aws INNOVATE

MIGRATE. MODERNIZE. BUILD.

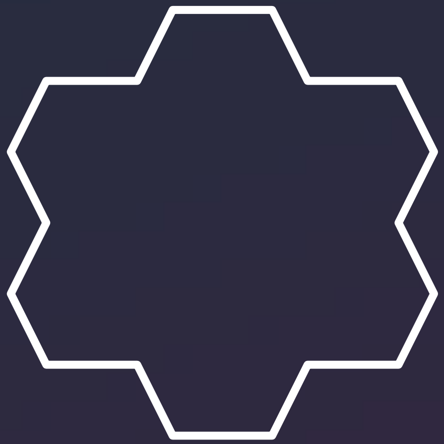
26 SEP,2024

AWS INNOVATE
MIGRATE. MODERNIZE. BUILD.

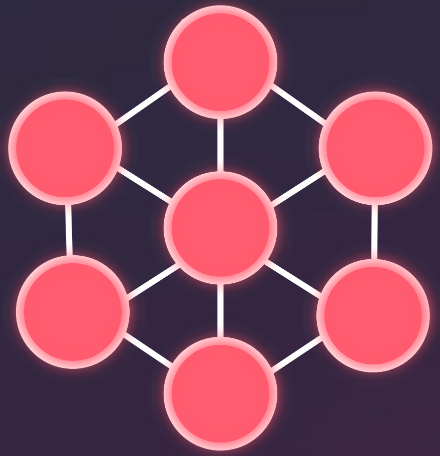
Integration patterns for building distributed applications

Jan Tan
Principal Solutions Architect
AWS

Moving from monoliths to microservices



Monolith
Does everything



Microservices
Do one thing

Reducing complexity

Operational model

Serverless

Integration

Integration patterns

Automation

DevOps

Visibility

Observability and monitoring

Compute operational models

Least ↑ What you manage ↓ Most

	AWS manages	Customer manages	
AWS Lambda Serverless functions	Data source integrations Physical hardware, software, networking, and facilities Provisioning	Application code	
Amazon Elastic Container Service (Amazon ECS) on AWS Fargate Serverless containers	Container orchestration, provisioning Cluster scaling Physical hardware, host operating system/kernel, networking, and facilities	Application code Data source integrations Security configuration and updates Network configuration	Management tasks
Amazon ECS on Amazon Elastic Compute Cloud (Amazon EC2) Container-management -as-a-service	Container orchestration control plane Physical hardware, software, networking, and facilities	Application code Data source integrations Work clusters	Security configuration and updates, network configuration, firewall, management tasks
Amazon EC2 Infrastructure-as-a-service	Physical hardware, software, networking, and facilities	Application code Data source integrations Scaling Security configuration and updates Network configuration Management tasks	Provisioning, managing scaling and patching of servers

Serverless is much more than compute

Compute

AWS Lambda

AWS Fargate

Data stores

Amazon Simple Storage Service (Amazon S3)

Amazon Aurora Serverless

Amazon DynamoDB

Integration

Amazon EventBridge

Amazon API Gateway

Amazon Simple Queue Service (Amazon SQS)

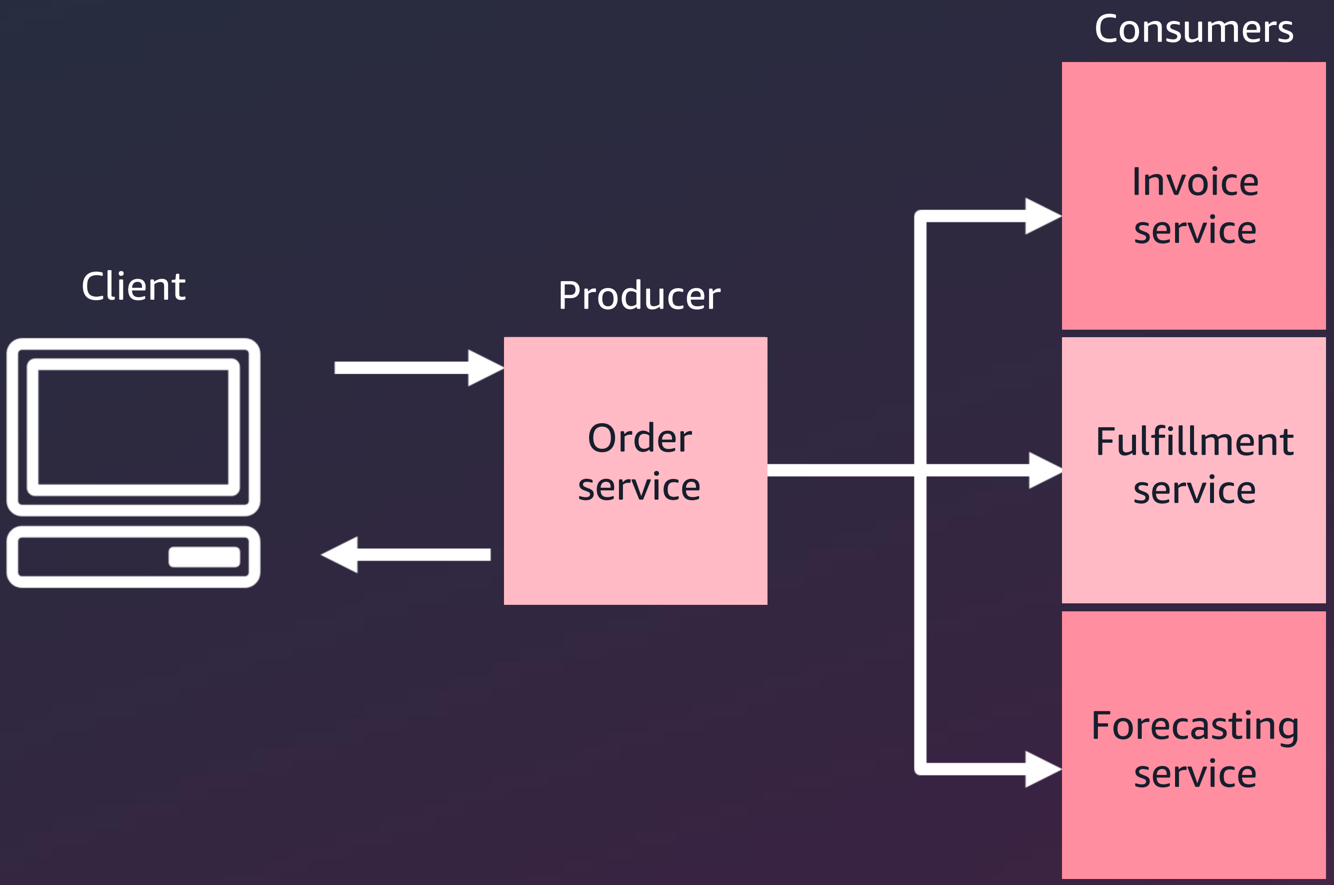
Amazon Simple Notification Service (Amazon SNS)

