New Relic.

Flink Positive

Caito Scherr, Nik Davis

Safe Harbor

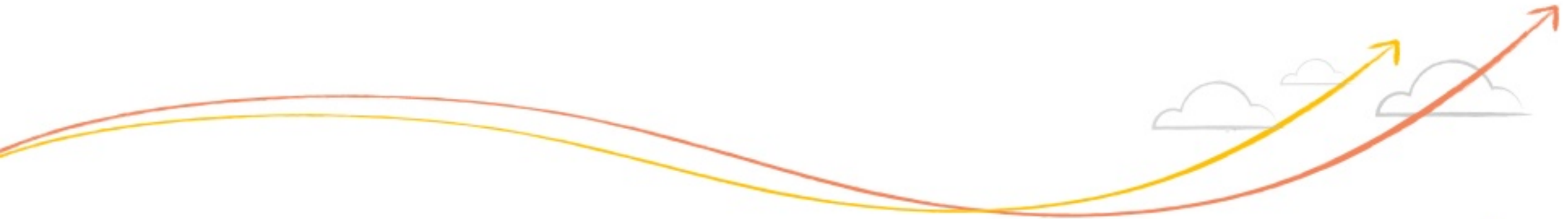
This presentation and the information herein (including any information that may be incorporated by reference) is provided for informational purposes only and should not be construed as an offer, commitment, promise or obligation on behalf of New Relic, Inc. ("New Relic") to sell securities or deliver any product, material, code, functionality, or other feature. Any information provided hereby is proprietary to New Relic and may not be replicated or disclosed without New Relic's express written permission.

Such information may contain forward-looking statements within the meaning of federal securities laws. Any statement that is not a historical fact or refers to expectations, projections, future plans, objectives, estimates, goals, or other characterizations of future events is a forward-looking statement. These forward-looking statements can often be identified as such because the context of the statement will include words such as "believes," "anticipates," "expects" or words of similar import.

Actual results may differ materially from those expressed in these forward-looking statements, which speak only as of the date hereof, and are subject to change at any time without notice. Existing and prospective investors, customers and other third parties transacting business with New Relic are cautioned not to place undue reliance on this forward-looking information. The achievement or success of the matters covered by such forward-looking statements are based on New Relic's current assumptions, expectations, and beliefs and are subject to substantial risks, uncertainties, assumptions, and changes in circumstances that may cause the actual results, performance, or achievements to differ materially from those expressed or implied in any forward-looking statement. Further information on factors that could affect such forward-looking statements is included in the filings New Relic makes with the SEC from time to time. Copies of these documents may be obtained by visiting New Relic's Investor Relations website at ir.newrelic.com or the SEC's website at www.sec.gov.

New Relic assumes no obligation and does not intend to update these forward-looking statements, except as required by law. New Relic makes no warranties, expressed or implied, in this presentation or otherwise, with respect to the information provided.

Who Are We?



Who Are We?

Caito Scherr

Nik Davis

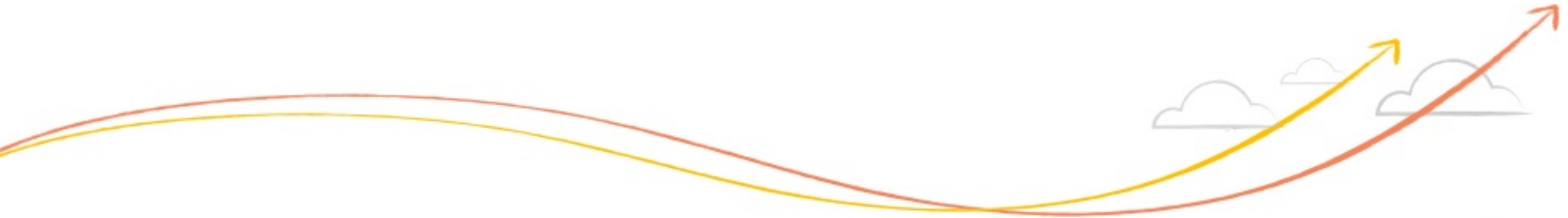
New Relic.

Portland Office

New Relic

Who Are We?

Why Are We Here?



Why Are We Here?

