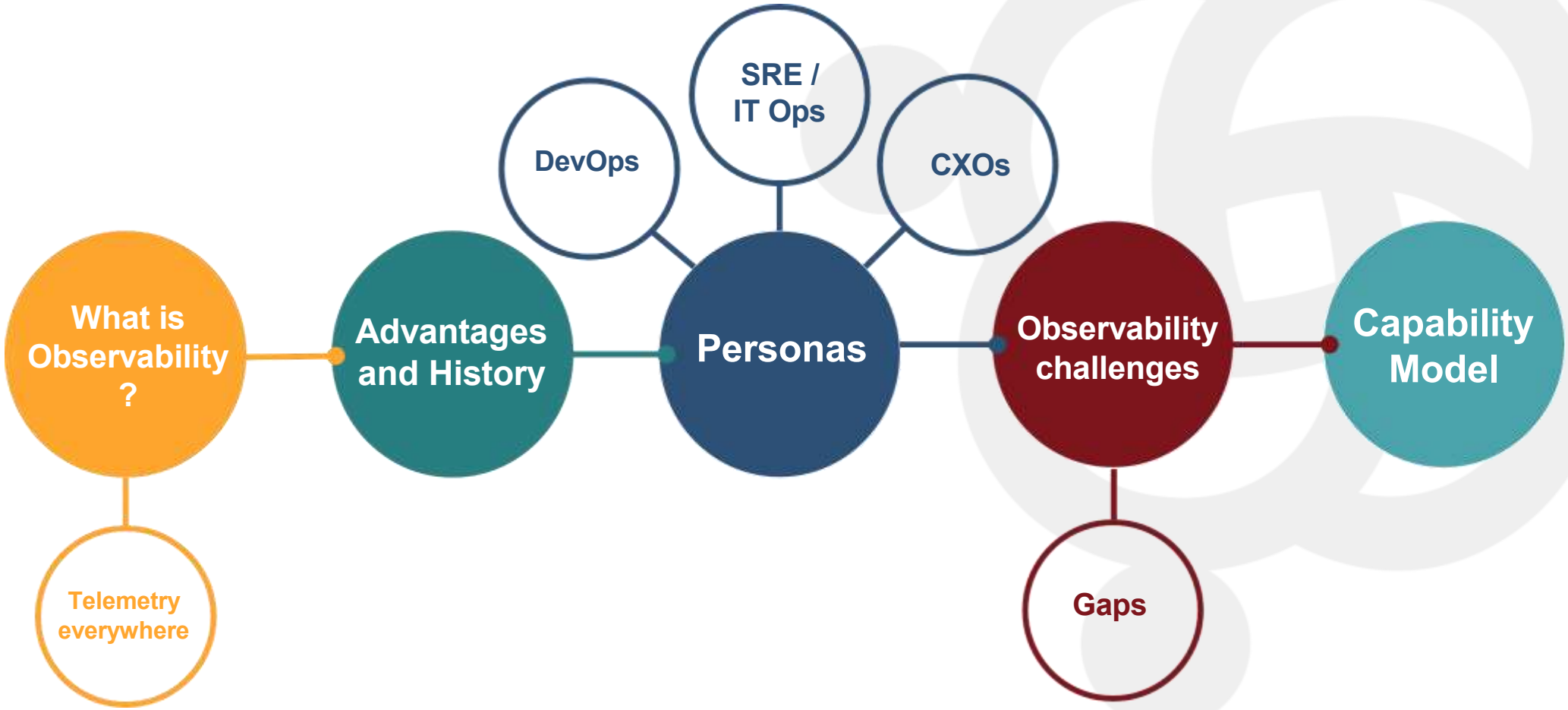
DevOps Institute
ADVANCING THE HUMANS OF DEVOPS

# Observability & Monitoring

# Flow: Talk Map



# What is Observability?

**Clue: It's not monitoring.**

Observability is a characteristic of systems; that they can be observed. It's closely related to a DevOps tenet: 'telemetry everywhere', meaning that anything we implement is emitting data about its activities. It requires intentional behavior during digital product and platform design and a conducive architecture. It's not monitoring. Monitoring is what we do when we observe our observable systems and the tools category that largely makes this possible.

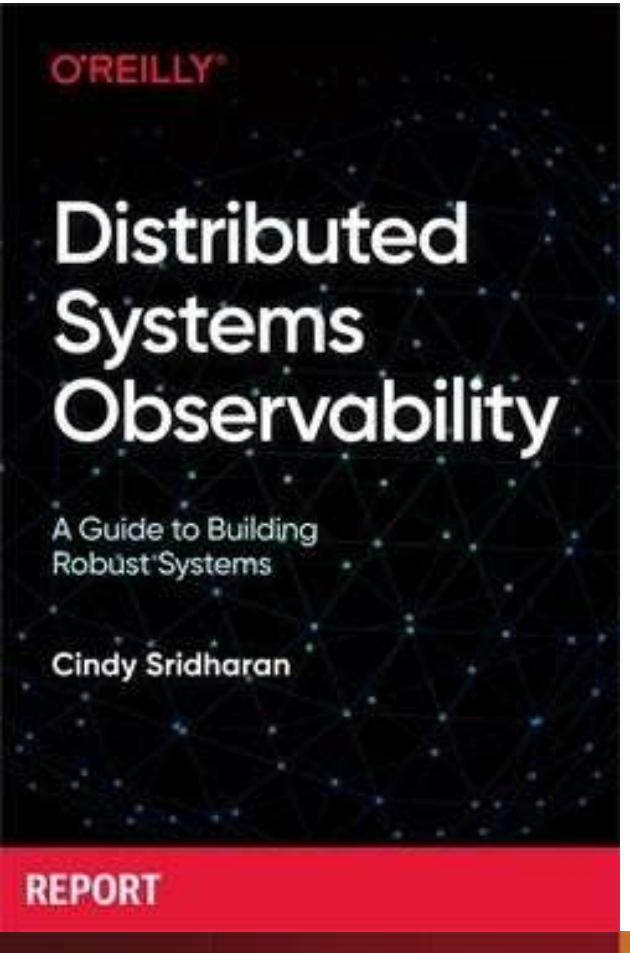
# Where has the concept come from?

**“On the General Theory of Control Systems’ by Rudolf E. Kálmán in 1960**

In control theory, observability is defined as a measure of how well internal states of a system can be inferred from knowledge of its external outputs.

# Applying Observability to Software Systems

## Observability's origins are in mechanical engineering but...

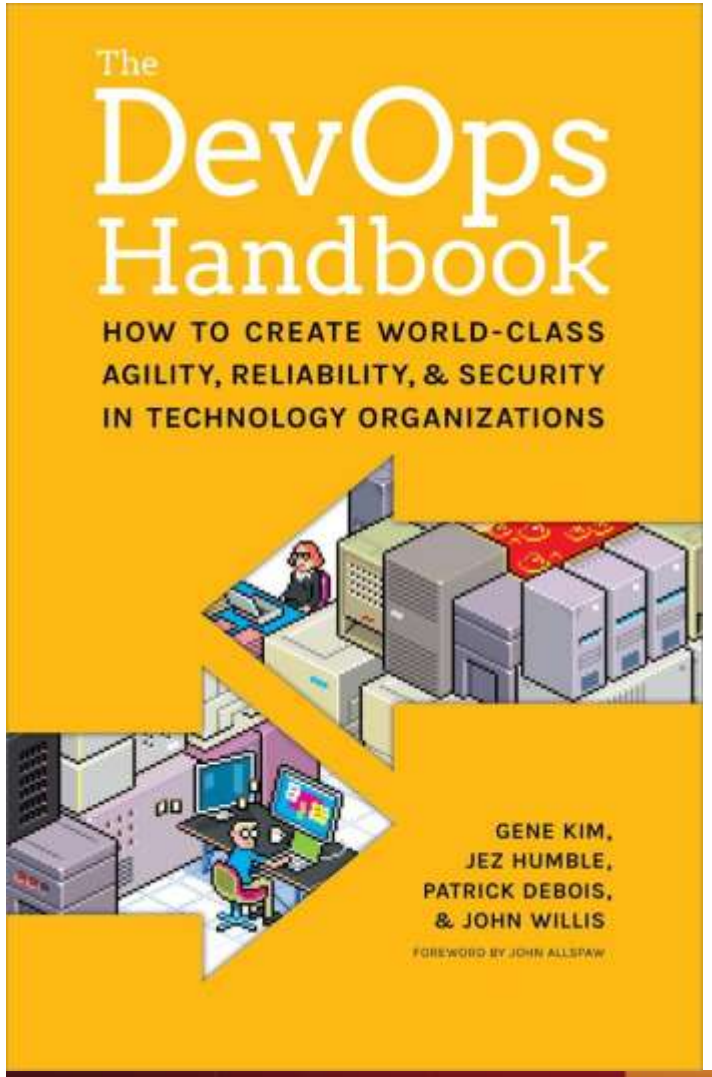



"We believe that observability requires evolving the essence of how we think about gathering the data needed to debug effectively. You must be able to understand:

- The inner workings of your applications/services
- Any system state your applications/services many have evolved to
- These things solely by observing that with external tools
- That state, no matter how extreme or unusual"

# Telemetry Everywhere

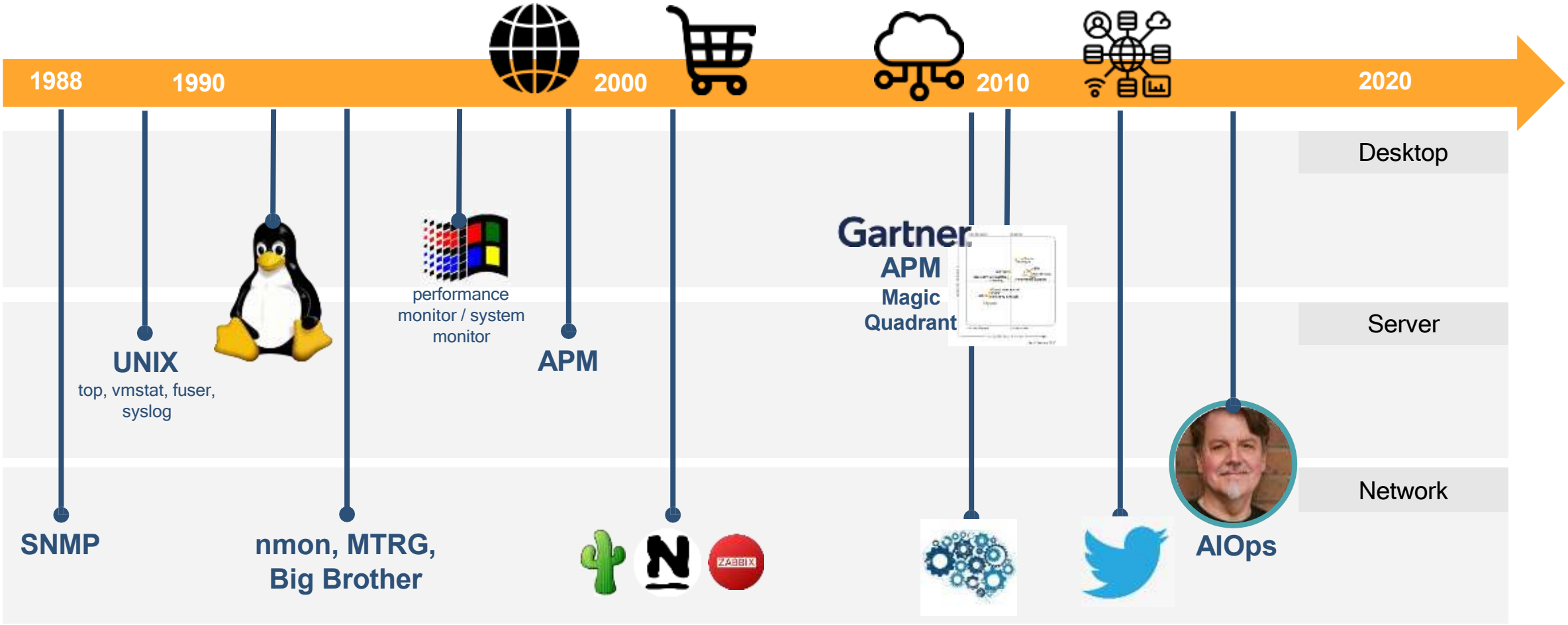
## Is it the same as observability?



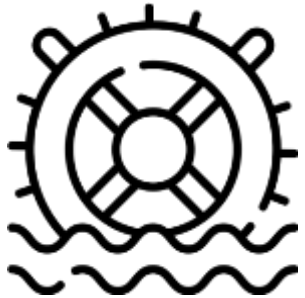

"We need to design our systems so that they are continually creating telemetry, widely."

"Telemetry is what enables us to assemble our best understanding of reality and detect when our understanding of reality is incorrect."

# Evolution of Monitoring to Observability



1



The industrial revolution

1771

2

The age of steam and railways

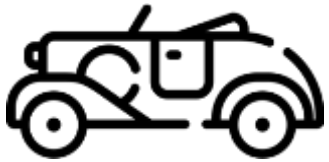
1829

3

Age of steel, electricity and heavy engineering

1875

4



Age of oil, automobiles and mass production

1908

5

Age of information and telecomms

1971

# Digital Disruption and Transformation

Digital transformation is the integration of digital technology into all areas of a business resulting in fundamental changes to how businesses operate and how they deliver value to customers. Beyond that, it's a cultural change that requires organizations to continually challenge the status quo, experiment often, and get comfortable with failure. This sometimes means walking away from long-standing business processes that companies were built upon in favor of relatively new practices that are still being defined.

# Advantages of Observability

**Leaders are...**


- **2.9 times** as likely to enjoy better visibility into application performance
- Almost **twice as likely** to have better visibility into public cloud infrastructure
- **2.3 times** as likely to experience better visibility into security posture
- **Twice as likely** to benefit from better visibility into on-premises infrastructure
- **2.4 times** likelier to have a tighter grasp on applications, down to the code level
- **2.6 times** likelier to have a fuller view of containers (including orchestration)
- **6.1 times** likelier to have accelerated root cause identification (43% of leaders versus 7% of beginners)

# CALMS and Observability

| Culture | Automation | Lean | Measurement | Sharing |
|---|---|---|---|---|
| Visibility and transparency builds trust<br><br>Data-driven not opinion-driven conversations<br><br>Fast feedback on experiments<br><br>A tool that supports team autonomy: “We build it, we own it” | Accelerated root cause(s) analysis and insights<br><br>Pre-emptive warning and forecasting operating behavior<br><br>Automated service assurance<br><br>Data discovery, crunch & insights | Accelerates flow (MTTx)<br><br>Removes handoffs and delays between teams<br><br>Observability across the end-to-end value stream<br><br>Focus on customer experience | Real data that measures progress and improvements<br><br>operations, SRE, SLOs and error budgets<br><br>Actionable insights based on streaming data<br><br>Telemetry everywhere | Provides a shared platform for collaborative analysis<br><br>Builds a knowledge base so local discoveries become global improvements<br><br>ChatOps |

# The Three Pillars


## LOGS

An event log is an immutable, timestamped record of discrete events that happened over time

Easy to generate and instrument.

Can cause performance issues.

## METRICS

Numeric representation of data measured over intervals of time.

Well-suited to dashboards and aggregation.

Historically poor dimensionality.

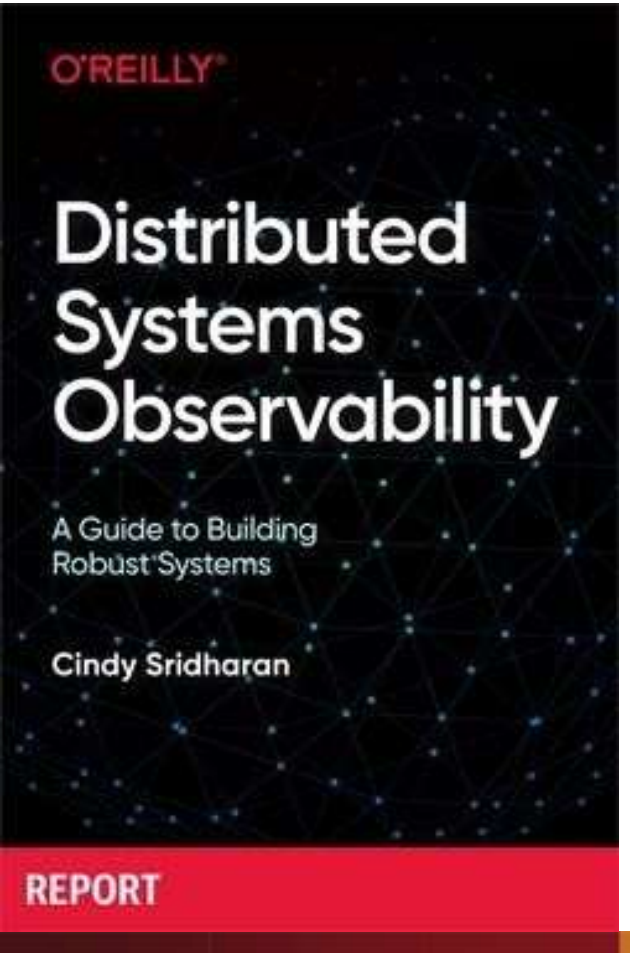
## TRACES

A representation of a series of causally related distributed events that encode the end-to-end request flow through a distributed system.

Myriad use cases.

Very challenging to retrofit.

