



---

ST.LOUIS

# SERVERLESS



# *Serverless on AWS*

2023-Q1 Meeting & Workshop

presented by Jack Frosch<sup>2</sup>

# About Me

[in](#) [@jackfrosch](#)



**NEXT.js**

 React

 Gatsby



 **amazon web services™**



 NFJS

 ST.LOUIS  
**SERVERLESS**

 aws  
Cloud Development Kit

 **Pulumi**

 K<sup>n</sup>

 docker

 kubernetes



 kafka

 Groovy

 GRAILS



 spring



# About Me

[in](#) [tw](#) @jackfrosch



NEXT.js

React

Gatsby



amazon  
web services™

NFJS

ST.LOUIS  
SERVERLESS

aws  
Cloud Development Kit

Pulumi



K<sup>n</sup>

docker

kubernetes



kafka

Groovy

GRAILS



spring



# About Us

- St. Louis Serverless covers a range of serverless topics
  - AWS
  - Kubernetes
  - Managed, serverless offerings from a variety of vendors
  - Edge functions from traditionally UI
- We're trying a quarterly, hybrid meeting / workshop format
  - Formerly short, low-code, passive, one-hour meetings
  - In 2023, we'll do longer, mixed presentation and lab, meetings



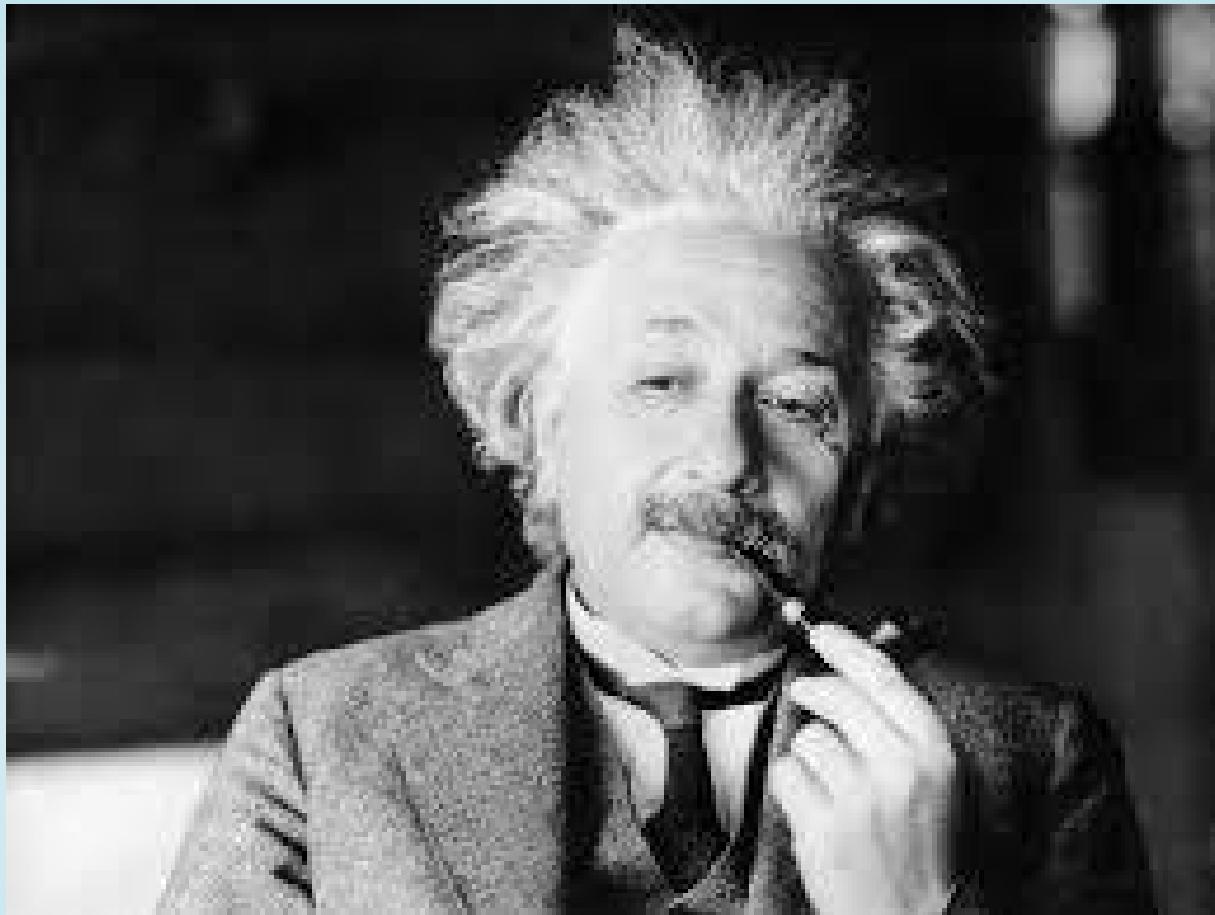
# Next Meeting - July 11th

- 1st Tuesday is July 4th, so this is second Tuesday
- Topic - Roll Your Own Serverless with Knative



# Overview

- About our group
- What is serverless
- IaC with AWS Cloud Development Kit (CDK) + Lab1
- FaaS with Lambda + Lab2
- Serverless Triggers to Lambda + Lab3
- Orchestrating workflows with Step Functions
- Serverless Databases + Lab 4



# What is serverless?

# Let's start with what it isn't



Serverless does not mean, "No servers"



# Amazon Web Services (AWS):

*“ Serverless computing allows you to build and run applications and services without thinking about servers...”*



In summary...