AWS Step Functions

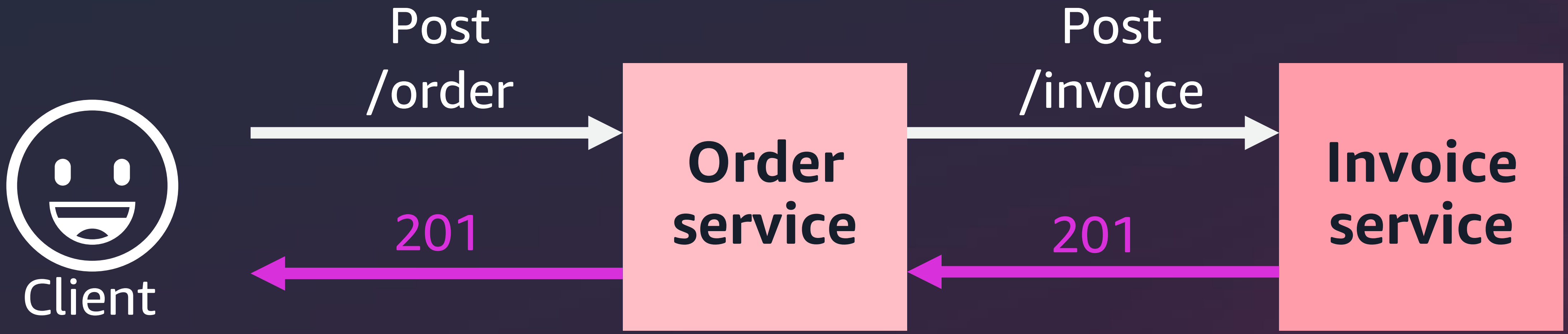
AWS AppSync

Integration patterns

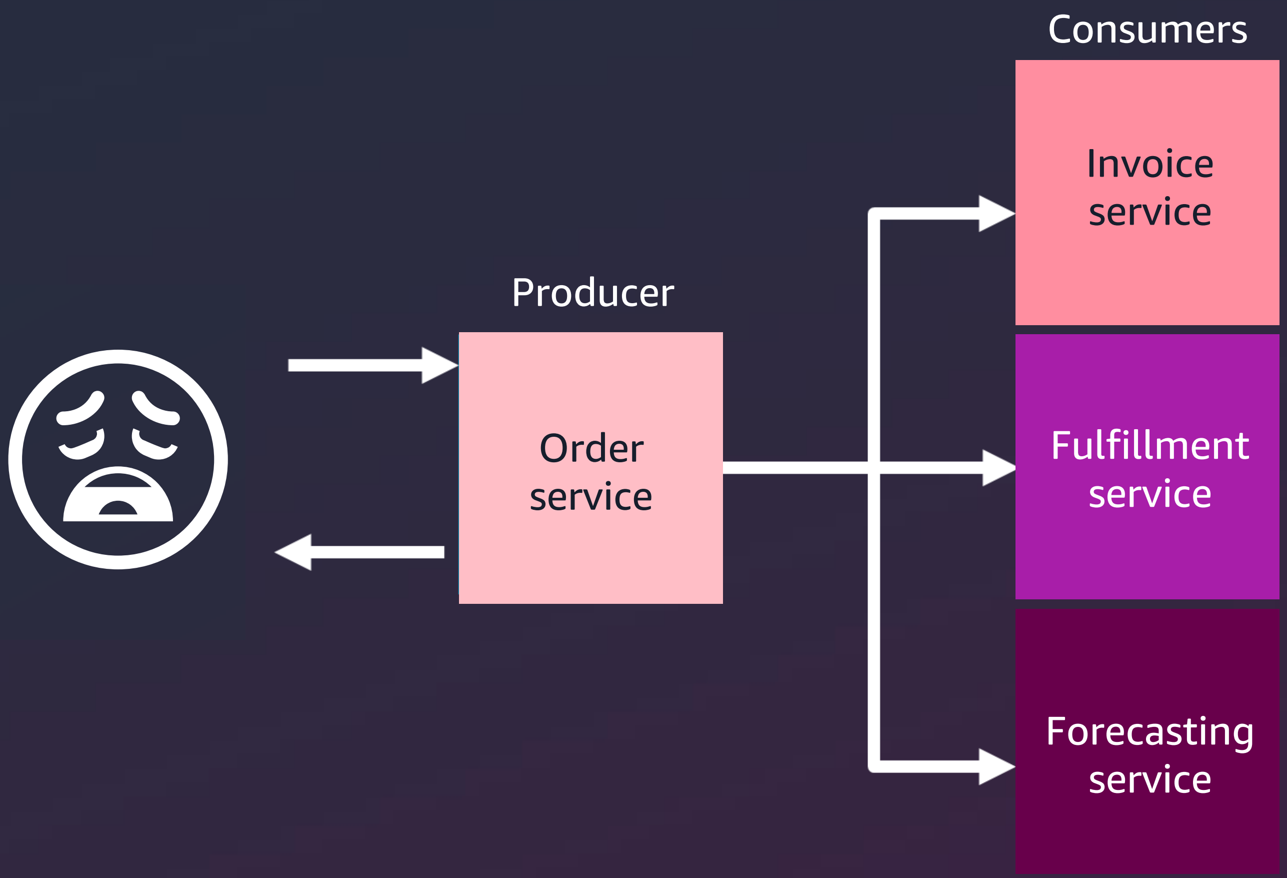
Use case: eCommerce order / fulfillment flow



Microservices start simple



Synchronous API challenges over time



Coupling between producers and consumers

Multiple points of failure

Varying degrees of quality of service

External dependencies

Think asynchronous choreography/orchestration

Orchestration and choreography

Orchestration	Choreography
• One system **controls** the flow between components • Easier to do end-to-end monitoring, timeout, etc. • Centralized business logic	• Pass messages between **bounded contexts** of services • The flow is an **emergent property** of events being sent • Easier to extend, modify, and build upon the messages being passed

Source: https://medium.com/theburningmonk-com/choreography-vs-orchestration-in-the-land-of-serverless-8aaf26690889