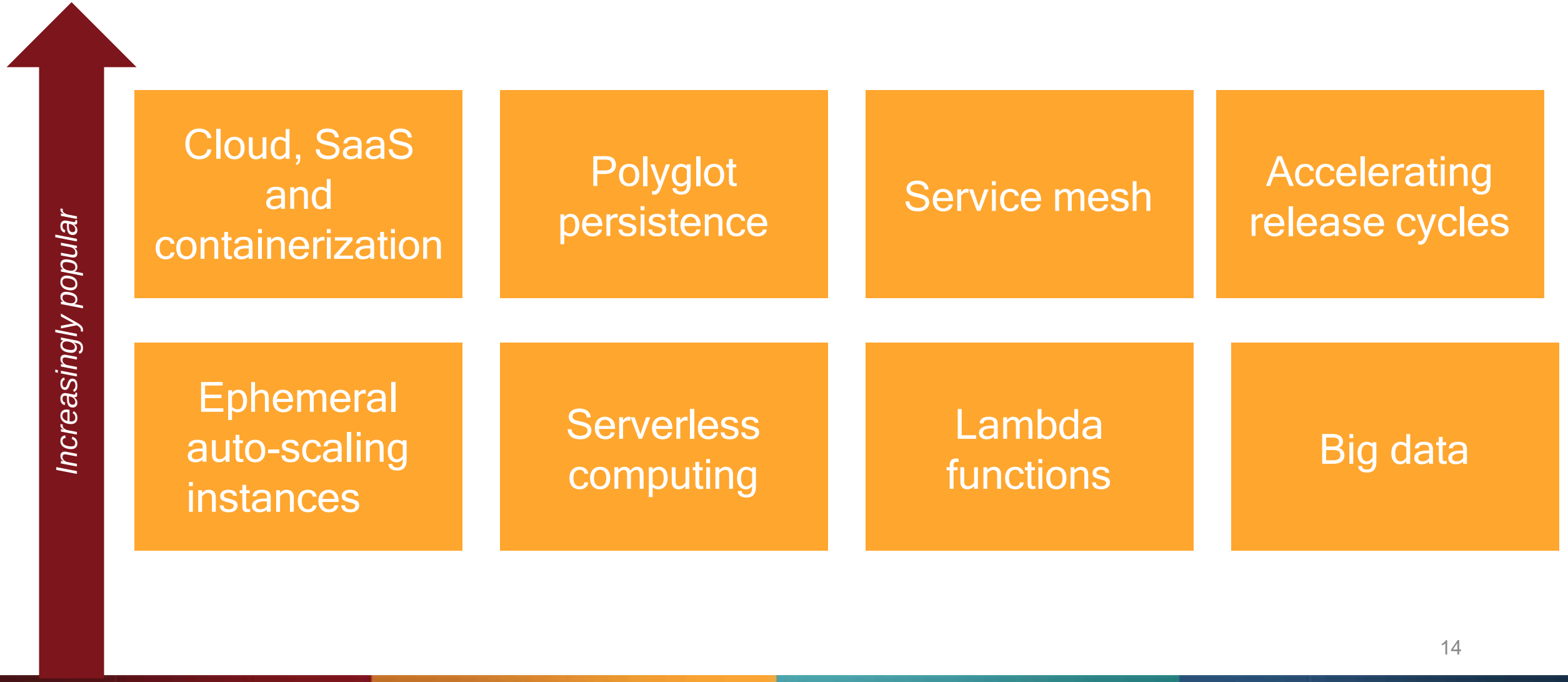
# Hidden Assumptions of Metrics




- Your application is monolithic in nature
- There is one stateful data store (“the database”)
- Many low-level systems metrics are available and relevant (e.g., resident memory, CPU load average)
- The application runs on VMs or bare metal, giving you full access to system metrics
- You have a fairly static set of hosts to monitor
- Engineers examine systems for problems only after problems occur
- Dashboards and telemetry exist to serve the needs of operations engineers
- Monitoring examines “black-box” applications that are inaccessible
- Monitoring solely serves the purposes of operations
- The focus of monitoring is uptime and failure prevention
- Examination of correlation occurs across a limited (or small) number of dimensions