*“Serverless lets you think about servers less.”*

# Path to Serverless



# Path to Serverless

# Path to Serverless

Bare Metal

# Path to Serverless

Virtual Machines



Bare Metal

# Path to Serverless



IaaS



Virtual Machines



Bare Metal

# Path to Serverless

PaaS



IaaS

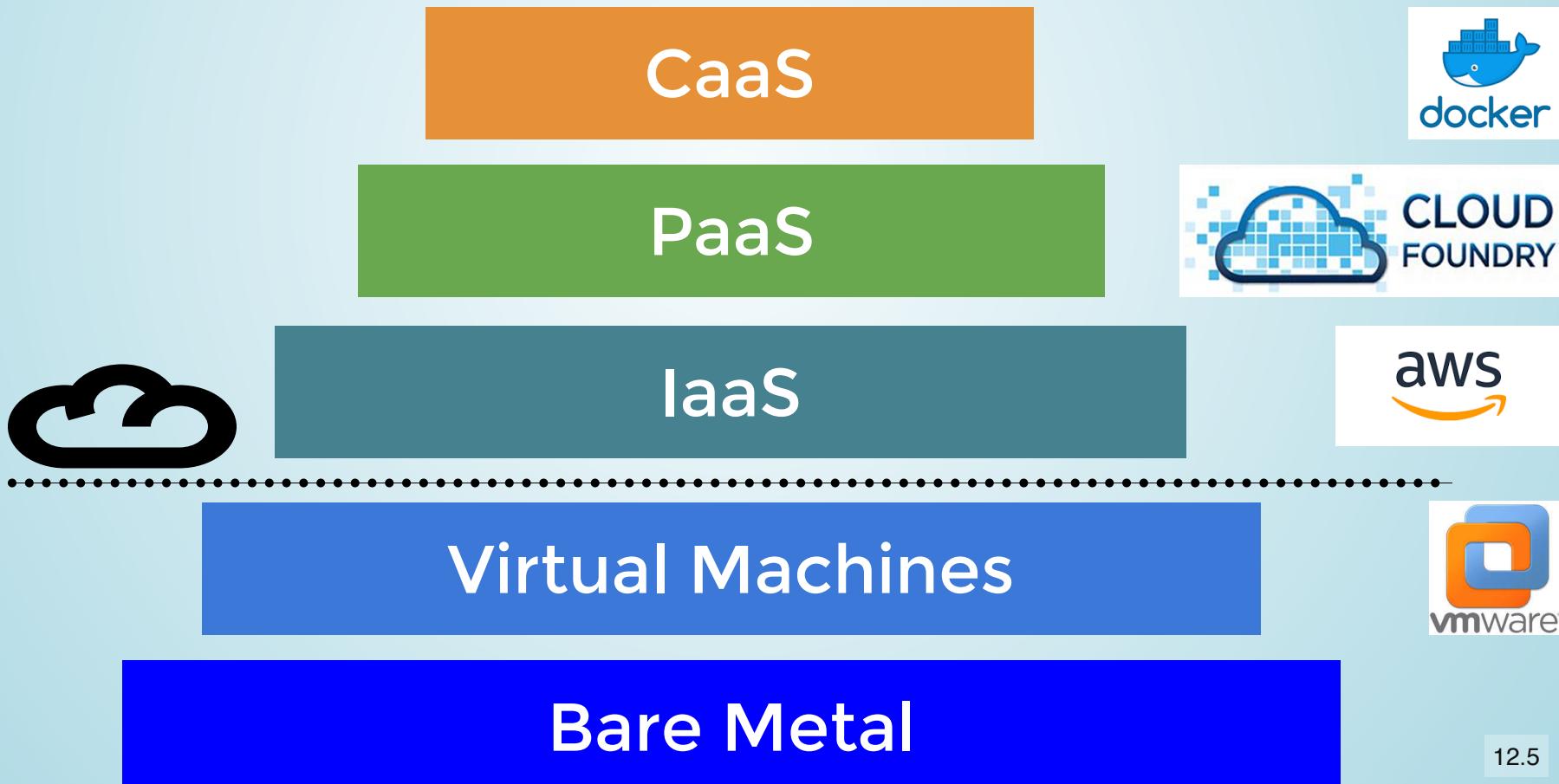


Virtual Machines

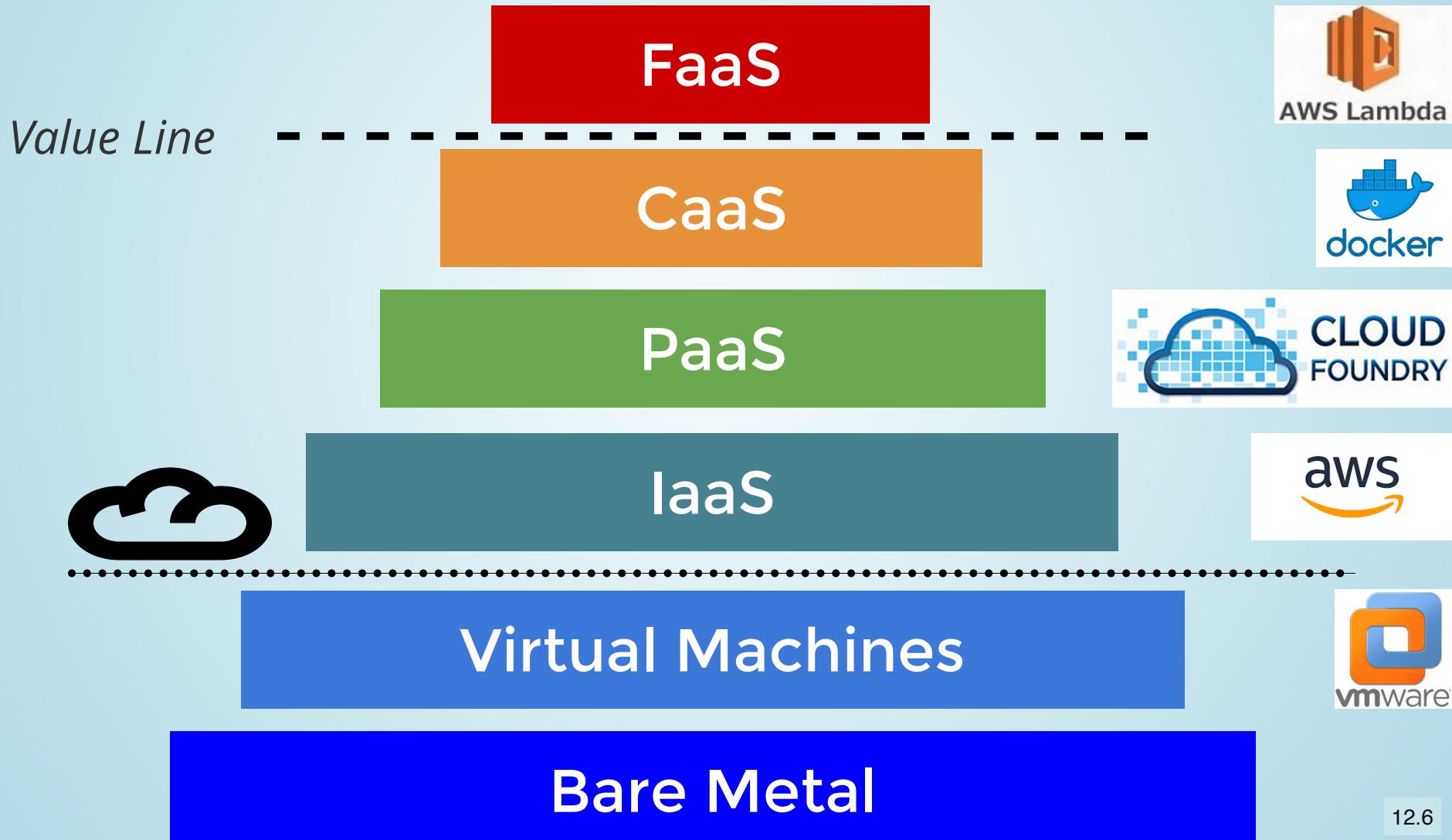


Bare Metal

# Path to Serverless



# Path to Serverless



# Serverless includes...

# Serverless includes...



Functions

# Serverless includes...



Functions



API Gateways

# Serverless includes...



Functions



API Gateways



Queues & Event Buses

# Serverless includes...



Functions



API Gateways



Queues & Event Buses



Notifications

# Serverless includes...



Functions



API Gateways



Queues & Event Buses



Notifications



Monitoring & Logging

# Serverless includes...



Functions



API Gateways



Queues & Event Buses



Notifications



Monitoring & Logging



Storage

# Serverless includes...



Functions



API Gateways



Queues & Event Buses



Notifications



Monitoring & Logging



Storage

and much, much more... Fargate, Aurora Serverless, etc.



# Advantages of Serverless



# Advantages of Serverless

- Development simplicity



# Advantages of Serverless

- Development simplicity
- Zero provisioning or management



# Advantages of Serverless

- Development simplicity
- Zero provisioning or management
- Scale up based on events



# Advantages of Serverless

- Development simplicity
- Zero provisioning or management
- Scale up based on events
- Scale to zero



# Advantages of Serverless

- Development simplicity
- Zero provisioning or management
- Scale up based on events
- Scale to zero
- Highly available



# Advantages of Serverless

- Development simplicity
- Zero provisioning or management
- Scale up based on events
- Scale to zero
- Highly available
- Usage-based billing