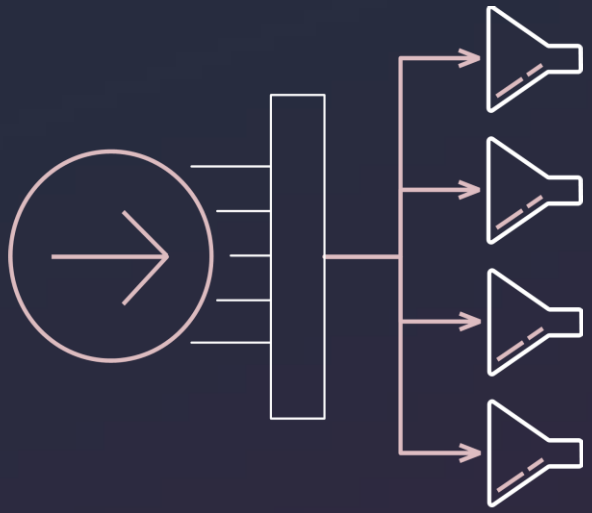
Choreography

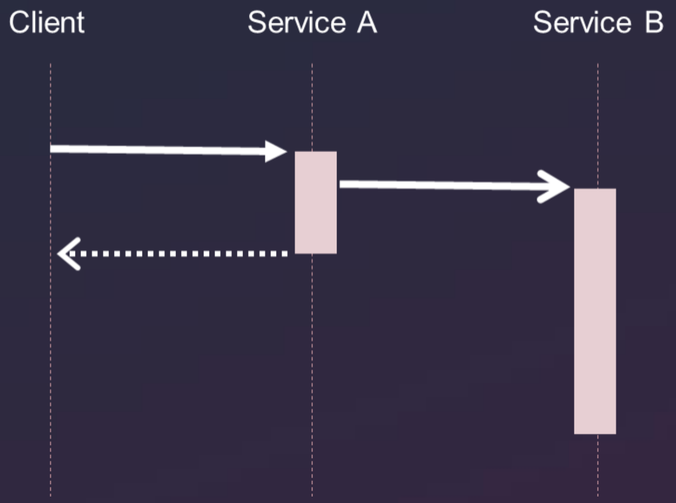
event

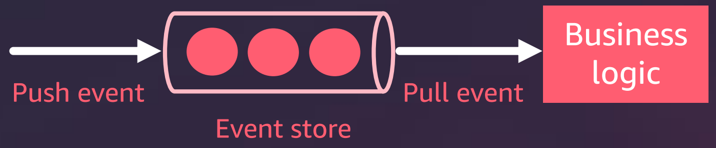
[i-’vent] **noun**

A signal that a system’s state has changed

Event-driven architecture (EDA) components







Event routers

Abstract producers and consumers from each other

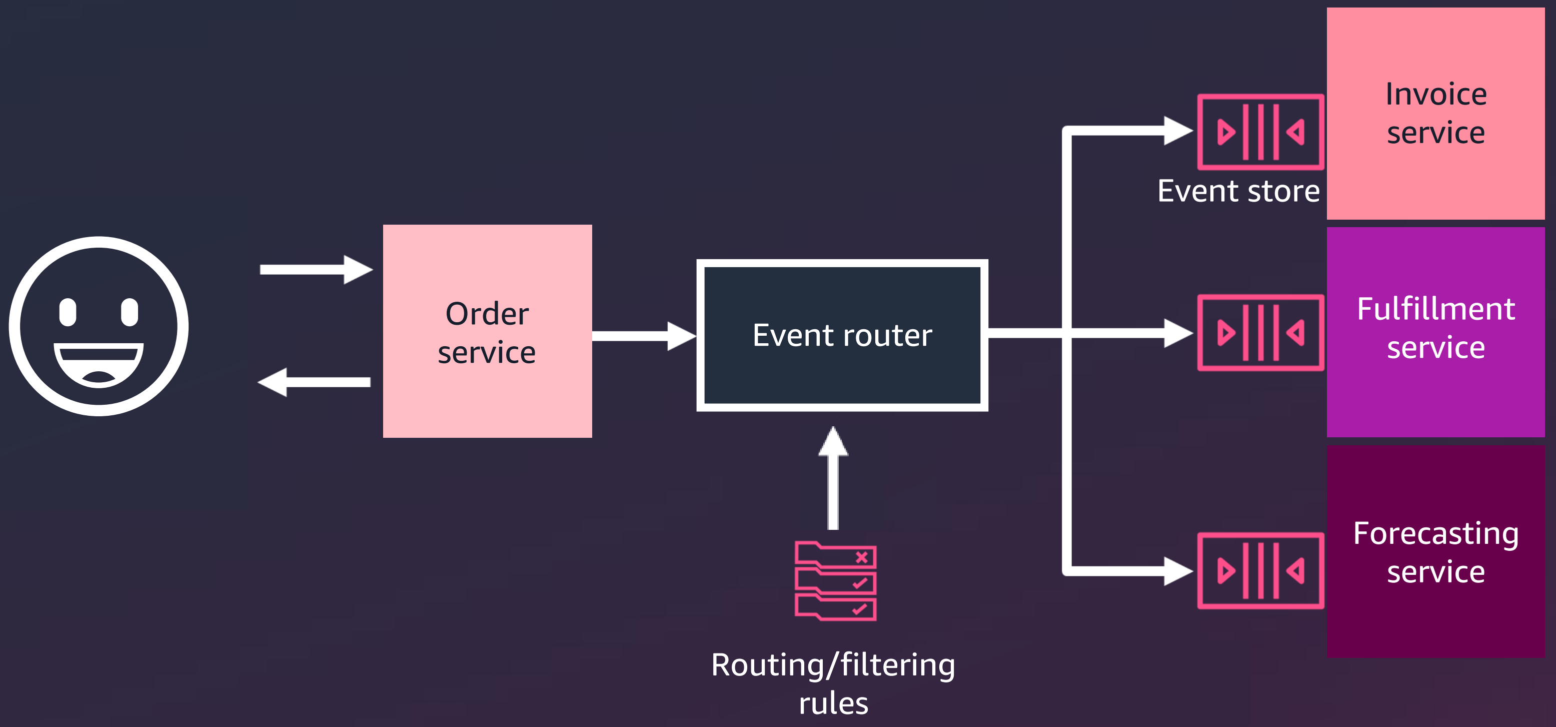
Asynchronous events

Improve responsiveness and reduce dependencies

Event stores

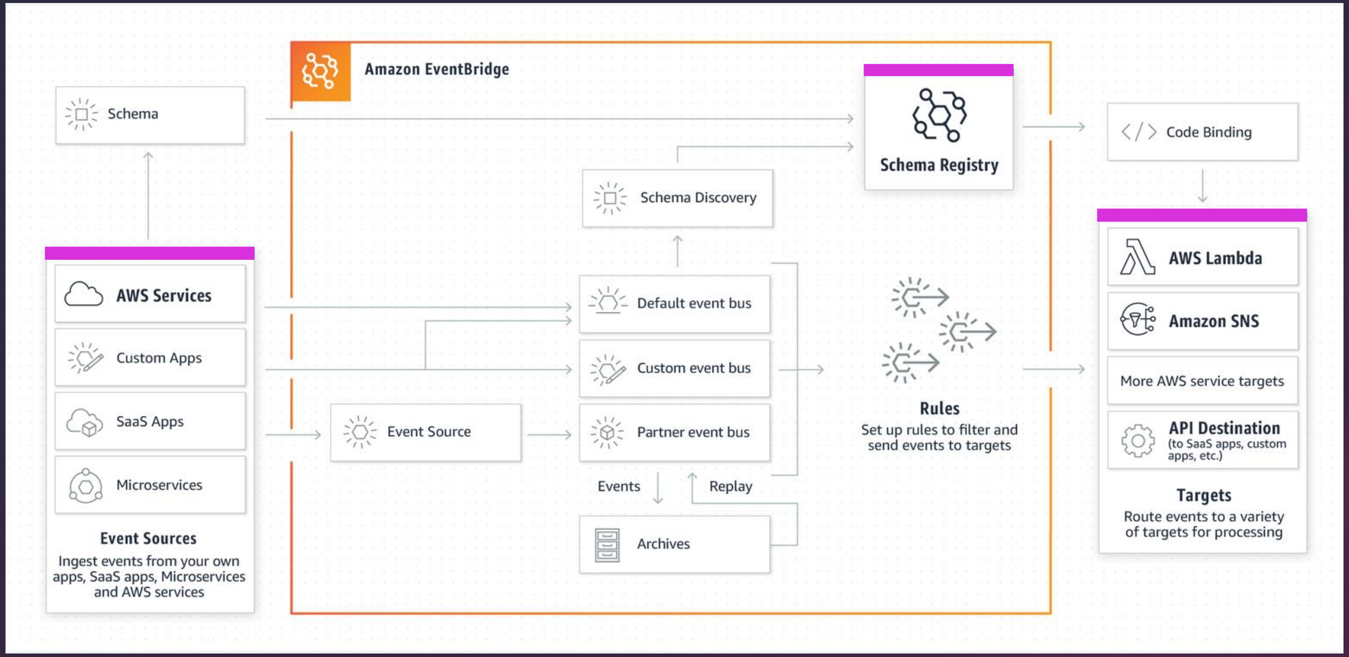
Buffer messages until services are available to process

Reliable, resilient, and independently scalable

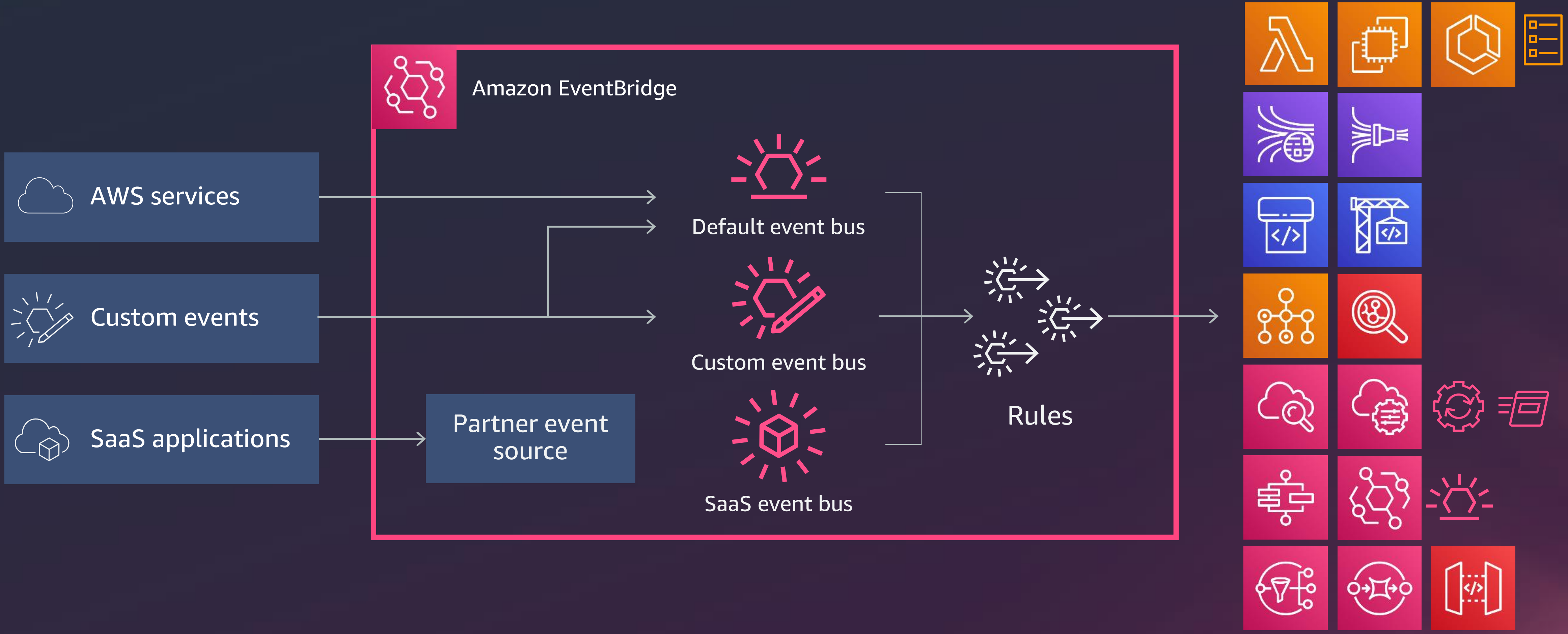


Amazon EventBridge - as an event router

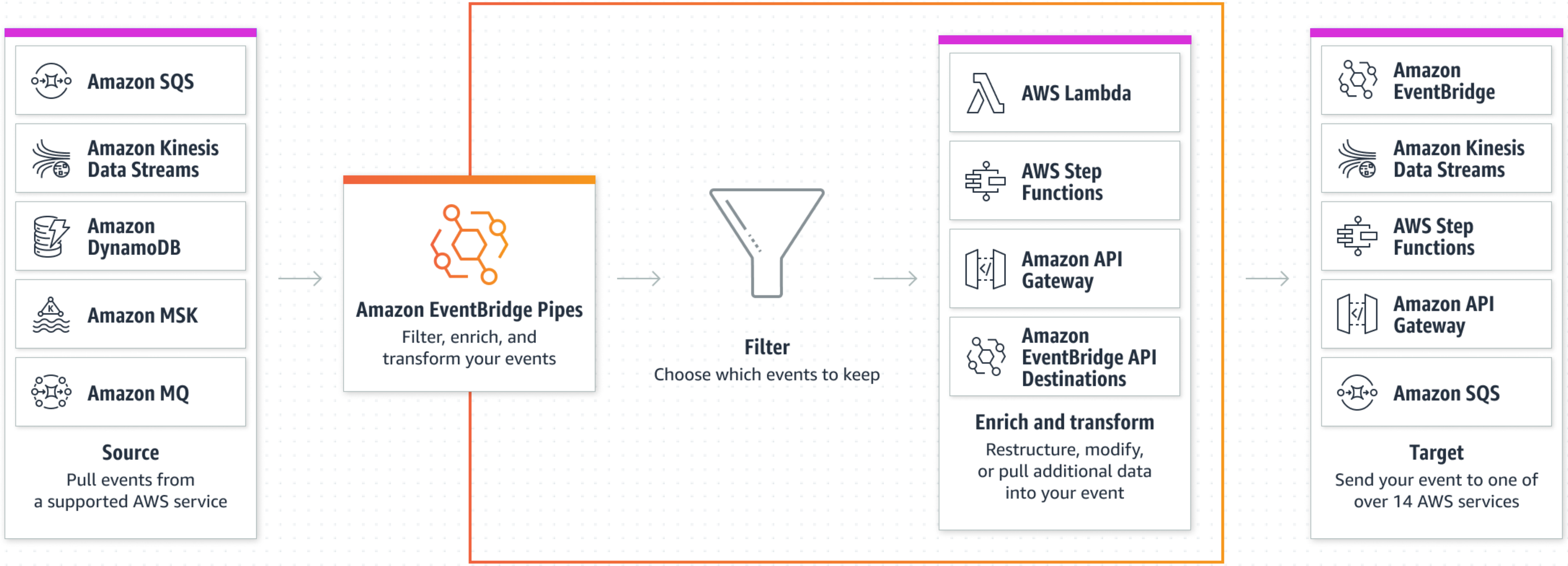
Amazon EventBridge is a simple, flexible, fully managed, pay-as-you-go event bus service that makes it easy to ingest and process data from AWS services, your own applications, and SaaS applications.



Amazon EventBridge architecture



Amazon EventBridge Pipes


Amazon SQS
Amazon Kinesis Data Streams
Amazon DynamoDB
Amazon MSK
Amazon MQ
Source
Pull events from a supported AWS service
Amazon EventBridge Pipes
Filter, enrich, and transform your events
Filter
Choose which events to keep
AWS Lambda
AWS Step Functions
Amazon API Gateway
Amazon EventBridge API Destinations
Enrich and transform
Restructure, modify, or pull additional data into your event
Amazon EventBridge
Amazon Kinesis Data Streams
AWS Step Functions
Amazon API Gateway
Amazon SQS
Target
Send your event to one of over 14 AWS services

How are pipes different from event buses?

Event buses

Many publishers to many consumers

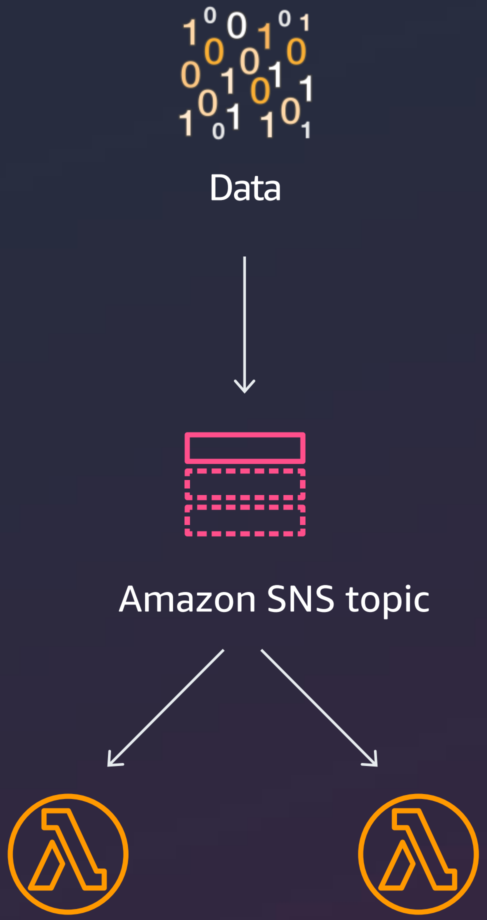
Pipes

Single publisher to single consumer

Amazon Simple Notification Service (Amazon SNS)

Fully managed publish/subscribe messaging for microservices, distributed systems, and serverless applications

Amazon Simple Notification Service (Amazon SNS)

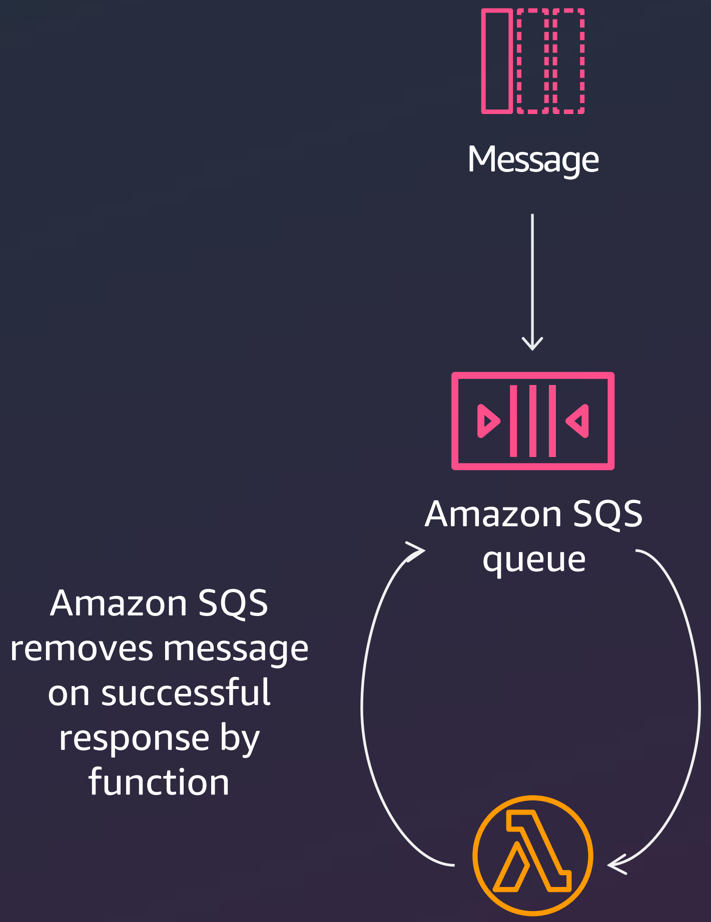


- Publish/subscribe messaging
- Messages are published to a topic with multiple subscribers – “fan out”
- High throughput, highly reliable message delivery
- Messages can be filtered and only sent to certain subscribers

Amazon Simple Queue Service (Amazon SQS)

Fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications

Amazon Simple Queue Service (Amazon SQS)

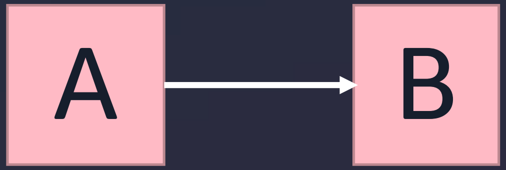

Message
Amazon SQS queue
Amazon SQS removes message on successful response by function

- Any volume of messages
- Messages processed in batches
- At least once delivery with standard queues, exactly once with Amazon SQS FIFO queues
- Visibility timeout allows handling failures
- Service long poll queues

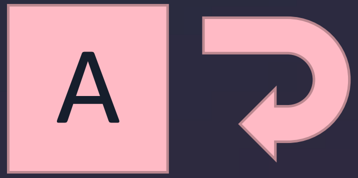
Orchestration

Why orchestration?

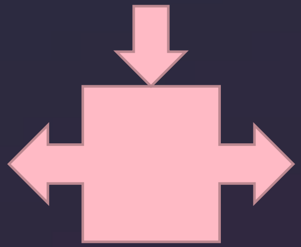
"I want to sequence tasks"



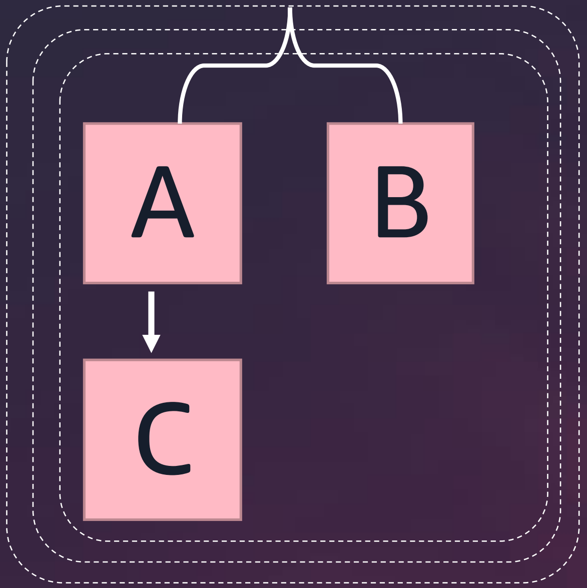
"I want to retry failed tasks"



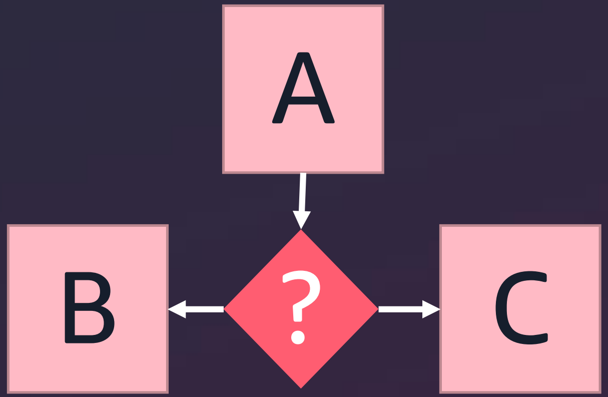
"I want try/catch/finally"



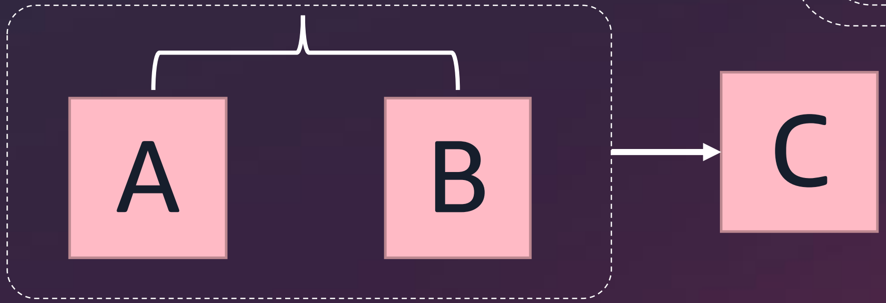
"I want concurrent and iterative tasks"



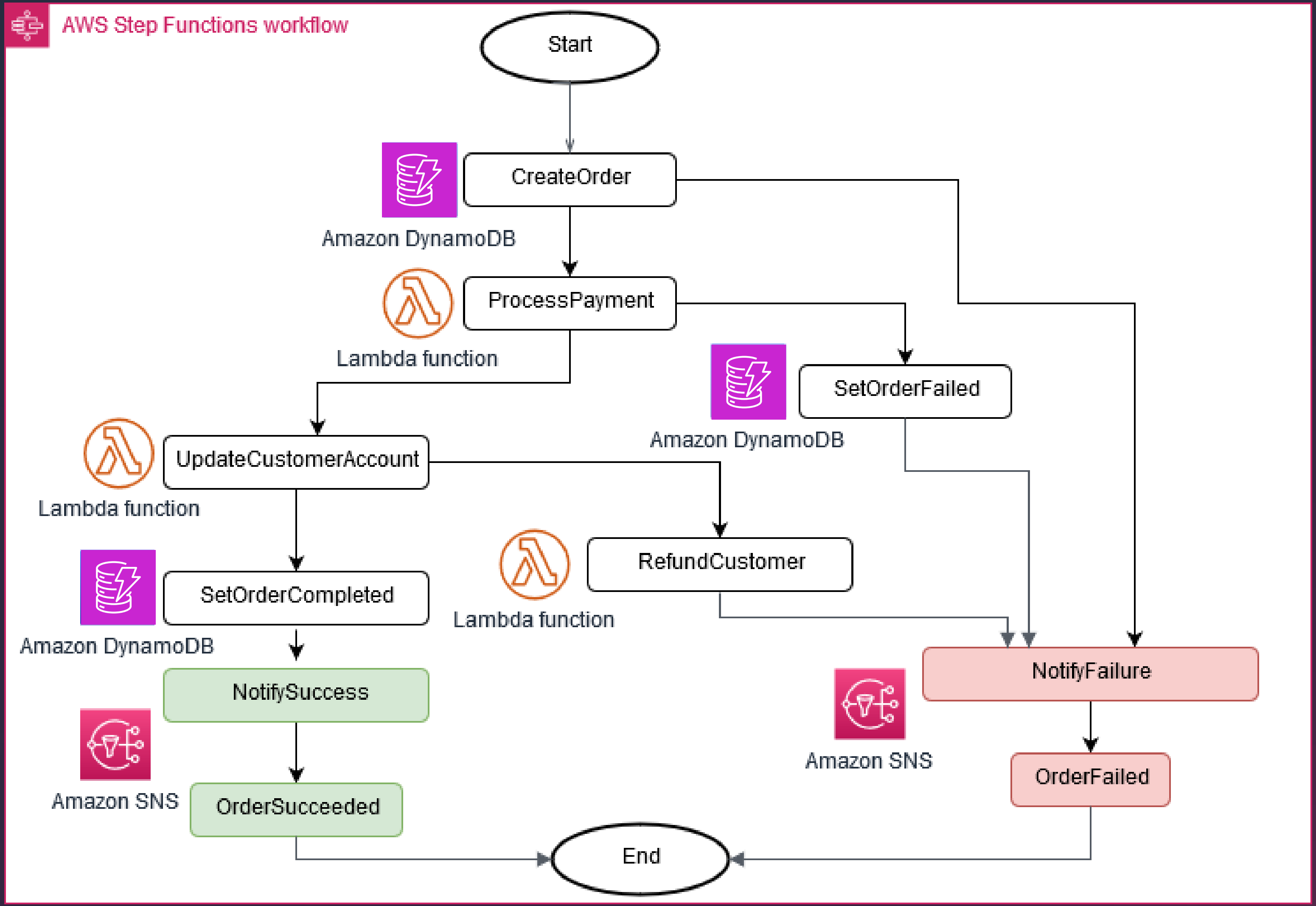
"I want to select tasks based on data"



"I want to run tasks in parallel"



Saga orchestration

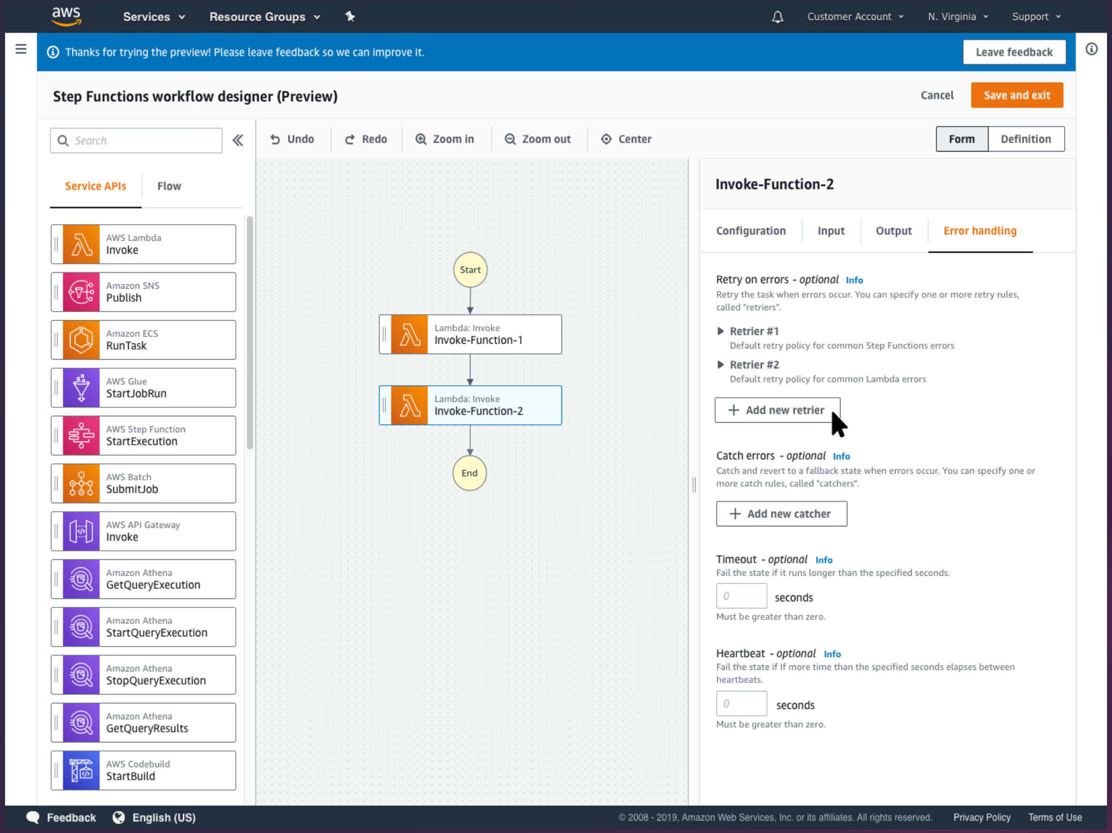


AWS Step Functions

The **workflows** you build with AWS Step Functions are called **state machines**, and **each step** of your workflow is called a **state.**

When you execute your state machine, **each move** from one state to the next is called a **state transition.**

You can **reuse components**, easily edit the sequence of steps, or swap out the code called by task states as your needs change.



Thanks for trying the preview! Please leave feedback so we can improve it.

Step Functions workflow designer (Preview)

Retry on errors - *optional* Info
Retry the task when errors occur. You can specify one or more retry rules, called "retriers".

Retrier #1
Default retry policy for common Step Functions errors

Retrier #2
Default retry policy for common Lambda errors

Catch errors - *optional* Info
Catch and revert to a fallback state when errors occur. You can specify one or more catch rules, called "catchers".

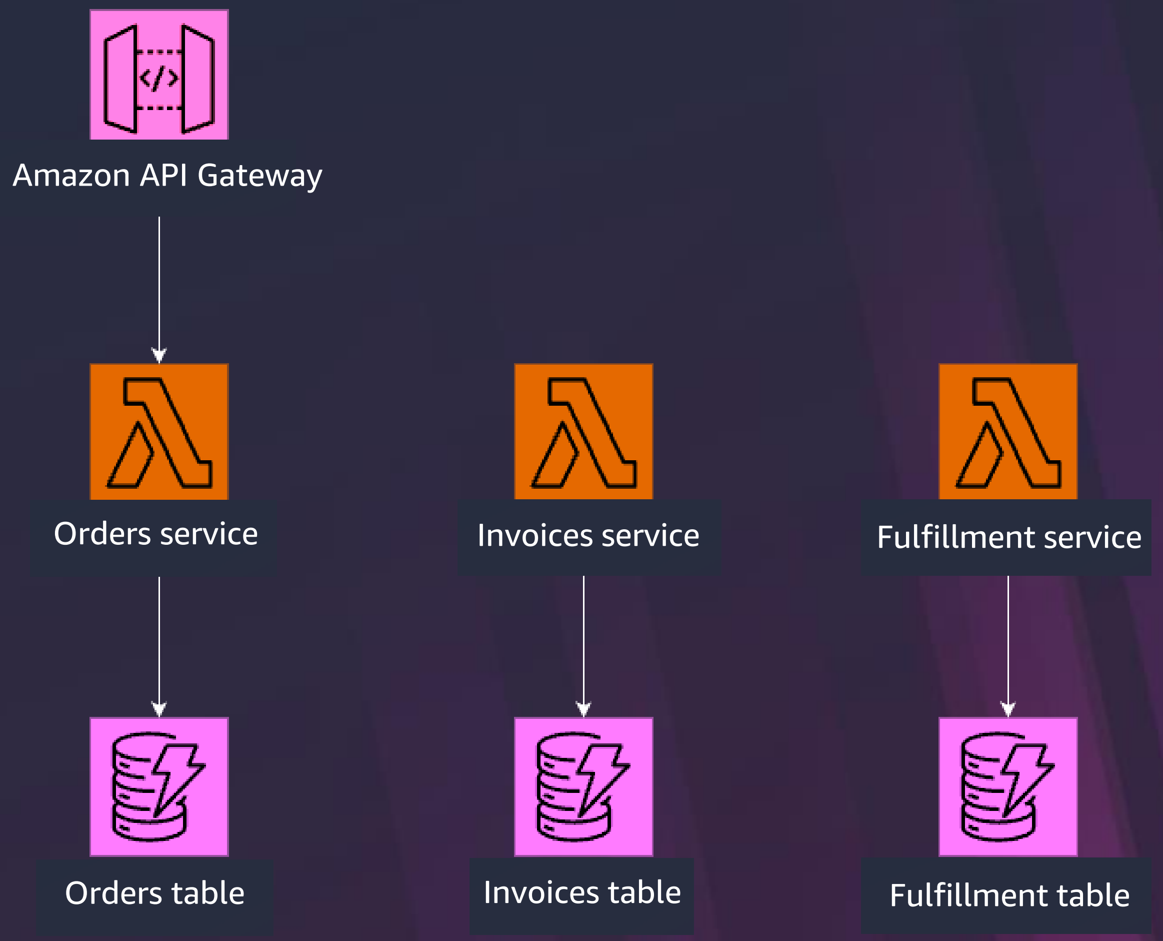
Timeout - *optional* Info
Fail the state if it runs longer than the specified seconds.
0 seconds
Must be greater than zero.

Heartbeat - *optional* Info
Fail the state if more time than the specified seconds elapses between heartbeats.
0 seconds
Must be greater than zero.

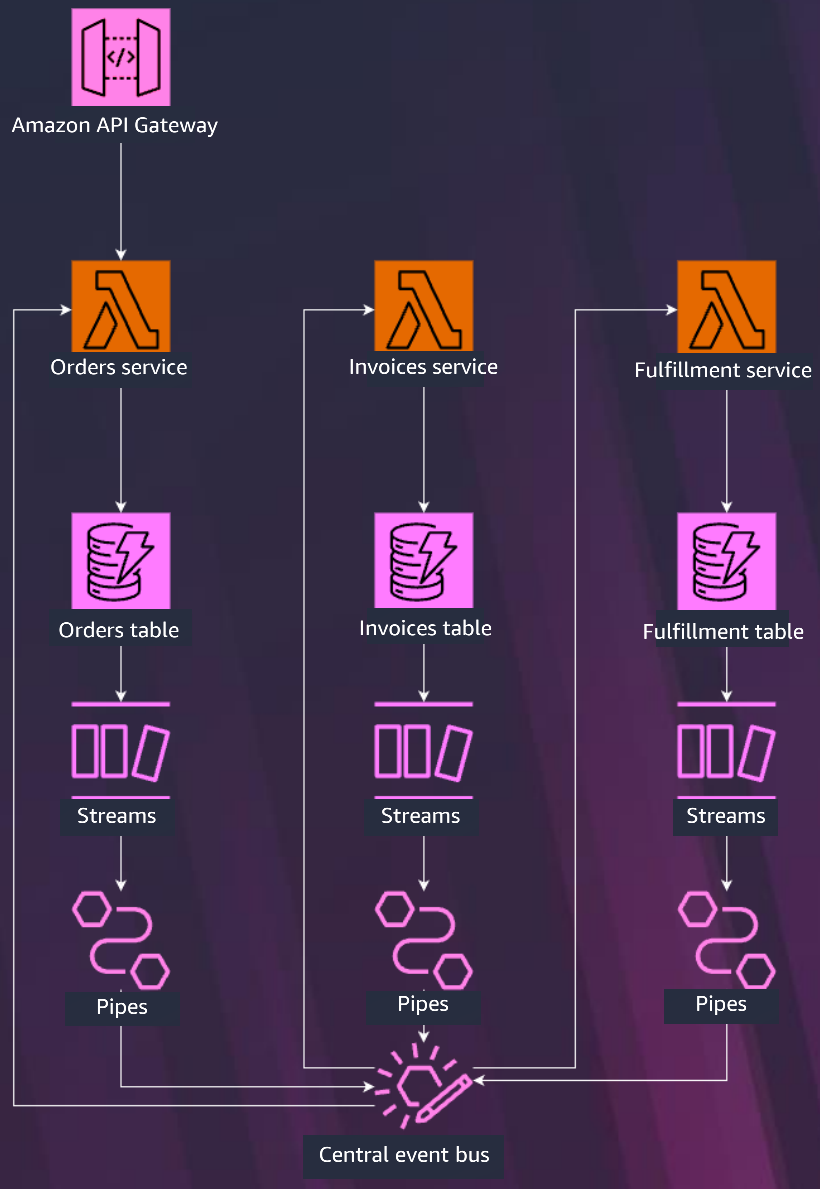
© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Step Functions Workflow Studio

Demo


Amazon API Gateway
Orders service
Invoices service
Fulfillment service
Orders table
Invoices table
Fulfillment table

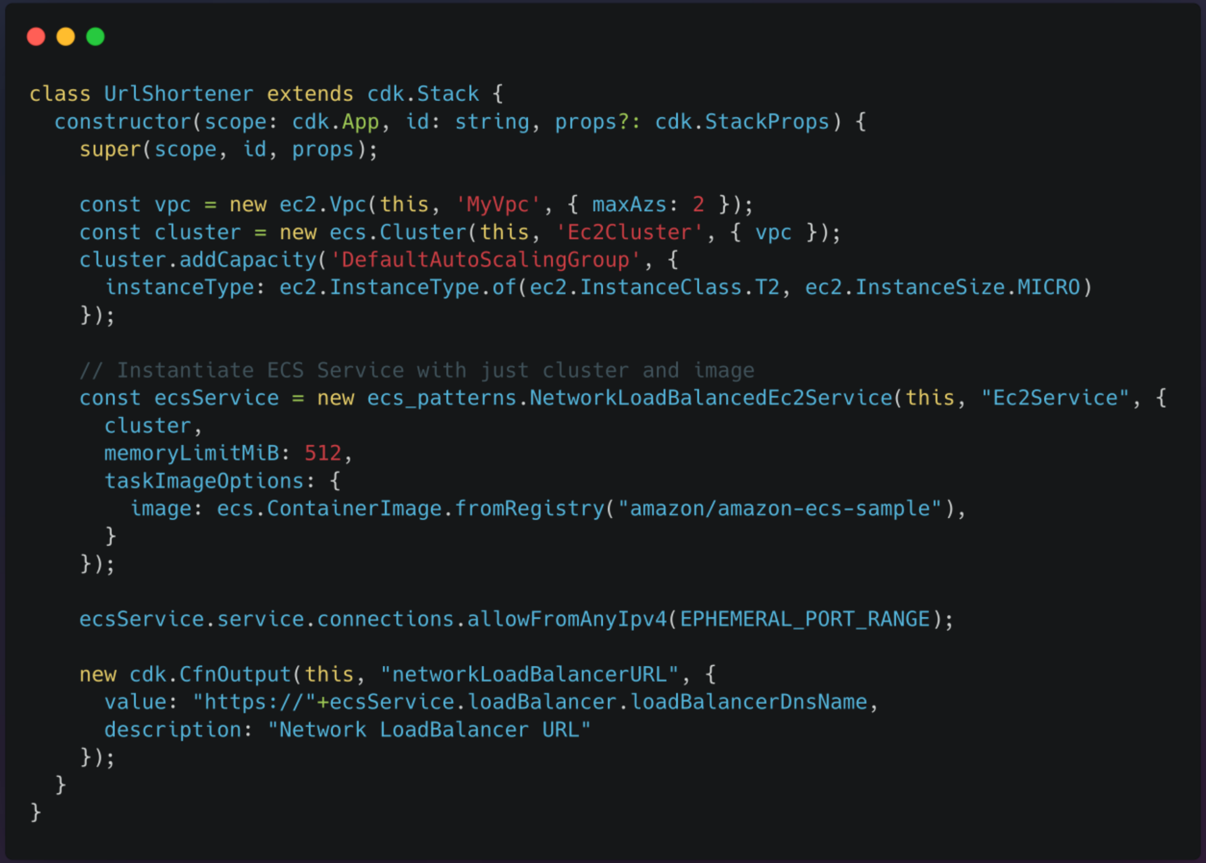
Demo
Target architecture



DevOps

AWS Cloud Development Kit (AWS CDK)

A multi-language software development framework for modeling cloud infrastructure as reusable components



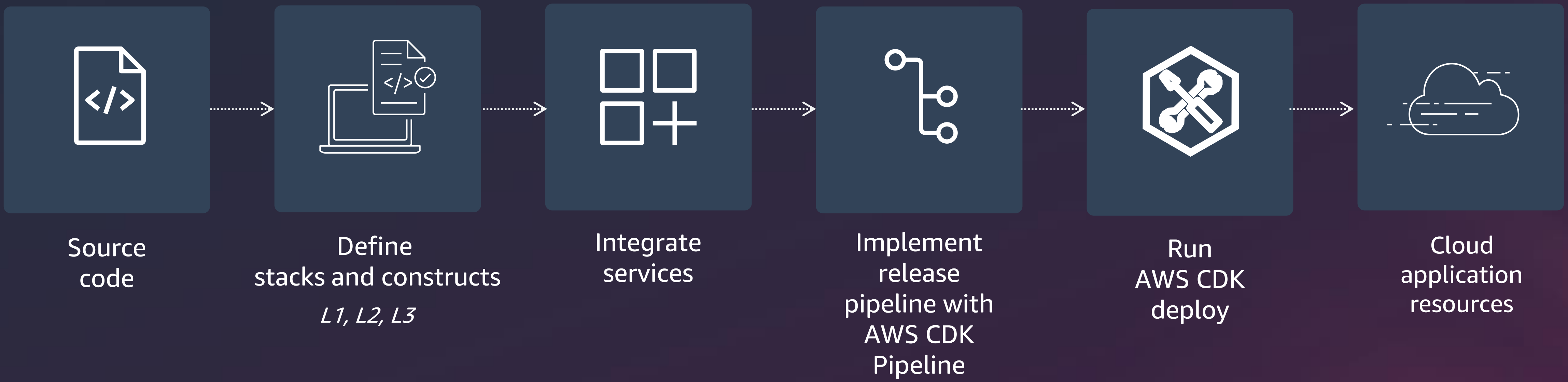
```
class UrlShortener extends cdk.Stack {
  constructor(scope: cdk.App, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    const vpc = new ec2.Vpc(this, 'MyVpc', { maxAzs: 2 });
    const cluster = new ecs.Cluster(this, 'Ec2Cluster', { vpc });
    cluster.addCapacity('DefaultAutoScalingGroup', {
      instanceType: ec2.InstanceType.of(ec2.InstanceClass.T2, ec2.InstanceSize.MICRO)
    });

    // Instantiate ECS Service with just cluster and image
    const ecsService = new ecs_patterns.NetworkLoadBalancedEc2Service(this, "Ec2Service", {
      cluster,
      memoryLimitMiB: 512,
      taskImageOptions: {
        image: ecs.ContainerImage.fromRegistry("amazon/amazon-ecs-sample"),
      }
    });

    ecsService.service.connections.allowFromAnyIpv4(EPHEMERAL_PORT_RANGE);

    new cdk.CfnOutput(this, "networkLoadBalancerURL", {
      value: "https://"+ecsService.loadBalancer.loadBalancerDnsName,
      description: "Network LoadBalancer URL"
    });
  }
}
```

