Evolution of Containers and Serverless

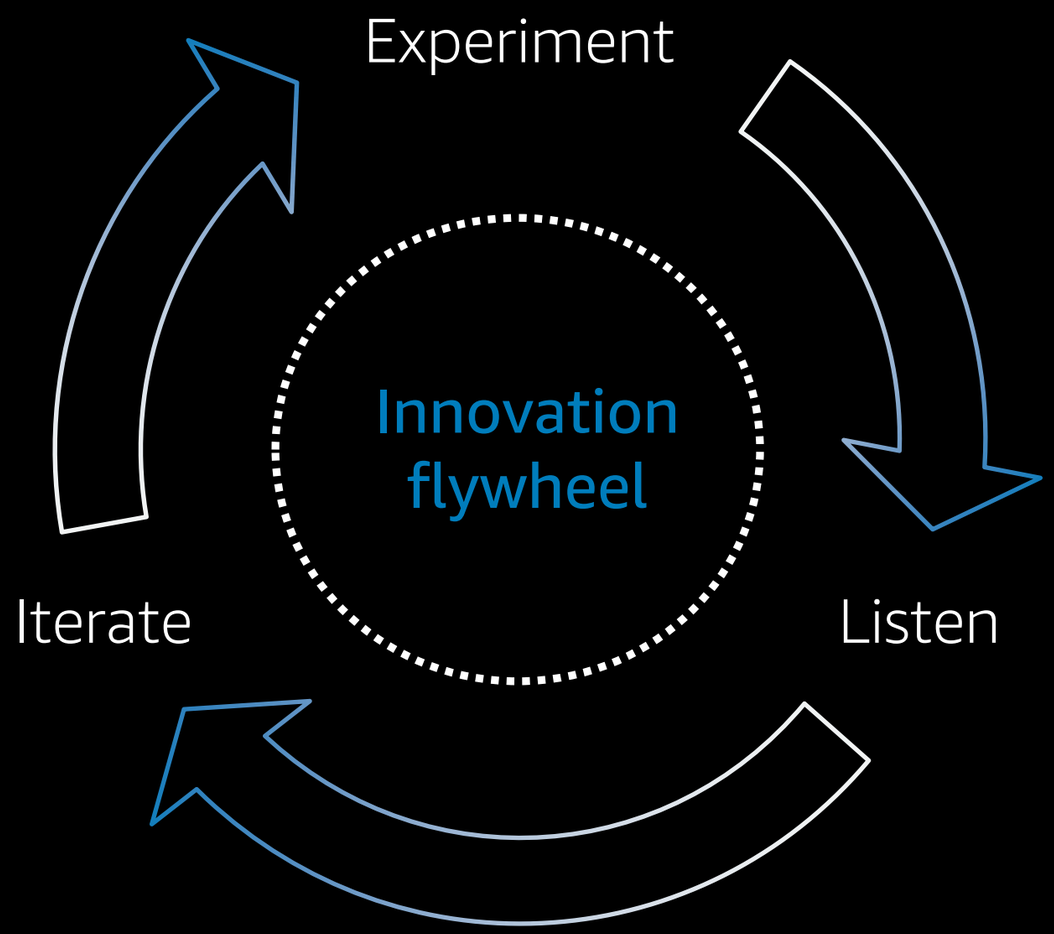
Steffen Grunwald, @steffeng

18. December 2019

"We want to be a **large company** that's also an **invention machine**. We want to combine the extraordinary customer-serving capabilities that are enabled by size with the speed of movement, nimbleness, and risk-acceptance mentality normally associated with entrepreneurial start-ups."

Jeff Bezos
CEO, Amazon

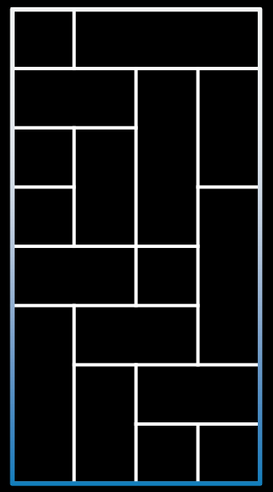
Experiments power the engine of rapid innovation



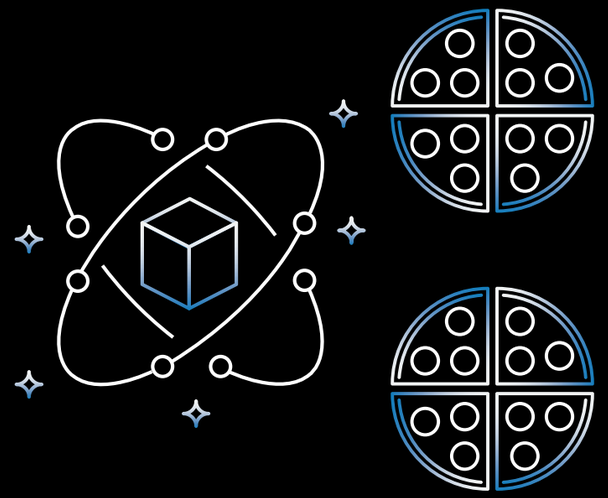
Development transformation at Amazon: 2001–2009

