Reducing MTTR and False Escalations: Event Correlation at LinkedIn

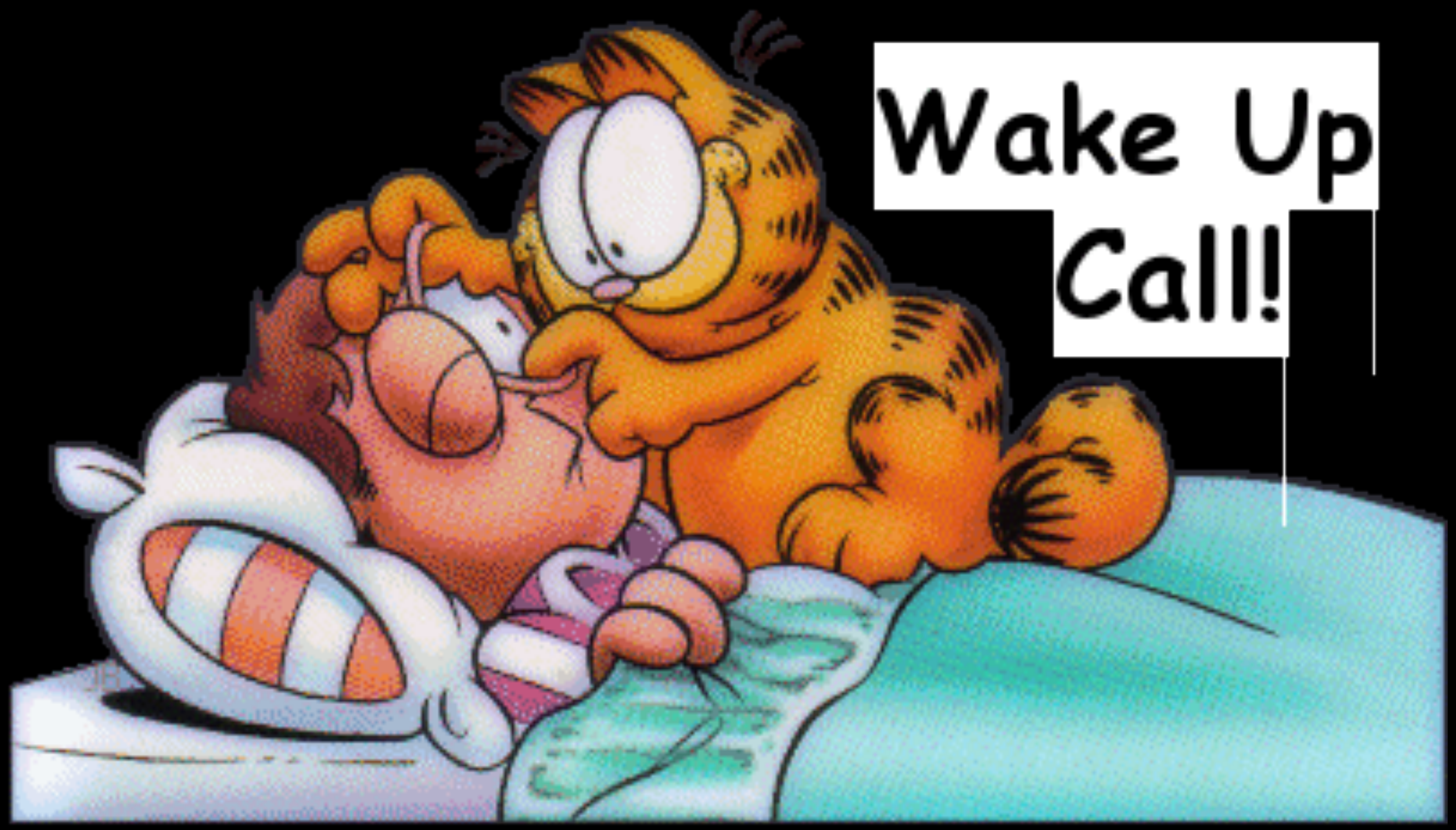
Michael Kehoe

Staff SRE

Rusty Wickell

Sr Operations Engineer

False Escalations



- Have you ever?
  - Been woken because your service is unhealthy because of a dependency
  - Been woken because someone believes your service is responsible
  - Spent hours trying to work out why your service is broken