# Development Simplicity

Order Management

Authentication

Product Catalog

Web UI

Payment Processing

# Development Simplicity

Order Management  
Authentication

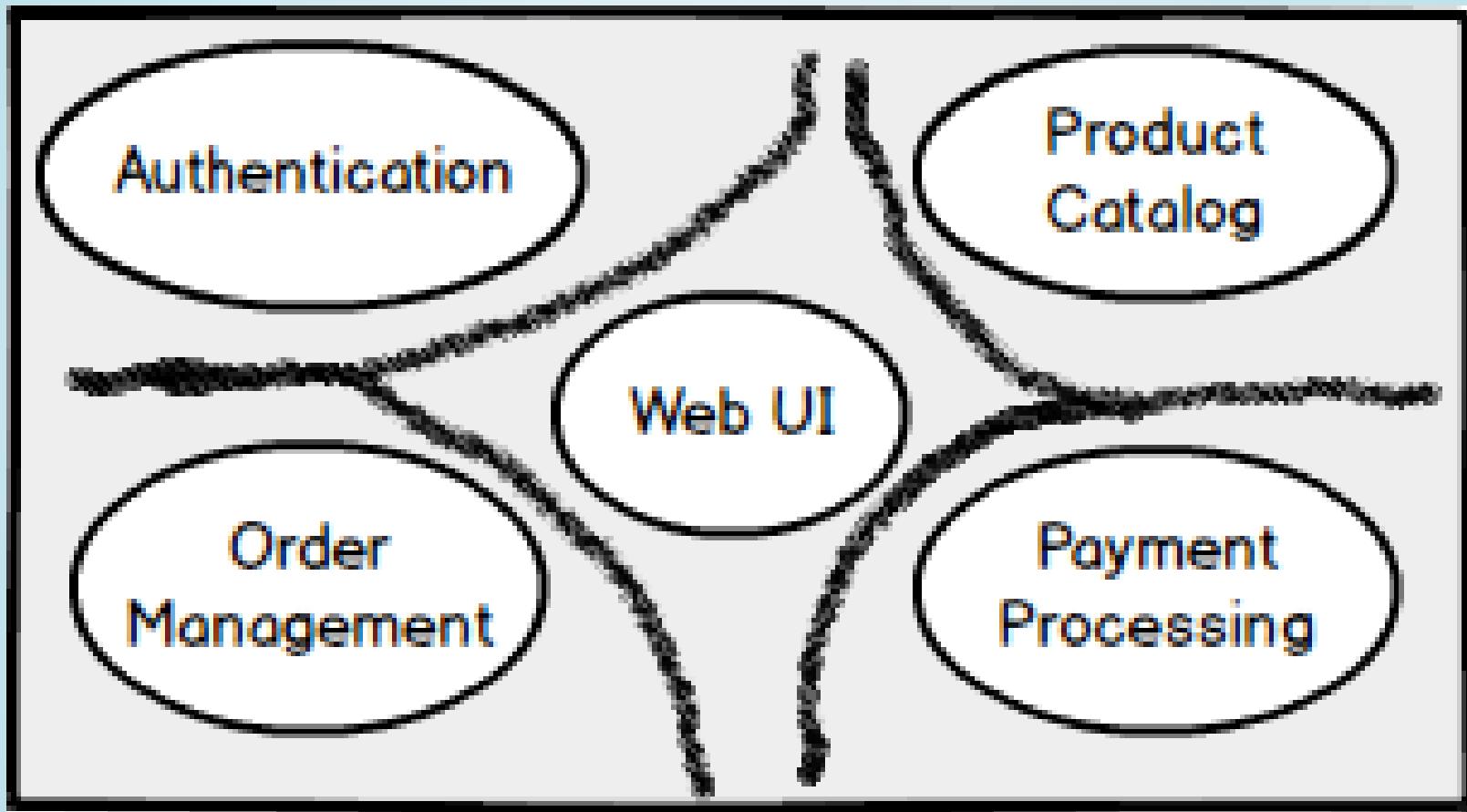
Product Catalog

Web UI

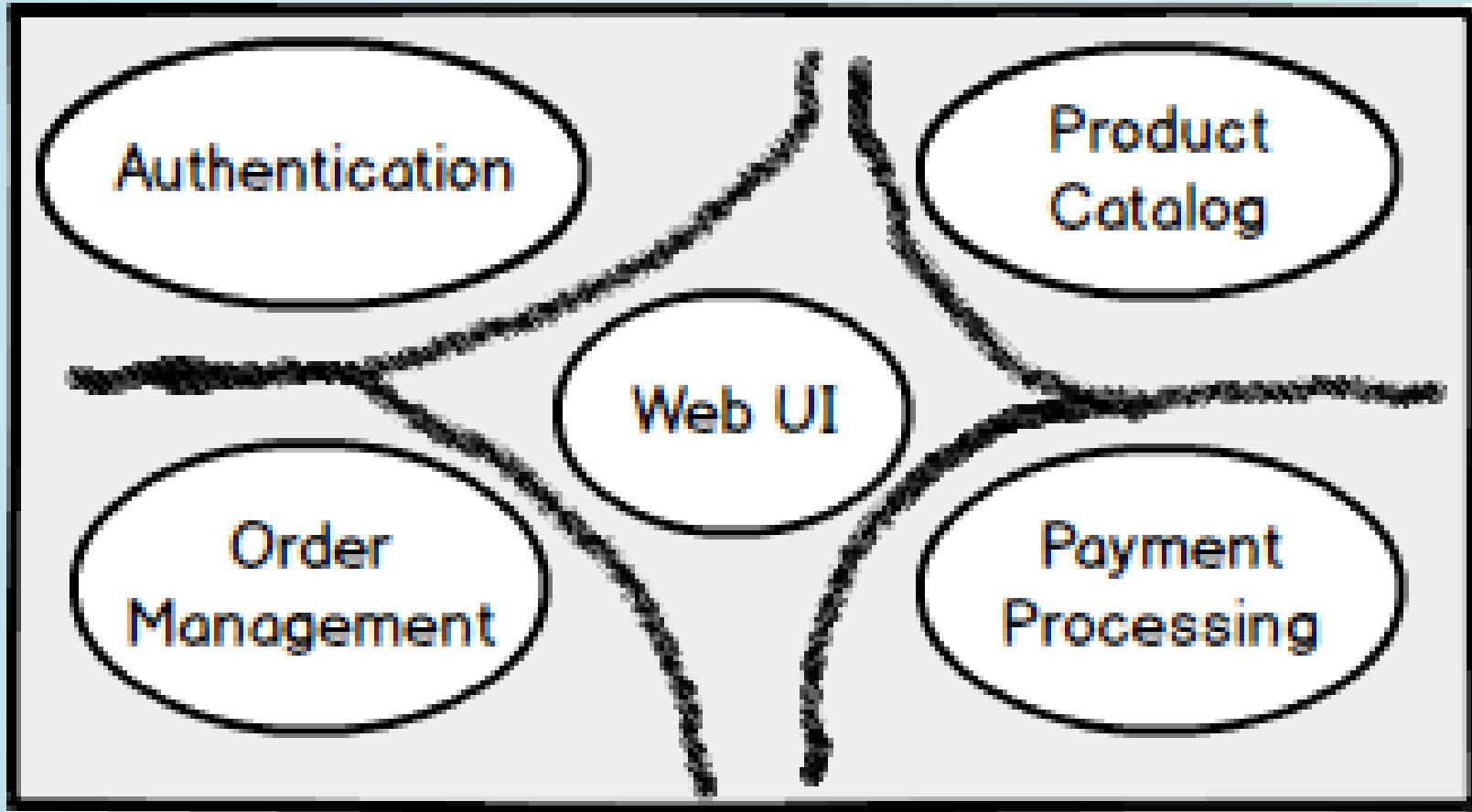
Payment Processing

Over time, monolith applications become increasingly difficult to understand, maintain and enhance

# Development Simplicity



# Development Simplicity



We started breaking up monoliths into  
microservices along *bounded context* seams