Lesson learned: Decompose for agility

2001 ----------> **2009**

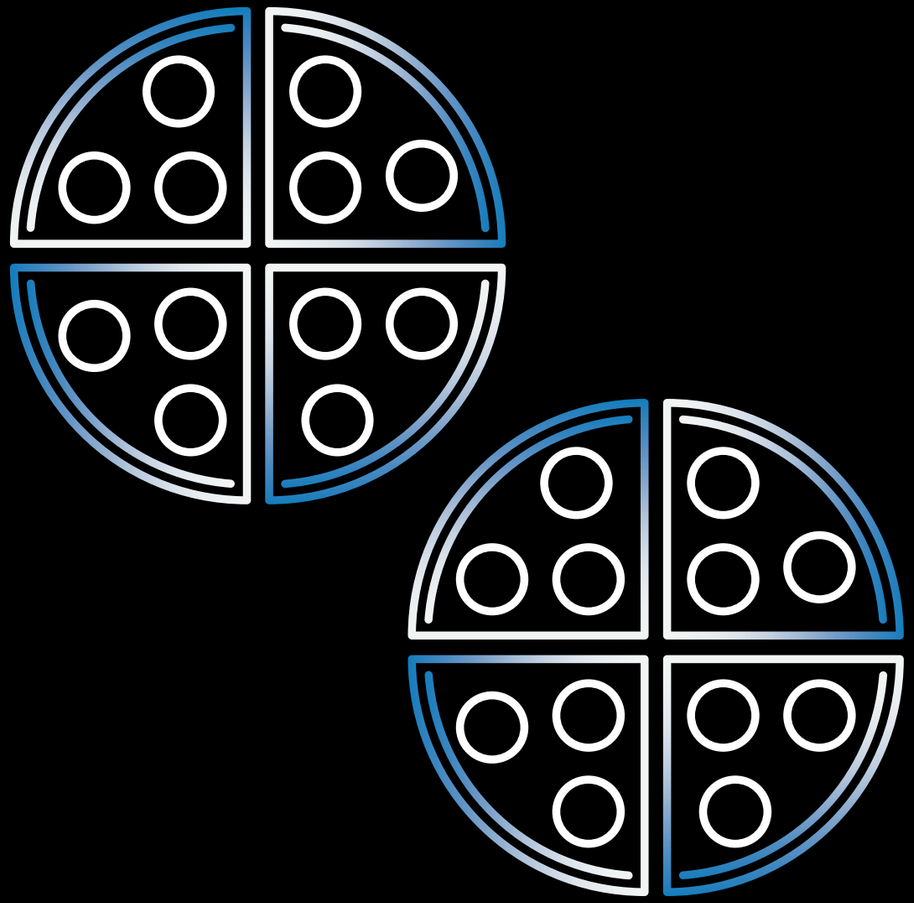


Monolithic application
+ teams



Microservices
+ 2 pizza teams

Two-pizza teams



Full ownership

Full accountability

"DevOps"

