
AWS
re:Invent

FSI303

A Tale of Two Cities: Goldman Sachs's hybrid migration approach

Vicky Ledda
Vice President
Global Investment Research Engineering
Goldman Sachs

Yun Zhang
Vice President
Core Engineering
Goldman Sachs

AWS re:Invent

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

aws

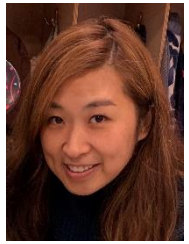
Introductions

Vicky Ledda



- VP, Investment Research Engineering
- Grew up in Milan, Italy
- Studied and worked in the UK
- Moved to NY last year

Yun Zhang



- VP, Core Engineering
- Grew up in Hefei, China
- Started programming from age 8
- Studied and worked in the US

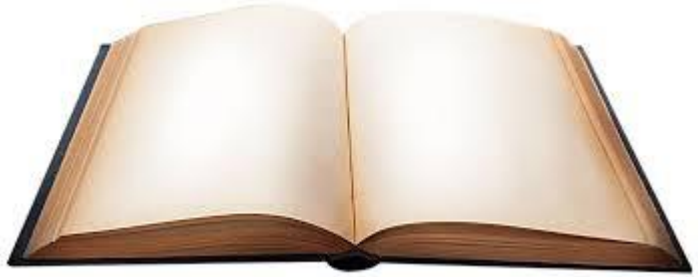
A Tale of Two Cities

"It was the best of times, it was the worst of times,

it was the spring of hope, it was the winter of despair,

we had everything before us, we had nothing before us."

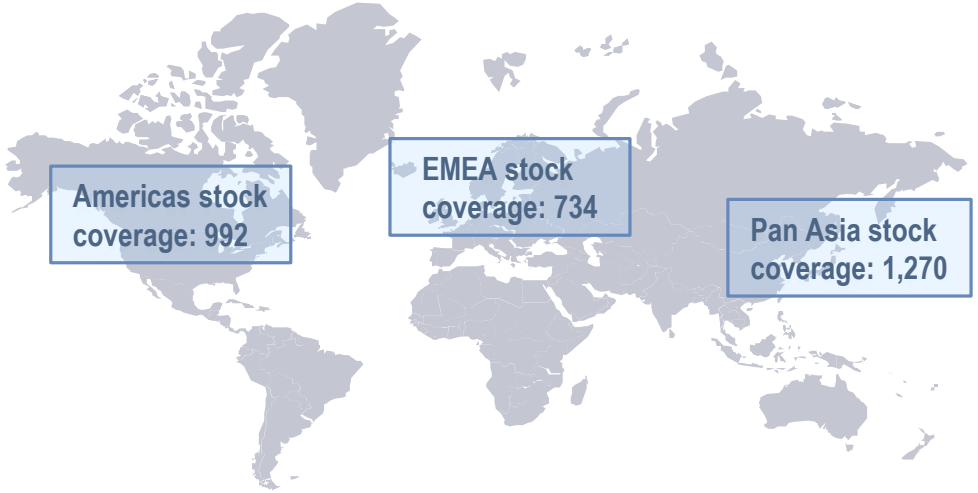
A Tale of Two Cities - Charles Dickens, 1859



Global Investment Research (GIR): an overview

Reach

GIR consists of ~1,000 people in 21 offices around the world, with ~3,000 stocks and over 40 economies & currencies and every major market under coverage



Service

Market-leading convening power

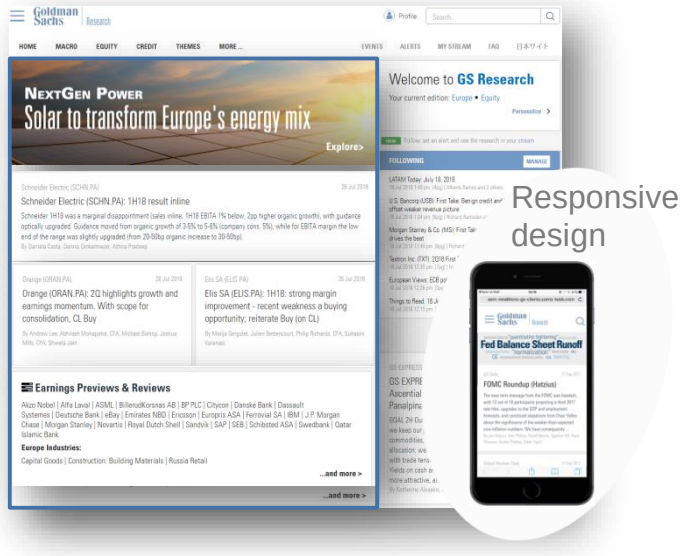
- Market-leading conferences, plus our webcasts, bring together the key players across industries and regions



Thematic research



The GS Research portal



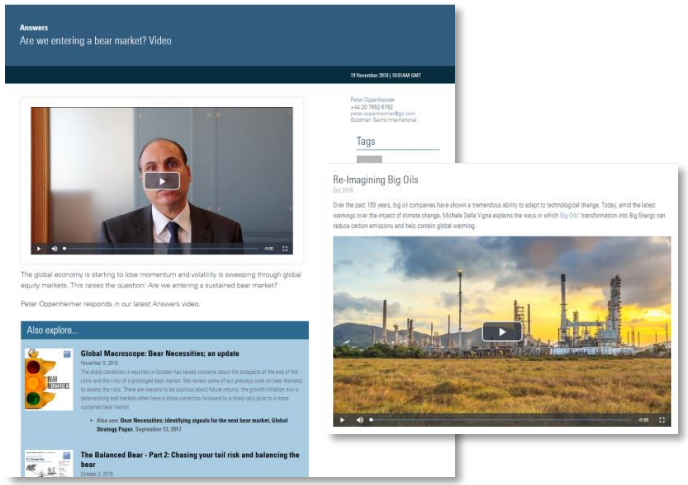
Responsive design

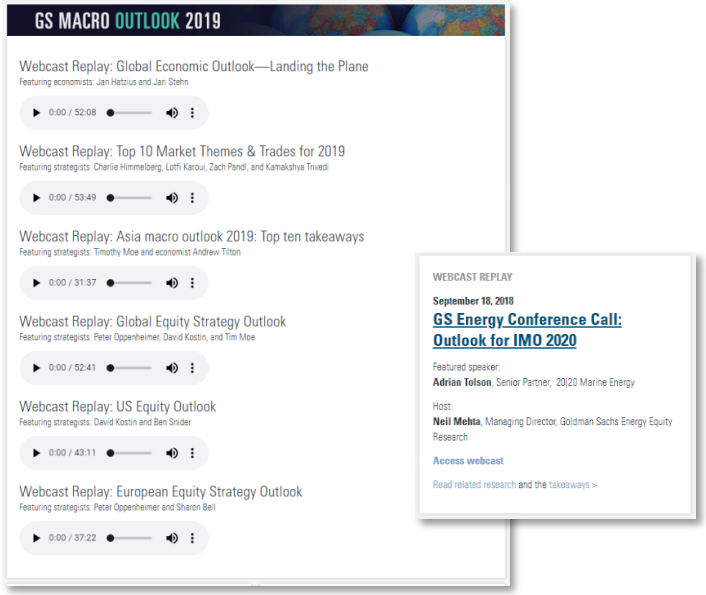
Source: Goldman Sachs Global Investment Research

Reaching our clients: an omnichannel approach

Webcast replays plus videos and podcasts of our analysts are available on the Audibles page.





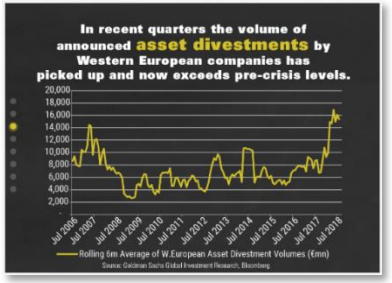


Responsive design with GS Mobile



Animated GIFs and infographics help cement the story.





