
aws
INNOVATE
MODERN APPLICATIONS EDITION
27&28 October 2021

Evolving monolith to microservices — architecture patterns, software delivery and operational models

Anshul Sharma

Senior Solutions Architect
Amazon Web Services

@anshuldsharma

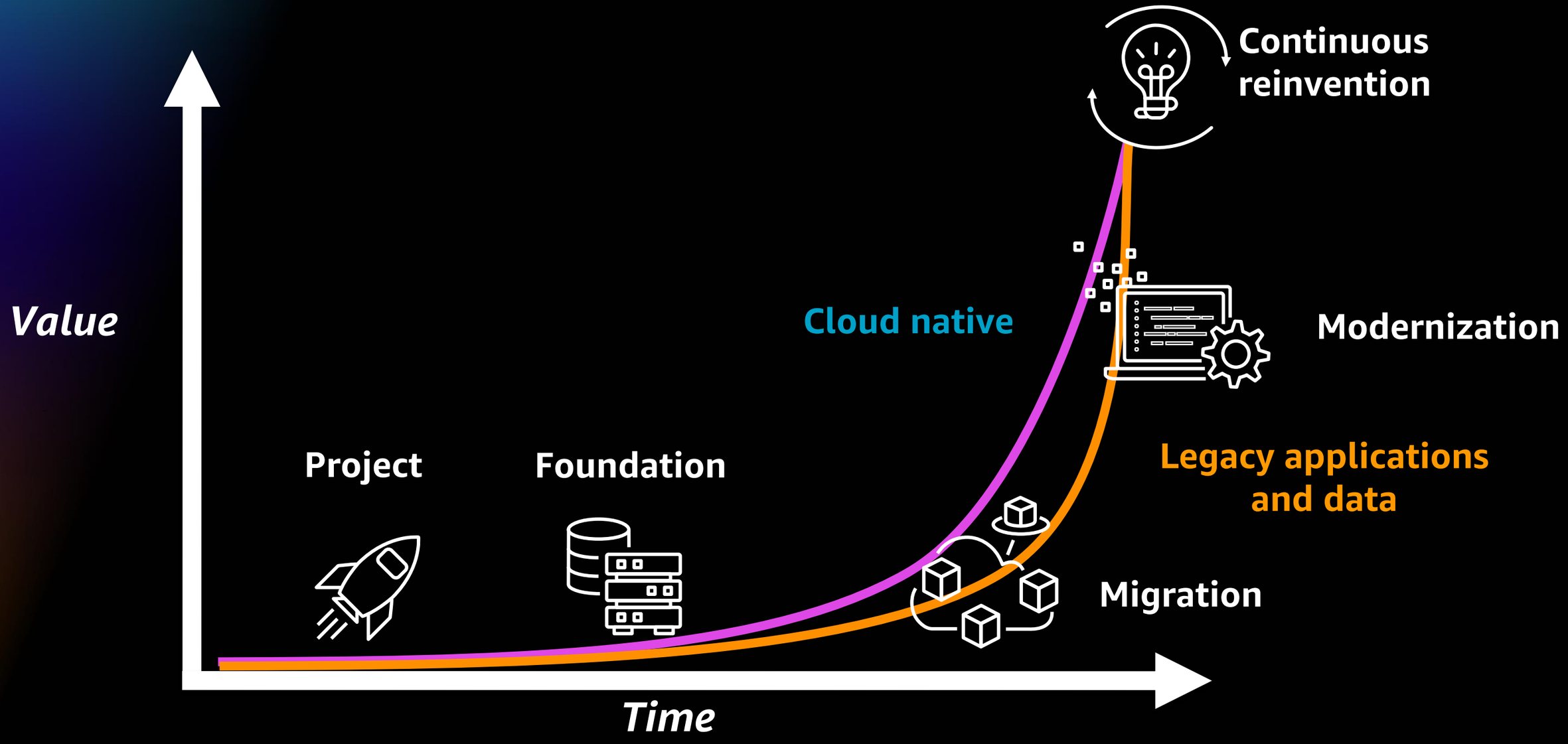
aws

Agenda

- Migration & modernization
- Breaking down the monolith into microservices to build modern applications
    - Architecture patterns
    - Operational model
    - Software delivery

Migration & modernization

Getting started in your cloud journey



Migration planning

Current IT snapshot

Asset inventories

App configuration data

Performance information

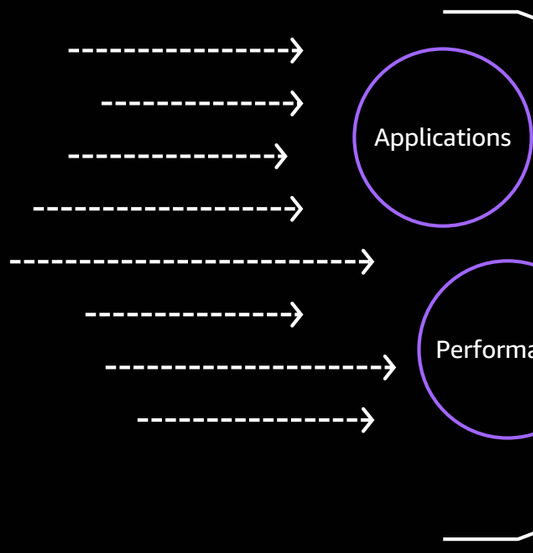
CMDB

SLA/OLA

Architecture

Tribal knowledge

Discover & organize data



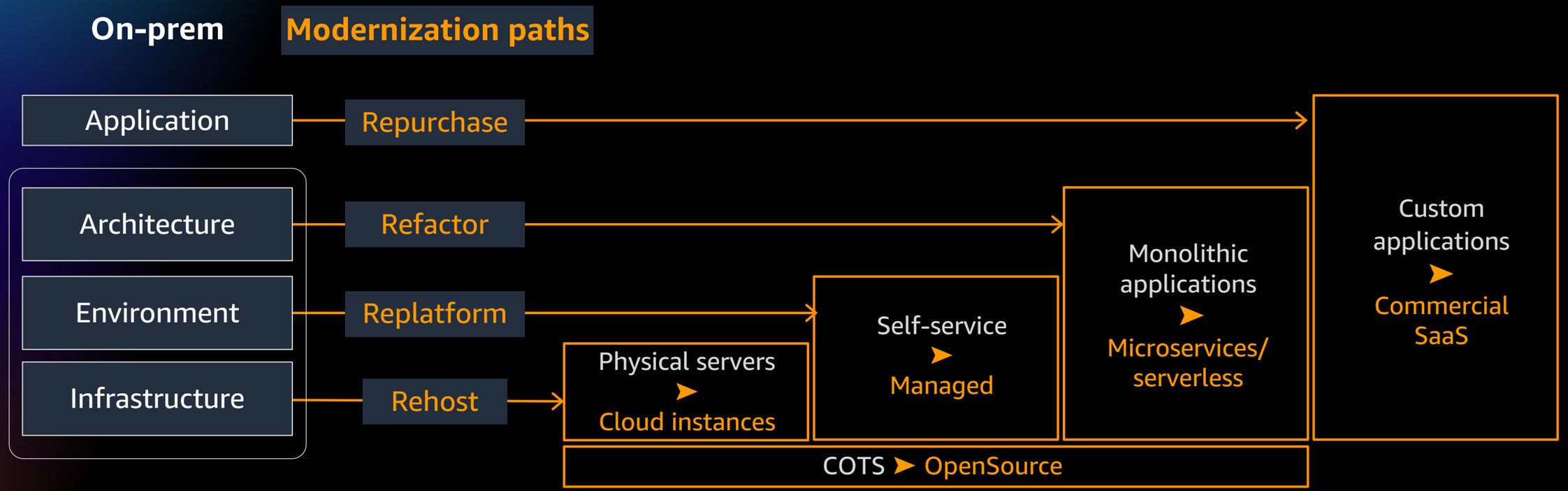
Migration strategies for each workload (i.e. 7Rs)

- Refactor
- Replatform
- Repurchase
- Rehost
- Relocate
- Retain
- Retire

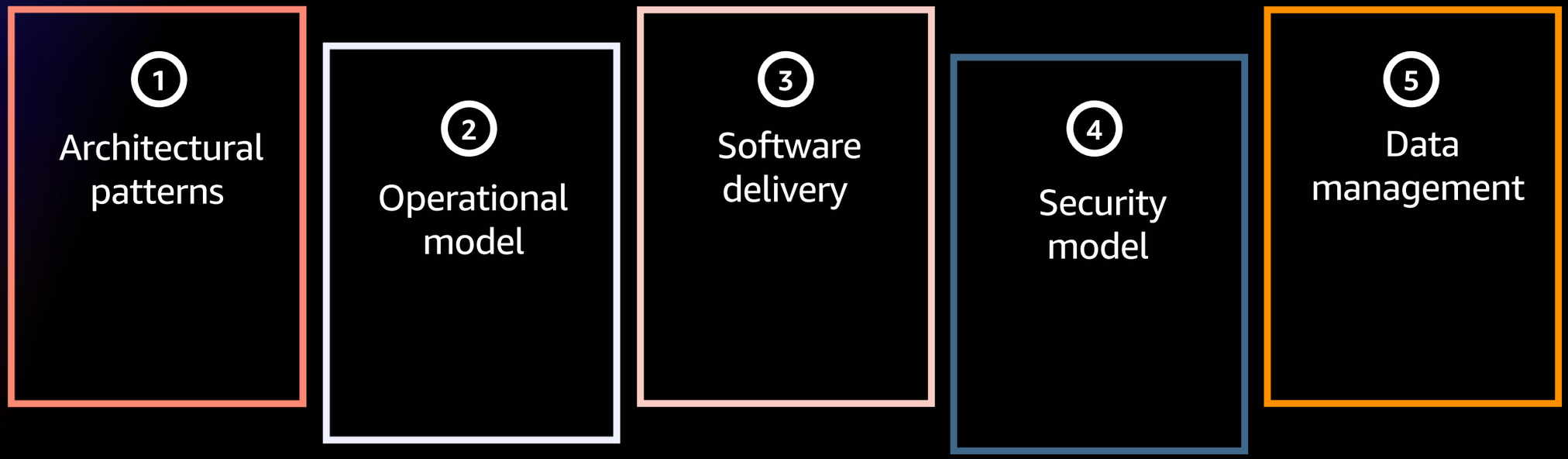
Level of effort

Migration strategy decision criteria should be based on both business and technical needs

Modernization example



What changes have to be made in our new world?



Changes to the architectural patterns

Monoliths are OK

Monolith

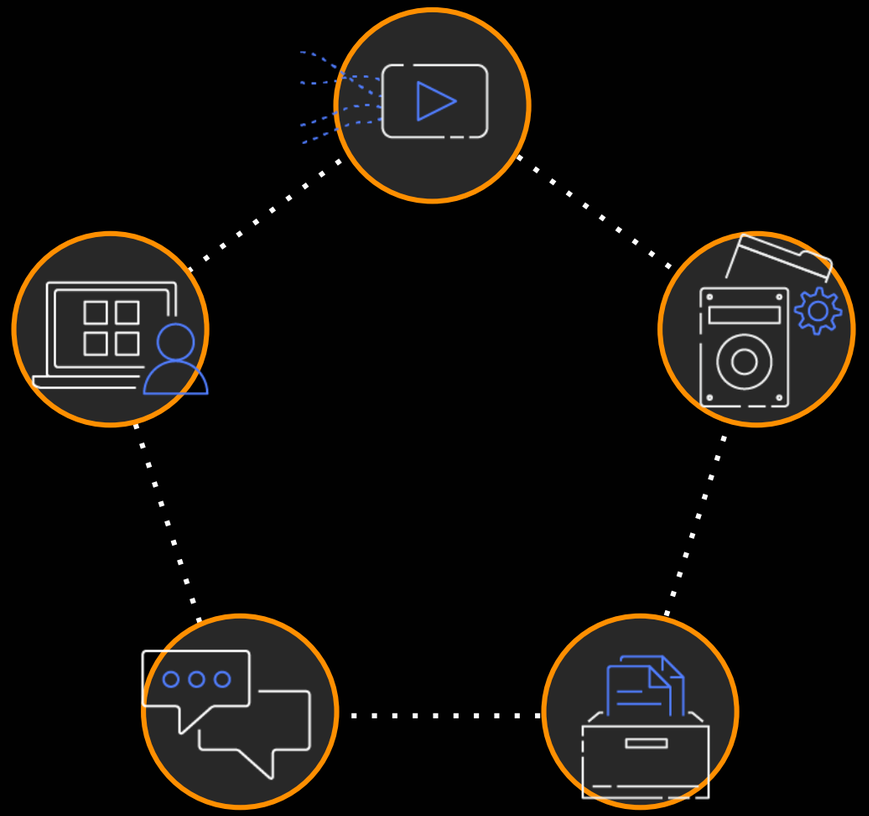
Does everything

Monolith challenges

- Poor agility
- Tightly coupled
- Over-provisioning for scale
- Operational management
- Achieve high availability
- Hard to fail fast

Microservices

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



Microservices

Does one thing

Common questions

How do I break the monolith?

How do I get started?

What workloads do I move first?

How do we manage workloads in the cloud?

What do I have in my environment?

How do I get my team re-skilled?

What should I move to the cloud?

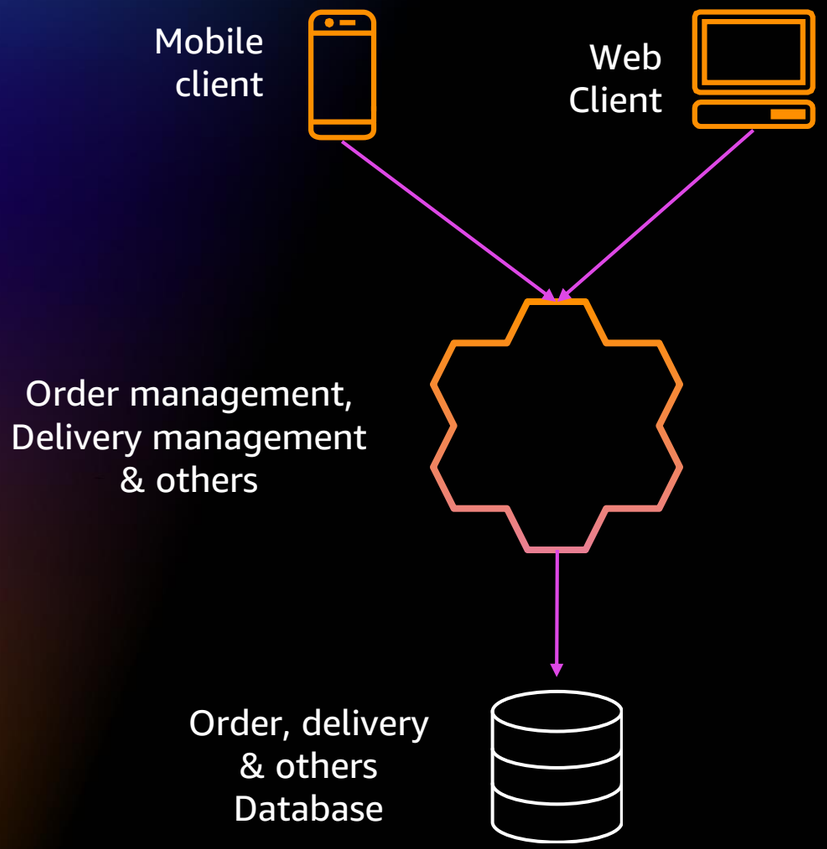
How do I migrate these workloads?

Microservices refactoring

1. Implement new functionality as services
2. Extract services from the monolith

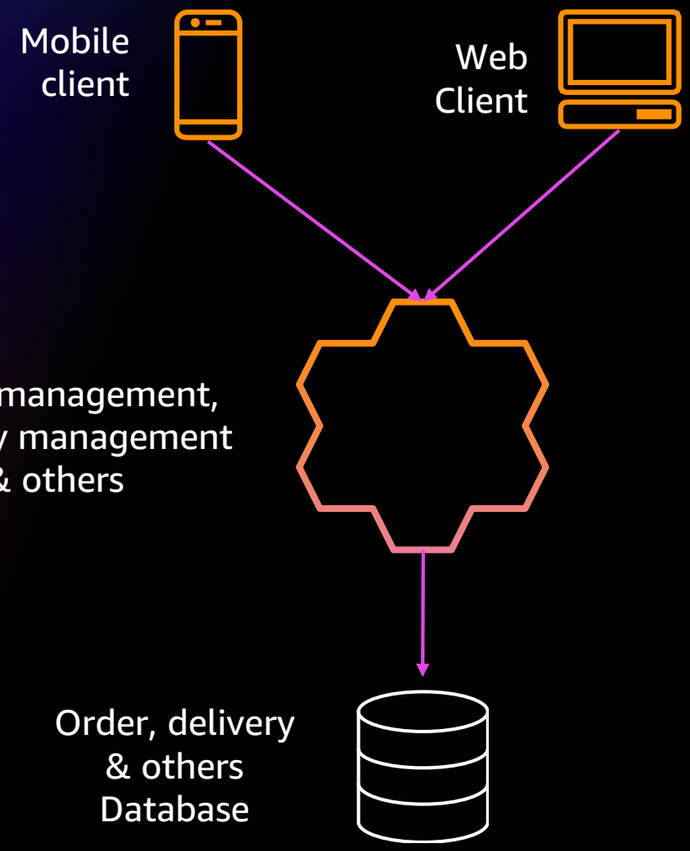
Microservices refactoring

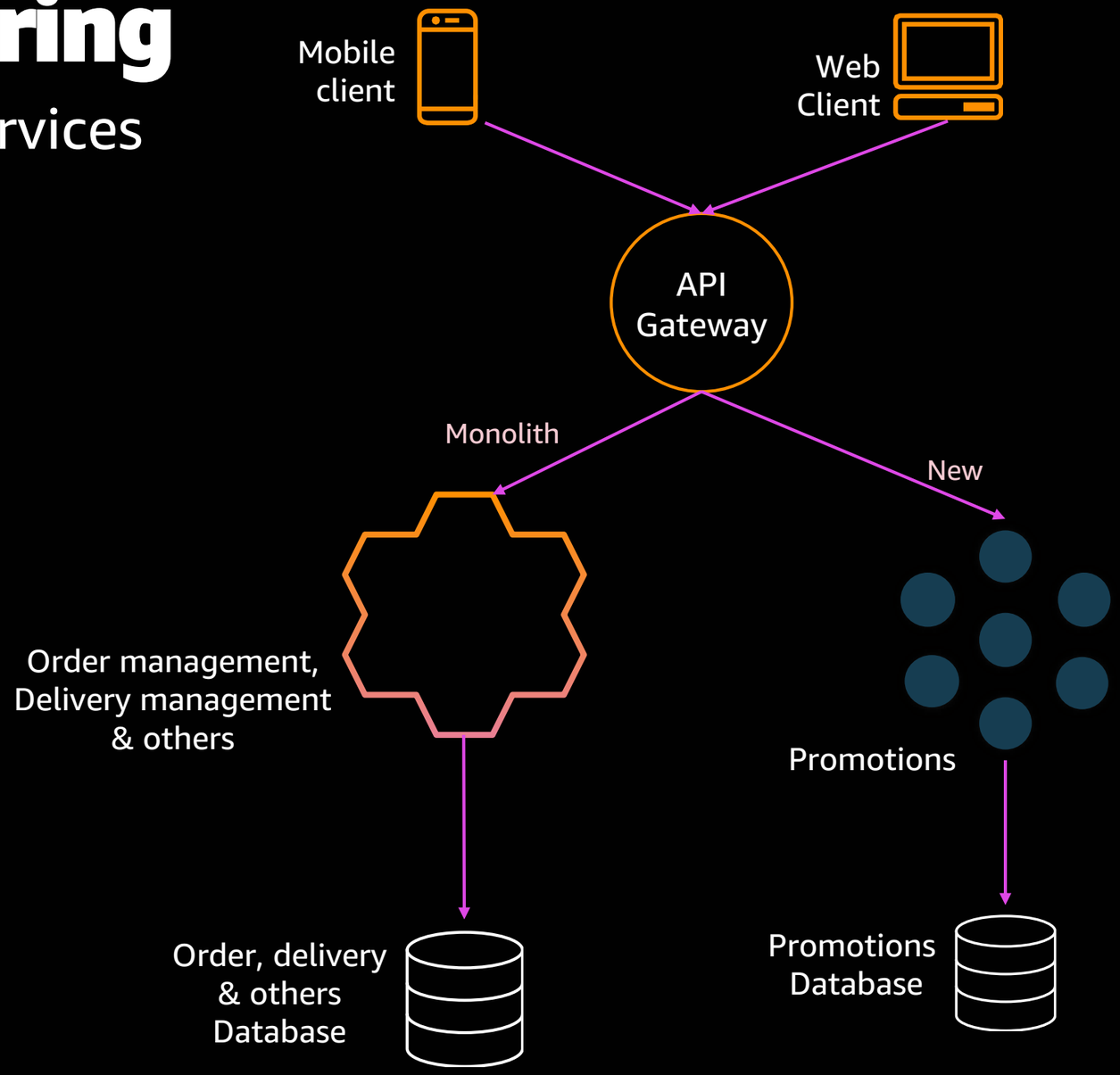
- Implement new functionality as services



Microservices refactoring

- Implement new functionality as services



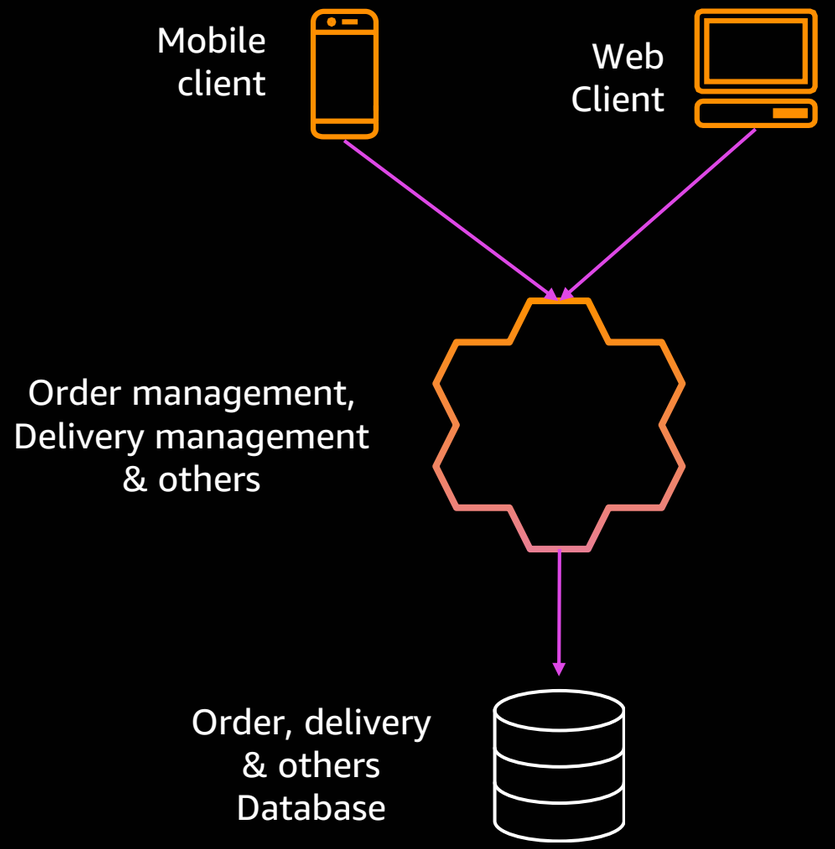


Breaking the Monolith

How do I get started with decoupling?

- Start simple
- Decouple capability, not code
- Capabilities that change frequently and are important to the business
- Minimize dependency back to Monolith
- Macro vs micro

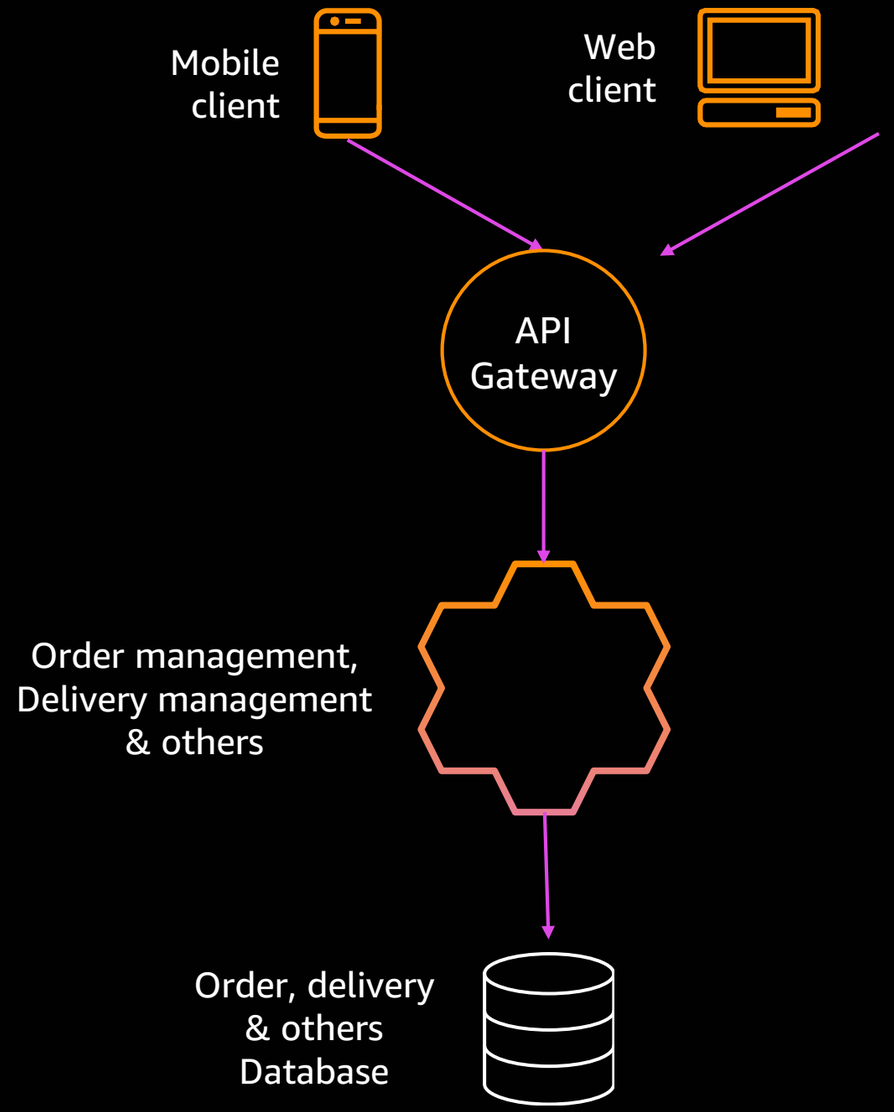
Sample monolith application



Add API Layer for abstraction

- Security
- Resiliency
- Operations monitoring
- Real-Time
- Lifecycle management
- Metering

… and more

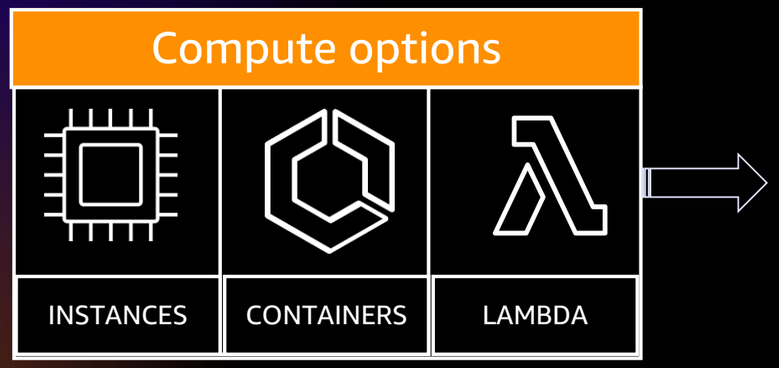


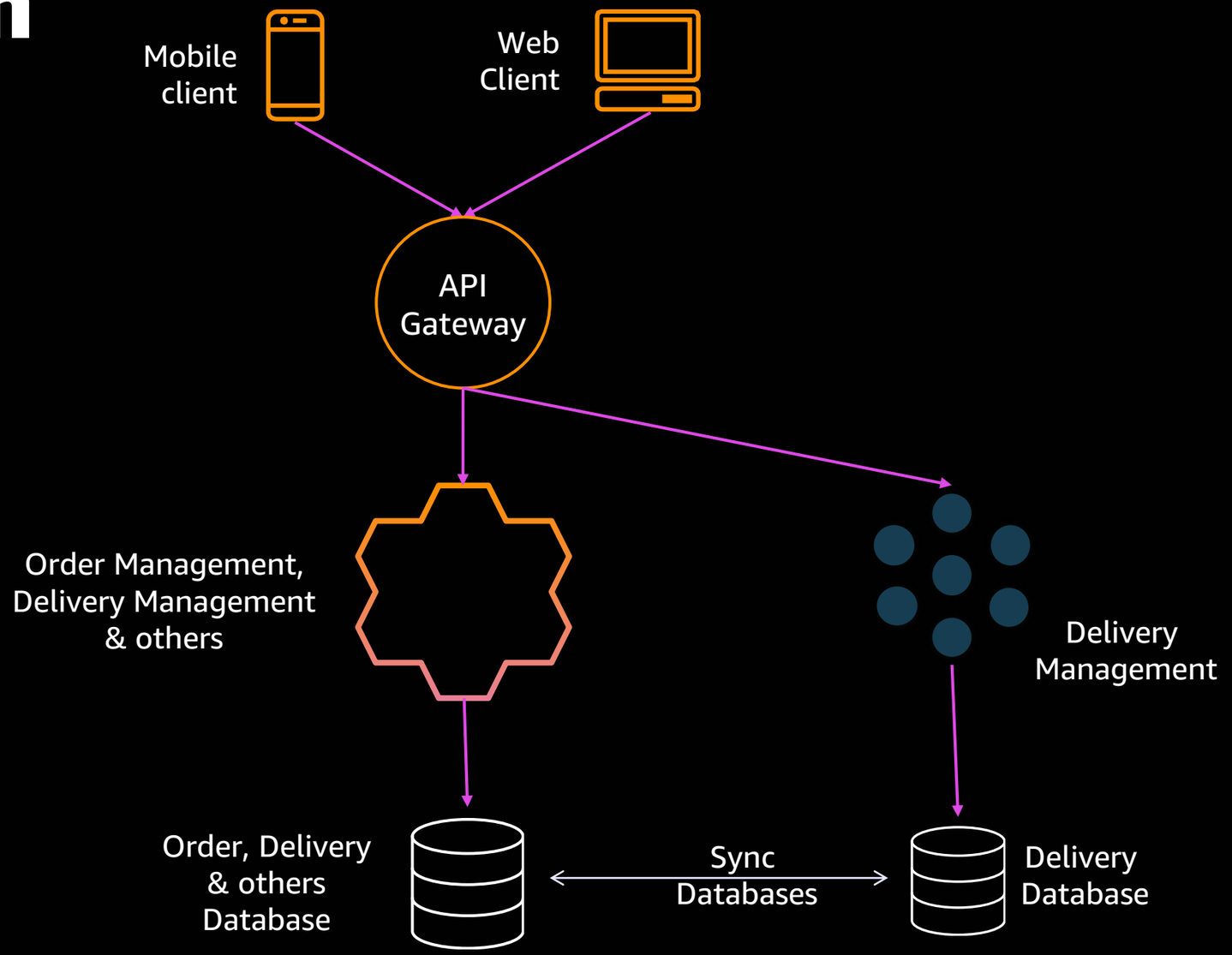
Strangler Pattern

Retire & Rewrite

vs

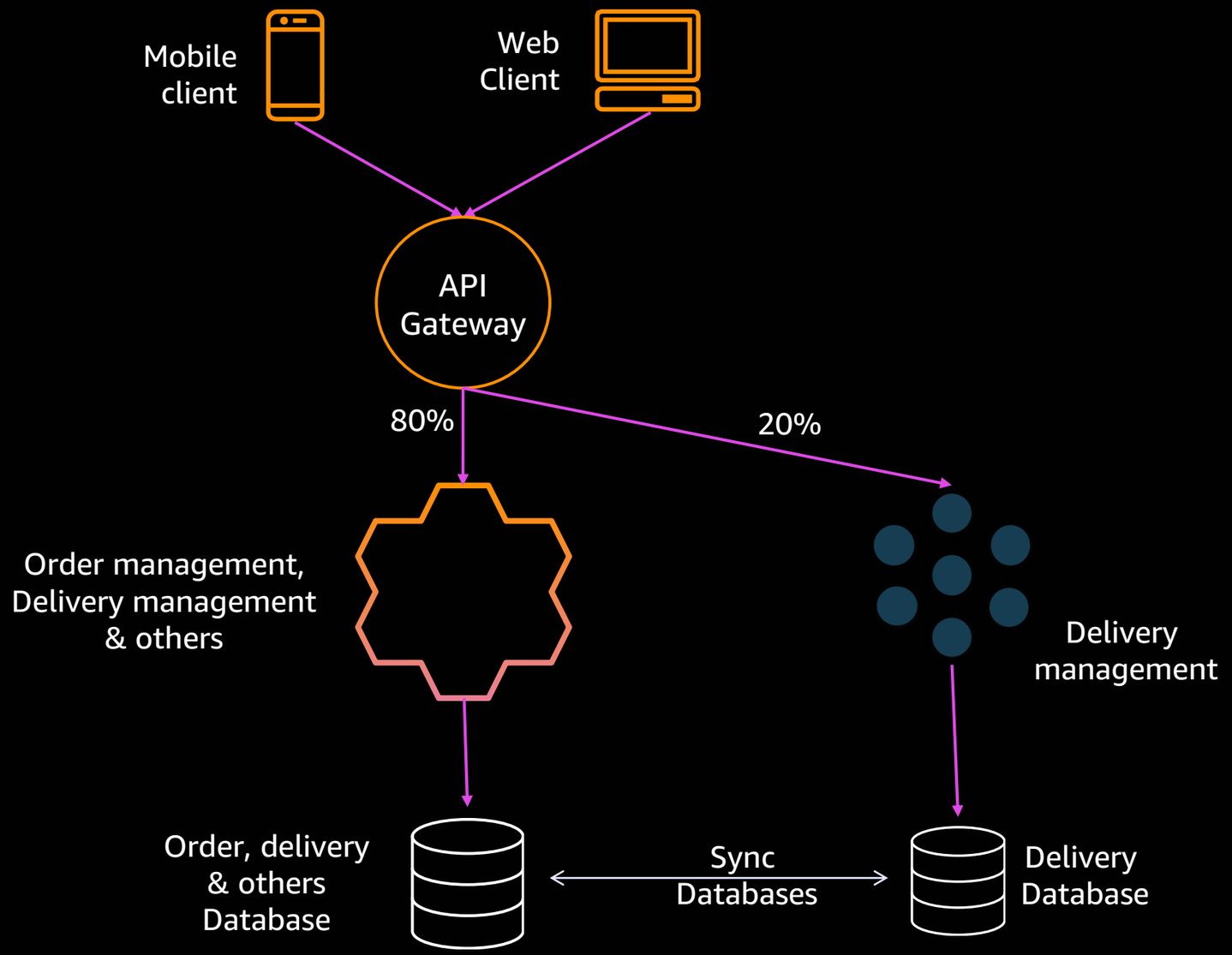
Extract & Reuse





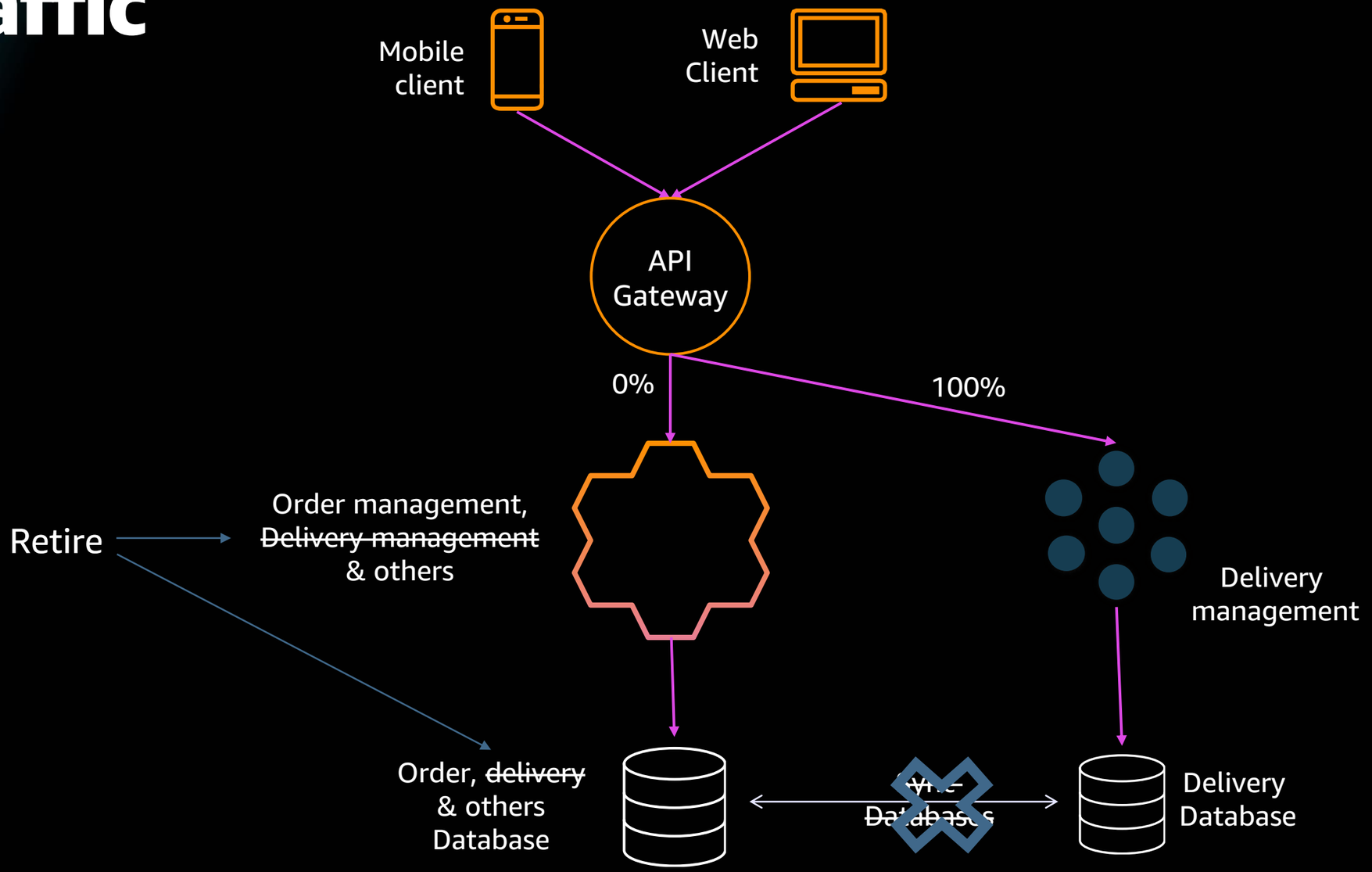
Toggle Traffic

Test application rollback, if required



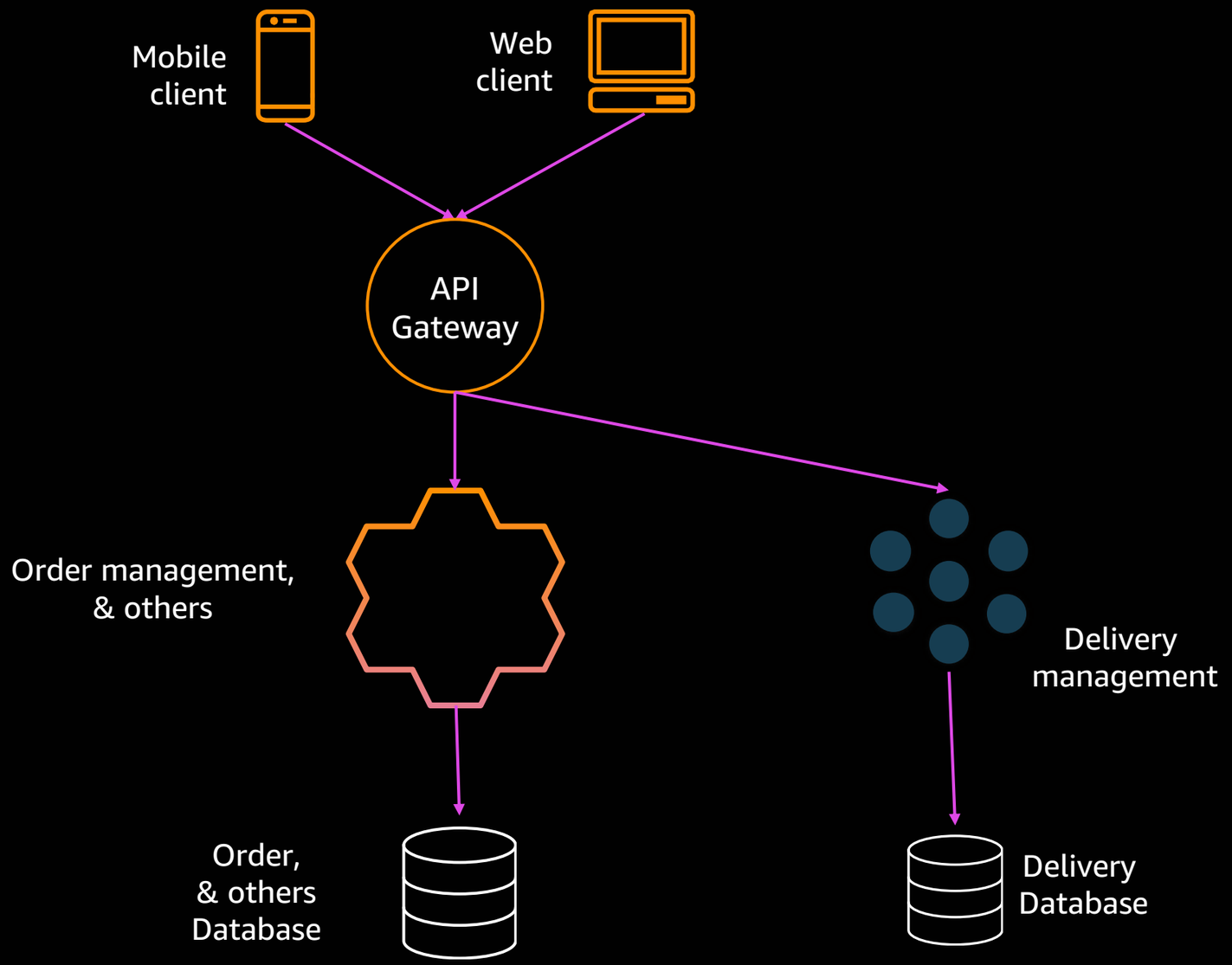
Toggle Traffic

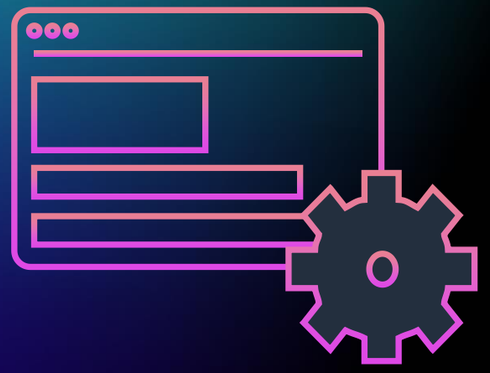
Redirect traffic
Retire old code



Co-existence

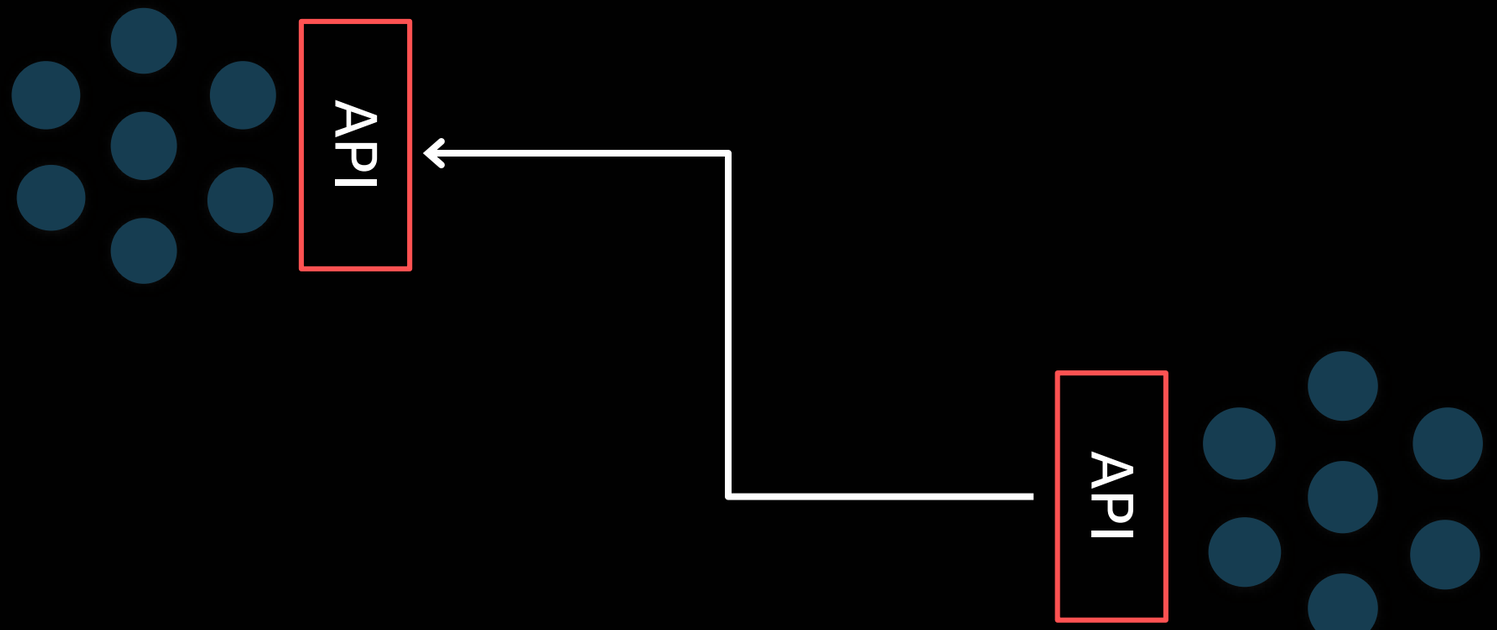
Continue refactoring the monolith





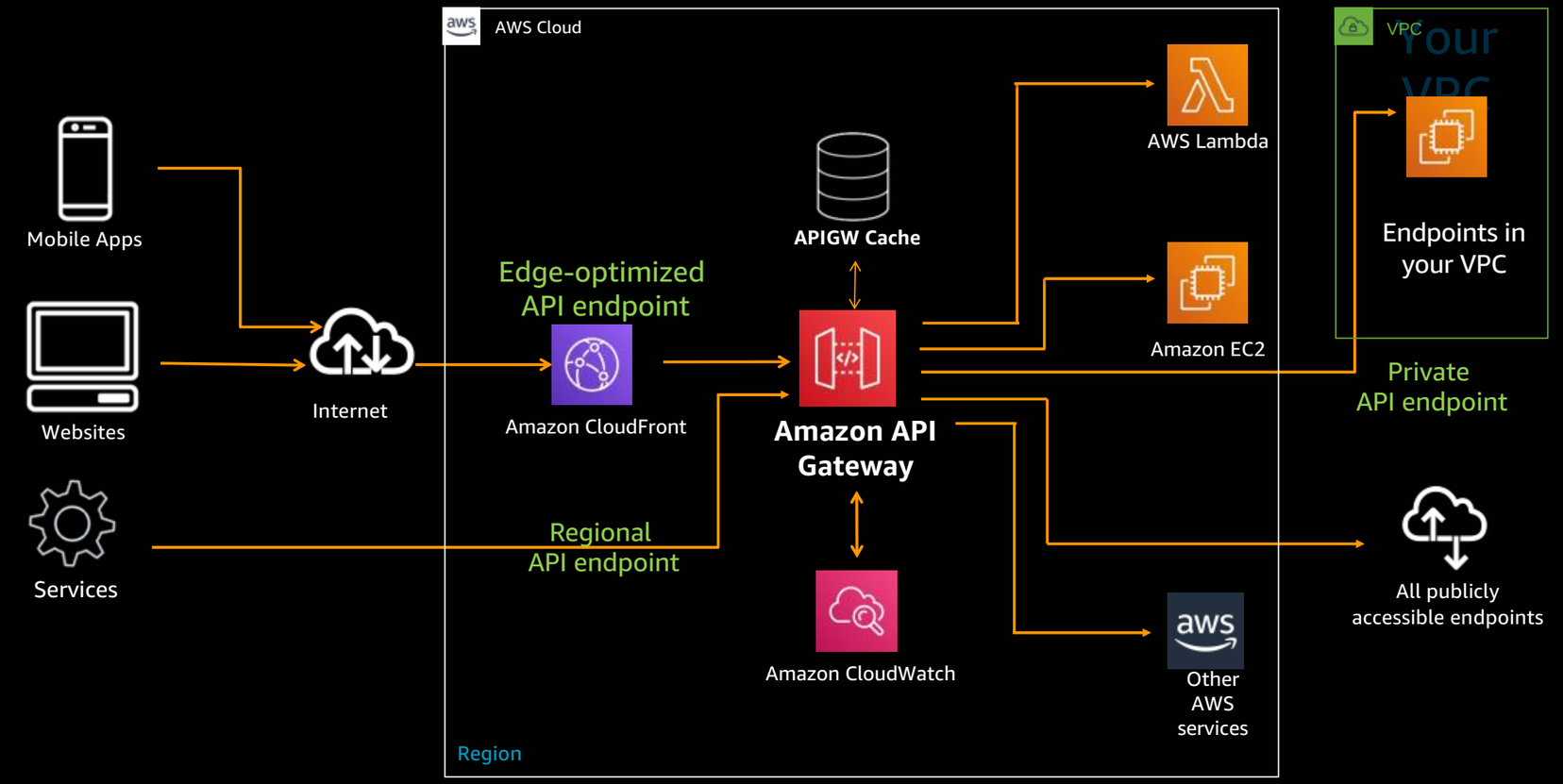
APIs are the front door of microservices

APIs are hardened contracts



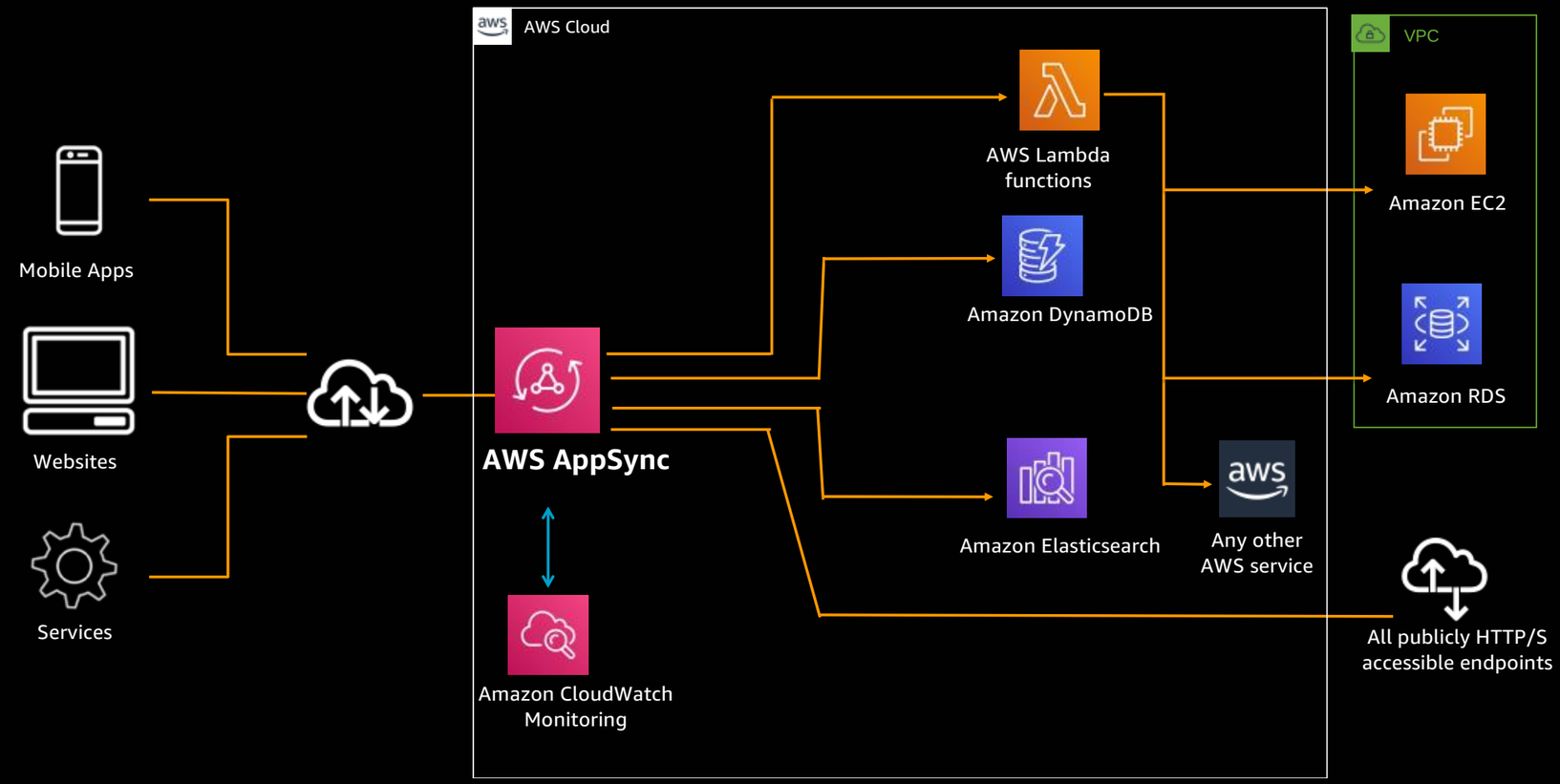
Amazon API Gateway – RESTful APIs and WebSocket APIs

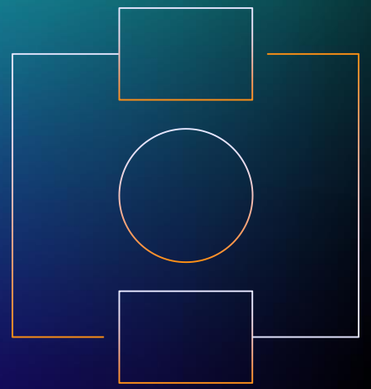
- Create a unified API frontend for multiple microservices
- DDoS protection, caching and throttling for your backend
- Authenticate and authorize requests to a backend
- Throttle, meter, and monetize API usage by third-party developers



AWS AppSync – Managed GraphQL APIs

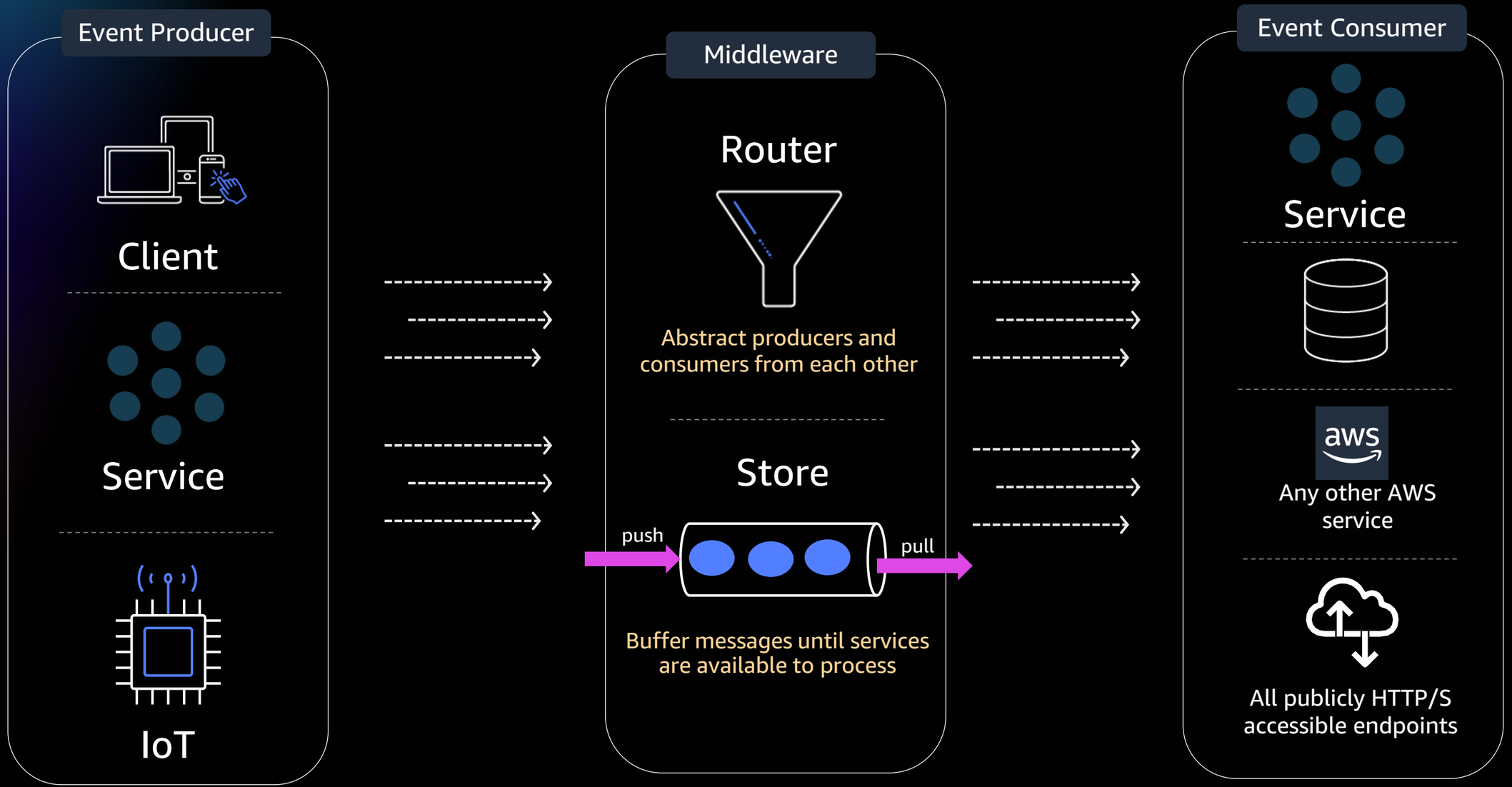
- Offline data synchronization - Interact with and update your data, even when offline, with the AWS Amplify DataStore
- Data querying, filtering, and search in app with preconfigured access to AWS data sources
- Enterprise security and fine-grained access control





Events are the connective tissue of modern applications

Event-driven architecture



It’s not just about computing & infrastructure!

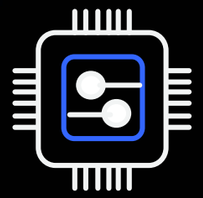
Data model and store

Relational

Referential integrity, ACID transactions, schema-on-write

Common Use Cases: Lift and shift, ERP, CRM, finance

AWS Service(s): **Amazon Aurora, Amazon RDS**

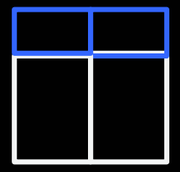


In-memory

Query by key with microsecond latency

Common Use Cases: Leaderboards, real-time analytics, caching

AWS Service(s): **Amazon ElastiCache**

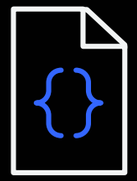


Key-value

High throughput, low-latency reads and writes, endless scale

Common Use Cases: Real-time bidding, shopping cart, social, product catalog, customer preferences

AWS Service(s): **Amazon DynamoDB**

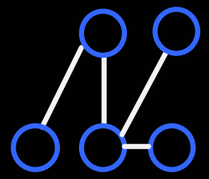


Document

Store documents and quickly access querying on any attribute

Common Use Cases: Content management, personalization, mobile

AWS Service(s): **Amazon DocumentDB**



Graph

Quickly and easily create and navigate relationships between data

Common Use Cases: Fraud detection, social networking, recommendation engine

AWS Service(s): **Amazon Neptune**

Time-series

Collect, store, and process data sequenced by time

Common Use Cases: IoT applications, event tracking

AWS Service(s): **Amazon Timestream**

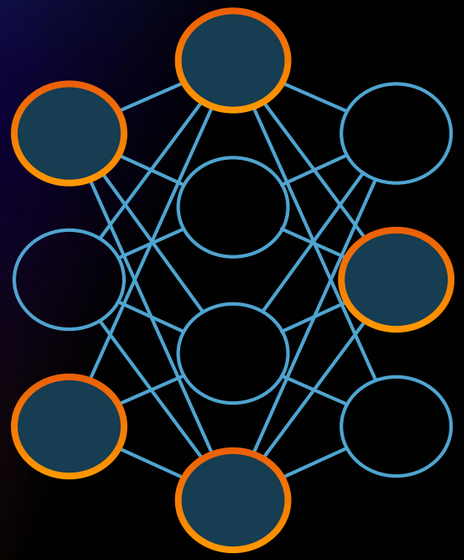
Ledger

Complete, immutable, and verifiable history of all changes to application data

Common Use Cases: Systems of record, supply chain, health care, registrations, financial

AWS Service(s): **Amazon QLDB**

Changes to the operational model



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

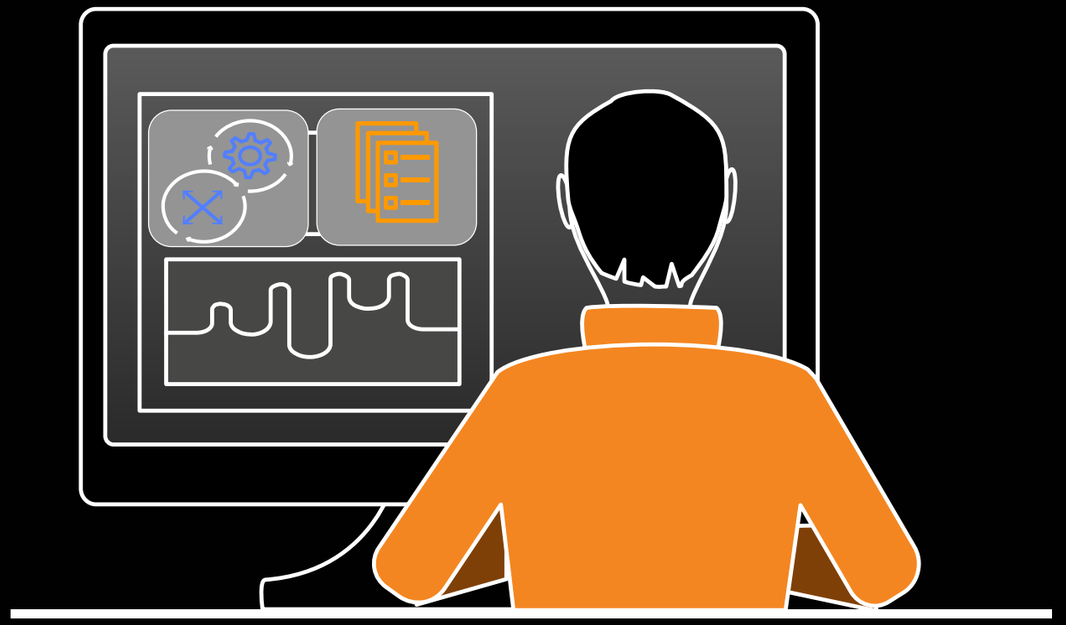
AWS operational responsibility models

	On-premises	Cloud				
Compute	Virtual machine	Amazon EC2	AWS Elastic Beanstalk		AWS Fargate	AWS Lambda
Databases	MySQL	MySQL on Amazon EC2	Amazon RDS for MySQL	Amazon RDS	Amazon Aurora	Amazon DynamoDB
Storage	Storage					Amazon S3
Messaging	ESBs		Amazon MQ	Amazon Kinesis		Amazon EventBridge/ Amazon SQS/Amazon SNS
Analytics	Hadoop	Hadoop on EC2	Amazon EMR	Amazon OpenSearch Service		Amazon Athena

Comparison of operational responsibility – Compute

	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, and management tasks
Amazon ECS/Amazon 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 and management tasks
Amazon 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

Application should guide infrastructure



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


API Gateway


Amazon SQS


Amazon SNS


AWS Step Functions


AWS AppSync

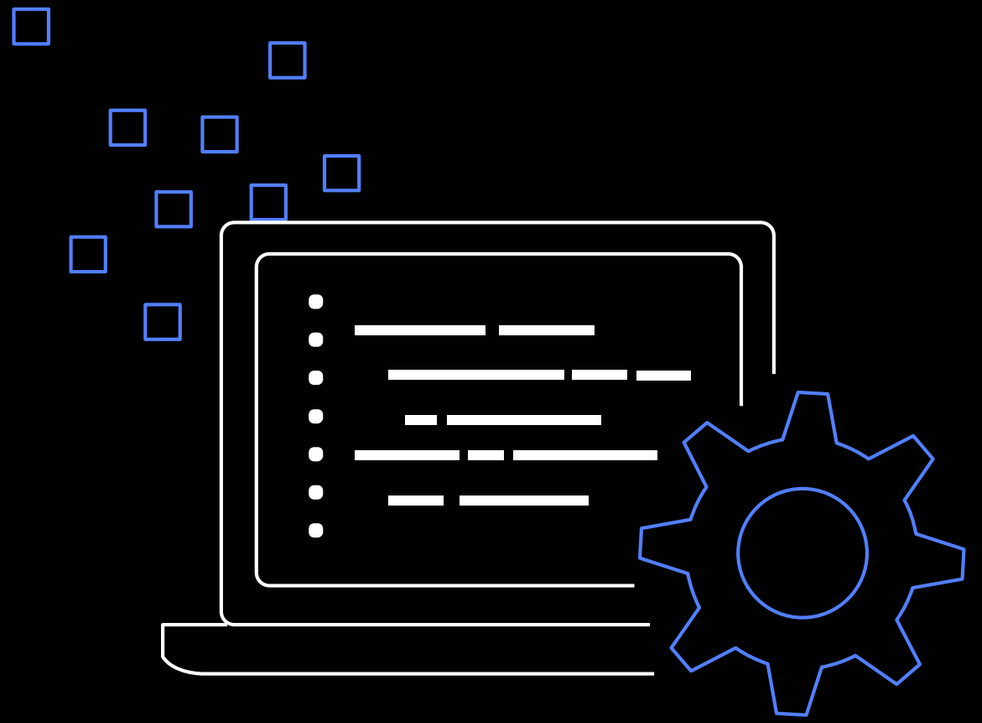

Amazon EventBridge

https://aws.amazon.com/serverless/

aws

Accelerating developer productivity

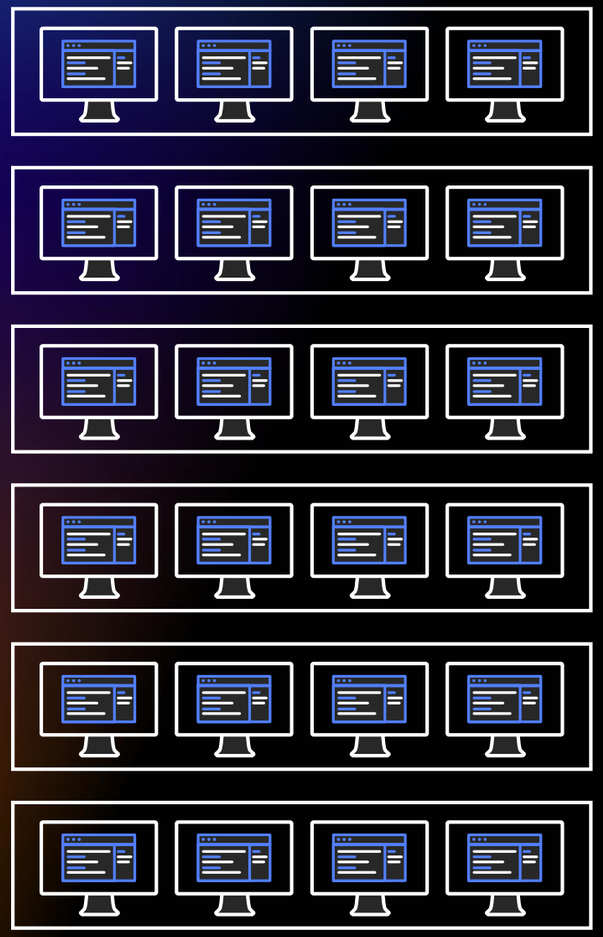
As you move up the layers of simplicity at AWS, your team goes faster



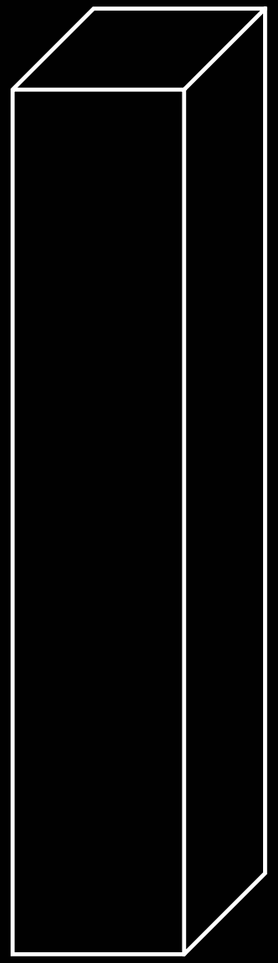
Changes to the delivery of software

Monolith development lifecycle

Developers

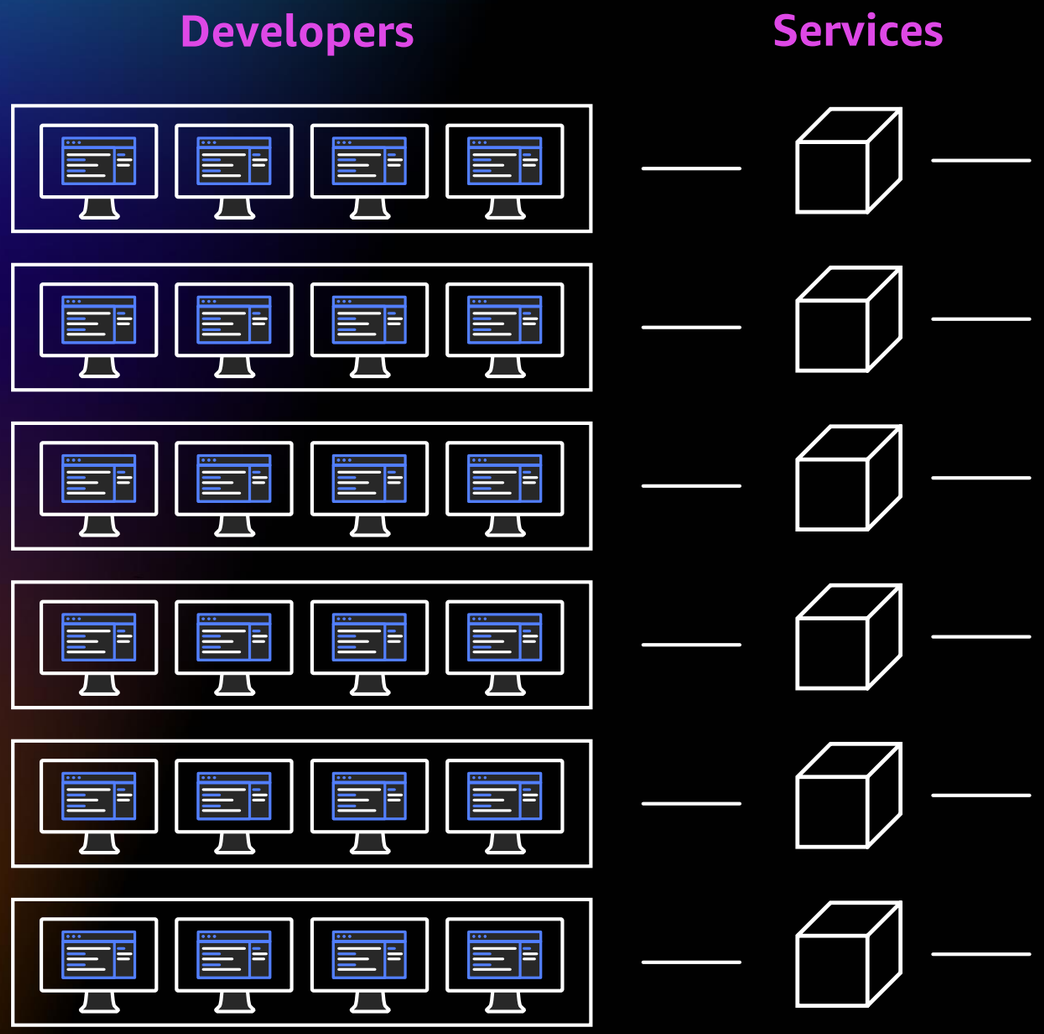


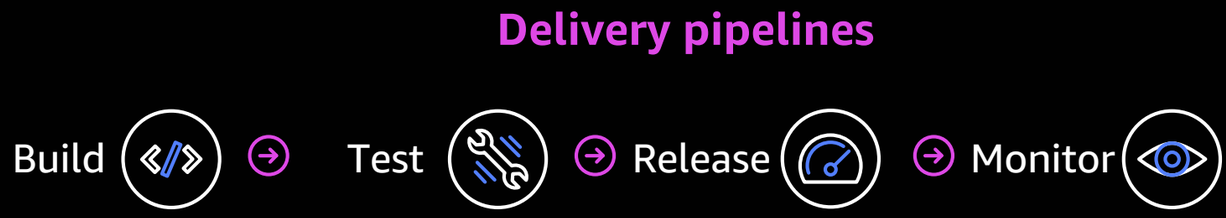
Services



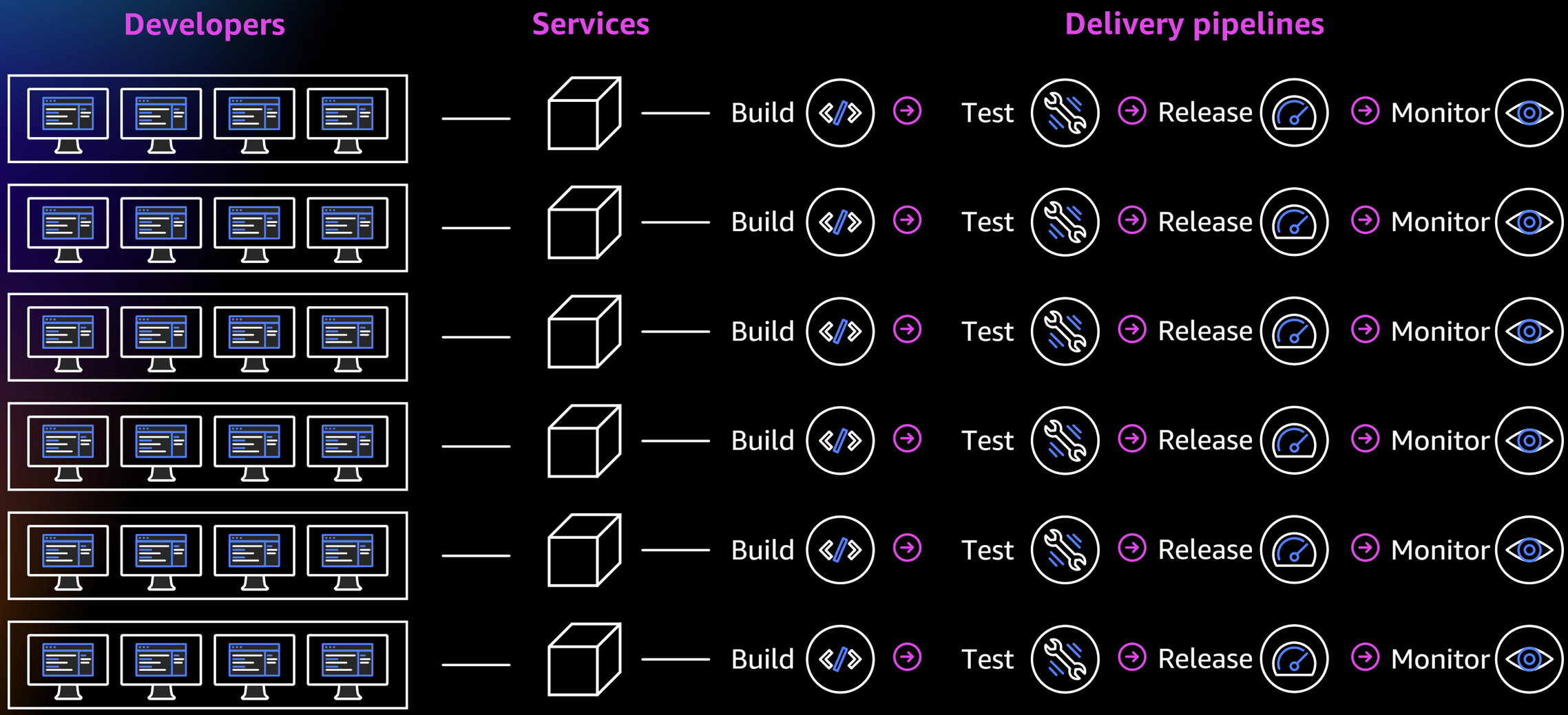
Delivery pipelines

Microservice development lifecycle

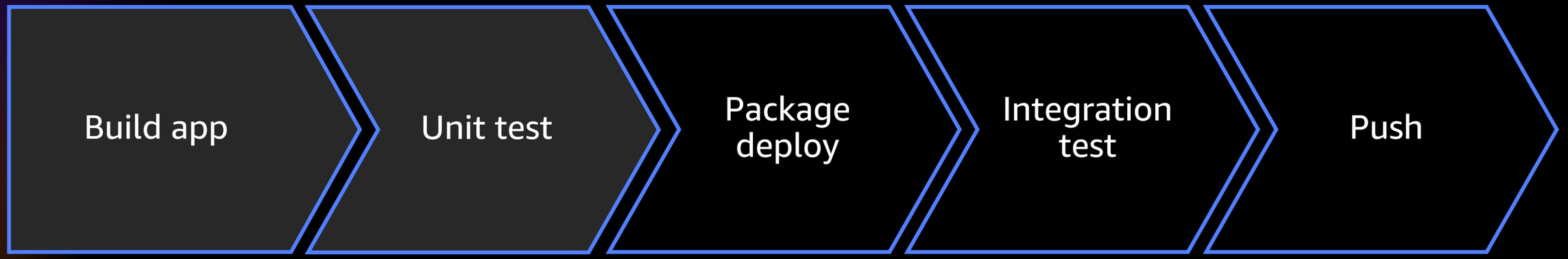




Microservice development lifecycle

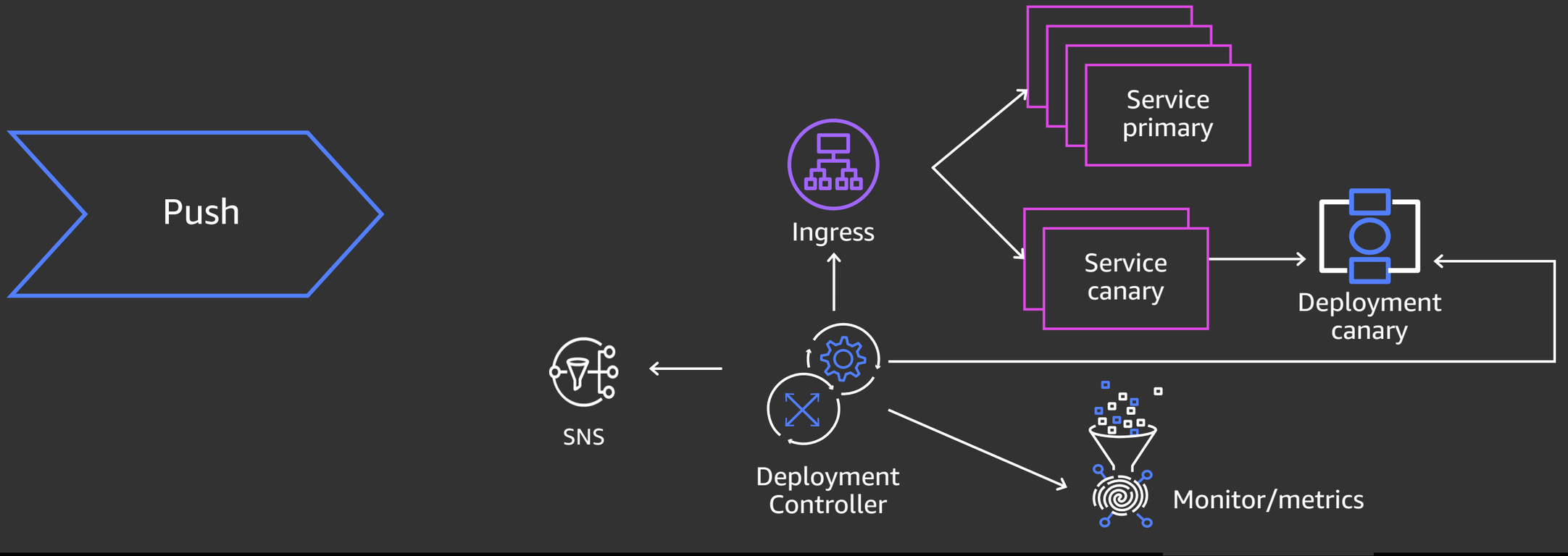


Pipeline per team

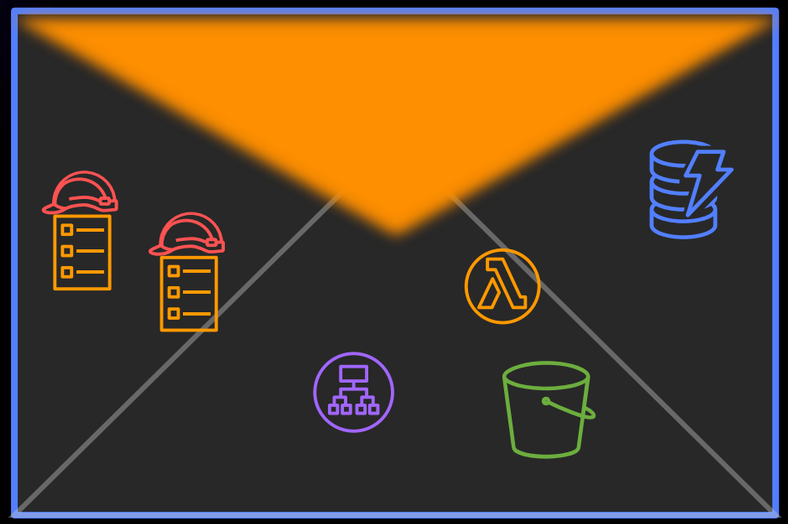


Automated deployment

Build app → Unit test → Package deploy → Integration test



Infrastructure and application



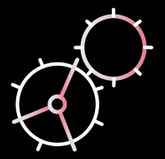
Cloud resources and application

Common code review process

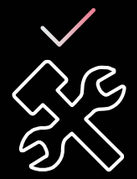
Deployed as a package

Best practices

Decompose for agility
(microservices, 2 pizza teams)

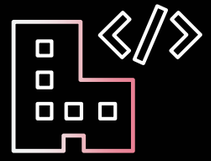


Automate everything



Standardized tools

Belts and suspenders
(governance, templates, DevSecOps)

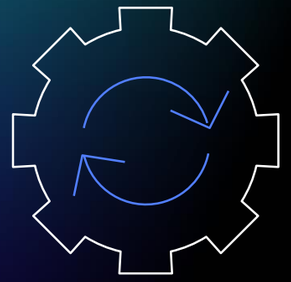


Infrastructure as Code

Five pillars of modern applications

1. Modern applications are application first, not infrastructure first
2. Modern applications are serverless
3. Modern applications automate everything possible
4. Modern applications make security everyone's job
5. Modern applications allow you to extract the most value from your data

Deliver innovation



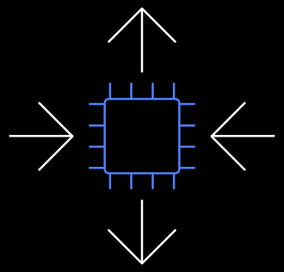
Agility

"We relied on AWS Lambda to get our platform on the market in under four weeks. Within six months, we had scaled to 40,000 users without running a single server."

—A Cloud Guru

"We can have a commit roll into production in literally minutes—as well as provide a bunch of flexible routing options dynamically."

—Pinpoint



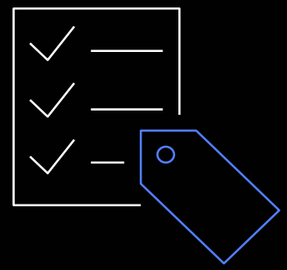
Elasticity

"Using Lambda-based serverless applications, Resnap can run multiple Machine Learning models on an average of 600 photos, which results in thousands of invocations and still generates a photo book within one minute."

—Resnap

"Our serverless-based approaches allow us to serve ads to audiences 60% faster than with instance-based approaches."

—Infinia Mobile



Total cost efficiency

"Our costs dropped by more than 25% and our monthly average time to complete data processing dropped to 7 seconds, making the process over 99% faster."

—Speed Shift Media

"Using AWS Lambda & AWS Step Functions, we cut customer onboarding times from 20 minutes to 30 seconds and their 'expected costs are $20 USD per 10,000 orders.'"

—Mercury

https://aws.amazon.com/lambda/resources/customer-testimonials/

Visit the Modern Applications Resource Hub for more resources

Dive deeper with these resources to help you develop an effective plan for your modernization journey.

- Build modern applications on AWS e-book
- Build mobile and web apps faster e-book
- Modernize today with containers on AWS e-book
- Adopting a modern Dev+Ops model e-book
- Modern apps need modern ops e-book
- Determining the total cost of ownership: Comparing Serverless and Server-based technologies paper
- Continuous learning, continuous modernization e-book
- … and more!

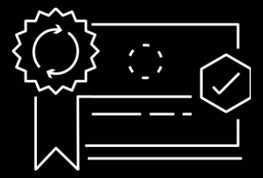
https://bit.ly/3yfOvbK

Visit resource hub »

AWS Training and Certification

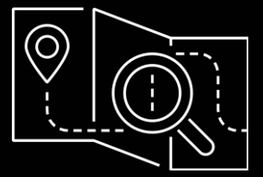
Accelerate modernization with continuous learning

Free digital courses, including:
Architecting serverless solutions
Getting started with DevOps on AWS



Earn an industry-recognized credential:
AWS Certified Developer – Associate
AWS Certified DevOps – Professional

Hands-on classroom training (available virtually) including:
Running containers on Amazon Elastic Kubernetes Service (Amazon EKS)
Advanced developing on AWS



Create a self-paced learning roadmap
AWS ramp-up guide - Developer
AWS ramp-up guide - DevOps

Take Developer and DevOps training today

Learn more about Modernization training for you and your team

Thank you for attending AWS Innovate Modern Applications Edition

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-apj-marketing@amazon.com


twitter.com/AWSCloud


facebook.com/AmazonWebServices

youtube.com/user/AmazonWebServices


slideshare.net/AmazonWebServices


twitch.tv/aws

Thank you!

Anshul Sharma

Sr. Solutions Architect
Amazon Web Services

 @anshuldsharma