Pivotal

PRODUCT DOCUMENTATION

Service Metrics for Pivotal Cloud Foundry® Documentation

Version 1.4.3

Published: 12 March 2019

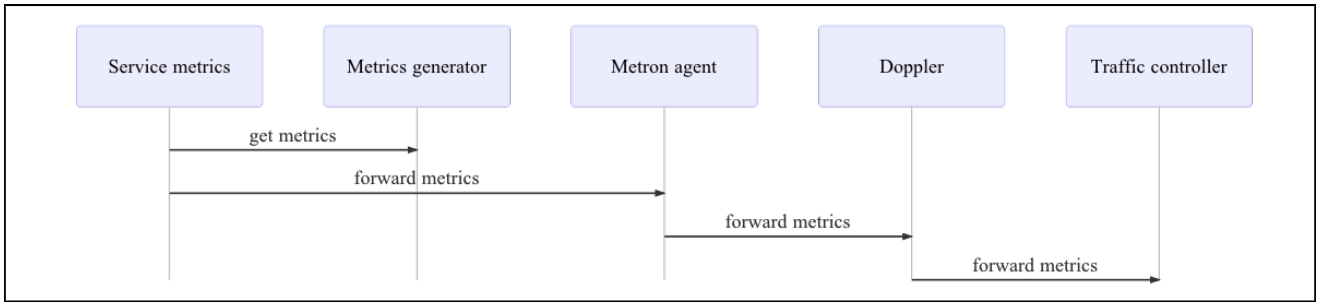
Table of Contents

Service Metrics for Pivotal Cloud Foundry

Overview

The service-metrics SDK allows service authors to integrate with Loggregator, the Cloud Foundry logging and metrics system, using a co-locatable BOSH release.

How Do Service Metrics Work?



Service metrics are co-located with the service job. The service job runs as a daemon process and executes a metrics generation command at regular intervals. The service author generally provides a binary for that purpose. In this case the metrics generation command invokes the binary.

Service metrics receives the output of the metrics generation command and performs validation on the format.

If validation succeeds, Service metrics sends the metrics to the Metron agent deployed on the VM. Metron forwards metrics to Doppler. In turn, Doppler can send the metrics to a third-party app such as Papertrail or to the Loggregator Traffic Controller.

Service Metrics Release Notes

New Features

- None

Breaking Changes

- None

Bug fixes

- Metrics command args no longer logged in error cases

Integrating a Service with Service Metrics

What is Required of Service Authors?

The service author provides a metrics collection executable, packaged as a BOSH release. The metrics collection executable collects relevant metrics from a service, and the Service-metrics job collects and forwards them to the Metron agent.

Service authors must design the executable for frequent calls. The metrics job calls the executable every minute by default, but the operator can change the interval.

The executable can accept command line arguments to produce metrics. There is no pre-defined interface for the arguments, and service authors must decide upon the design for their service. The format of the arguments should be documented for the service author so that they can configure their deployment manifest.

Create a Service Metrics Release

The executable you write must be callable from the command line and optionally accept arguments. See the following example:

```
./bin/generate-metrics --all --config /var/vcap/jobs/some-service/conf.yml
```

Output

The return type must be a JSON array. Every element in this array is a metric, and each metric sample must provide the following keys: `key`, `value`, `unit`.

```
[
  {
    "key": "average_delay",
    "value": 0.02,
    "unit": "ms"
  },
]
```

Pivotal recommends that you follow the Metrics 2.0 spec for unit names, and use SI units and prefixes where appropriate.

Deploying a Release with Service Metrics

Upload Required Releases

Upload the following releases to your BOSH director:

- Your service release
- The service-metrics release
- Your service-name-metrics release

Write a BOSH Manifest

The service manifest must have a non-errand instance group that co-locates the following jobs:

- The `<YOUR-SERVICE-NAME>` job from the service release
- The `service-metrics` job from the service metrics release
- The `<YOUR-SERVICE-NAME>-metrics` job from -metrics release
- The `metron_agent` job from the Loggregator release

```
instance_groups:
- name: service-metrics
  instances: 1
  jobs:
  - name: <YOUR-SERVICE-NAME>
    release: service-release
  - name: service_metrics
    release: service-metrics
  - name: <YOUR-SERVICE-NAME>-metrics
    release: <YOUR-SERVICE-NAME>-metrics
  - name: metron_agent
    release: loggregator
  stemcell: trusty
  vm_type: medium
  networks:
  - name: service-metrics
  azs: [eu-west-1c]
```

Properties

The service metrics job requires the following configuration properties to be present:

Field	Type	Description	Required	Default Value
`origin`	string	the name of the service, so it can reference metrics originating from that service in the logs	yes	“”
`metrics_command`	string	the command to generate the metrics	yes	“”
`metrics_command_args`	array of strings	any args provided to `metrics_command`	no	[]
`execution_interval_seconds`	int	how often the metric generation command runs	no	60
`debug`	boolean	turn verbose mode on/off	no	false
`monit_dependencies`	array of strings	an array of jobs that must run before monit attempts to start the service metrics job. This is a way to define job dependencies, which are not supported by BOSH.	no	[]

If the `metrics_command` fails, for example if the `[MY-SERVICE]-metrics` binary exits with a non-zero exit code, the service-metrics job will not start, or will exit with 0 if it was already running. In this case, the BOSH instance shows as `failing` and `monit` will try to restart the service-metrics job.

An example snippet is shown below:

```
properties:
  service_metrics: #the service metrics release template expects the property key to be service_metrics, even though the job is called service-metrics
    origin: *name
    execution_interval_seconds: 5
    metrics_command: /bin/echo
    metrics_command_args:
    - "-n"
    - '[{"key":"service-dummy","value":99,"unit":"metric"}]'
    monit_dependencies: []
    nats:
      machines:
      - 10.0.1.109
      port: 4222
      user: nats
      password: <REPLACE-WITH-PASSWORD>
    etcd:
      machines:
      - 10.0.1.110
  metron_agent:
    zone: z1
    deployment: *name
    dropsonde_incoming_port: 3457
  metron_endpoint:
    shared_secret: <REPLACE-WITH-SECRET>
  loggregator:
    etcd:
      machines:
        - 10.0.1.110
  loggregator_endpoint:
    shared_secret: <REPLACE-WITH-SECRET>
```

The service metrics release does not currently support the BOSH v2 manifest format, which allows job level properties.

Logging

Service metrics logs to files in `/var/vcap/sys/log/service-metrics`, and also to syslog.

For forwarding syslog to a third party syslog drain (e.g. papertrail) we recommend co-locating the syslog-release.