Service logs & metrics

# Goals

Create a **common concept** how to provide logs & metrics of CF service instances to the developer:



Couple of suggestions

1. We should look at a log buffering mechanism like Kafka in between services and consumer of the logs. This also gives an advantage to have concurrent consumers draining the log with a very fast write throughput on producer side
2. Log drain on Services can follow a logging Driver kind of design which allows to hook on to different mechanisms of log drains and sinks. As an example in this case the logging driver can hook on to rsyslog and push the logs to Kafka broker on a topic defined per service. The Drivers can be deployed via the BOSH add on mechanism

# Non-goals

* For now, focus on dedicated service instances only (shared services might need additional efforts to split up logs for each service instance)
* How to further handle the emitted logs/metrics (i.e. Logging as a Service) is not part of this concept (but this concept should allow developing Logging as a Service on top)

# 

# 

# User stories

### #1: As a CF developer, I want to get logs of my service instances sent to a specific endpoint for further processing/visualization

Idea: cf create-service my-mongo-db-container -c '{ logs-endpoint: "syslog://xyz" }'  
Similar experience to when binding an app to a service with a syslog drain.

- NFR: Use well-defined standards, i.e. syslog? Can we ship it to a Logstash running on CF?

This is where the logging driver suggested above can help. To putting load on service to send logs to target destinations, we can decouple it via a Kafka setup to have a consumer which can send logs to say Logstash and in parallel also allow draining via a cli extension.

### #2: As a CF developer, I want to get metrics of my service instances sent to a specific endpoint for further processing/visualization Idea: cf create-service my-mongo-db-container -c '{ metrics-endpoint: "syslog://xyz" }' Similar experience to when binding an app to a service with a syslog drain.

- NFR: Use well-defined standards, i.e. statsd, influx…?

### #3: As a CF developer, I want to get logs (metrics?) of my service instances using CF CLI similar to cf logs

Idea: cf service-logs my-mongo-db-container

(Optional)

Less important since cf logs is anyway mainly used during dev-time, in production most use splunk, elk etc.

### #4: As a CF developer, I want to send my service logs/metrics to a 3rd party logging service in the marketplace in a UX-friendly way

(Optional)

If I have a logging service in the marketplace (i.e. ELK), connecting this ELK service instance to consume my service instance logs should work smoothly. Same for metrics.

### #5: As a CF developer, I want > 99% of my logs/metrics to arrive at the endpoint

Reliability

### #6: As a CF service operator, I want to filter out sensitive logs of my service software before shipping to the developer

We can not ship 100% of some service's logs to the dev but need to filter it before.

### #7: As a CF service operator, I want to ship logs that my service software stores on the filesystem

The concept should support sending logs that are written to the FS (i.e. like Filebeat does)

### #8: As a CF service operator, I want to ship logs that my service software sends to syslog

(Optional) Some service software might only send to syslog, so we probably should support this too.

### #9: As a CF service operator, I want to use a custom script to gather metrics to ship

For metrics, we need a way to gather the metrics from the service software. This is entirely different for most software (some might expose metrics using the API, others require CLI commands). If we provide service operators a way to bring their own scripts to gather metrics (similar to [telegraf's exec plugin](https://github.com/influxdata/telegraf/tree/master/plugins/inputs/exec)), we would cover most of these.

### #10: As a CF platform operator, I can not afford to store logs permanently on disk

If we have some component in transit that writes the logs to the disk, be aware that this can explode size-wise.

If we store in Kafka we can configure a retention period for logs.

### #11: As a CF platform operator, I want to reuse the components of this concept for use with shared services

We should keep in mind that we should extend this concept also to use with shared services (i.e. MariaDB Galera clusters).

### #12: As a CF platform operator, I want to reuse the components of this concept to also ship logs/metrics to my platform log/metrics endpoint

As a platform operator, I'm also interested in the service logs & metrics. Ideally, the components used to ship stuff to the end user could also ship it to the platform operators endpoint.

### #13: As a Kubernetes developer, I want to consume my service logs/metrics, too

Concept should work also when used outside of CF.

# Standard REST APIs could be used to front end the consumer of Kafka topic. CF Cli extension can be one type of consumer and Kubernetes can be another or any other for that matter.

# 

# Implementation ideas

**Reuse loggregator**

Would be nice, since loggregator pipeline already solves push (syslog drain) and pull (firehose, cf cli) and offers infrastructure. Only extending trafficcontroller and metron agent to support service instance logs is needed.

However, Adam (PM of loggregator) thinks additional load of service instance logs would possibly kill loggregator (which is struggling with reliability already); so they're not working in this direction.

**Filebeat & Telegraf**

Colocate Filebeat (Logs) and Telegraf (Metrics) on service instance and configure them to ship to user-provided endpoints.

Issues: Filtering logs is difficult (#6 and #11)

**Custom agent**

Colocate custom agent on service instance and configure them to ship to user-provided endpoints (adding exact functionality that we need, i.e. filtering)

Issues: Reinvent the wheel when it comes to shipping logs etc.

High level Diagram on how the setup can work with Log Shipping via Kafka.

Aspects around Security etc are not covered here and can be detailed later