# Zero Server Management

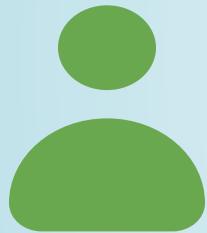




# Scale up based on events



# Scale up based on events



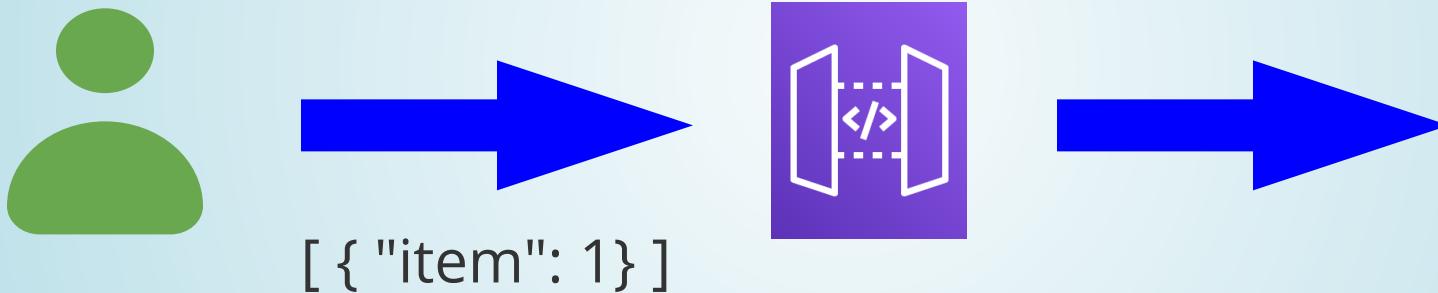


# Scale up based on events



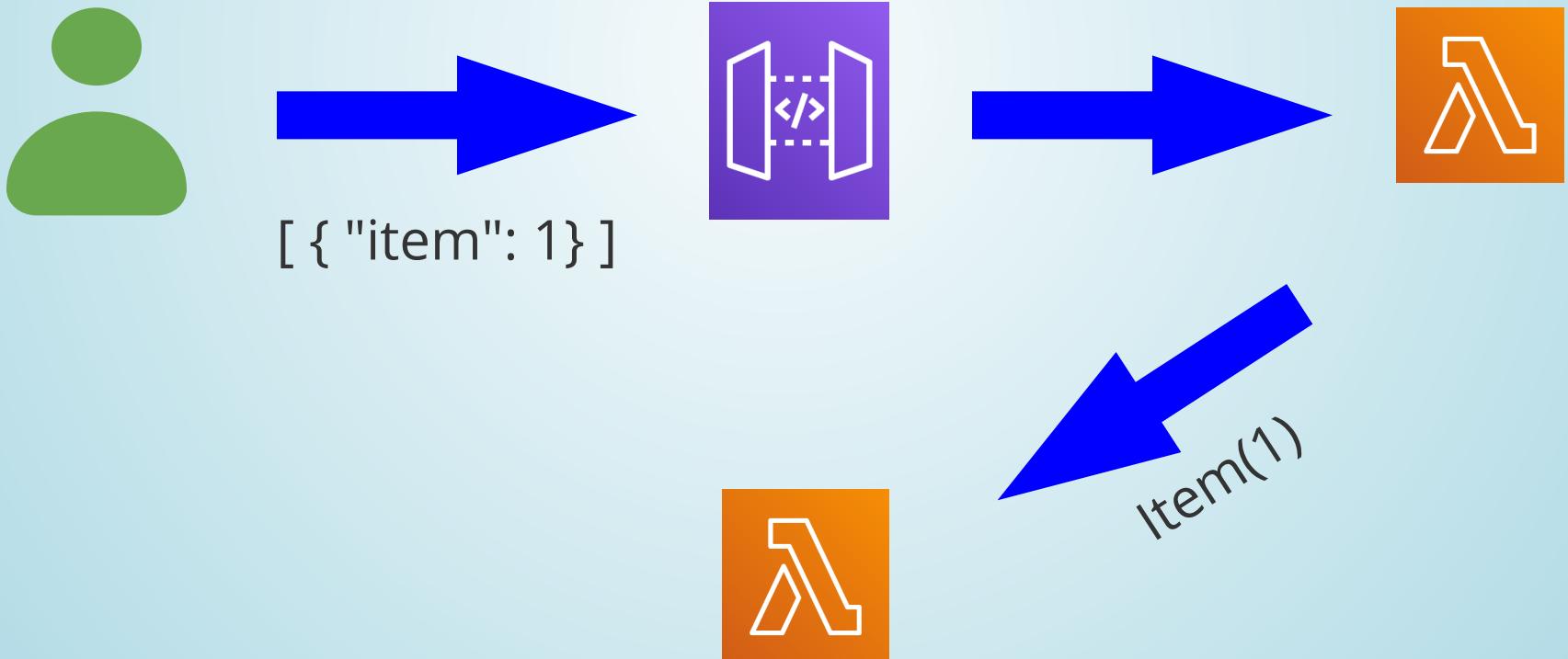


# Scale up based on events





# Scale up based on events

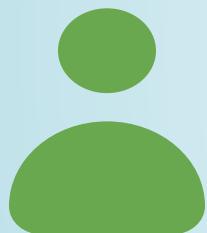




# Scale up based on events

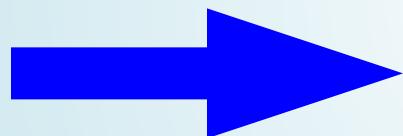
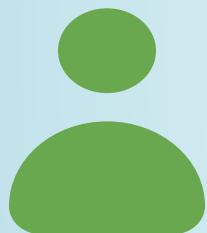


# Scale up based on events

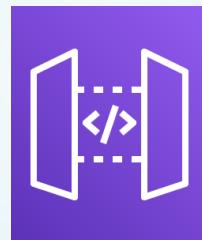




# Scale up based on events

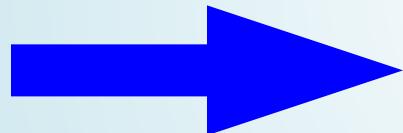
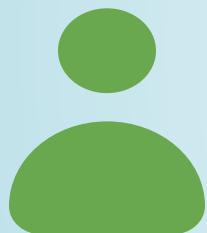


```
[ { "item": 1},  
  ...  
  {"item" : 3} ]
```

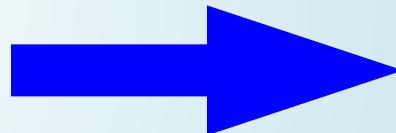
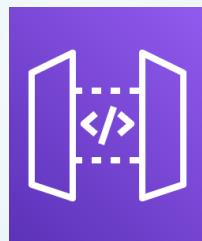




# Scale up based on events

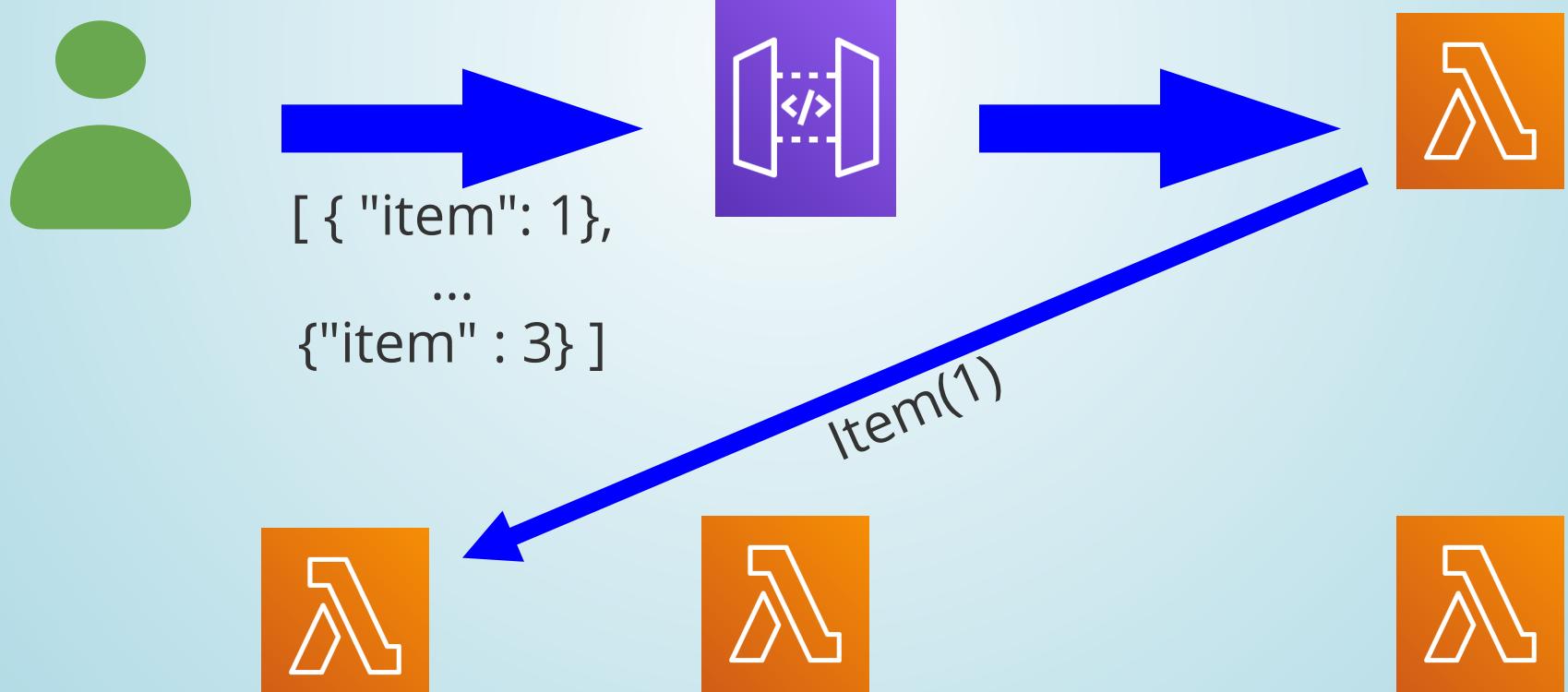


```
[ { "item": 1},  
  ...  
  {"item" : 3} ]
```

