Modern Application Design: Building effective services

Shaun Ray
Senior Manager, Developer Evangelism, AWS

Agenda

1. **The Business Case for Modern Applications**

2. **Architectural Patterns**
   - Monoliths and microservices
   - Event driven systems
   - Workflow orchestration and state

3. **Operating Model**
   - Deployment Model
   - Choosing the right abstraction
   - Distributed Tracing and Monitoring

4. **Software Delivery**
   - Pipelines
   - Deployment

aws

Modern Applications – The Business Case

The new normal: companies are increasingly global and products are increasingly digital

47%

of CEOs said they are being challenged by the board of directors to make progress in digital business

79%

of CIOs believe that digital business is making their IT organizations better prepared to change

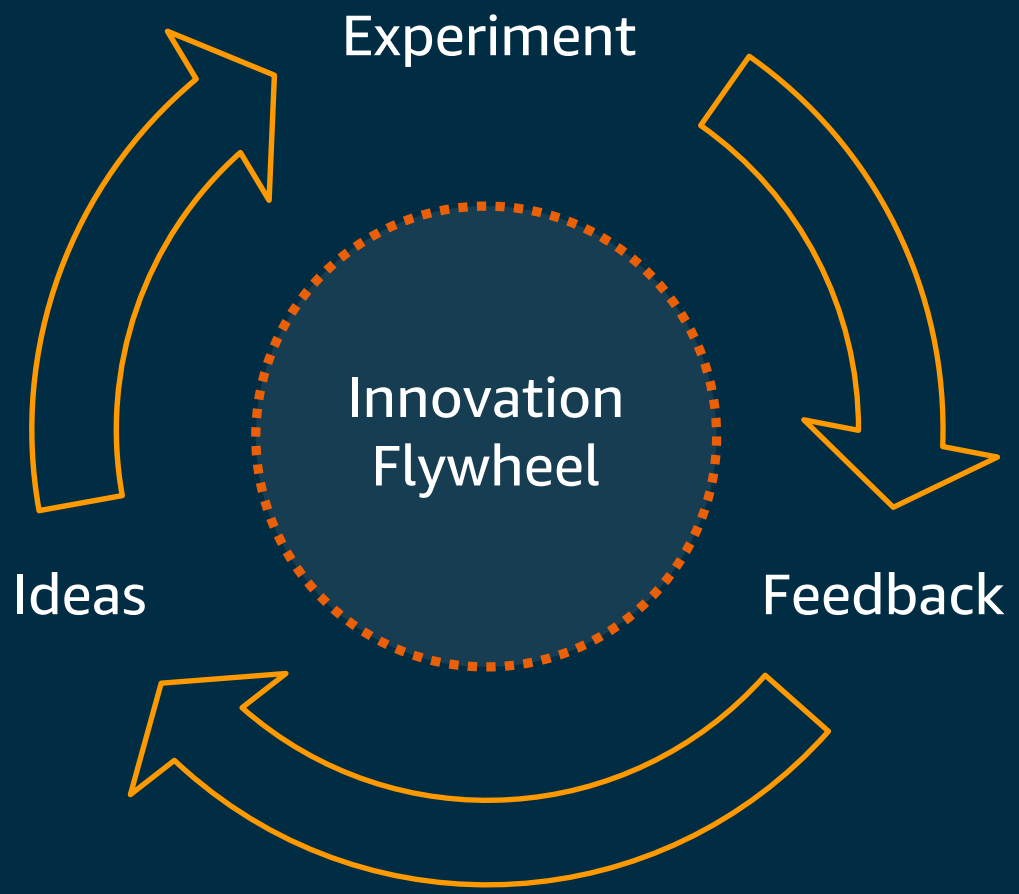
67%

of all business leaders believe that they must pick up the pace of digitalization to remain competitive

Source: Gartner

aws

To maintain competitive advantage, digital businesses must innovate as rapidly as possible



AWS customers are pioneering modern applications

reduced overall compute costs by 95%

cut processing time from 36 hours to 10 seconds

created a stock trade validation system in 3 months

releases over 50+ deployments per hour

aws

Modern applications

Built on containers
and serverless

Microservices architecture
and distributed

Each component is
autonomous and independent

See impact of
isolated errors

aws

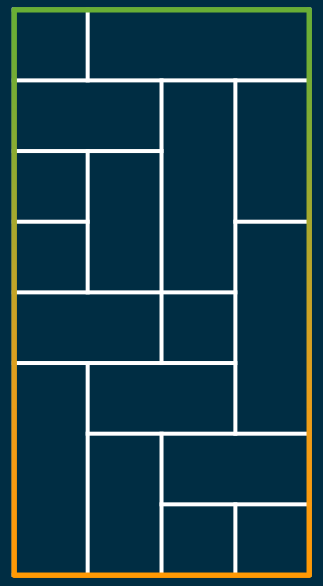
What changes have to be made in this new world?

