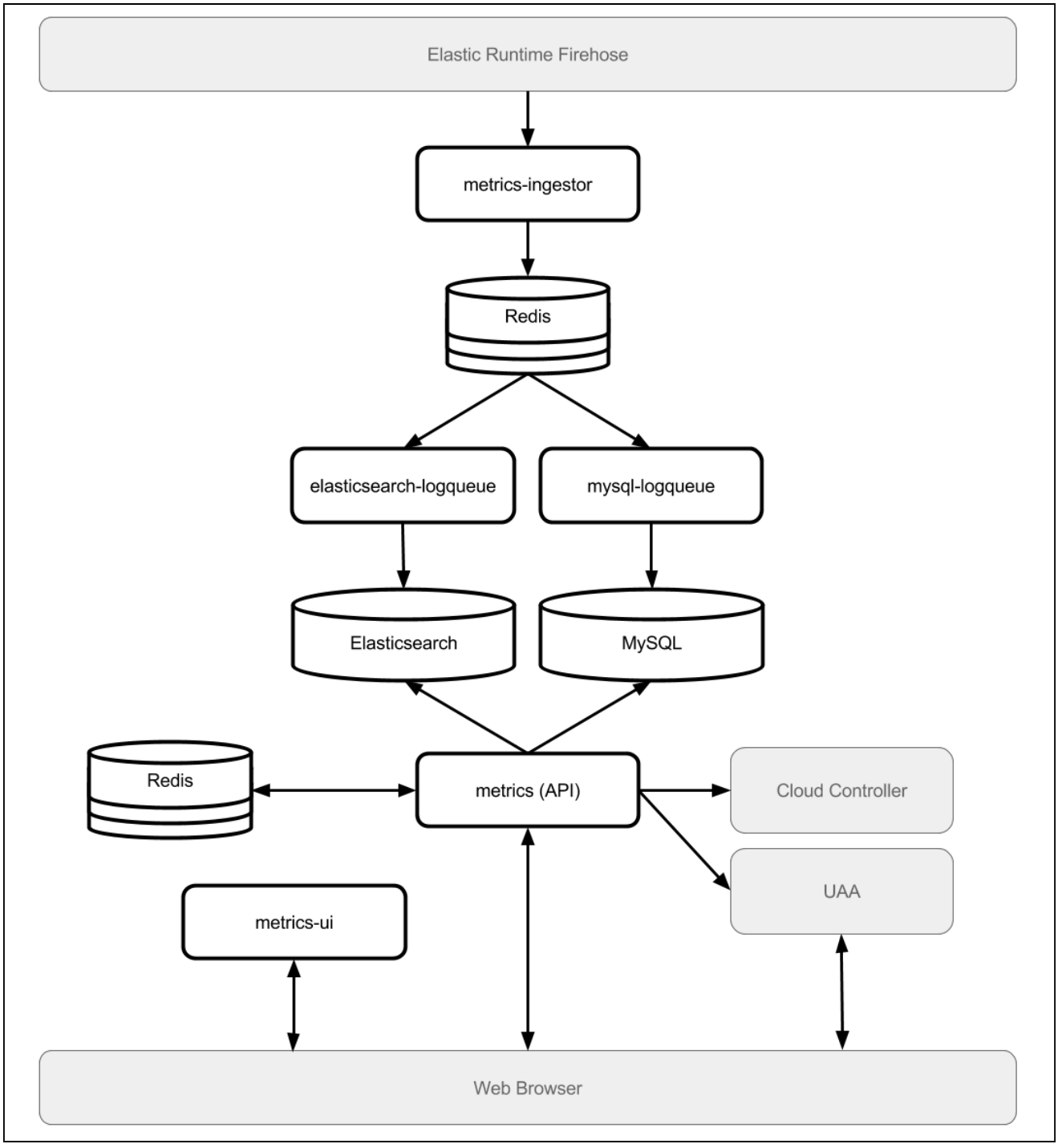
PCF Metrics Product Architecture

This topic describes the product architecture of Pivotal Cloud Foundry (PCF) Metrics.

Overview

The diagram below displays the components of PCF Metrics, as well as the Cloud Foundry components that the PCF Metrics system interacts with.

PCF Metrics deploys several Cloud Foundry apps as part of the install process. These components are the bold rectangles in the diagram. The cylinders represent the data storage components of PCF Metrics.



