
AWS
re:Invent
NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

PRT283

SPONSORED BY SUMO LOGIC

Is it ODD to shift left? Becoming elite DevSecOps performers

Colin Fallwell

Field CTO
Sumo Logic

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Today’s reality/industry trends

Process capability

Observability-driven design (ODD)

How to build elite teams

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Today’s modern systems are complex

Stack complexity

Accelerating change

Increasing areas of impact

Deep systems



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The reality for most . . .

Burnout is at an all-time high.

Many teams are more reactionary than ever.



>90%

of organizations moving to cloud-native architectures running production Kubernetes have experienced a significant outage or security breach within 12 months

– 2021 Red Hat survey

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Process capability

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Capability of process

Traits of a capable process are

- Can be measured with predictability across a range of one or more dimensions
- Exhibits low variability in the measures used to infer the state of reliability
- Measures are deeply correlated to outputs and/or outcomes

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Process engineering

Strong process capability depends on:

- Knowing the voice of the customer (VOC)
- Measuring the voice of the process (VOP)
- Intention in the form of specification limits
  - Upper and lower specification limits (USL/LSL)
- High sampling rate for data

$$C_p = \frac{\text{Specification width}}{\text{Process width}} = \frac{\text{USL - LSL}}{6\sigma}$$

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Measuring metrics

Good metrics are consistent

- Exhibit tight performance corridors
- Are predictable (for example: as load increases, responsiveness remains stable to a predictable point)



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Improving capability of process

- Build a framework for metrics (high value vs. supportive)
- Eliminate queues and backlogs anywhere they exist
- Break services often, battle-test code
- Publish mocks for the services you write and get your producers to do so as well
- Identify canaries and validate continuously
- Automate nearly everything
- Chaos engineer – develop good game theory
- Destroy and rebuild clusters often

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Organizational capabilities and best practices

- DevSecOps and business analysis are intelligent, continuous, immutable, and real time; data is unified and democratized
- Common instrumentation libraries are used across the organization; metadata is consistent and declarative
- CI, telemetry, and communication pipelines are autonomous, consistent, and declarative
- Observability is cleanly annotated, and instrumentation is done with domain/aspect-oriented programming
- Metrics for monitoring health/performance are declarative
- Dashboards, alerts, and alert thresholds are declarative and deployed with each merge
- A control plane evaluates outputs against rules, and validates canaries, scales up/down, rolls back intelligently, handles level-0 auto-remediation well

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Observability-driven development (ODD)

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

> “As a developer, if I’m not getting feedback from the code that I’m pushing and the changes that I’m making, I will be burned out immediately, and probably after a year I will leave the company. But the problem isn’t in the company; it’s in the relationship between the developer and production.”

— **Ran Leibman**

Production engineer, Facebook

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why ODD?

The ultimate goal of ODD: achieve a level of process capability (Cp) that lets you go straight from development to production