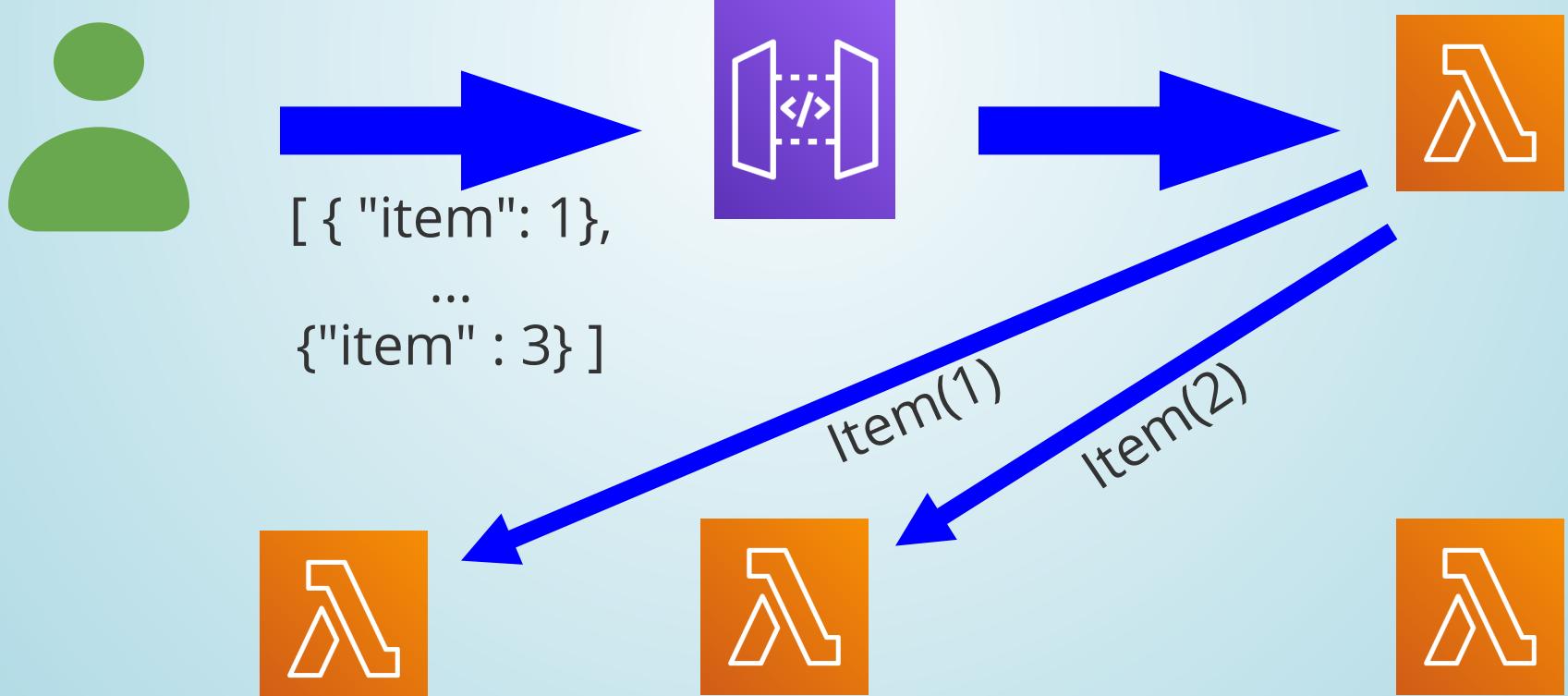




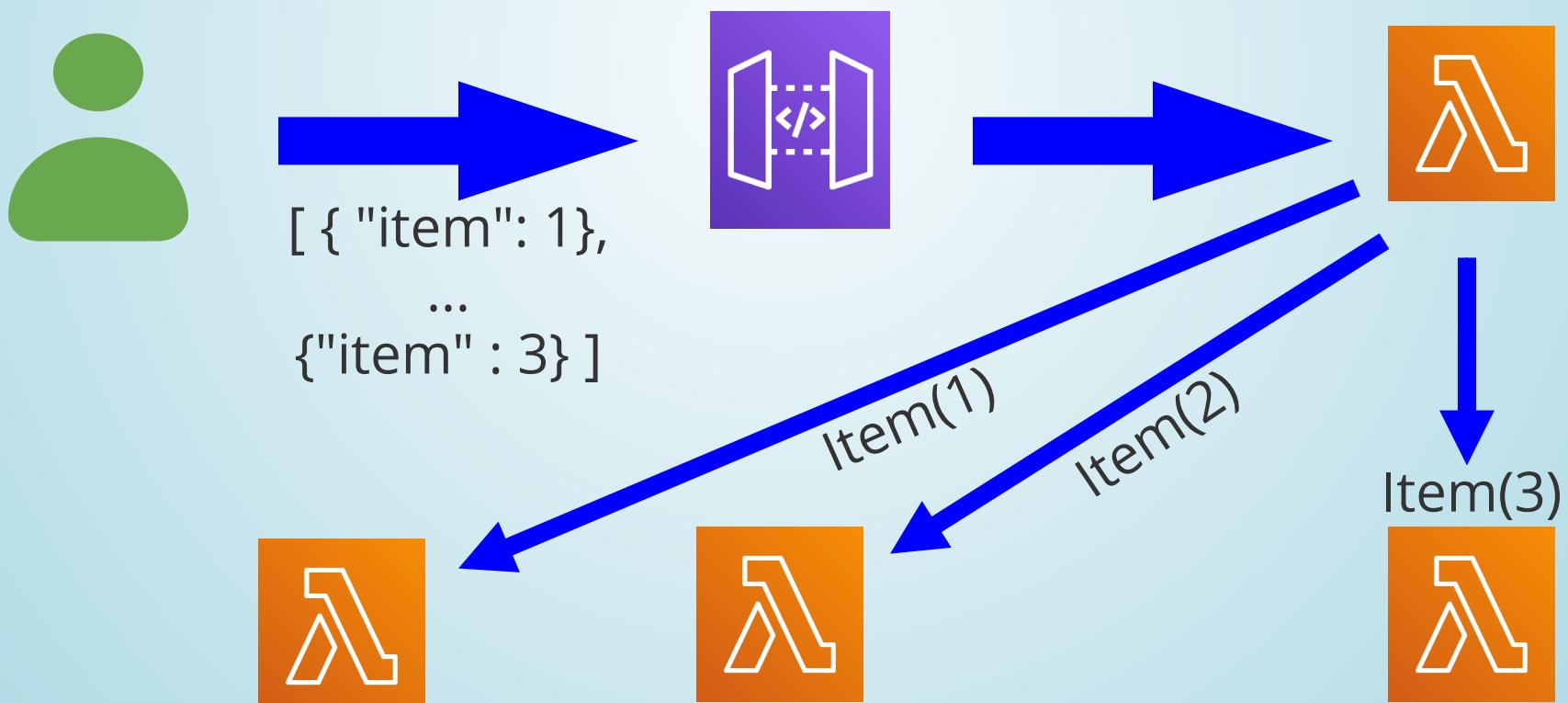
# Scale up based on events



# Scale up based on events



# Scale up based on events



# AWS Lambda Event Sources

# AWS Lambda Event Sources

- Amazon S3

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront
- Amazon Kinesis Data Firehose

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront
- Amazon Kinesis Data Firehose
- Other Lambdas / Destinations

# AWS Lambda Event Sources

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- EventBridge (nee CloudWatch Events)
- AWS CodeCommit
- Scheduled Events (EventBridge)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront
- Amazon Kinesis Data Firehose
- Other Lambdas / Destinations
- ... More!