See the following sections to understand the processes that happen within the PCF Metrics system.

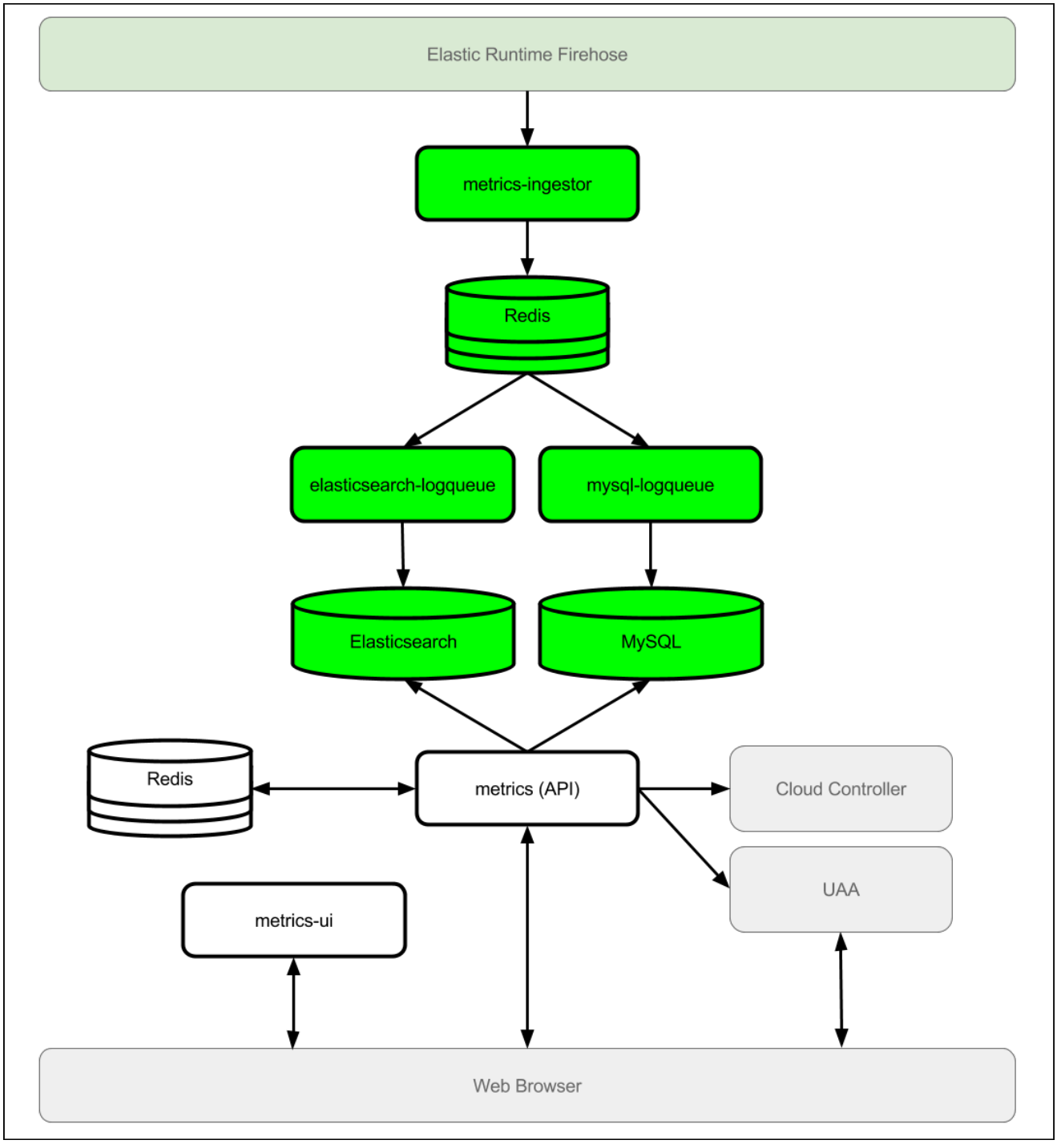
How Data Flows from the Firehose to the Datastores

This section describes how PCF Metrics fills its datastores. PCF Metrics uses three datastores:

- The MySQL component stores metric and event data from the apps running on your PCF deployment.
  - Examples of events are `start` and `stop`.
  - Examples of metrics are *container metrics* such as CPU and *network metrics* such as Requests.
- The Elasticsearch component stores logs from the apps running on your PCF deployment.
- The Redis component is used to cache data ingested from the Firehose and data queried by the Metrics API.