Goldman Sachs

Why migrate to AWS?

Building a highly available, scalable, and efficient content distribution engine



Enable 24/7 research availability to meet regulatory requirements

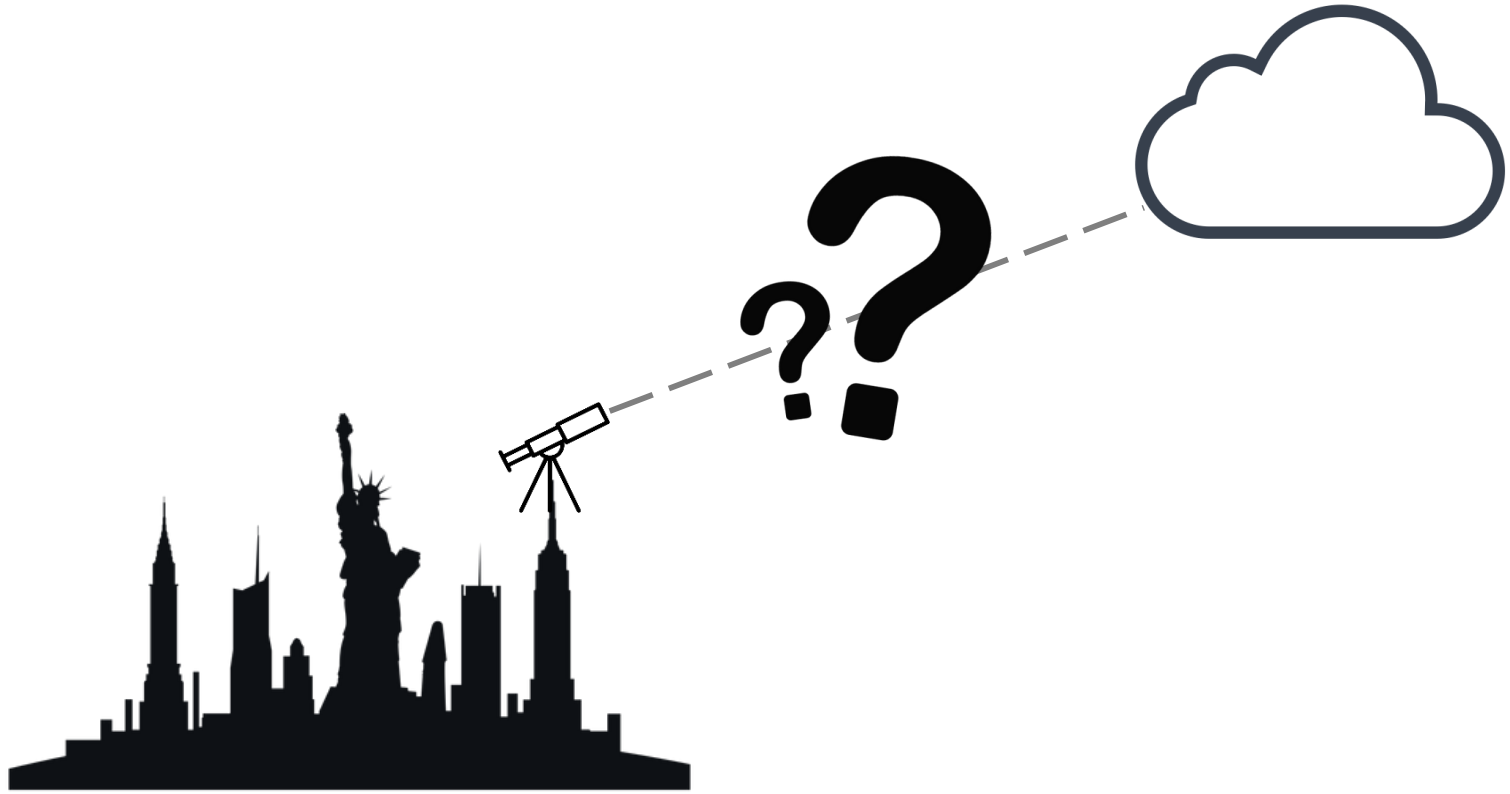


Improve resiliency, performance and security by taking advantage of AWS



Extend research to expanded consumers, including the public

Sounds great, but how?



Getting started

Developers use SkyLab, a firm-sponsored and monitored AWS account for discovery work



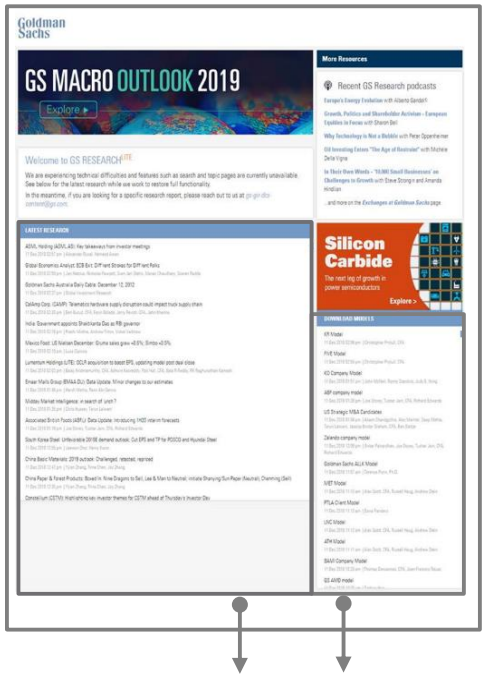
In cooperation with AWS Solutions Architects, AWS Immersion Days, and lots of experimentation



Internal consulting and security reviews for compliance and risk control

The journey

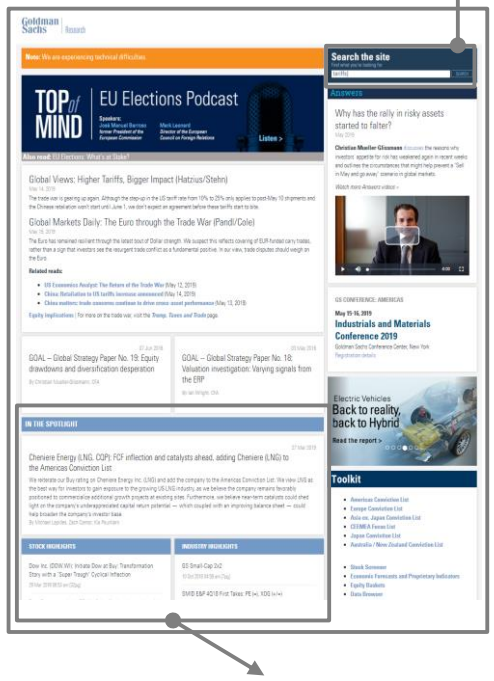
Research Lite



Latest research and models sorted by most recent published date

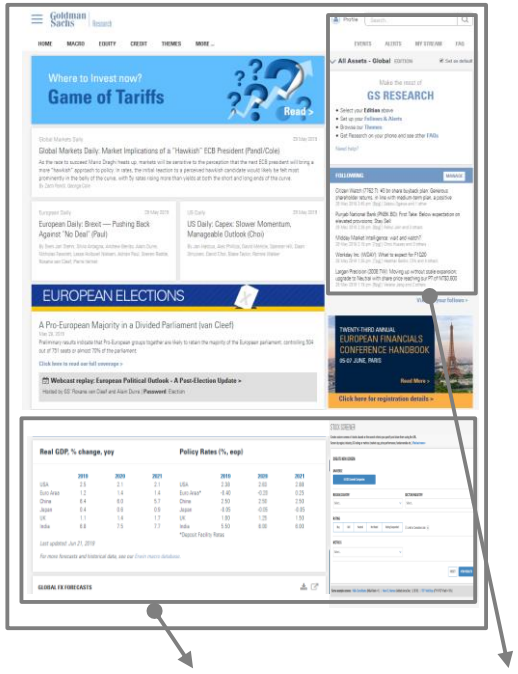
Parity

Search



Custom-curated components based on search algorithms

Primary



Research data

Personalization (follows & alerts)