Testing in production has numerous benefits

- Compressed cycle time
- High quality data
- Exposes weak process

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

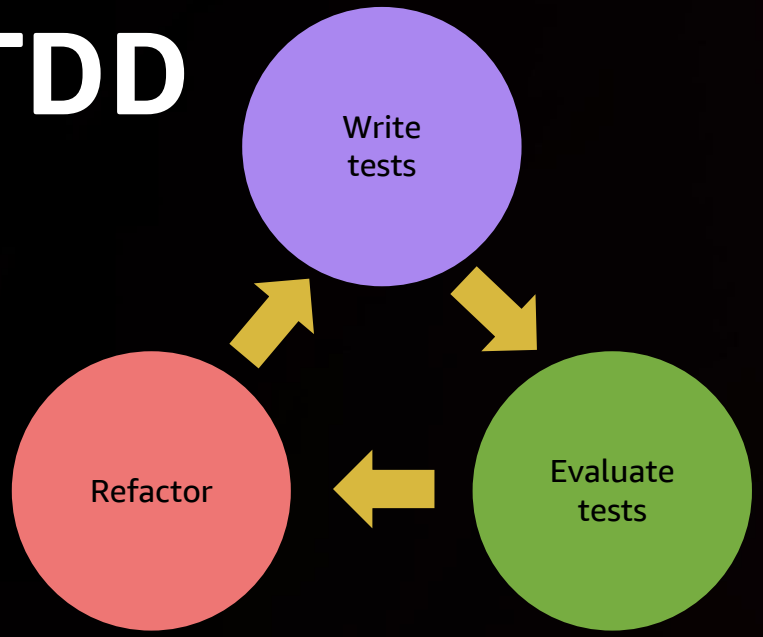
Building observable systems

- What you measure, why you measure, and how you measure is shifted left to the earliest phases of development
- Engages business leaders, including finance, marketing, security, operations, and business intelligence
- Measure things in concise, reliable, and predictable ways

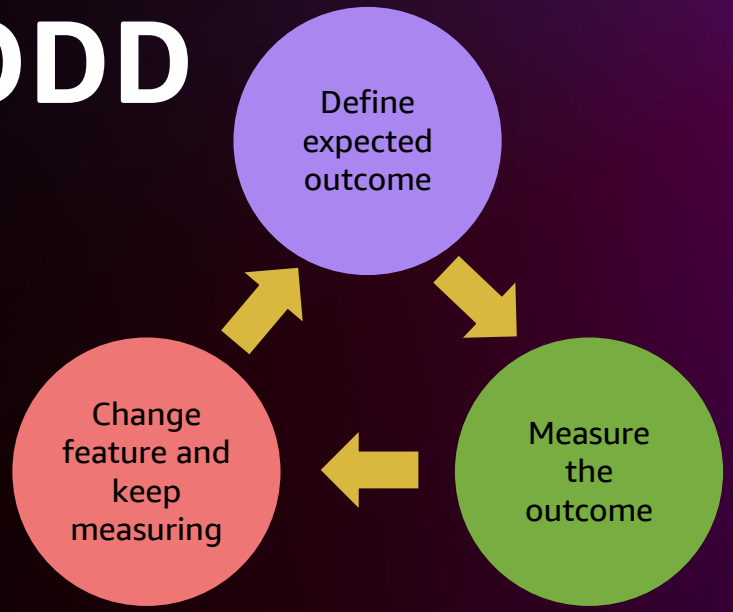
- Consolidate point tools and proprietary agents, embrace OSS where possible, and unify telemetry
- Converge all telemetry to a platform that can unify the entity model (UEM)
- Standardize metadata, key metrics, and logging and implement tracing

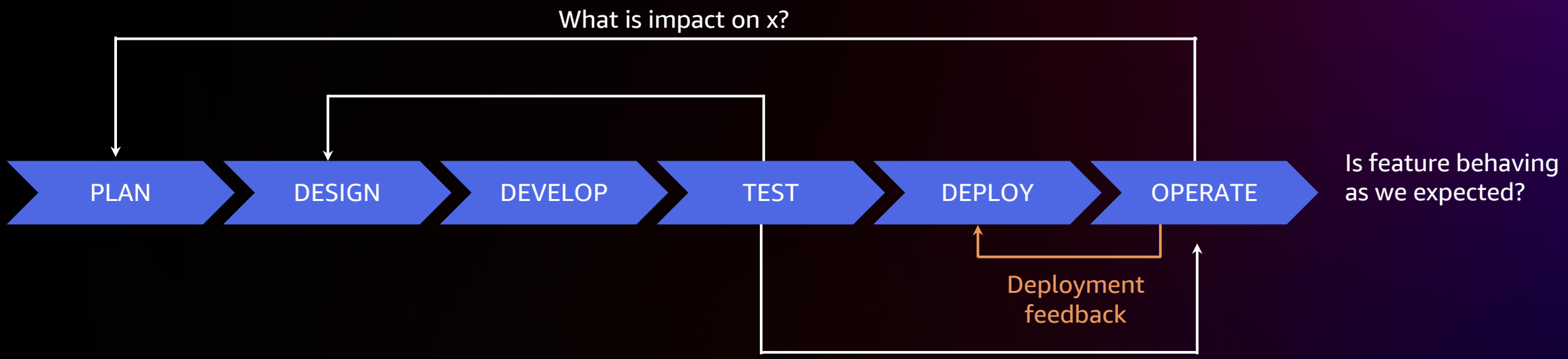
aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