Focused innovation

This transformation takes

Architectural patterns

Operational model

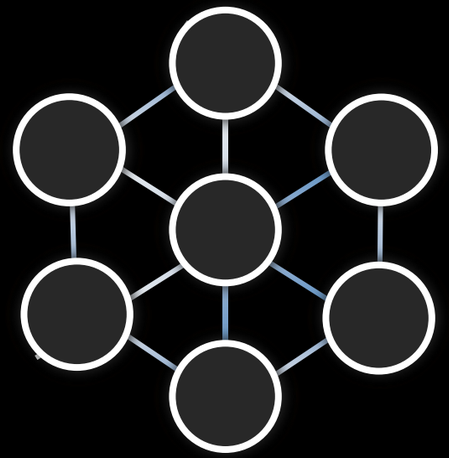
Software delivery

Changes to the architectural patterns

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

Monolith

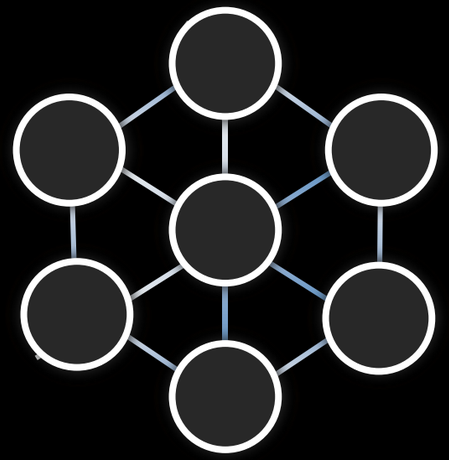
Does everything



Microservices

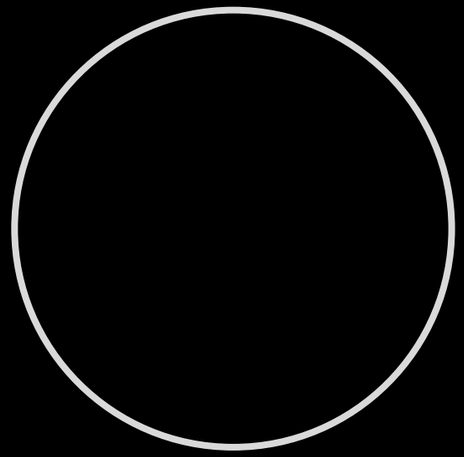
Does one thing

When a small team can make a change, release velocity can increase