Components

The diagram below highlights the components involved in the process of getting metric and log data into the Elasticsearch and MySQL datastore.



Process

The following table describes how the components act during each stage.

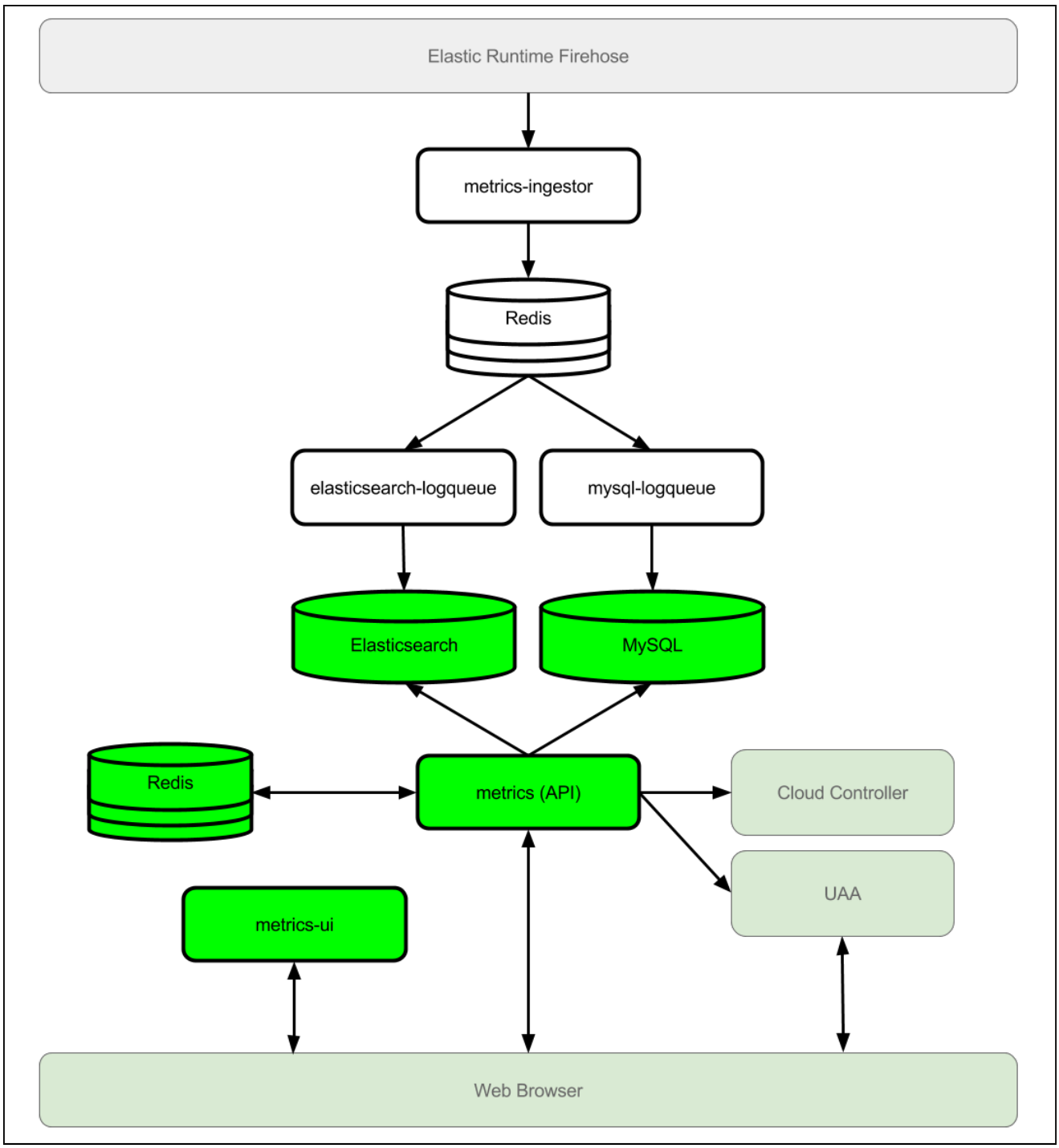
Stage	Description
1	The `metrics-ingestor` app receives app logs, container metrics, and network metrics from the Firehose and forwards them to Redis. Redis does the following: • Acts as a buffer for events and metrics data • Acts as a cache
2	Each of the logqueues acts independently, writing information to the datastores: **Elasticsearch logqueue** The `elasticsearch-logqueue` app reads data from Redis and checks whether the Elasticsearch datastore is available. If available, the app writes logs to the Elasticsearch datastore. **MySQL logqueue** The `mysql-logqueue` app reads data from Redis and checks whether the MySQL datastore is available. If the datastore is available, the app does the following: • Inserts container and network metrics into MySQL with 1-minute granularity • Parses app log messages for an app event name and inserts the event into MySQL