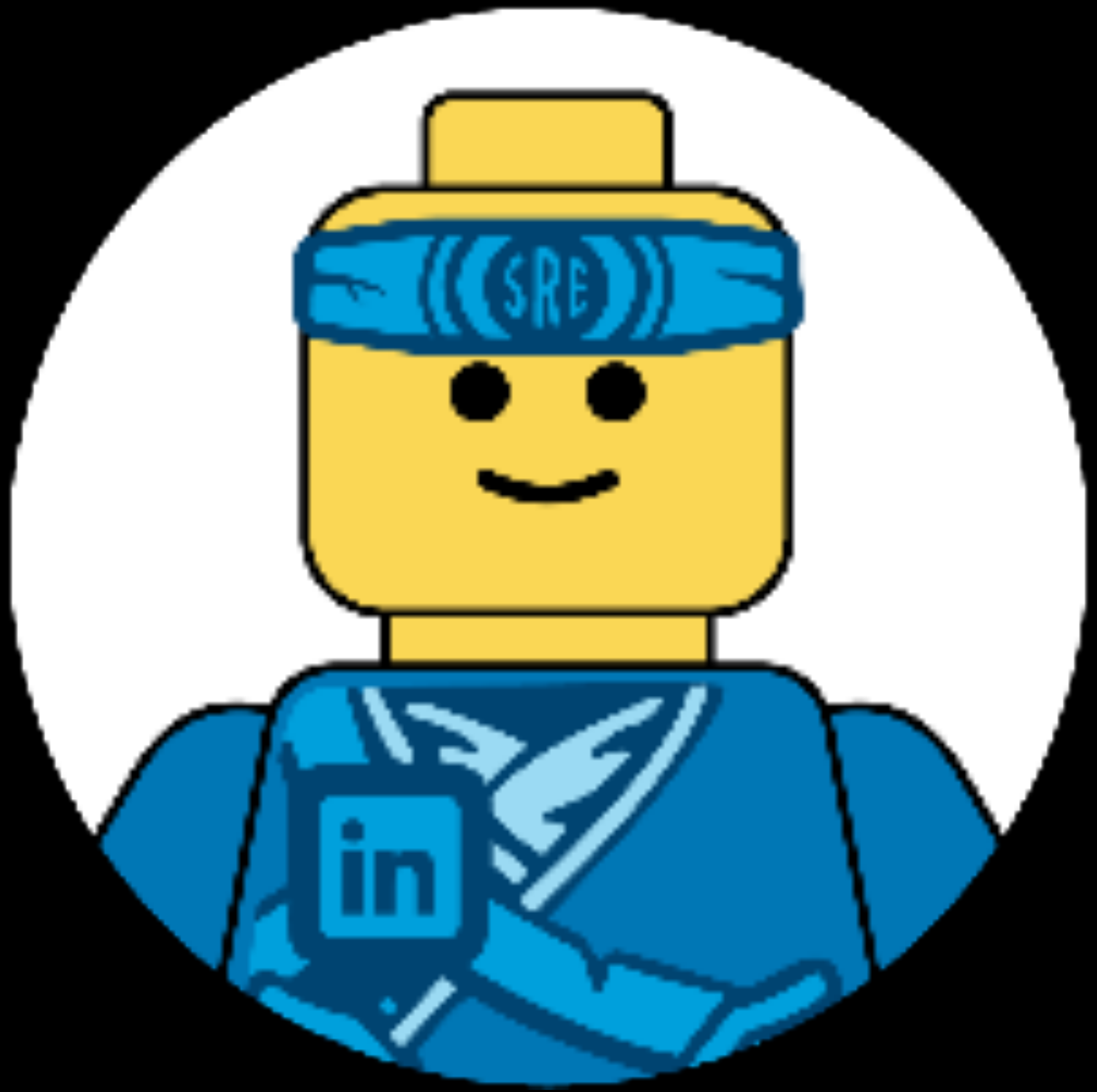
Today’s agenda

1. Introductions
2. The Problem Statement
3. Architecture Considerations
4. Platform Overview
5. Ecosystem Integration
6. Key Takeaways
7. Q&A

Introduction

Who are we?