Goldman Sachs

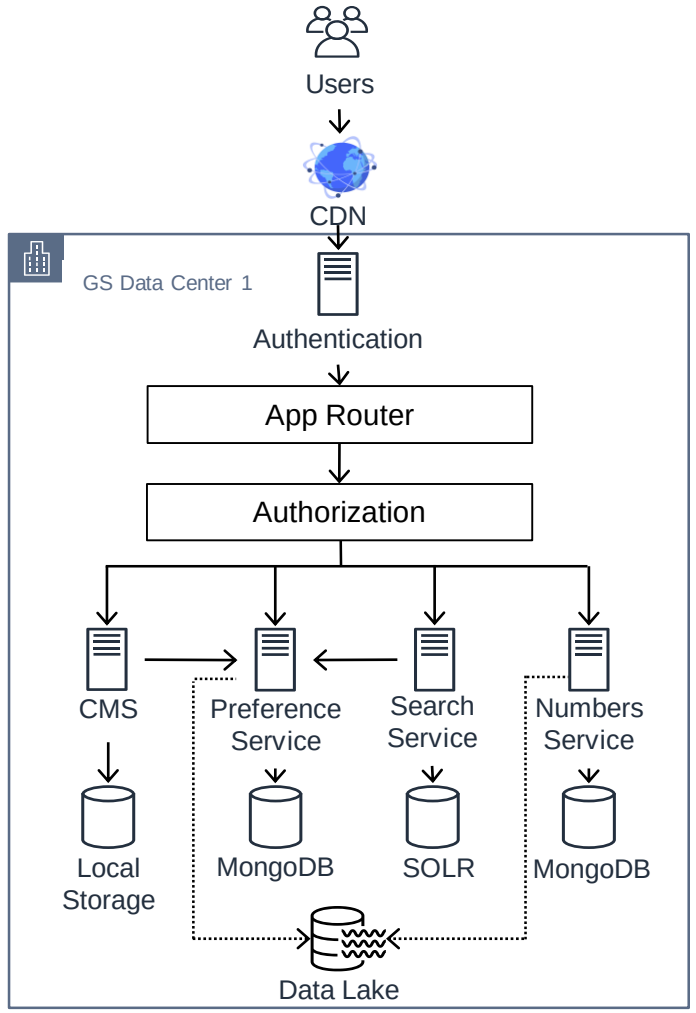
Chapter One

Achieving resiliency: Research lite on AWS



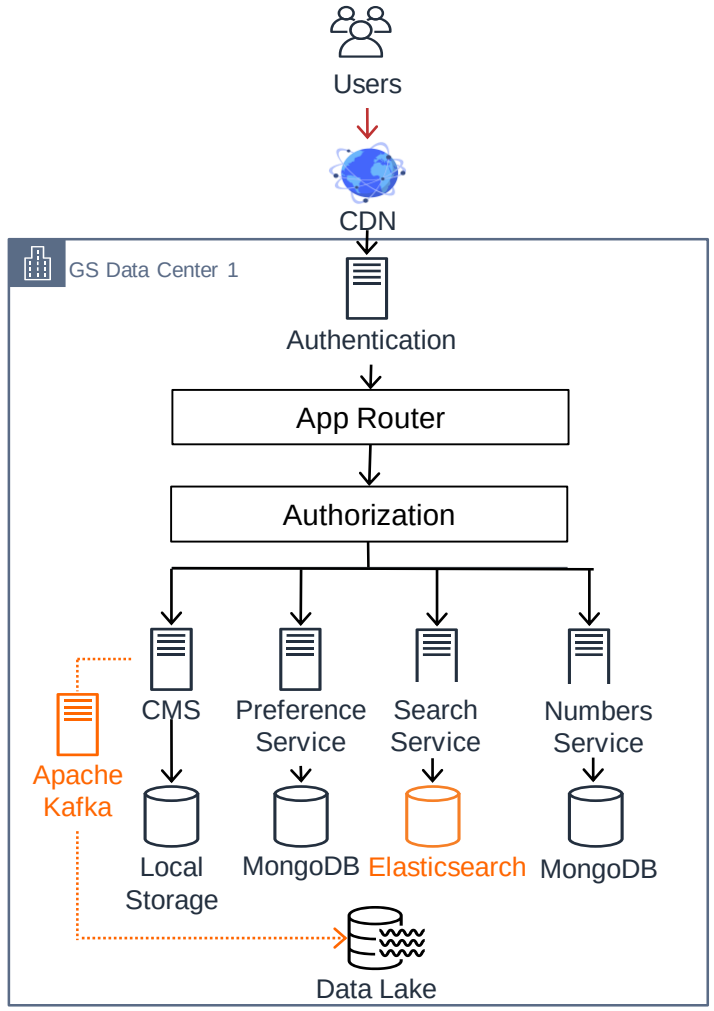
Where we started...

- Live-Live across two data centers
- Multiple third-party products
- Many dependencies on proprietary services
- Many “point-to-point” dependencies



Re-engineering *before* we moved to AWS

- Moved to an event-based architecture
- Used platforms that would be AWS-ready
- Moved away from monolithic deployments



Design principles



Cloud First

- Avoid “Lift & Shift”
- Design for resilience
- Use of managed services



Infrastructure as Code

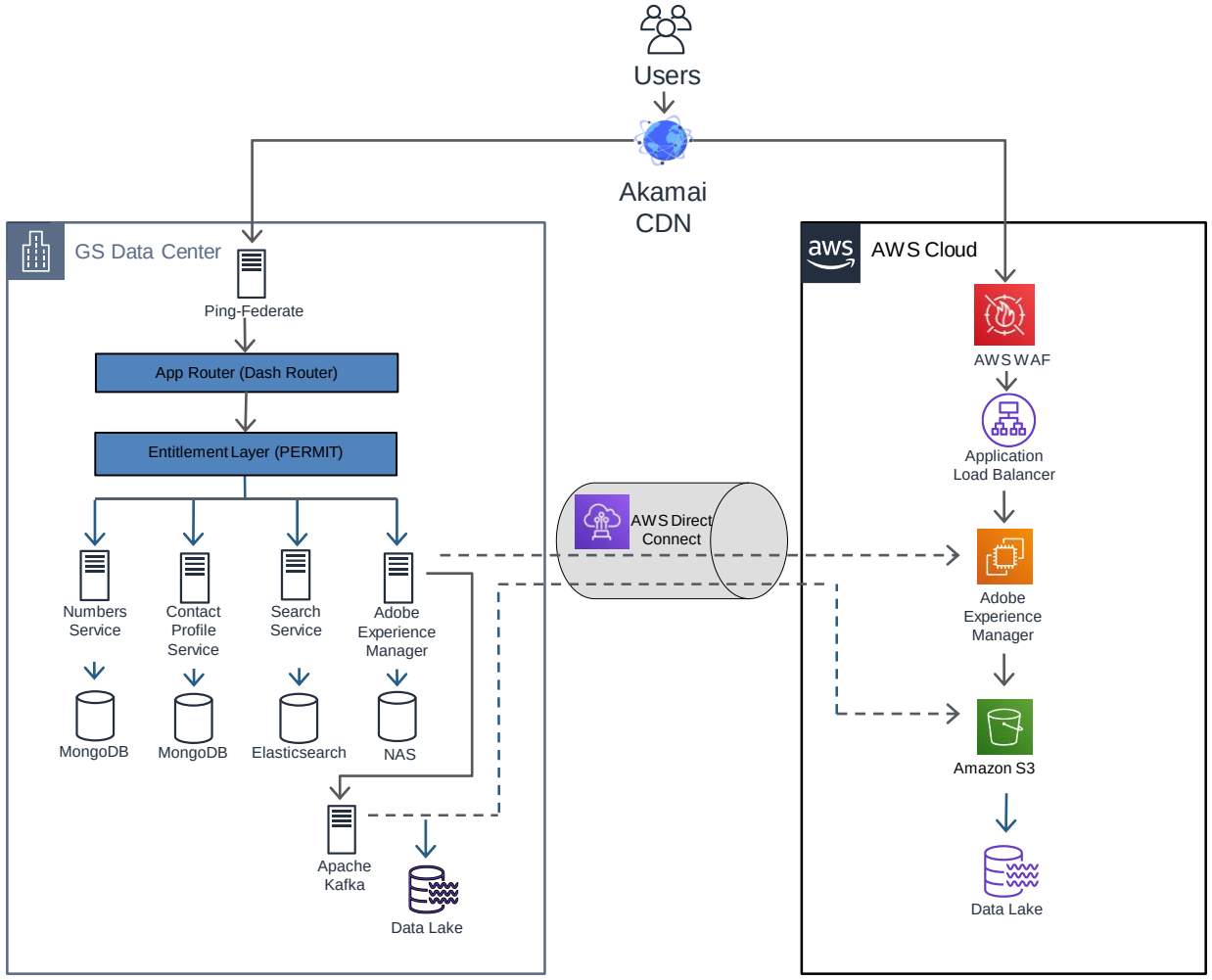
- Terraform to set up infra
- Automate all infrastructure setup
- SDLC for infra changes