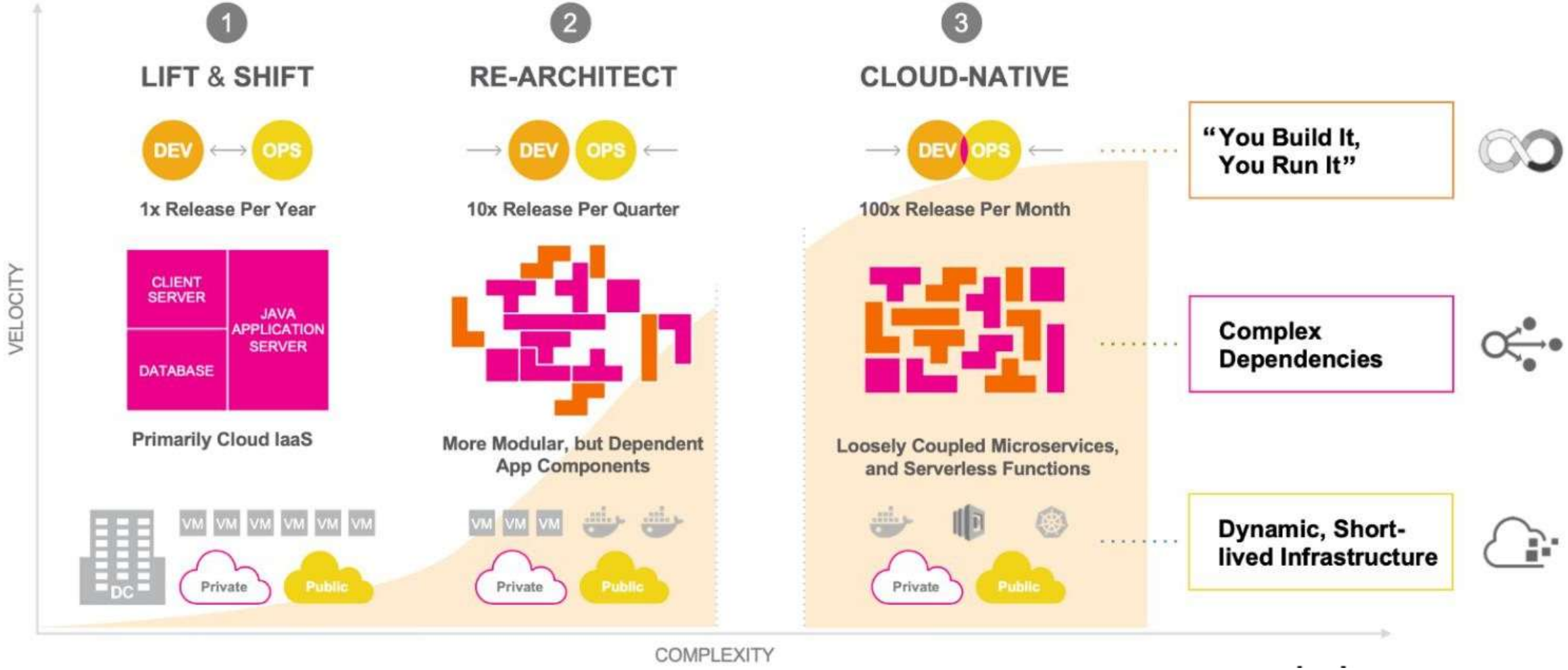
# The Progressive Platforms

**From monoliths to microservices - APIs rule**



# Cloud is a Critical Enabler

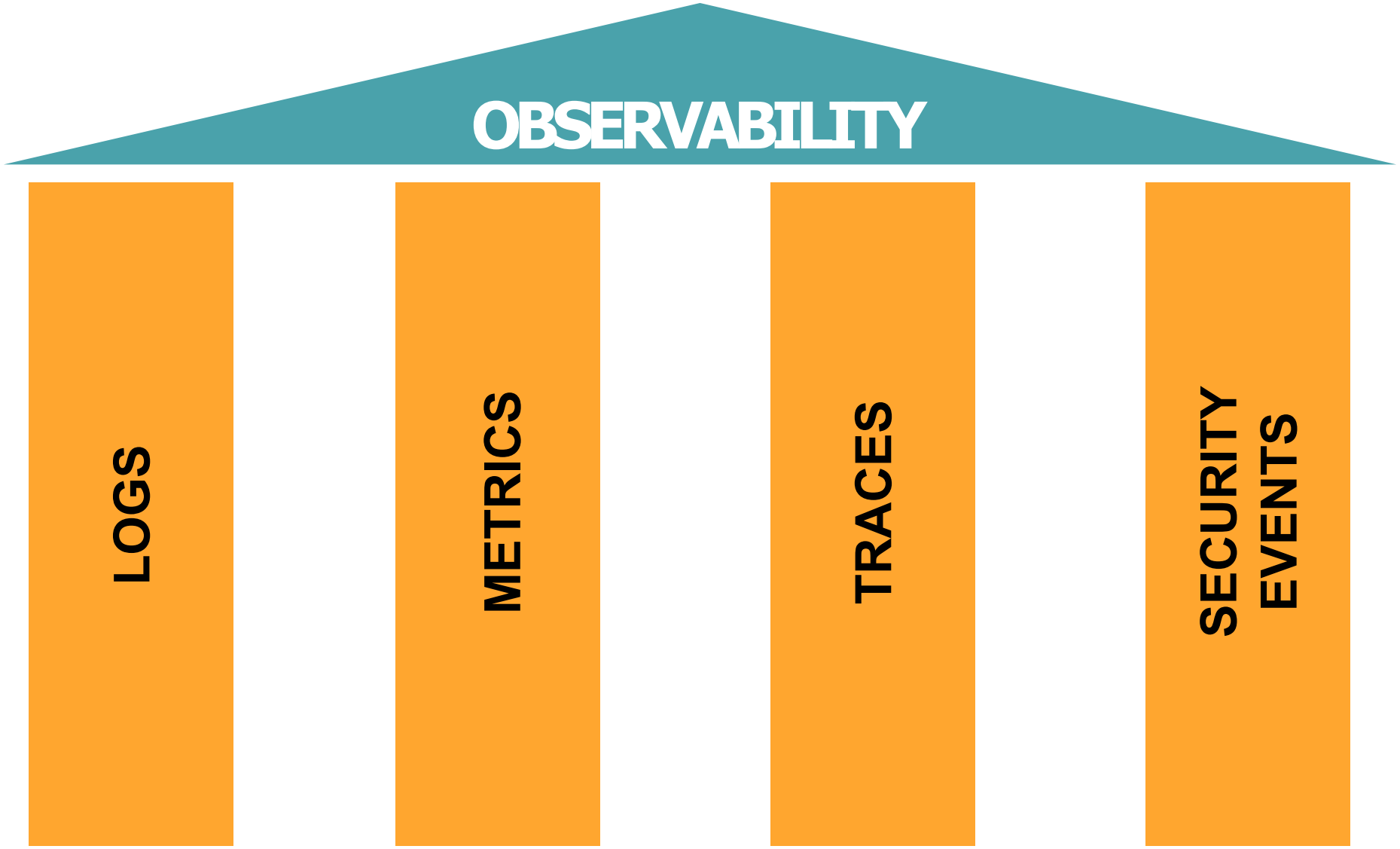
**But it increases complexity**




**Mike Polisky, Splunk**

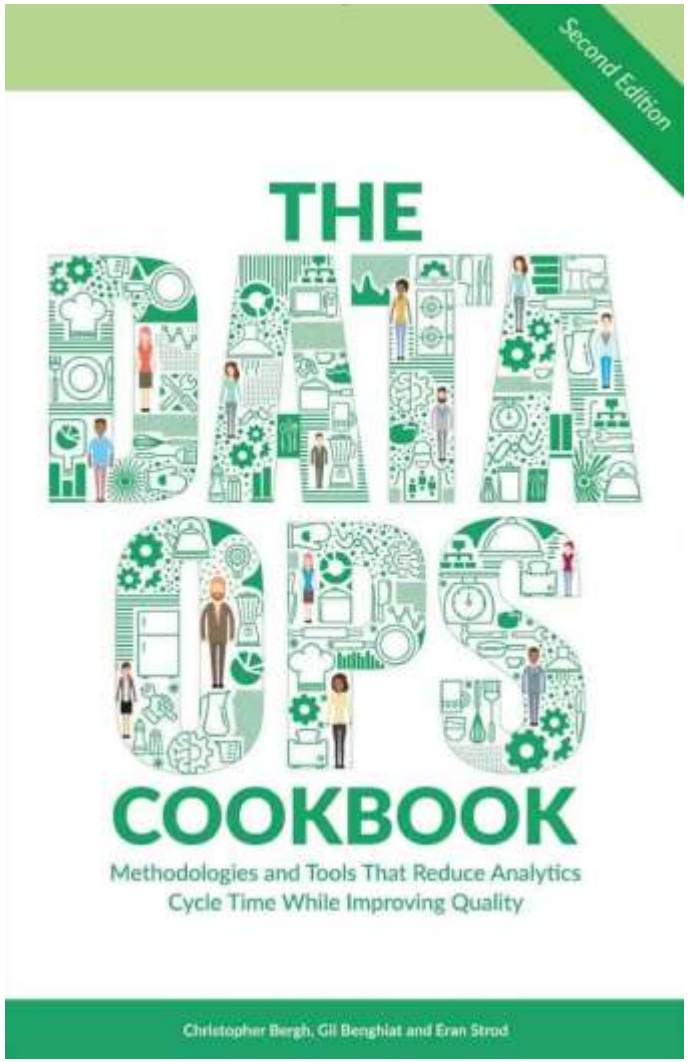
# DevSecOps & Observability

**Four Pillars!**



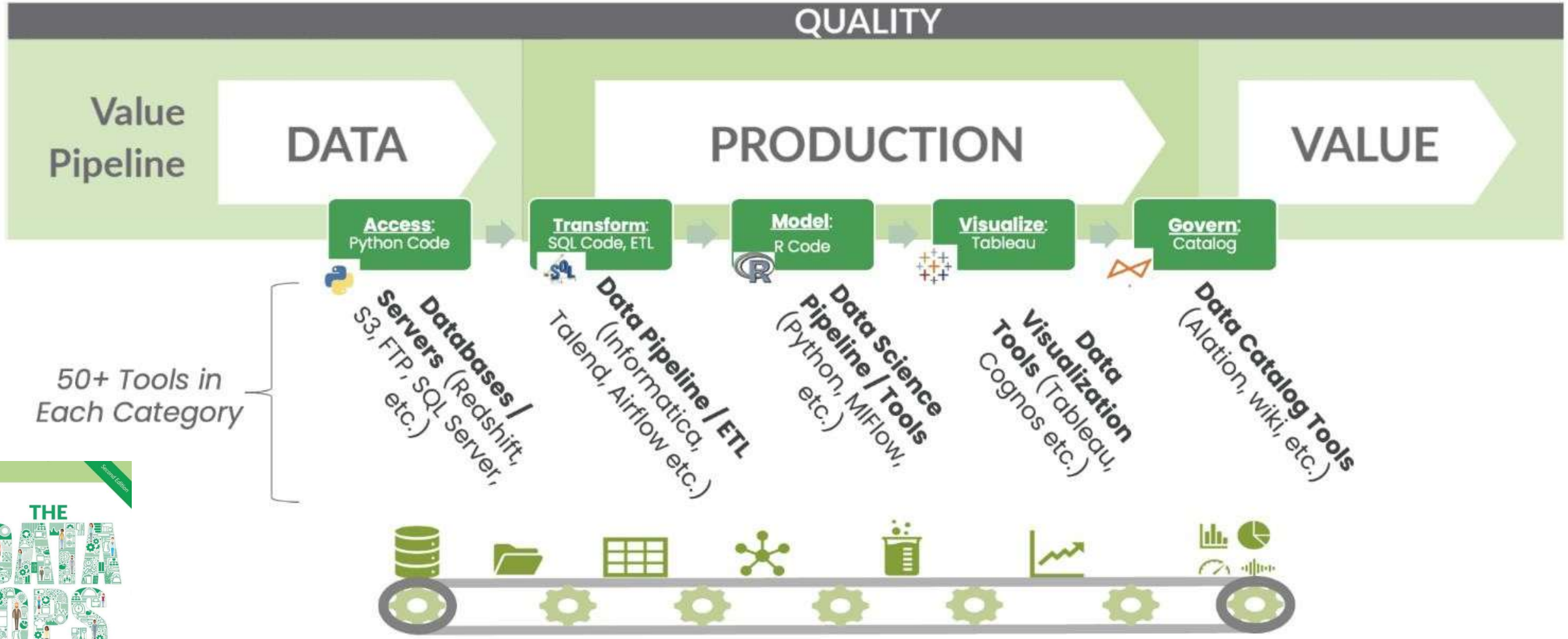
# DataOps & Observability

**The set of technical practices, cultural norms, and architecture that enable low error rates.**



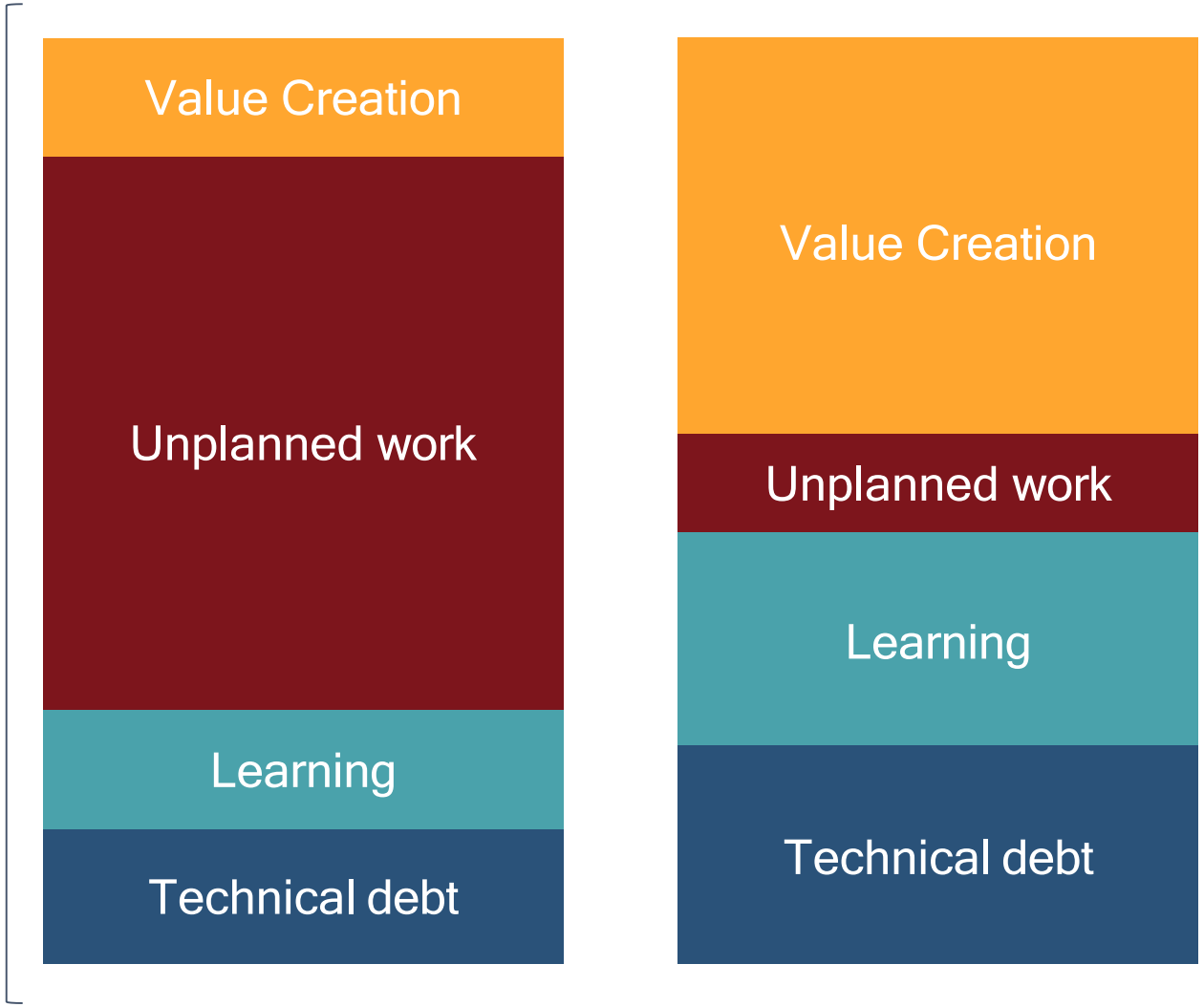

- Avoid manual tests
- Data operations is manufacturing
- Tie tests to alerts
- Focus on the process
- Errors are a huge team productivity drain
- Find errors before your customers do
- Data integrity over quality testing
- Run tests in preprod and live

# The Data Factory



# The Cost of Unplanned Work

What the team spends their time doing

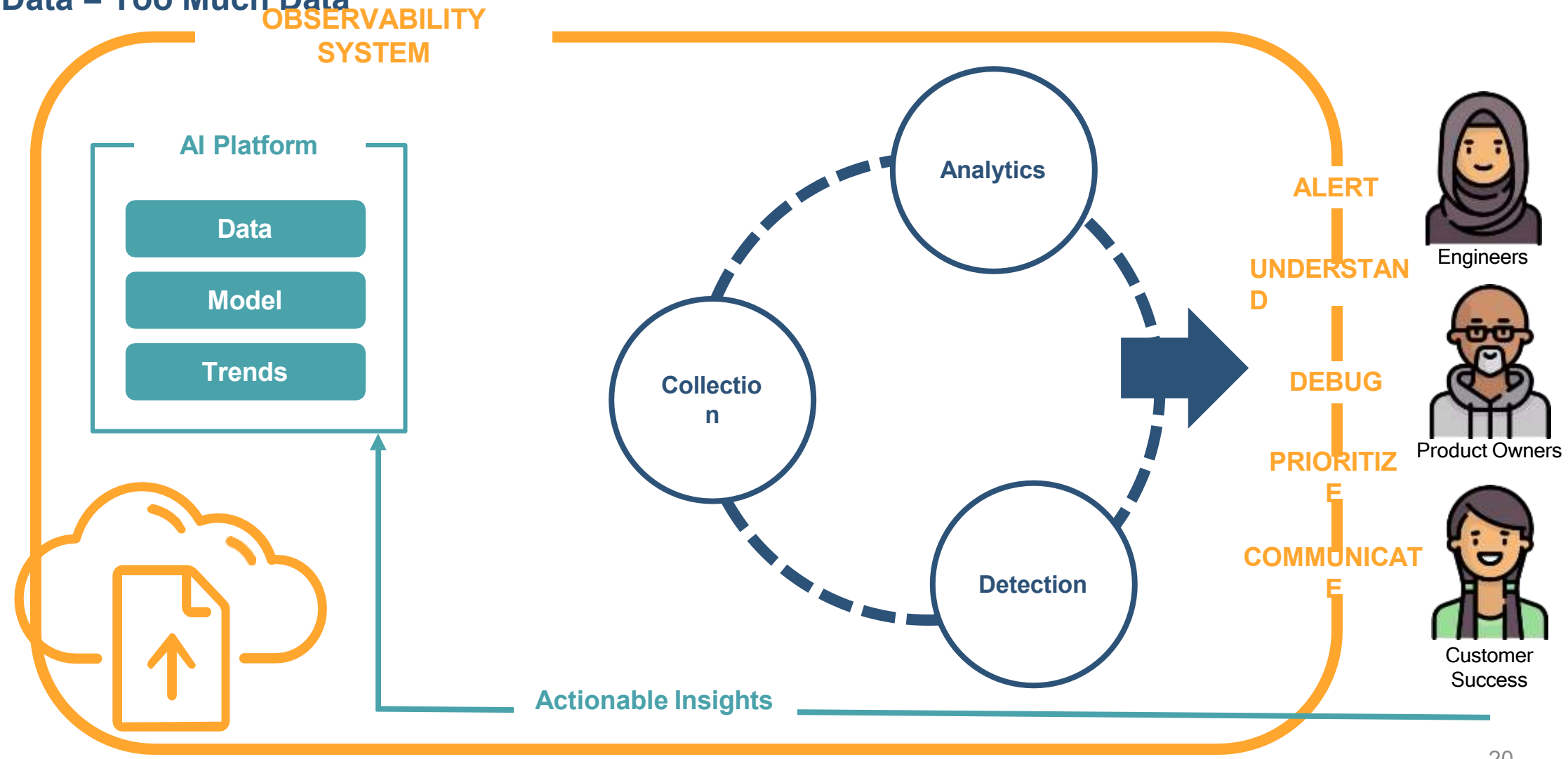



Without observability

With observability

# AI and Observability

**Big Data = Too Much Data**



# Cardinality Matters

**High-cardinality data is the most useful for debugging**

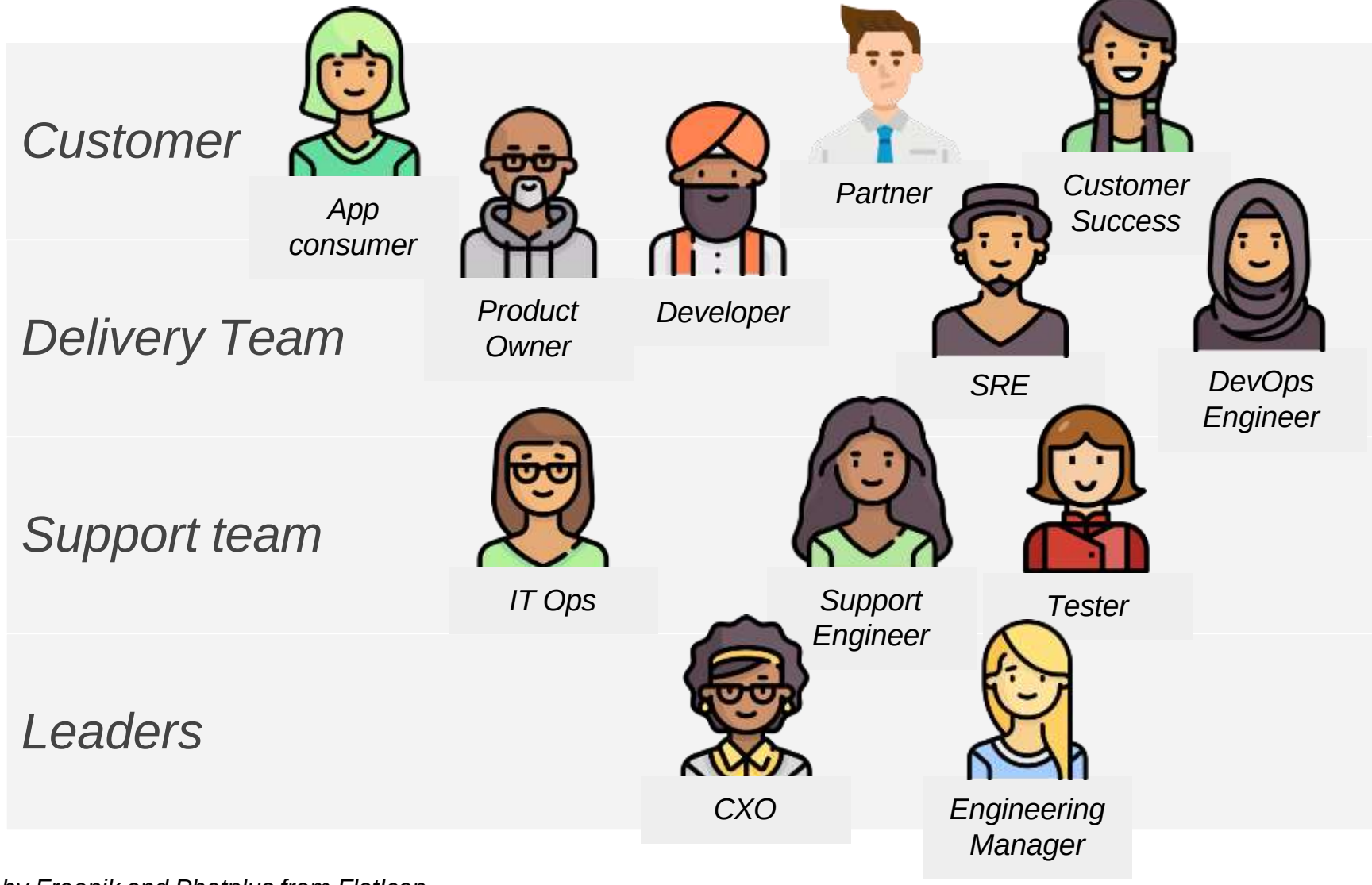
| LOW | HIGH |
| --- | --- |
| Database column has lots of duplicate values in a data set | Database column has a large percentage of completely unique values |

| | |
| --- | --- |
| *User ID* | *012345* |
| *First Name* | *Helen* |
| *Last Name* | *Beal* |
| *Gender* | *Female* |
| *Species* | *Human* |

Highest possible cardinality

Lowest possible cardinality

# Observability Personas




*Icons by Freepik and Phatplus from FlatIcon*

# DevOps Persona

## How Observability Helps Organizations Adopt CICD Practices

- Making CICD metrics available for data-driven conversations builds trust within and between teams and across the organization for continued DevOps investment
- Reducing the risks associated with test and release failures drives test automation coverage
- Using ODD drives developer behaviors based on feedback and the wisdom of production
- providing insights into the flow of work and feedback on value realization, alongside value stream management in particular
- Building resilience as the CICD pipeline and DevOps toolchain become business critical infrastructure

*Icons by Freepik and Phatplus from FlatIcon*

# ITOps Persona

## How Observability Helps IT Operations Evolve (AIOps)

More time for value experimentation

Step 5 — Add more automation for self-learning systems

Step 4 — Use chaos engineering for antifragility

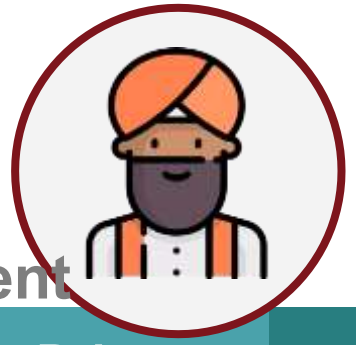
Step 3 — Pay down technical debt for increased stability

Step 2 — Automate toil using AI insights

Step 1 — Reduce MTTR through noise reduction

*Icons by Freepik and Phatplus from FlatIcon*

# The Developer Persona



## Observability Driven Development: X-Driven Development

| Test-Driven | Behavior-Driven | Hypothesis-Driven | Impact-Driven | Observability-Driven |
|---|---|---|---|---|
| TDD | BDD | HDD | IDD | ODD |
| A software development process relying on software requirements being converted to test cases before software is fully developed, and tracking all software development by repeatedly testing the software against all test cases. This is as opposed to software being developed first and test cases created later. | An agile software development process that encourages collaboration among developers, quality assurance testers, and customer representatives in a software project. It encourages teams to use conversation and concrete examples to formalize a shared understanding of how the application should behave. | Hypothesis-driven development is a prototype methodology that allows product designers to develop, test, and rebuild a product until it's acceptable by the users. It is an iterative measure that explores assumptions defined during the project and attempts to validate it with users' feedbacks. | **EMERGING**<br><br>Takes small steps towards achieving both impact and vision. Impact Driven Development balances the development of a vision with creating real impact for users. It makes sense that the first phase of your product development should involve some users. | **EMERGING**<br><br>Adds another layer to software development by encouraging the development team to think about the application availability and uptime throughout their development process and similar to unit-testing development, wrap their code with more verbose logging, metrics and KPIs |

# SRE Persona

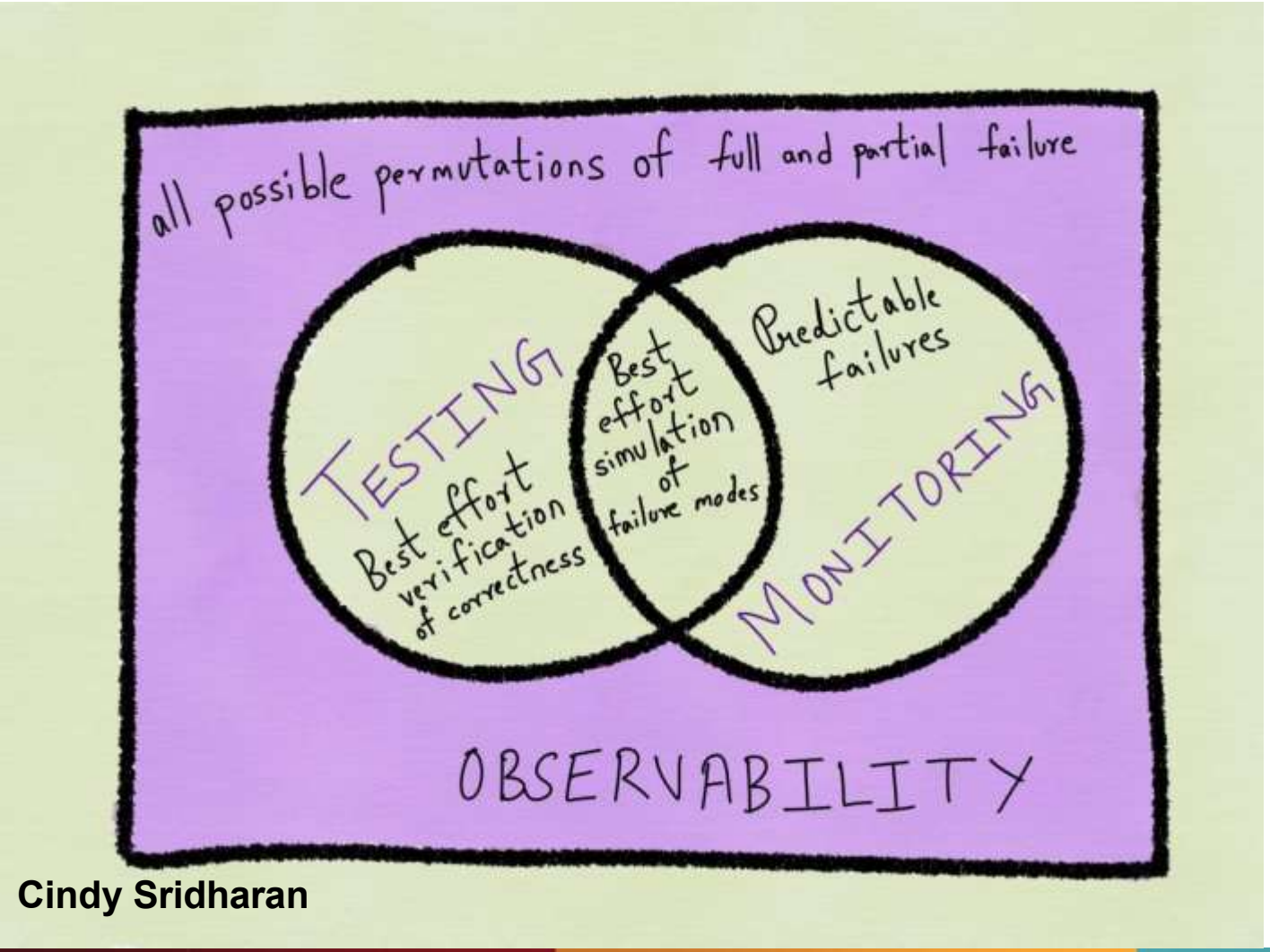
## How Observability Supports SRE's Goals

- Reducing the toil associated with incident management – particularly around cause analysis – improving uptime and MTTR
- Providing a platform for inspecting and adapting according to SLOs and ultimately improving teams' ability to meet them
- Offering a potential solution to improve when SLOs are not met and error budgets are over-spent
- Relieving team cognitive load when dealing with vast amounts of data – reducing burnout
- Releasing humans and teams from toil, improving productivity, innovation and the flow and delivery of value
- Supporting multifunctional, autonomous teams and the "we build it, we own it" DevOps mantra
- Completing the value stream cycle by providing insights around value outcomes that can be fed back into the innovation phase

*Icons by Freepik and Phatplus from FlatIcon*

# QA and Testing Persona

## Observability is Testability




**Cindy Sridharan**

"As a by-product, TDD identifies the aspects/parts of the system can be certainly observed."