Thesis

You can be successful in your Flink project using data-driven development

Thesis

You can be successful in your Flink project using data-driven development

1. Proving viability early with a Proof of Concept

Thesis

You can be successful in your Flink project using data-driven development

1. Proving viability early with a Proof of Concept
2. Letting data from our Flink monitoring drive our development

Agenda

- **The Challenge**
- Proof of Concept
- Flink Monitoring
- 6000 Meter View

Agenda

The Challenge

Proof of Concept

Flink Monitoring

6000 Meter View

Agenda

The Challenge

Proof of Concept

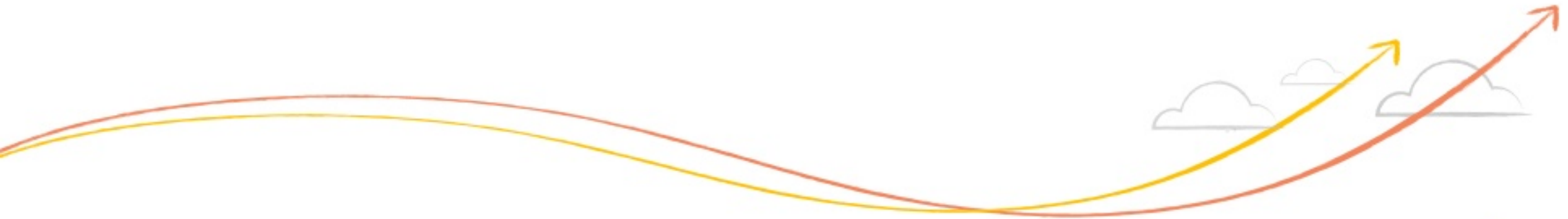
Flink-Monitoring

6000 Meter View

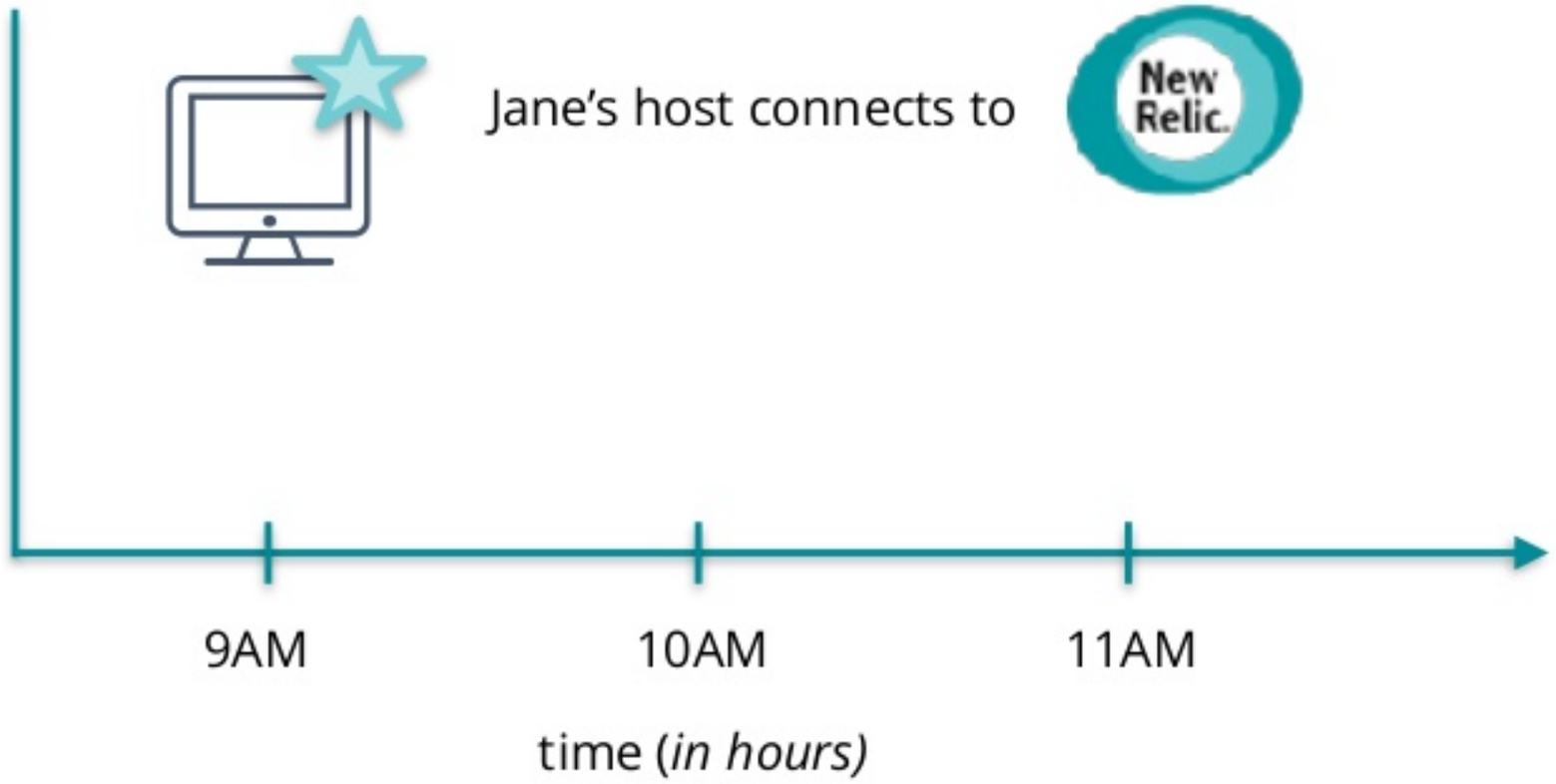
Agenda

- The Challenge
- Proof of Concept
- Flink Monitoring
- **6000 Meter View**

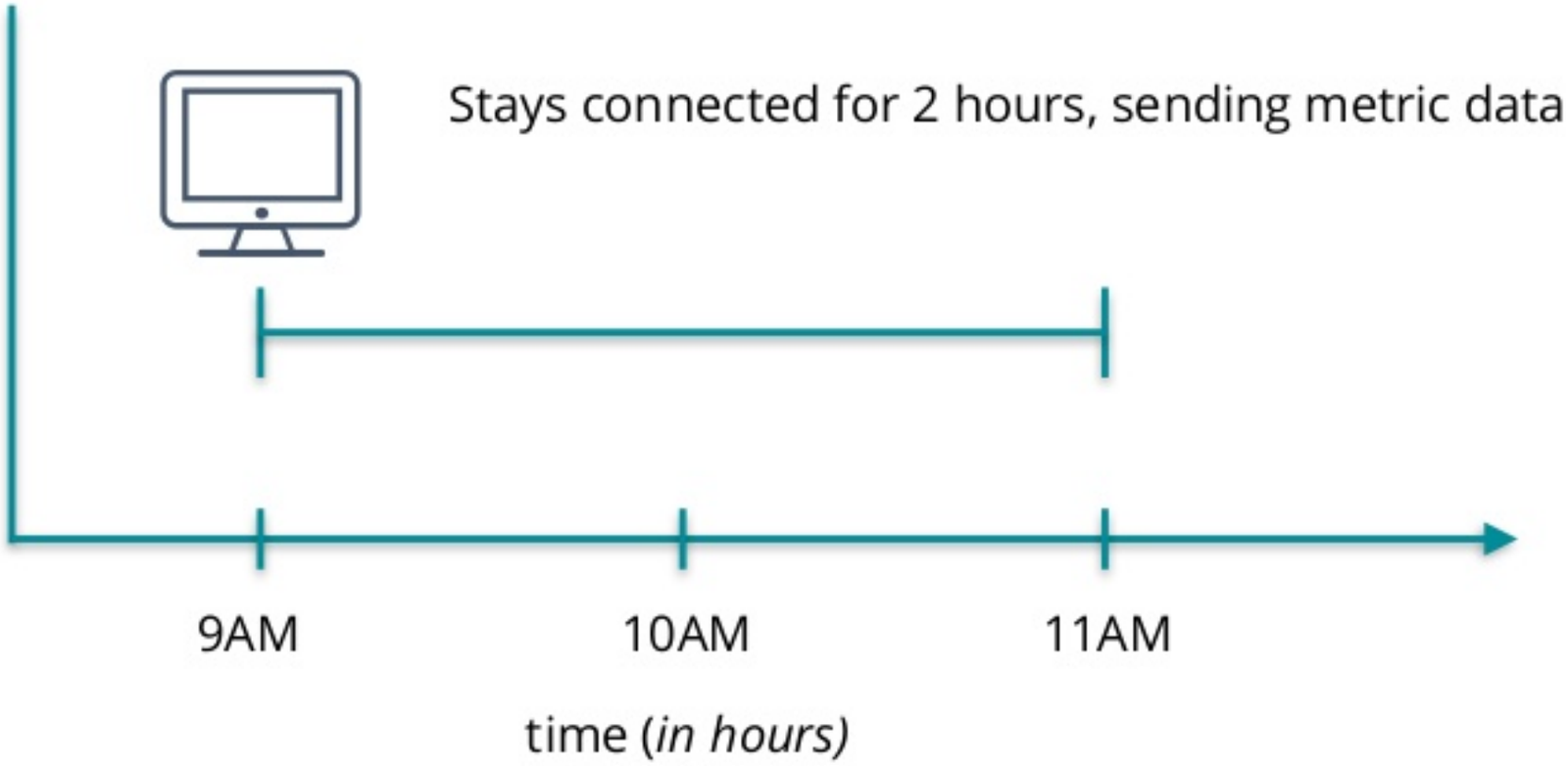
The Challenge



Usage



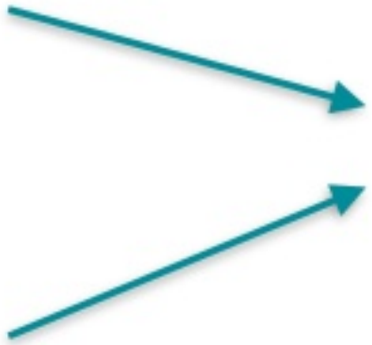
Usage



Usage

Agent records

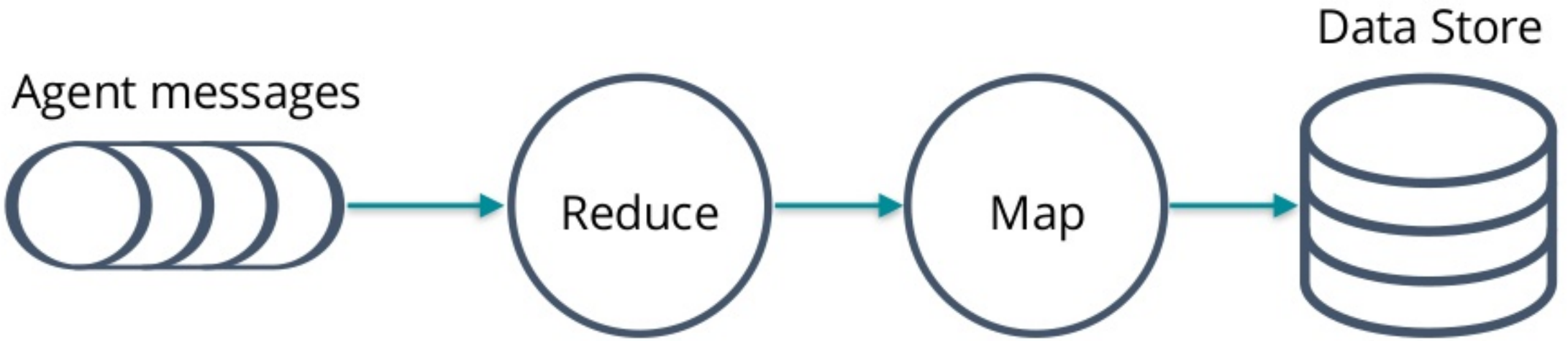
```
{
  "hostname": "Jane's host",
  "timestamp": "2018-09-05T10:00:00+02:00"
}
...
{
  "hostname": "Jane's host",
  "timestamp": "2018-09-05T11:00:00+02:00"
}
```