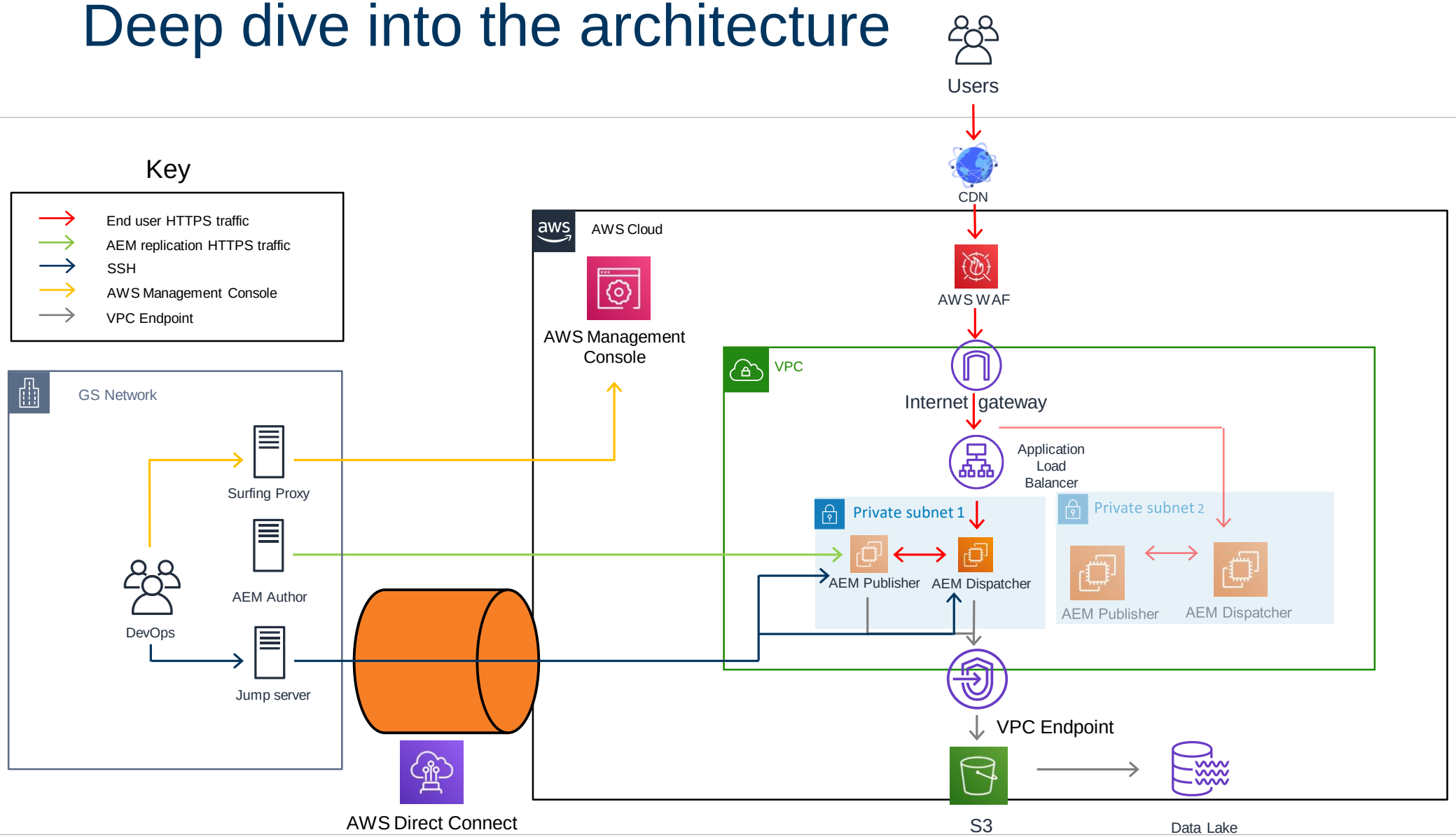
Security First

- Vulnerability scanning (Qualys)
- Protection against attacks (e.g. AWS WAF)
- Real-time infrastructure monitoring
- Encryption
- Identity Management

High-level architecture



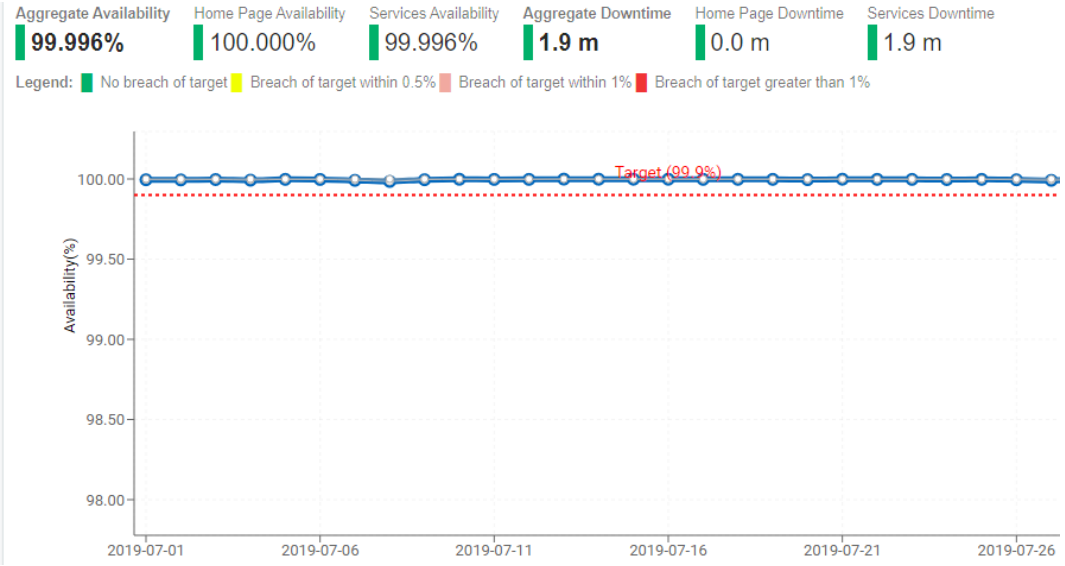
Deep dive into the architecture



We did it!