# Scale to Zero



# Scale to Zero

- Zero or few customers
  - New Project
  - Startup



# Scale to Zero

- Zero or few customers
  - New Project
  - Startup
- Development 7am - 7pm



# Scale to Zero

- Zero or few customers
  - New Project
  - Startup
- Development 7am - 7pm
- Periodic loads
  - Evening batch runs



# Scale to Zero

- Zero or few customers
  - New Project
  - Startup
- Development 7am - 7pm
- Periodic loads
  - Evening batch runs
- Occasional, sporadic loads
  - "Forgot password"
  - CI Build
  - Testing



# Highly Available





# Usage-based Billing



# Usage-based Billing

- Zero or few customers
  - No charge until customers start using it



# Usage-based Billing

- Zero or few customers
  - No charge until customers start using it
- High loads 7am - 7pm
  - No charge after 7pm



# Usage-based Billing

- Zero or few customers
  - No charge until customers start using it
- High loads 7am - 7pm
  - No charge after 7pm
- Periodic loads
  - No charge until batch job runs



# Usage-based Billing

- Zero or few customers
  - No charge until customers start using it
- High loads 7am - 7pm
  - No charge after 7pm
- Periodic loads
  - No charge until batch job runs
- Occasional, sporadic loads
  - No charge until load appears



# However...



# However...

... and there's always a however



# Disadvantages of Serverless



# Disadvantages of Serverless

- Development complexity



# Disadvantages of Serverless

- Development complexity
- Testing complexity



# Disadvantages of Serverless

- Development complexity
- Testing complexity
- Deployment complexity



# Disadvantages of Serverless

- Development complexity
- Testing complexity
- Deployment complexity
- Cold start performance



# Development Complexity



# Development Complexity

- With a monolith, we can have
  - One project
  - One build pipeline
  - One deployment artifact
  - One place to look for a problem



# Development Complexity

- With a monolith, we can have
  - One project
  - One build pipeline
  - One deployment artifact
  - One place to look for a problem
- With microservices, we compound this



# Development Complexity

- With a monolith, we can have
  - One project
  - One build pipeline
  - One deployment artifact
  - One place to look for a problem
- With microservices, we compound this
- With serverless, it's even more complex



# Testing Complexity



# Testing Complexity

- Unit testing functions is easy



# Testing Complexity

- Unit testing functions is easy
- Integration testing is harder



# Testing Complexity

- Unit testing functions is easy
- Integration testing is harder
- We often depend on cloud provider services



# Testing Complexity

- Unit testing functions is easy
- Integration testing is harder
- We often depend on cloud provider services
- Frameworks can help



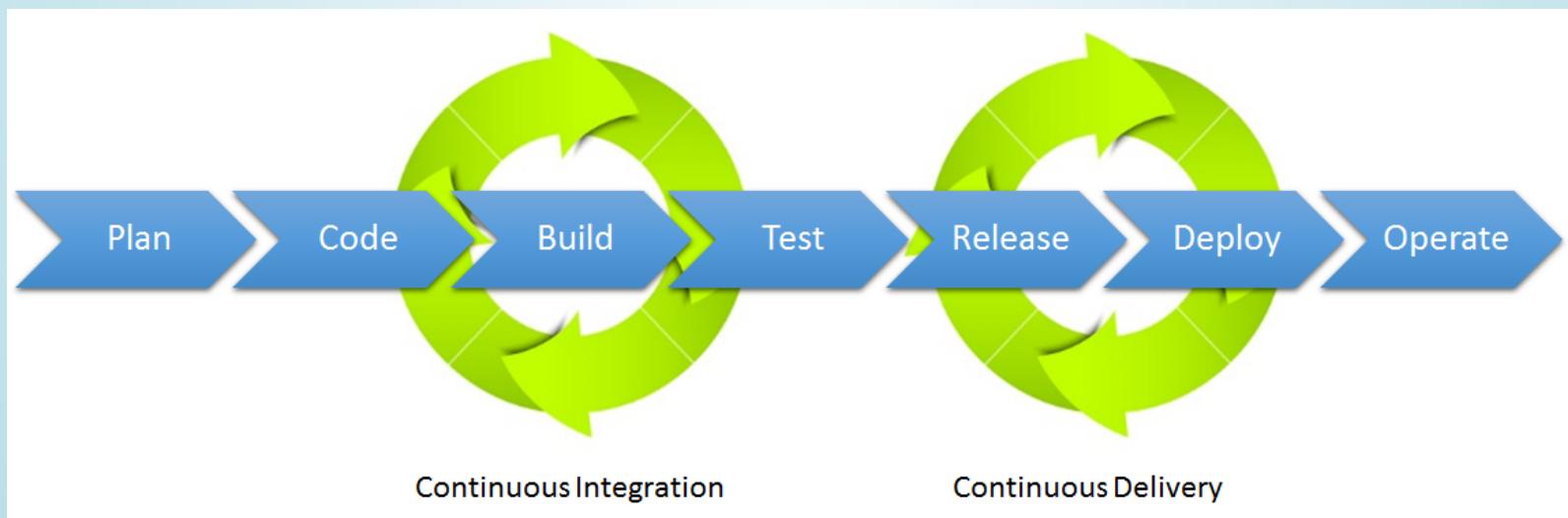
# Testing Complexity

- Unit testing functions is easy
- Integration testing is harder
- We often depend on cloud provider services
- Frameworks can help
- We just need to accept real integration testing must be done on the cloud



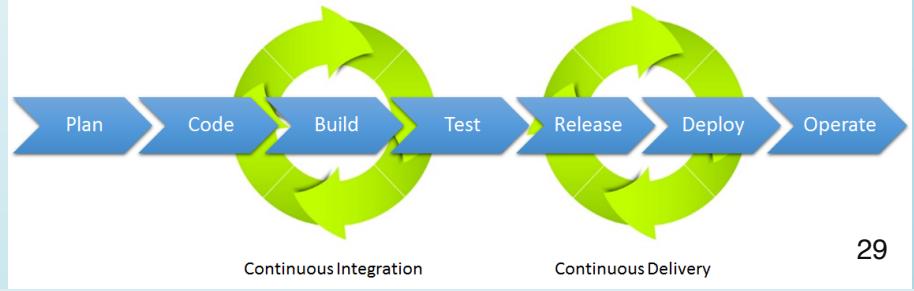
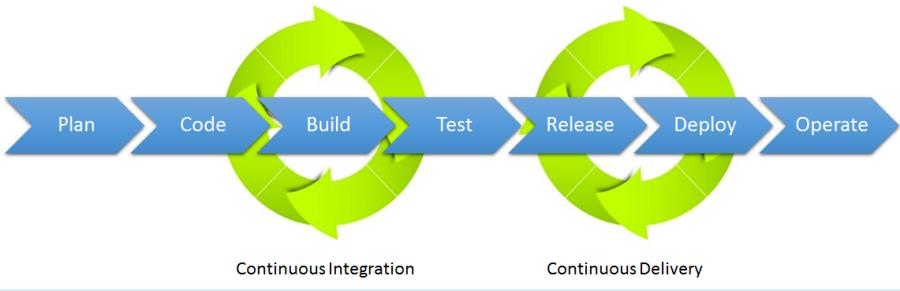
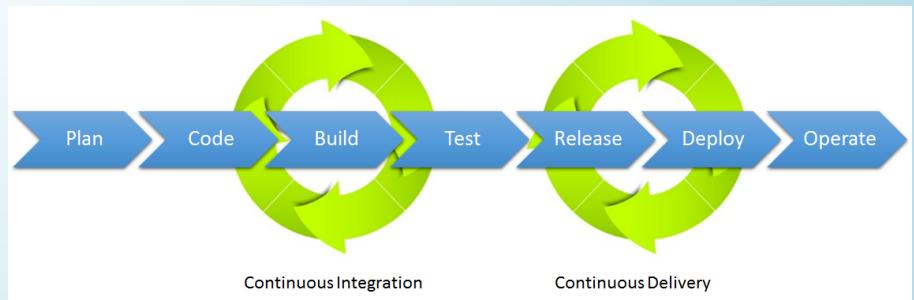
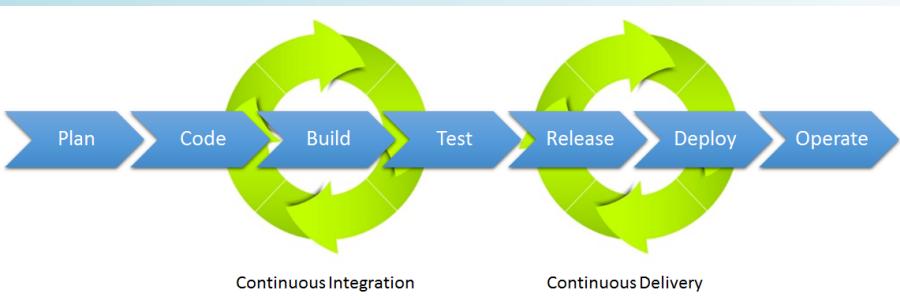
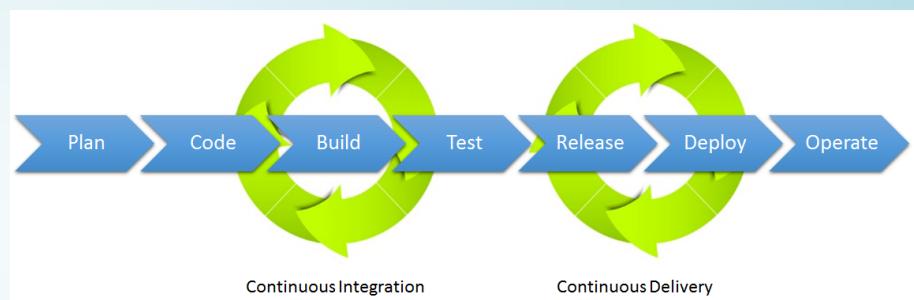
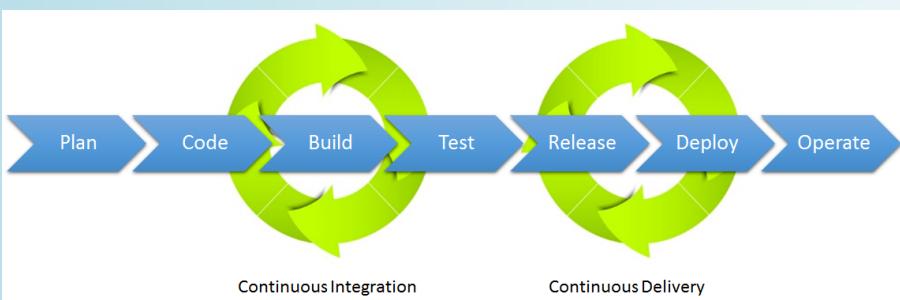
# Deployment Complexity

