aws SUMMIT

INDIA | MAY 25, 2023

GSAWS004

Breaking monolith into microservices, while deploying with AWS

Jyotisankar Behera
Technical Account Manager
AWS India

Rama Krishna Sanjeeva
Enterprise Solutions Architect
AWS India

Agenda

- Why break a monolith to microservice
- How to break a monolith to microservice
- Why to deploy a microservice in AWS
- How to deploy a microservice in AWS

Why break a monolith to microservice

Why to break a microservice

- My monolithic app works fine. Why should I refactor to Microservices?
- Is the refactoring worth the pain and effort?

Challenges in a monolith architecture

Challenges

01 Big team working on the same workspace and contains a wide range of features and functions. Makes it difficult to trace and fix defects.

02 Gets bigger and bigger with every release which makes the software development iteration cycle longer.

03 Small change or bug fix in any module can take down the entire application

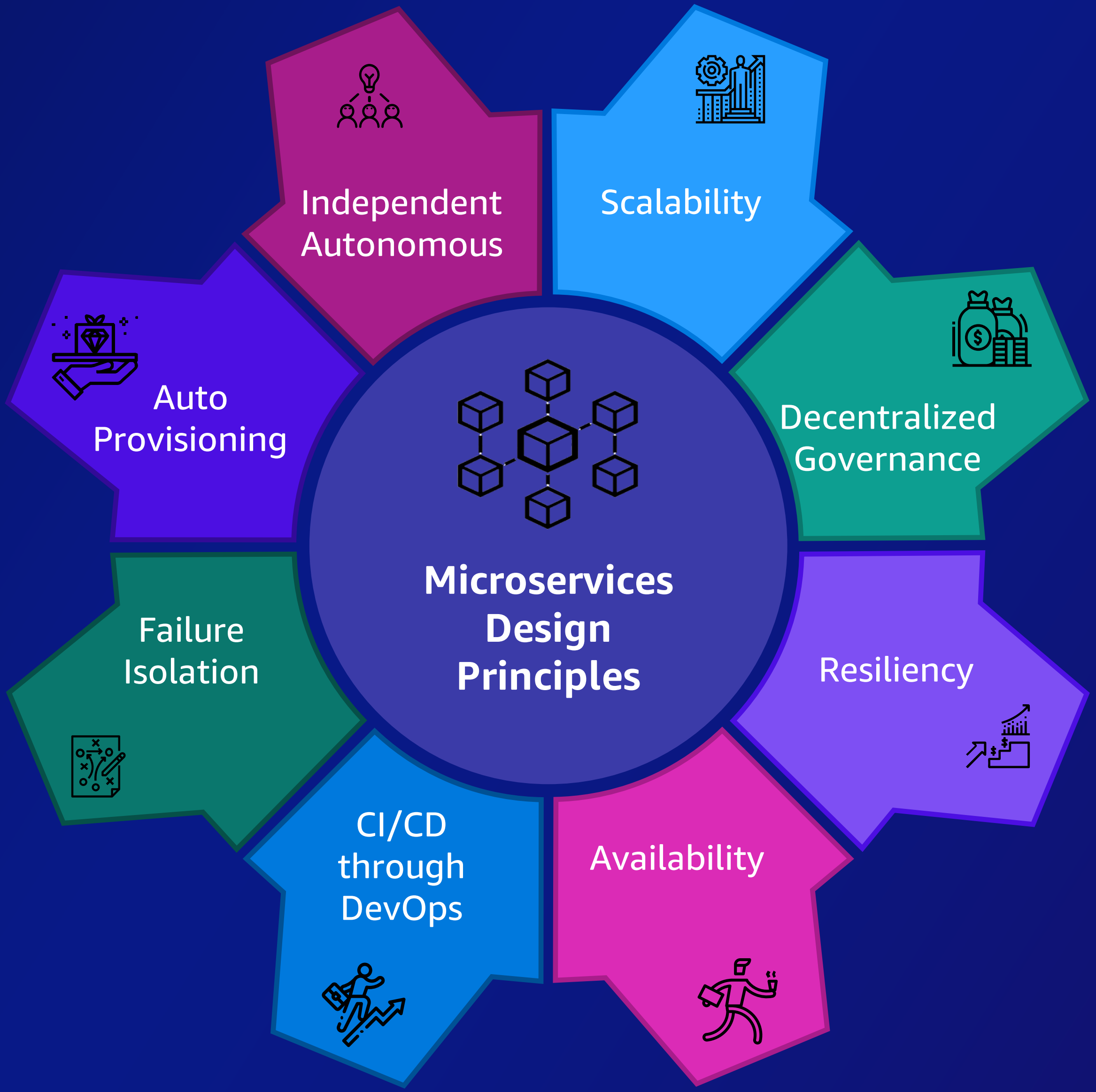
04 Scaling is challenging and has a limit as everything is tightly coupled and all resources are shared.

05 Makes it difficult to implement continuous integration/deployment.

06 Updating to new technology and new process is almost impossible

How to break a monolith to microservice

Microservices architecture attributes



Decomposition patterns

Decompose monolith according to

Business Capabilities

Business capability is something that a business does in order to generate value.

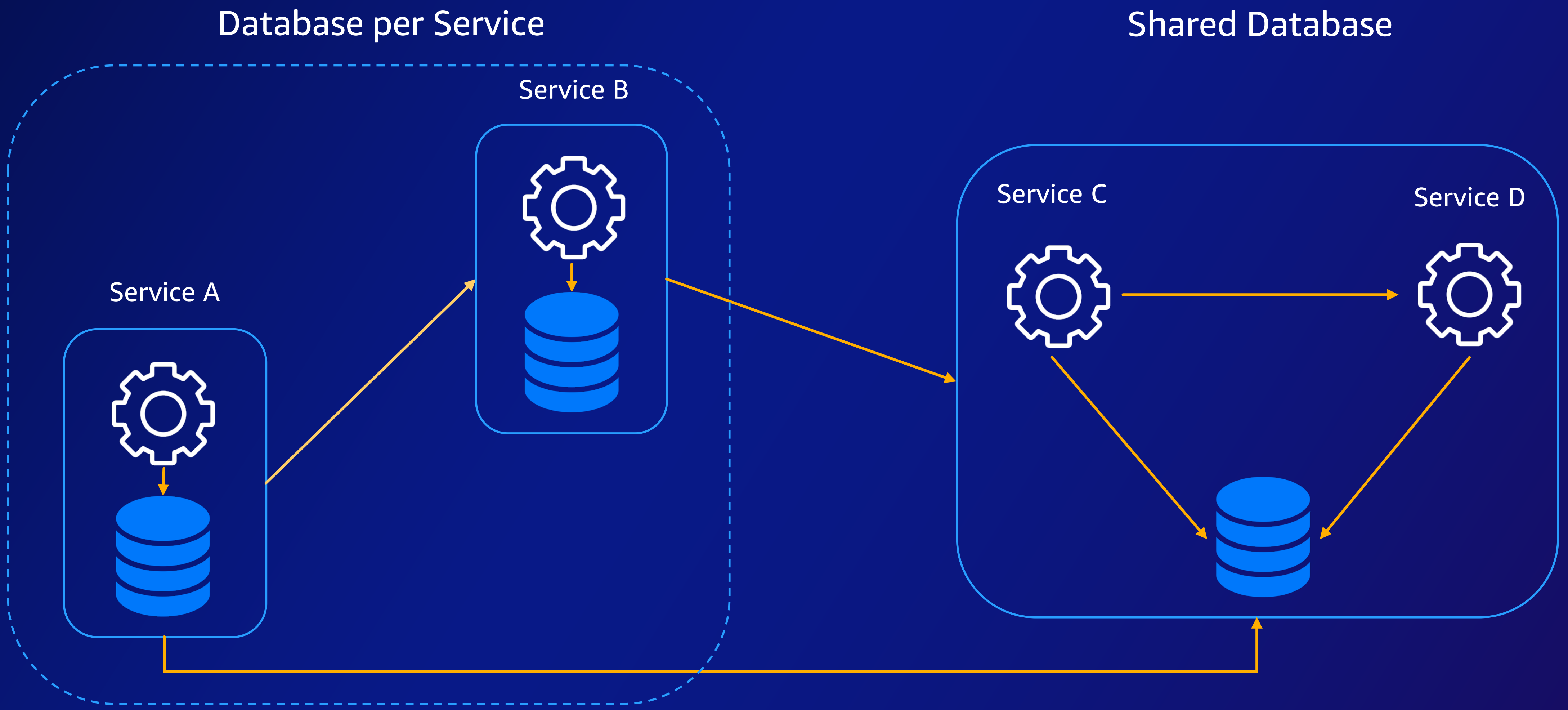
Sub-Domain of application

Domain Driven Design(DDD) uses sub-domain and bounded context concepts to segregate the microservices.

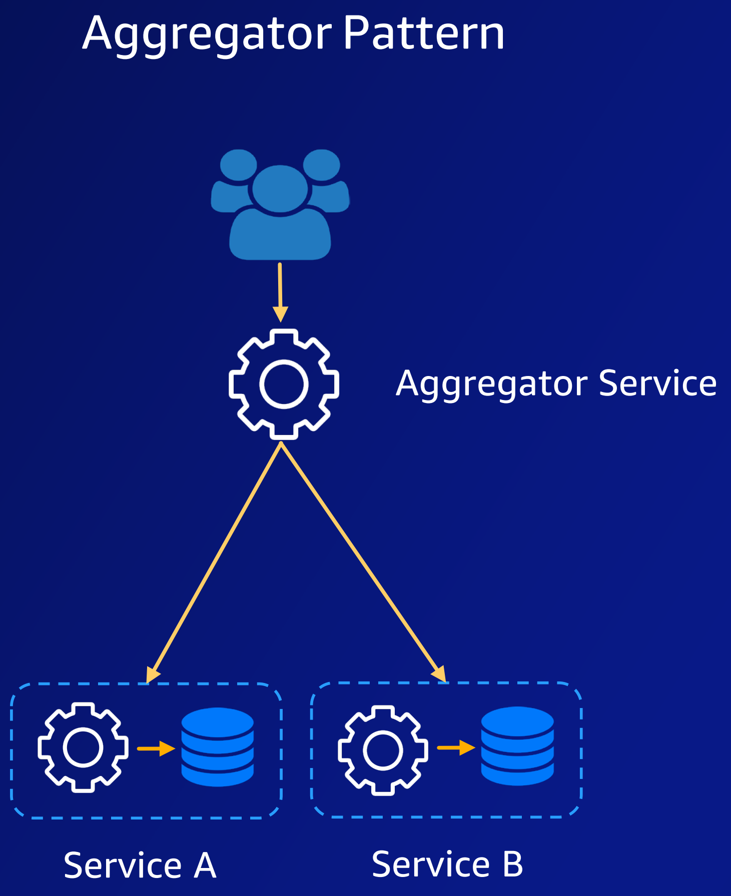
Strangler or Vine Pattern

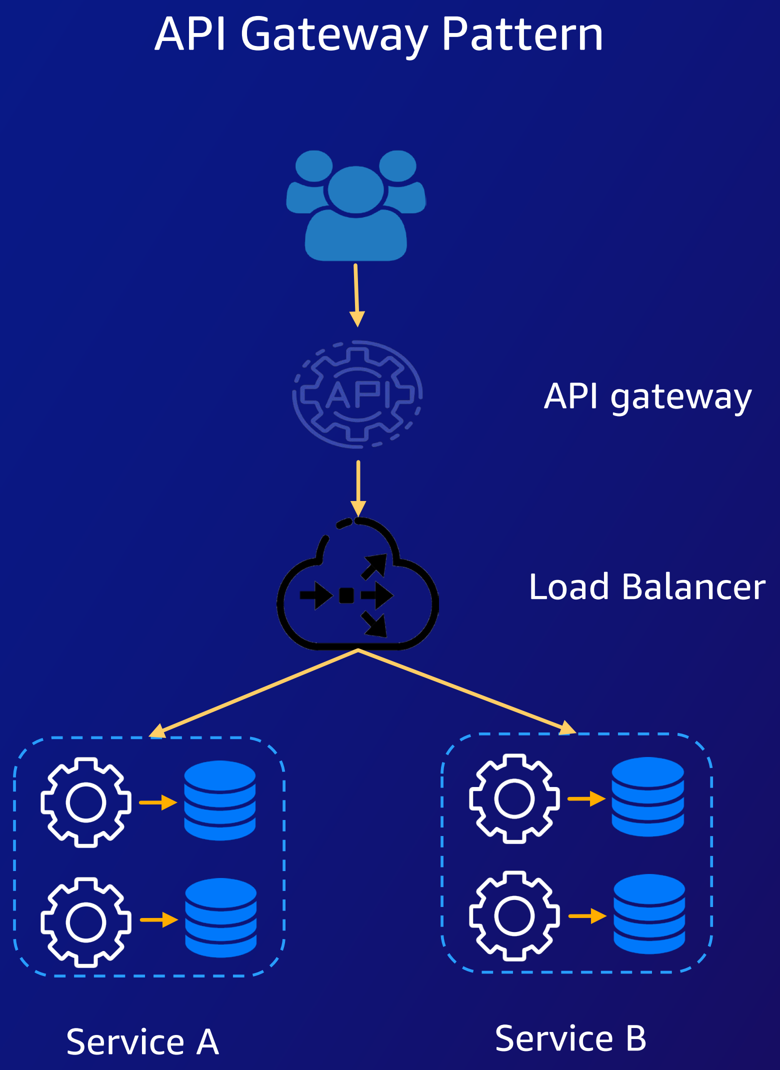
Create a new system around the edges of the old one and letting it grow slowly until the old system is strangled.

Database patterns

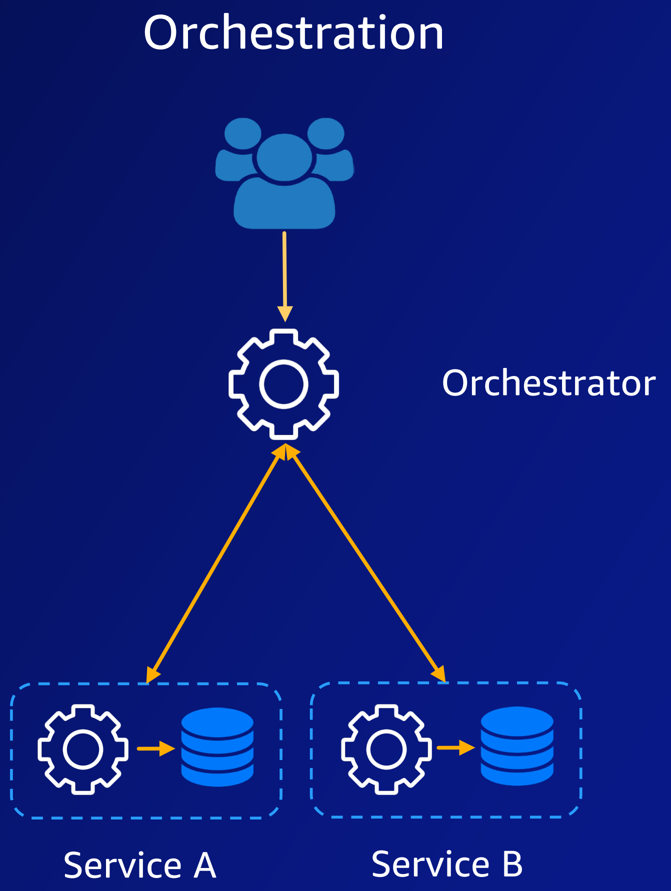


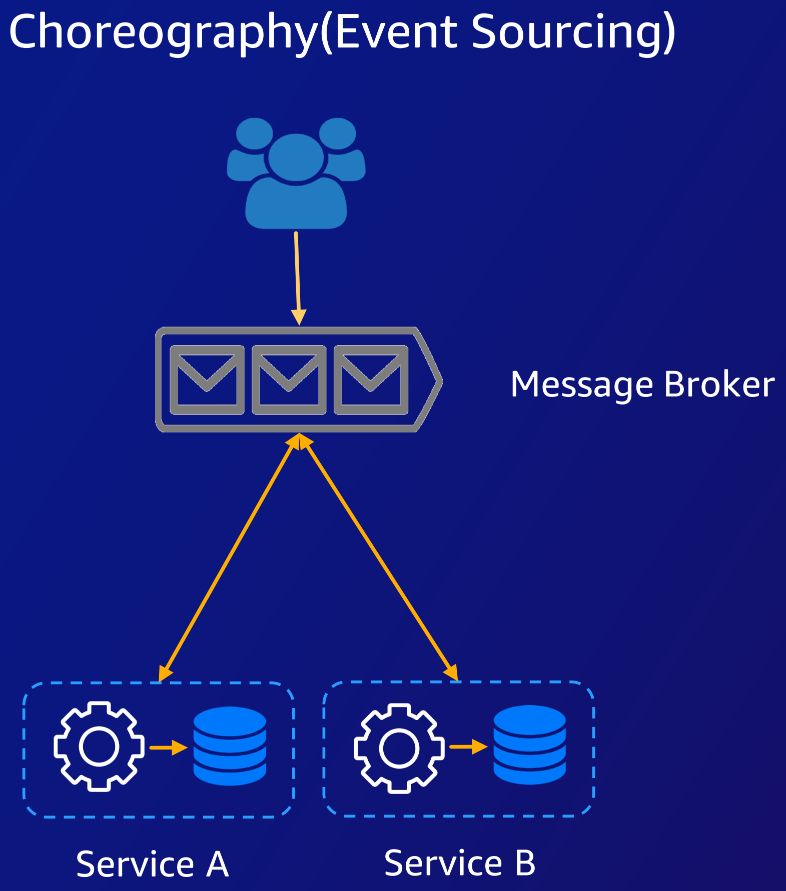
Integration pattern





Saga design pattern





Observability patterns

1. **Log Aggregation** — Centralized logging service that aggregates logs from all the microservices at one place. e.g : AWS cloudwatch

2. **Performance Metrics** — Metrics services which gathers statistics about individual operations and provides reporting and alerting. e.g : Prometheus

3. **Distributed Tracing** — Traces the requests which spans multiple services to track if any errors. e.g : AWS X-Ray

4. **Health Checks** — Each service needs an endpoint to check the health of the application and alerts when the backend logic is not working or connection to other service is down. e.g : Liveliness Probe

Cross-cutting concern patterns

1. **External Configuration** — Externalize all the configurations for all the environments including endpoint URLs and credentials for the services.

2. **Service Discovery** — Service registry is required which keeps the metadata of each service along with health checks and routes the requests to only healthy instances.

3. **Circuit Breaker** — Monitors the consecutive failures from a service and stops all transaction when repeated failures. Allows traffic after a certain time when test transactions succeed.

4. **Deployment Patterns** — Several deployment patterns are followed like Rolling deployment, Blue-Green deployment, Canary deployment.The aim is to minimize downtime while making releases.

Why to deploy a microservice in AWS

Why to deploy a microservices in AWS

- Broadest set of container and serverless compute offerings in the market

- Compute services offer deep integrations with the rest of over 200+ fully featured service offerings in AWS

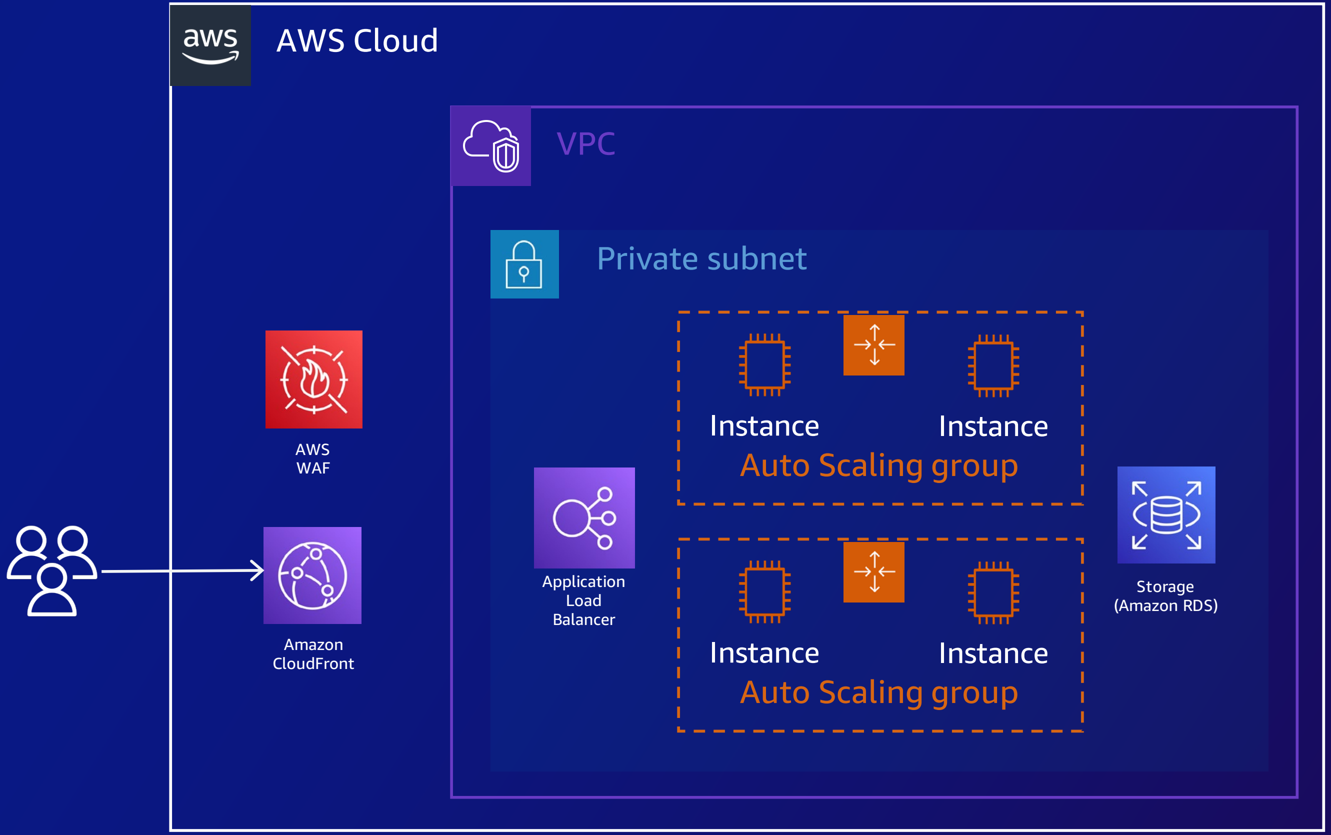
- Supports and adopts open source software and is a substantial contributor back to the ecosystem

- Accelerating rate of innovation and in releasing services and features

How to deploy a microservice in AWS

Modernized workload using Amazon EC2

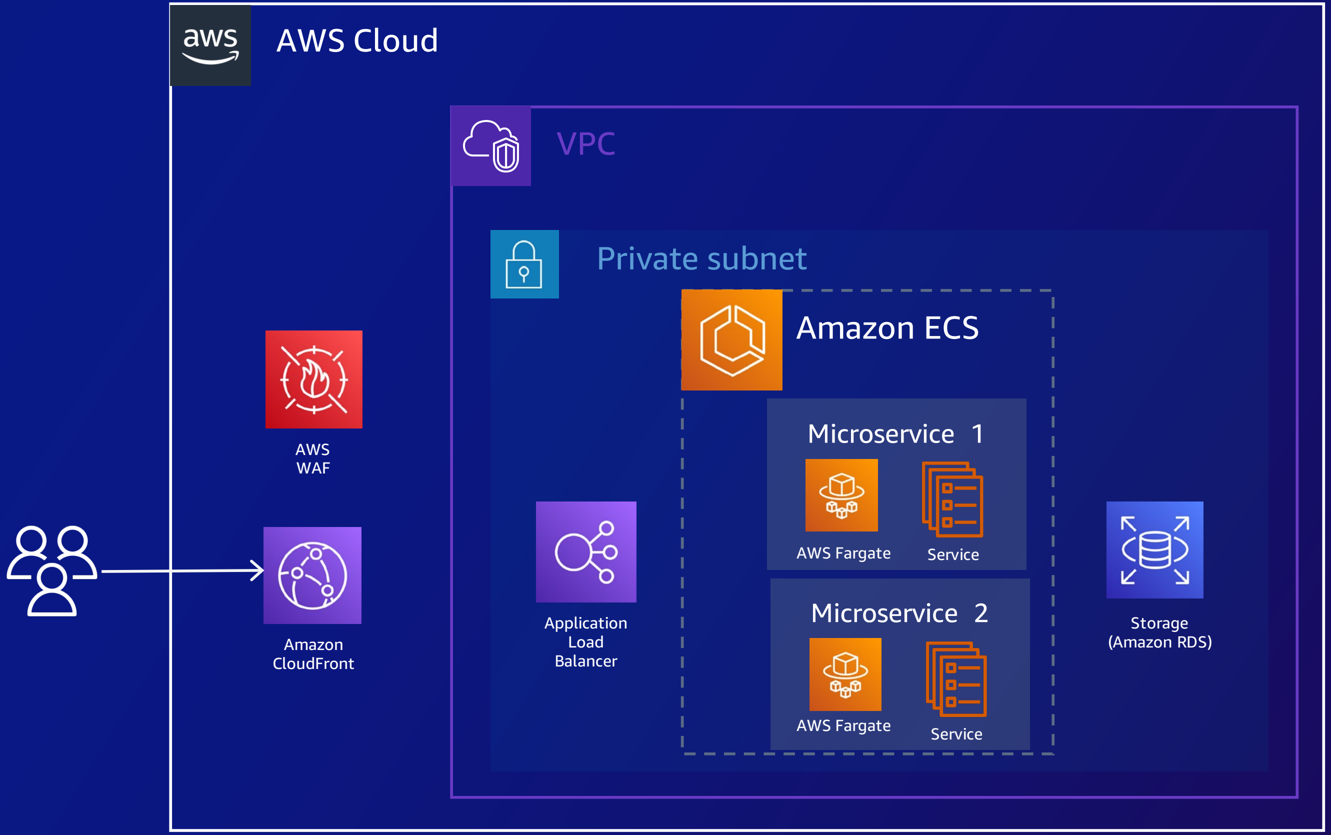
- Decompose application into independent deployable micro-services
- Leverage Amazon EC2 Auto Scaling for scaling
- Challenges
  - Deployment
  - Granular scaling



Modernized workload using Amazon ECS

AMAZON ECS(ELASTIC CONTAINER SERVICE)

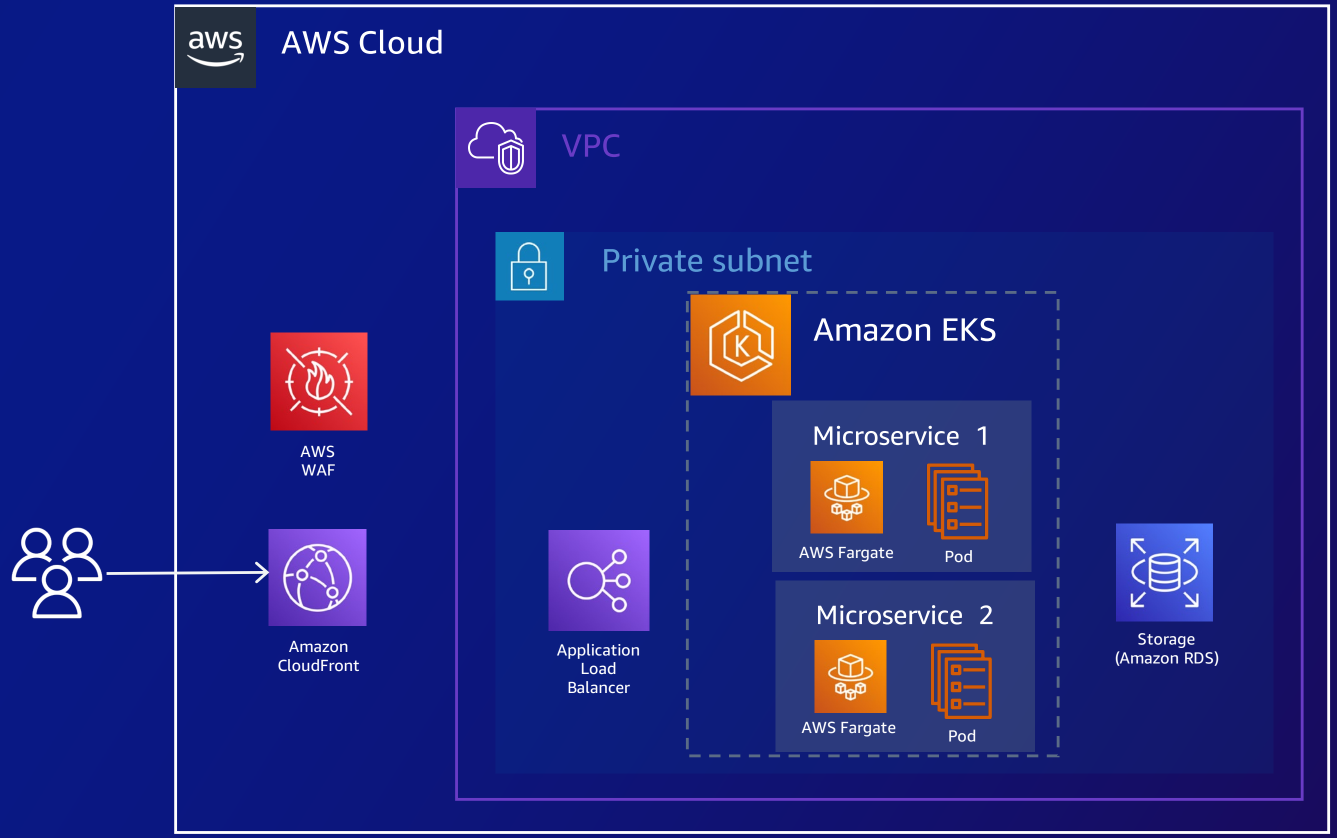
- AWS-opinionated way to run containers at scale
- Fully managed by AWS
- AWS App2Container to containerize with minimal efforts
- Reduced operational burden



Modernized workload using Amazon EKS

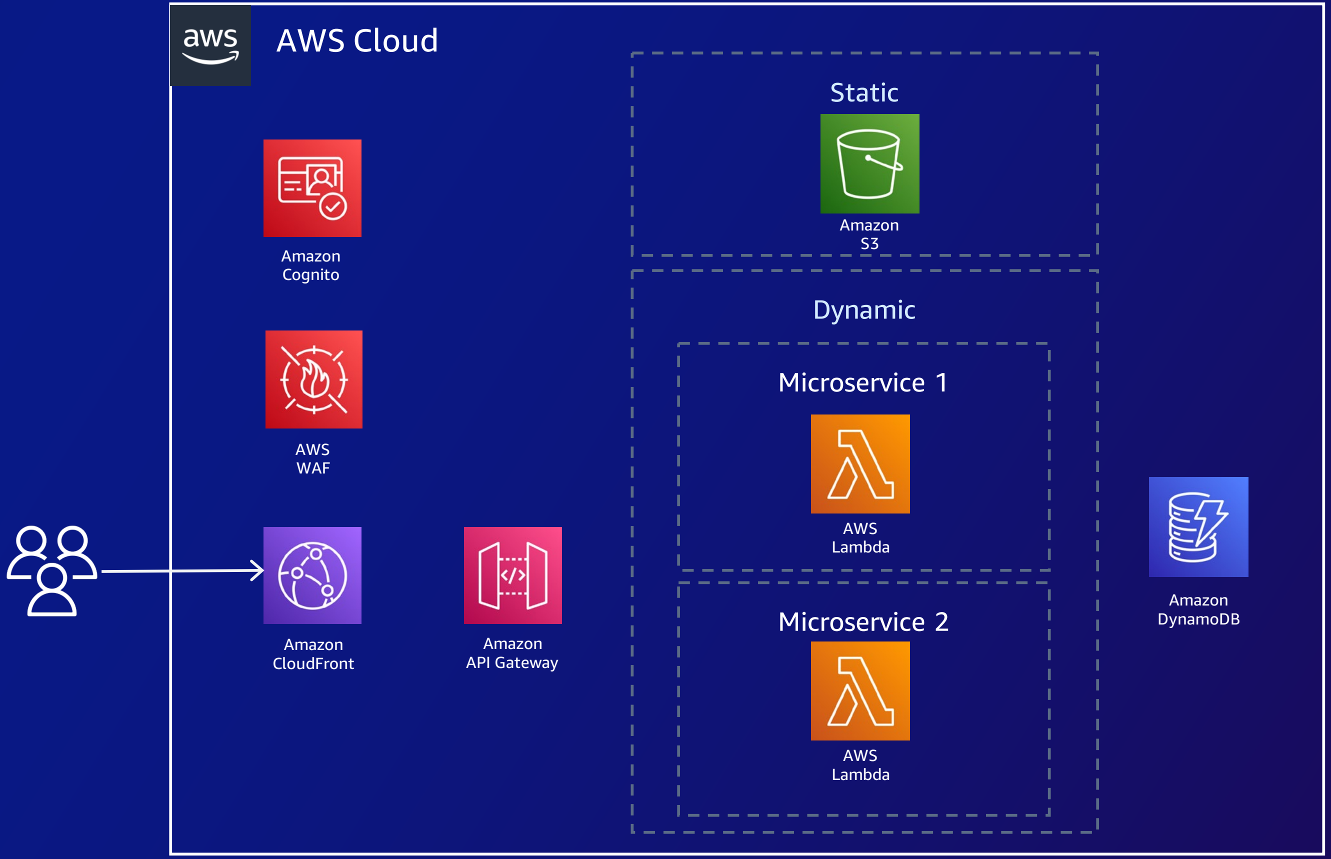
AMAZON EKS(ELASTIC KUBERNETES SERVICE)

- Managed Kubernetes service
- EKS runs upstream Kubernetes and is certified Kubernetes conformant
- Built-in integrations with AWS services

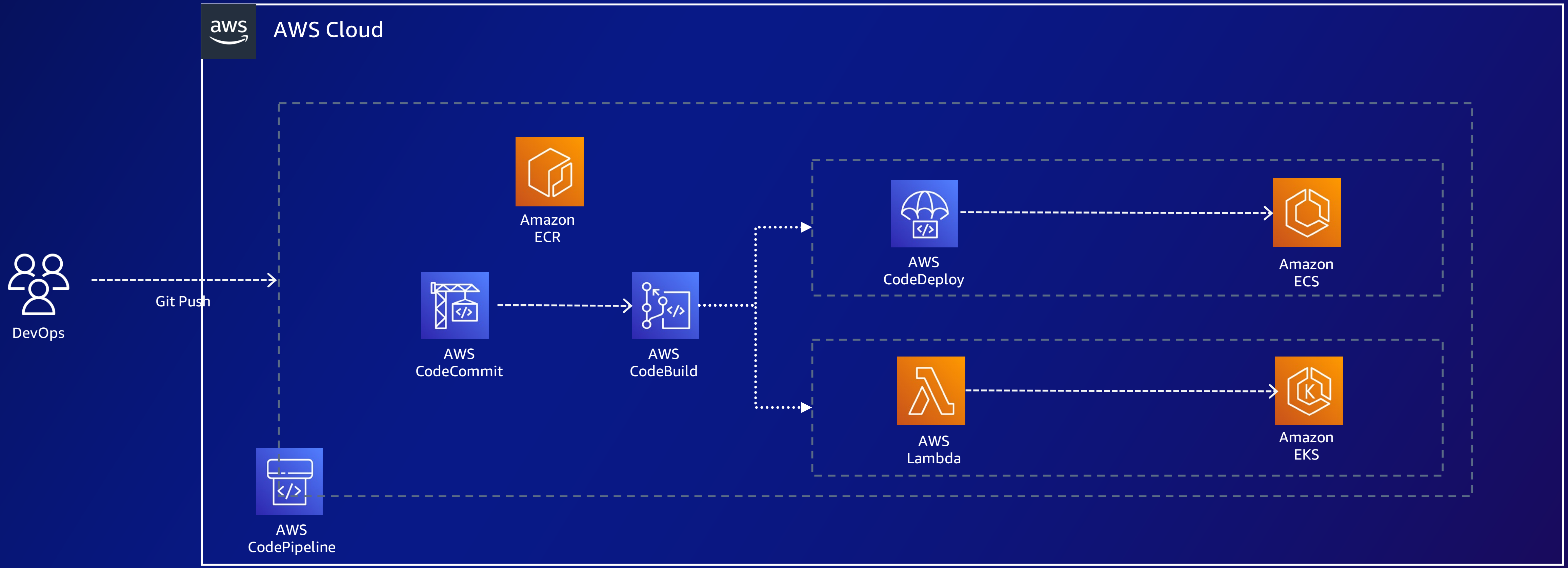


Modernized workload using AWS Lambda

- No server management
- Flexible, automated scaling
- Automated high availability
- Increased agility and optimized cost model

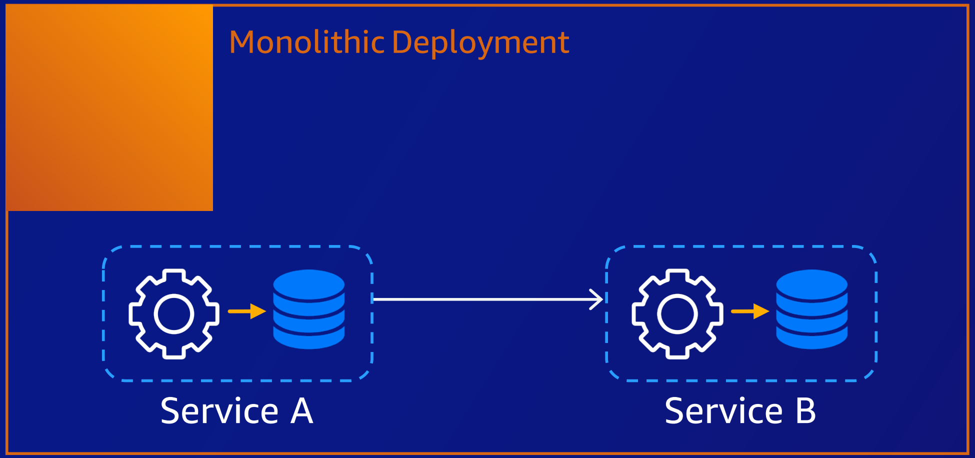


Container continuous integration and continuous delivery on Amazon ECS and Amazon EKS - Architecture



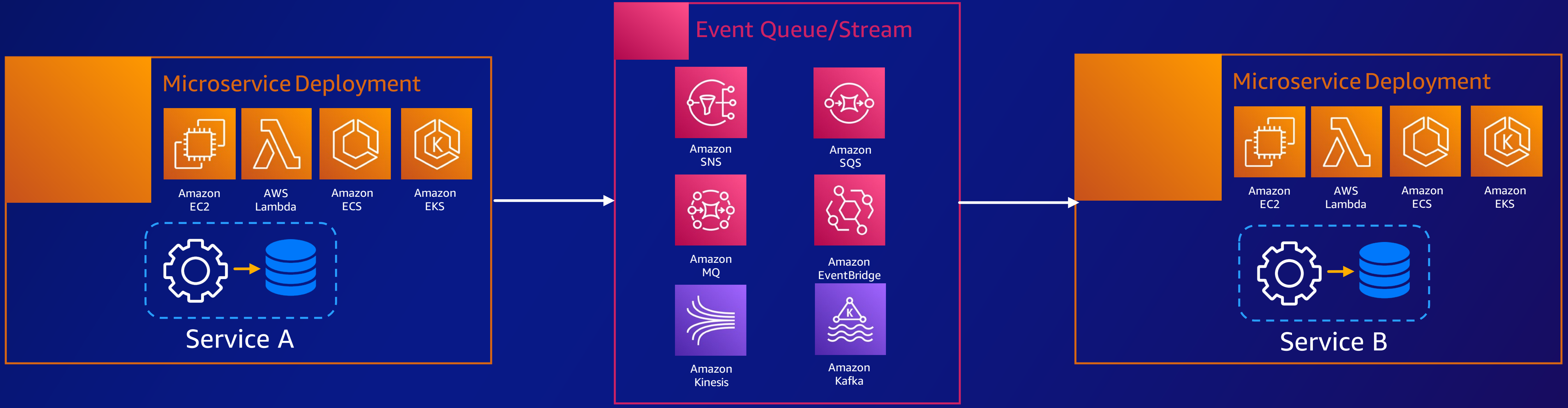
Service integration patterns - Synchronous

Monolithic Deployment	Scaling, availability, security complexity
Tight Coupling	Cascading failure
Single Technology Stack	Challenge in maintainability and team autonomy



Service integration patterns - Asynchronous

Event Driven Architecture	Deployment as microservice
Loose Coupling	Sagas for data consistency
Independent Scaling	Message/Event Store



Key Takeaways

- Agility
- Flexible scaling
- Easy deployment
- Technological freedom
- Reusable code
- Resilience

Additional resources

Getting Started with Microservices on AWS

https://aws.amazon.com/microservices/

Whitepaper - Implementing Microservices on AWS

https://docs.aws.amazon.com/whitepapers/latest/microservices-on-aws/microservices-on-aws.html skillbuilder.aws

Your time is now

Build in-demand cloud skills your way

Thank you!

Please complete the session survey

Jyotisankar Behera
Technical Account Manager
AWS India

Rama Krishna Sanjeeva
Enterprise Solutions Architect
AWS India