- Took four months
- Within two weeks of release, it was used
- Allowed us to achieve 99.99 availability



Chapter Two

Achieving parity: Research portal on AWS



A long way to go

Despite our joy, the client experience was not ideal. While it kept us in business, we had a long way to go…

- Authentication
- Authorization
- Financial numbers
- Personalization
- Client preferences

Managed Services: Implementation considerations

- Reduced operational overhead
- Maximized time to market
- Explicit SLAs

- Tied to the provider
- Lose finer control
- Cost

We made the decision to rely on **managed services,** which required re-engineering:

- Routing mechanism - **Amazon API Gateway**
- Search & indexing – migrated to **Amazon Elasticsearch Service**
- Database **– Amazon Aurora**

Design decision: Serverless vs. Containers

Serverless

- ✓ Minimal infrastructure provisioning
- ✓ Lighter SDLC
- ✓ Time to market
- ✗ Can get complex quickly
- ✗ Testing is harder
- ✗ Can’t share code base with on-premises

Containers

- ✓ Portability
- ✓ Suitable for complex applications
- ✓ Time to market
- ✗ Harder to set up and maintain
- ✗ Operational overheard
- ✗ Harder to scale

We used both

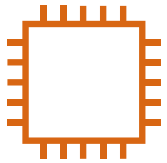
Simple workflow, ETL, triggers

Third-party products or existing services

Design decision: A hybrid approach



AWS Fargate



No instances to manage



Automatic scaling



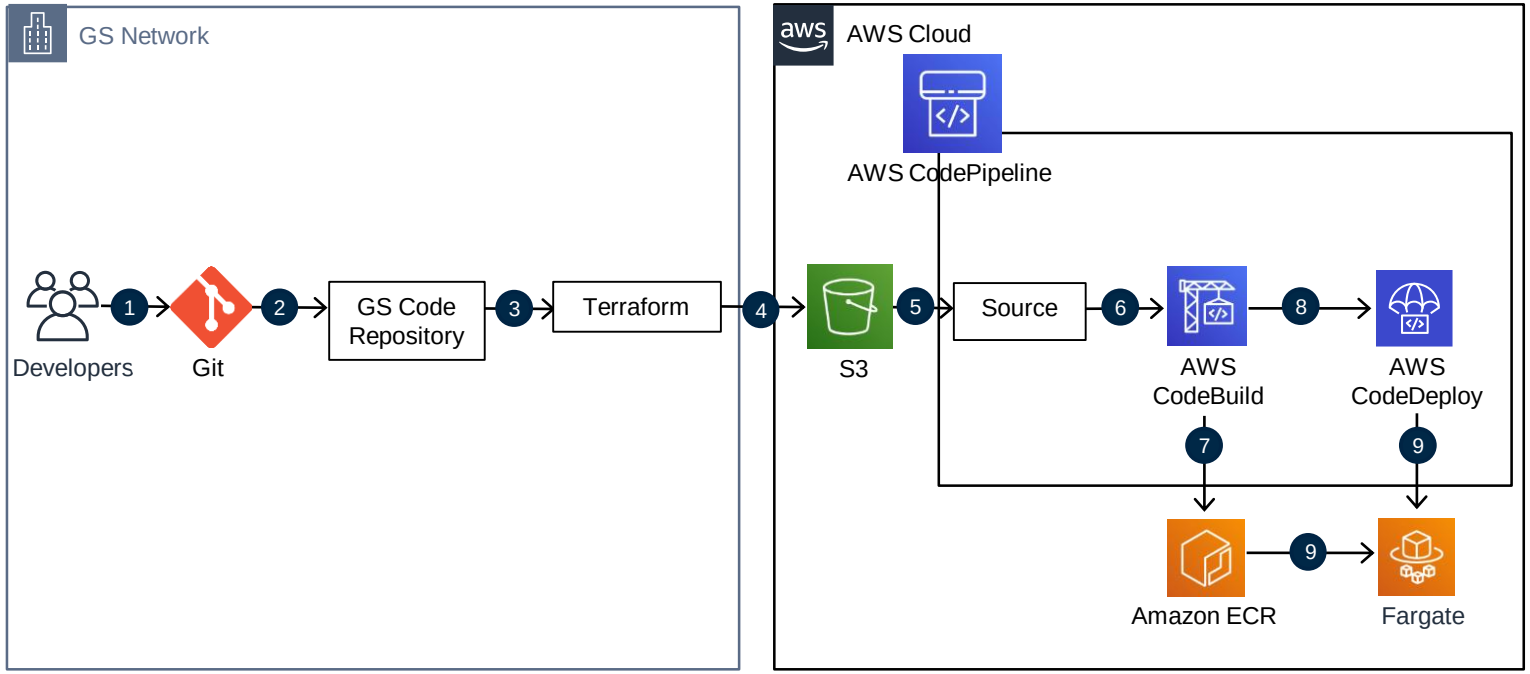
Portability/ one code base



Deep integration with AWS offerings

Software Deployment Lifecycle with AWS Fargate

1. Developers push their code to Git
2. Code artifact to Code Repository
3. Deployment is authorized
4. Artifact is released to S3
5. S3 upload triggers CodePipeline
6. Once the code is pulled, CodeBuild stage is triggered
7. CodeBuild pushes the built image to Amazon ECR
8. CodePipeline progresses and triggers CodeDeploy stage
9. CodeDeploy creates new tasks, verifies health, redirects traffic and removes previous task



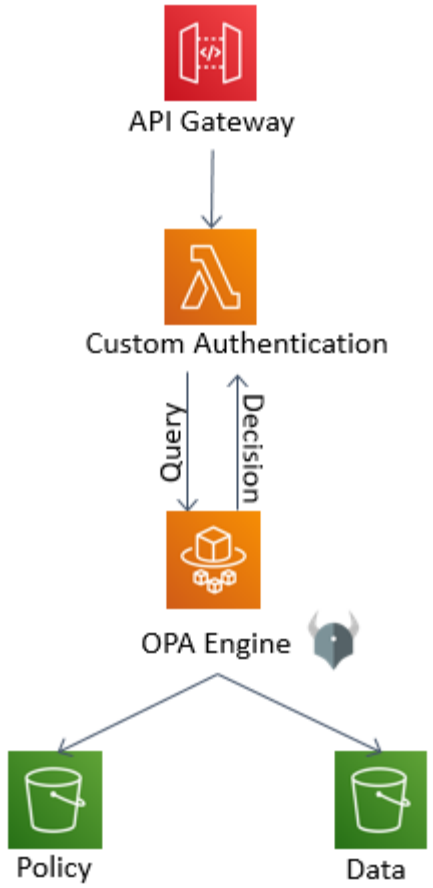
Design decision: Open Source

Used Open Source on AWS & on premises

Example:

- Open Policy Agent, an open source policy engine

- Integrates well with AWS services
  - Exposed via API Gateway
  - We hosted it as a service on Fargate
  - Integrated it with CloudWatch
  - Built with AWS CodeDeploy



Entitlements with Open Policy Agent

A way to **define** and **enforce** rules that read:

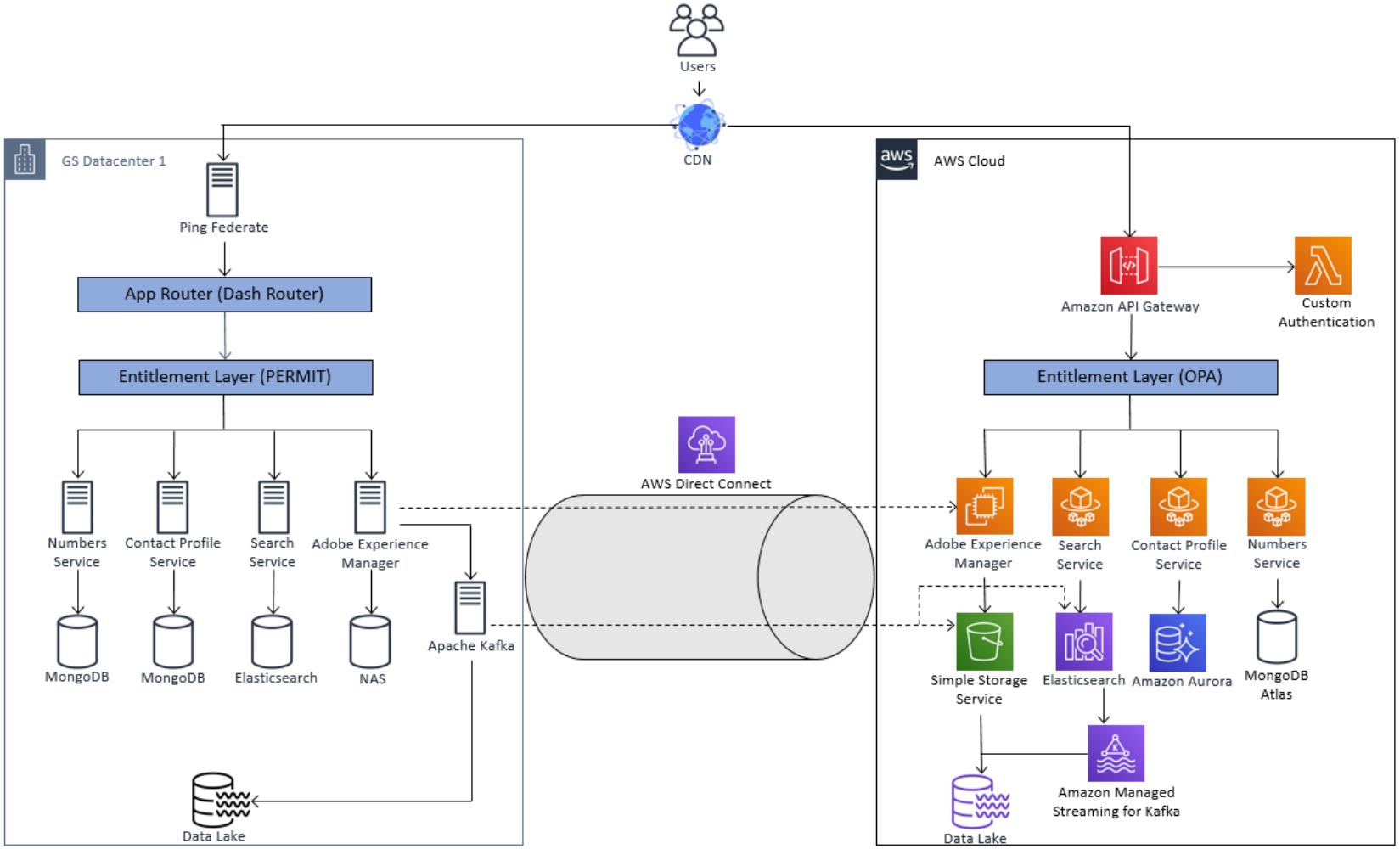
Identity **I**
Can/cannot perform Operation **O** on Resource **R**
For ALL combinations of **I, O, and R** in the ecosystem

OPA features:

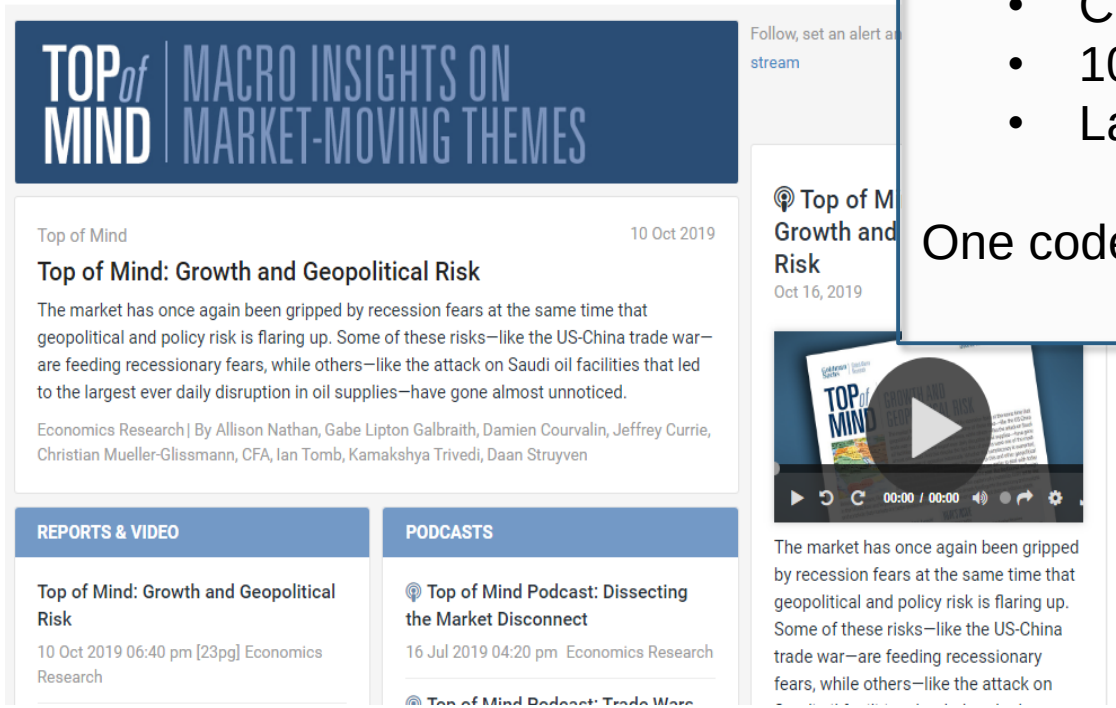
- Open source, general-purpose policy engine
- Declarative Policy Language (Rego)

  Can user X do operation Y on resource Z?
- Can run as a library, sidecar, host-level daemon
- Management APIs for control and observability
- Tooling to build, test, and debug policy

High-level architecture



We did it!



With a hybrid approach

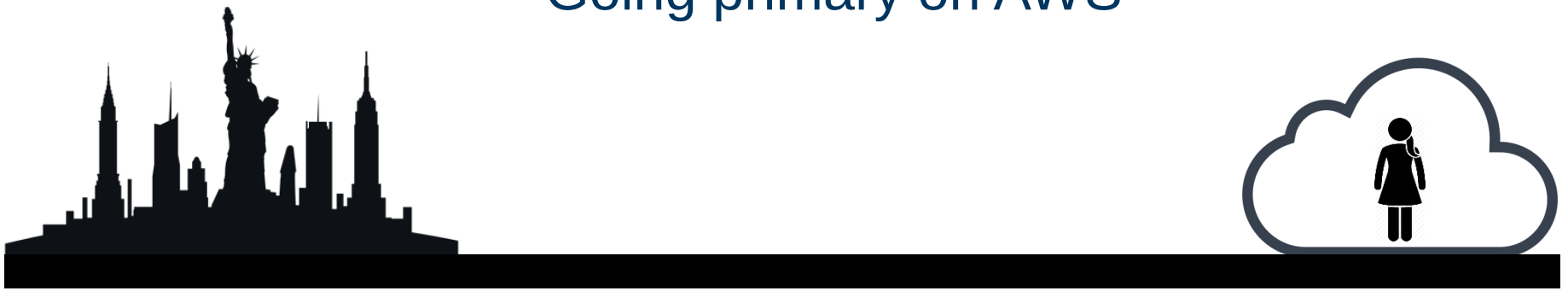
- Containers for third-party products
- 10+ services on Fargate
- Lambda functions for various workflows

