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

AIM205

Fidelity elevates developer operations with ML-powered insights

Vijay Ranganathan

Vice President, SRE
Fidelity Investments

Ashok Kanjamala

Principal Product Manager, Amazon DevOps Guru
AWS

Agenda

About Fidelity

Observability Maturity Model, DoD (Definition of Done)

What is Amazon DevOps Guru?

Fidelity’s DevOps Guru engagement approach

Issues and remediations, integrations, self-healing

Key takeaways

1946

Fidelity Management & Research Company founded

1965

Fidelity purchases first computer

1995

Fidelity becomes an internet pioneer with first mutual fund to create a home page

2016

Fidelity's first product application deployed to the cloud

2019

Fidelity launches multi-cloud hybrid strategy

Today

5,700 + applications on public cloud

Massive scale

Shared responsibility

Each individual team to look after their own application's health

Applications

Hundreds of workloads moving to the cloud across the enterprise

Applications are tiered

Accounts

Hundreds of accounts spread across multiple regions managed by multiple distinct internal business units

Organizational units

Variety of internal business units are migrating to the cloud, all at different levels of maturity in their cloud journey

Developers

17,000 technologists spread across the globe

Application availability (focus areas)

Senior tech executive

I want a consistent way to capture anomalous behavior so my engineering teams can manage and correlate metrics to quickly resolve issues.

I need to determine where we are doing a good job and areas where we can improve.

BU owners

I want my availability expectations defined and documented so that I can measure and report. I want to relay expectations to other dependencies.

I want to continuously measure the performance of my business service and user experience so that I can detect before customers and alert on our error budget burn-down.

I want to enable my teams to focus on the roadmap items.

Operations leader

I need visibility of the service adoption and the impact on the production support data (incidents, MTTR).

I want to enable teams to significantly reduce operational downtime.

I want to enable teams to track and meet their availability targets.

Software engineering team and architect

I want to leverage the ML-powered insights to help identify, correlate, and remediate operational issues.

I want to utilize the integration with ServiceNow to quickly end efficiently deliver issue/recommendations to the right people at the right time.

I want to use the proactive insights to identify problems around resource exhaustion in advance.

SRE and platform engineering

I want a common set of anomalous metrics and recommendations for each AWS service, so I can guide application teams appropriately.

I want to have well-documented self-service onboarding steps, so I can help teams with their adoption.

I want to measure the value-add with respect to MTTR.

Enterprise-wide data transparency (same data with different filter/lens)

Why our observability strategy matters

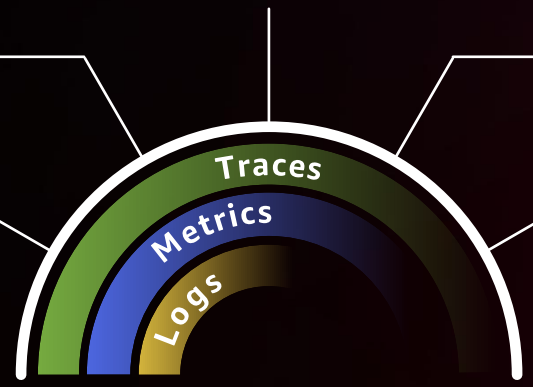
Identification of strategic improvement opportunities
Identifying weak link in the distributed systems

Minimize MTTR
Expedites troubleshooting by pinpointing the root cause
Minimize time to diagnose and resolve

Enabling data-driven decisions
Augmenting runtime resources

Understanding operational health
Reveal health of primary and supporting resources
Monitor critical signals improving MTTD

Improve business and technical agility
Accelerate developer velocity



Observability value curve

The three I's (eyes) of observability

Instrumentation

- Common patterns to capture the right signal in the right way
- Emphasize metrics-based SLIs and SLOs
- Leverage auto-instrumental tracing
- Enforce standards for log messages

Inspection

- Use each health indicator (metrics, traces, logs)
- Provide common platform for interacting and exploring observability signals

Insights

- Deliver actionable alerting capabilities
- Provide trend detection to warn of issues before they impact customers
- Consistently applied metadata to afford quickly connecting various signal categories

Observability Maturity Model + DoD

Transactional tracing across apps/bus **Predictive AIOps**	**Level 4** Self-healing and optimize	Cross-app observability and tracing Self-optimizing apps and infrastructure
RUM **Region health checks** **Customer performance**	**Level 3** Customer availability	Real customer experience Error budgets prioritizing backlog
Synthetic monitoring **Ops or SLI/SLO dashboard** **Distributed tracing for cloud platforms**	**Level 2** Common and predictive	Common SLI/SLO Application dependency metrics
Alerting **Correlated logging**	**Level 1** Basic (responsive)	Runtime measurements Faster application recovery
Disk I/O, CPU, memory, ping, network I/O, logging, process **Logging, process**	**Level 0** Monitoring	Are my systems running? Have I exceeded system capacity?