Architectural patterns

Operational model

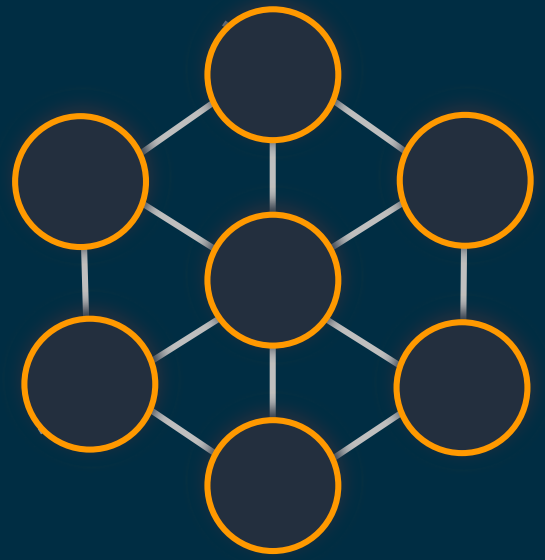
Software delivery

Architectural Patterns



Monolith

Does everything



Microservices

Does one thing

Architectural Concerns

Monolith

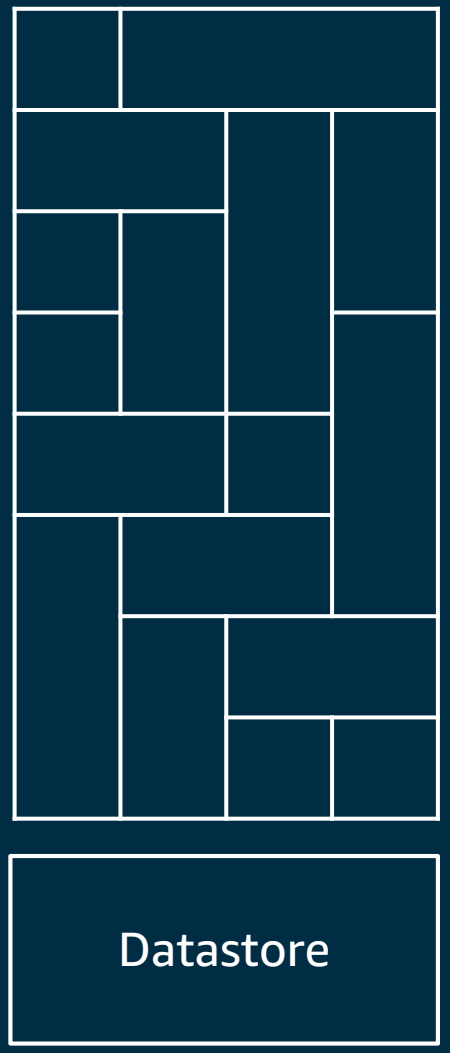
Single Artifact

Shared Datastore

Shared Interface

Shared Logging

Single Deployment



Microservice

Application Code

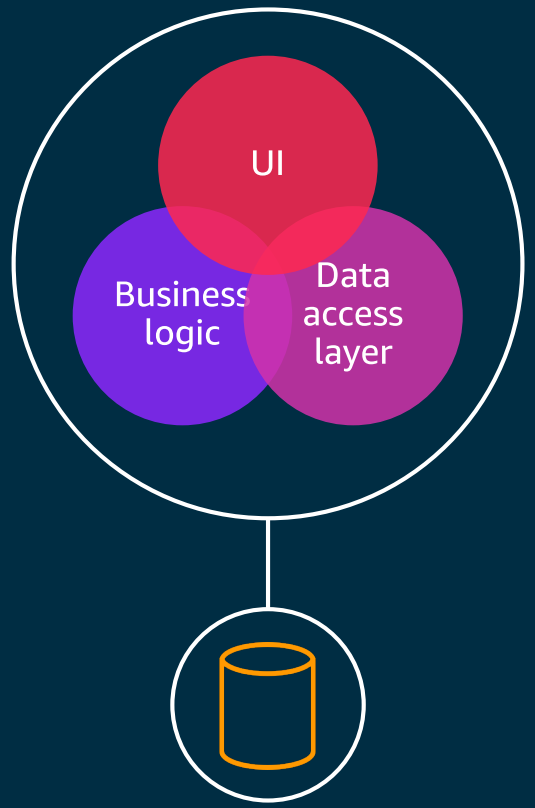
Datastore

Interface

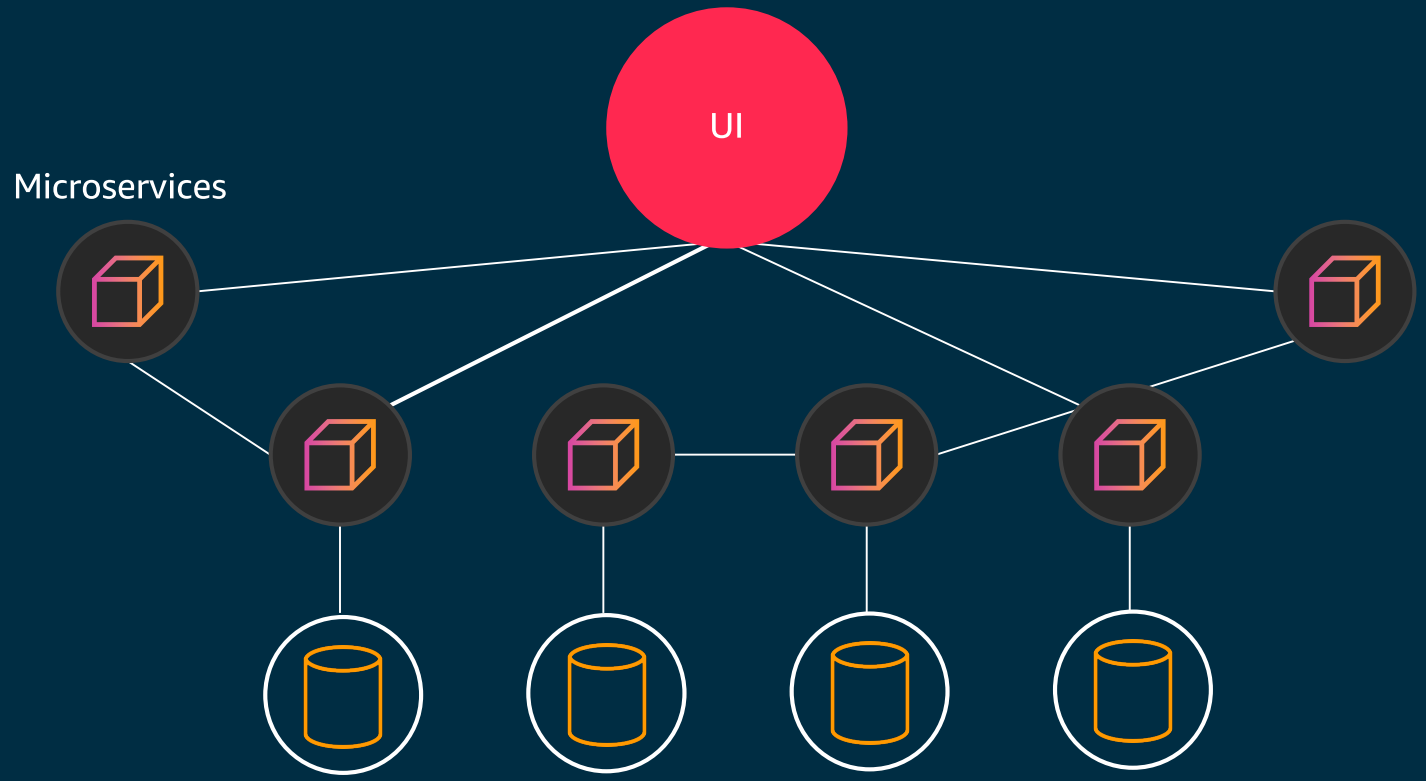
Logging

Deployment logic

Architecture Choices

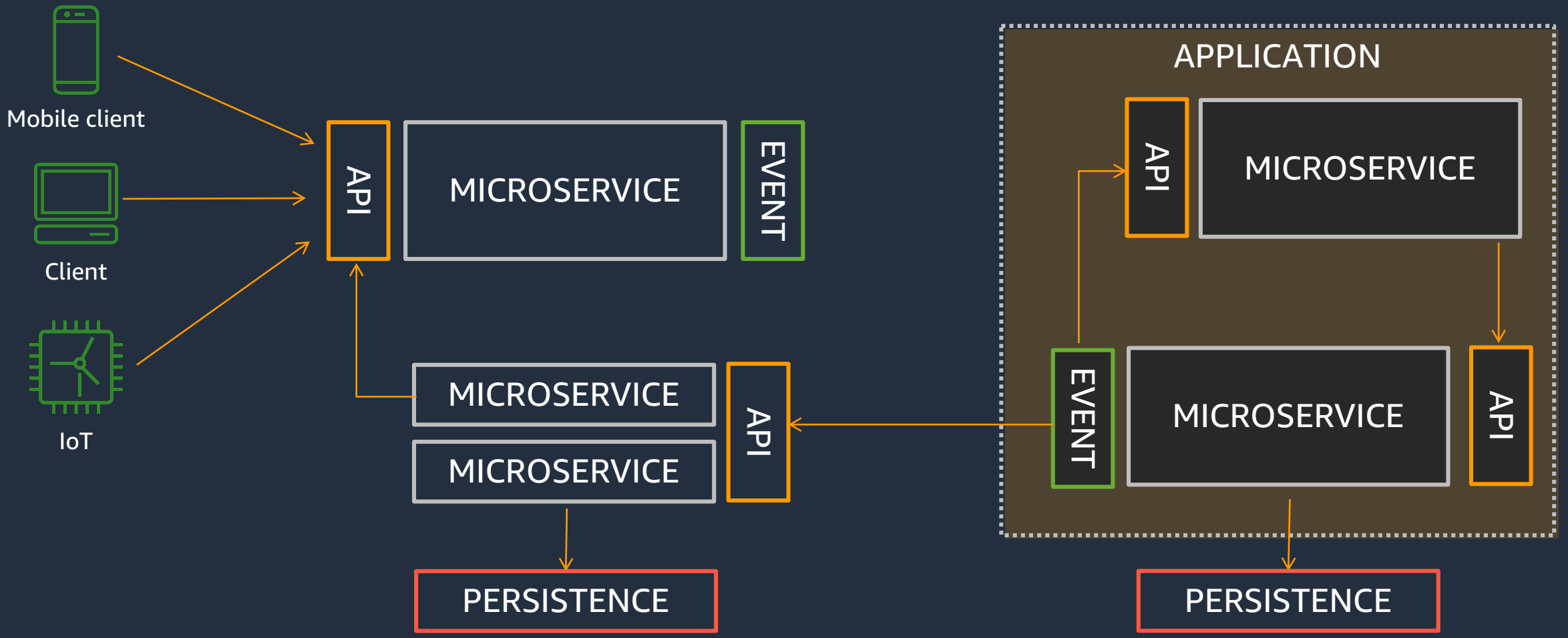


Monolithic architecture

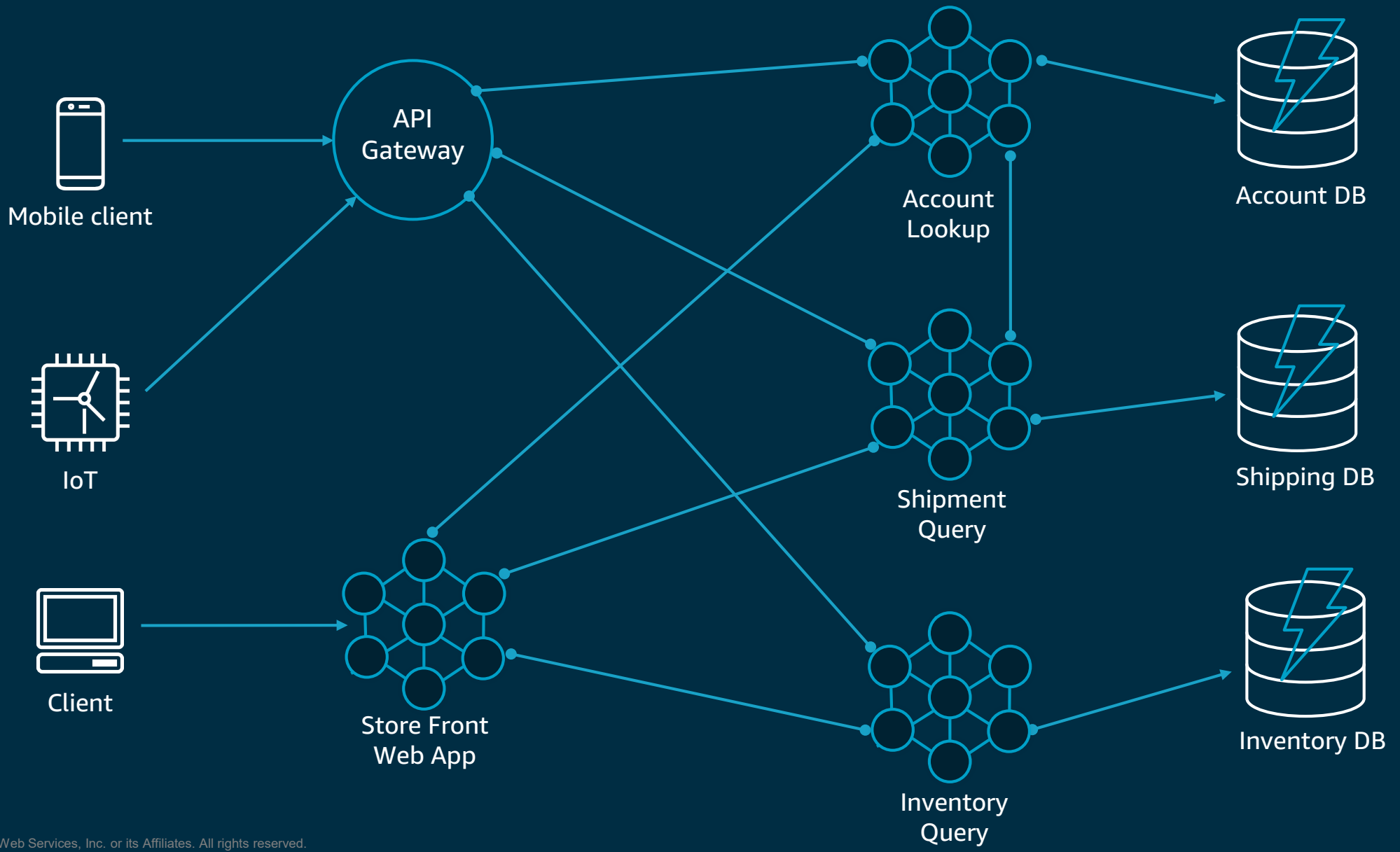


Microservices architecture

Microservices architectures



Monoliths and Microservices

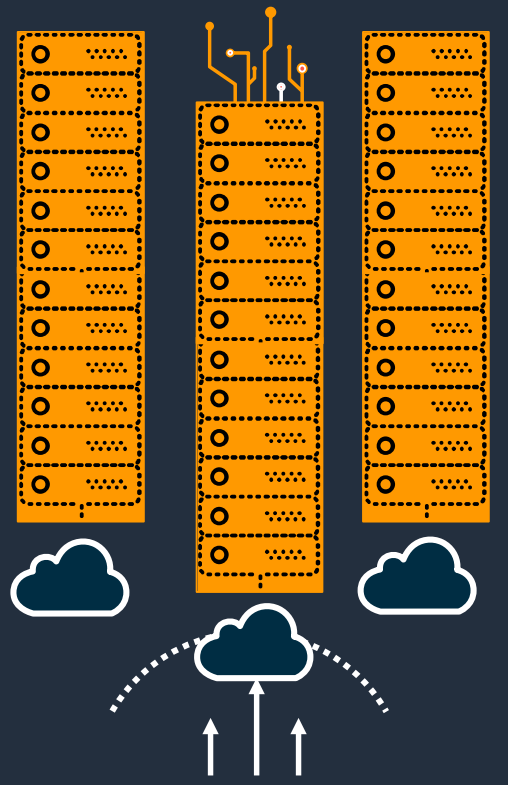


Let’s take a look at the evolution of computing

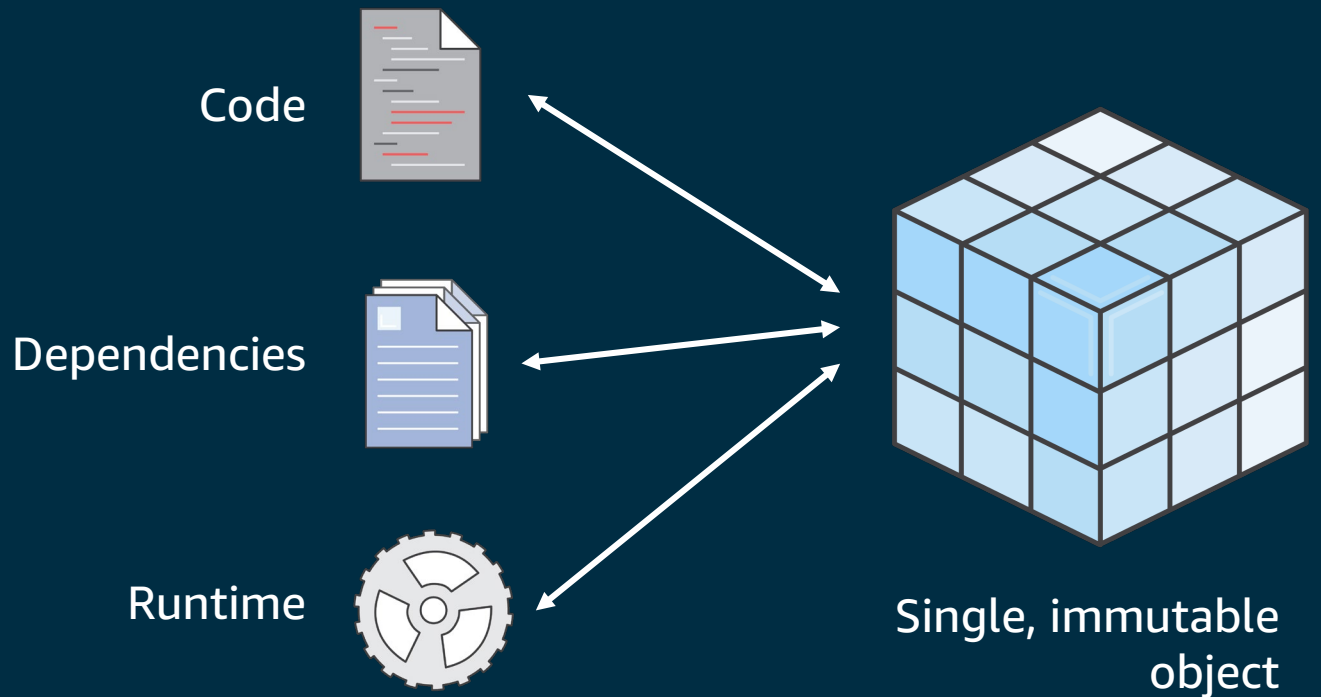
- Higher utilization
- Faster provisioning speed
- Improved uptime
- Disaster recovery
- Hardware independence