Familiar
Your language – just code

Tool support
Autocomplete – inline documentation

Abstraction
Sane defaults – reusable classes

Composing applications with AWS CDK


Source code
Define stacks and constructs
L1, L2, L3
Integrate services
Implement release pipeline with AWS CDK Pipeline
Run AWS CDK deploy
Cloud application resources

Observability

Three pillars of observability

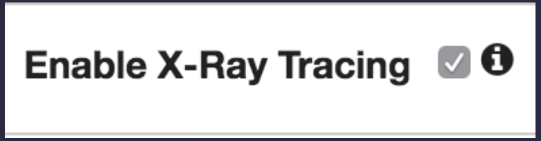
Metrics	Logs	Traces
Numeric data measured at various time intervals (time series data); SLIs (request rate, error rate, duration, CPU%, etc.)	Timestamped records of discrete events that happened within an application or system, such as a failure, an error, or a state transformation	A trace represents a single user's journey across multiple applications and systems (usually microservices)

Definitions from: Distributed Systems Observability
https://www.oreilly.com/library/view/distributed-systems-observability/9781492033431/

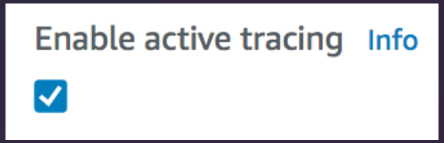
AWS X-Ray

End-to-end view of requests flowing through an application

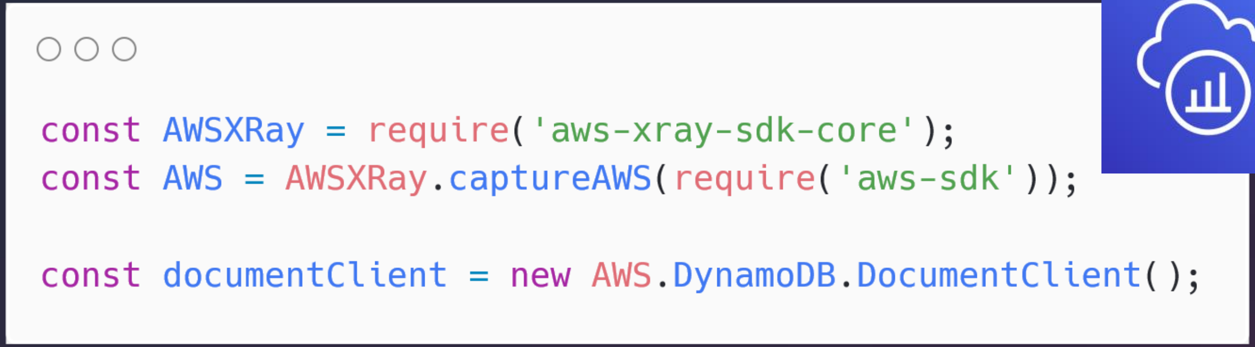
AWS Lambda: Instruments incoming requests for all supported languages and can capture calls made in code.


Enable X-Ray Tracing ✓

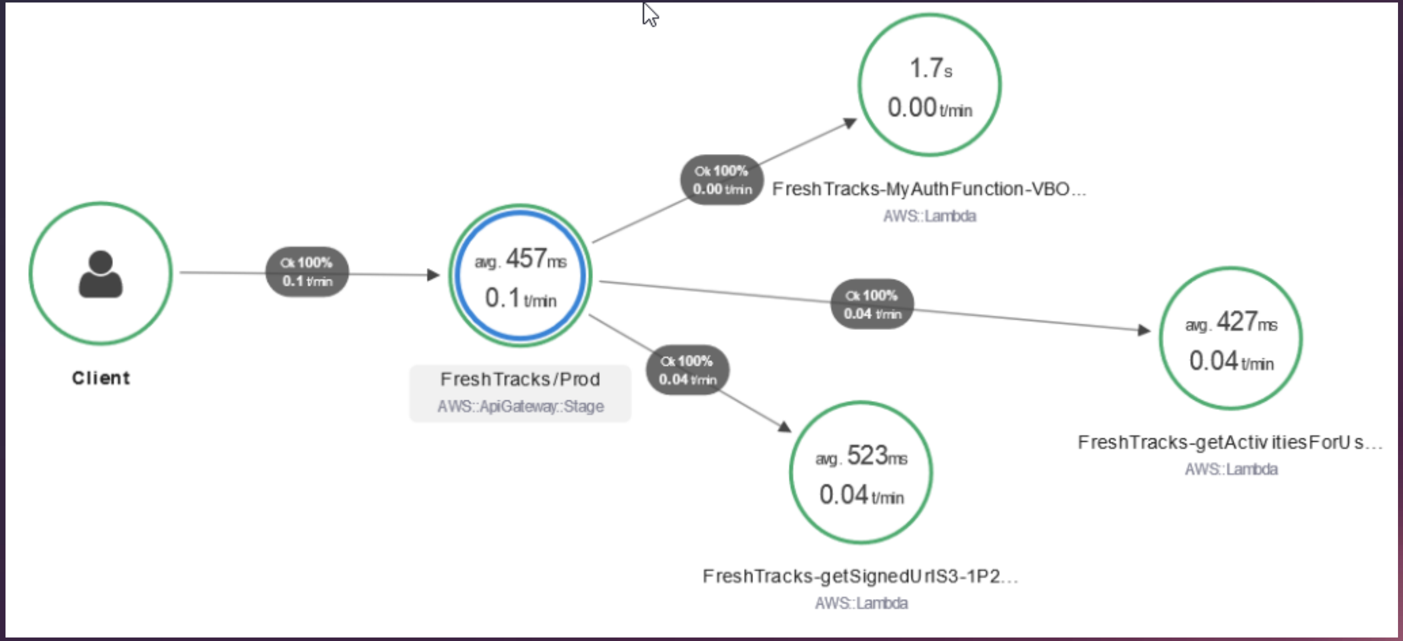
Amazon API Gateway: Inserts a tracing header into HTTP calls and reports data back to AWS X-Ray itself.


Enable active tracing Info

✓



```
const AWSXRay = require('aws-xray-sdk-core');
const AWS = AWSXRay.captureAWS(require('aws-sdk'));

const documentClient = new AWS.DynamoDB.DocumentClient();
```



Summary

Visit the Migrate. Modernize. Build. resource hub

Dive deeper into these resources:

- 6 steps to success with generative AI
- Understanding the costs of generative AI
- 5 ways a secure cloud infrastructure drives innovation
- 10 ways to optimize costs and innovate with AWS
- Containers and serverless recommendation guide
- Running Windows workloads on AWS: Your questions answered
- Top 10 reasons to choose AWS for SAP

… and more!

https://tinyurl.com/migrate-modernize-build

Visit resource hub

Thank you for attending AWS Innovate – Migrate. Modernize. Build.

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

linkedin.com/company/amazon-web-services

twitch.tv/aws

Thank you!