Amazon DevOps Guru

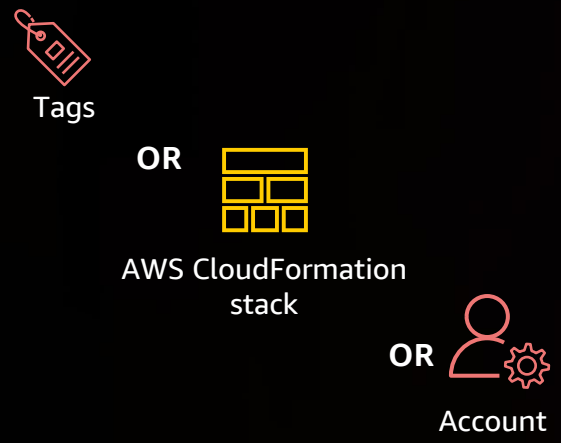
Amazon DevOps Guru

ML-POWERED CLOUD OPERATIONS SERVICE TO IMPROVE APPLICATION AVAILABILITY

DevOps Guru is an ML-powered service that makes it easy for developers and operators to automatically detect issues to improve application availability and reduce expensive downtime – no machine learning experience required

Get started with Amazon DevOps Guru

Select your application by tags, CFN stacks, or entire account

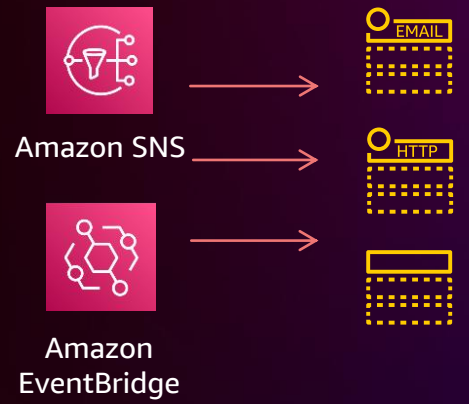


Select coverage

DevOps Guru will start analyzing metrics/logs to generate insights

Generate insights

Integrate with OpsCenter, Amazon SNS, Amazon EventBridge, and third-party platforms



Integrate with your workflow

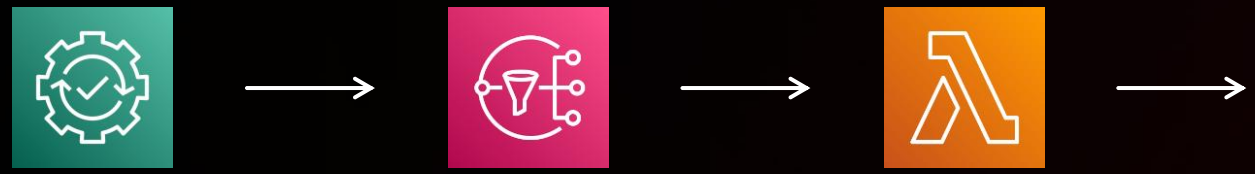
Engagement approach

Service type engagement	Production incident engagement
Targeted 4 AWS services (Lambda, ELB, Amazon API Gateway, Amazon SQS)	Target high-priority production incidents
Direct engagement with 3 business units	Direct engagement with 2 business units
Prove the value-add	Prioritized 2 AWS services (Amazon EKS, DynamoDB) Prove the value-add

Issues and recommendations (AWS services)

Resource	Issue	Auto healing job
Amazon SQS	Dead letter queue for SQS queue not configured	**Set up an SQS dead letter queue.** Without an SQS dead letter queue, failure to process the message will result in the message being continually retried by Lambda until the message expires from the queue and is lost. The retries use up your resources and add to your cost.
Amazon Kinesis	RetryLimit for Lambda Kinesis consumer has not been configured	**Set MaximumRetryAttempts.** Without limits on retries, Lambda will continually retry a failed batch until the messages expire from the stream. When this happens, the Lambda function is effectively blocked.
Amazon EKS	Amazon EKS deployment is having trouble scaling	**Set up Auto Scaling for your pods** to prevent this situation.
DynamoDB	Failure destination for Lambda DynamoDB has not been configured	**Set up dead letter queue** to prevent data loss in certain situations.
DynamoDB	DynamoDB table point-in-time recovery not enabled	**Enable point-in-time recovery** to prevent data loss.
Lambda	Lambda concurrent executions reaching account limit	**Increase the Lambda function concurrency** to handle your current load.
Lambda	Lambda provisioned concurrency usage lower than expected	**Reduce the Lambda function concurrency** to save cost.

Integrations with ServiceNow, self-healing



- Retrieve the current RCU for the table
- Increase the RCU by a certain number
- Enable Auto Scaling
- Update scaling policy

Self-healing automation platform

Key takeaways

- Self-healing is the key value-add using proactive insights
- Started with 6 AWS services, plans to add 3 more (Amazon RDS, Amazon ECS, AWS Step Functions)
- Share learnings (corrections), share techniques, common issues, common patterns across business units
- Focused approach to resolving the critical signals consistently (working with AWS to improve)

Thank you!

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