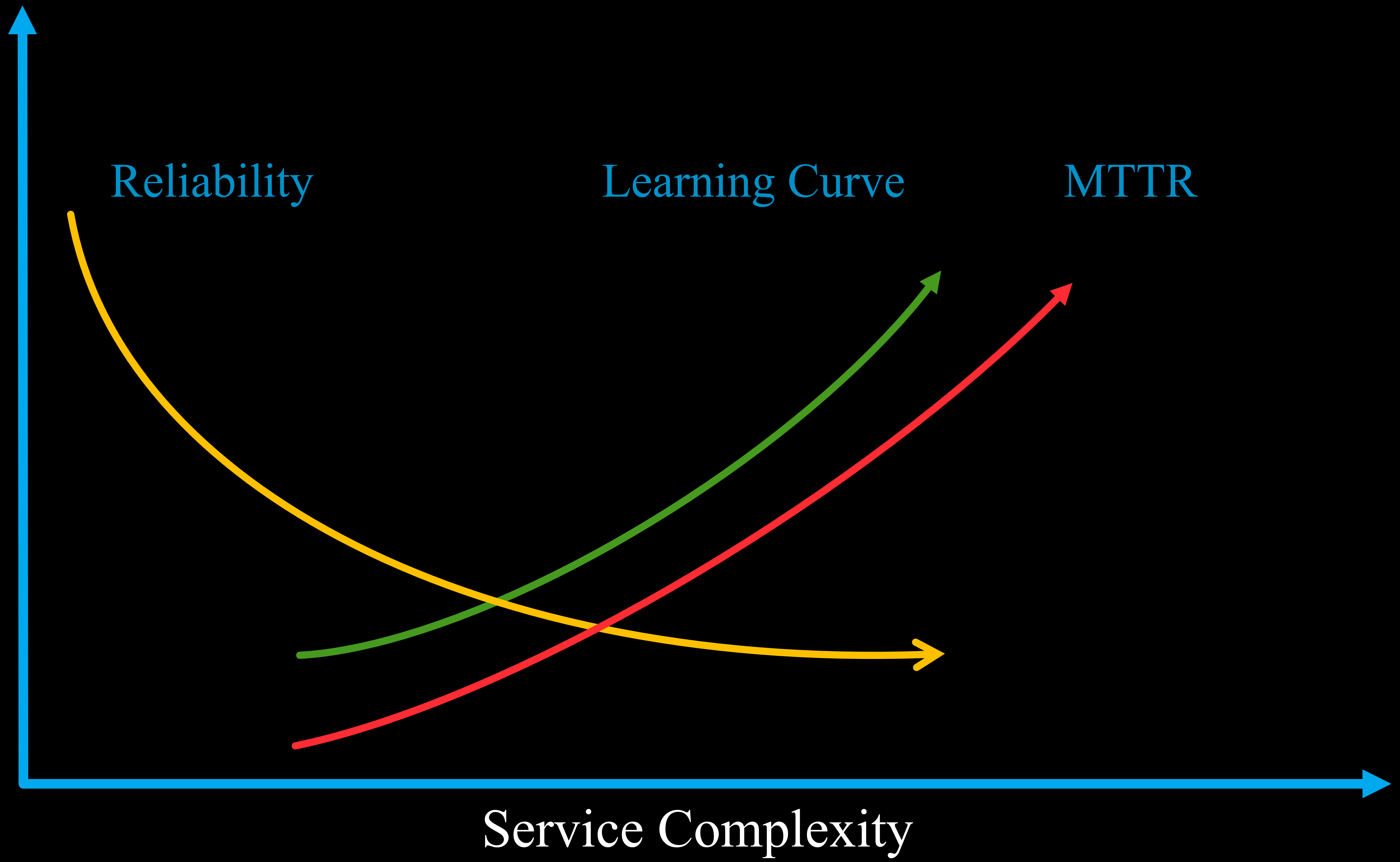
PRODUCTION-SRE TEAM AT LINKEDIN



- Assist in restoring stability to services during site-critical issues
- Develop applications to improve MTTD and MTTR
- Provide direction and guidelines for site monitoring
- Build tools for efficient site-issue detection, correlation & troubleshooting,

Problem Statement

Problem Statement



Problem Statement

Learning Curve

Understanding services is harder

High MTTR

Complexity delays identification of cause

False Escalations

Lack of understanding results in false escalations

Project Goals

Project Goals

Unified API

Internal application shows high latency/ errors

Web Frontend

External monitoring show high latency/ errors

Project Goals

Reduce MTTR

Reduce impact on members

Reduce False Escalations

Less disruptions to oncall SRE’s

Project Goals

Applicable Use-cases

Internal application shows high latency/ errors

Non-Applicable Use-cases

External monitoring show high latency/ errors

Architecture Considerations

Architecture Considerations

Real-Time Metrics Analysis

Running metric correlation via stream-processing

Ad-Hoc metric analytics

Metric correlation on demand

Alert Correlation

Processing alerts and performing

Architecture Considerations

REAL-TIME METRIC ANALYTICS

- Pros
  - Fast response time
  - Ability to do advanced analytics in real-time
- Cons
  - Resource intensive = Expensive

Architecture Considerations

AD-HOC METRIC ANALYTICS

- Pros
  - Smaller resource footprint
- Cons
  - Analysis time is slow

Architecture Considerations

ALERT CORRELATION

- Pros
  - Leverage already existing alerts
  - Strong signal-to-noise ratio
- Cons
  - Analysis constrained to alerts only (boolean state)

Architecture Considerations

EVALUATION

- Alert Correlation gives us strong signal
- Real-time analytics is expensive, but useful
- Ad-Hoc metric analytics is slower, but cheaper