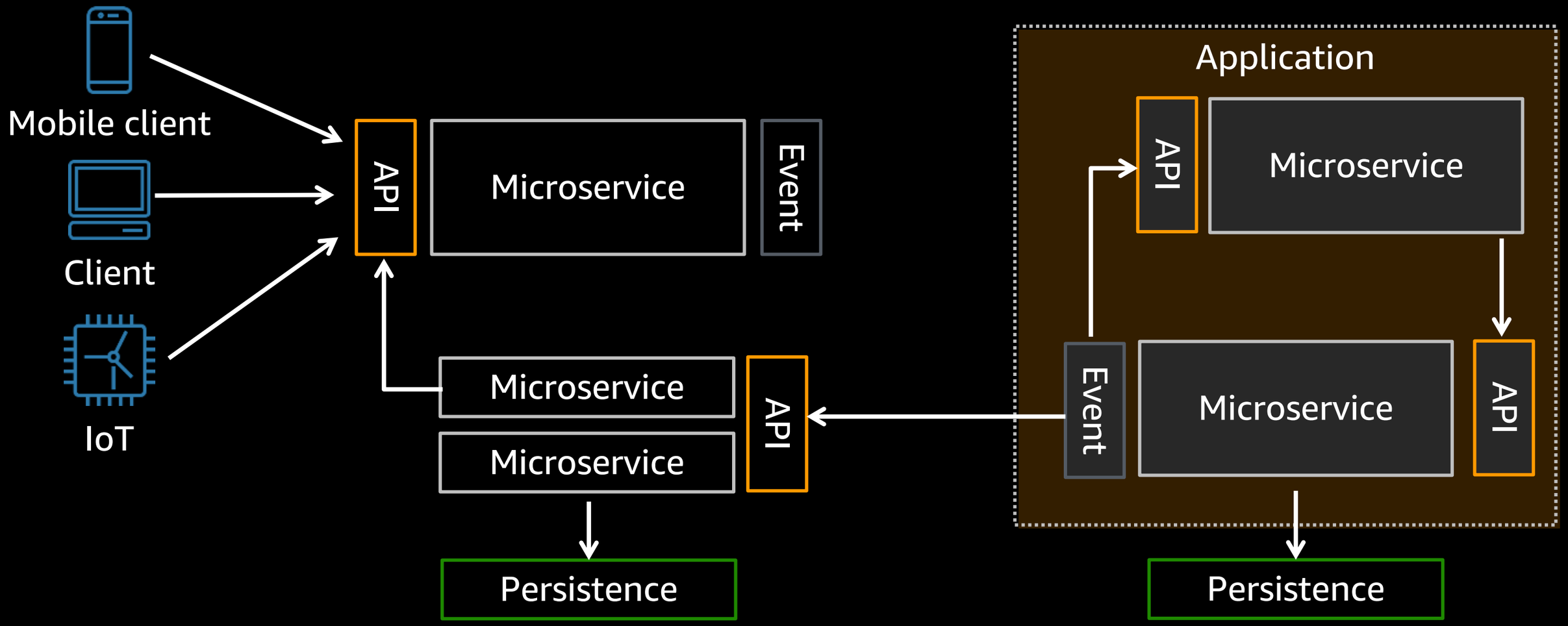
Multiple teams

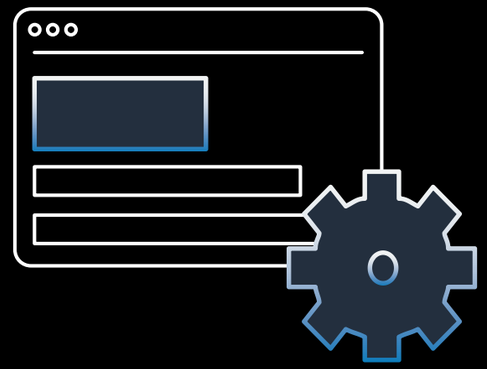
Loosely connected



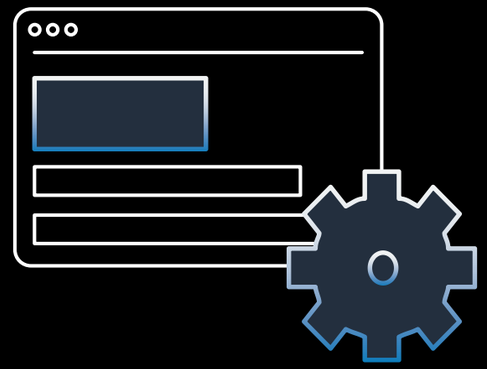
One team

Focus and ownership



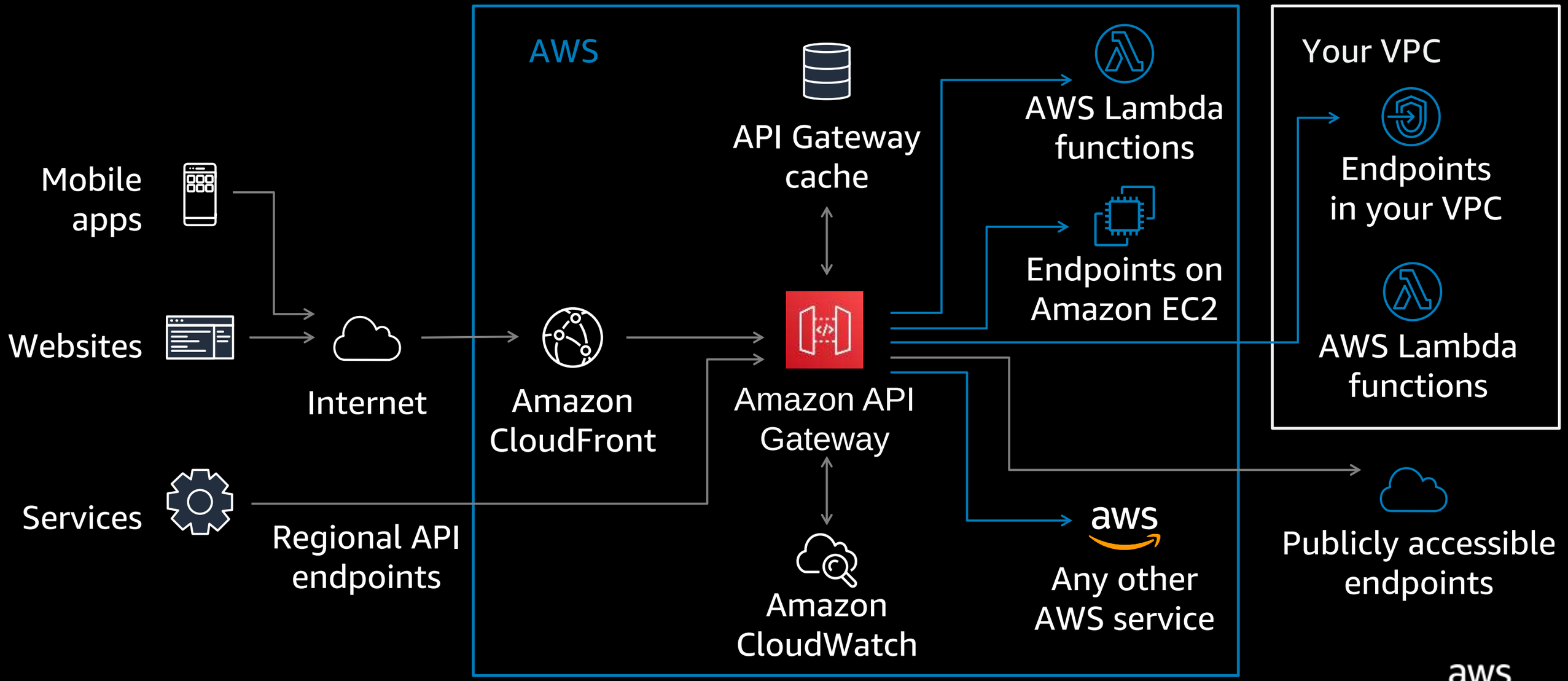


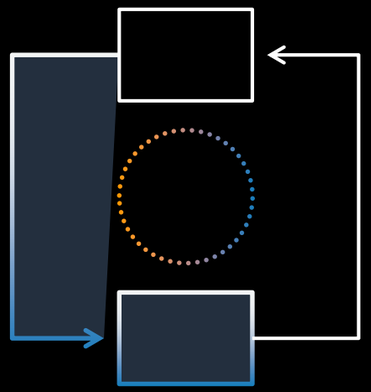
APIs are the front door of microservices



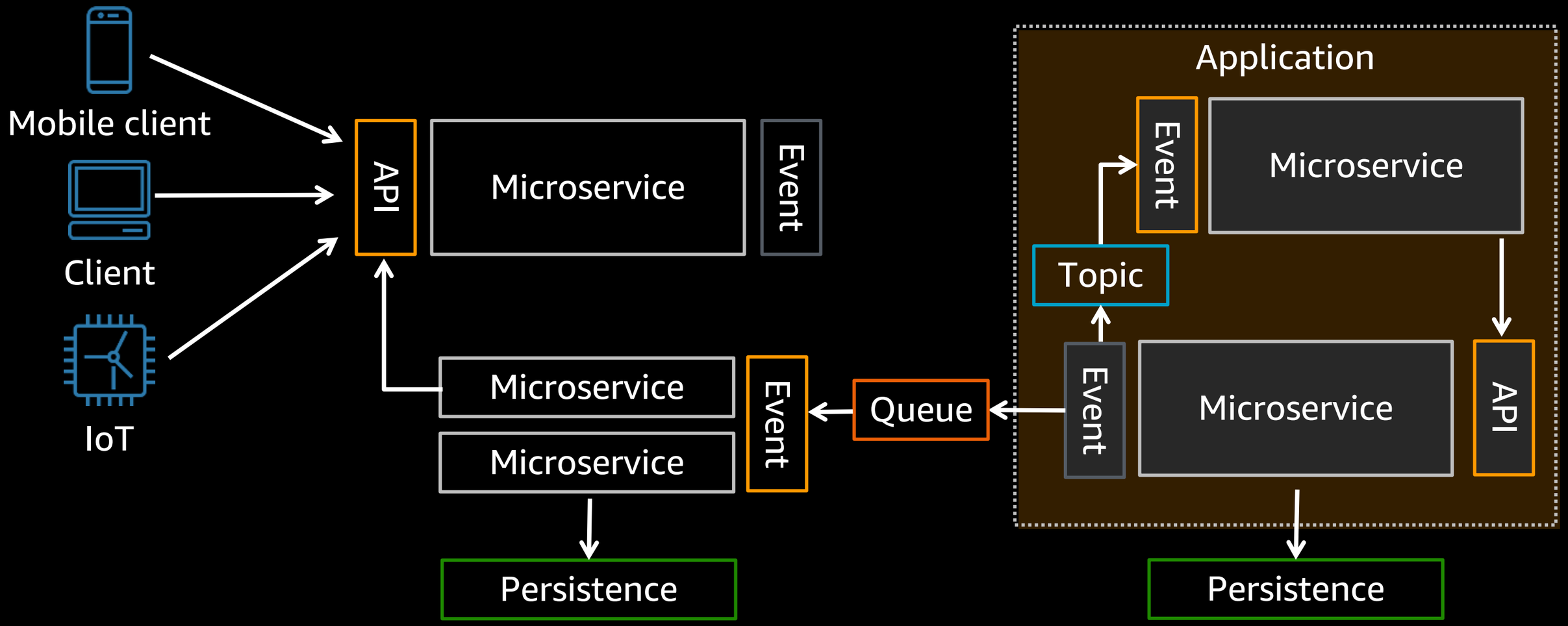
APIs are the “hardened contract” between teams