- Trade CAPEX for OPEX
- More scale
- Elastic resources
- Faster speed and agility
- Reduced maintenance
- Better availability and fault tolerance

Virtual Servers in the Cloud



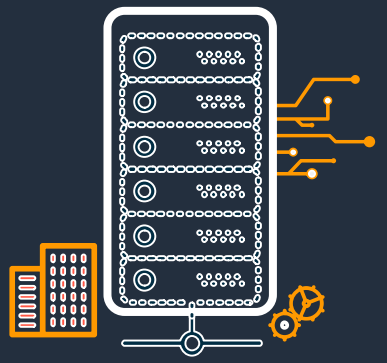
What is a container?



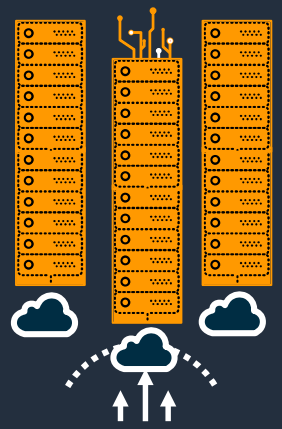
Evolving to Serverless

Physical Servers in Datacenters

Virtual Servers in Datacenters



Virtual Servers in the Cloud



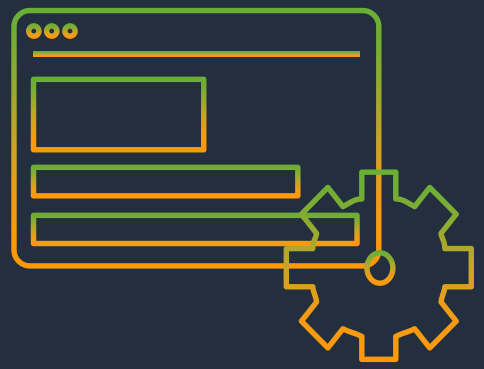
Lambda

Simple Notification Service

Step Functions

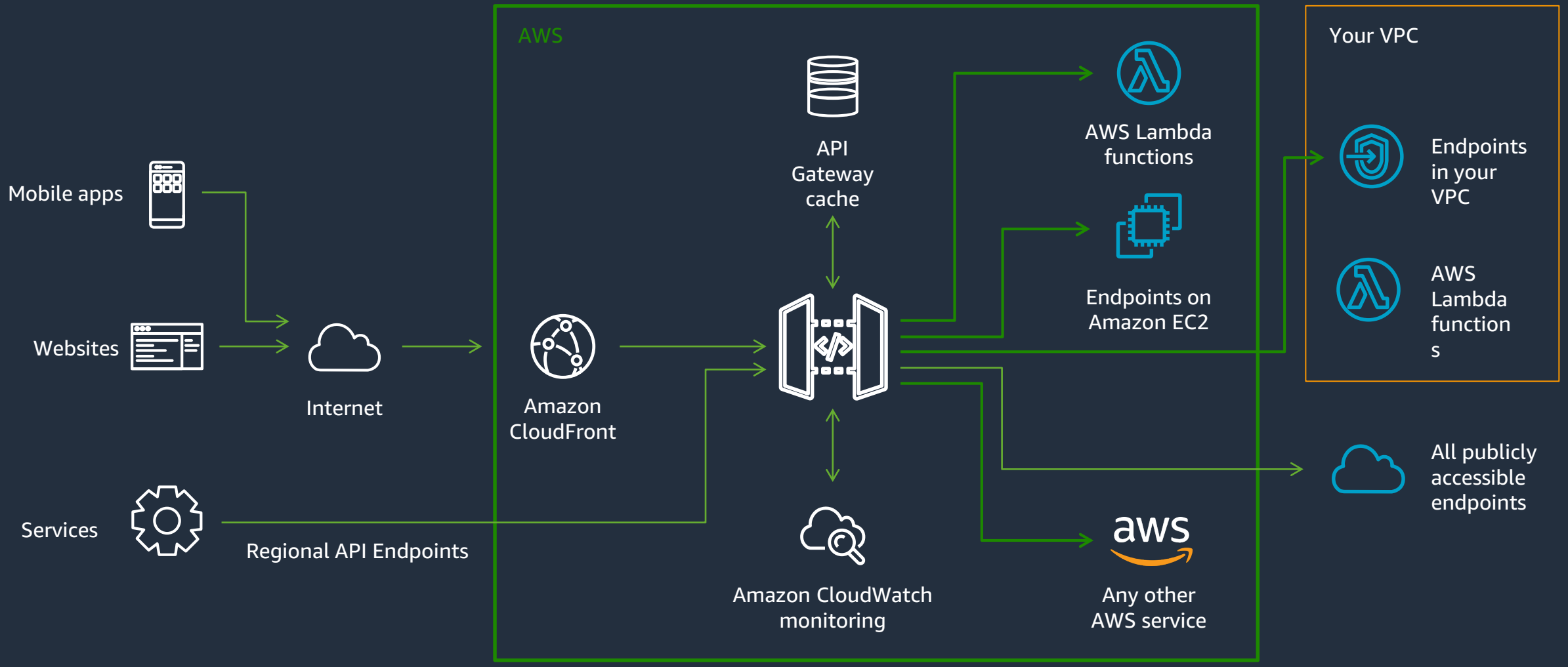
Simple Queue Service

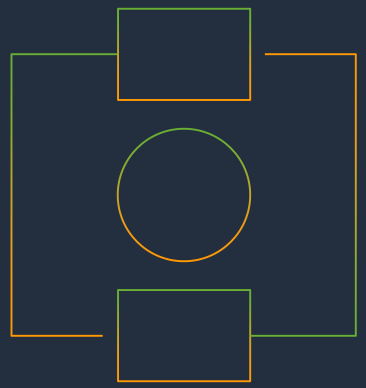
API Gateway



APIs are the front door of microservices

Manage APIs with API Gateway

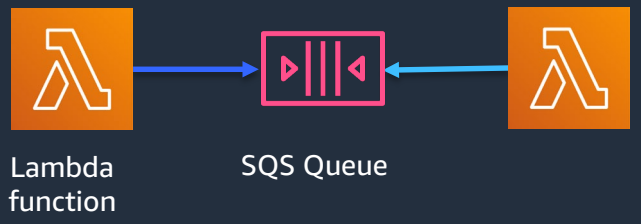




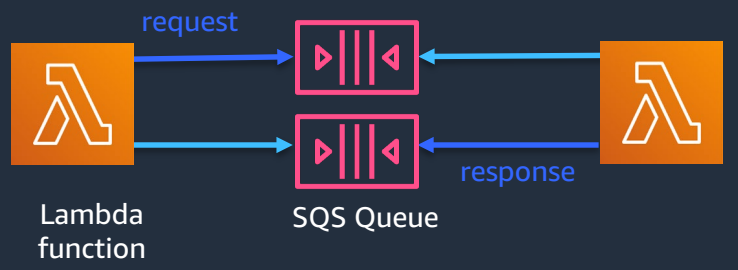
Event-driven architectures

Microservices messaging patterns

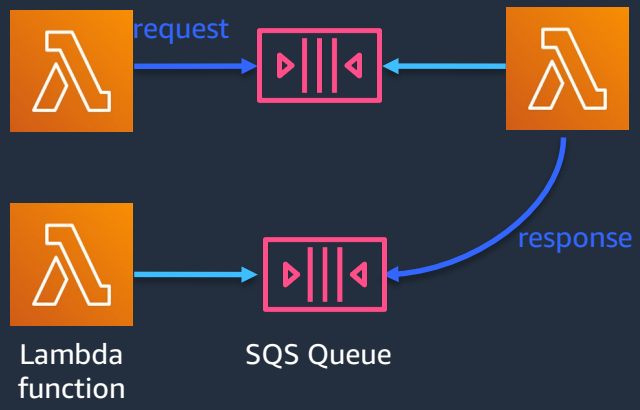
One-Way



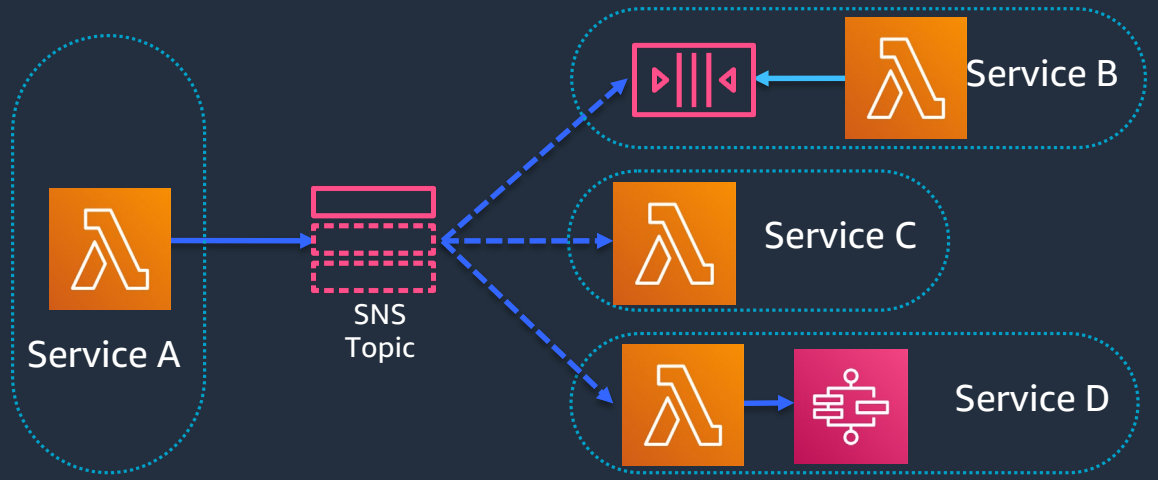
Request / Response



Return Address

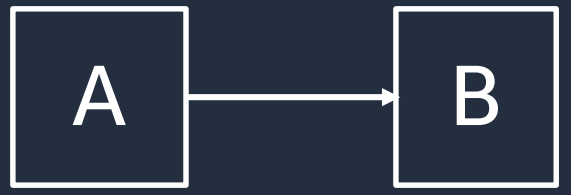


Publish / Subscribe

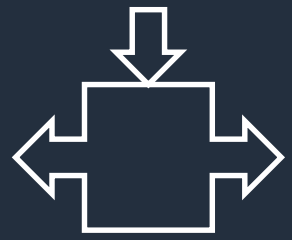


“I want to
retry failed tasks”