Platform Overview

Platform Overview

Call Graph

Understanding how services depend on each other

Ad-Hoc Metric Correlation

K-Means analysis

Alert Correlation

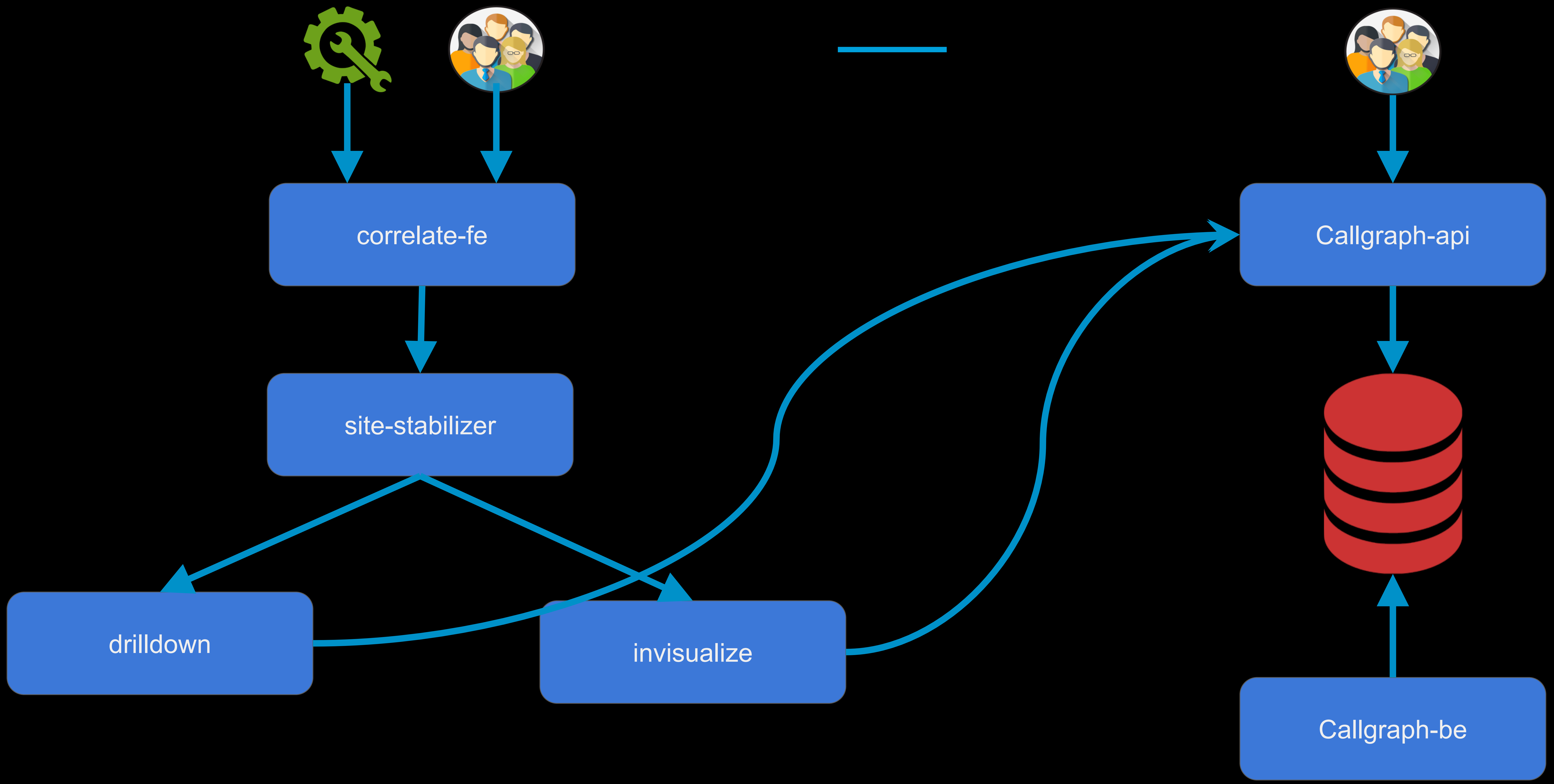
Using alerts to confirm performance

Recommendations Engine

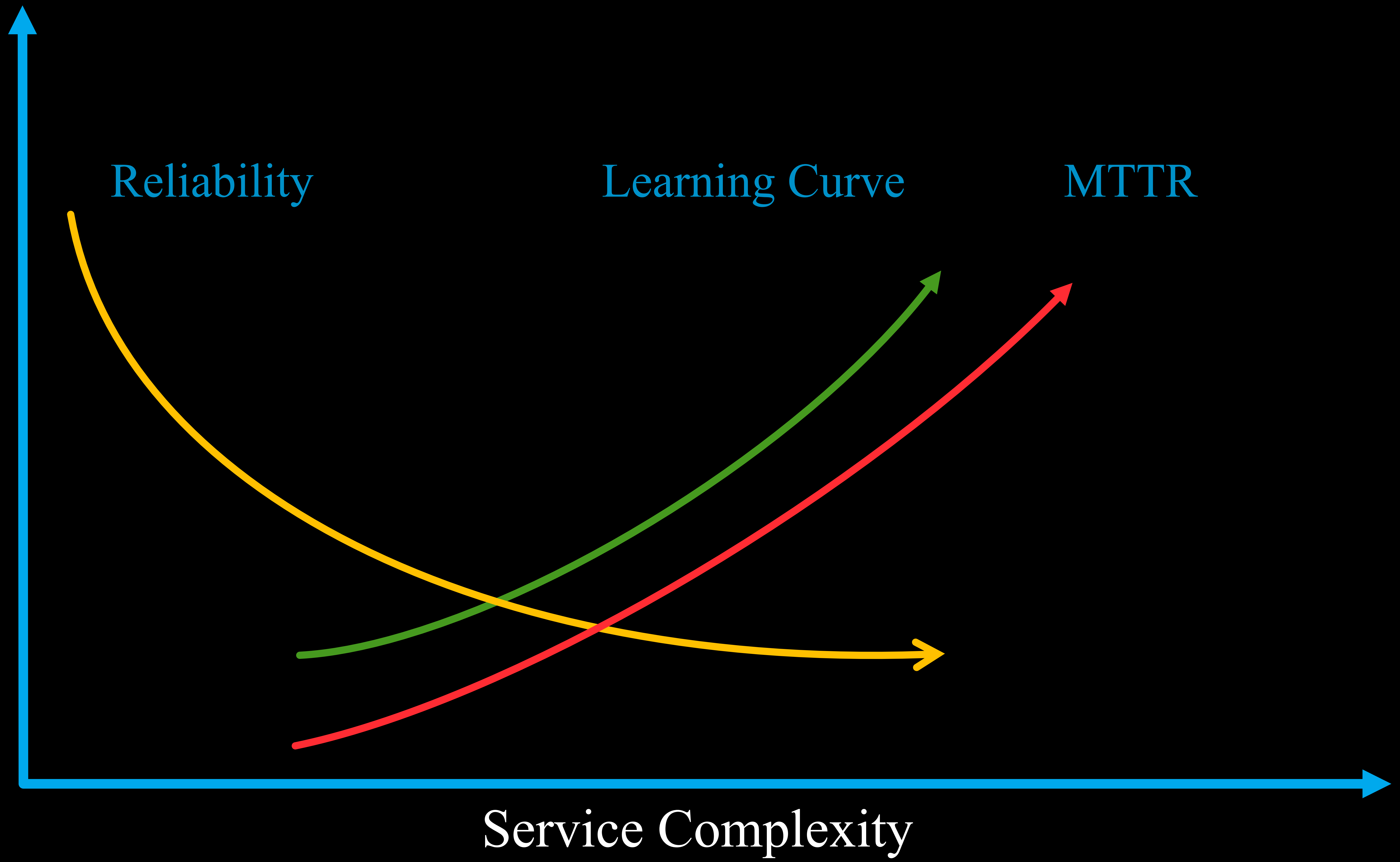
Collating and decorating data

Correlation Engine Overview

Architecture



Problem Statement

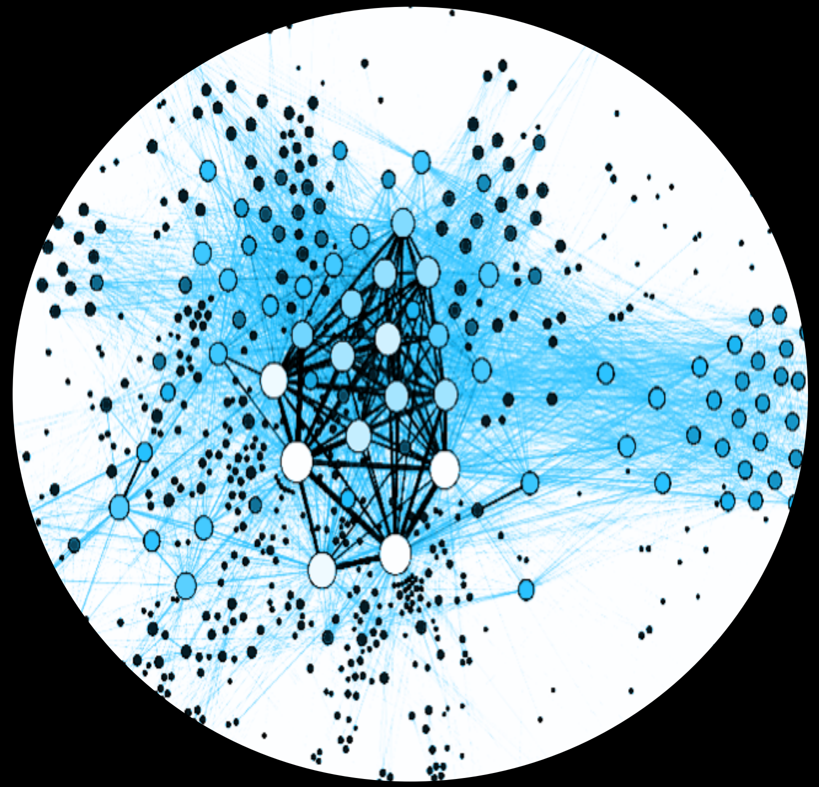


Learning Curve

Scattered Knowledge

Outdated Documentation

Poor Dependency Understandings



Callgraph

Created
Programmatically

Interface
API and a User Interface

Lookup
Service/ API

Stores
Callcount, latency and error rates

How do we map service

Service Discovery

Services, APIs, Protocols

Metrics

Destination service,
Endpoint, Protocol

Site Stabilizer | Real Time and Ad-Hoc Metrics Analysis

Approaches that we tried

Threshold

Challenge: Not all metrics had thresholds

Statistical

Challenge: expensive, real time processing, tuning based on the individual metrics behaviour

Machine Learning

Challenge: expensive, real time processing, tuning based on the individual metrics behaviour