TDD



ODD

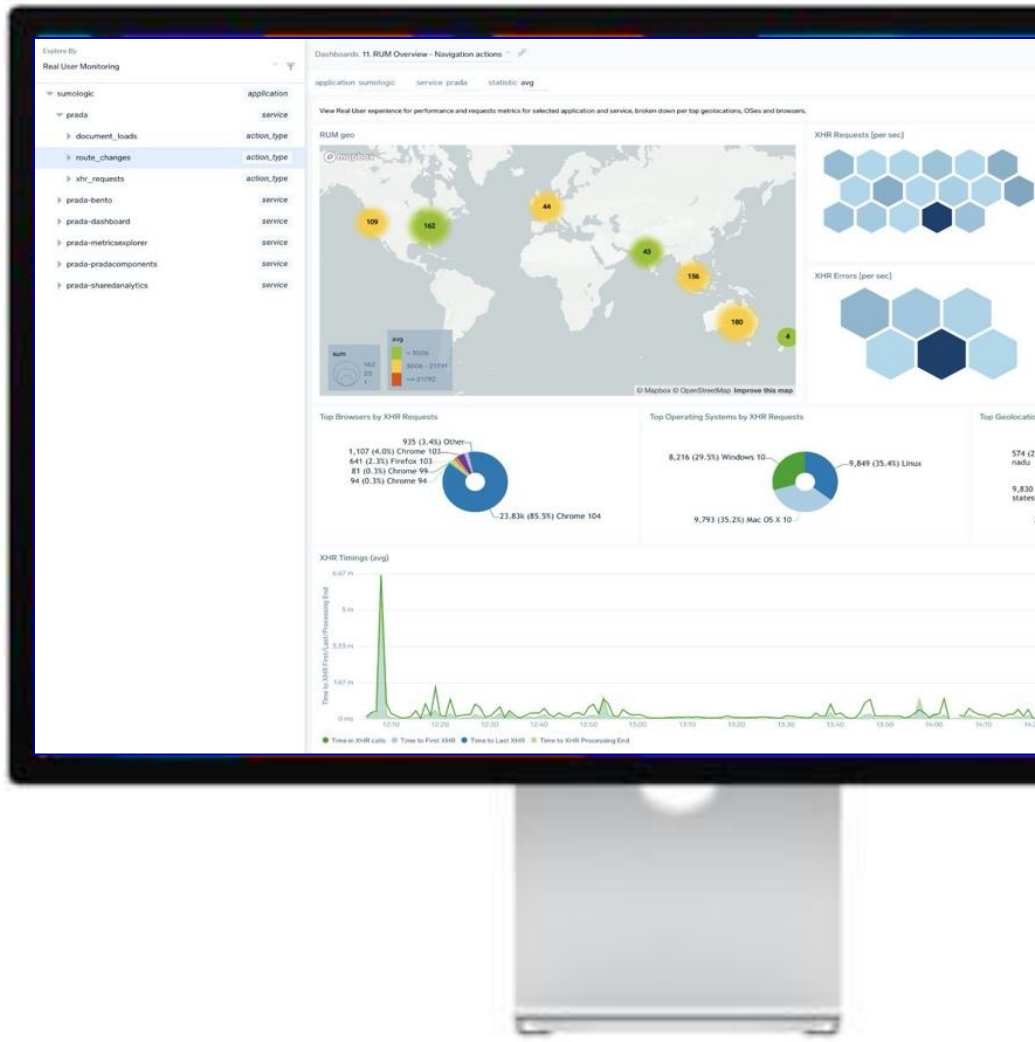




© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practices in ODD

- Unify metrics, traces, logs and events, aligned to feature level
- Know the voice of your customer – how is value delivered?
- Measure process, measure experience – consistently and continuously
- Code for predictability as much as possible
  - Avoid lazy coding – e.g. Getall() type statements