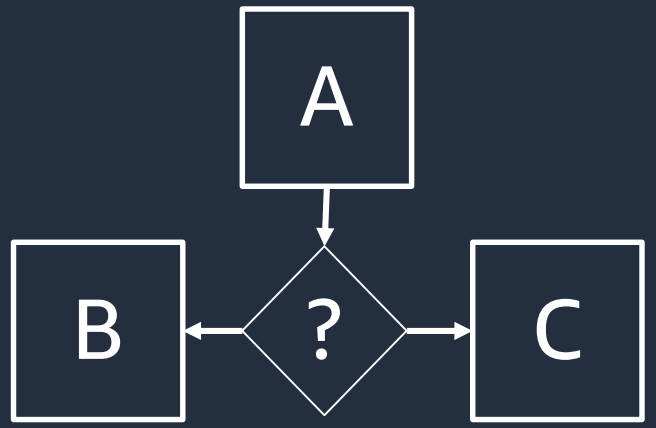
“I want to
sequence tasks”



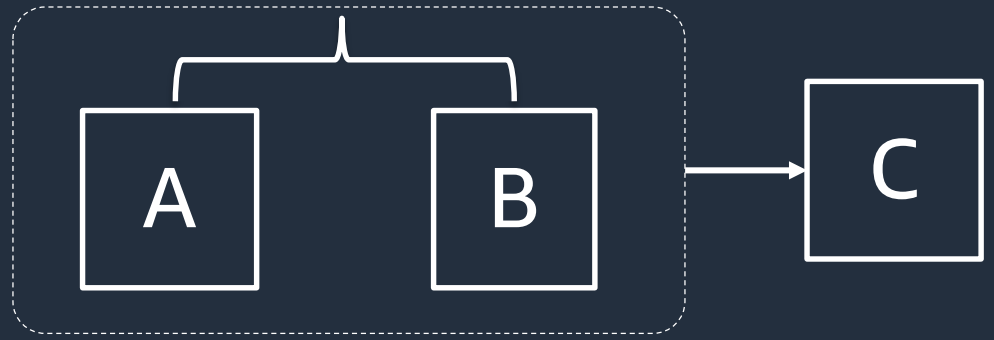
“I want
try/catch/finally”



“I want to
select tasks based on data”

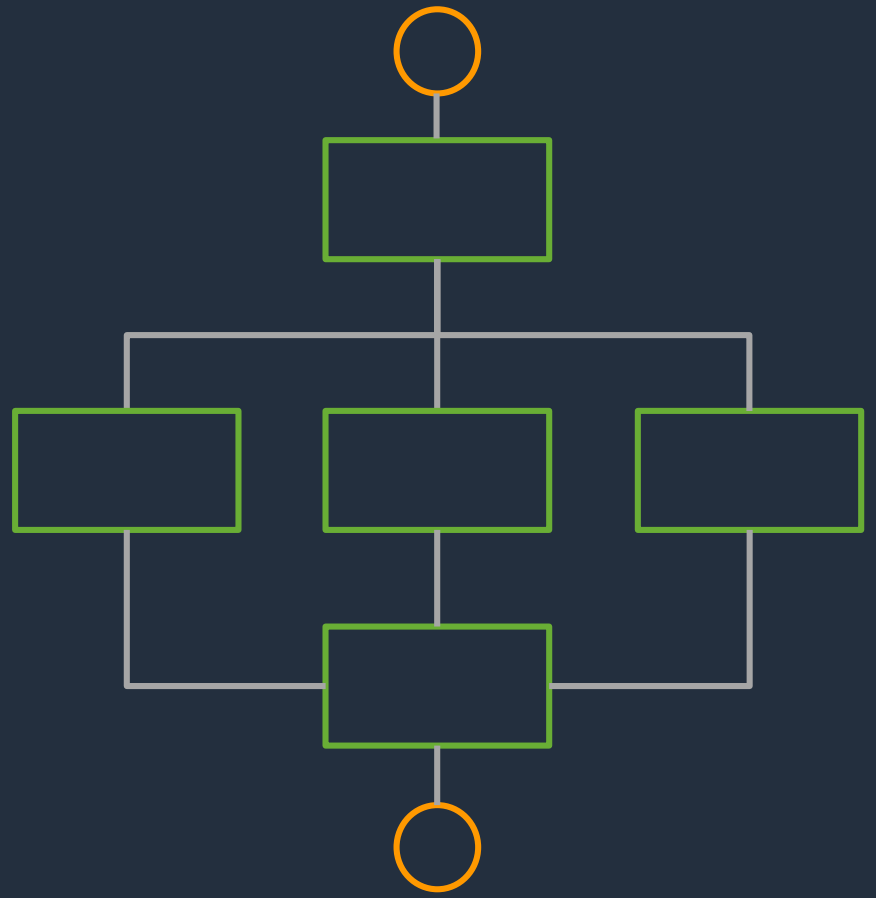


“I want to
run tasks in parallel”



Build workflows to orchestrate everything

Track status of data and execution

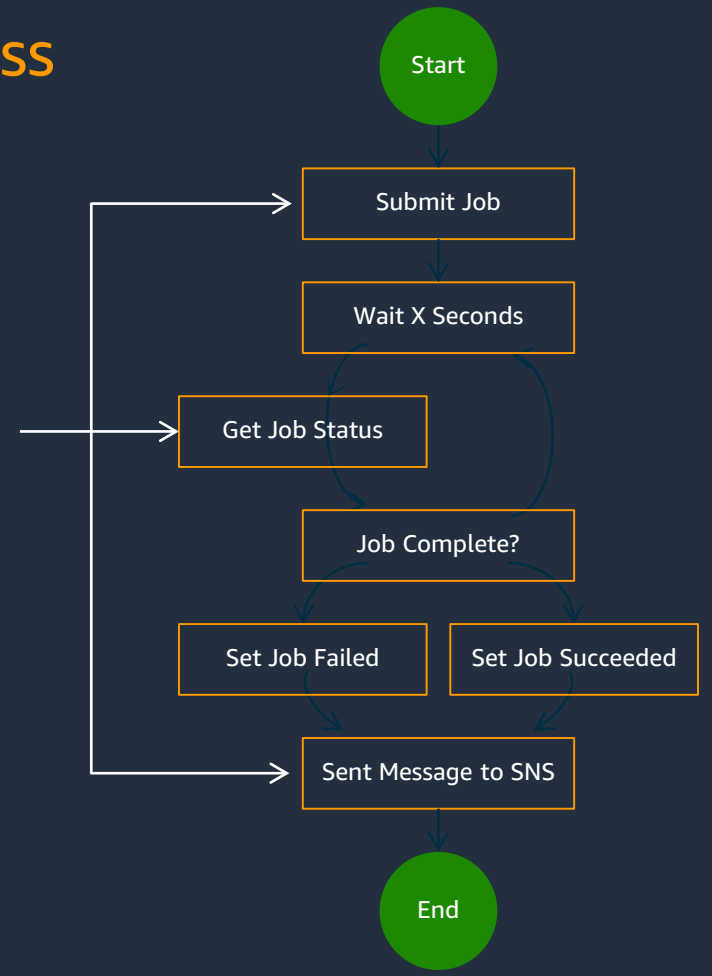


Remove redundant code

Simpler integration, less code

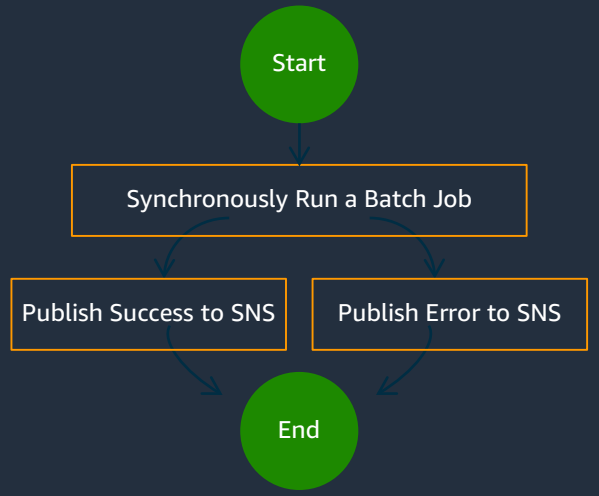
With serverless polling

AWS Lambda functions



With new service integration

No Lambda functions

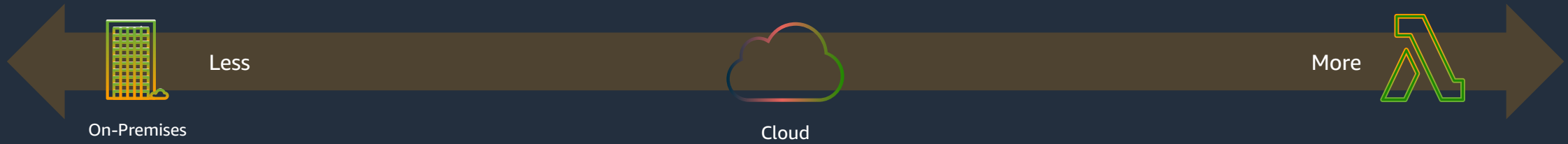


Cloud-native architectures are small pieces, loosely joined

Changes to the operational model

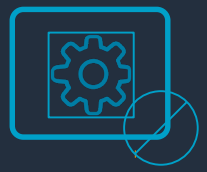
Isn’t all of this very hard now that we have lots of pieces to operate?

AWS operational responsibility models



Compute	Virtual Machine	EC2	Elastic Beanstalk		Fargate	AWS Lambda
Databases	MySQL	MySQL on EC2	RDS MySQL	RDS Aurora	Aurora Serverless	DynamoDB
Storage	Storage					S3
Messaging	ESBs		Amazon MQ	Kinesis		SQS / SNS
Analytics	Hadoop	Hadoop on EC2	EMR	Elasticsearch Service		Athena

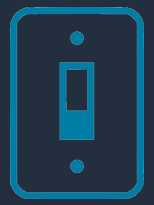
What is serverless?