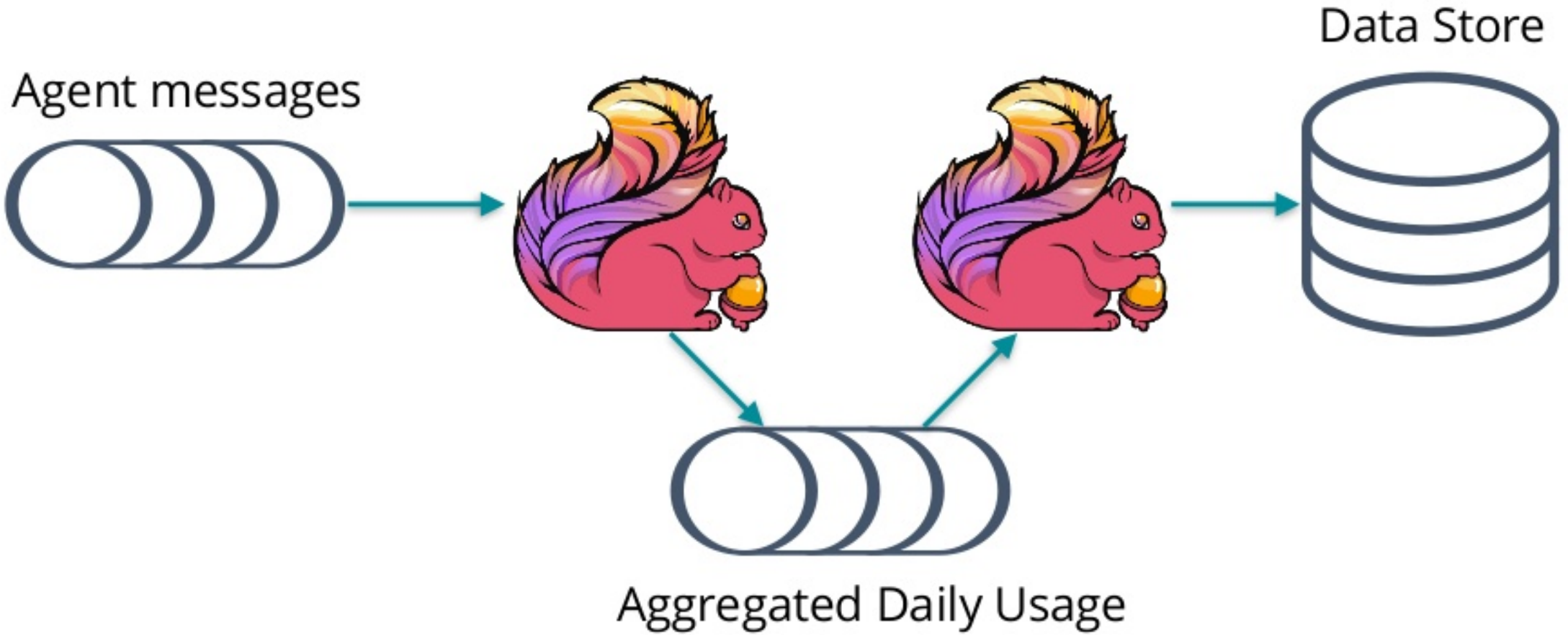
Summary *daily* record

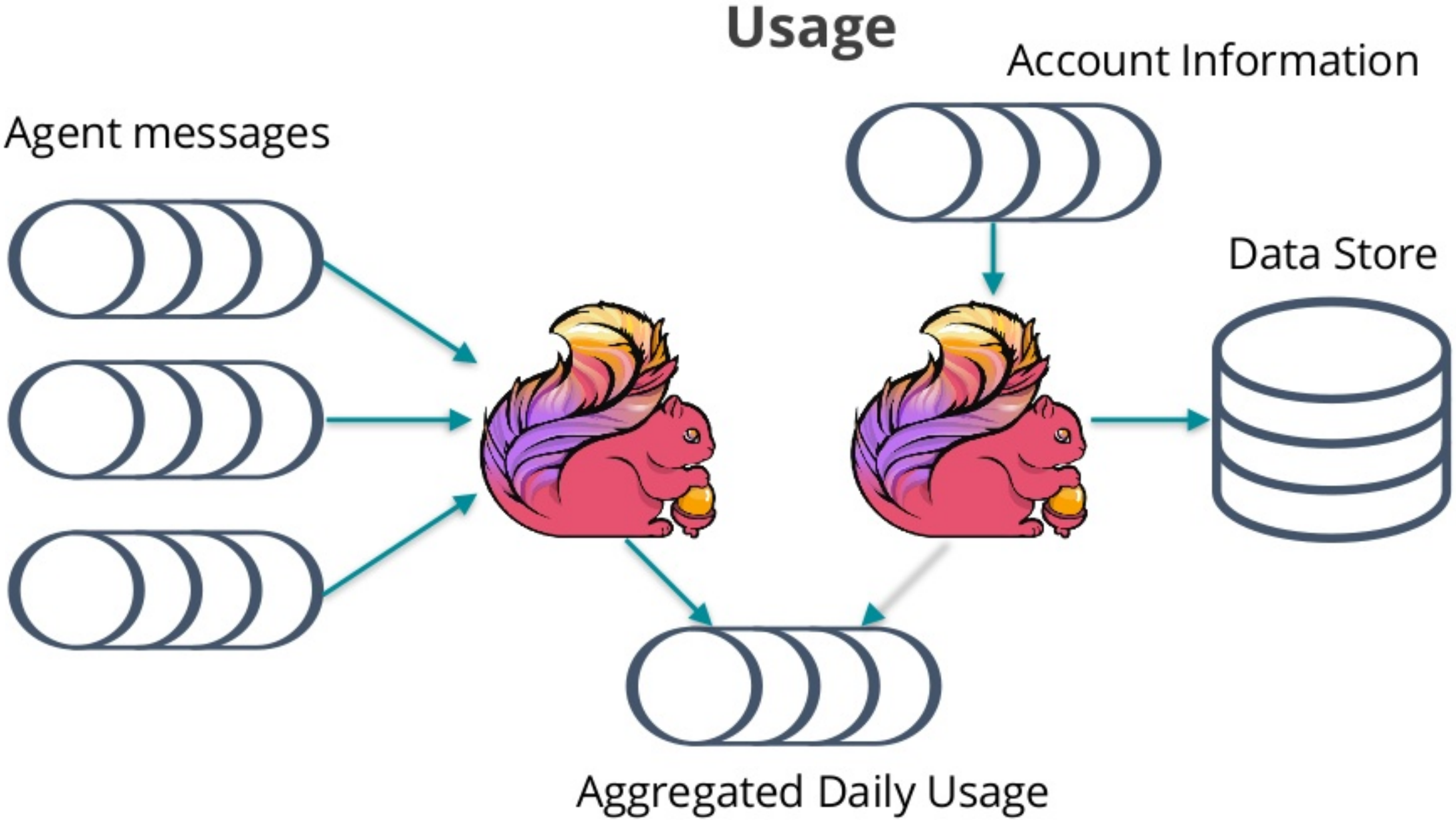
```
{
  "hostname": "Jane's host",
  "daysSinceEpoch": 17779,
  "hoursUsed": 2
}
```

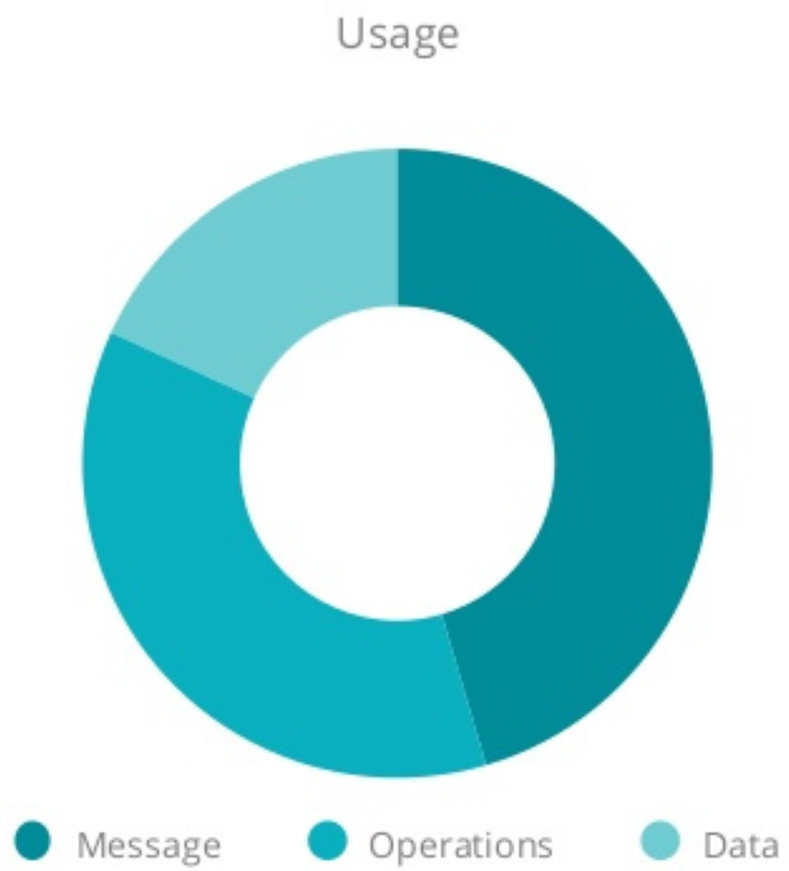
Usage



Usage







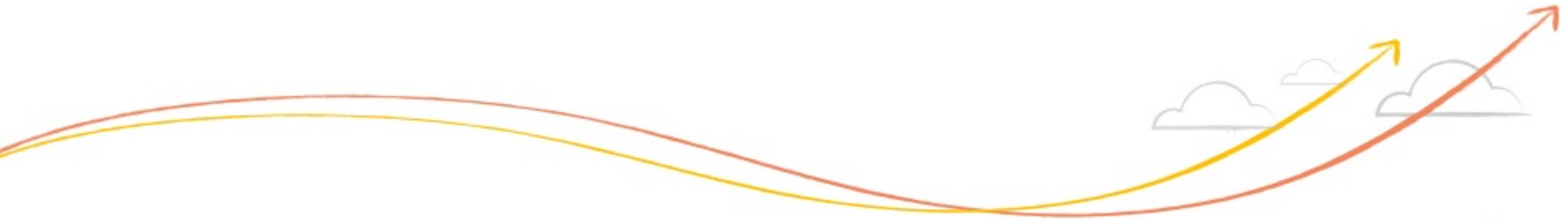
Ingests **5M** messages

Runs **15M** operations

Processes **3GB** data

EVERY MINUTE

Proof of Concept



Proof of Concept: What

- Tiny working model

Proof of Concept: What

- Tiny working model
- Data-driven development

Proof of Concept: What

- Tiny working model
- Data-driven development
- Technically viable

Proof of Concept: What

- Tiny working model
- Data-driven development
- Technically viable
    - Steel thread: start to finish testing

Proof of Concept: What

- Tiny working model
- Data-driven development
- Technically viable
  - Steel thread: start to finish testing
  - Minimal, miniaturize

Proof of Concept: Why

- Prove viability early

Proof of Concept: Why

- Prove viability early
- Goal: miniaturize components to build quickly, keep scale and technical requirements

Proof of Concept: Why

- Prove viability early
- Goal: miniaturize components to build quickly, keep scale and technical requirements
- **Make it quick, make it work**

Proof of Concept: Pre-Work

- Rubric

Proof of Concept: Pre-Work

- Rubric
  - Time windowing

Proof of Concept: Pre-Work

- Rubric
  - Time windowing
  - Parallelizing

Proof of Concept: Pre-Work

- Rubric
  - Time windowing
  - Parallelizing
  - Development velocity

Proof of Concept: Pre-Work

- Rubric
  - Time windowing
  - Parallelizing
  - Development velocity
    - Maintainability