Manage APIs with Amazon API Gateway





Decouple using messaging



Decouple state from code using messaging

Messaging

Amazon Simple Queue Service

Queues

Simple

Fully managed

Any volume

Amazon Simple Notification Service

Pub/sub

Simple

Fully managed

Flexible

Amazon EventBridge

Event Bus

Rapid

Fully managed

Real time

And data streams

Data stream capture

Amazon Kinesis Data Streams

Ingest

Data streams

Data processing

Real time

Amazon DynamoDB

Data store

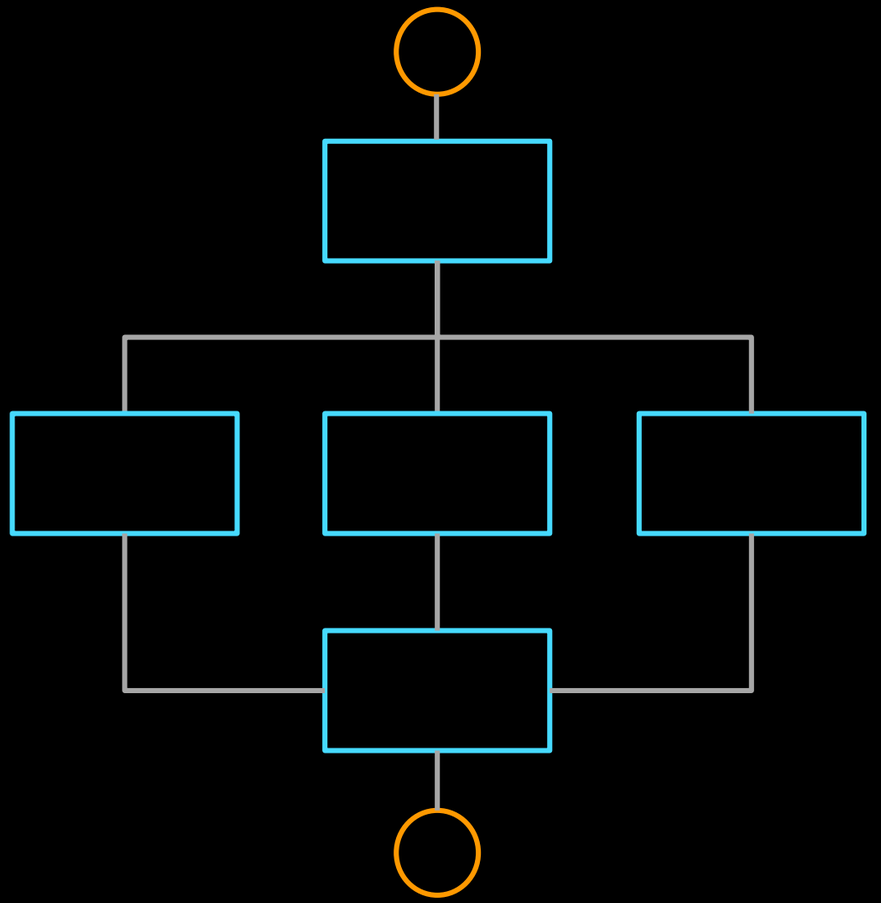
Microservices

Performance at scale

Fast and flexible

Build workflows to orchestrate microservices

Track status of data and execution



Remove redundant code

Build workflows to orchestrate microservices

Workflow Management

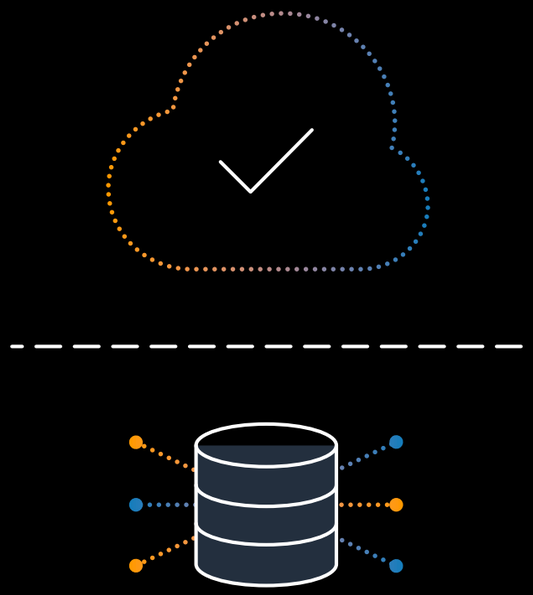
AWS Step Functions

Ingest

Auditable

Intuitive

Long-running



Microservice architectures are small pieces, loosely joined

Changes to the operational model

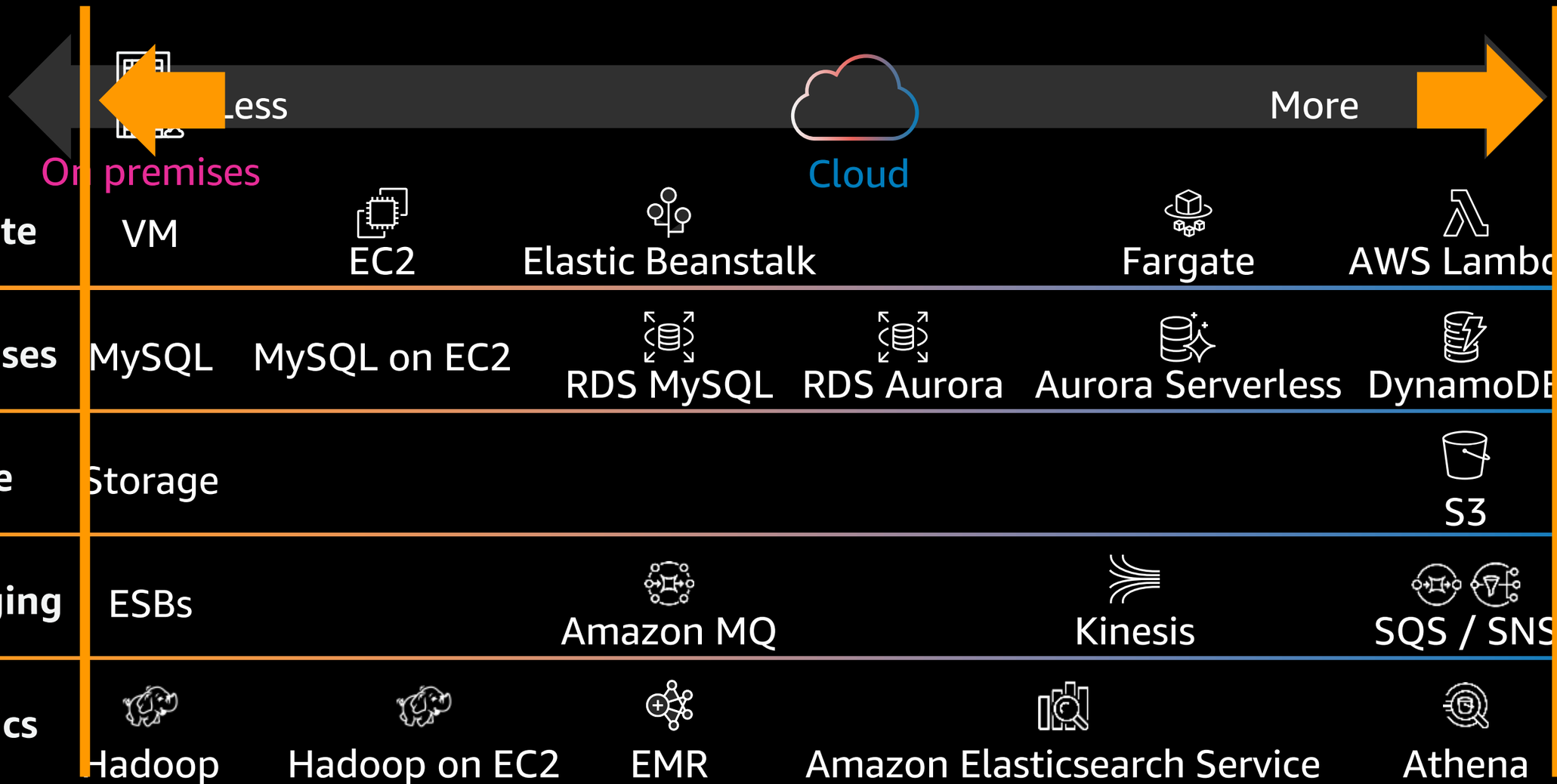
AWS operational responsibility models

Compute	VM	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	Amazon Elasticsearch Service		Athena

This is where the differentiation happens…

Compute	VM	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	Amazon Elasticsearch Service		Athena

. . . and this requires skill, work, and experience.



What is serverless?

No infrastructure 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