  - Offload heavy workloads to async, batch, queuing
- What upstream/downstream components could derail the CX, can risks be mitigated?
  - Circuit breakers or other patterns?



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Mastering metrics in performance engineering

Wrong way

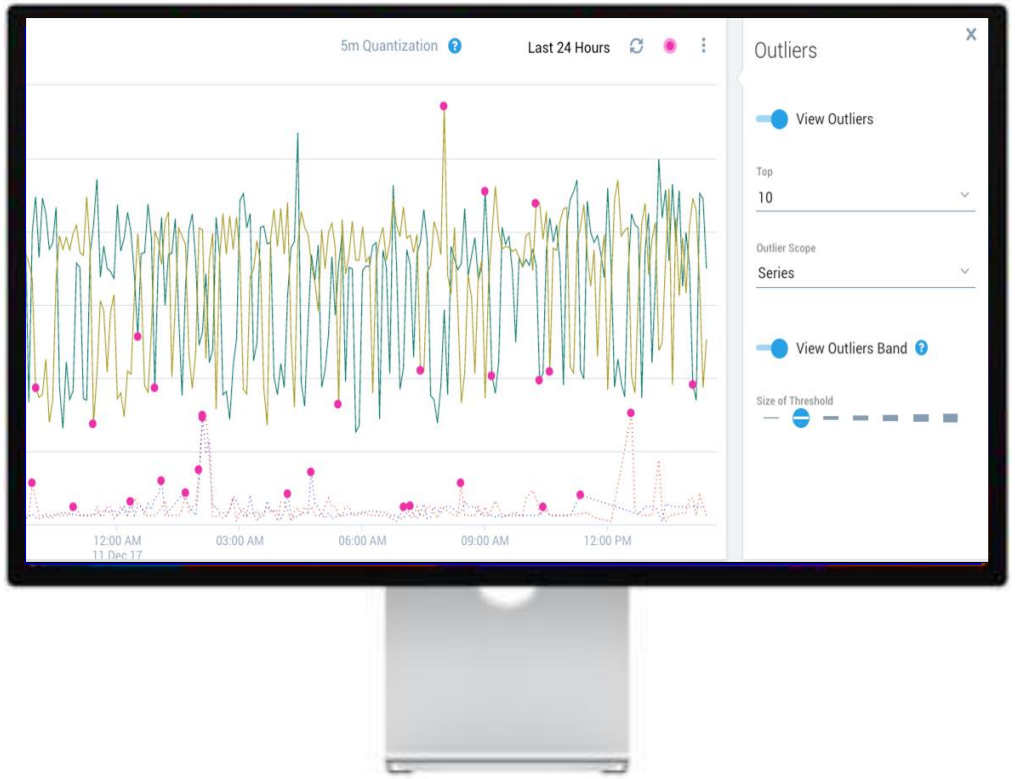
Alert on metrics when they >3–4 std. deviations from avg. or baseline