No infrastructure provisioning, no management

Automatic scaling

Pay for value



Highly available and secure

Serverless is an operational model that spans many different categories of services

COMPUTE


AWS Lambda


AWS Fargate

DATA STORES


Amazon S3


Amazon Aurora Serverless


Amazon DynamoDB

INTEGRATION


Amazon API Gateway


Amazon SQS


Amazon SNS

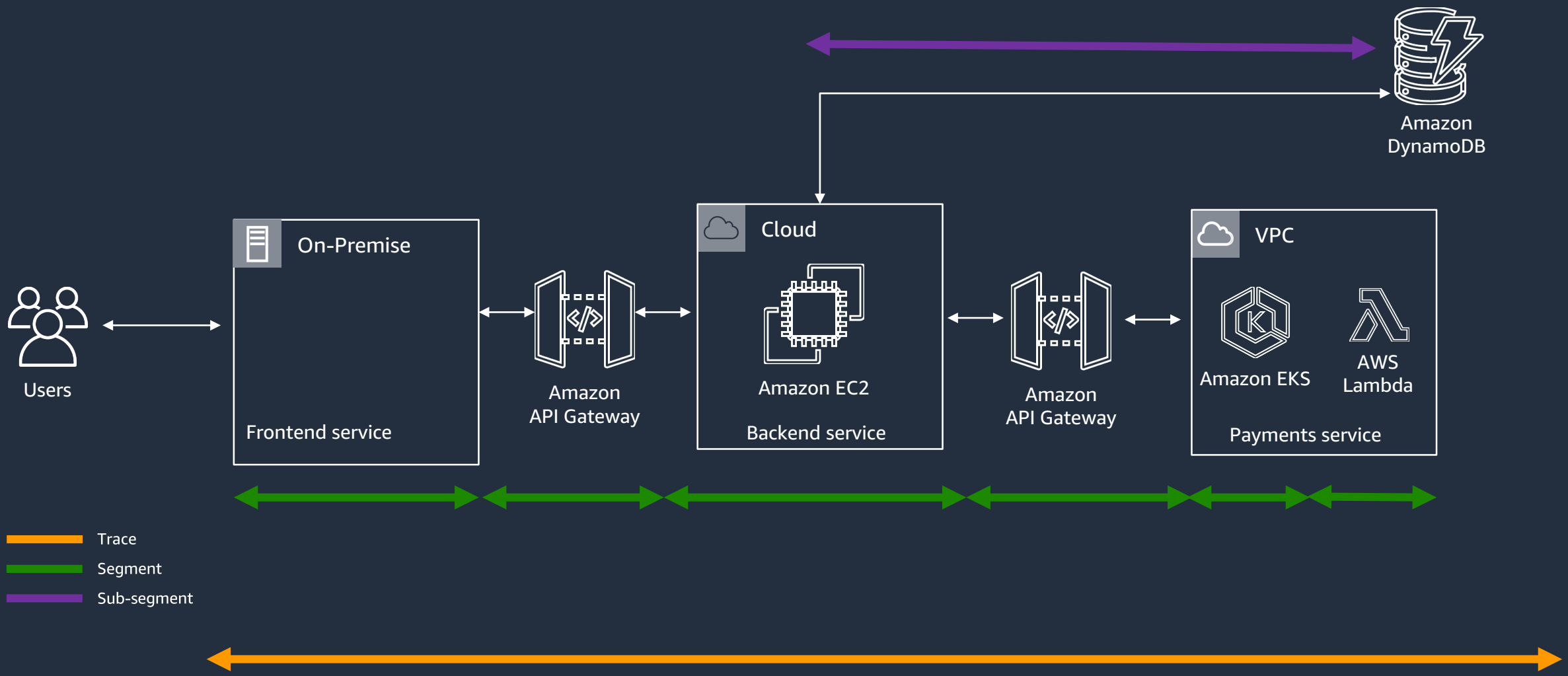

AWS Step Functions


AWS AppSync

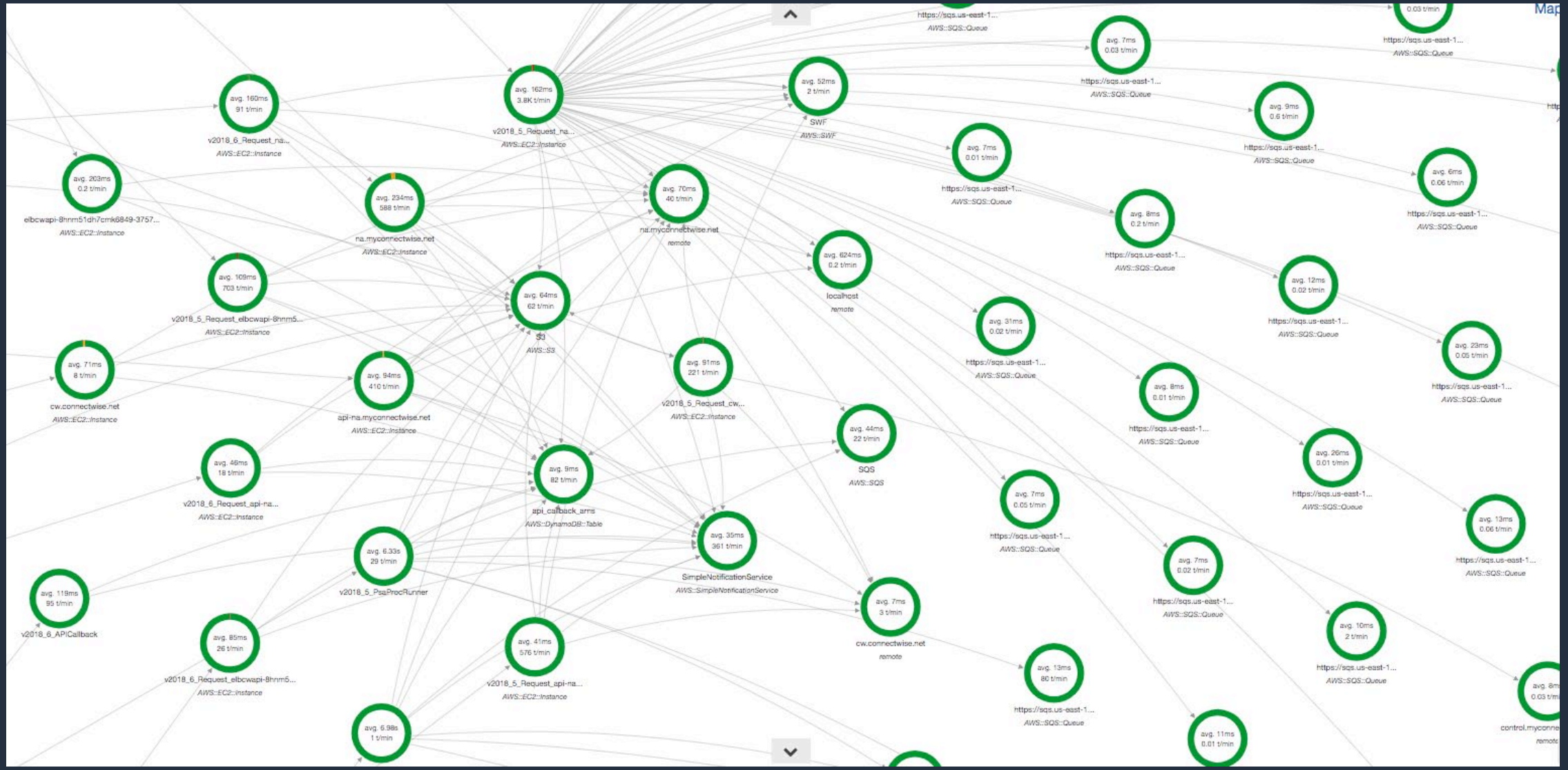
Comparison of operational responsibility

	AWS manages	Customer manages
AWS Lambda Serverless functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate Serverless containers	• Container orchestration, provisioning • Cluster scaling • Physical hardware, host OS/kernel, networking, and facilities	• Application code • Data source integrations • Security config and updates, network config, management tasks
ECS/EKS Container-management as a service	• Container orchestration control plane • Physical hardware software, networking, and facilities	• Application code • Data source integrations • Work clusters • Security config and updates, network config, firewall, management tasks
EC2 Infrastructure-as-a-Service	• Physical hardware software, networking, and facilities	• Application code • Data source integrations • Scaling • Security config and updates, network config, management tasks • Provisioning, managing scaling and patching of servers