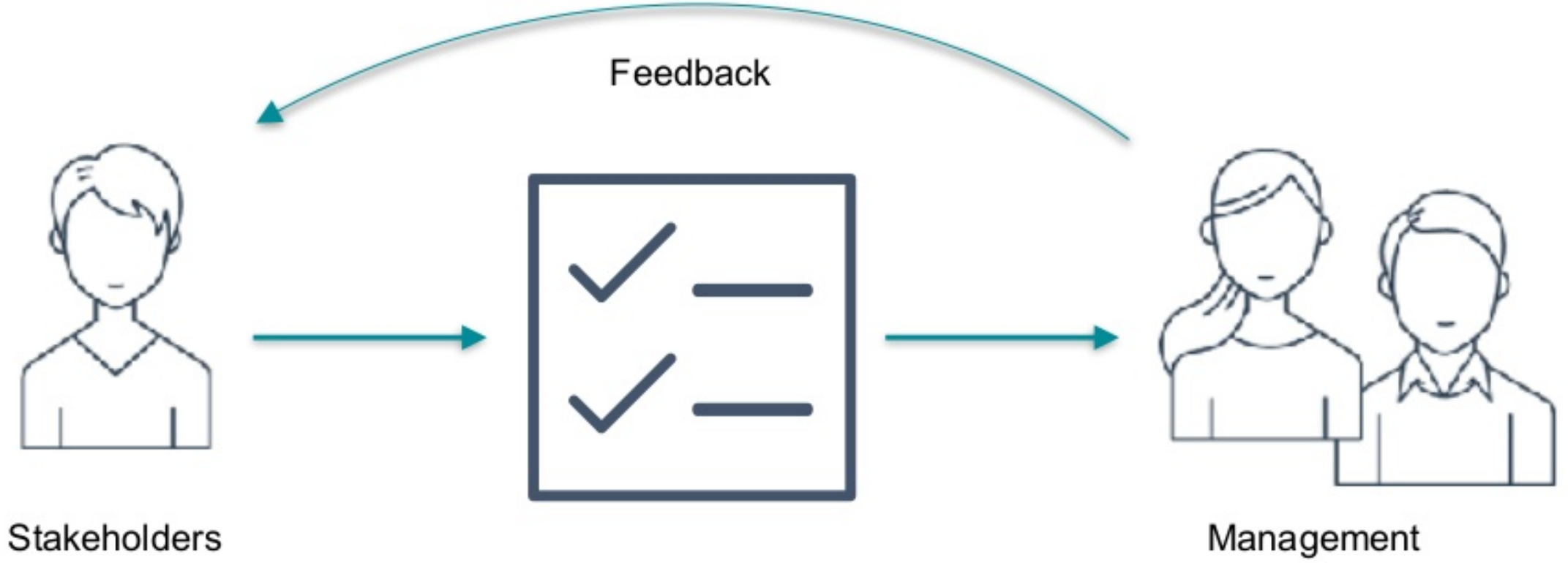
Proof of Concept: Pre-Work

- Rubric
  - Time windowing
  - Parallelizing
  - Development velocity
    - Maintainability
  - Failure modes

Proof of Concept: Pre-Work

- Rubric
  - Time windowing
  - Parallelizing
  - Development velocity
    - Maintainability
  - Failure modes
  - Testing

Proof of Concept: Pre-Work



Proof of Concept: How

Prove a steel thread

```
DataStream<DailyHostSummary> dailyHostStream = agentDataStream
                .keyBy("accountId", "hostId")
                .window(TumblingEventTimeWindows.of(Time.minutes(3))) // change to 24 hours (1440 minutes)
                .fold(new DailyHostSummary(), new DailyHostSummarizer());
```

Proof of Concept: How

```
agentHeartbeats
    .assignTimestampsAndWatermarks(
    new BoundedOutOfOrdernessTimestampExtractor<AgentHeartbeat>(
        Time.seconds(env.getMaxAllowedLatenessApm())
      ) {
          @Override
          public long extractTimestamp(AgentHeartbeat element) {
              return element.getHeartbeatTime() * 1000;
          }
    })
    .name(...).uid(...)
    .map(value -> new HostState(value))
    .keyBy("consumingAccountId", "hostId")
    .window(TumblingEventTimeWindows.of(Time.minutes(env.getAggregationWindowDurationInMinutes())))
    .reduce(new ReduceHostUsageFunction())
    .name(...).uid(...)
    .map(new MapHostStateToEventFunction(env.isProduceApmProductionEvent()))
    .name(...).uid(...)
    .map(value -> (ApmDailyBillingRecord) value)
    .map(new MapOverrideInstanceSizeFunction()).name(...).uid(...)
    .map(new MapCloudInstanceSizeFunction()).name(...).uid(...)
    .map(new MapAgentCalculatedInstanceSizeFunction()).name(...).uid(...)
    .map(new MapDefaultInstanceSizeFunction()).name(...).uid(...)
    .map(new MapInstanceSizeCapFunction()).name(...).uid(...)
    .map(value -> (NrDailyHost) value);
```

Proof of Concept: How

```
agentHeartbeats
    .assignTimestampsAndWatermarks(
    new BoundedOutOfOrdernessTimestampExtractor<AgentHeartbeat>(
        Time.seconds(env.getMaxAllowedLatenessApm())
      ) {
          @Override
          public long extractTimestamp(AgentHeartbeat element) {
              return element.getHeartbeatTime() * 1000;
          }
    })
    .name(...).uid(...)
    .map(value -> new HostState(value))
    .keyBy("consumingAccountId", "hostId")
    .window(TumblingEventTimeWindows.of(Time.minutes(env.getAggregationWindowDurationInMinutes())))
    .reduce(new ReduceHostUsageFunction())
    .name(...).uid(...)
    .map(new MapHostStateToEventFunction(env.isProduceApmProductionEvent()))
    .name(...).uid(...)
    .map(value -> (ApmDailyBillingRecord) value)
    .map(new MapOverrideInstanceSizeFunction()).name(...).uid(...)
    .map(new MapCloudInstanceSizeFunction()).name(...).uid(...)
    .map(new MapAgentCalculatedInstanceSizeFunction()).name(...).uid(...)
    .map(new MapDefaultInstanceSizeFunction()).name(...).uid(...)
    .map(new MapInstanceSizeCapFunction()).name(...).uid(...)
    .map(value -> (NrDailyHost) value);
```

Proof of Concept: How

Make it quick

```
DataStream<DailyHostSummary> dailyHostStream = agentDataStream
                .keyBy("accountId", "hostId")
                .window(TumblingEventTimeWindows.of(Time.minutes(3))) // change to 24 hours (1440 minutes)
                .fold(new DailyHostSummary(), new DailyHostSummarizer());
```

Proof of Concept: How

```
AGGREGATION_WINDOW_DURATION_IN_MINUTES =
    Integer.parseInt(getEnvWithDefault("AGGREGATION_WINDOW_DURATION_IN_MINUTES", "1440"));
```

Proof of Concept: How

```
agentHeartbeats
    .assignTimestampsAndWatermarks(
    new BoundedOutOfOrdernessTimestampExtractor<AgentHeartbeat>(
        Time.seconds(env.getMaxAllowedLatenessApm())
      ) {
          @Override
          public long extractTimestamp(AgentHeartbeat element) {
              return element.getHeartbeatTime() * 1000;
          }
    })
    .name(...).uid(...)
    .map(value -> new HostState(value))
    .keyBy("consumingAccountId", "hostId")
    .window(TumblingEventTimeWindows.of(Time.minutes(env.getAggregationWindowDurationInMinutes())))
    .reduce(new ReduceHostUsageFunction())
    .name(...).uid(...)
    .map(new MapHostStateToEventFunction(env.isProduceApmProductionEvent()))
    .name(...).uid(...)
    .map(value -> (ApmDailyBillingRecord) value)
    .map(new MapOverrideInstanceSizeFunction()).name(...).uid(...)
    .map(new MapCloudInstanceSizeFunction()).name(...).uid(...)
    .map(new MapAgentCalculatedInstanceSizeFunction()).name(...).uid(...)
    .map(new MapDefaultInstanceSizeFunction()).name(...).uid(...)
    .map(new MapInstanceSizeCapFunction()).name(...).uid(...)
    .map(value -> (NrDailyHost) value);
```

Proof of Concept: Overview

- Tiny, working model

Proof of Concept: Overview

- Tiny, working model
- Having done the PoC meant that we had:

Proof of Concept: Overview

- Tiny, working model

- Having done the PoC meant that we had:
  - Confidence in our choice and roadmap

Proof of Concept: Overview

- Tiny, working model

- Having done the PoC meant that we had:
  - Confidence in our choice and roadmap
  - Understanding —> Velocity

Proof of Concept: Overview

- Tiny, working model

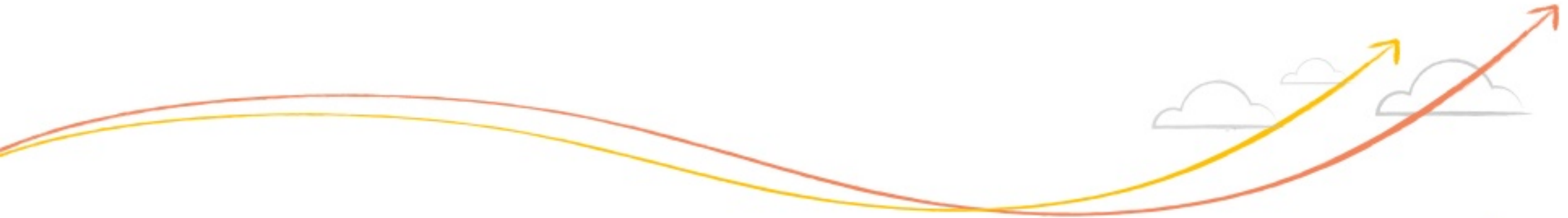
- Having done the PoC meant that we had:
  - Confidence in our choice and roadmap
  - Understanding —> Velocity
  - Groundwork

Proof of Concept: Overview

- Tiny, working model

- Having done the PoC meant that we had:
  - Confidence in our choice and roadmap
  - Understanding —> Velocity
  - Groundwork
  - Troubleshooting done early

Flink Monitoring



Flink Monitoring

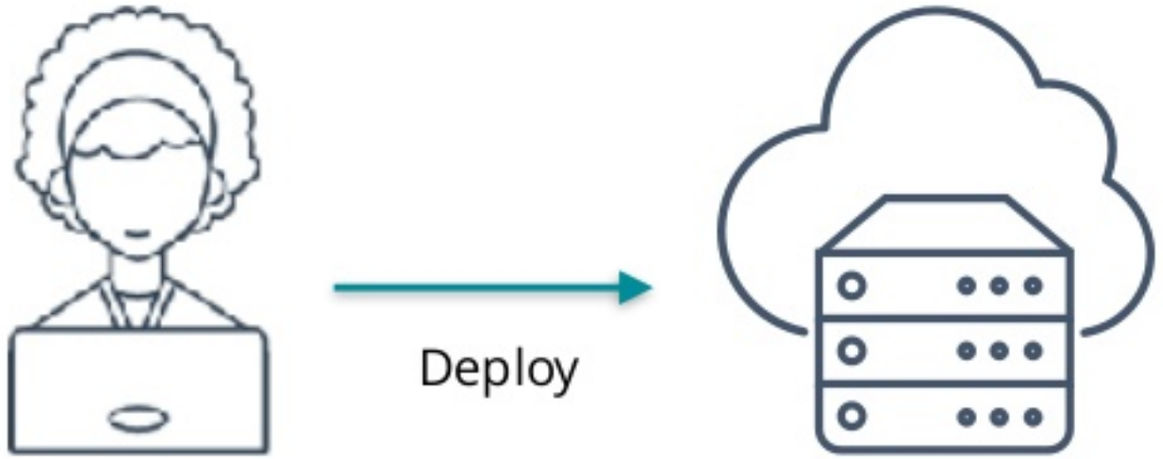
Data-driven development leads to real gains in both code quality and developer velocity

Flink Monitoring

It’s not complicated!