- Result: Alerting on change absent context about drift

Right way

Define performance corridor (LSL/USL) 2–3 std. deviations in width

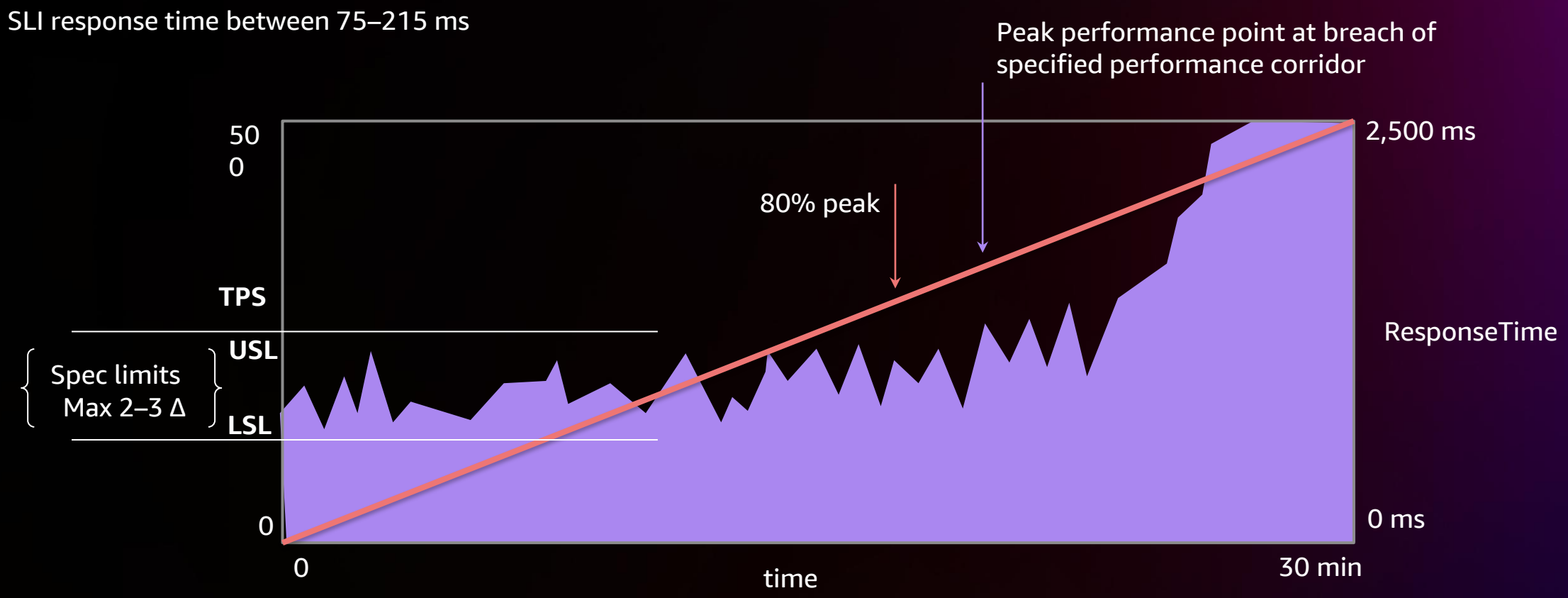
- Measure the count of times metrics fall outside the corridor and budget to law of 9s
- Result: Improve process capability



aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Performance engineering

RAMP TO BREAKING AND LEVERAGE MOCKS TO ISOLATE CODE PERFORMANCE



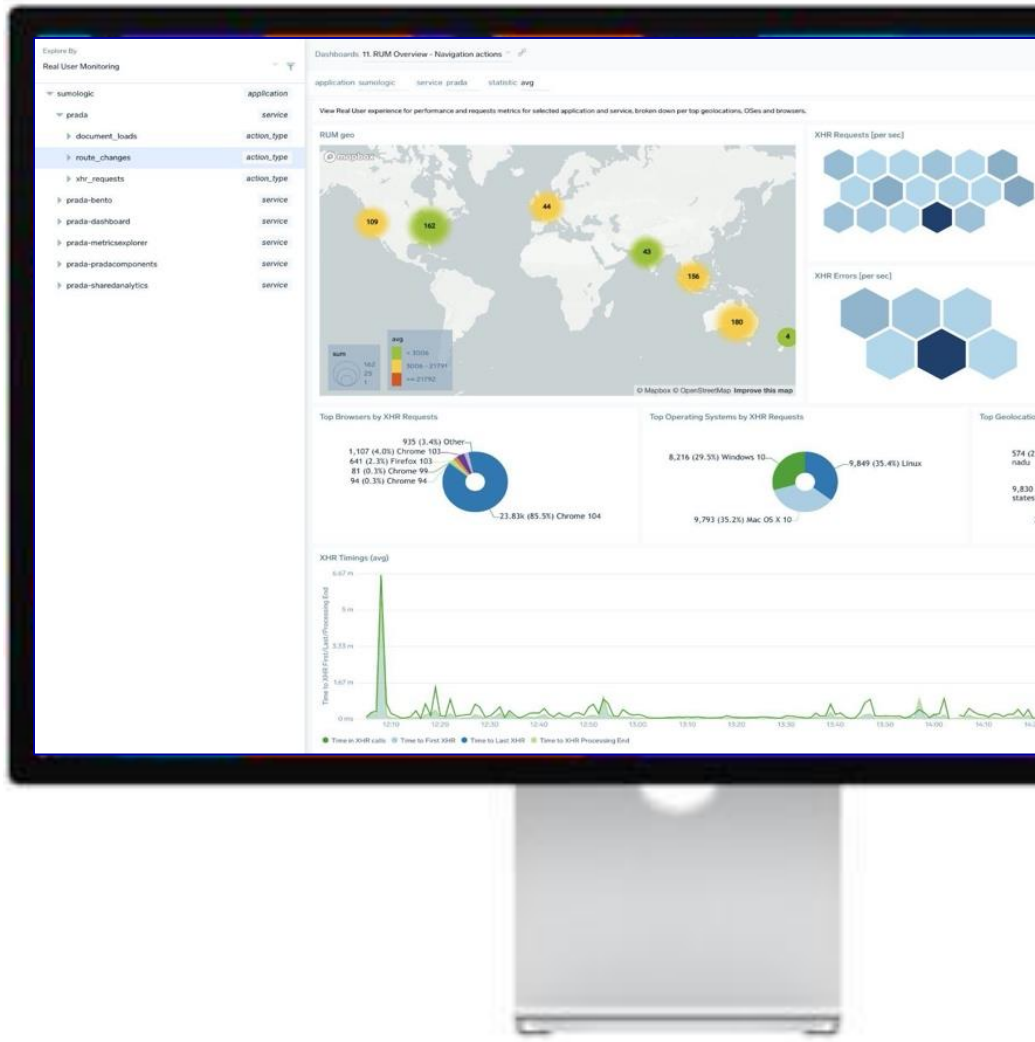
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practices in measuring

- Define the attributes of features in request/response
  - Ensure you measure features with consistency

    Unique code paths should get their own metrics
  - Bucket transactions by consistent taxonomy

    E.g., measure discrete method operations, metadata in headers
- Identify canary metrics (less is more) (TPS/Resp.T/Errors)
  - Metrics must infer the internal state of the system and relate CX
- Measure TP, create budget to the law of 9’s



aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practices in observability as code

Develop/Extend OSS telemetry SDKs to suit needs of enterprise, standardize

- Annotate performance specs, logging, metrics, span events, code ownership, monitors/alerting
- Or Terraform it; point is, make sure it lives in the codebase
- Include all relevant business contextual data to logs/traces/metrics as attributes; higher cardinal data is better
- Data should be product- and process-oriented, and high value

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Best practices in observability as code

- Always use some sort of domain-oriented observability abstraction when adding observability to domain code – the areas of your codebase that are focused on business logic
- Using something like a domain probe keeps that domain code decoupled from the technical details of your instrumentation infrastructure and makes testing your observability a feasible endeavor
- Want cleaner code? Use aspect-oriented programming

aws
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Retrofitting existing codebases