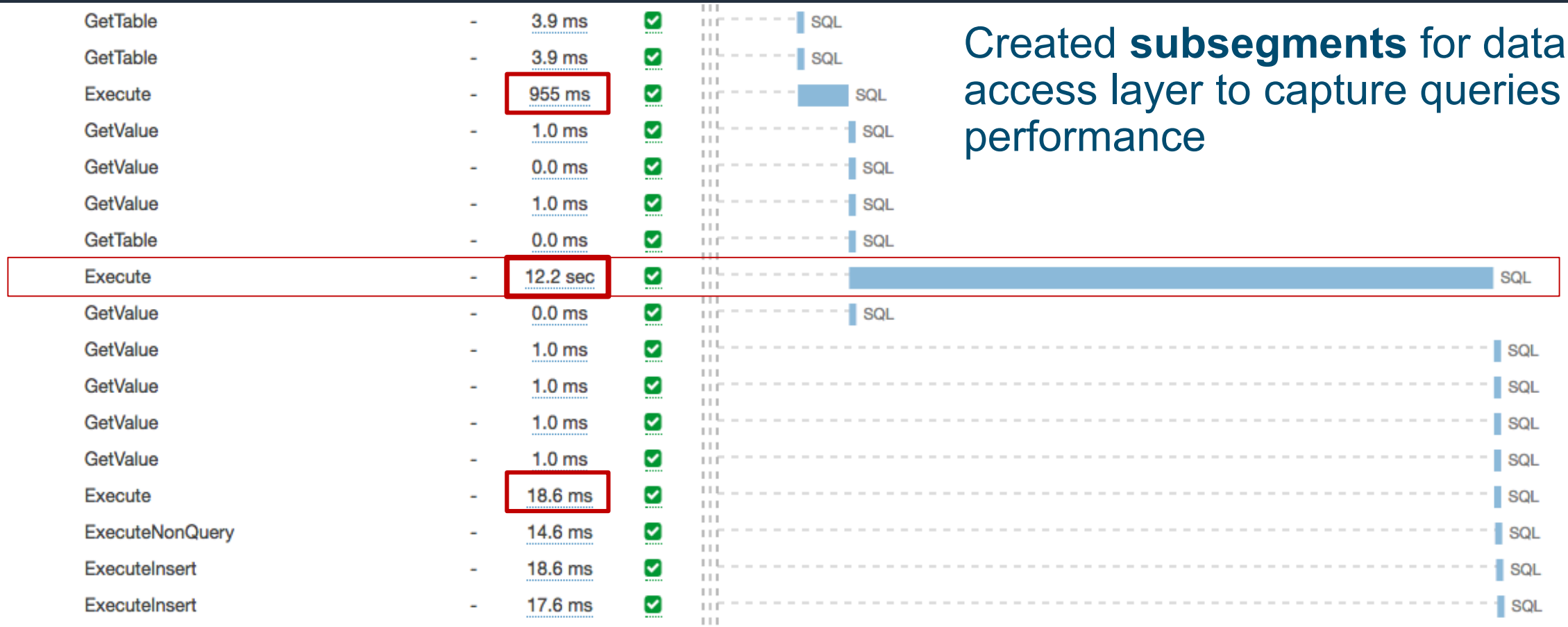
Tracing for your modern application



Real life Distributed Tracing

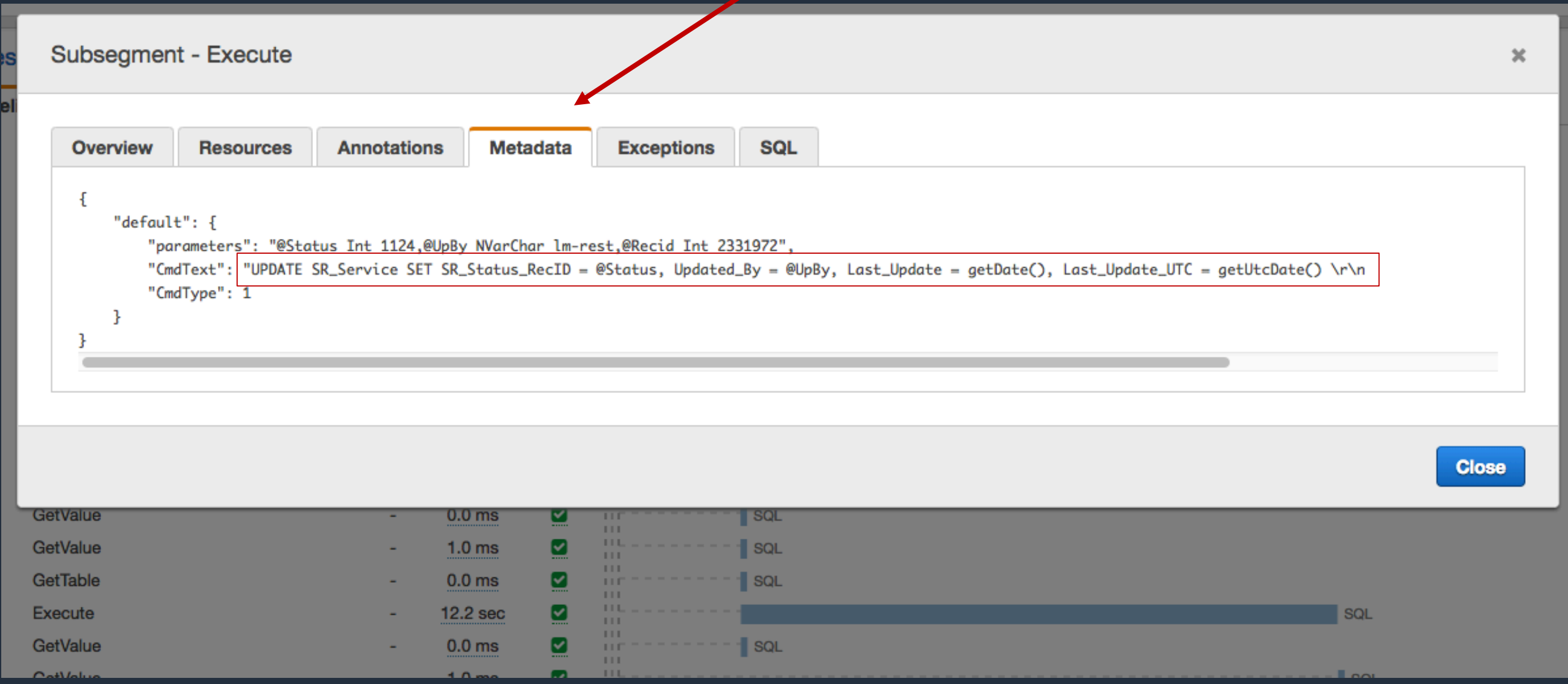


Troubleshooting SQL queries performance



Created **subsegments** for data access layer to capture queries performance

Included the SQL call as a metadata of the segment



SQL performance logs

Build visibility into performance of database calls using X-Ray APIs

aws xray get-trace-summaries --start-time <value> --end-time <value>

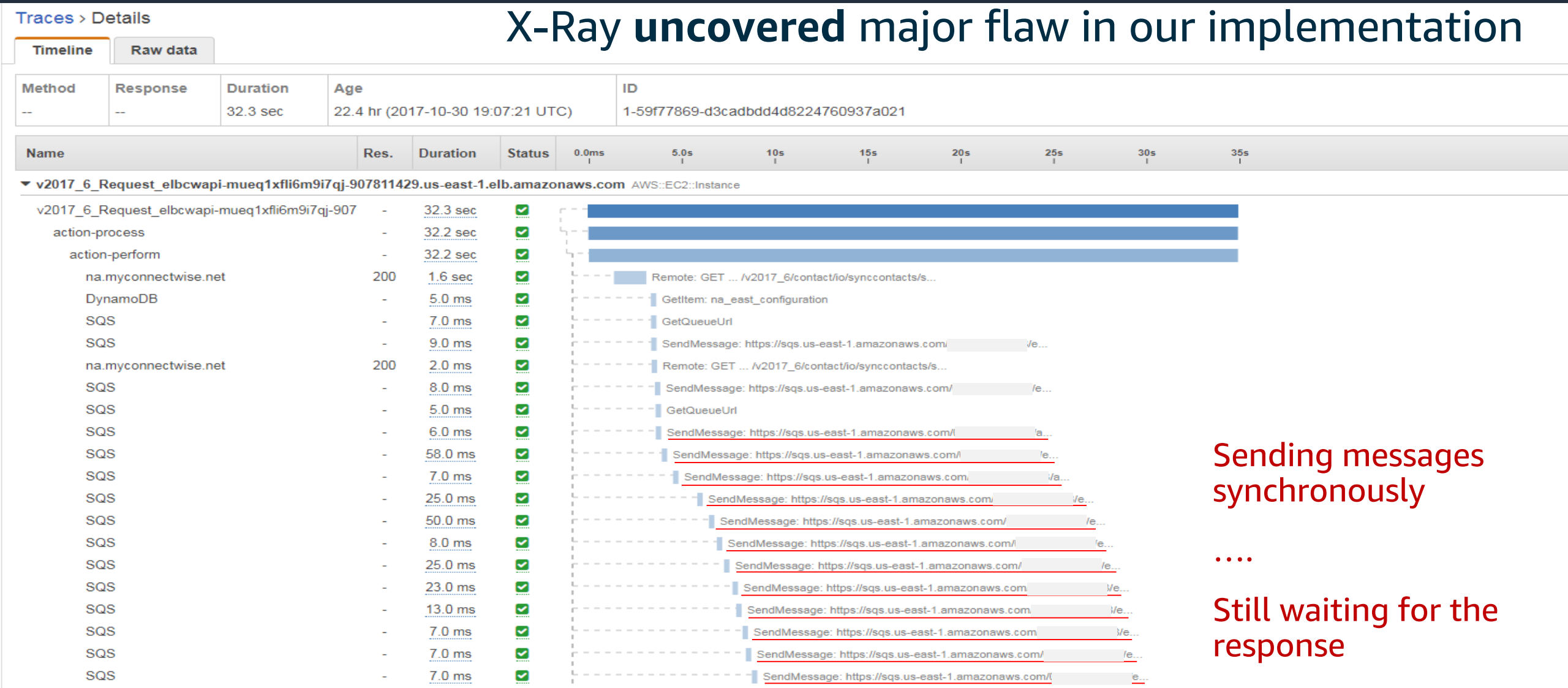
aws xray batch-get-traces --trace-ids <trace-ids>

Pass trace-ids for SQL calls

Latency (ms)	Parameters	SQL Command
7.0	@RecID Int 32639064	IF EXISTS (SELECT 1 FROM dbo.Schedule_Detail sd LEFT JOIN dbo.Schedule s on s.Schedule_RecID = sd.Schedule_RecID . . .
34.8		IF EXISTS (SELECT sr.SR_Location_RecID FROM DBO.SR_LOCATION as sr WHERE sr.SR_Location_RecID =24) SELECT . . .
1.0	@recid Int 65	SELECT Owner_Level_RecID FROM dbo.User_Defined_Field_Owner_Level WHERE User_Defined_Field_RecID = @recid
18.6	@serviceRecId Int 11111899	SELECT Company_RecID FROM dbo.SR_Service WHERE SR_Service_RecID = @serviceRecId
123.7	@problemFlag Int 1 . . .	exec dbo.usp_getSRDetailTable @problemFlag, @resolutionFlag, @internalAnalysisFlag, @serviceRecID, @includeChild
1201.8	@problemFlag Int 1 . . .	exec dbo.usp_getSRDetailTable @problemFlag, @resolutionFlag, @internalAnalysisFlag, @serviceRecID, @includeChild

Entity sync issue

X-Ray **uncovered** major flaw in our implementation

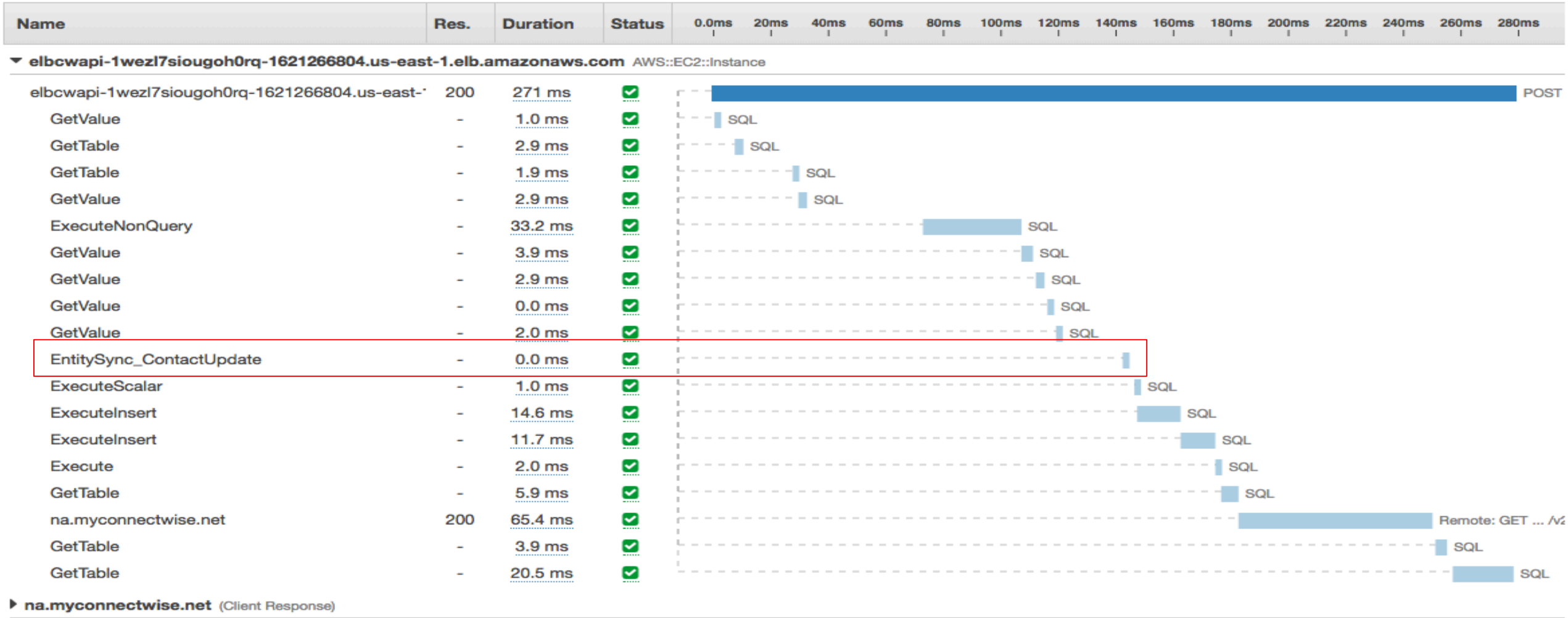


Sending messages synchronously

....