How the PCF Metrics UI Retrieves Data from the Datastores

This section describes the flow of data through the system when you interact with the PCF Metrics UI.

Components

The diagram below highlights the components involved in this process.



Process

The following table describes how the components act during each stage.

Stage	Description
1	A user launches `metrics.SYSTEM-DOMAIN` in a browser and enters their UAA credentials.
2	After the UAA authorizes the user, the browser does the following: 1. Retrieves through the Cloud Controller API a list of apps that the user can access 2. Displays a page in which the user can select any app returned by the Cloud Controller API
3	A user selects an app from the dropdown menu, which does the following:

	1. Opens an AJAX connection to the metrics API to retrieve metrics and logs for the specified time frame
4	The metrics API receives the requests from the browser and does the following: 1. Communicates with the UAA and Cloud Controller to confirm that the user can access data for the requested app 2. Fetches the data from MySQL and Elasticsearch and then streams it to the browser

PCF Metrics Release Notes and Known Issues

This topic contains release notes for Pivotal Cloud Foundry (PCF) Metrics.

v1.4.7

Release Date: September 12, 2018

Notes

The following list describes what's new in PCF Metrics v1.4.7:

- Updated GoLang version to `10.3`

v1.4.6

Release Date: July 16, 2018

Notes

The following list describes what's new in PCF Metrics v1.4.6:

- The `push-apps` metrics-ui app memory allocation has been bumped to 256 MB to address errand failures.
- Applied fix to allow errands to run while PCF Metrics v1.4 and v1.5 are installed simultaneously during upgrade in Ops Manager.

v1.4.5

Release Date: June 5, 2018

Notes

The following list describes what's new in PCF Metrics v1.4.5:

- The time scrubber end-time issue is fixed.
- The Node.js buildpack issue is fixed.
- golang is upgraded to v1.9.6.
- The CF CLI is upgraded to v6.36.2 to support golang v1.9.x reversion.

v1.4.4

Release Date: April 24, 2018

Notes

The following list describes what's new in PCF Metrics v1.4.4:

- Upgraded to the cf CLI v6.36.1 to resolve known `cf auth` errand failures.

v1.4.3

Release Date: January 17, 2018

Notes

The following list describes what's new in PCF Metrics v1.4.3:

- UI bug fixes for metrics dropdown selection on dashboard
- Upgraded to MySQL 36.10.0
- Upgraded major stemcell version to 3445.x

v1.4.2

Release Date: October 31, 2017

Notes

The following list describes what's new in PCF Metrics v1.4.2:

- Bug fix for metrics configurable retention window, which will ensure that if you have set the retention to < 14 days, the data is correctly pruned for correct partions
- Bug fix for delay in getting metrics from apps to PCF Metrics datastore. In 1.4.0, it was around 15 mins depending on # of apps and metrics emitted.

v1.4.0

Release Date: September 28, 2017

Notes

The following list describes what's new in PCF Metrics v1.4.0:

- **Custom App Metrics**: PCF Metrics v1.4.0, when used along with Metrics Forwarder, supports custom app metrics. Users can configure their apps to send the metrics to PCF Metrics and view them on the PCF Metrics dashboard. **Spring Boot Actuator metrics** will be available on the PCF Metrics dashboard as an out-of-the-box feature for Spring Boot apps.
- **Instance-Level Metrics**: The PCF Metrics dashboard now enables users to view metrics for a specific app instance. Users can toggle between app instance and aggregated views and choose their preferred method of aggregation across instances.
- **Improved UI**: PCF Metrics v1.4.0 introduces major UI enhancements of the metrics dashboard, including interactive features such as adding, deleting, reordering, and expanding charts, to support custom metrics and optimize readability.
- **Improved Tile Configuration UX and VM Footprint Reduction** The following changes have been made to the PCF Metrics tile configuration:
  - Users can configure the number of Metrics API instances under the **Metrics Components Config** section.
  - MySQL VM footprint has been reduced by switching from a Galera cluster to a single-node MySQL. The number of MySQL instances is no longer configurable by the user in the **Resource Config** section.
  - The unused Metron job was removed from **Resource Config**.
  - Redis job has been added to **Resource Config**. You can configure the number of instances and their size. For more information on how Redis is used in PCF Metrics, see PCF Metrics Product Architecture.