This...





# Becomes this...





# Cold Start Performance



# Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request



# Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request
- Different languages and frameworks have different cold start penalties



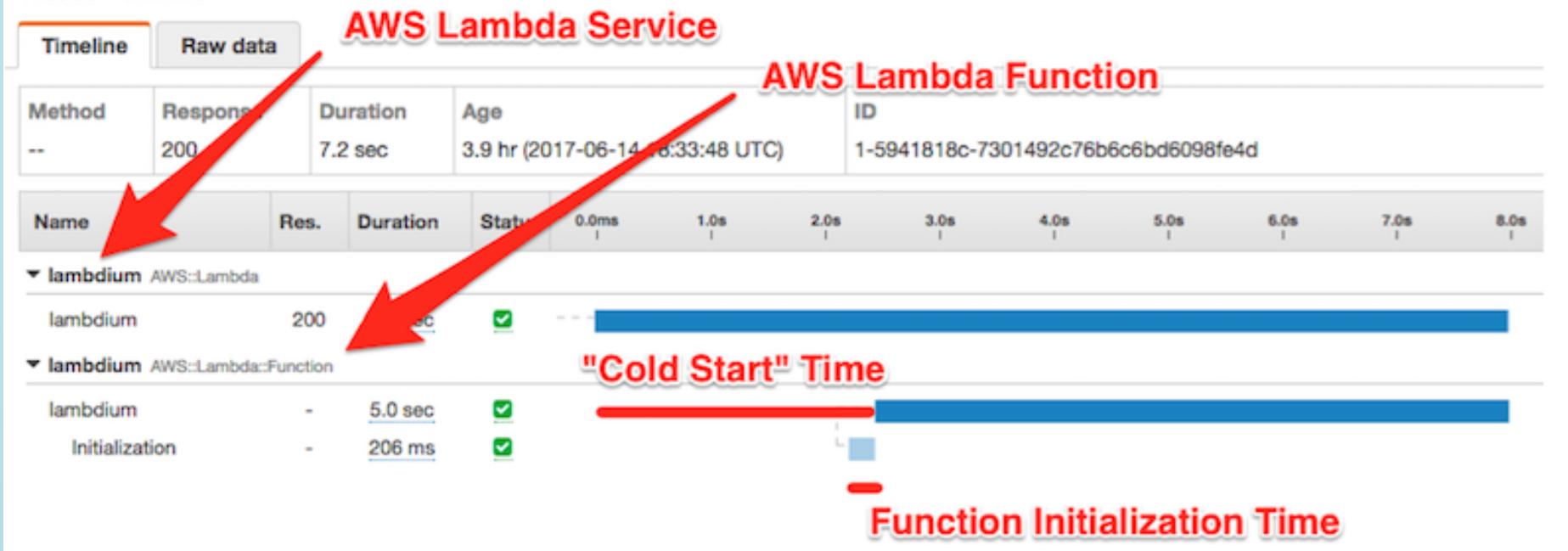
# Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request
- Different languages and frameworks have different cold start penalties
- Available memory dramatically affects cold start performance



# Cold Start Performance

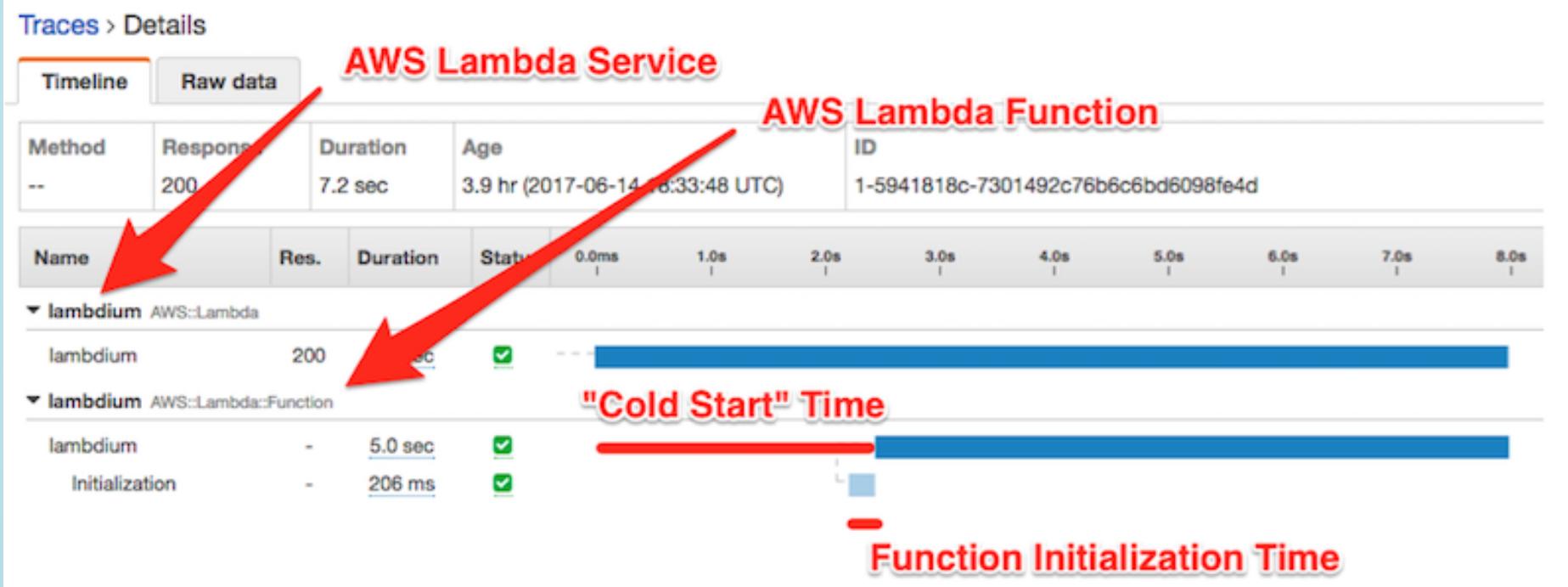
Traces > Details



<https://blog.newrelic.com/engineering/lambda-functions-xray-traces-custom-serverless-metrics/>