*Venkatesh-Prasad Ranganath*

# CIO and Leaders

## Every business is a technology business

## A radically different approach is needed


of CIOs say their organization will lose their competitive edge if IT is unable to spend less time "keeping the lights on"

**CIOs say the most critical capabilities they need to manage the performance of their digital services are:**

| | |
|---|---|
| Visibility across the entire cloud and IT environment | 55% |
| Automation to reduce the need for manual instrumentation and intervention | 54% |
| Ability to identify the relationship between IT performance and business metrics (i.e., conversions) | 51% |
| Ease of use and rapid time to value from performance management solutions | 51% |
| Ability to correlate alerts and reduce noise so IT teams can focus on what matters to the business | 45% |
| The ability to capture metrics, logs, and traces in a single platform | 41% |

# Observability and Value Stream Management

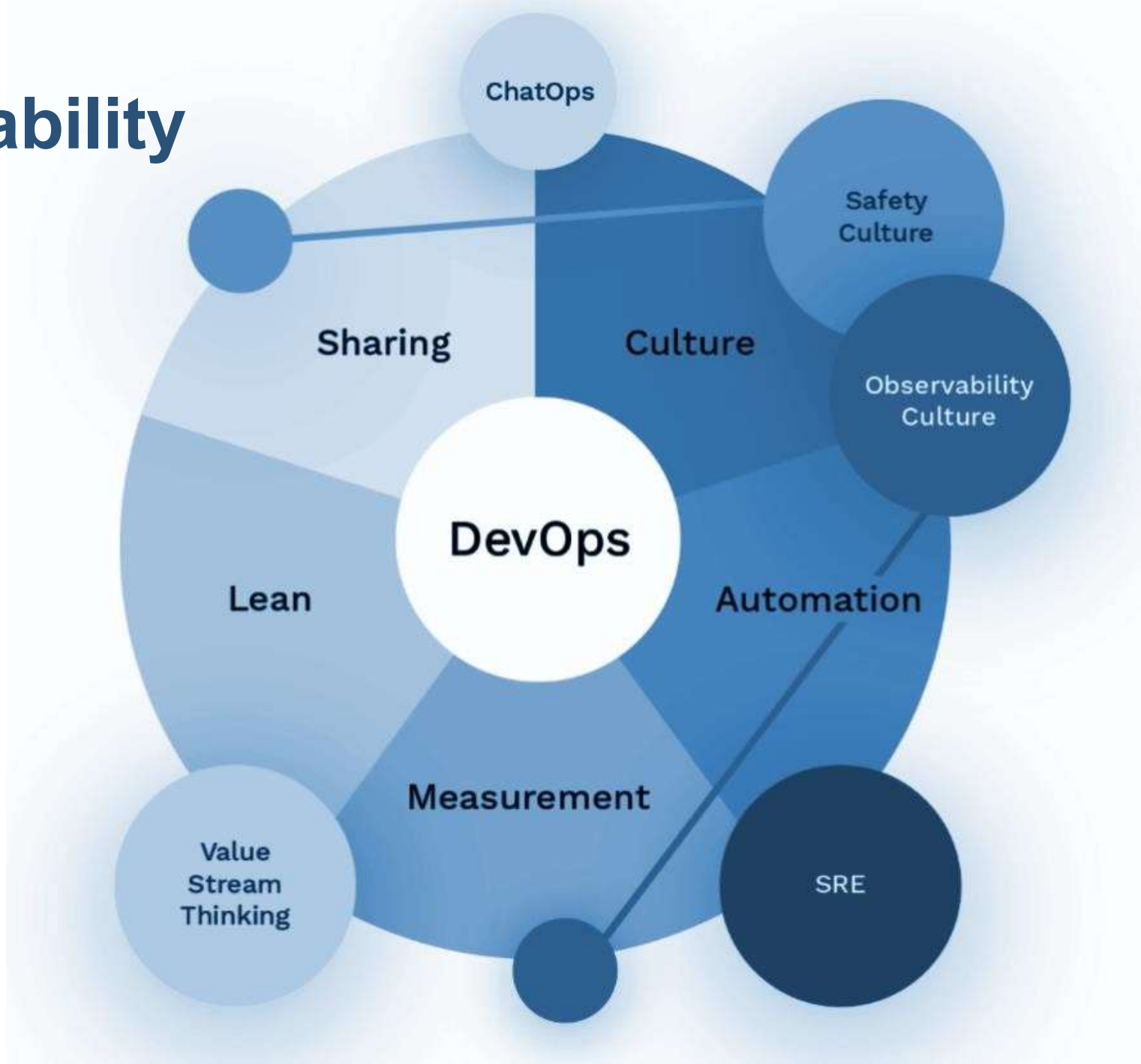
## How Observability Helps Organizations Adopt VSM Practices

- Shortening MTTR means more time for innovation and more time to spend on adapting ways of working to optimize flow and more value outcomes delivered to customers
- Reducing the risk and costs associated with outages directly improving a product teams Profit and Loss (P&L) or cost:value ratio
- Improving customer delight which can be understood by metrics such as Net Promoter Score (NPS) and referrals
- Using observability throughout the value stream, so in pre-production and across the DevOps toolchain, means less defects and more predictability in production
- Making the value stream visible and making value measureable

# Observability Gaps

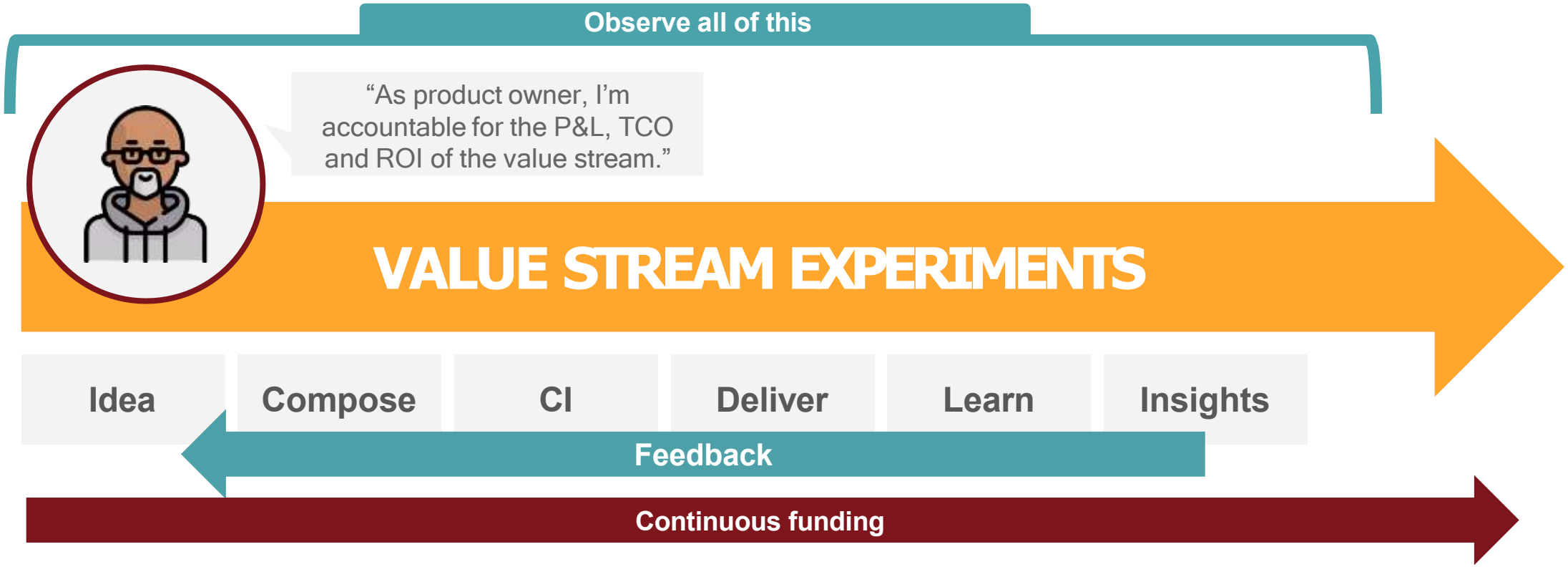
## UNKNOWN UNKNOWNS

# Observability Culture

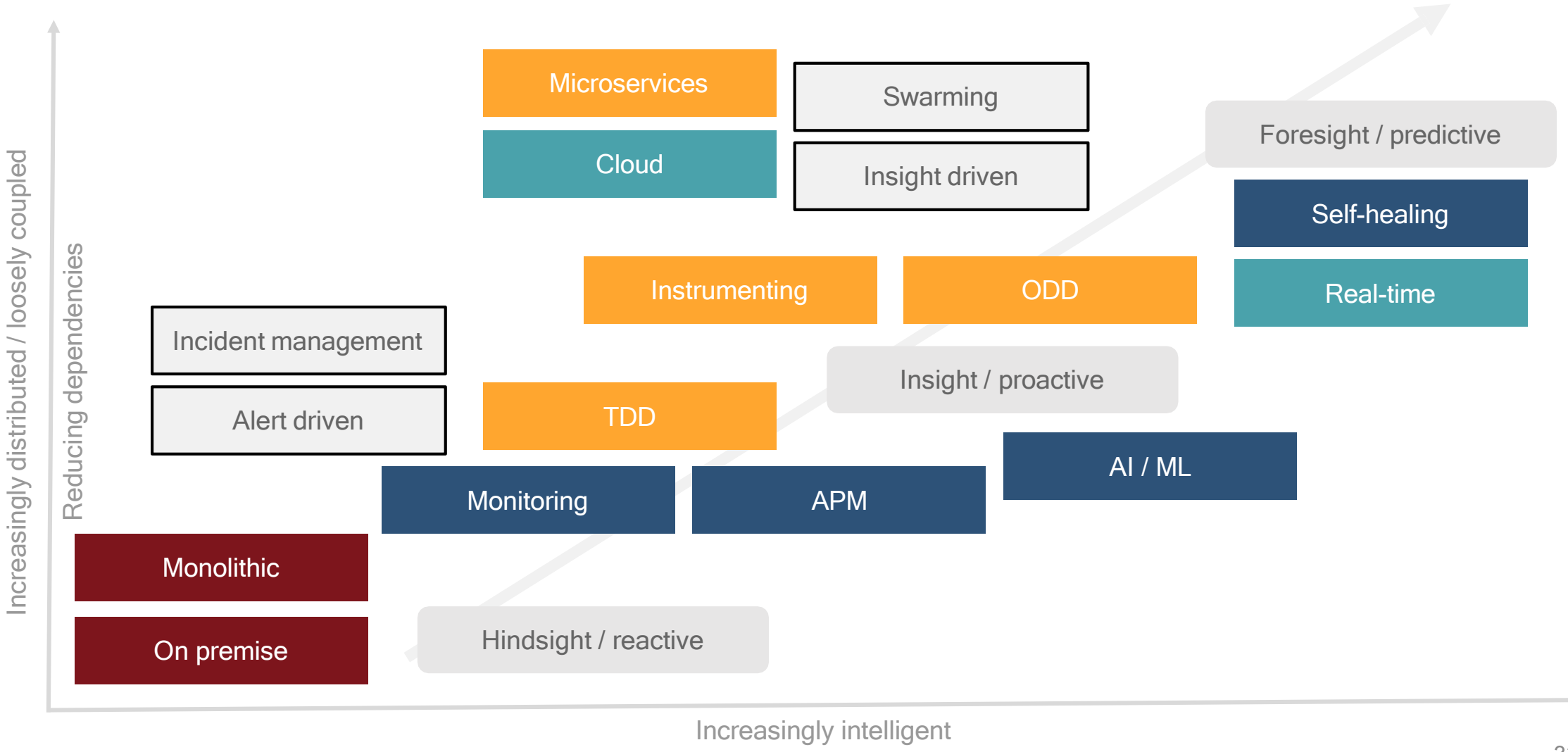


# Observability and Funding

**The value stream or product owner is a mini-CEO**



# Observability Capability Model



# Where to Get Started

| | |
|---|---|
| 1 | Set your vision: define your goals as “just enough” observability |
| 2 | Understand your priorities |
| 3 | Extract data from the digital services you don’t own (realtime) |
| 4 | Instrument your own digital products and services |

# Instrumentation Principles

1. Generate unique request IDs
2. Generate one event per service/hop/query/etc
3. Wrap any call out to any other service/data store as a timing event
4. Collect lots of context
5. Add redundant information
6. Add two fields for errors
7. Opt for wider events (more fields)
8. Don’t be afraid to add fields that don’t exist in wider contexts
9. Spend time thinking about field names
10. Add units to field names

# What Fields?

## WHO'S TALKING TO YOUR SERVICE?

- IP address
- load balancer
- proxy
- User ID
- email address
- user_agent
- SDK version

## WHAT ARE THEY ASKING?

- URL request
- Handler
- HTTP headers
- Accept/refuse
- Pass garbage?
- Batched?
- Zipped?
- Object ID?

## HOW DID THE SERVICE RESPOND?

- Response time
- Service calls
- Metadata
- Success
- Object ID?
- Host name
- Container ID
- Build ID
- K8s pod
- AWS cluster

# Sampling

## Observability at scale

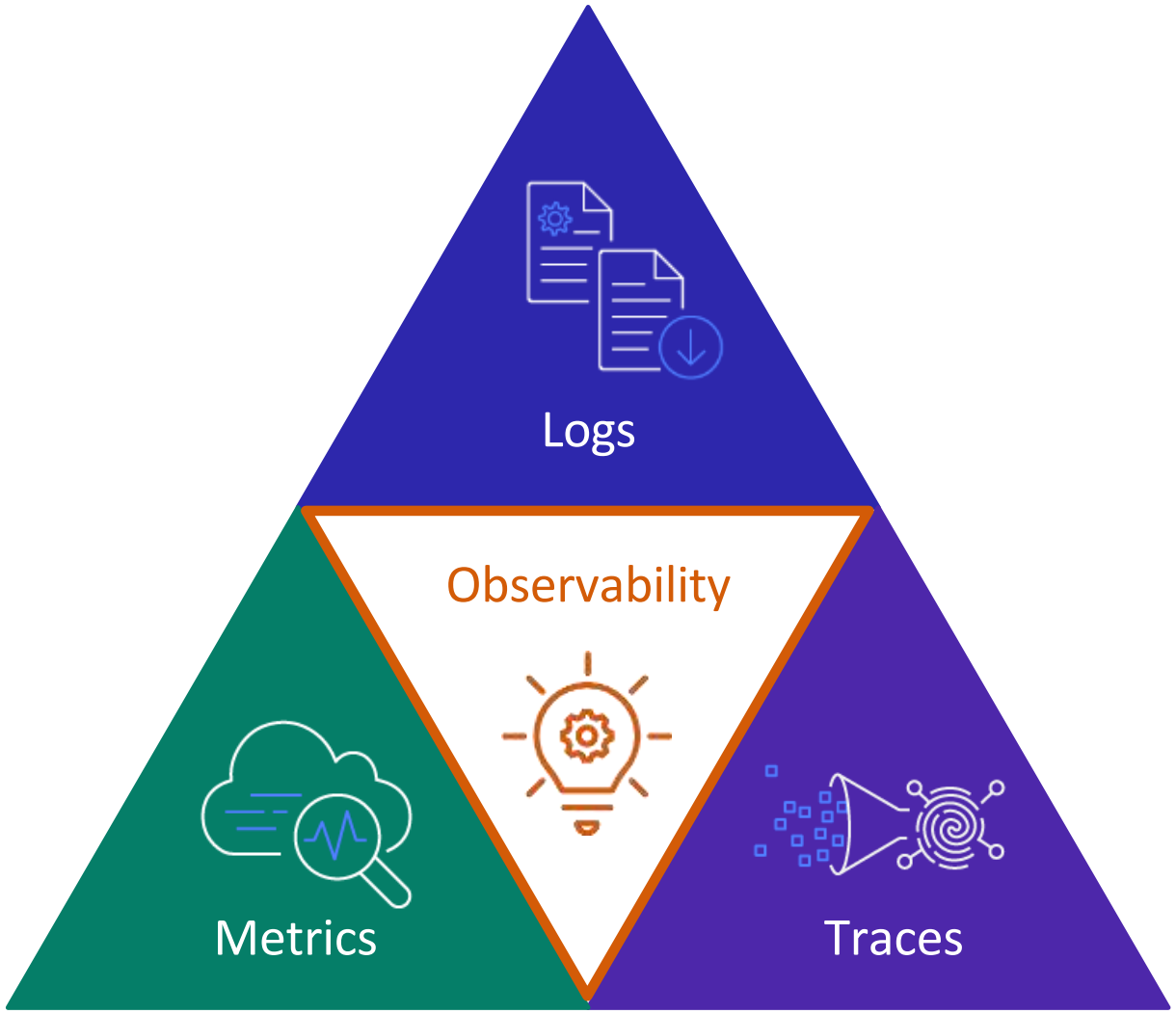
| Constant sampling | Dynamic sampling | Constant throughput | Constant throughput per key | Average sample rate |
|---|---|---|---|---|
| Submit one event for every n events you wish to represent | Vary the sample rate based on characteristics of the incoming traffic | Specify the maximum number of events per time period you want to send | Maximum number of events sent per key | *A* given overall sample rate across all traffic to capture more of the infrequent traffic |
| Simple and easy to implement | Tremendous flexibility in how you choose the individual events that will be representative of the entire stream of traffic | With a relatively even split of traffic among your keys and fewer keys than desired throughput rate, great at capping resources | Scales: retain detail per-key as the key space grows | *Good w*hen rare events are more interesting than common events |
| Lack of flexibility | If there are too many different types of traffic, enumerating them all to set a specific sample rate for each can be difficult | Doesn't scale at all | *A*s traffic grows within an individual key, visibility into the details for that key is lost | High-volume traffic is sampled very aggressively, which may not be desirable |