One code base shared between on premises and AWS

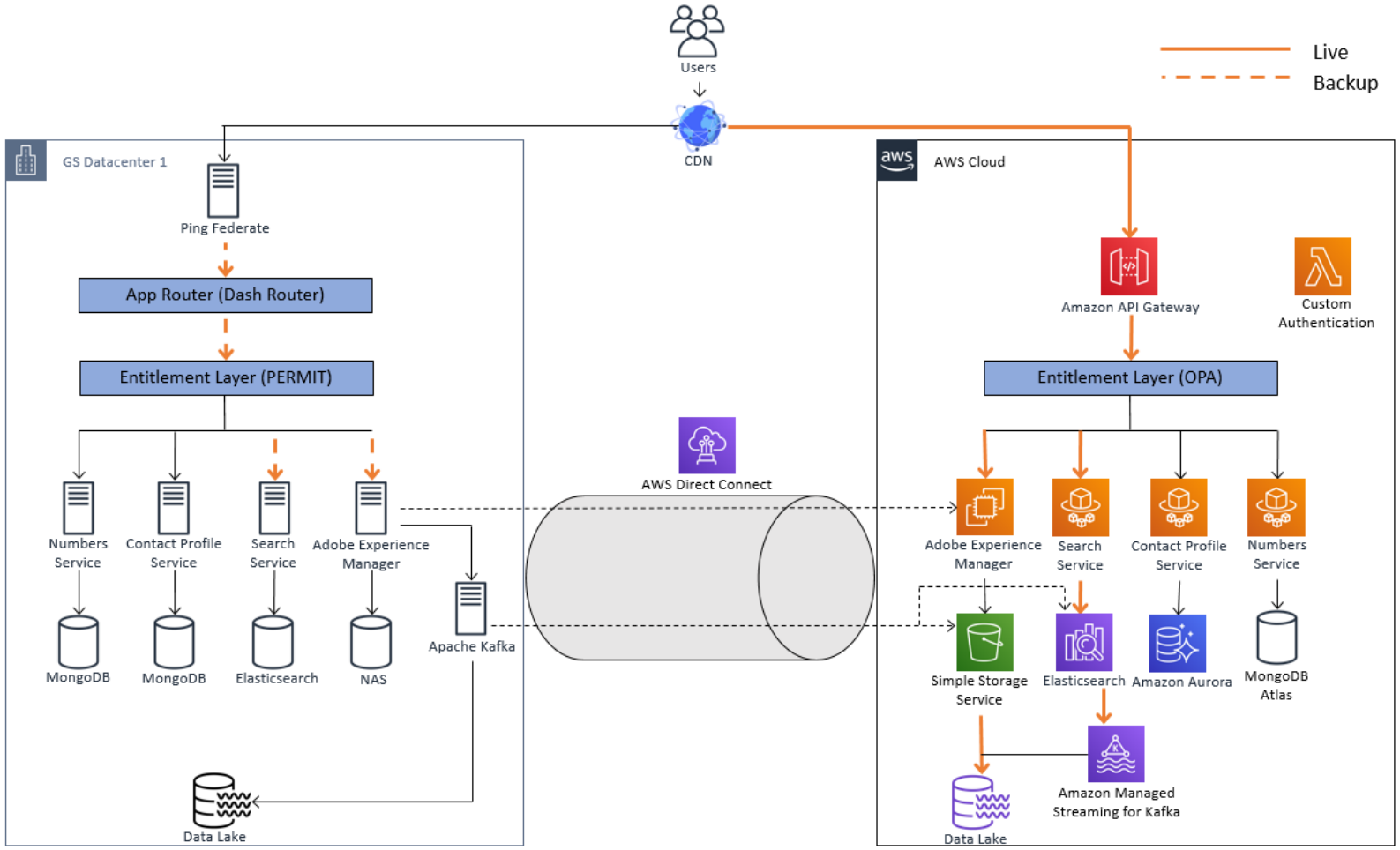
Goldman Sachs

Chapter 3

Going primary on AWS



High-level architecture



Going primary via a three-part journey

1

Select components become primary on AWS



2

Primary on AWS, fail back to on-premises



3

Multi-region on AWS & demise of on-premises



Our checklist

Operational support

Strategy
- Operational Support Framework
- Guidelines on severity of alerts/incidents
- SLIs & SLOs defined

Monitoring & alerting
- Implement workflows with monitoring & alerting tools
- Infra monitoring (CloudWatch + DataDog)
- Application performance monitoring (DataDog + AppDynamics)
- Alerting & escalation (PagerDuty)
- Support dashboards
- Measure against SLIs & SLOs

SDLC

SDLC
- Migrate to GitLab in parallel account
- Enable dynamic scalability
- Regular re-paving of environment
- Blue/Green & canary deployments

Tests
- Automated smoke tests running regularly
- Automated acceptance tests with each deployment
- Load testing and capacity planning

Security & costs

Cyber Security
- CDN controls
- Additional AWS WAF rules
- Add Signal Sciences
- Auto-remediation rules

Vulnerabilities & patching
- Qualys – vulnerability scanning
- AMI Build Factory
- AWS Managed Services patching
- Asset inventory

Risk
- Address tech risk issues
- Non-GIR dependencies

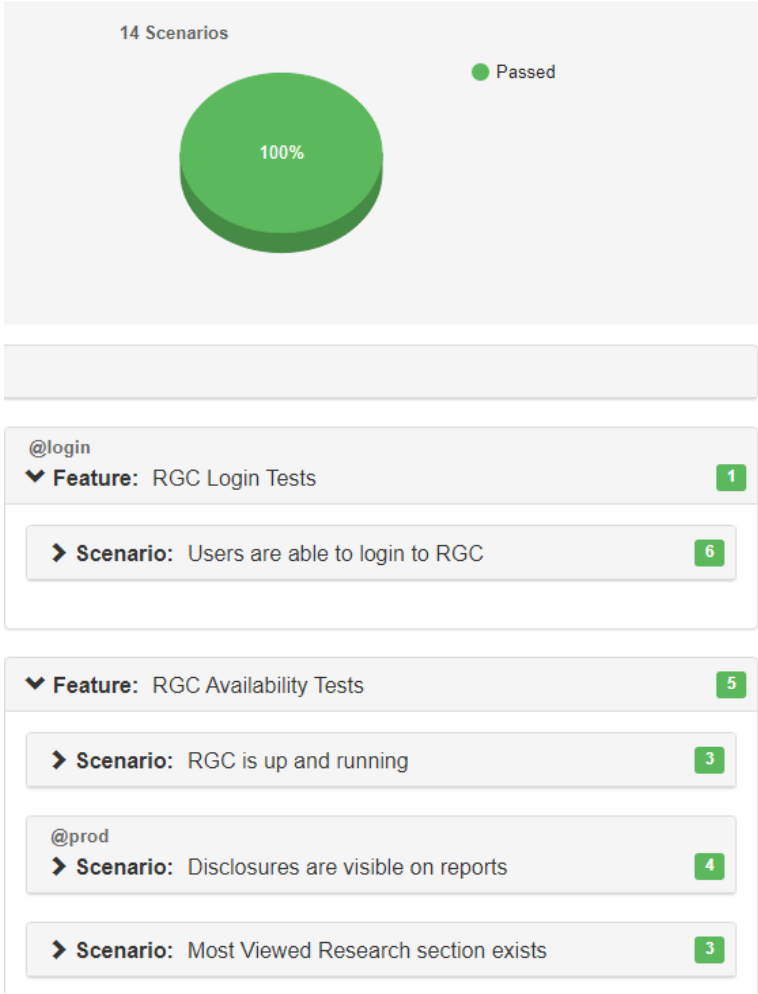
Costs
- Design with cloud costs in mind
- Provide transparency of costs to Dev teams