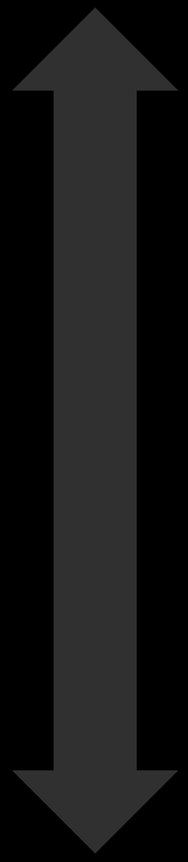

Amazon SNS


AWS Step Functions


AWS AppSync

Comparison of operational responsibility

More opinionated



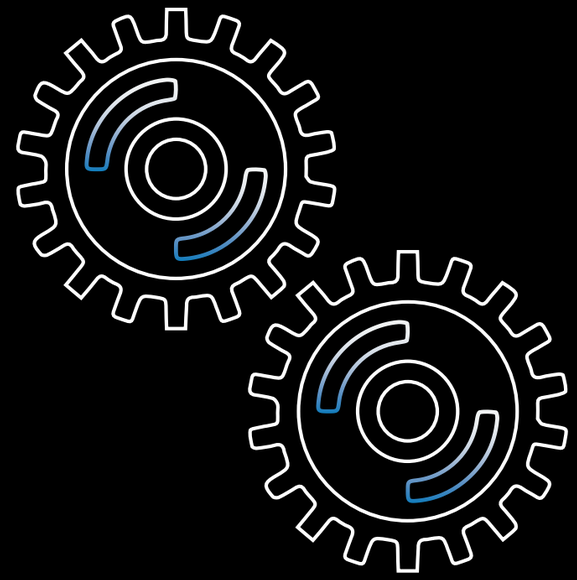
Less opinionated

	AWS manages	Customer manages
AWS Lambda Serverless functions	• Data source integrations • Physical hardware, software, networking, and facilities • Provisioning	• Application code
AWS Fargate (new: EKS+ECS) 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

Lambda Layers & Custom Runtimes

Lambda Layers

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



Custom Runtimes

Bring any Linux compatible language runtime

Community Lambda Layers

Custom Runtimes

C++

Rust

Erlang

Bash

Node.js v10, v11

PHP 7.1, 7.2, 7.3

Pypy 3.5

Haskell

Utilities

AWS CLI

Ffmpeg

Git + SSH

Kubectl

MySQL + PHP

SoX

Tesseract

Pandoc

Monitoring

Datadog

Epsagon

IOpipe

Thundra

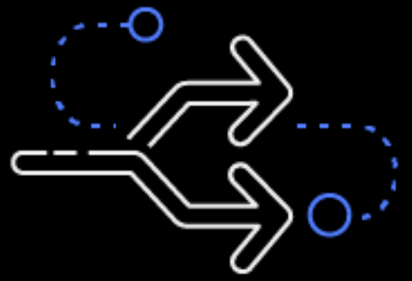
Security

PureSec

Protego

AWS X-Ray to analyze and debug distributed apps

Map all services and ephemeral resources



Follow end to end interactions

Identify customer impact

Support for serverless

Application-level networking for all your resources

- Consistent microservice communications
- Observability & traffic control
- Container orchestration compatibility
- Fully managed

What's next? Public Roadmaps on GitHub!