Still waiting for the response

Entity sync resolved

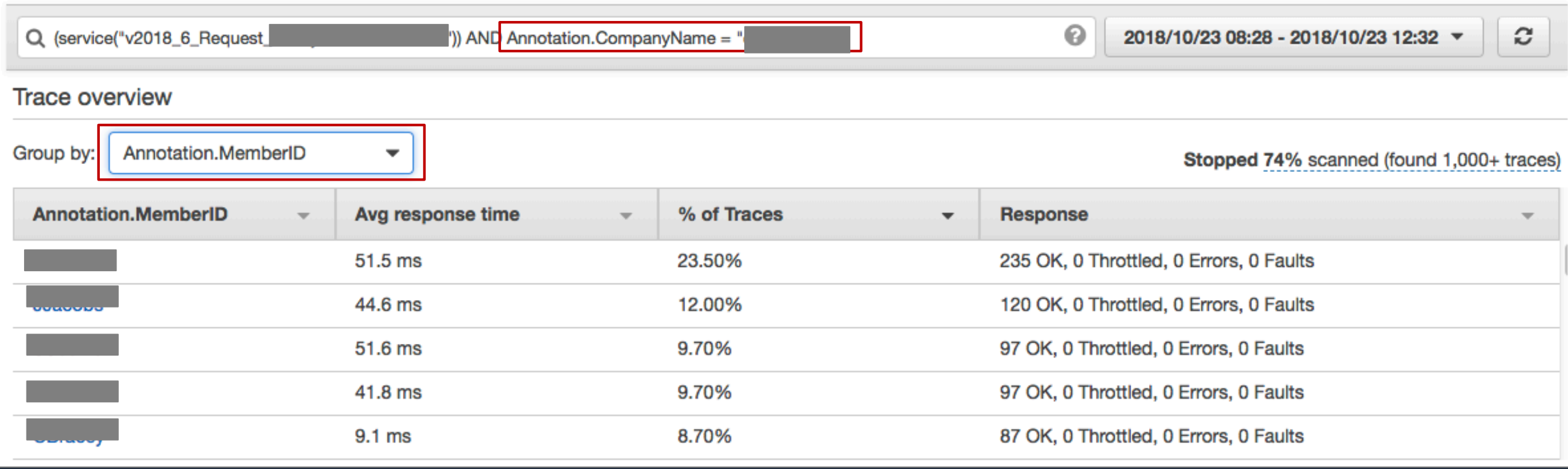


Syncing is now taking place on an **asynchronous worker threads**, no impact on main application

By user

Create **user** annotation from session in the existing segment

```
AWSXRayRecorder.Instance.SafeAddAnnotation("MemberID", HttpContext.Current.Request.Cookies["memberId"].Value);
```



Business impact - Usage analytics

Trace overview

Group by: Annotation.EntityType

Done 100%

Annotation.EntityType	Avg response time	% of Traces	Response
Ticket	1.9 sec	39.88%	268 OK, 0 Throttled, 0 Errors, 0 Faults
ContactName	258 ms	15.18%	102 OK, 0 Throttled, 0 Errors, 0 Faults
Contact	258 ms	15.18%	102 OK, 0 Throttled, 0 Errors, 0 Faults
Product	704 ms	6.99%	47 OK, 0 Throttled, 0 Errors, 0 Faults
Activity	704 ms	6.99%	47 OK, 0 Throttled, 0 Errors, 0 Faults

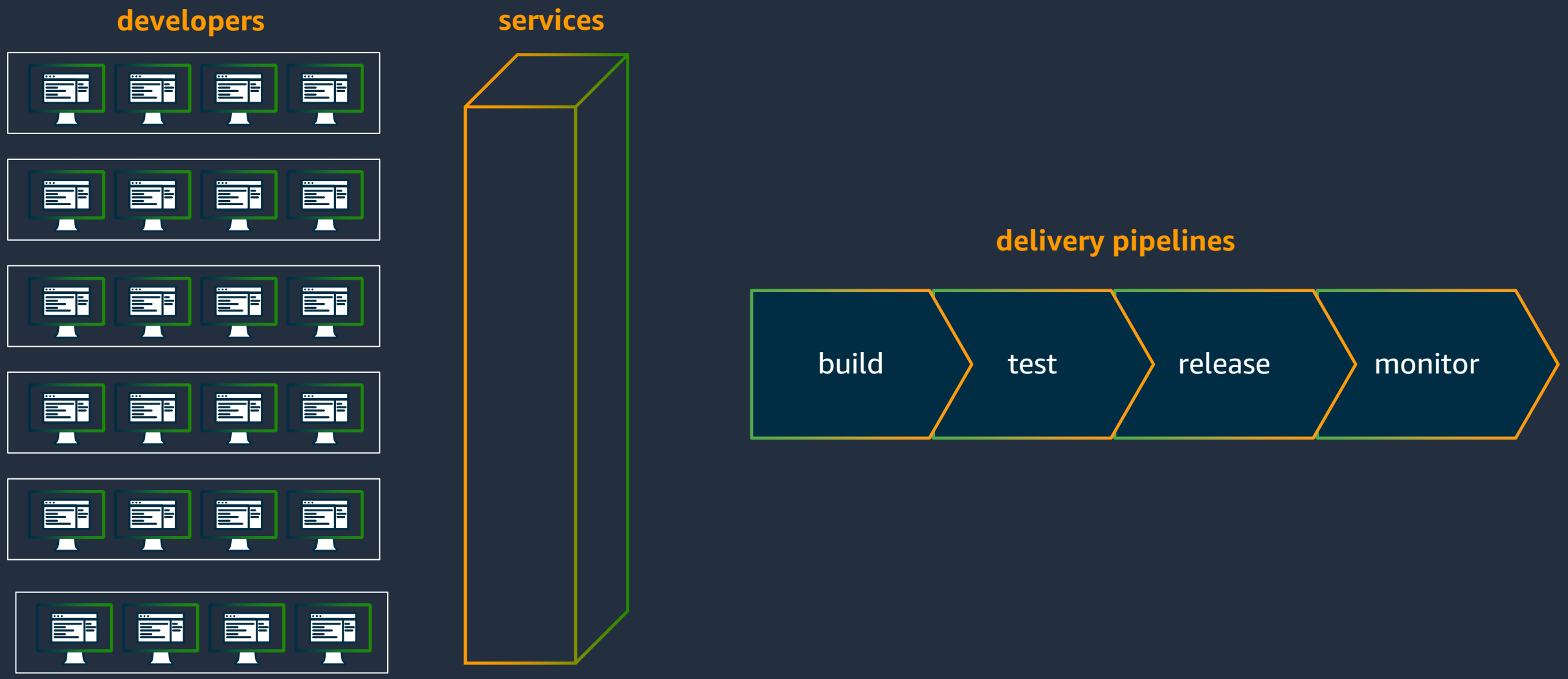
Annotation.EntityType indicates application modules

% of traces will give you usage metrics by modules within specific time frame

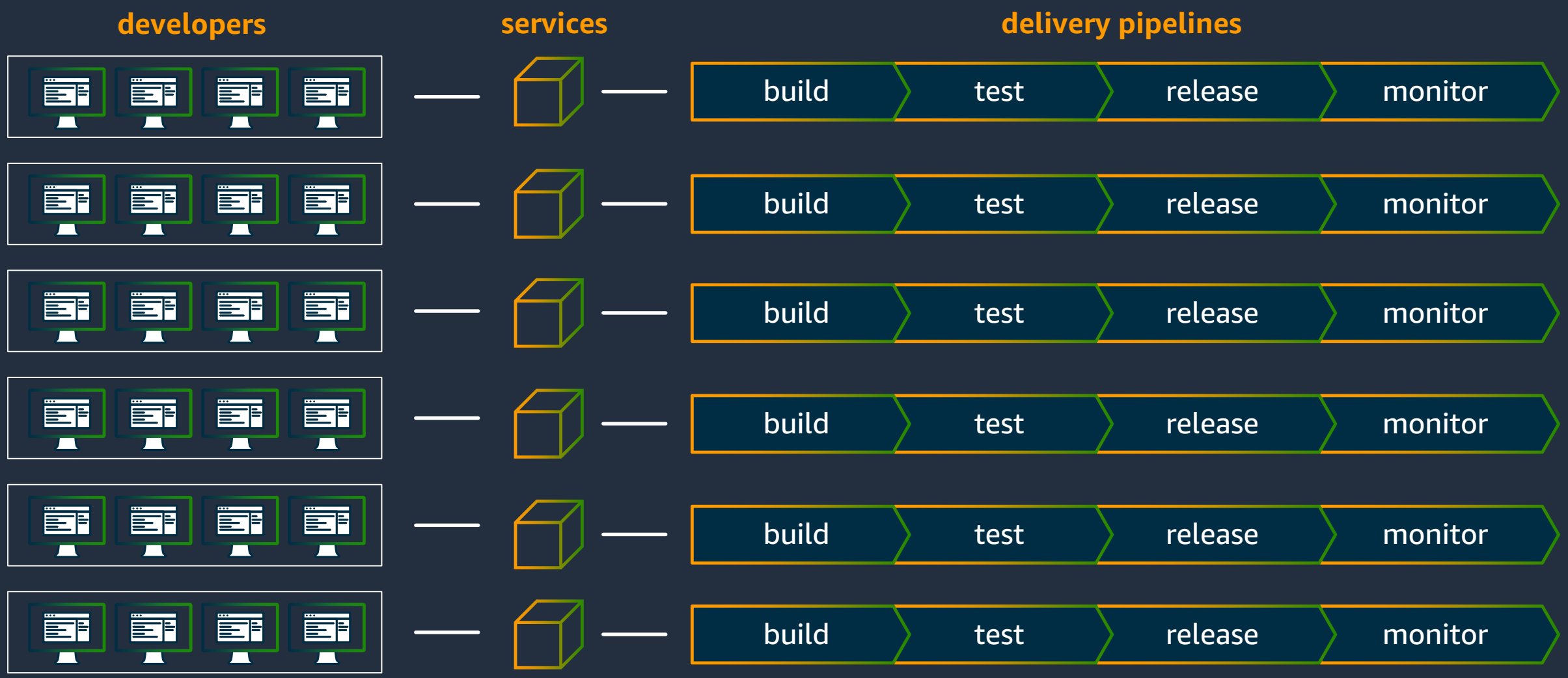
Changes to the delivery of software

How do I develop and deploy code in a serverless microservices architecture?