# Implementation Challenges

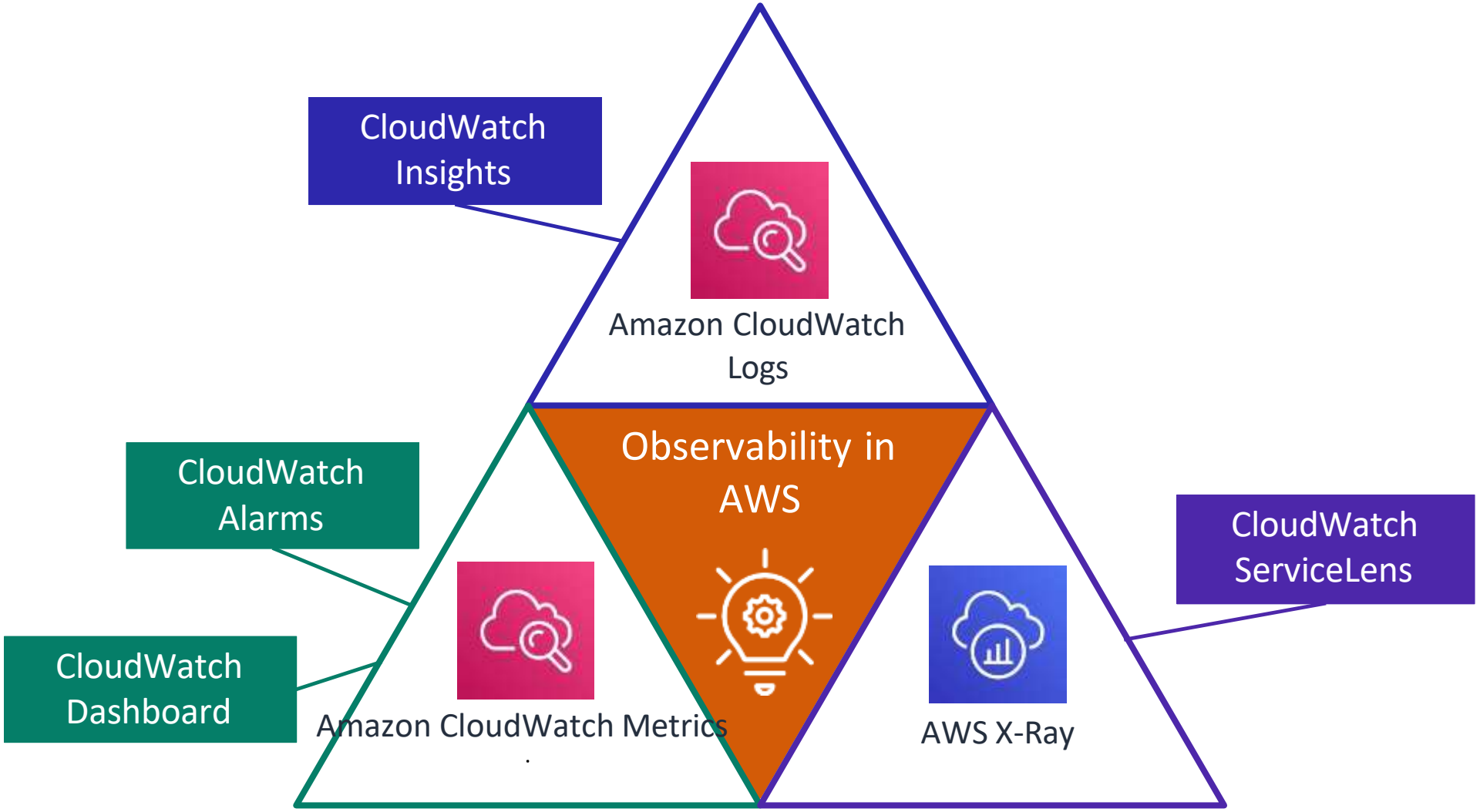
## And how to overcome them

| Challenge | Try This |
|---|---|
| No time | Build time into your sprints, calculate how much time you will save in the future to justify the investment now |
| No skills | Build learning time into your sprints or hackathons, dynamic learning organizations are the most successful - justify not building cultural debt |
| Other priorities | Work with the PO/business to limit WIP and justify through future increases in capacity for innovation |
| No tools | Practice ODD and build instrumentation in |
| Technical debt | Build time into sprints and identify technical debt as user stories in the product backlog |
| Wrong architecture | Work with architects to use strangler pattern to reduce dependencies in the monolith and incrementally work towards loosely-coupled |

# Observability is the goal

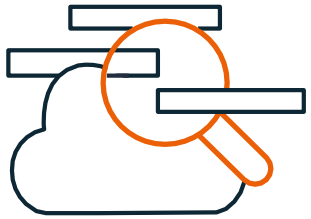


# Observability in AWS



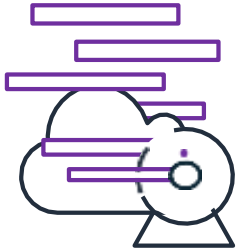
# AWS DevOps Tooling for Monitoring and Logging

Record logs and monitor application and infrastructure performance in near real-time



Cloud and network monitoring with Amazon CloudWatch

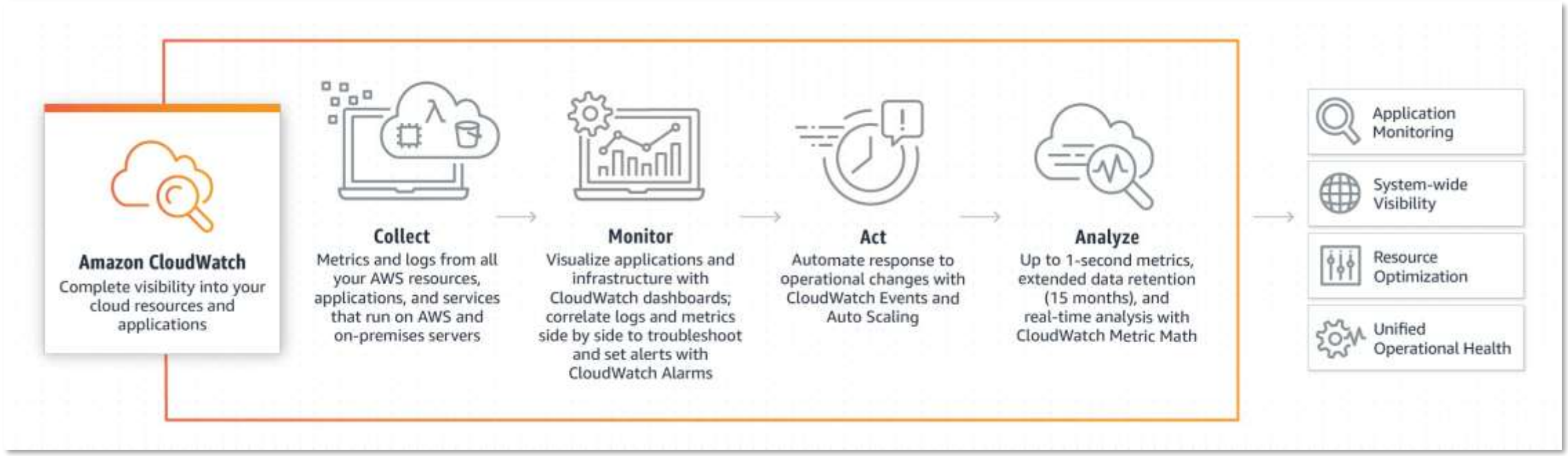
Distributed tracing with AWS X-Ray



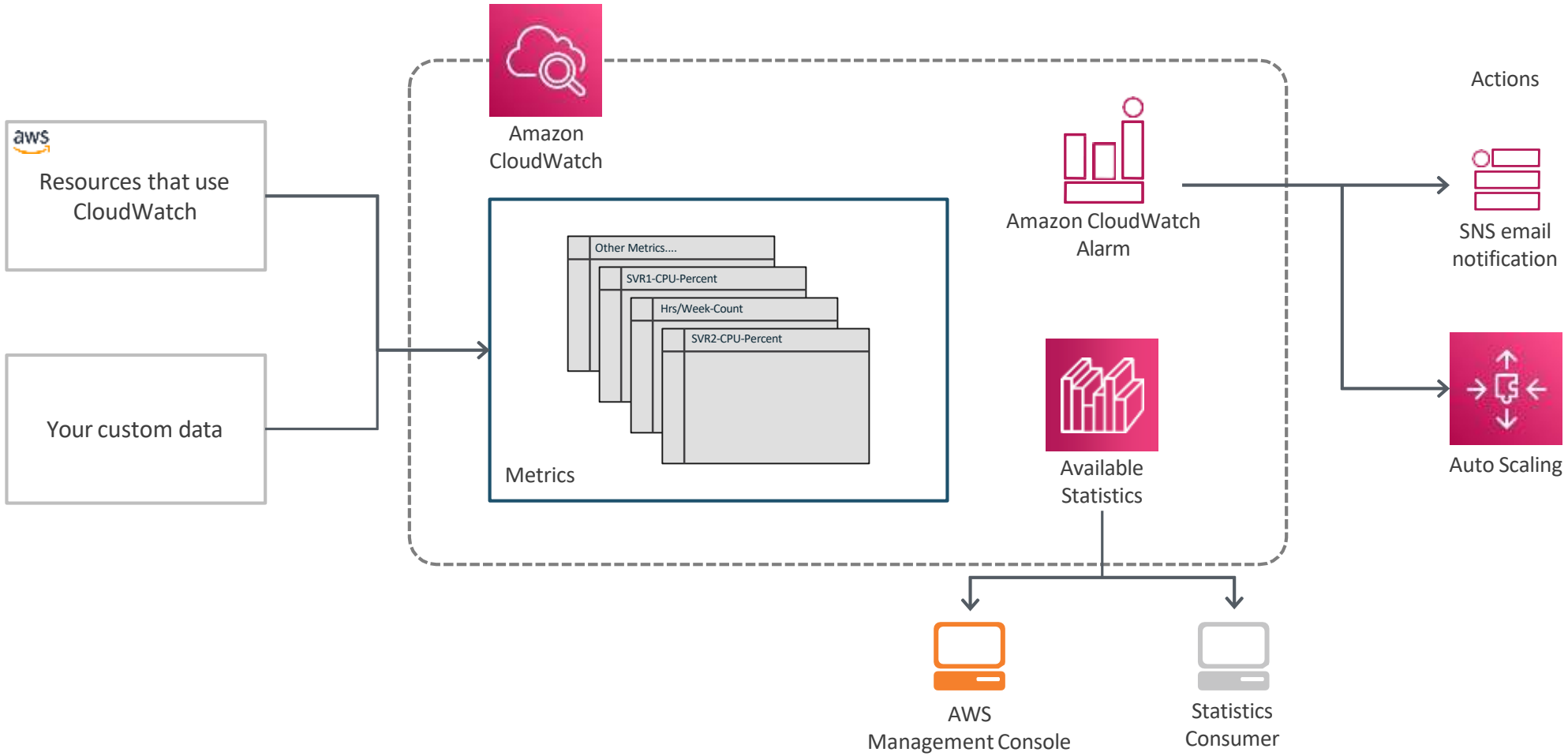
Activity and API usage tracking with AWS CloudTrail

# Amazon CloudWatch Anomaly Detection

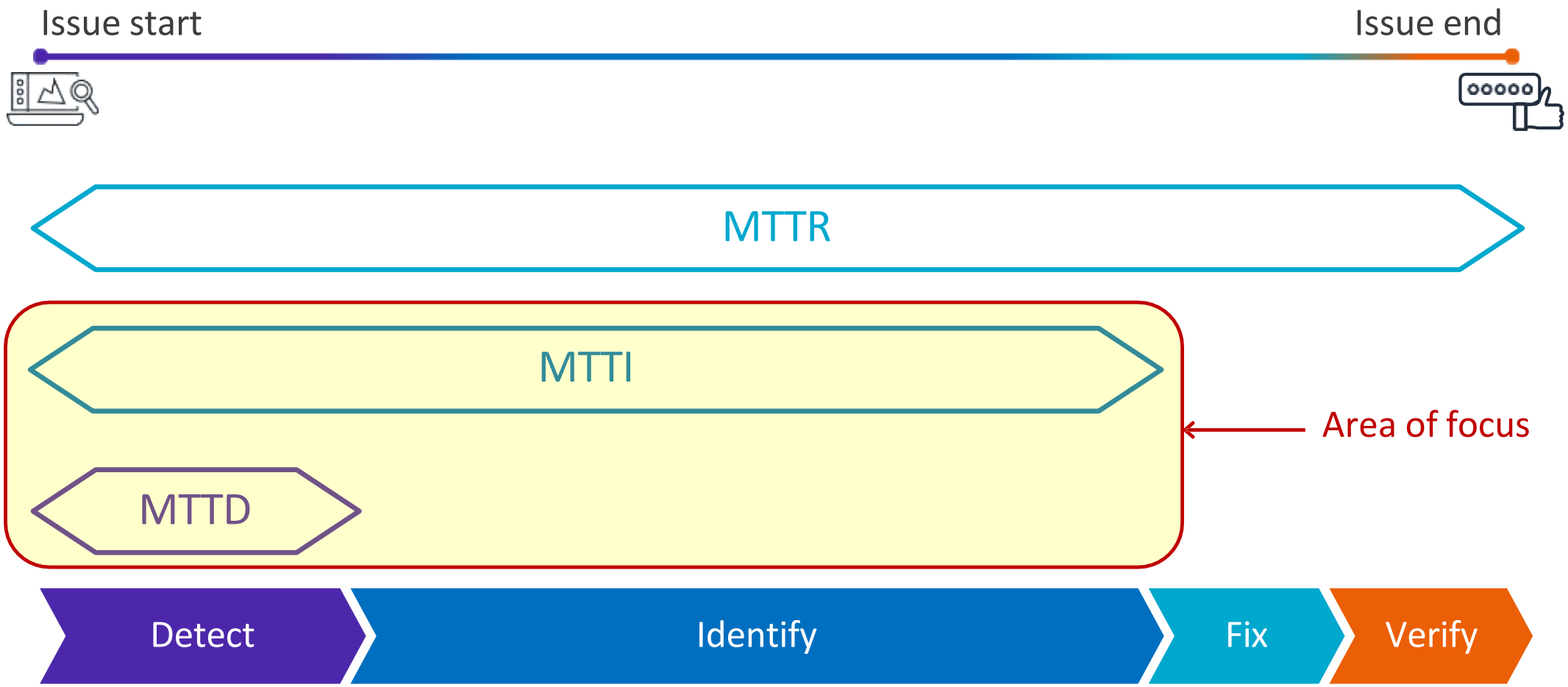
## Continuously analyze telemetry data and identify anomalous behavior