Zero BAU production access

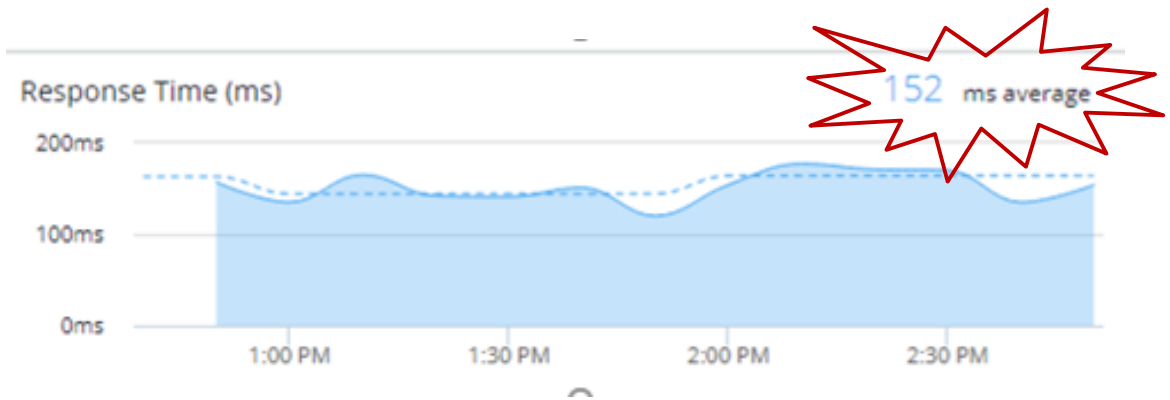
Strict ZERO production access policy

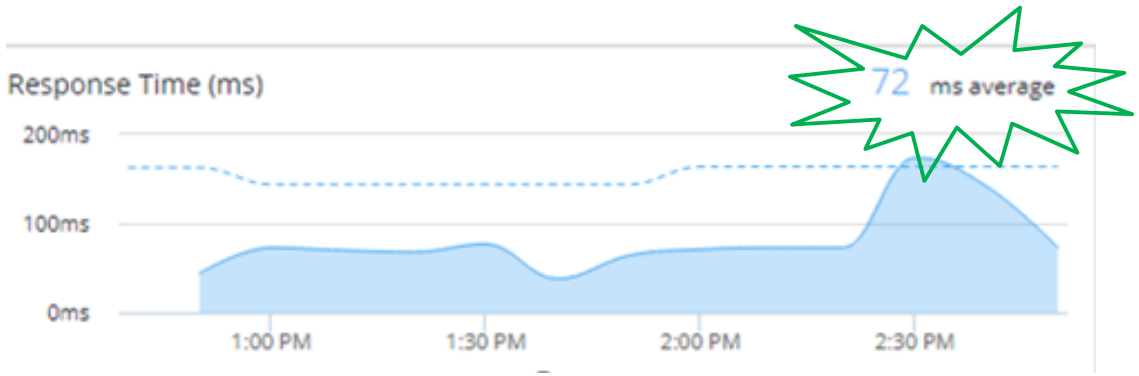
How:

- Automated smoke tests run constantly
- Library of pre-approved scripts
- Investment in tooling
- Blue/green deployments

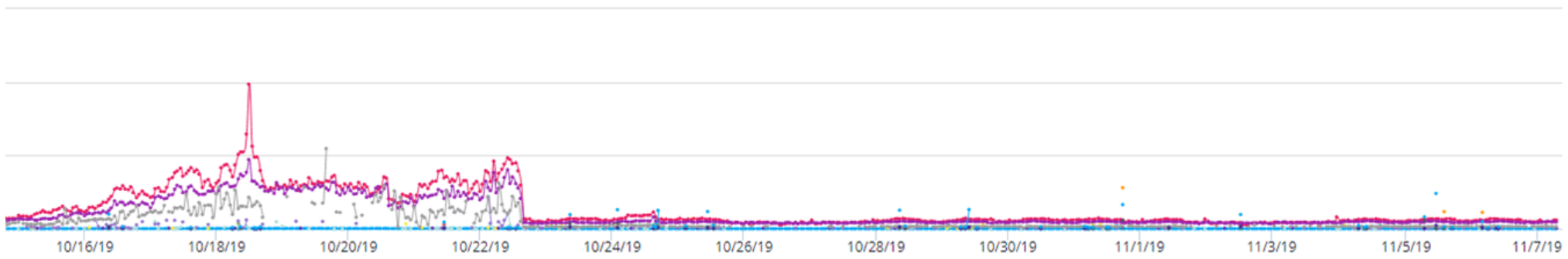


Positive results came very soon

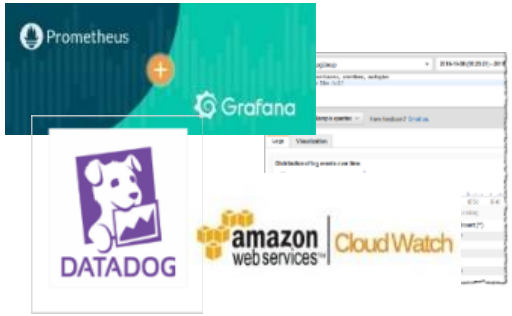




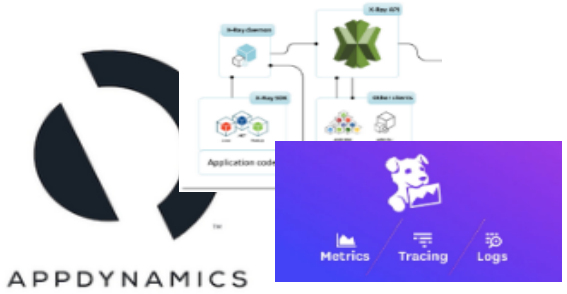
Response time after switching a service to AWS



Operational and security tooling



Infrastructure monitoring



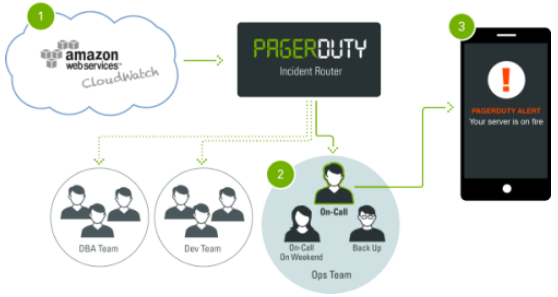
Application performance monitoring



Security monitoring



Centralized logs



Alerts and escalation

Goldman Sachs

Epilogue

Guiding Principles



Design

- Leverage managed services or open source products
- Adopt a cloud-native approach
- Build security into the design upfront



SDLC

- Infrastructure as Code
- Reusable modules to deploy artifacts from GS to AWS
- CI/CD



Operations

- Zero Production Access
- Blue/green deployments
- Regular environment repaving
- Clearly defined service levels

Wrapping up: It was more than just code

Cultural philosophy



Practices & patterns



Tools

Goldman Sachs

Migrating to AWS: Lessons learned

- Invest early in tooling for support and debugging…we did it too late
- Refresh your computer networking knowledge!
- Stay focused on your **end goal** and don't get distracted by plethora of AWS services…but stay open to experimentation
- Ok to fail…but fail fast and recover faster
- Recognize that all your security requirements are not pre-built and require proactive customization
- You will have quick wins but getting to **DONE** is a marathon

Migrating to AWS: What is next?

Getting to one city!

- 100% of Research traffic through AWS
- Multi-region deployments for resiliency and client experience
- Migrate additional Research capabilities to AWS
- Cloud-first approach for new services
- Demise on-premises infrastructure

Thank you!

Vicky Ledda
Vice President
Global Investment Research Engineering
Goldman Sachs

Yun Zhang
Vice President
Core Engineering
Goldman Sachs

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey in the mobile app.

© 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.