Flink Monitoring

Ship code

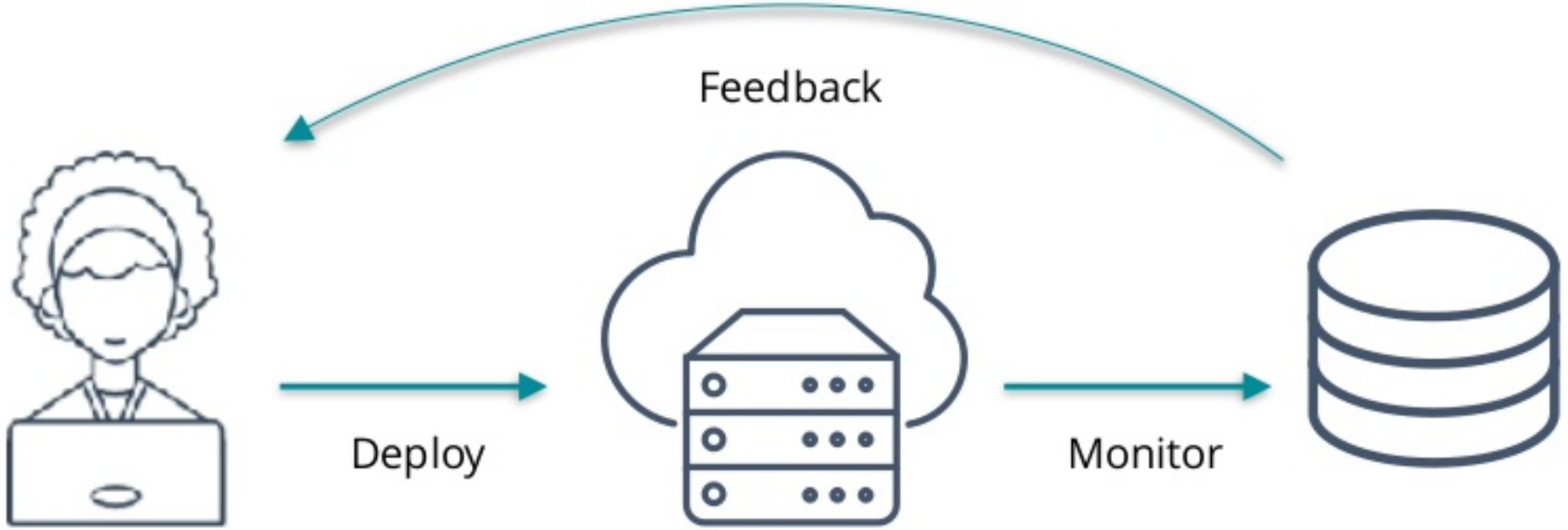


Flink Monitoring

Collect data

Flink Monitoring

Improve your product



Flink Monitoring

What do we track?

Flink Monitoring

What do we track? Examples

- State
  - size
  - backup duration

Flink Monitoring

What do we track? Examples

- State
  - size
  - backup duration
- Throughput

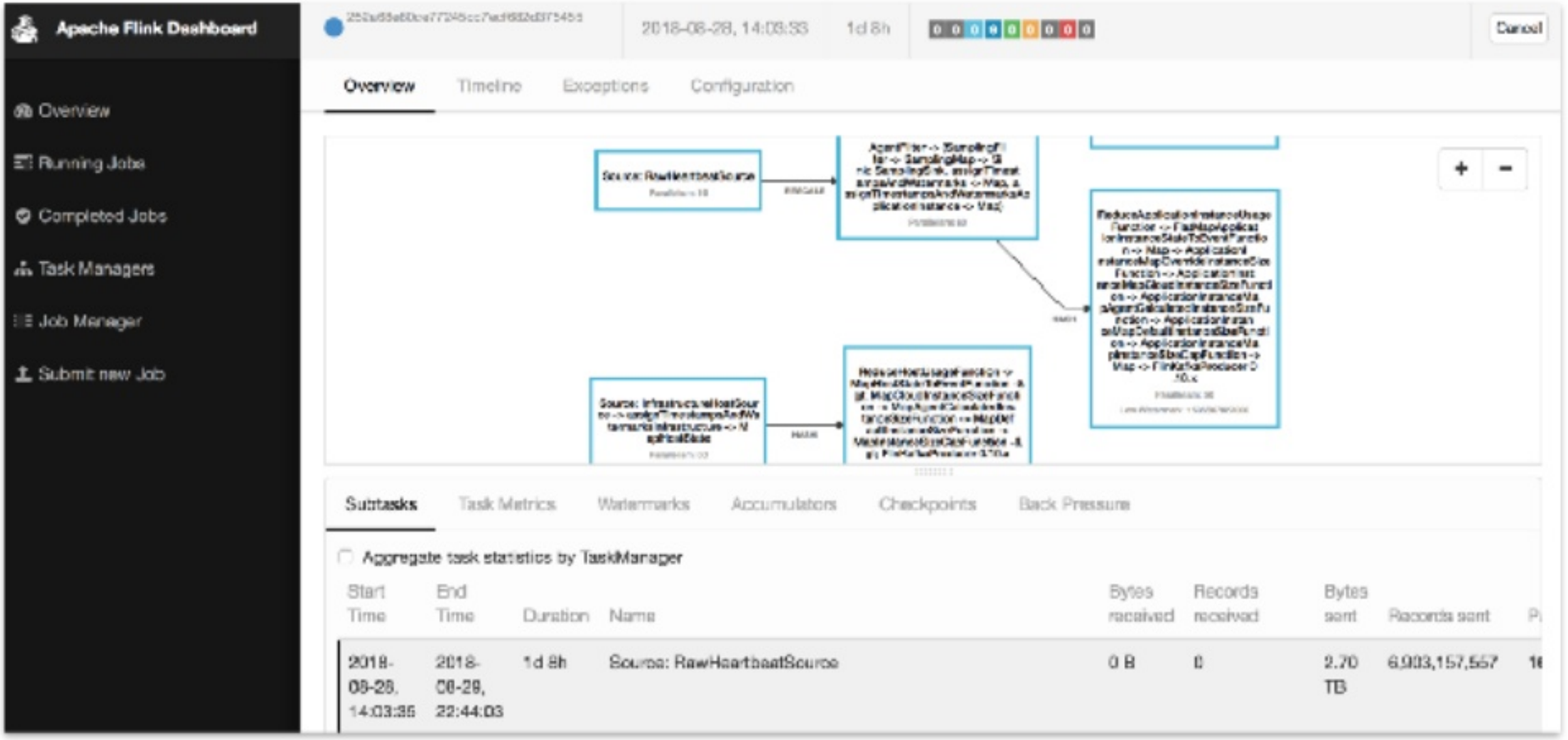
Flink Monitoring

What do we track? Examples

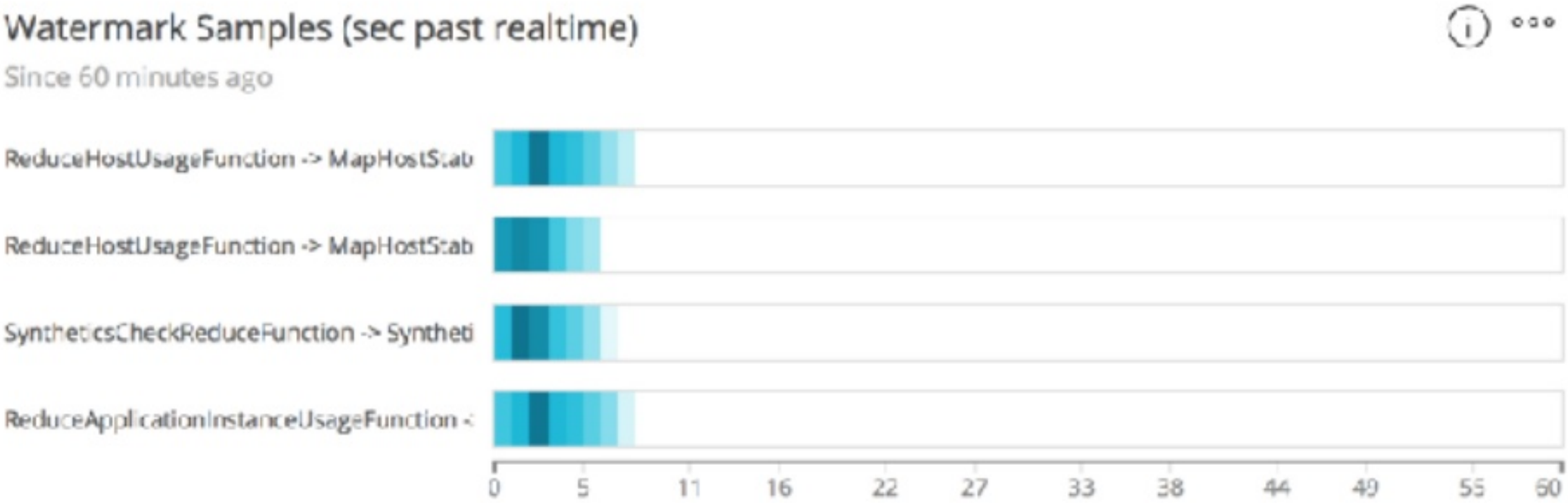
- State
  - size
  - backup duration
- Throughput
- Watermark

Flink Monitoring

Flink cluster UI is a start ...