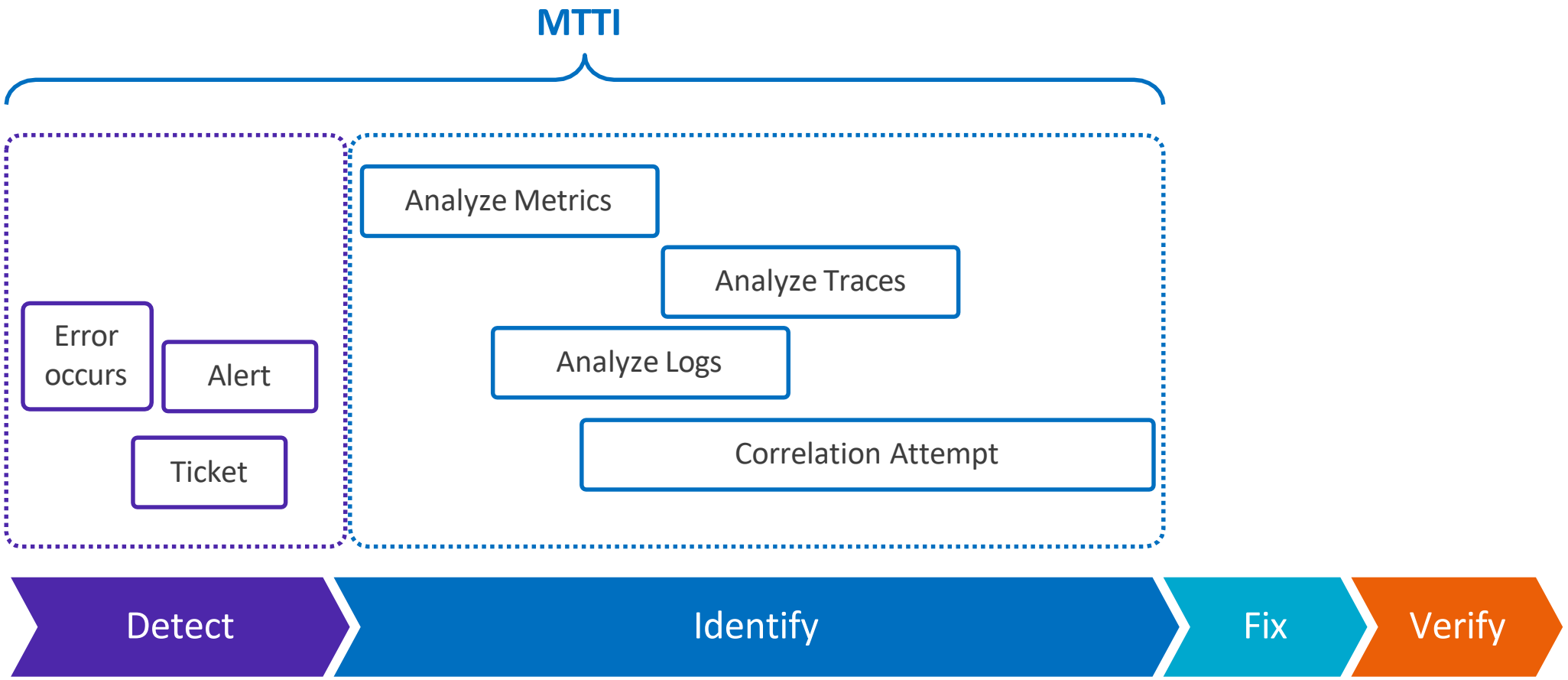
# How Amazon CloudWatch works



# Issue timeline

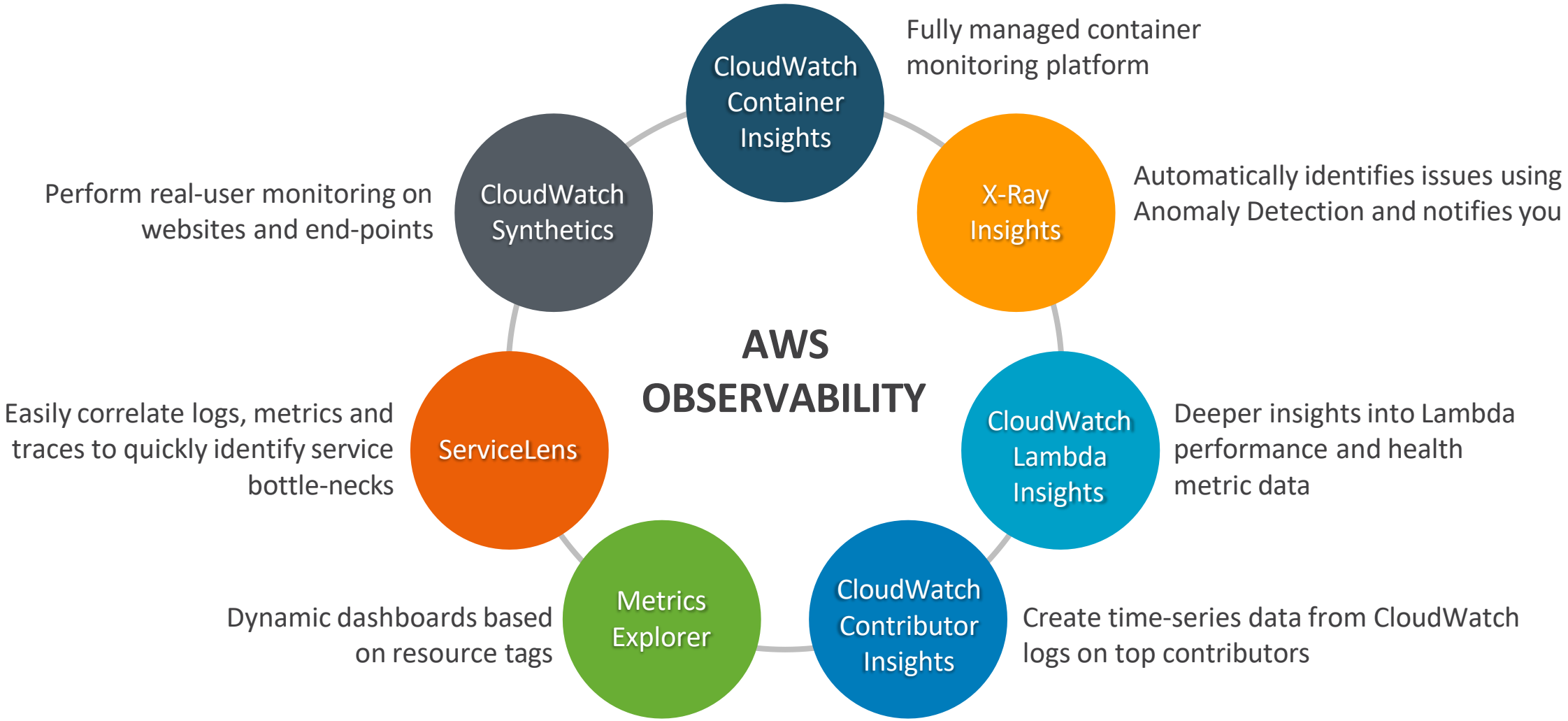


# Typical troubleshooting workflow



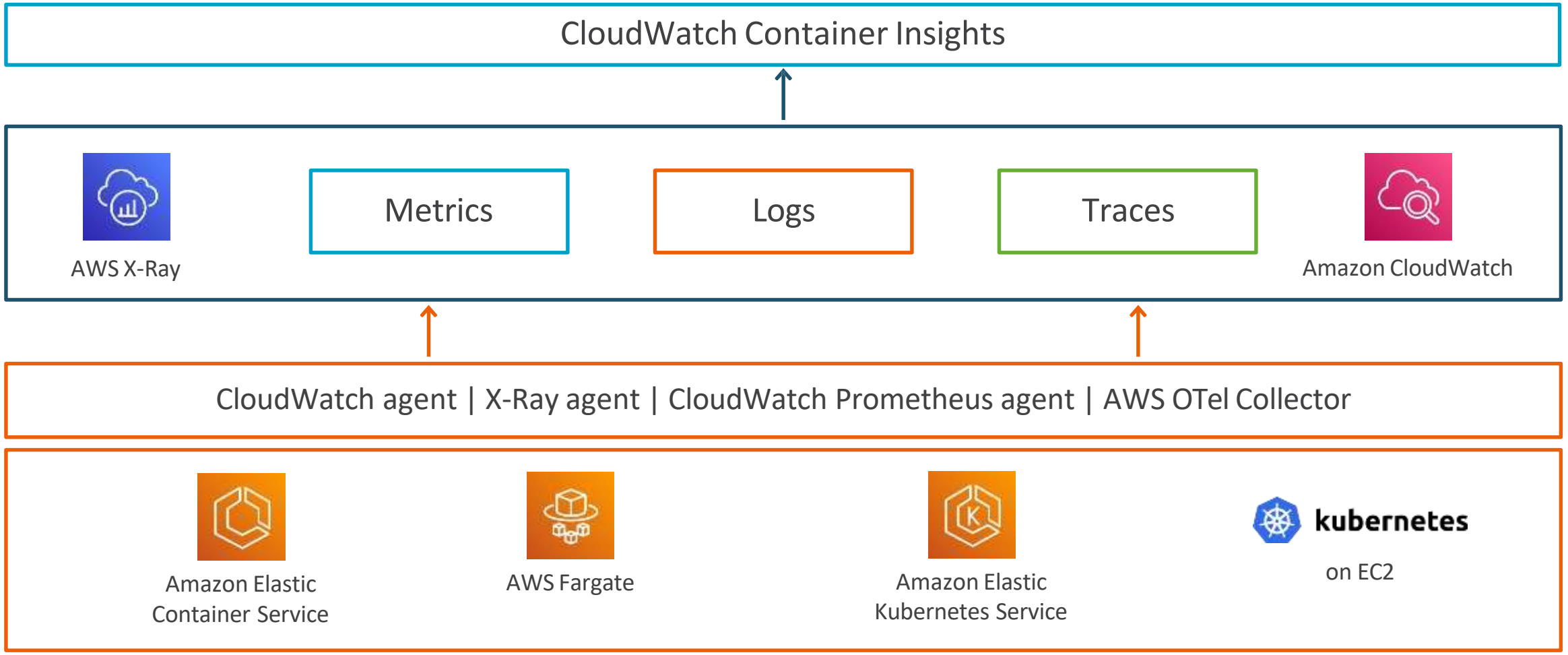
# AWS observability tools

## Infrastructure, application, and synthetic monitoring



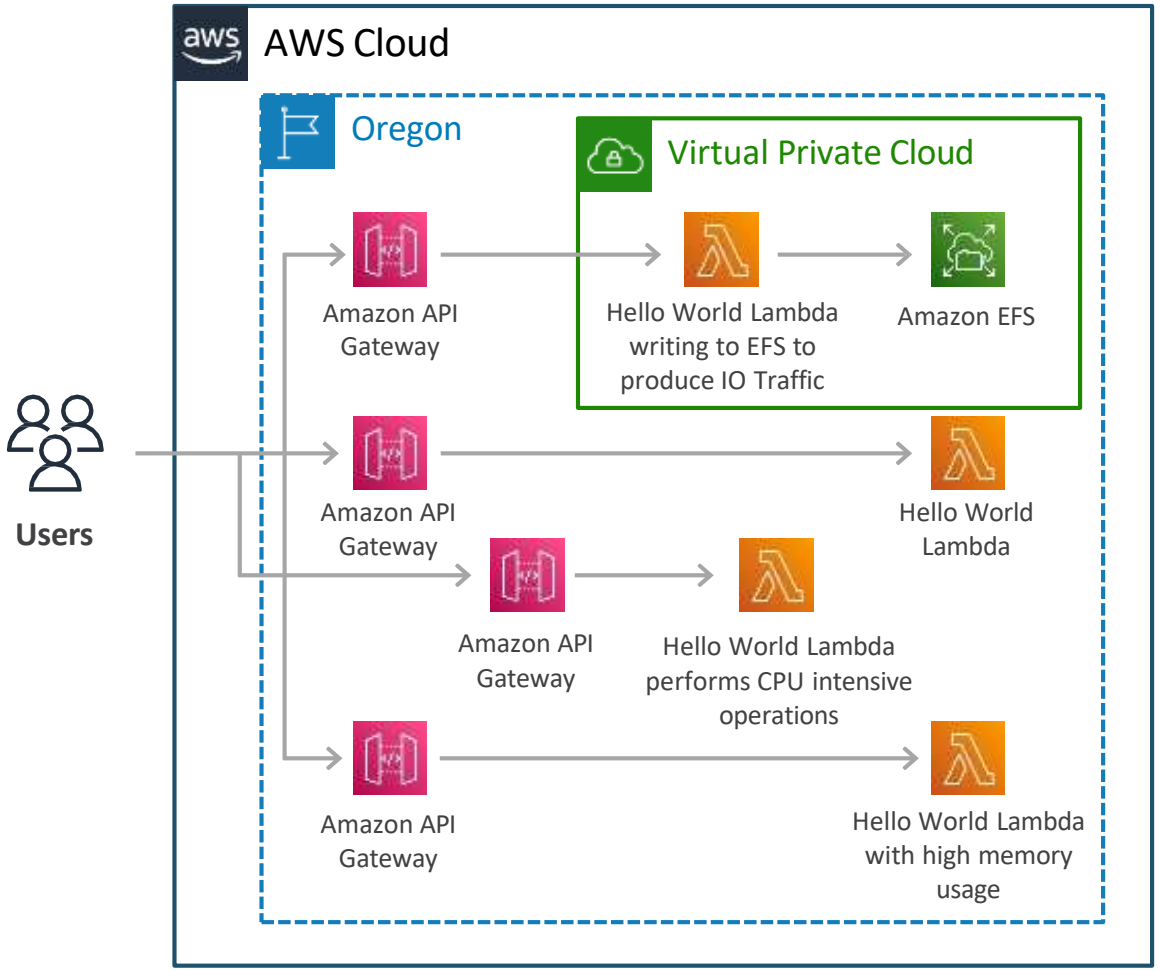
# CloudWatch Container Insights

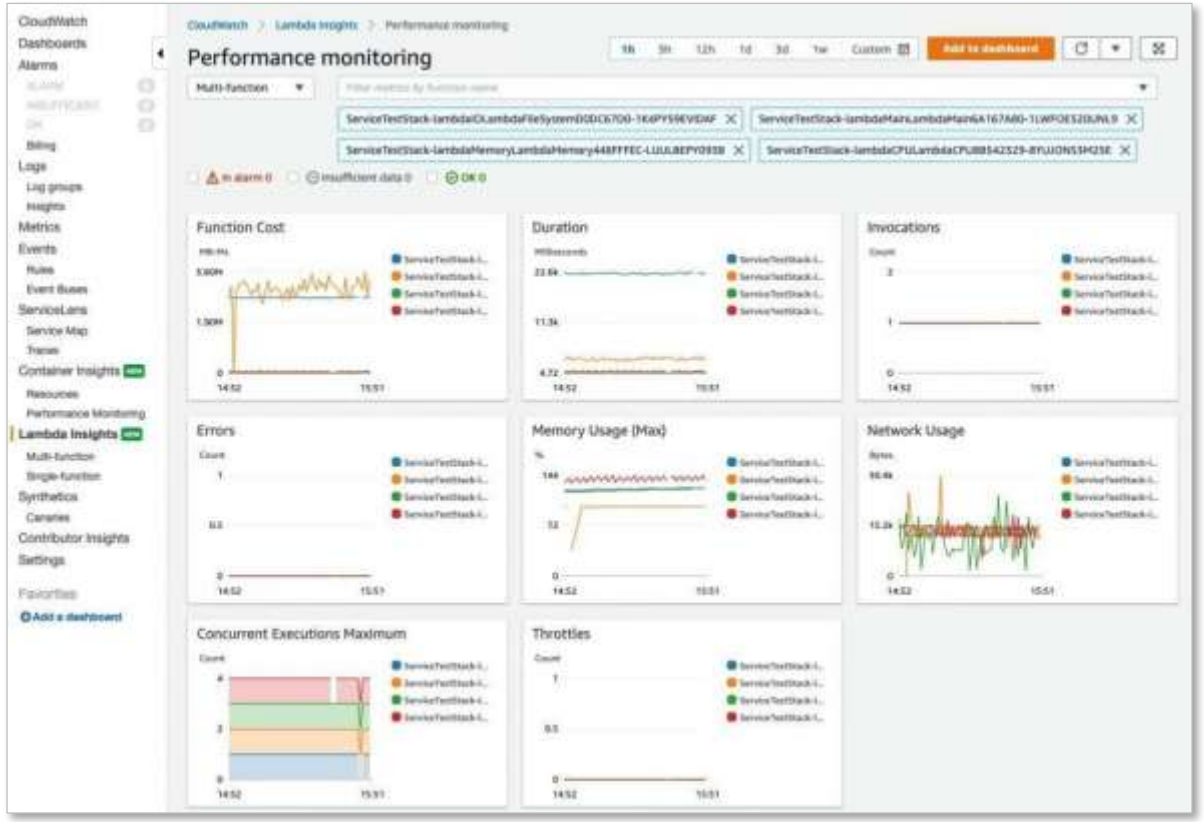
Collect, aggregate, and summarize metrics and logs containerized apps and services