Container Roadmap

Amazon Elastic Container Registry

Amazon Elastic Container Service

Amazon Elastic Container Service for Kubernetes

AWS Fargate

AWS App Mesh Roadmap

AWS App Mesh

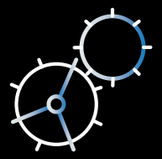
CloudFormation Coverage

AWS CloudFormation

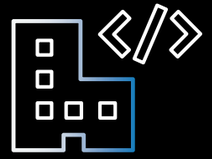
Changes to the delivery of software

How Amazon does DevOps

Decompose for agility
(microservices, 2 pizza teams)



Automate everything

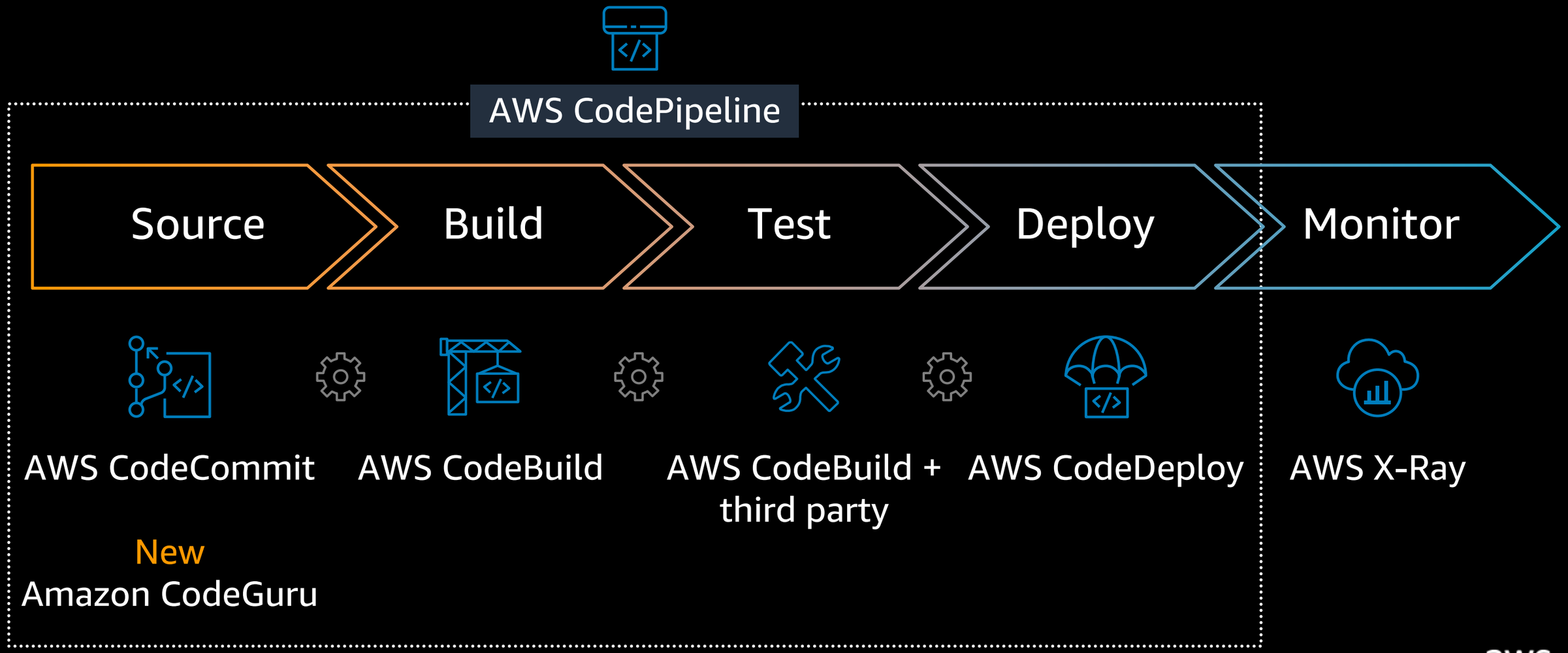


Infrastructure as code

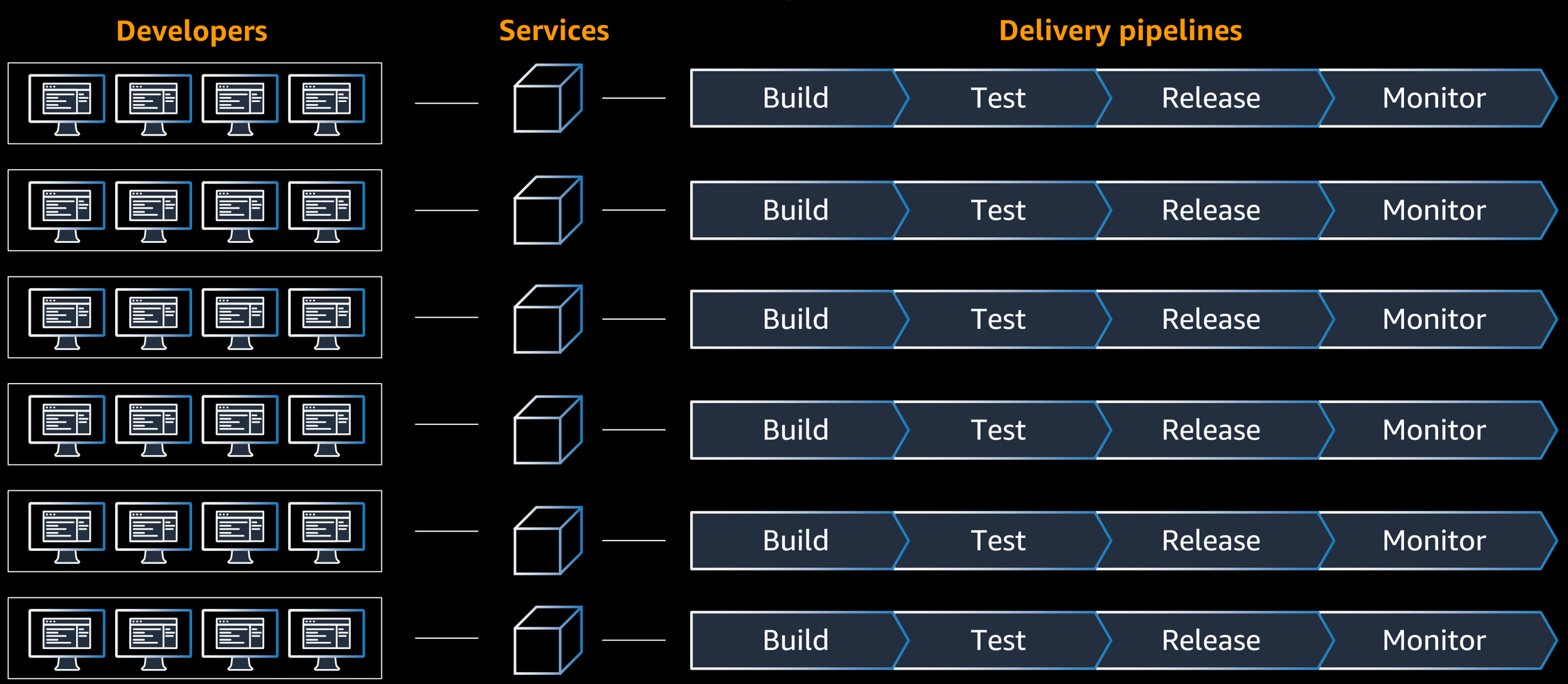
Belts and suspenders
(governance, templates)

Standardized tools

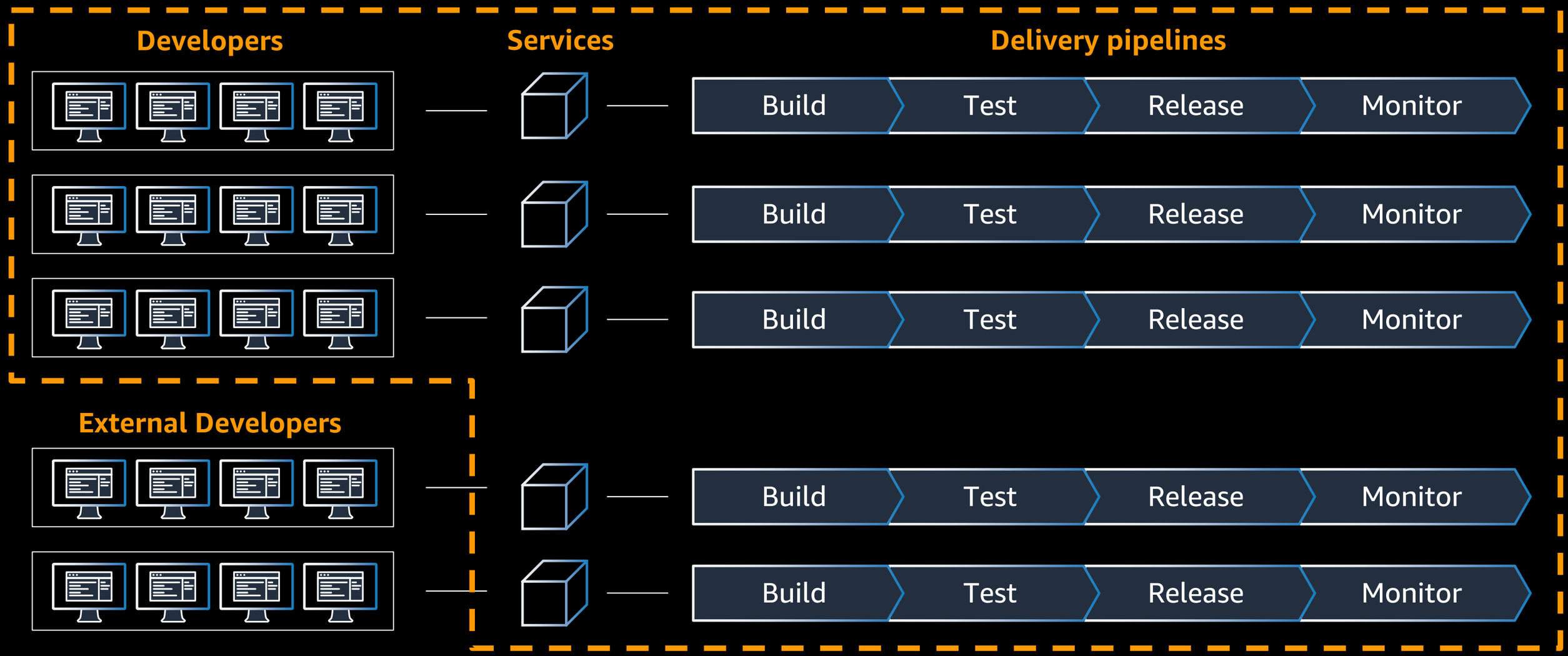
We Released the AWS Developer Tools for CI/CD



Each Service has its own Pipeline



Own the pipeline for external deliverables!



Use a Cookie-cutter approach!

Provisioning

AWS CodeStar

Toolchain

Fast

Opinionated

AWS CloudFormation

Custom Templates

Flexible

Any Service

Any Architecture

AWS Service Catalog

Custom Catalog

Governance

Tight Permissions

AWS Control Tower

Account Vending

Multi-Account

Minimal Blast Radius

Conclusion

Working Backwards: Serverless is a logical conclusion

Thank you!

Steffen Grunwald

Feedback?