- Only retrofit areas of your codebase that you're already working on for other reasons
- Don't allocate a dedicated effort to moving everything over in one go
- Be sure that "hot spots" in your code – areas that are frequently changing and likely more valuable to the business – are made more observable and easier to test
- Leave the rest to logging

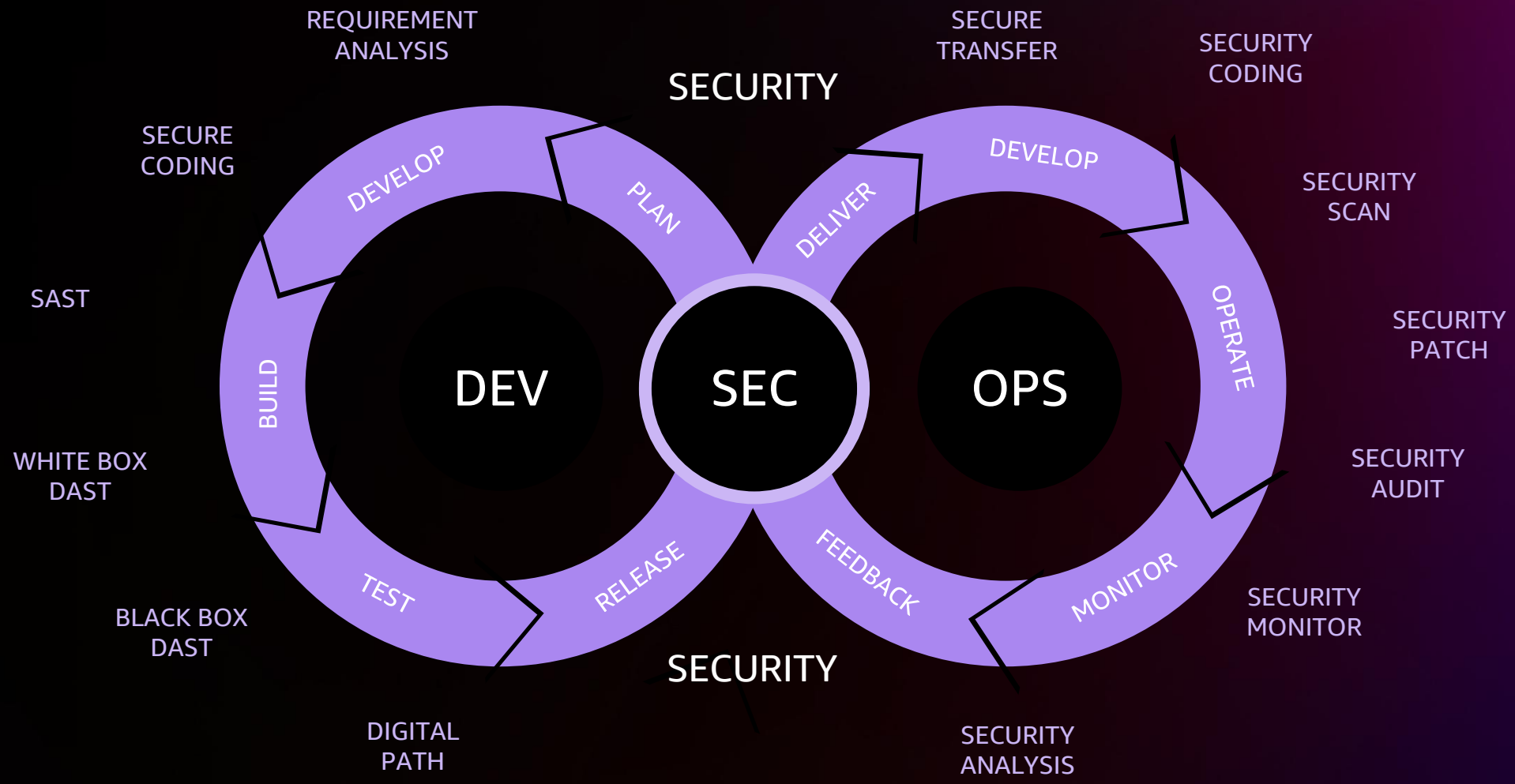
aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Building elite performing teams