# CloudWatch Lambda Insights

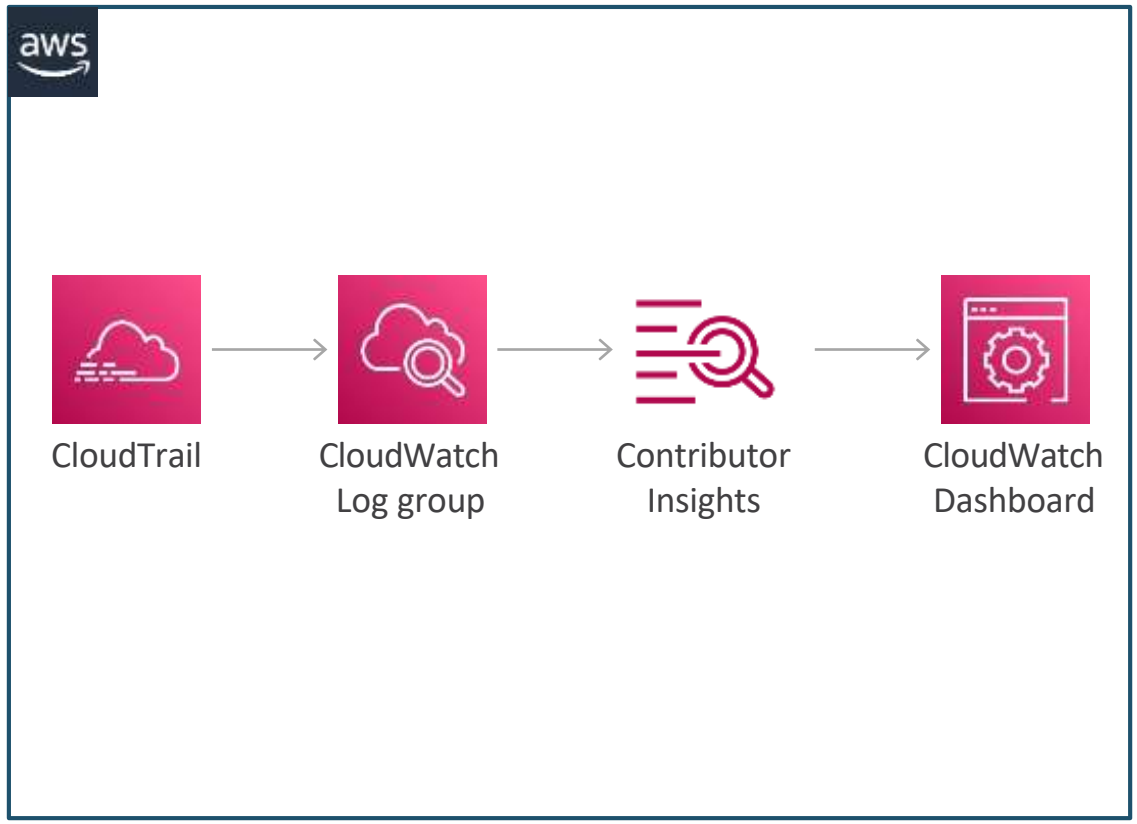
## Monitoring and troubleshooting for serverless apps on AWS Lambda

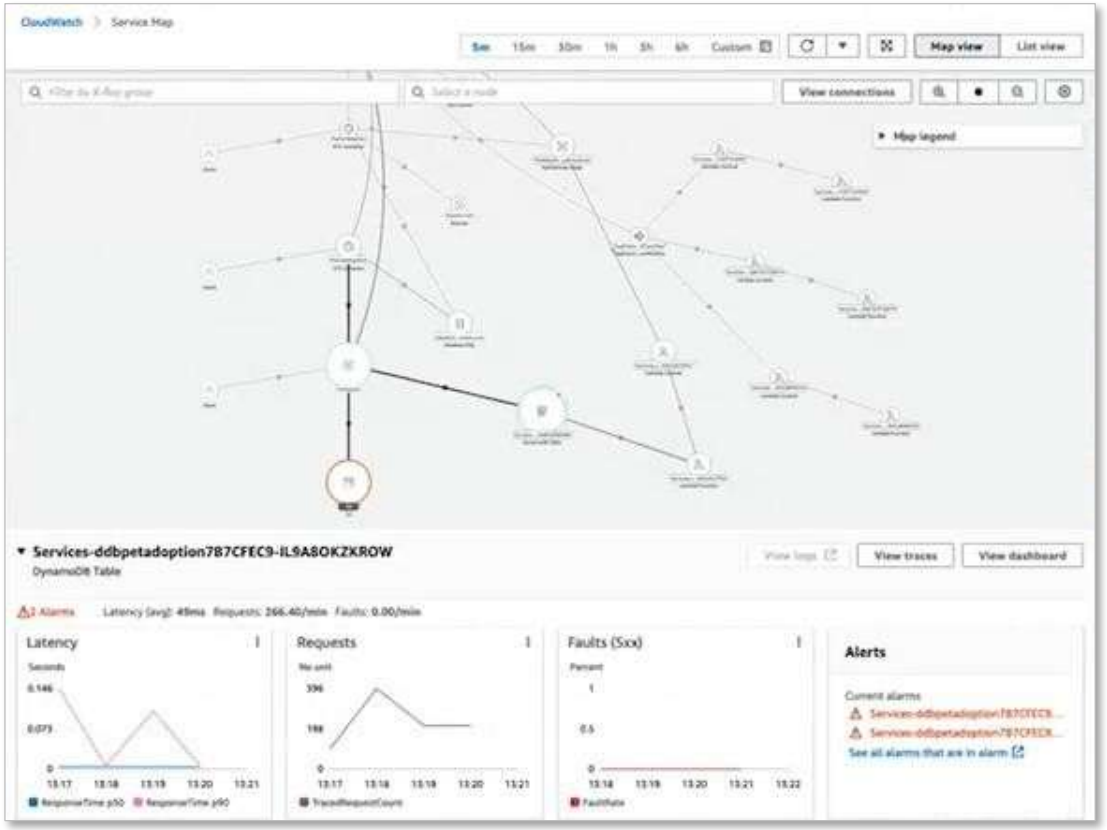


# CloudWatch Contributor Insights

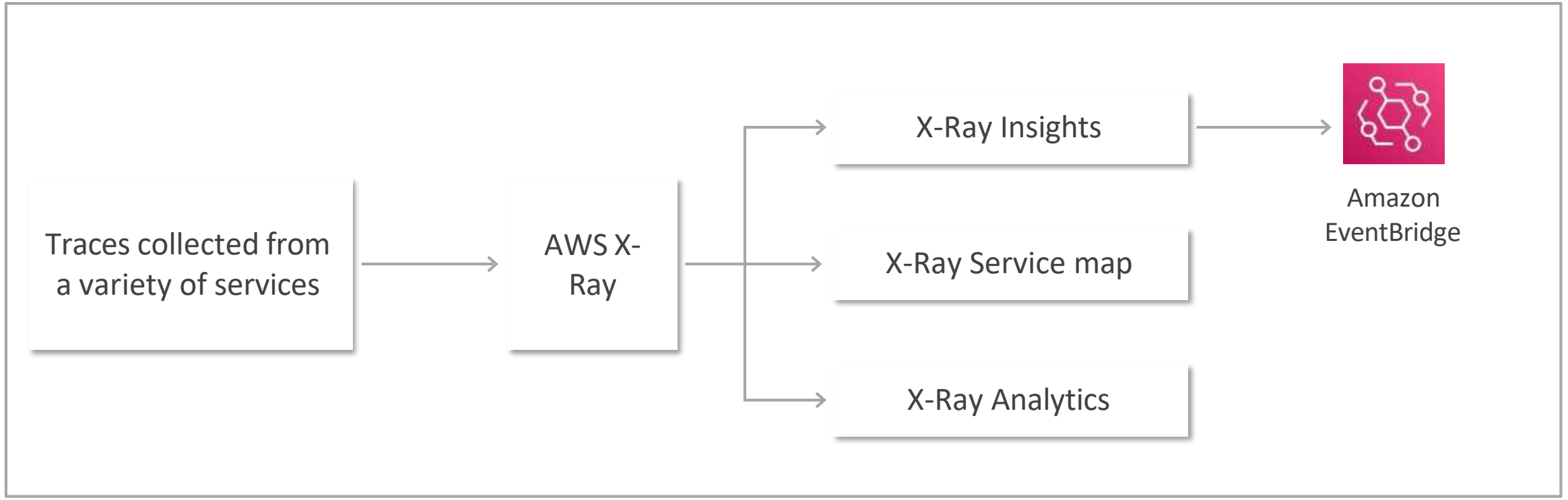
Analyze log data and create time series that display contributor data



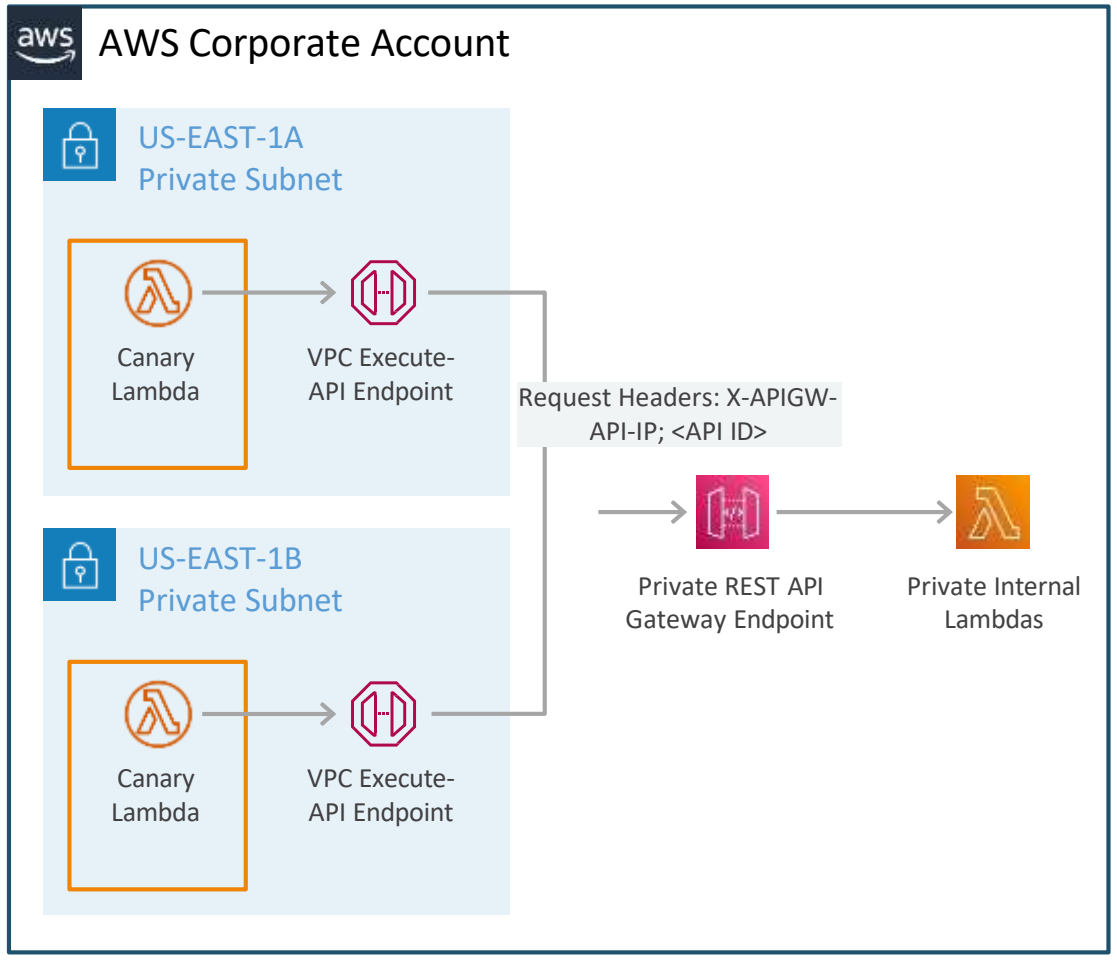
# X-Ray Insights

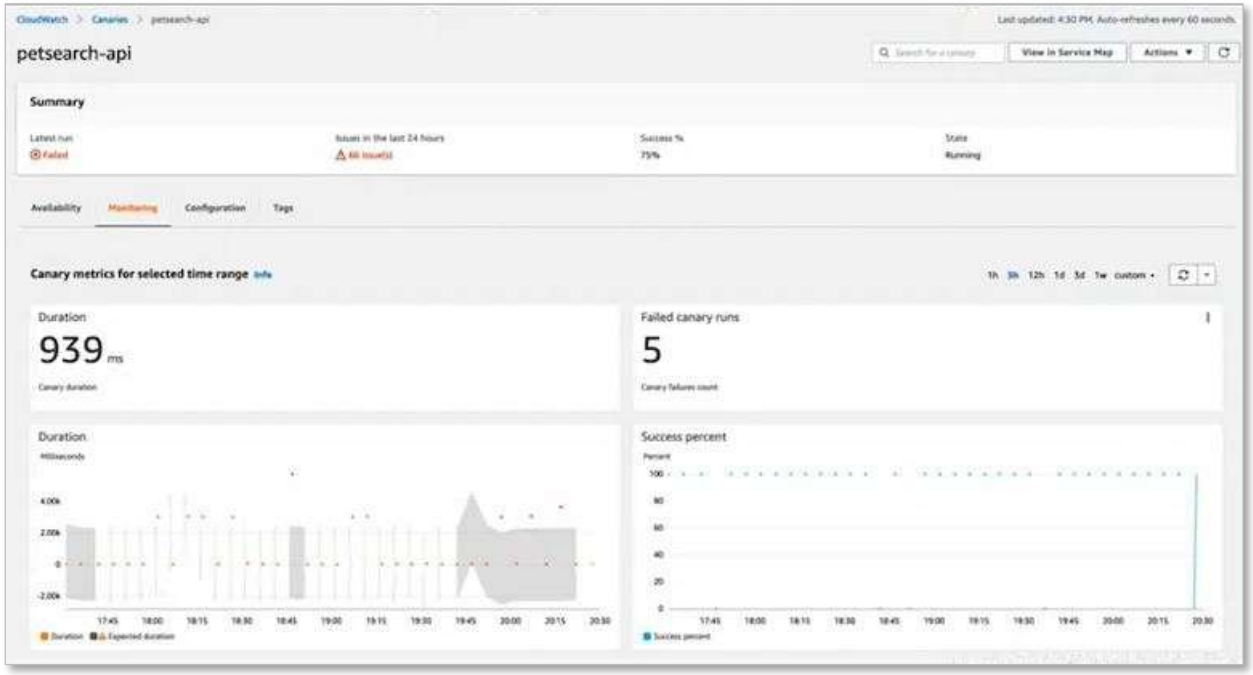
Create actionable insights about application anomalies