Monolith development lifecycle



Microservice development lifecycle



Best practices

Decompose for agility
(microservices, 2 pizza teams)

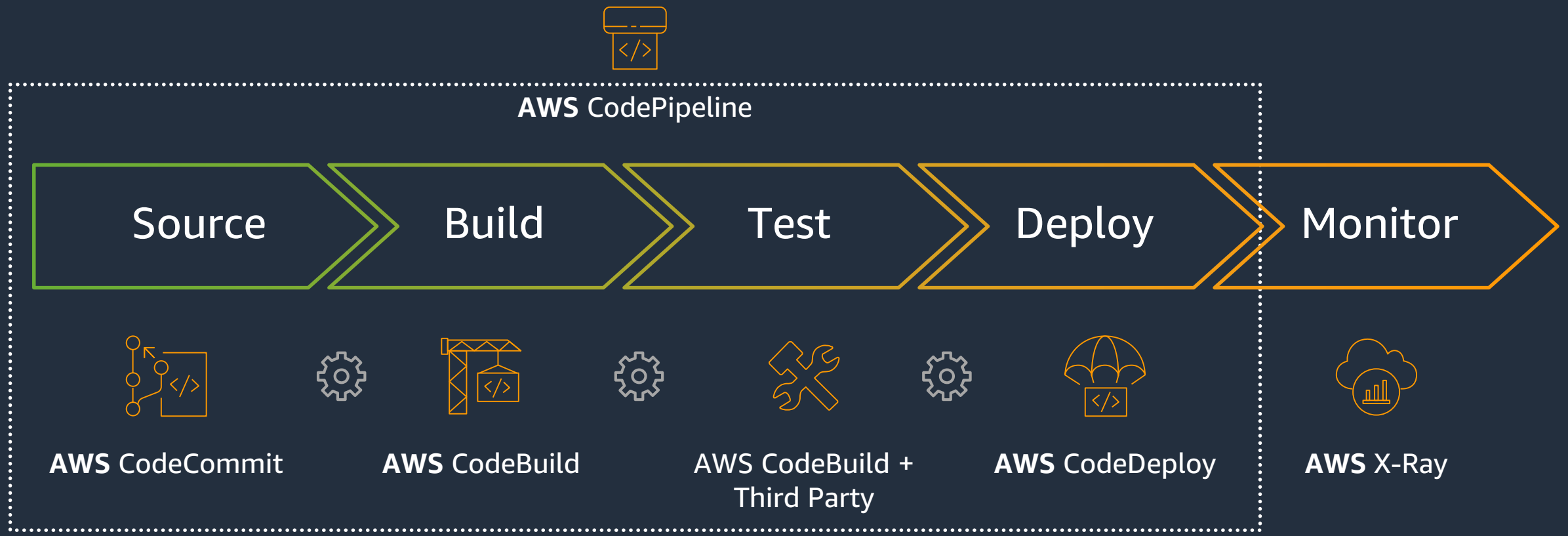
Automate everything

Standardized tools

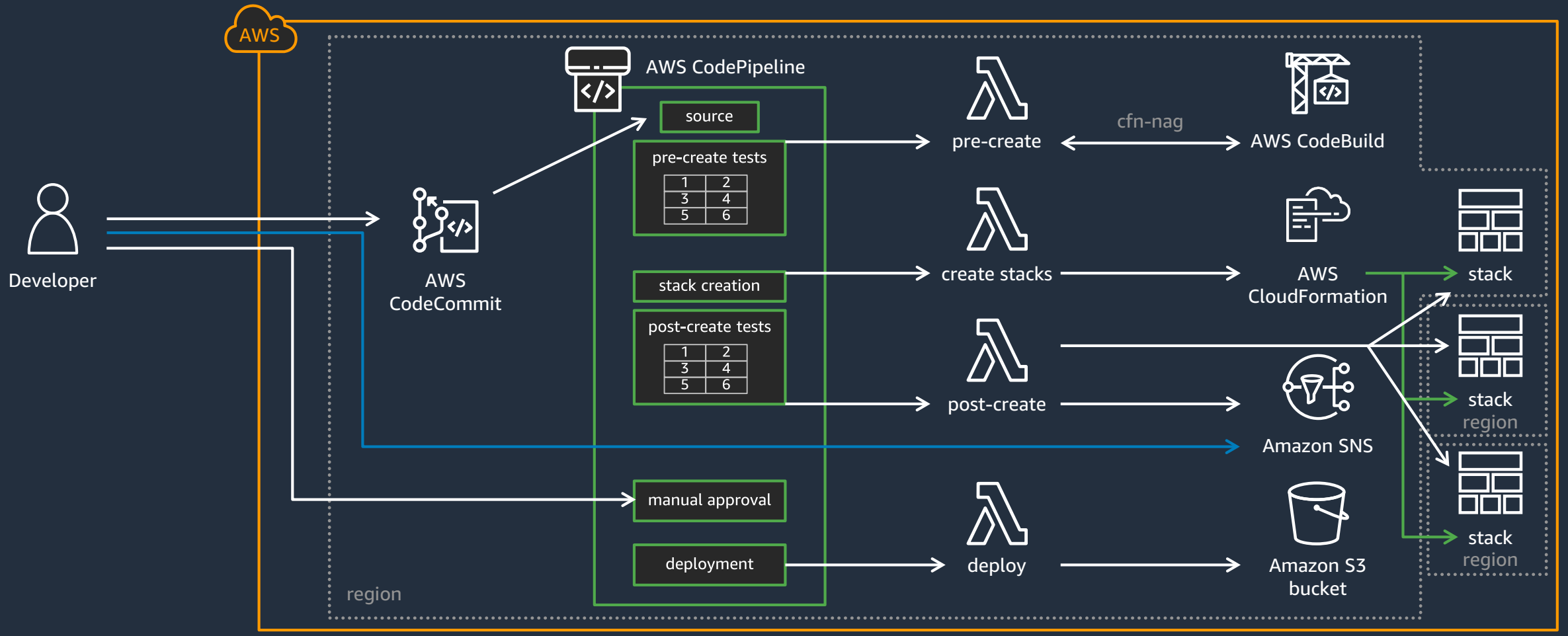
Belts and suspenders
(governance, templates)

Infrastructure as code

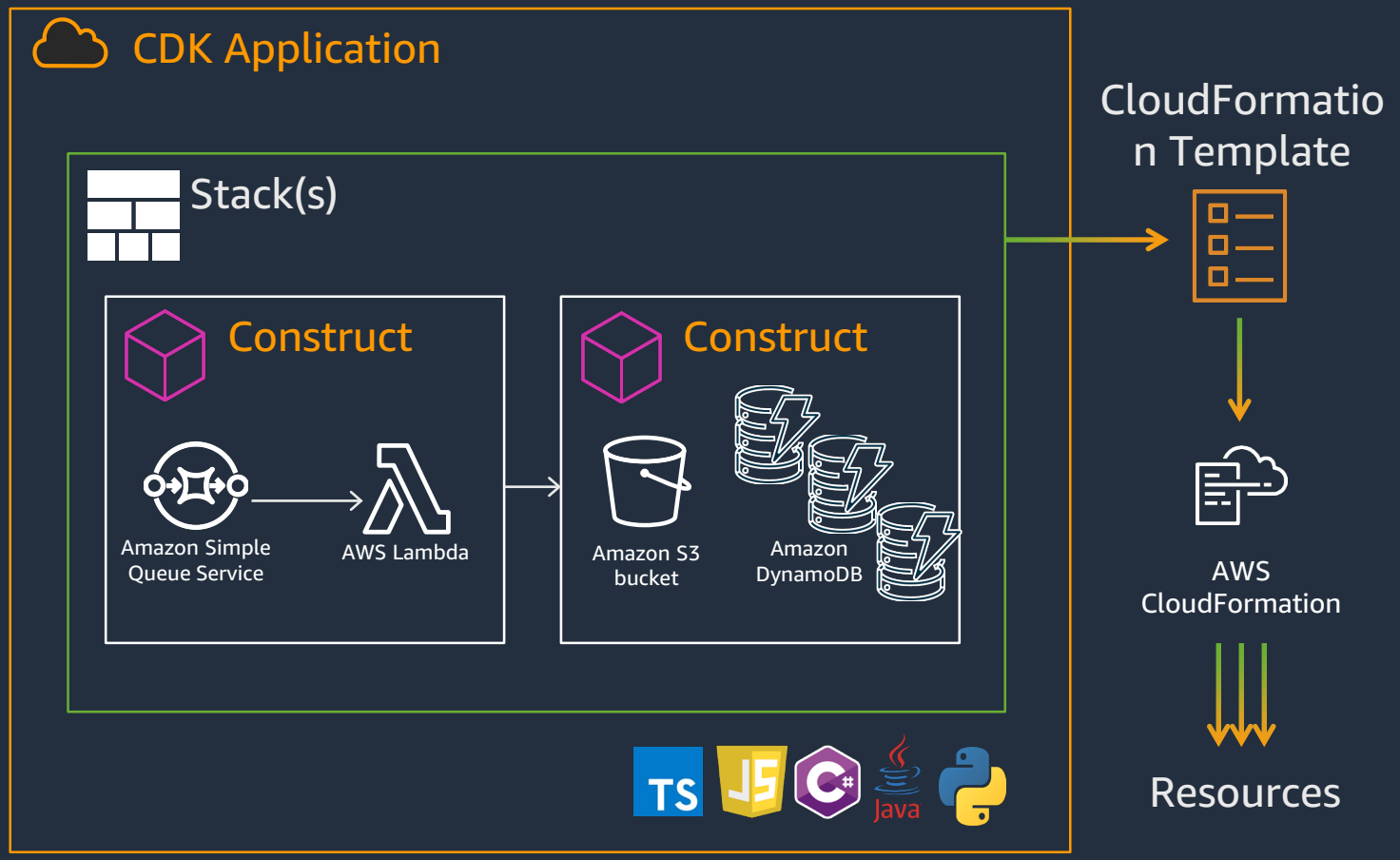
AWS Developer Tools for CI/CD



How can we best model and provision our infrastructure?



AWS Cloud Development Kit



AWS customers are pioneering modern applications

reduced overall compute costs by 95%

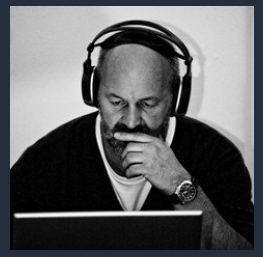
cut processing time from 36 hours to 10 seconds

created a stock trade validation system in 3 months

releases over 50+ deployments per hour

Where to learn more

Open Source Distributed Tracing CNCF | opentelemtry.io

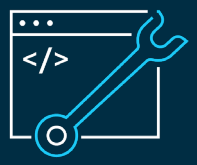


Modern Applications by Werner Vogels | allthingsdistributed.com

Modern Application Design AWS | aws.amazon.com/modern-apps

Conclusion

Your modern application development journey starts with AWS Training and Certification



Training

[Developing on AWS](#) is where you will learn how to use the AWS SDK to develop secure and scalable cloud applications. We will explore how to interact with AWS using code and discuss key concepts, best practices, and troubleshooting tips.

[AWS Certified Developer – Associate](#)

Developers with one or more years of hands-on experience on AWS

This exam validates an understanding of core AWS services, uses, and basic AWS architecture best practices. Examinees must demonstrate proficiency in developing, deploying, and debugging cloud-based applications using AWS.

Visit https://www.aws.training/

[AWS Certified DevOps Engineer – Professional](#)

DevOps engineers with two or more years of experience on AWS

This exam tests an engineer's experience provisioning, operating, and managing AWS environments. Examinees will show an understanding of how to build highly scalable, available, and self-healing systems on the AWS platform and to design, manage, and maintain tools to automate operational processes.

aws

Thank You for Attending
AWS Online Event: Modern Application Development

We hope you found it interesting! A kind reminder to **complete the survey.**
Let us know what you thought of today's event and how we can improve the event experience for you in the future.


aws-apac-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices


youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws