



Serverless

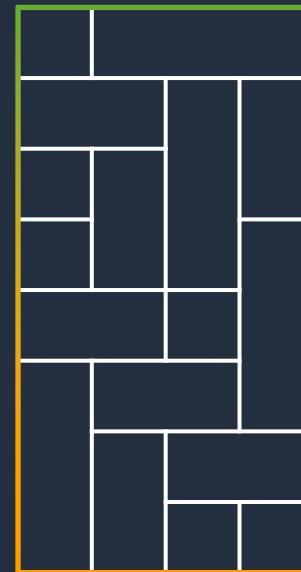
Modern Application Development

Greg Pryzby
Yasser Quraishi
Robert Ray

Development transformation at Amazon: 2001–2002

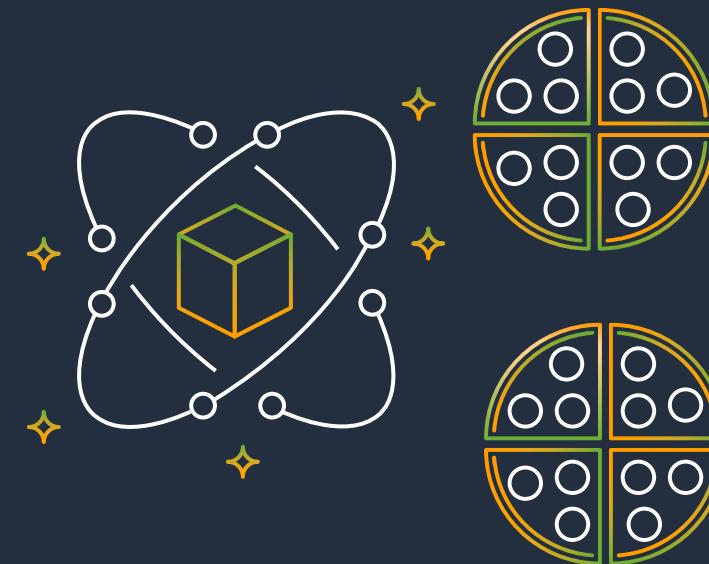
Lesson learned: decompose for agility

2001



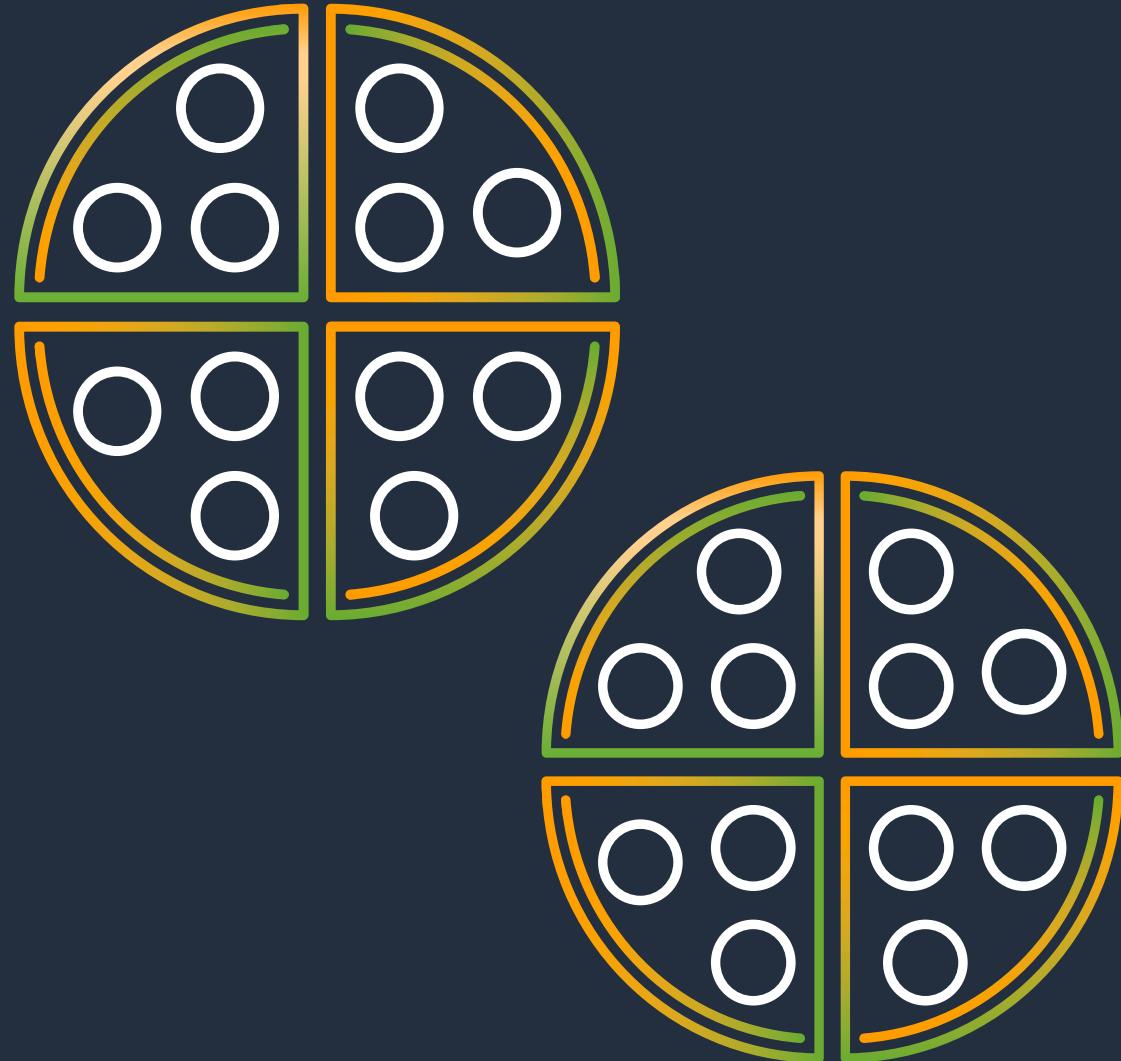
monolithic application
+ teams

2002



microservices
+ 2 pizza teams

Two-pizza teams



Full ownership

Full accountability

“DevOps”

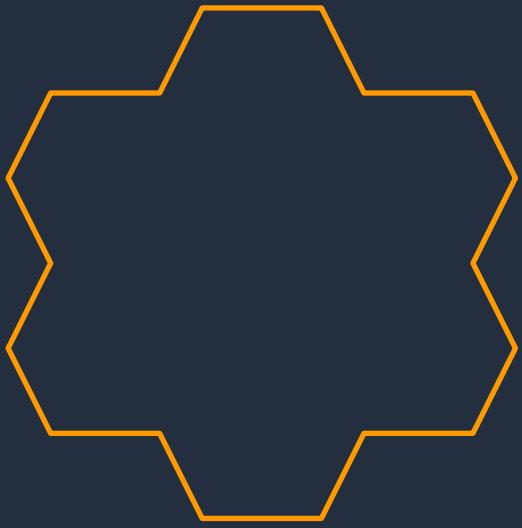
Focused innovation

What changes
have to be made
in this new world?

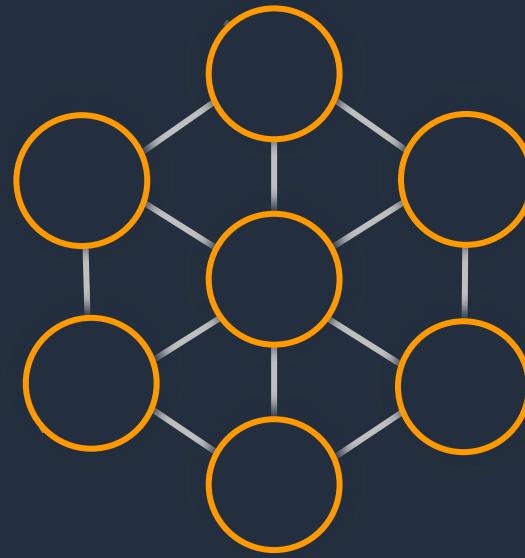
Architectural patterns
Operational model
Software delivery

Changes to the architectural patterns

When the impact of change is small, release velocity can increase

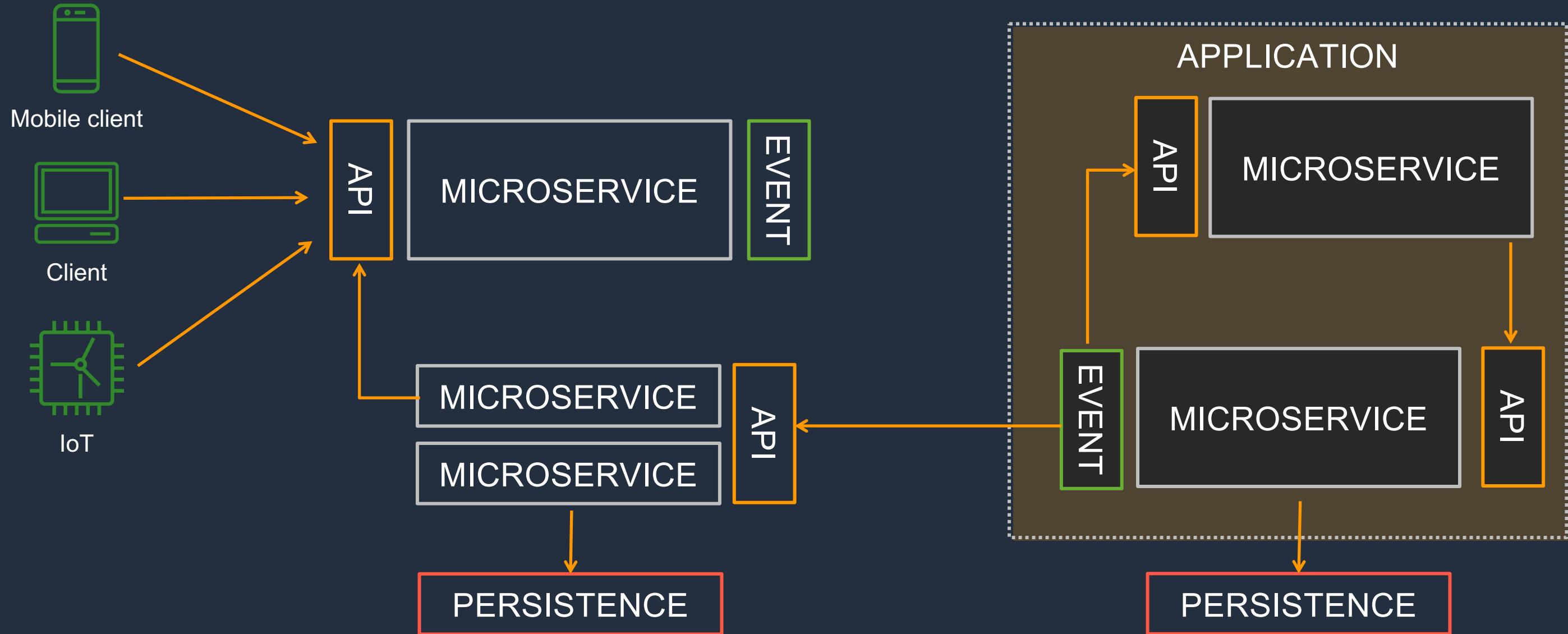


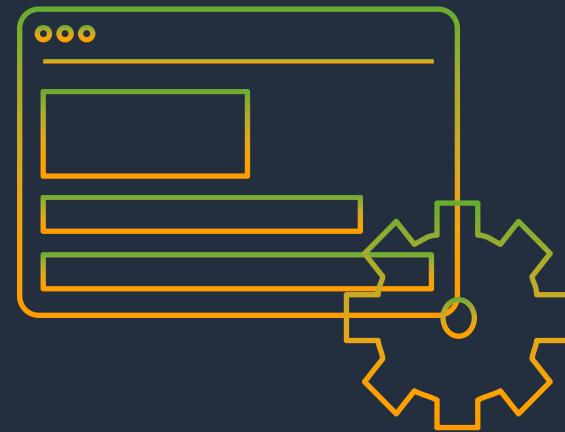
Monolith
Does everything



Microservices
Does one thing

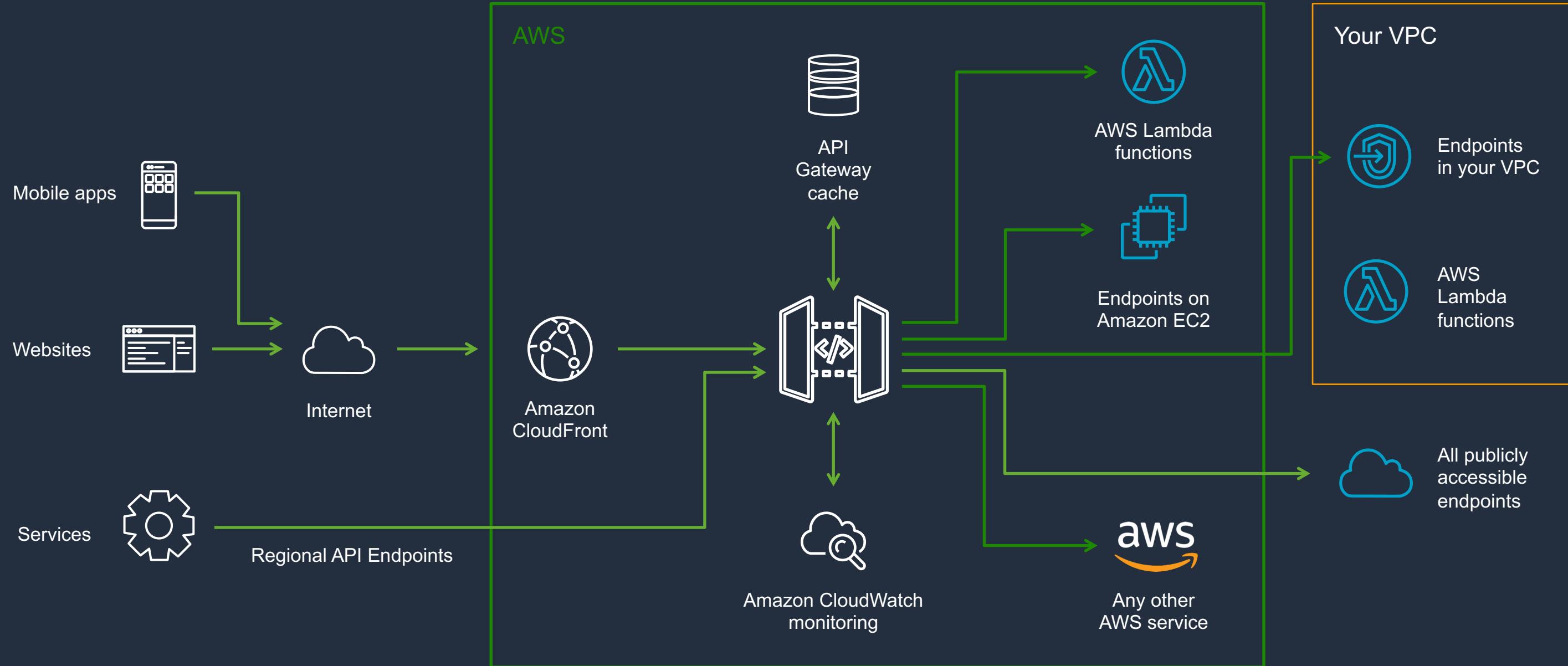
Microservices architectures





APIs are the front door of microservices

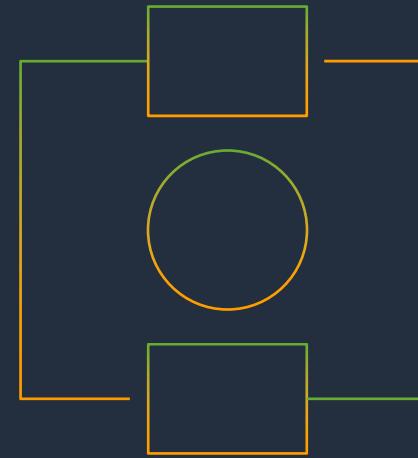
Manage APIs with API Gateway



© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark





Event-driven architectures

Decouple state from code using messaging

Messaging

		
Amazon Simple Queue Service	Amazon Simple Notification Service	Amazon CloudWatch Events
Queues	Pub/sub	Synchronization
Simple Fully-managed Any volume	Simple Fully-managed Flexible	Rapid Fully-managed Real-time

And data streams

Data Stream Capture



**Amazon Kinesis
Data Streams**

Ingest

Data streams
Data processing
Real-time



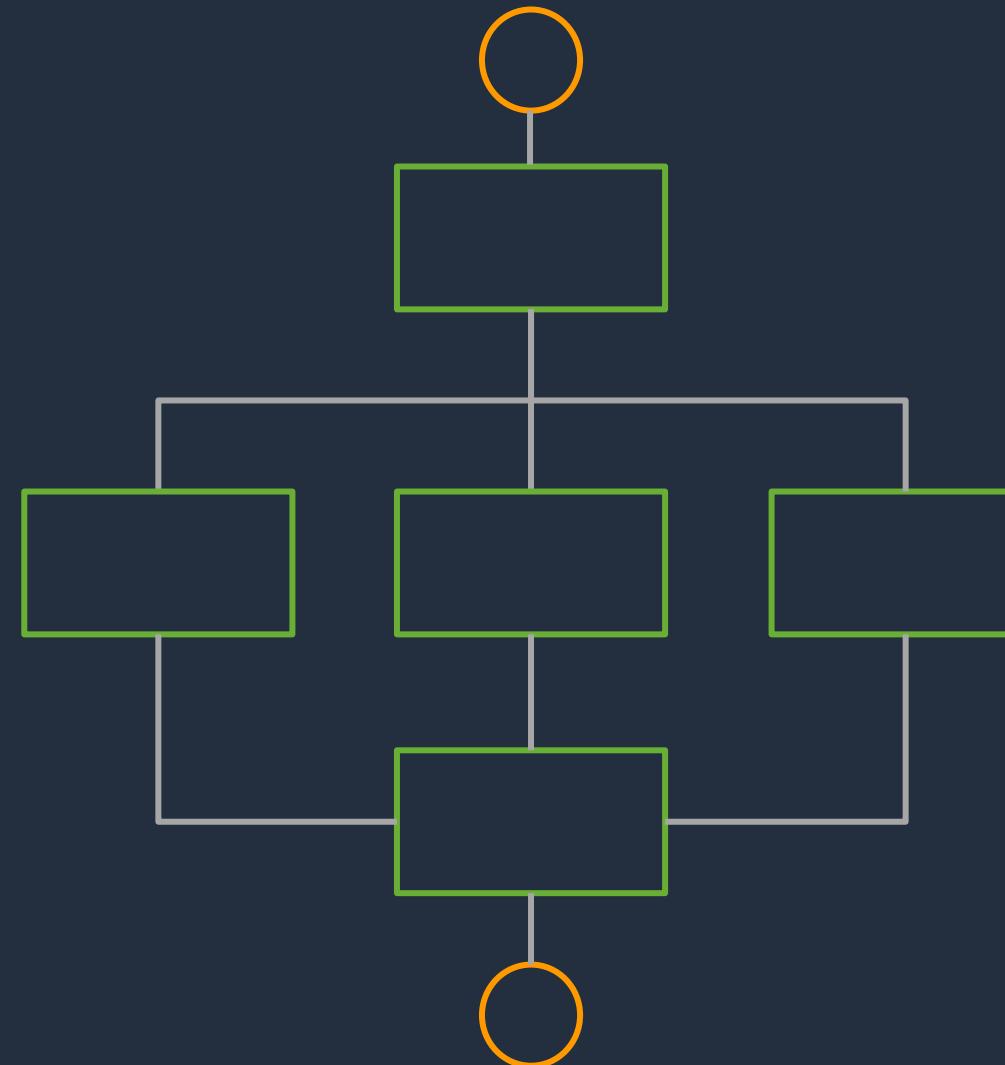
**Amazon
Dynamo DB**

Data Store

Microservices
Performance at scale
Fast and Flexible

Build workflows to orchestrate everything

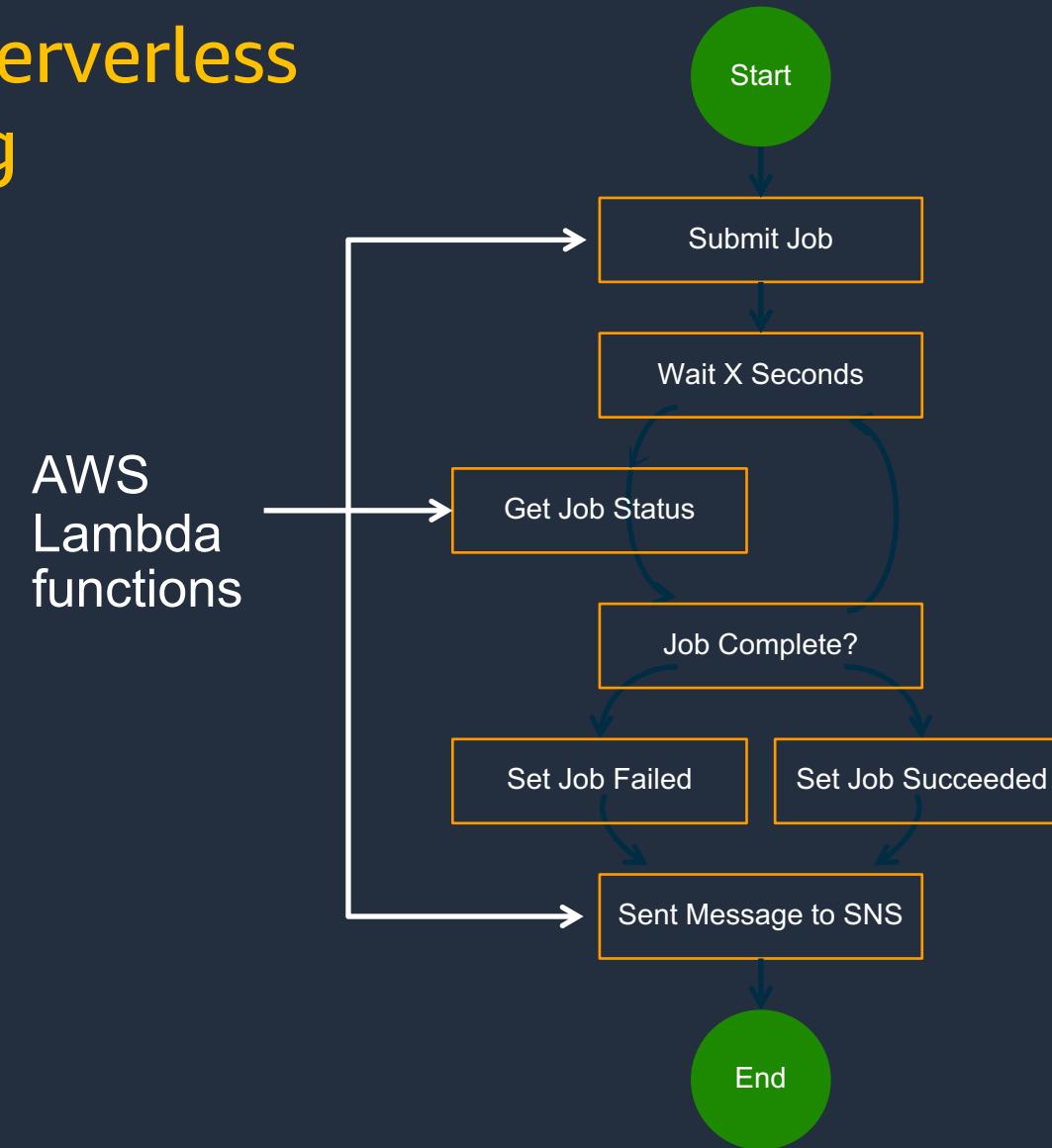
Track status of data
and execution



Remove
redundant code

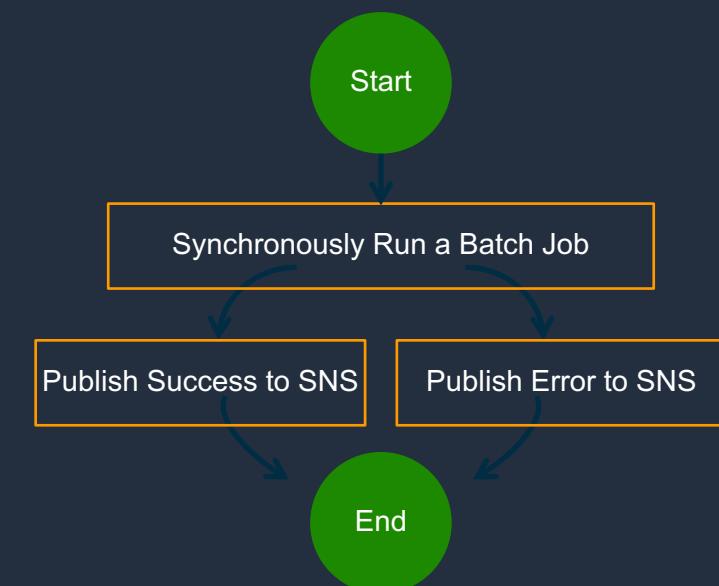
Simpler integration, less code

With serverless
polling



With new
service integration

No
Lambda
functions







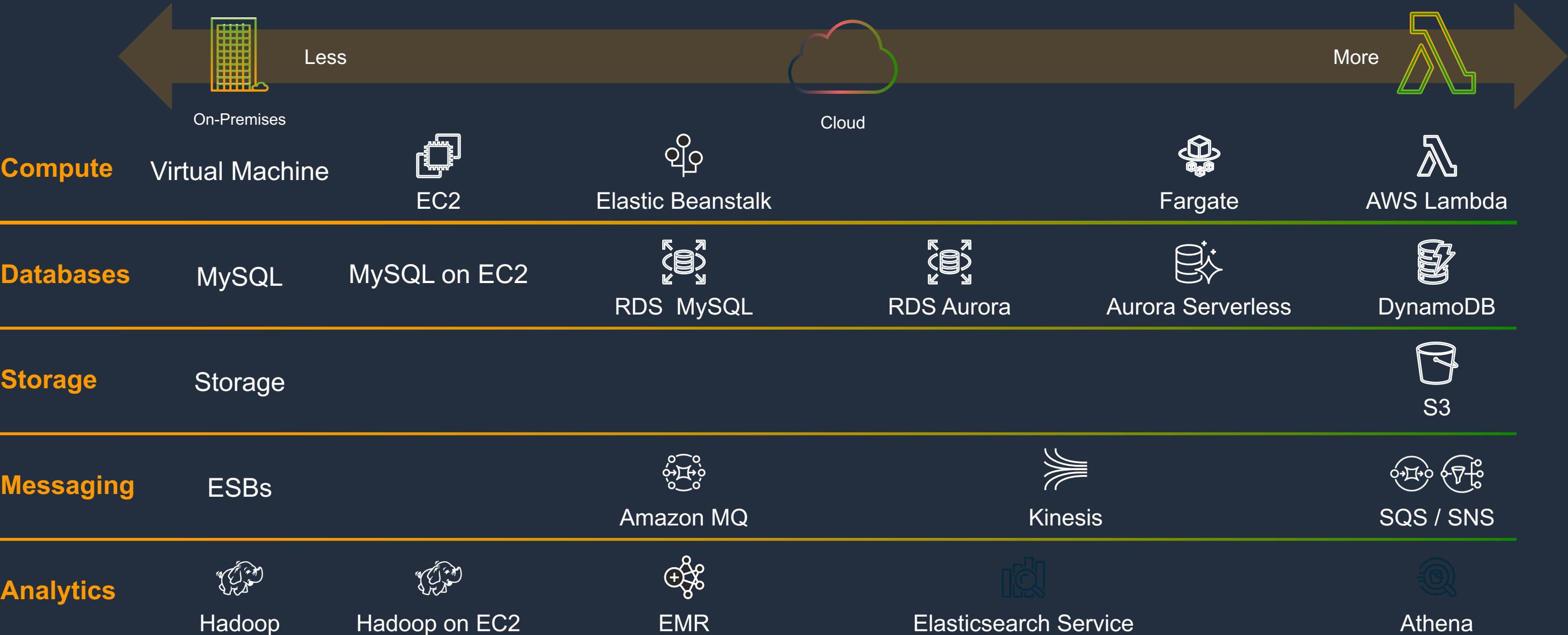
**Cloud-native architectures are small pieces,
loosely joined**

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

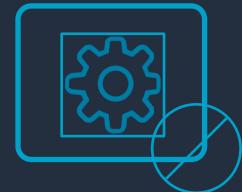


© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark



What is serverless?

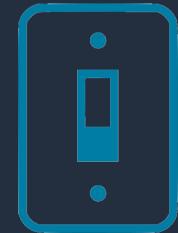


No infrastructure provisioning,
no management



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

Let's focus on compute for now



AWS Lambda

**Serverless event-driven
code execution**

Short-lived

All language runtimes

Data source integrations



AWS Fargate

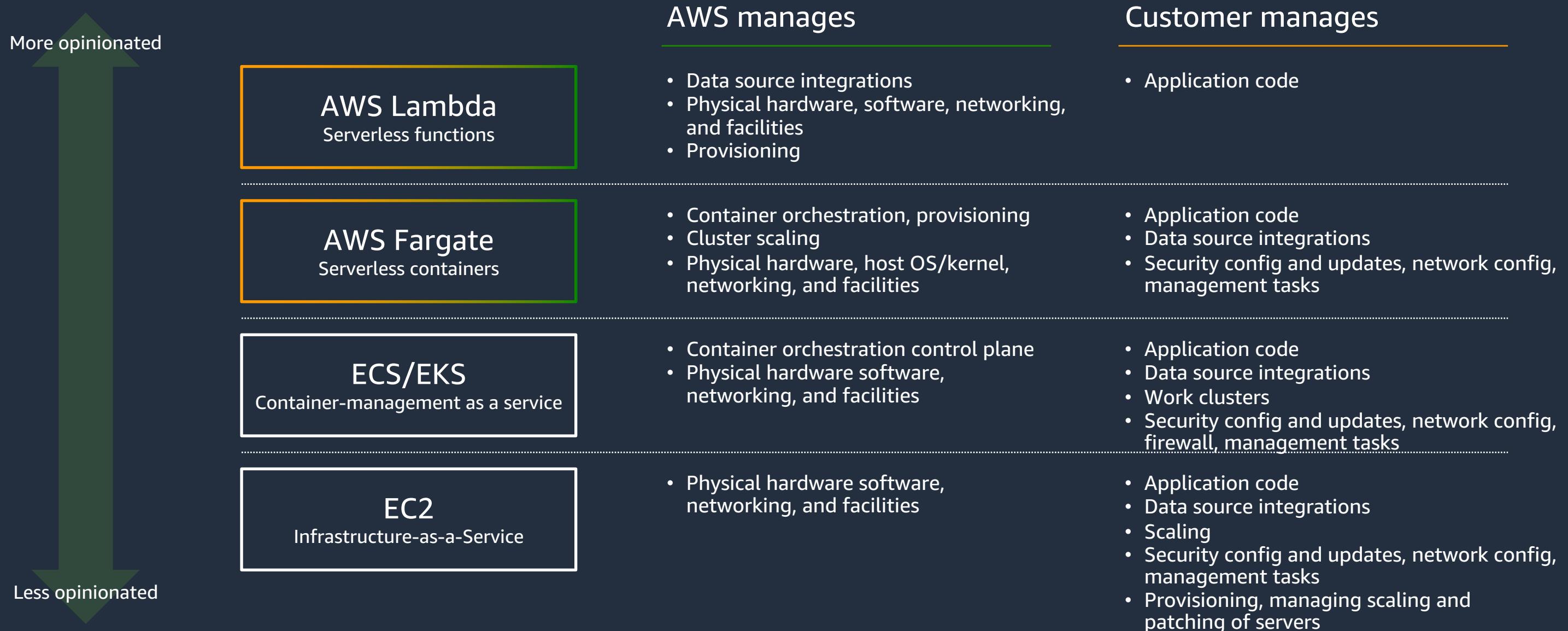
**Serverless compute engine
for containers**

Long-running

Bring existing code

Fully-managed orchestration

Comparison of operational responsibility



Making development easier with AWS Lambda



Accessible for all developers

Support for all runtimes with Lambda Layers and Runtime API

ISO, PCI, HIPAA, SOC, GDPR, and FedRamp compliances

Greater productivity

Toolkits for popular IDEs:
VSCode, IntelliJ, and PyCharm

Simplified deployment with nested apps

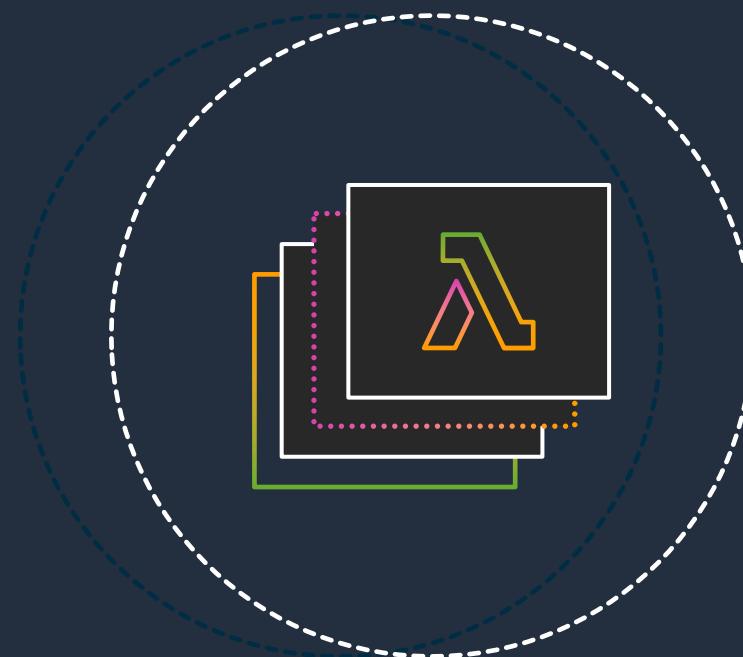
Enable new application patterns

15 minute functions
SQS for Lambda

Automatic Load Balancing for Lambda
Support for Kinesis Data Streams Enhanced Fan-Out and HTTP/2

Trillions of executions every month for hundreds of thousands of active customers

Lambda Layers



Lets functions easily share code: Upload layer once, reference within any function

Promote separation of responsibilities, lets developers iterate faster on writing business logic

Built in support for secure sharing by ecosystem

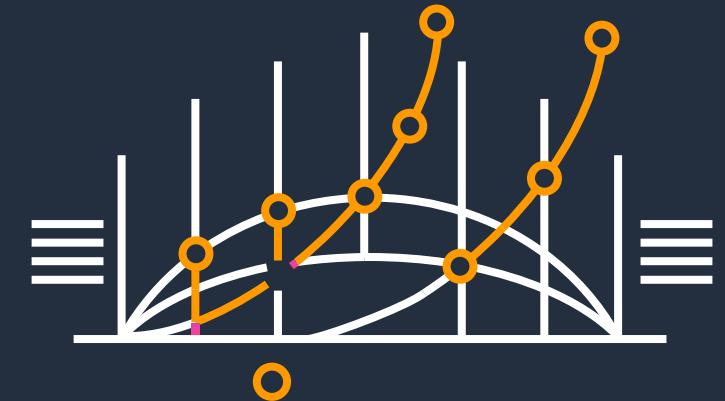
Accelerating Fargate and Lambda with Firecracker



Security



Speed by design



Scale and efficiency

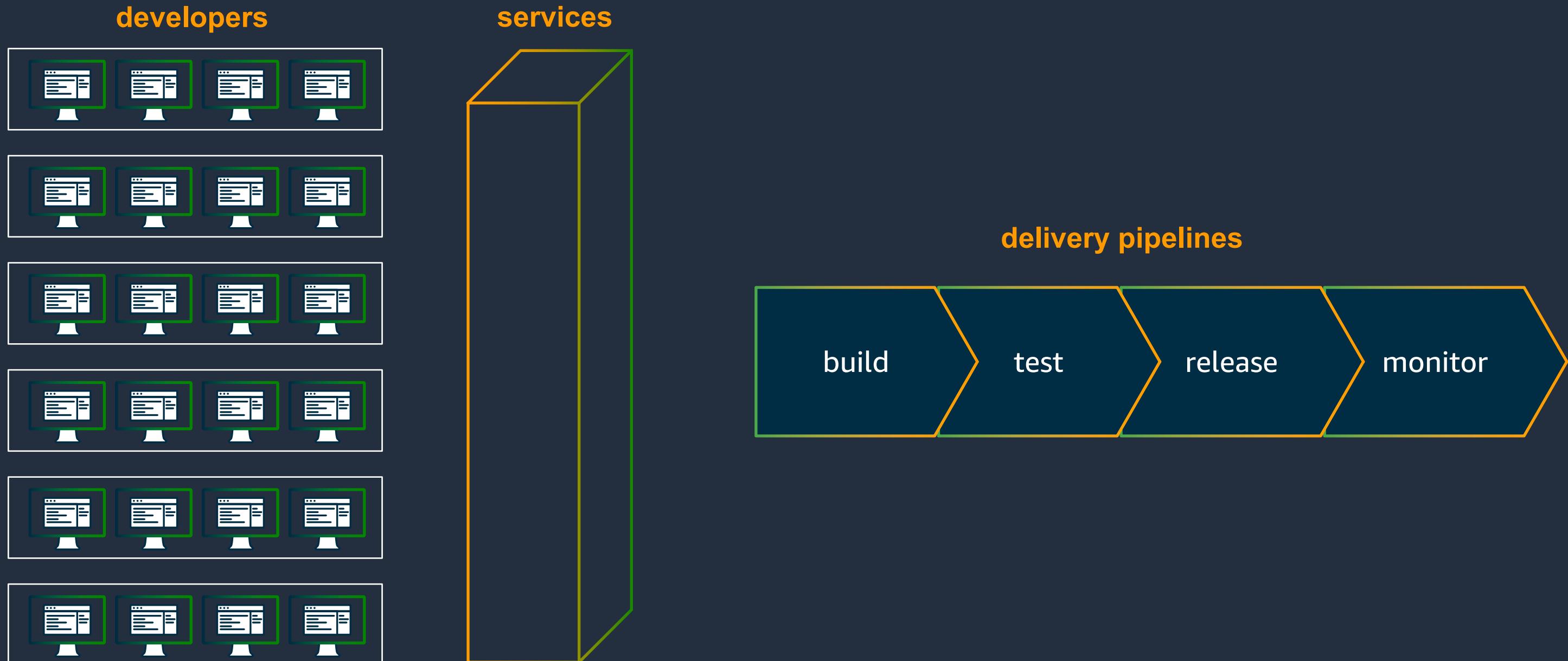
Firecracker is open sourced to enable broad access and innovation

Changes to the delivery of software

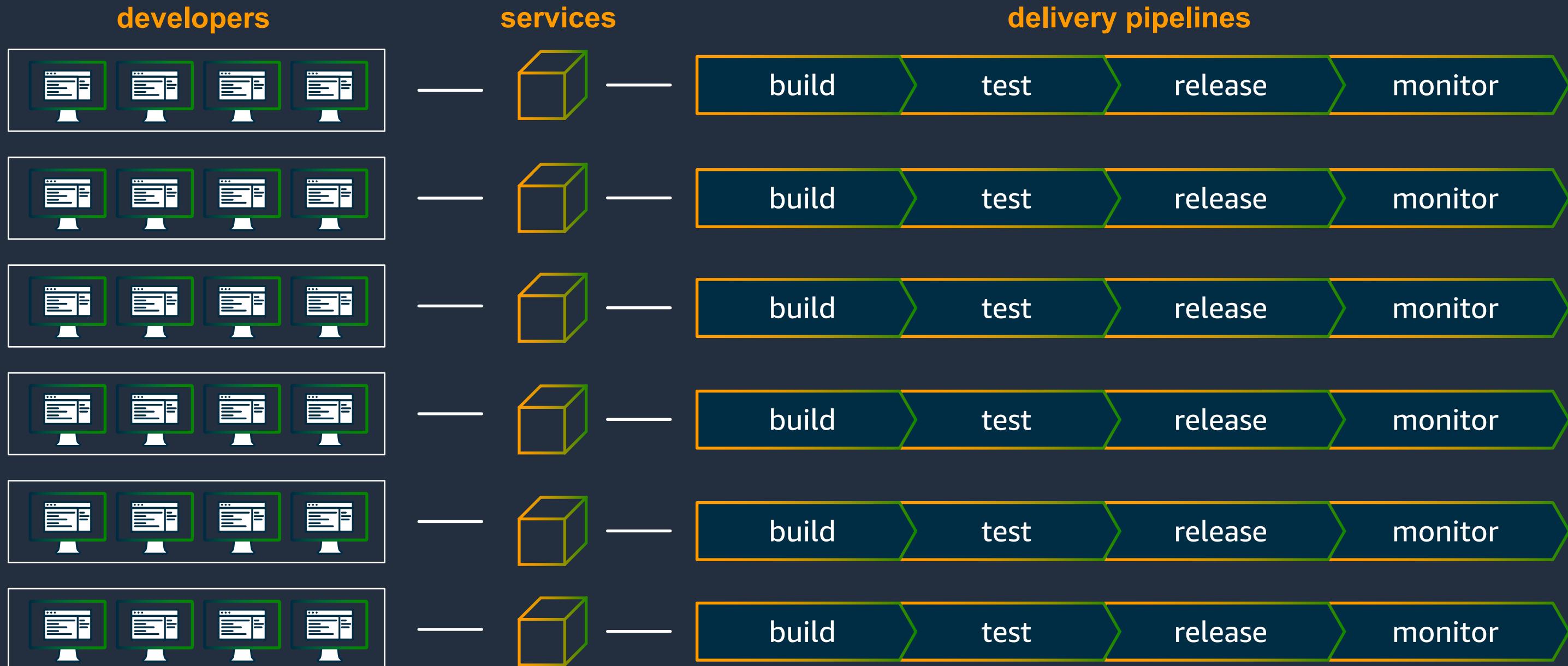


How do I develop and deploy code in a serverless microservices architecture?

Monolith development lifecycle



Microservice development lifecycle



Best practices



Decompose for agility
(microservices, 2 pizza teams)

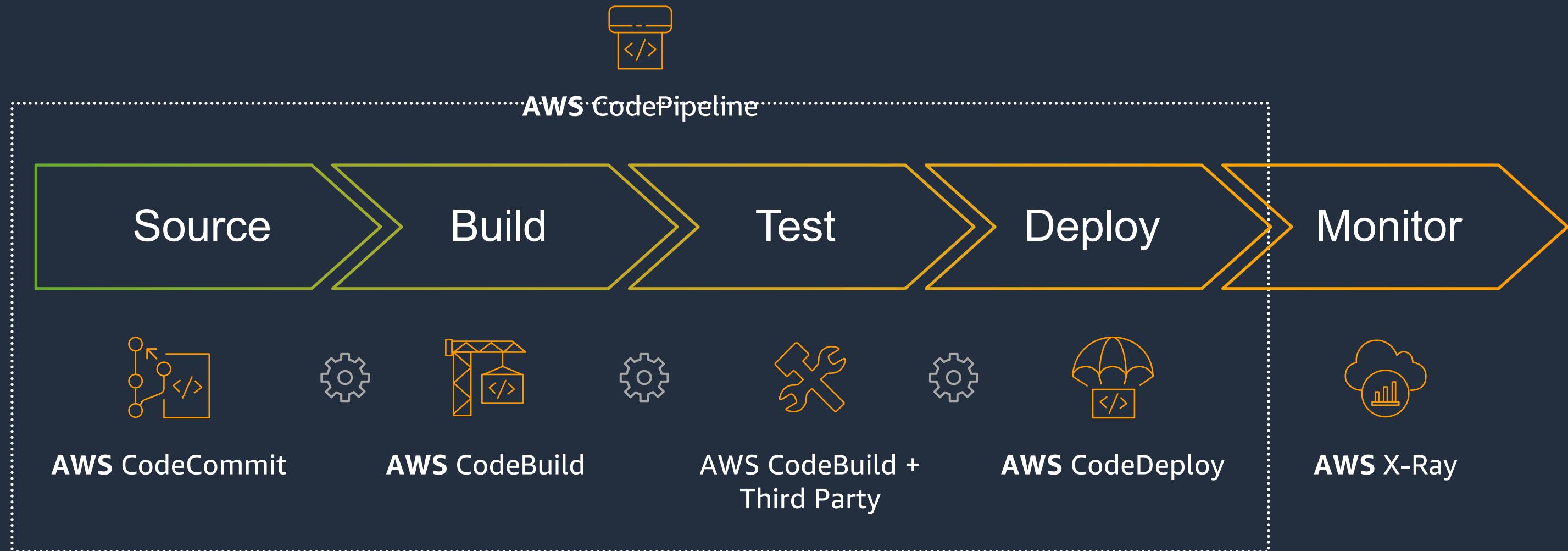
Automate everything

Standardized tools

Belts and suspenders
(governance, templates)

Infrastructure as code

AWS Developer Tools for CI/CD



AWS Developer Tools are focused on supporting containers and Lambda

2016

NOV



Support for Lambda deployment with CodePipeline and CloudFormation

2017

NOV



Support for rolling and blue/green Lambda deployments with CodeDeploy

DEC



Support for Fargate and ECS deployments in CodePipeline

2018

OCT



CodePipeline supports Config for improved governance

NOV



CodePipeline supports ECR as a source

NOV



Support for blue/green deployments for Fargate and ECS with CodeDeploy

AWS X-Ray is Built for Modern Applications



Analyze and debug
issues quickly



End-to-end view of
individual services



Identify customer
impact



Support for
Serverless

How do I edit and debug my serverless application code?



Author and debug Lambda applications on AWS using your favorite IDEs



AWS
Cloud9

Python, Node



AWS Toolkit
for PyCharm

Python

Developer
Preview



AWS Toolkit
for IntelliJ

Java, Python

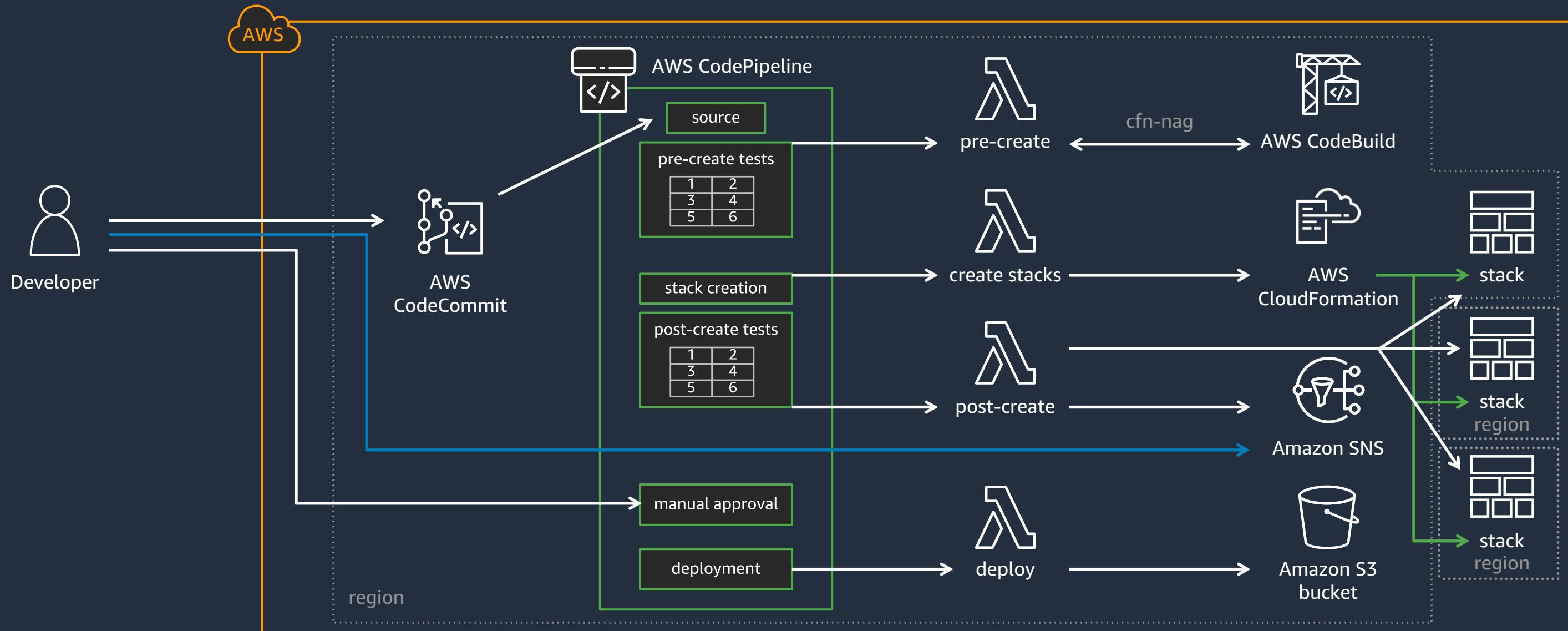
Developer
Preview



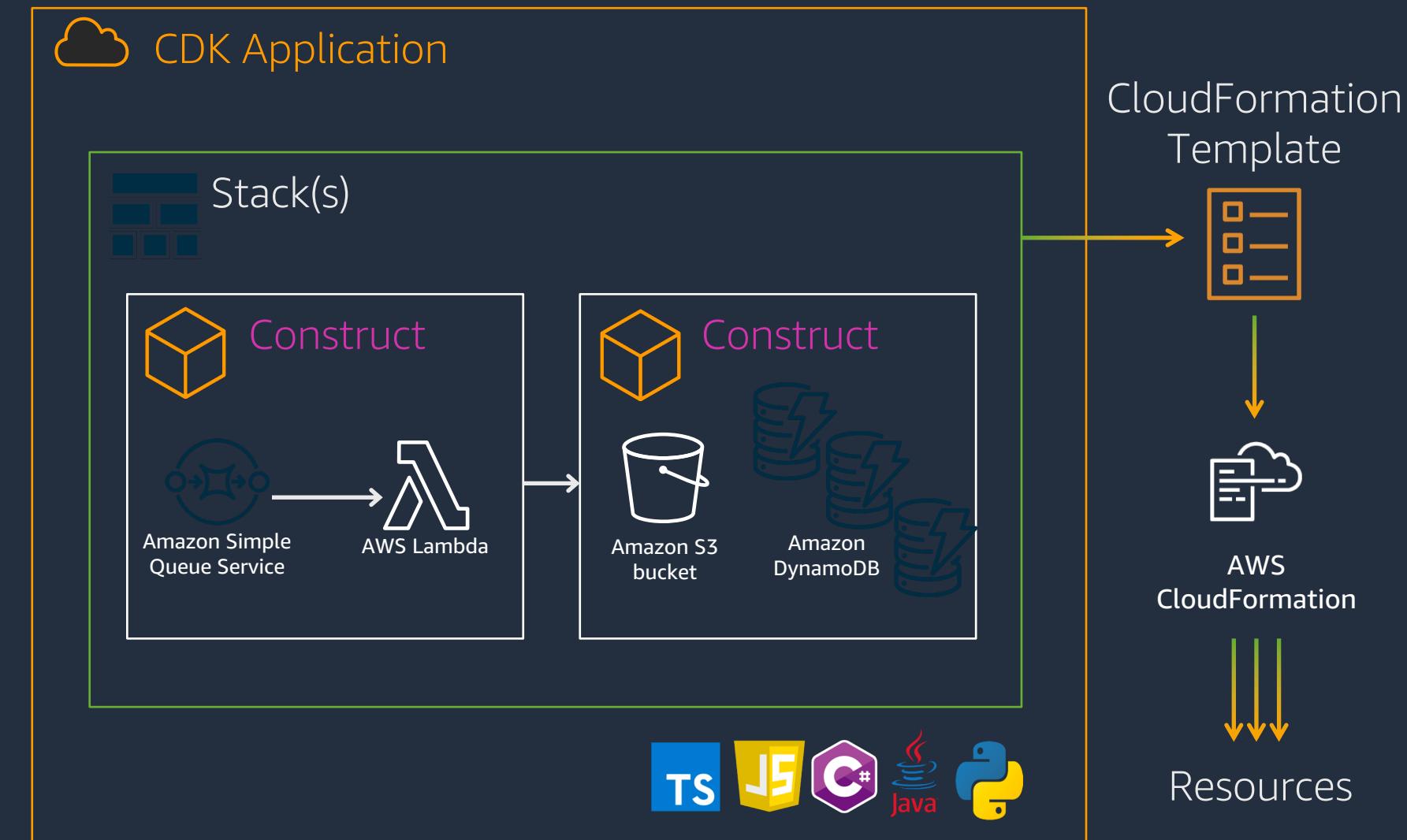
AWS Toolkit
for Visual Studio
Code

.NET, Node

How can we best model and provision our infrastructure?



AWS Cloud Development Kit



© 2018, Amazon Web Services, Inc. or its affiliates. All rights reserved.

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved. Amazon Confidential and Trademark



Application models simplify building serverless applications



AWS SAM

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

The Washington Post releases over 50+ deployments per hour

Common Adoption Patterns



Leap-frog adoption

Organic adoption

Incremental refactoring

Conclusion

We are building a **cloud** that best supports your modern application development needs, and we are innovating across the **entire stack**: from the hypervisor layer to the application construction layer.