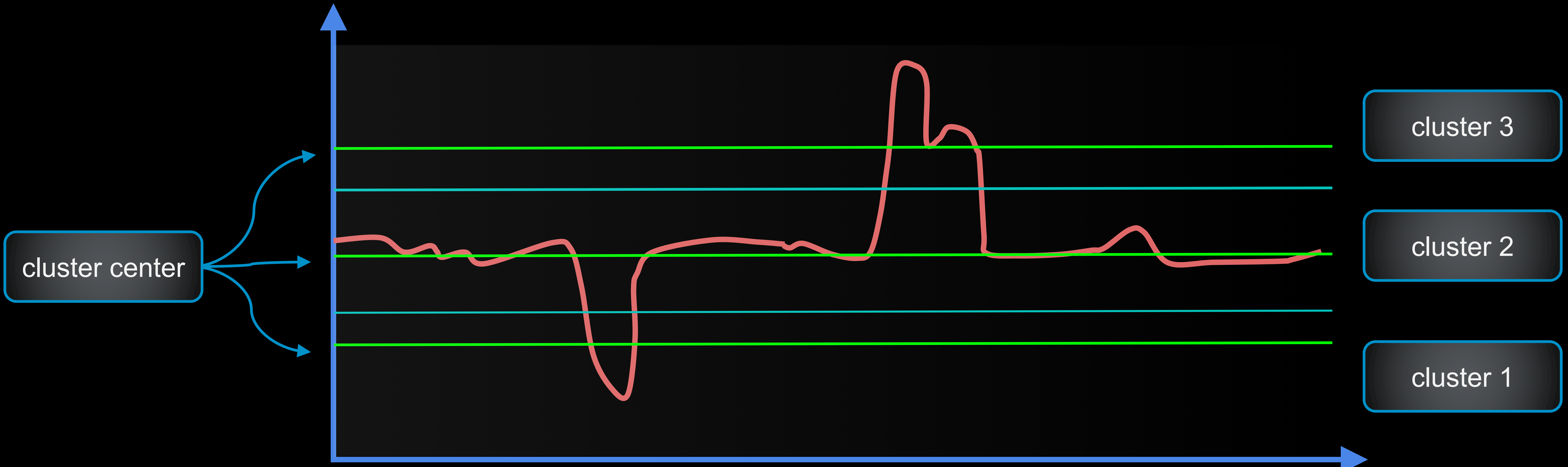
Clustering Algorithm

K-Means

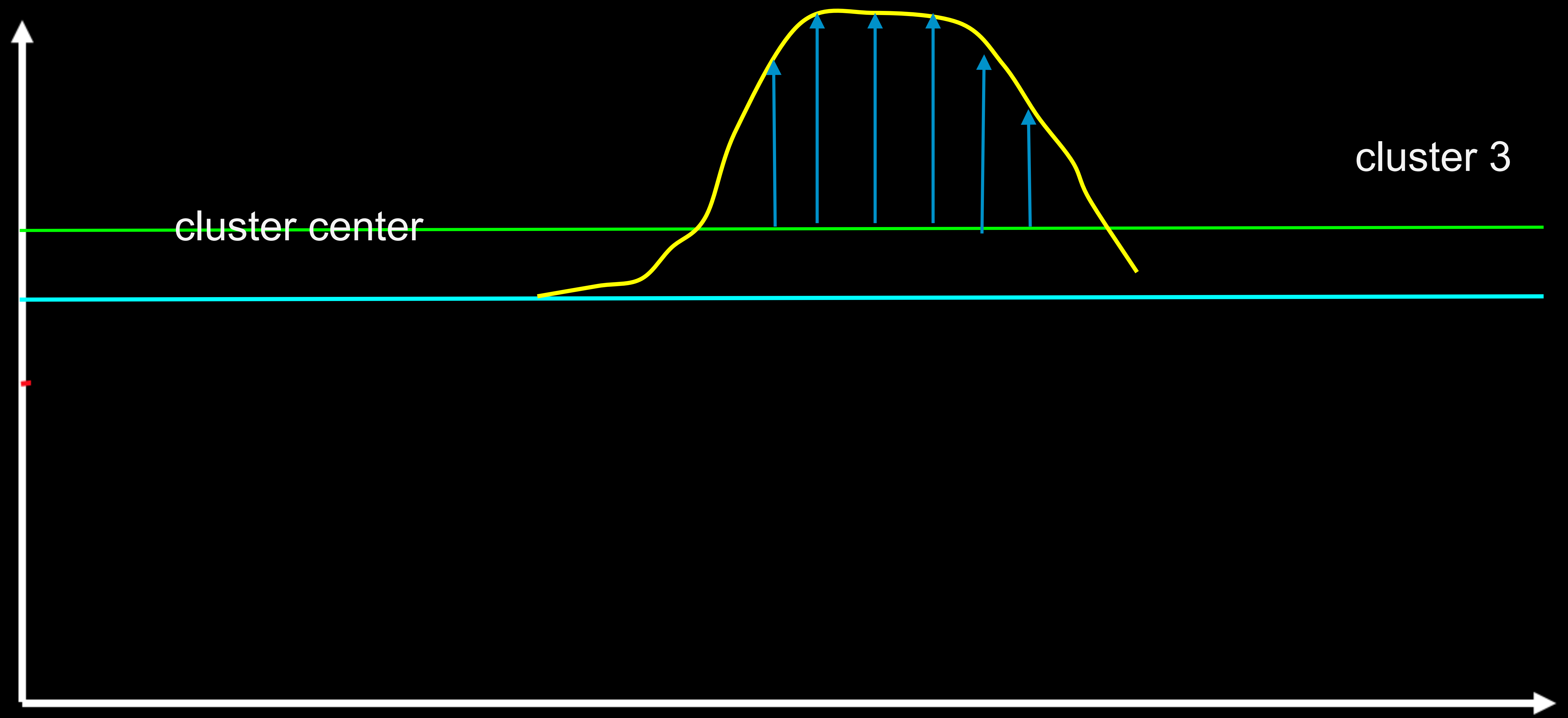
Partitions
n observations to *k* clusters