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Old ways no longer work



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Performance comparisons

Software deployment performance metric	Elite
Deployment frequency	on demand (multiple deploys per day)
Lead time for changes	less than one hour
Time to restore service	less than one hour
Change failure rate	0–15%

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

State of DevOps report findings . . .

Elite performing organizations

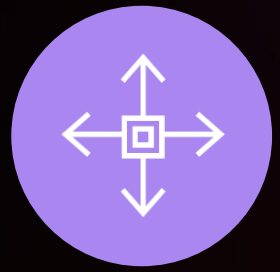
- Are up from 18% in 2018 to 26% in 2021 in 4 years
- Are 50x more likely to accelerate market expansion
- Are 80x more likely to be successful in M&As
- Are 24x more likely to fully embrace cloud-native capabilities
    - On-demand self-service
    - Broad network access
    - Resource pooling
    - Rapid elasticity
    - Measured services

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Performance metrics







SOFTWARE DEVELOPMENT

SOFTWARE DEPLOYMENT

SERVICE OPERATION

LEAD TIME

CHANGE FAIL

RELIABILITY

DEPLOYMENT FREQUENCY

TIME TO RESTORE

FOUR KEY METRICS

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Who is an elite performer?



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Pipeline goals

Standardize project types! Only one way to test and only one way to ship code for each type

Automated to do everything needed in PR and merge

Test often . . . like on every commit

Build artifacts often, code should be built as often as it’s tested, deployable anytime

Be environment-agnostic, maximize portability

Get consistent feedback, CI output the same regardless of language runtime

Short, tight feedback – engineers should get actionable feedback as fast as possible

Well-tested code, stored in version control that can easily be changed and developed at any time

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Goal #1:
Full automation, hierarchy, and control

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

CI best practices

- There are projects: Each project needs a configuration definition file (control.yml) that declares its project type. Each project type you ship should be supported.
  - java_library, java_app, ruby_gem, ruby_app, javascript_library etc.
- There are repos: Each repo has one or more projects of any type.
- There needs to be an automated way to set up a new project
- Each project needs to be built, scanned, tested, and linted
  - There needs to be a way to idempotently generate the configuration (e.g. .circleci/config.yml) for all the projects in a repo at once



Jeff Dickey, “12 Factor CLI Apps,” https://bit.ly/2Ohy9x6

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Goal #2:
Provide engineers feedback with speed and precision – the comms pipeline

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Comms best practices

- Let the author know as soon as possible when something is red, but don’t overdo it for redundant failures with the same job
  - E.g., a job on 20 containers where 18 saw failures, only notify once
  - Only notify once about all the green things
- Give as much context as possible without overwhelming the author
  - Concise and clear

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Improving comms – What is valuable?

CONTEXT AND STATUS ARE THE MOST IMPORTANT THINGS TO COMMUNICATE

For PRs

- Link to what branch failed
- Link to which repo and for which PR
- Link to failed job in the CI suite

For merges

- Link to commit diff
- Link and title of job that failed
- Link to the top-level workflow and @s for each contributor

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Wrapping up: A real-life example

(BOUTIQUE RETAILER)

- **Engineering teams are free to push code** to production environments
- **Services are written with mocks** that can be used by other teams
- **Automated performance tests of code** in the pipeline
- **CI is unified and all projects are idempotently managed** by SRE teams
- **Completely destroy and rebuild their Kubernetes clusters** each week
- **Leverage logs and metrics** for all their automation needs, **traces** for customer experience
- **All data is tagged** to feature level

aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!



Please complete the session survey in the **mobile app**

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.