Known Issues

The following sections describe the known issues in PCF Metrics v1.4.x.

metrics-ui Errand Failures

The `push-apps` errand metrics-ui allocation is set to 64MB by default, which can cause the errand to fail. Increasing the metrics-ui app memory allocation to 128MB resolves the issue and allows the errand to run as expected.

This can be changed via the Apps Manager or CLI after the initial failure by running the following command:

```
cf scale metrics-ui -m 128M
```

Please note that anytime the `push-apps` errand is re-run, the metrics-ui allocation reverts back to 64MB.

Node.js Buildpack Incompatibility

The latest versions of the Node.js buildpack are not compatible with PCF Metrics 1.4.4 and earlier. Please update to PCF Metrics 1.4.5 or later to address this issue.

Compatibility with Elastic Runtime

PCF Metrics v1.4.x requires Pivotal Application Service (formerly Elastic Runtime) v1.11.0 or later.

Using Custom App Metrics

To use custom app metrics in PCF Metrics, you must install the Metrics Forwarder tile and bind your apps to the Metrics Forwarder service. You can use PCF Metrics without Metrics Forwarder, but you will be able to view only default container metrics, network metrics, and app events.

Delay in Ingesting Metrics

To prevent metrics data backup in Redis, ensure that each app does not emit more than 400 metrics per minute. For deployments with a high number of apps, you may still experience a delay of up to 1-2 hours before metrics data appears on the dashboard.

Data Loss During Stemcell Upgrades

No metrics or logs data will be ingested during a stemcell or VM update. In tests on tile environments, stemcell updates resulted in approximately 5 minutes where logs data was not ingested.

Logs Data Export Limitation

Due to security, call pagination, and resource constraints, downloaded logs are limited to 10,000 log-line entries. If you need to export more logs data, we suggest looking directly at the Loggregator Firehose or utilizing the Log Cache CLI.

Smoke Test Failures Due to Elasticsearch

The PCF Metrics **Smoke Test** errand may fail if you have significant amount of log data stored in Elasticsearch and you perform one of the following actions:

- Upgrade from v1.3.x to v1.4.x
- Scale the number of Elasticsearch Master or Data instances

This happens because Elasticsearch may require up to 24 hours to re-allocate shards. During this time, you will not see any logs in the UI. However, no logs are lost. If you experience this issue, wait until your Elasticsearch cluster is healthy/green. Smoke tests should pass once you re-run the PCF Metrics install.

Smoke Test Failures Due to User Authentication

The PCF Metrics **Smoke Test** errand may fail if your deployment authenticates user sign-ons with an external SAML identity provider or an external LDAP

server. In some cases, these external user stores have an additional login procedure that prevents the errand from authenticating with the deployment and validating against the Metrics API.

If you experience this issue, disable the **Smoke Test** errand in the PCF Metrics tile and click **Apply Changes** to run the install again.

For more information about what configurations lead to this failure, see Configure Authentication and Enterprise SSO in *Configuring Elastic Runtime*.

For Operators Who Deploy PCF Metrics Using BOSH

If both of the following are true, you might experience issues while using PCF Metrics:

- You deploy PCF Metrics using BOSH instead of using the PCF Metrics tile in Ops Manager.
- You use self-signed certificates.

Pivotal recommends using certificates issued by a Certificate Authority for BOSH deployments of this product.

Past Minor v1.3.x

Release Notes for v1.3.x releases can be found here.

Past Minor v1.2.x

Release Notes for v1.2.x releases can be found here.

Past Minor v1.1.x

Release Notes for v1.1.x releases can be found here.

Past Minor v1.0.x

Release Notes for v1.0.x releases can be found here.