Store
Can be trained and saved

K-Means : How it works



Predict score



Ranking

Predict score

Using K-Means

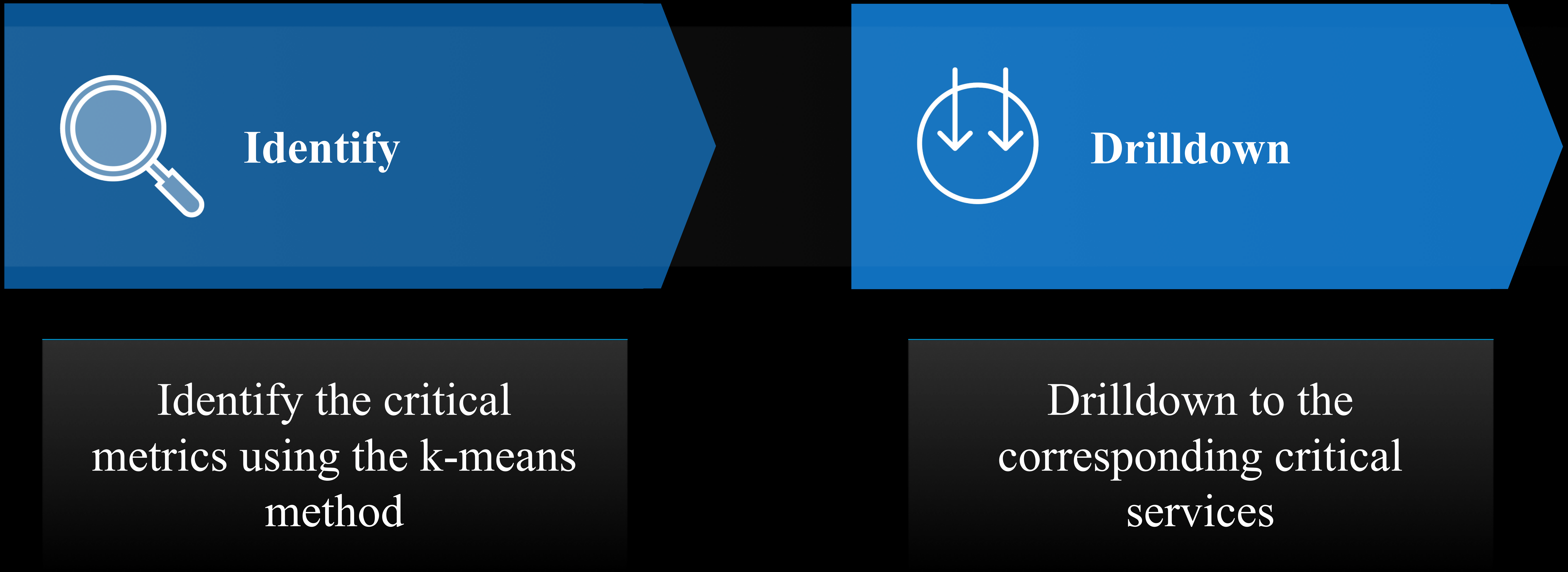
Trend score

Based on the trend of the time series

WoW

Leverage week on week data

Typical Workflow



Identify

Identify the critical metrics using the k-means method

Drilldown

Drilldown to the corresponding critical services

inVisualize | Alert Correlation and Visualization

inVisualize Assumptions

inVisualize

Alert Correlation and Visualization

Polls the monitoring system continuously for alerts

Correlates downstream alerts using Callgraph

Ingests and represents callcount, average latency, error rate from callgraph

inVisualize Assumptions

inVisualize

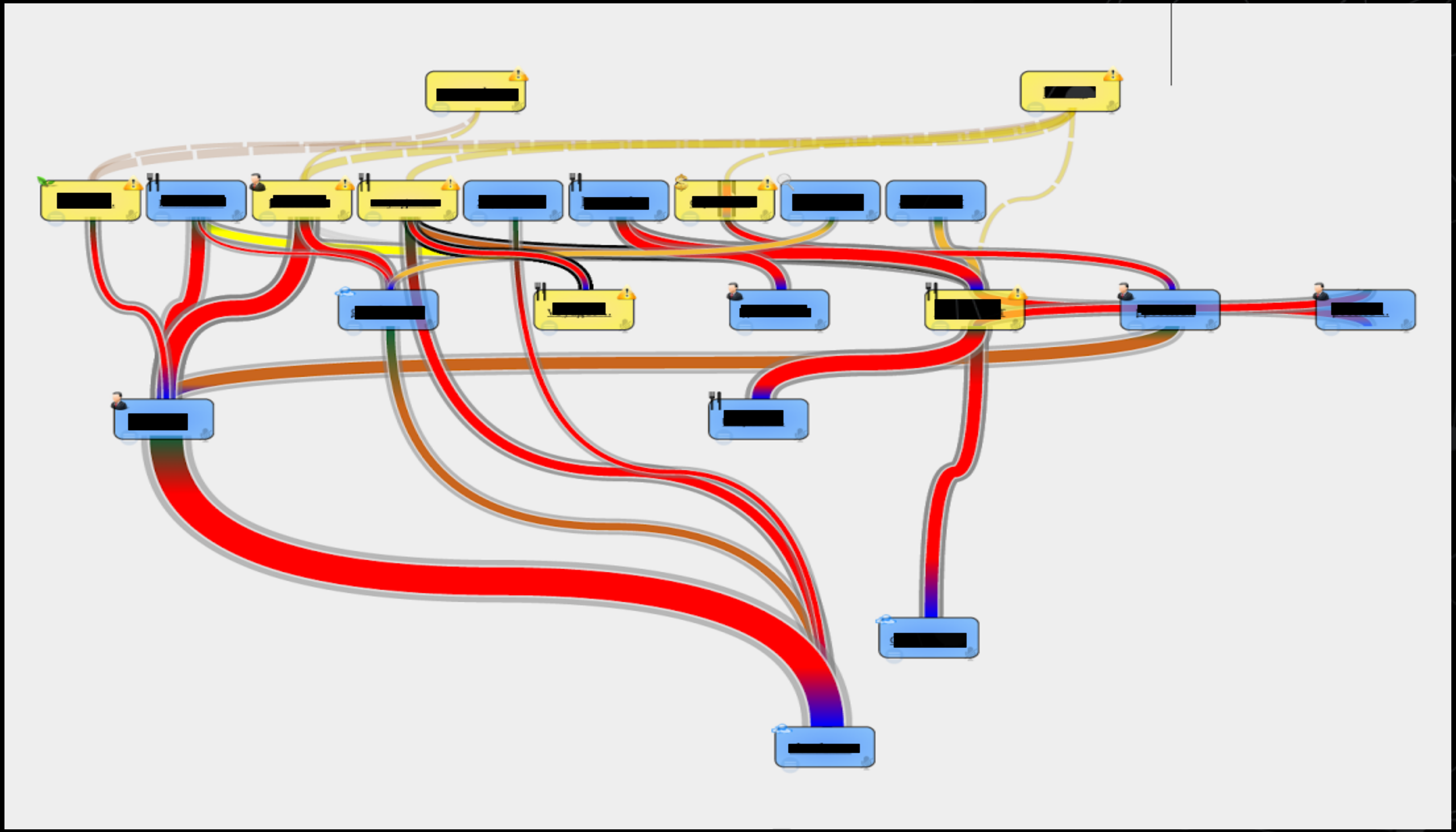
Alert Correlation and Visualization

Higher the alerts for a service, more likely it's affected or **broken**

Higher the callcount to a downstream, more **valuable** it is

Higher the change in latency/error to a downstream, more likely it's **broken**

inVisualize



inVisualize

Alert Correlation and Visualization

Save the states continuously for replay

Rank the services based on a score and accessible via api

Score is normalized between 0-100

Recommendation Engine

Recommendation Engine

Input

Service, colo, duration

Collate

Collates the outputs from Site stabilizer and **in**Visualize

User Interface

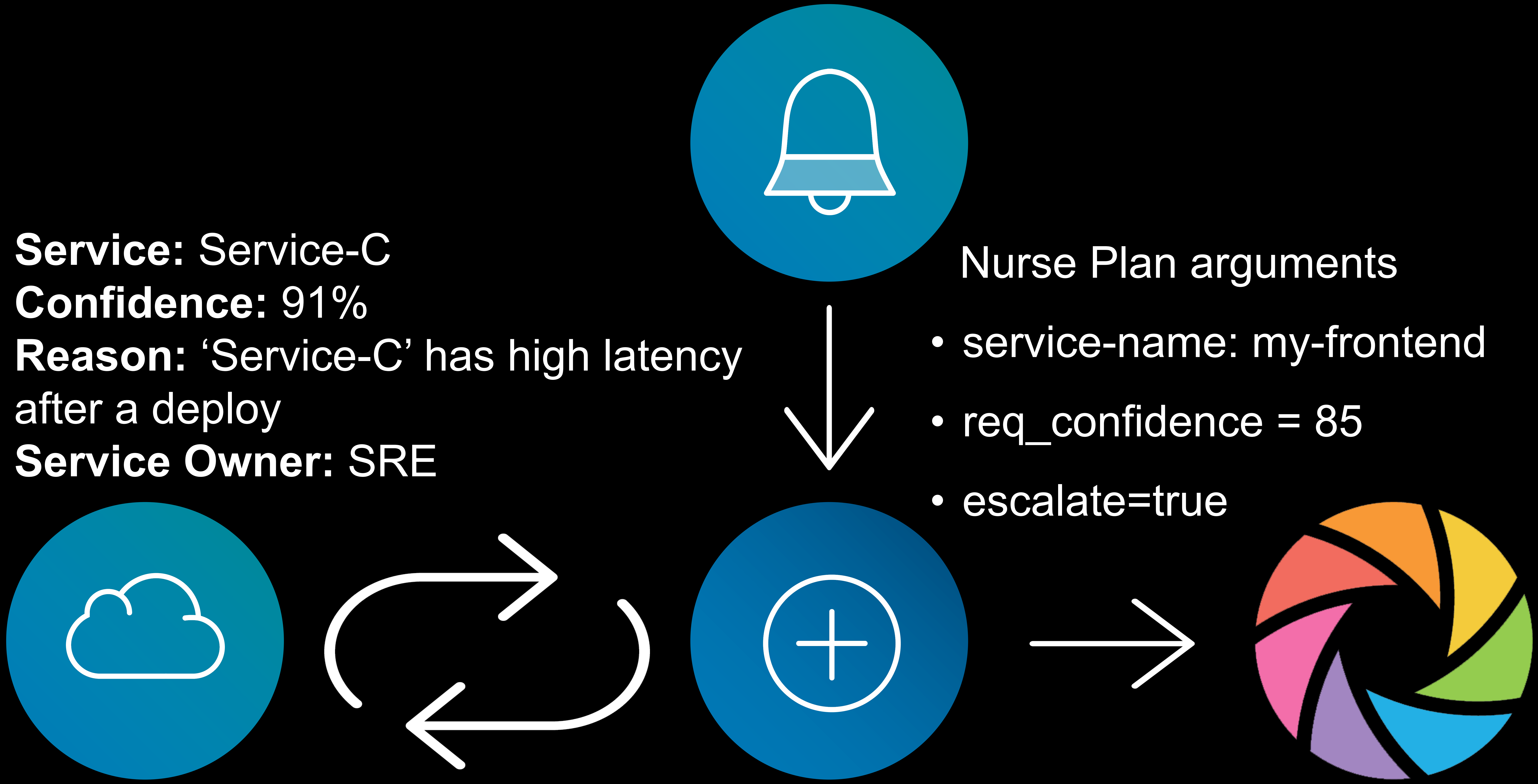
Responsible service, SRE team, correlation confidence score

Decorate

With information such as scheduled changes, deployments and A/B experiments

Ecosystem Integration

Ecosystem Integration



Service: Service-C
Confidence: 91%
Reason: ‘Service-C’ has high latency after a deploy
Service Owner: SRE

Nurse Plan arguments

- service-name: my-frontend
- req_confidence = 85
- escalate=true

Find what’s wrong with ‘my-frontend’ in DatacenterB

Escalate to correct SRE

Key Takeaways

Key Takeaways

Approach

Understand what correlation infrastructure makes sense

Dependencies

Understand dependencies

Key Takeaways

Feedback Loops

- Important to get some feedback on accuracy
- Provides a means to do reporting:
  - System effectiveness
  - Engineers saved from escalations
- Use feedback data to train system = Improve Results

Team

Michael Kehoe

Rusty Wickell

Reynold PJ

Govindaluri Kishore

Renjith Rejan

Q&A

LinkedIn