Flink Monitoring



Flink Monitoring

How?

Flink Monitoring

Flink has a great metrics system

Flink Monitoring

Flink has a great metrics system; use it!

Flink Monitoring

You can ...

Use an existing framework

Flink Monitoring

You can ...

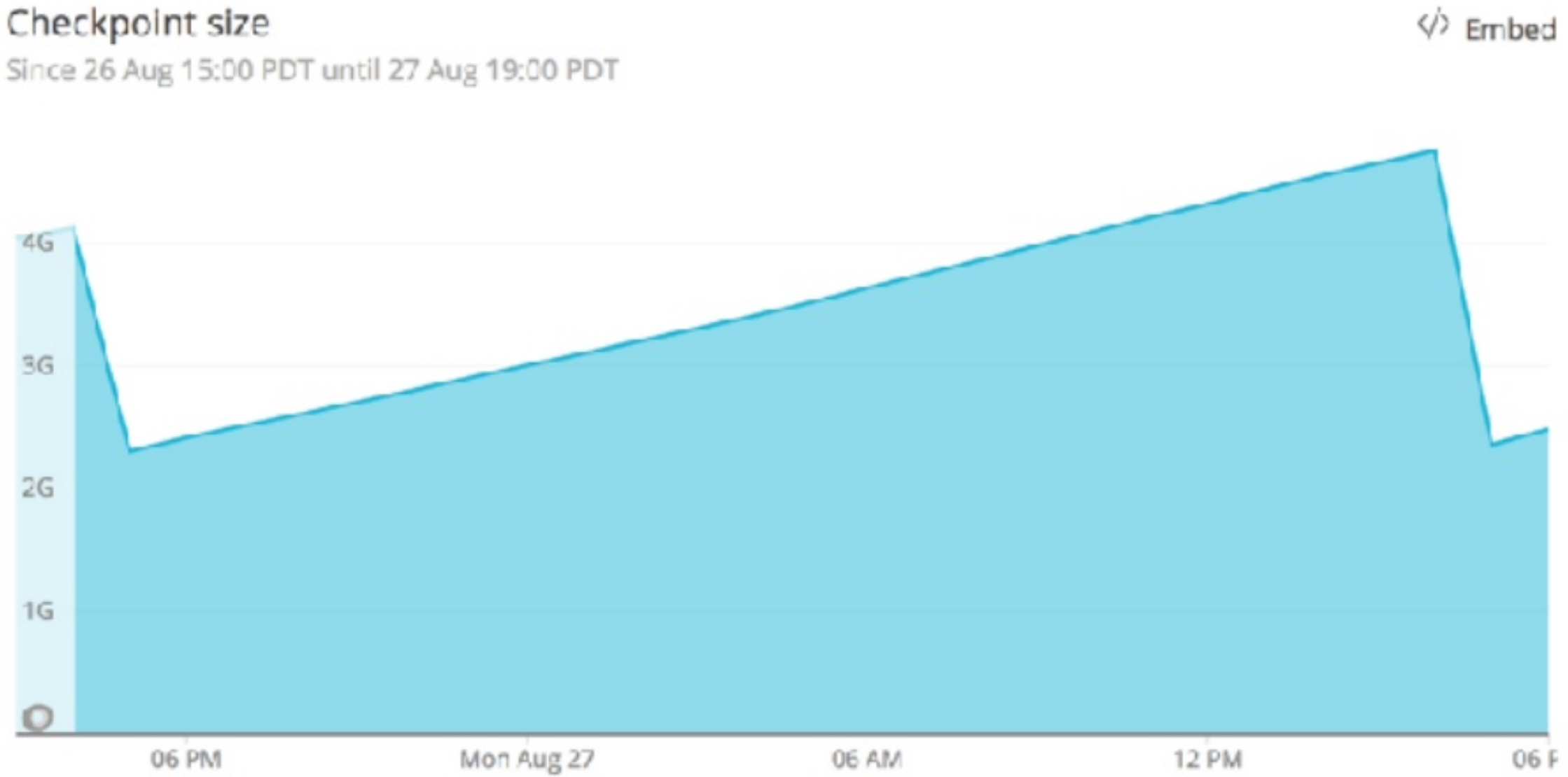
Use an existing framework

Write your own

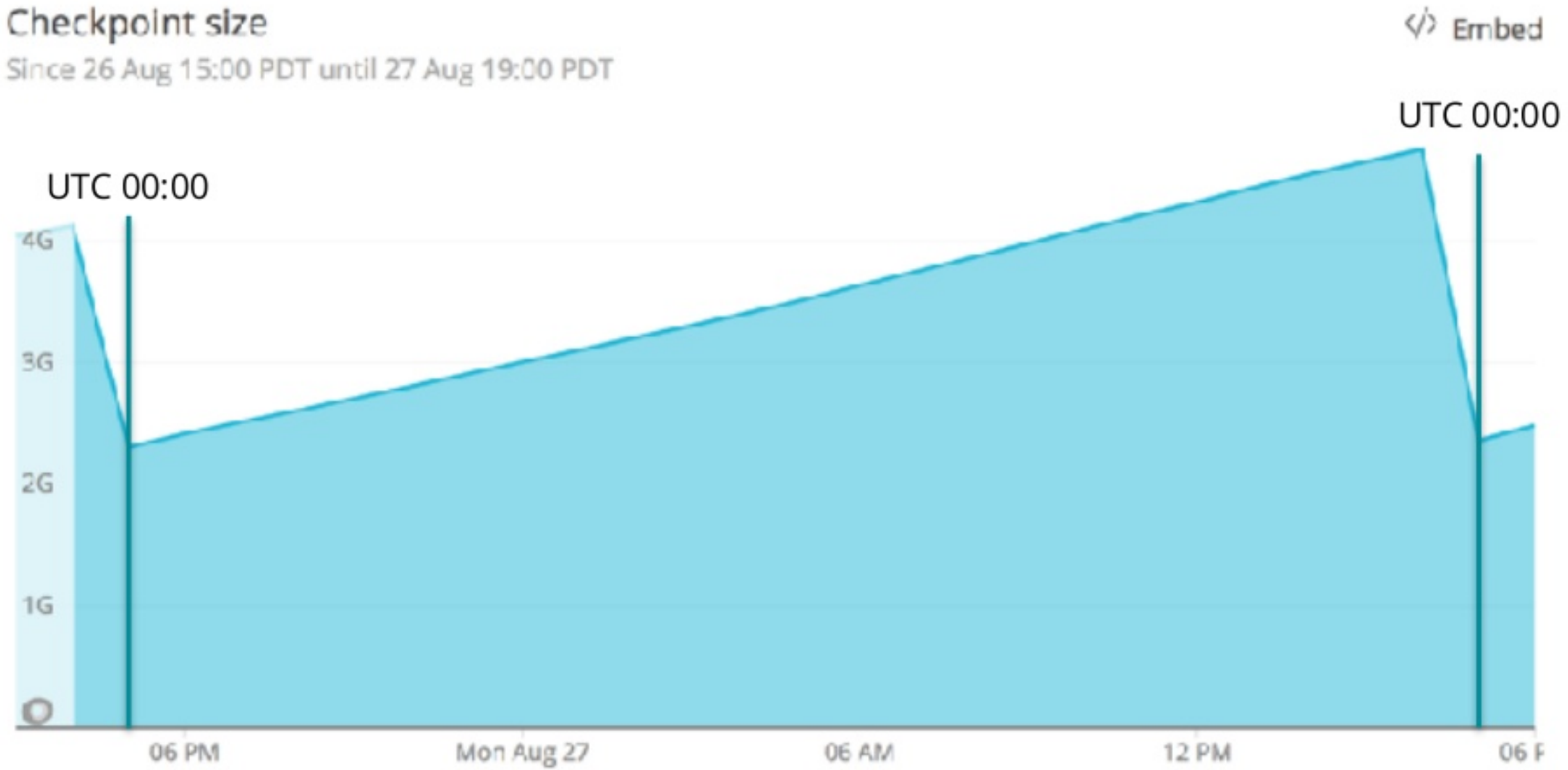
Flink Monitoring—Flink's interfaces

```
1   public interface MetricReporter {
2
3     void open(MetricConfig config);
4
5     void close();
6
7     void notifyOfAddedMetric(Metric metric, String metricName, MetricGroup group);
8
9     void notifyOfRemovedMetric(Metric metric, String metricName, MetricGroup group);
10
11  }
12
13  public interface Scheduled {
14
15    void report();
16
17  }
```

Flink Monitoring



Flink Monitoring



Flink Monitoring

We have data, now what?

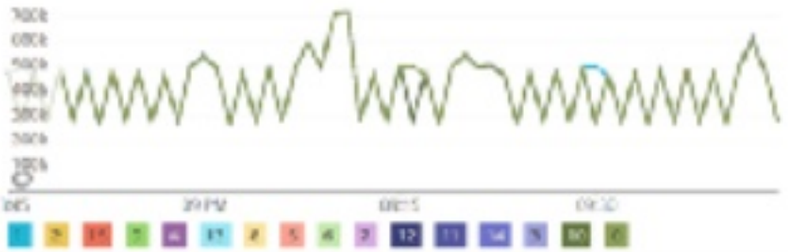
Flink Monitoring

Use it!