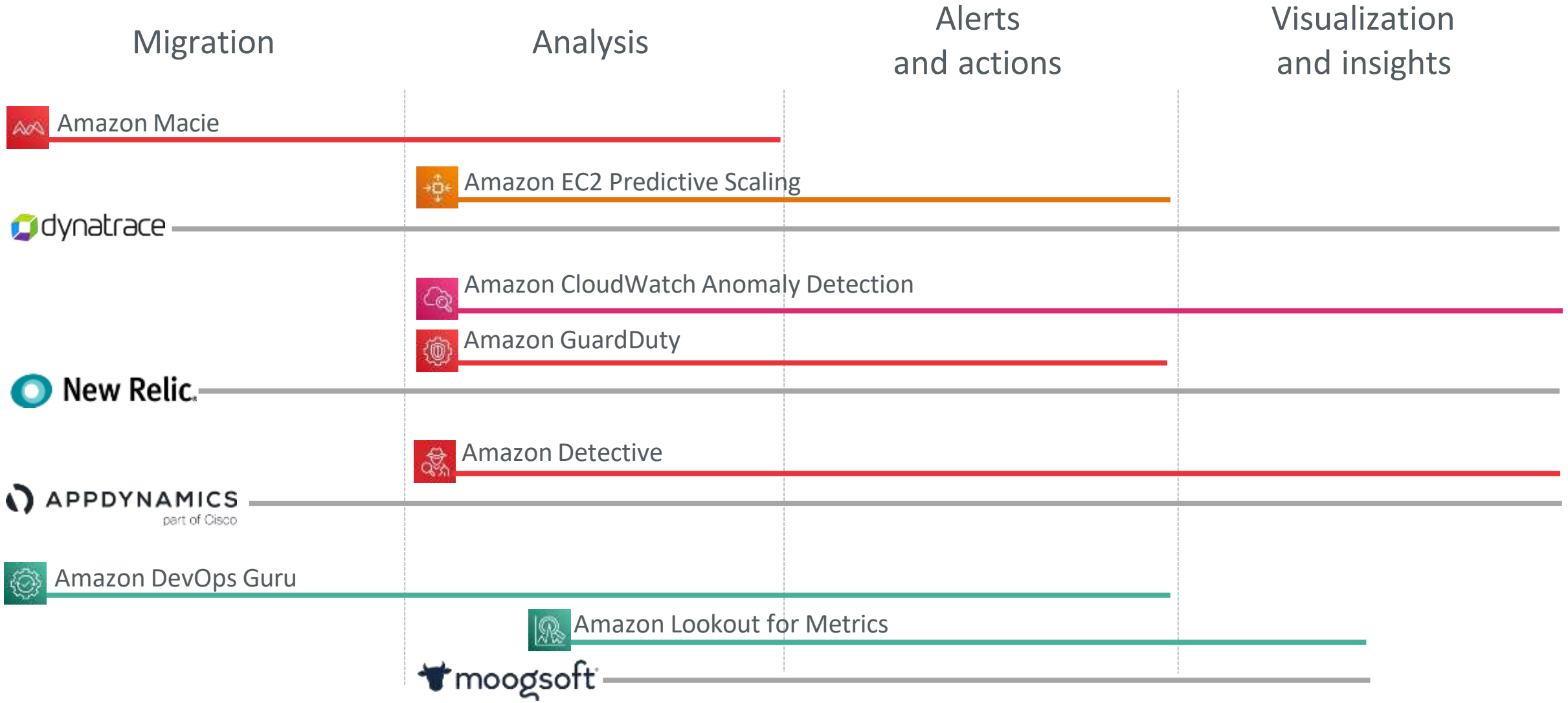
# CloudWatch Synthetics

## Create canaries to monitor your endpoints and APIs



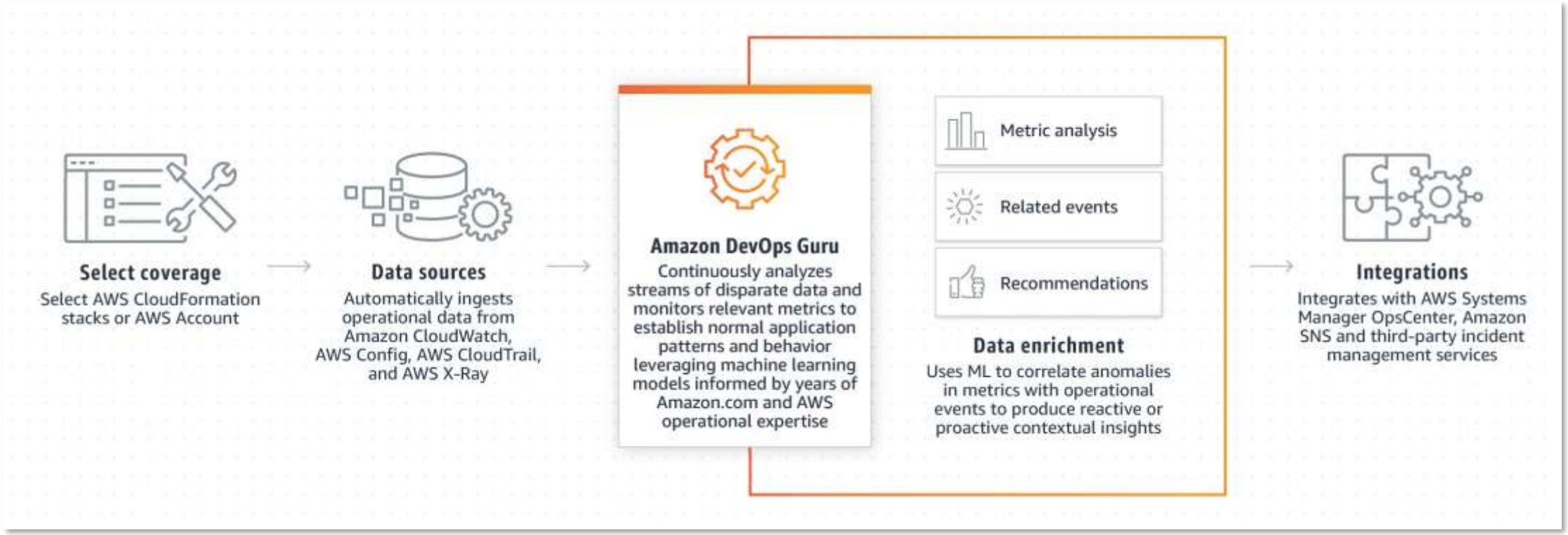



# AWS solutions



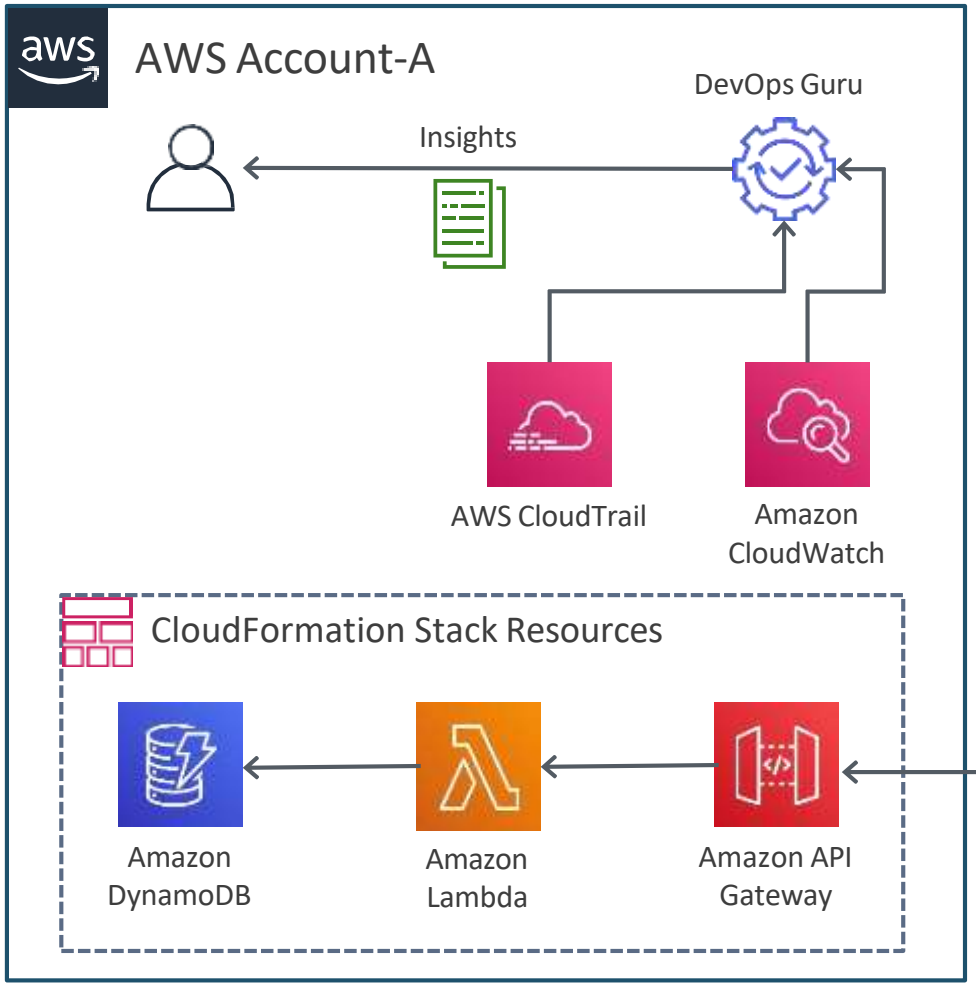
# Amazon DevOps Guru

ML-powered cloud operations service to improve application availability



# Amazon DevOps Guru

ML-powered cloud operations service to improve application availability




**1. Enabling DevOps Guru for the CloudFormation stack**

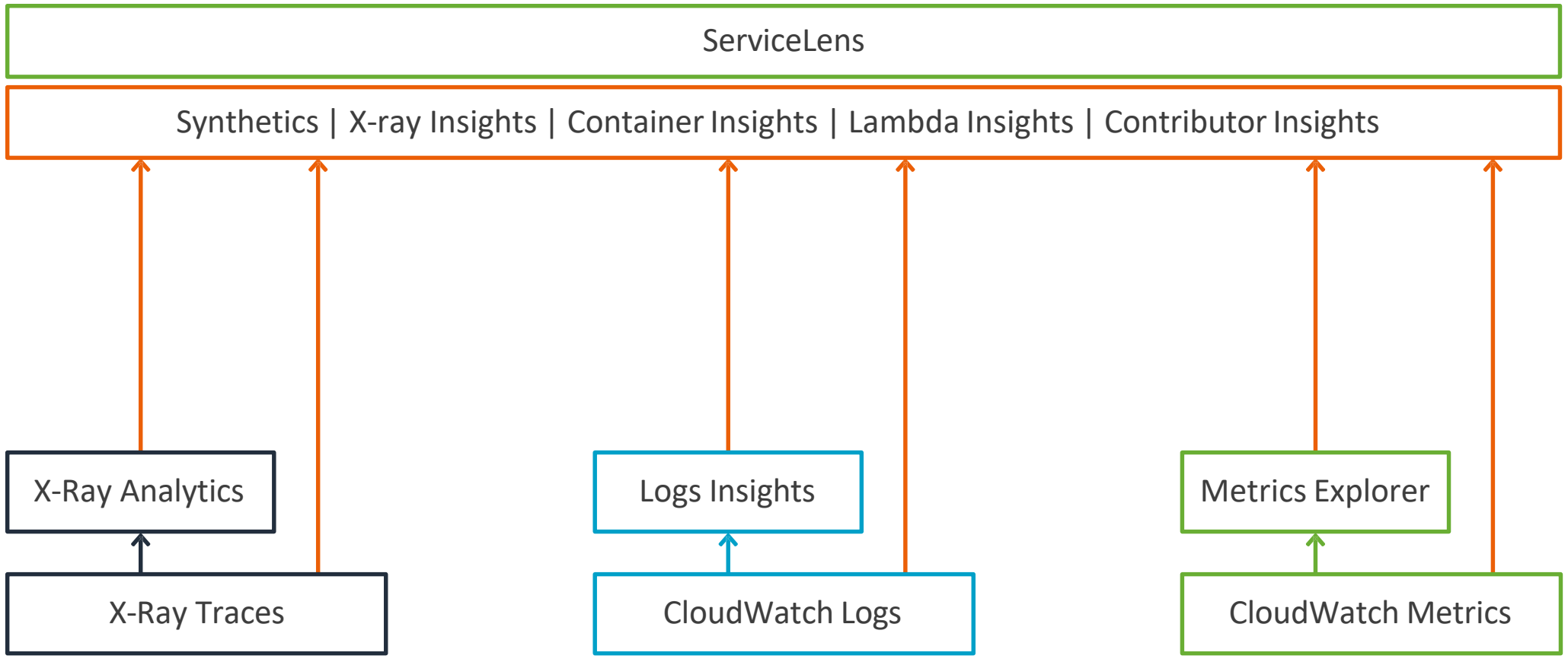
To enable DevOps Guru for CloudFormation, complete the following steps:

- Run the CloudFormation template to enable DevOps Guru for this CloudFormation stack:

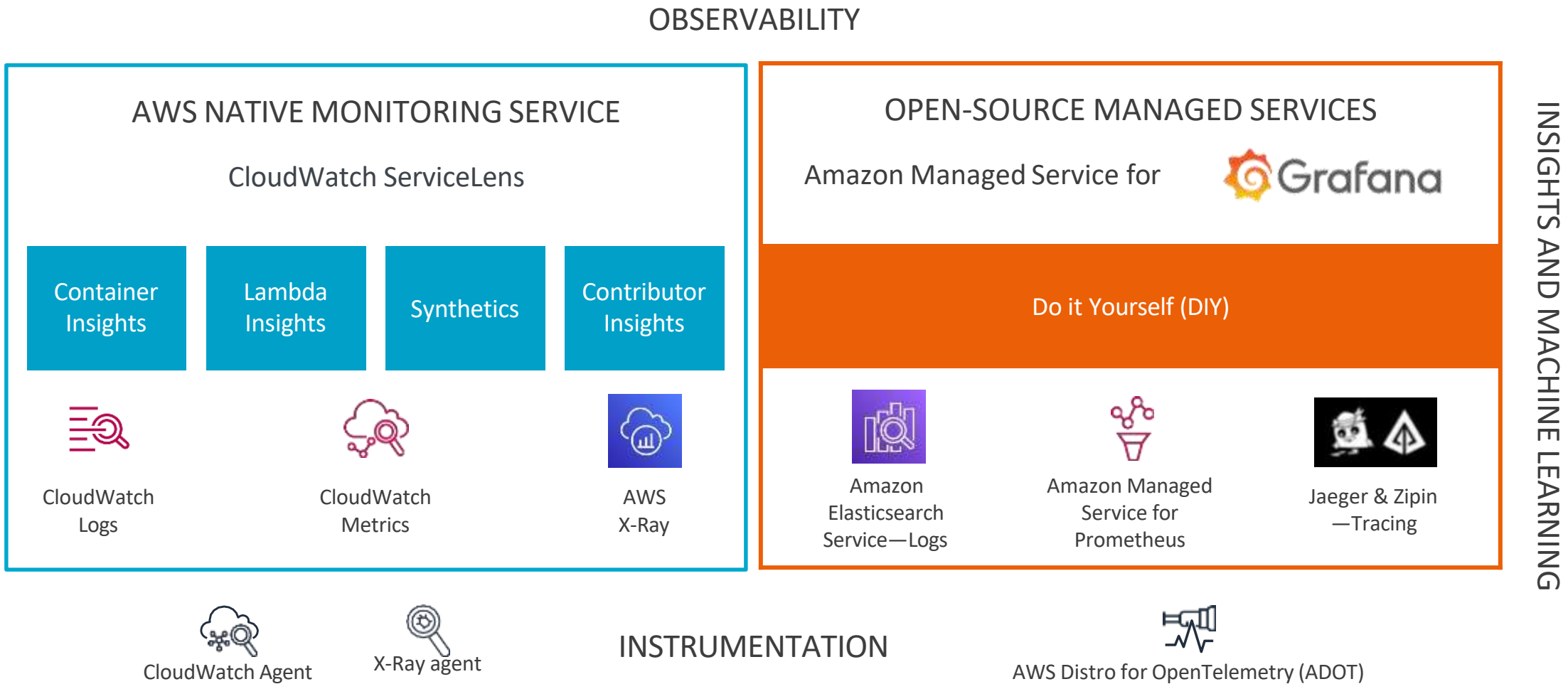
Bash

```
aws cloudformation create-stack \
--stack-name EnableDevOpsGuruForServerlessCfnStack \
--template-body file:///$PWD/EnableDevOpsGuruForServerlessCfnStack.yaml \
--parameters ParameterKey=CfnStackNames,ParameterValue=myServerless-Stack \
--region ${AWS_REGION}
```

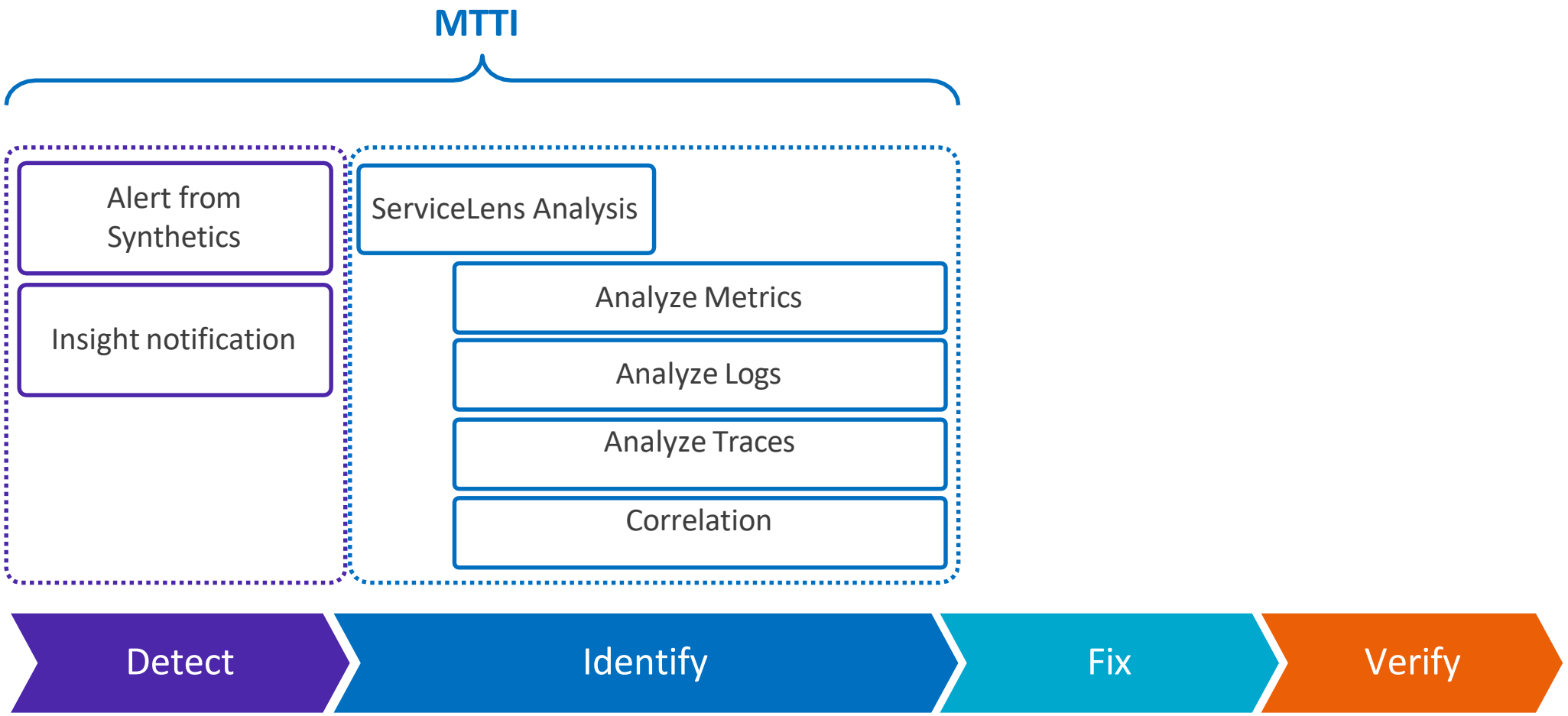
# Insights into apps and infrastructure



# AWS observability options



# New troubleshooting workflow



# Interacting with Amazon CloudWatch

AWS Management Console

AWS CLI

Amazon CloudWatch API

AWS SDKs

## ISVs Using CloudWatch

dynatrace

sumo logic

logz.io

LogicMonitor

PagerDuty

splunk>

New Relic.

APPDYNAMICS

# AWS Marketplace observability and monitoring solutions
















# How can you get started?

## Find

**A breadth of DevOps monitoring solutions:**

dynatrace

APPDYNAMICS part of Cisco

New Relic.

moogsoft

splunk>

sumo logic

THUNDRA

lumigo

bigpanda

## Buy

**Through flexible pricing options:**

Free trial

Pay-as-you-go

Budget alignment

Bring Your Own License (BYOL)

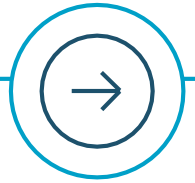
Private Offers

Billing consolidation

Enterprise Discount Program

Private Marketplace

## Deploy



**With multiple deployment options:**

SaaS

Amazon Machine Image (AMI)

CloudFormation Template

Containers

Amazon EKS/ Amazon ECS

AI / ML models

AWS Data Exchange

AWS Control Tower