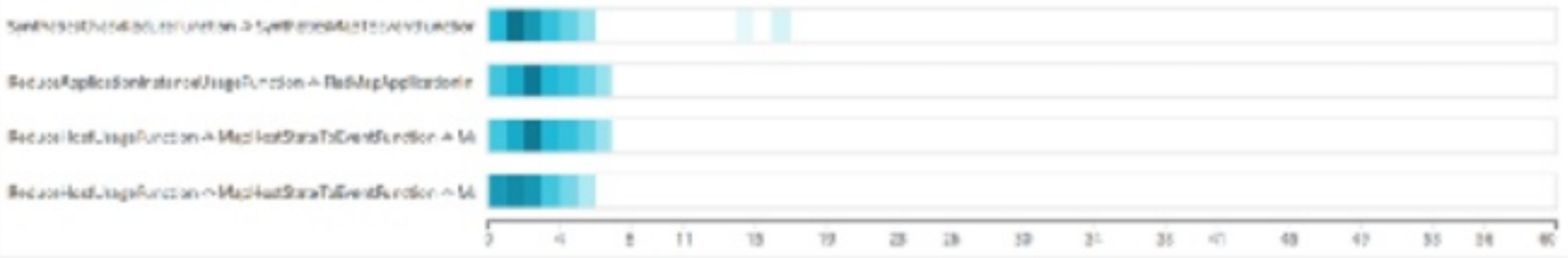
Flink Monitoring

Use it!

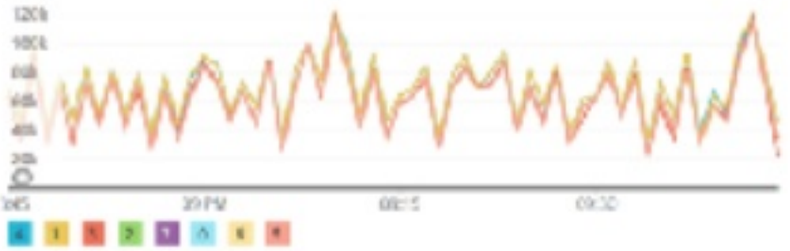
APM lag



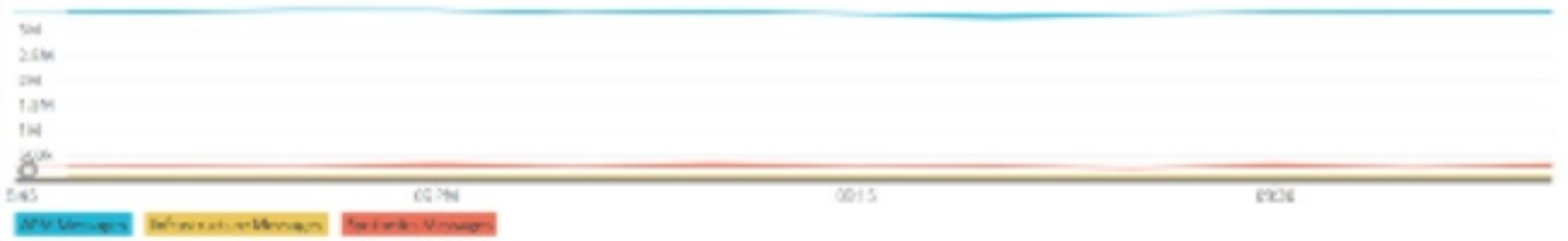
Watermark Samples (sec past realtime)



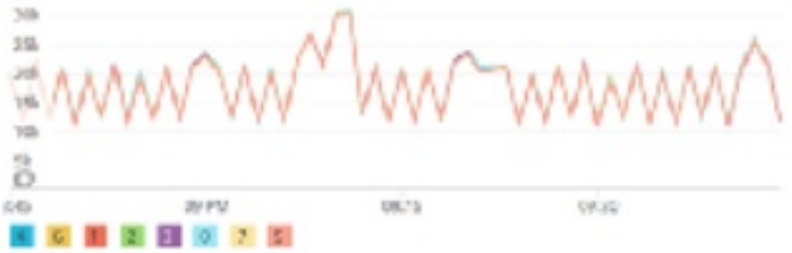
Synthetics Checks Lag



Kafka consumption



Infrastructure lag



Calculator Checkpoint Size

Throughput at Reducers

Flink Monitoring

Use it!

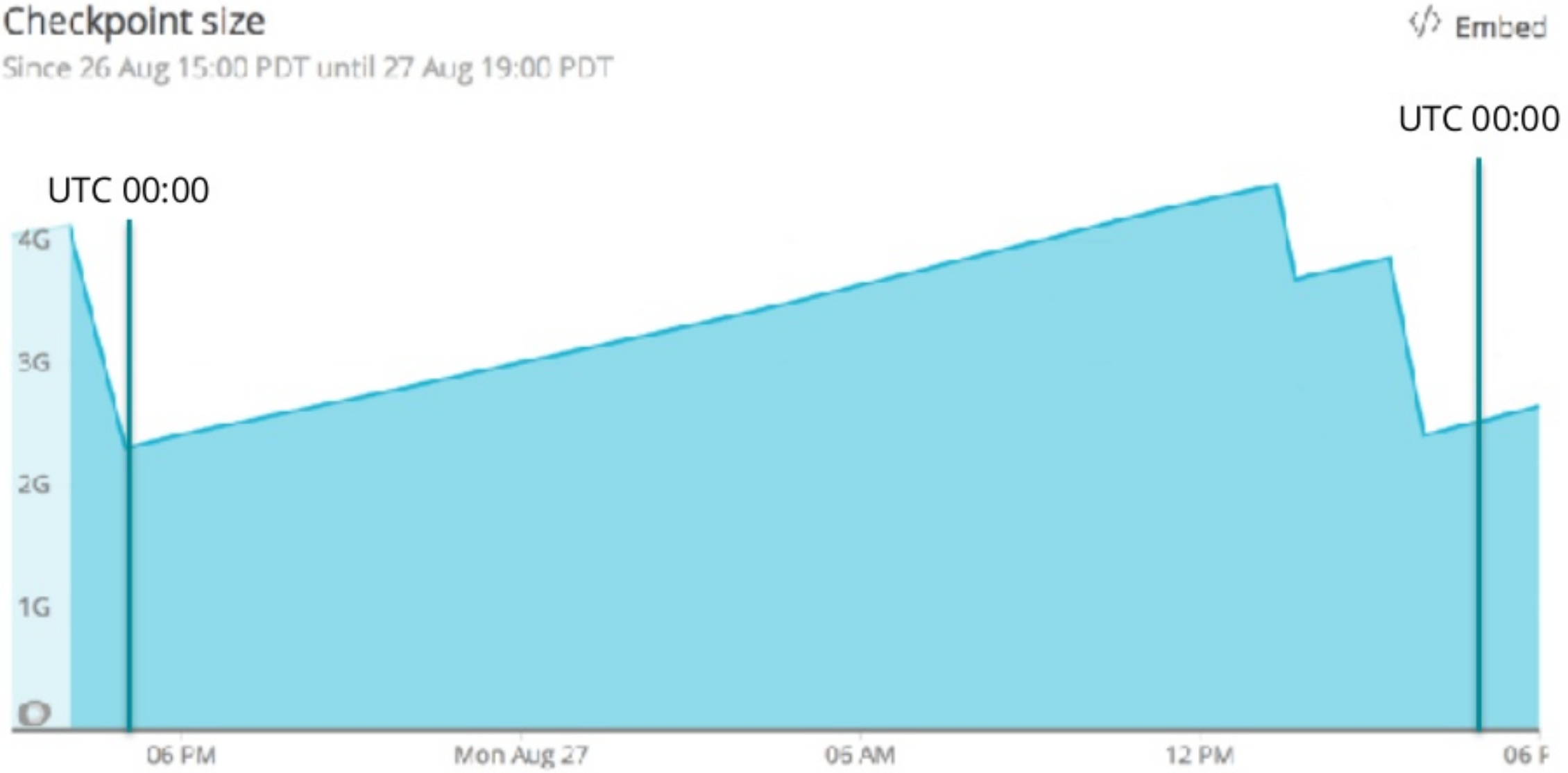
Flink Monitoring

Now what? Keep consuming your monitoring

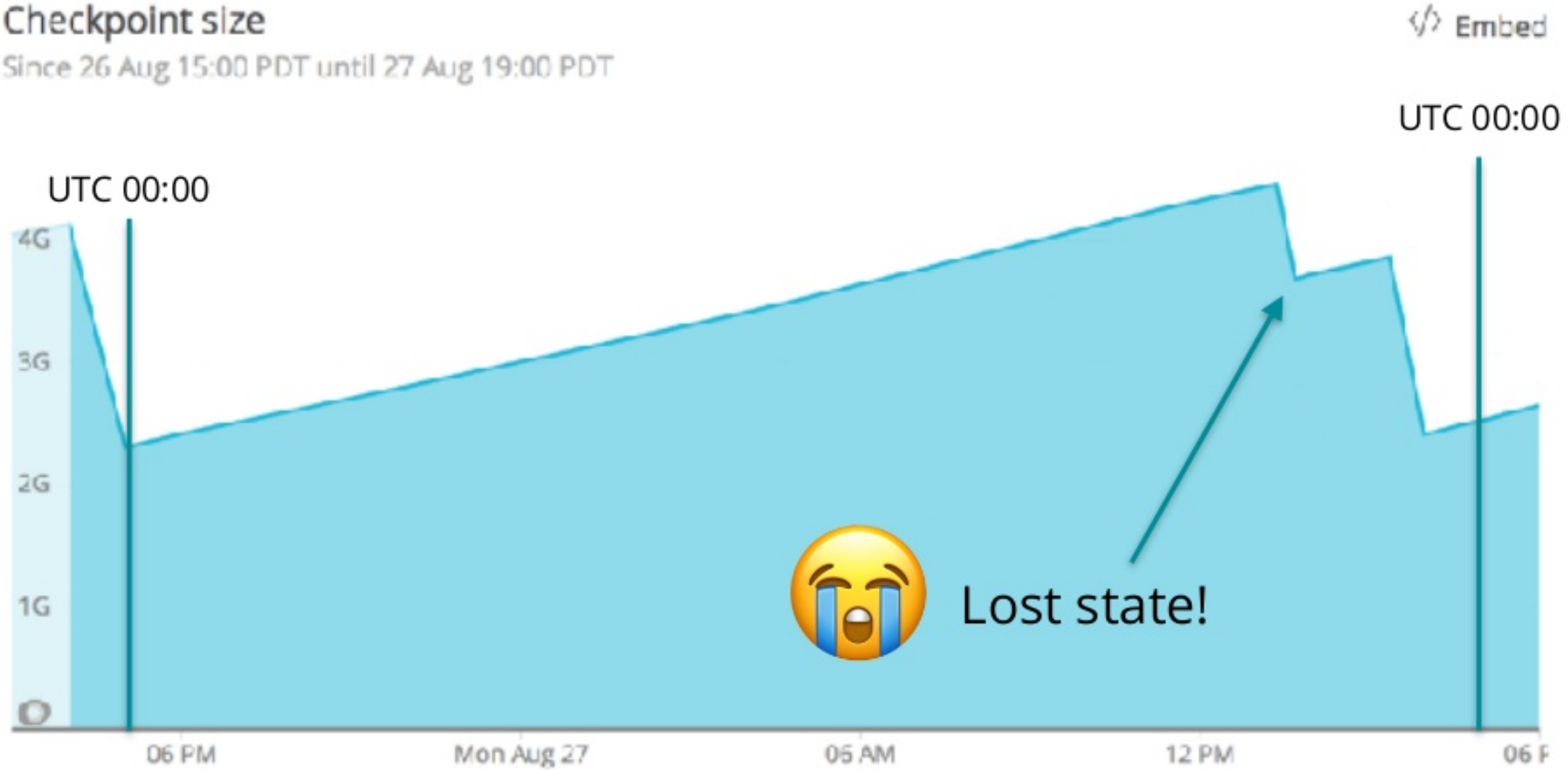
Flink Monitoring

Information radiation

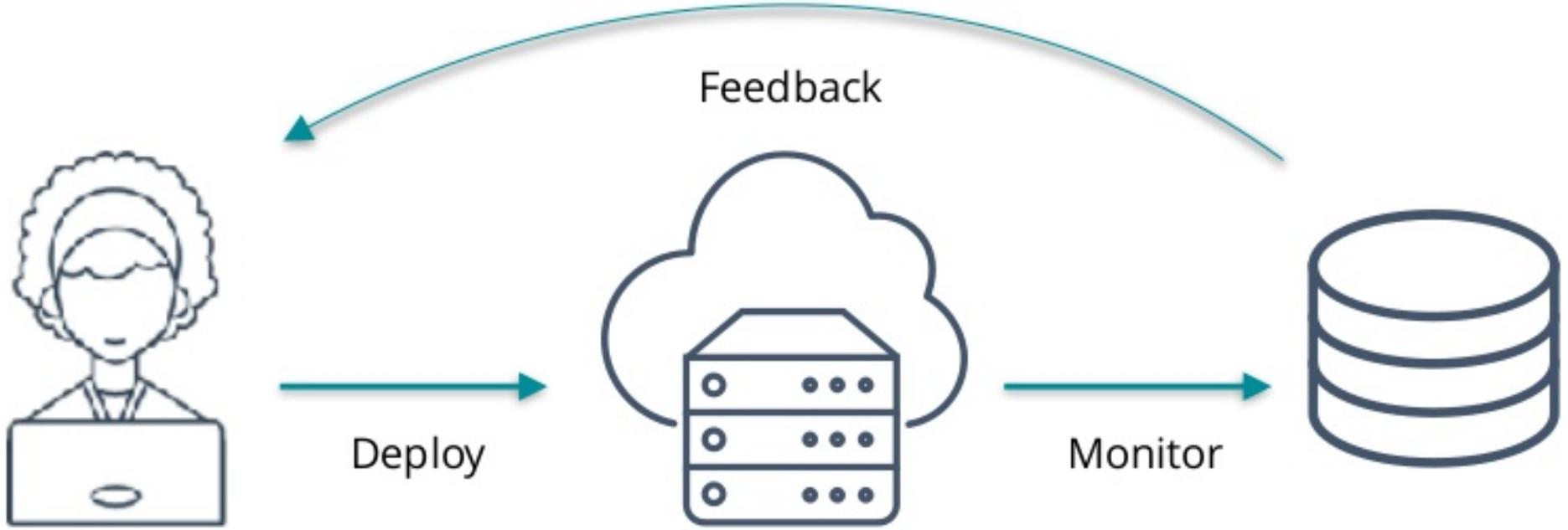
Flink Monitoring



Flink Monitoring



Flink Monitoring



Flink Monitoring

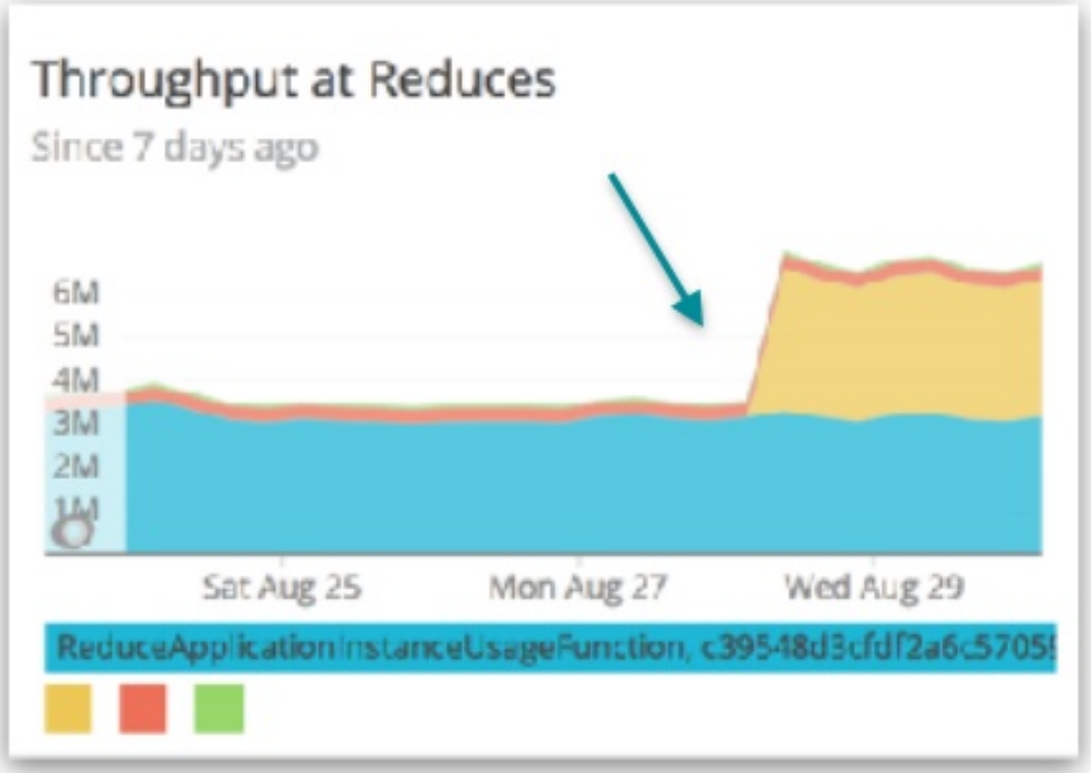
Metrics have some area for growth

- No distributed tracing, Flink-managed memory

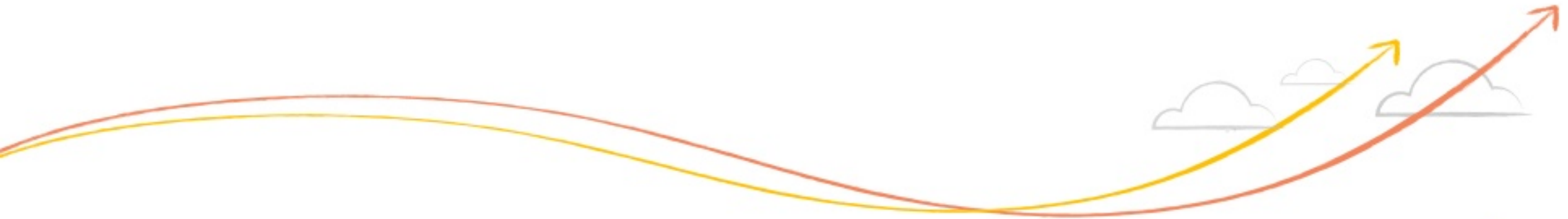
Flink Monitoring

Metrics have some area for growth

- No distributed tracing, Flink-managed memory
- Can be buggy



6000 Meter View



6000 Meter View

PoCs:

Flink Monitoring:

6000 Meter View

PoCs: **make it quick, make it work**

Flink Monitoring:

6000 Meter View

PoCs: **make it quick, make it work**

Flink Monitoring: **use it, everywhere!**

Thank You!

Flink Forward

AX Team (past and present)

Alysa Wood

Jen Hammond

Ralph Bodenner

Jeff Ostrin

Maureen Dugan

Moof Mayeda

Nate Borrebach

Ruby Andrews

Ron Crocker

Bill Green

Keep in touch!

Caito: cscherr@newrelic.com

Nik: ndavis@newrelic.com

New Relic Blog: https://blog.newrelic.com

New Relic Blog, new Flink article! https://blog.newrelic.com/engineering/what-is-apache-flink/

Join our team!! https://newrelic.com/about/culture

Photo Credits

Profile Photo, Caito

Matthew Scharr, 2017

Profile Photo, Nik

New Relic, 2017

AX Team Photo:

Moof Mayeda, 2018

Flink Logo:

https://flink.apache.org/material.html

https://www.apache.org/foundation/marks/

Portland Skyline:

Michael Wiley/

https://tinyurl.com/yc2aubdt/CC BY

Model Train:

Kevin Phillips/

https://tinyurl.com/yaoow6jo/CC BY