# Cold Start Performance

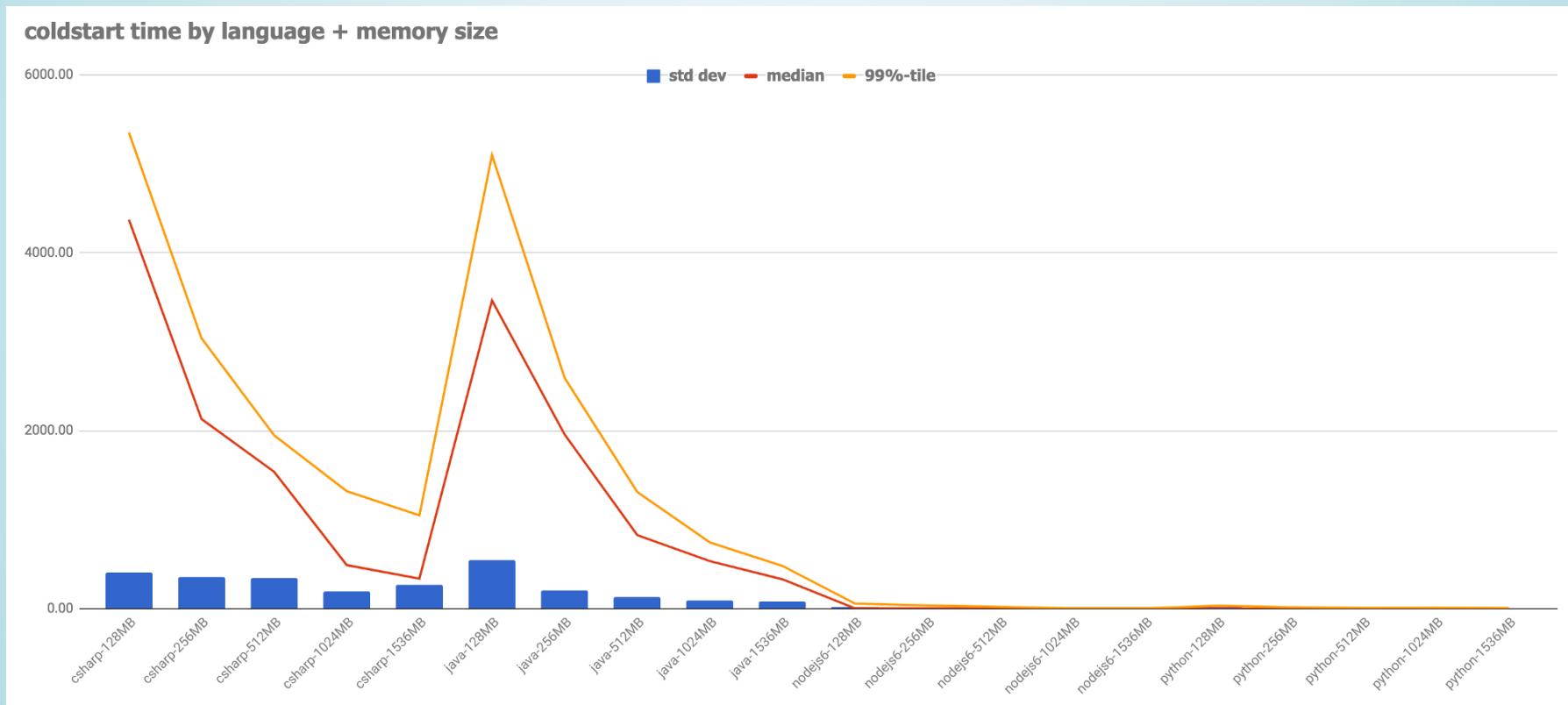


<https://blog.newrelic.com/engineering/lambda-functions-xray-traces-custom-serverless-metrics/>

\* SnapStart is a new cold-start mitigation for Lambda



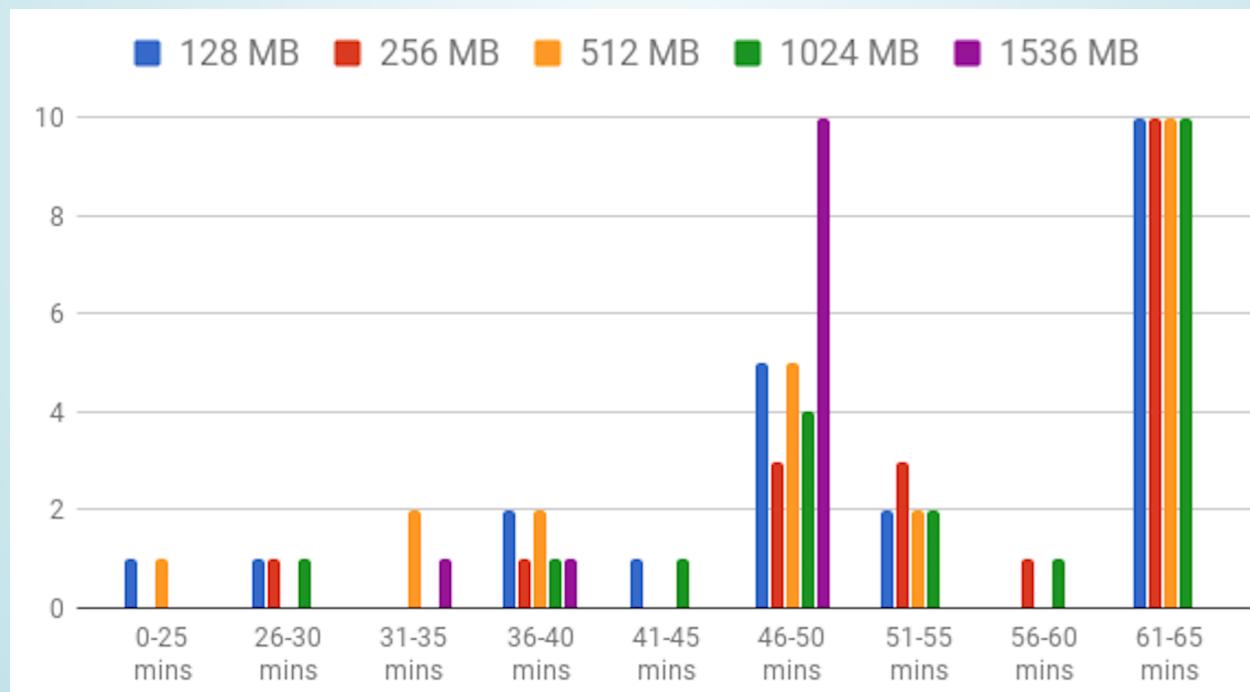
# Cold Start Performance



<https://read.acloud.guru/does-coding-language-memory-or-package-size-affect-cold-starts-of-aws-lambda-a15e26d12c76>

# How long will it stay warm?

	A	C	D	E	F	G	H	I	J	K
1	<b>memory allocation</b>	<b>0-25 mins</b>	<b>26-30 mins</b>	<b>31-35 mins</b>	<b>36-40 mins</b>	<b>41-45 mins</b>	<b>46-50 mins</b>	<b>51-55 mins</b>	<b>56-60 mins</b>	<b>Total</b>
2	128 MB	1	1	-	2	1	5	2	-	12
3	256 MB	-	1	-	1	-	3	3	1	9
4	512 MB	1	-	2	2	-	5	2	-	12
5	1024 MB	-	1	-	1	1	4	2	1	10
6	1536 MB	-	-	1	1	-	-	-	-	2



<https://read.acloud.guru/how-long-does-aws-lambda-keep-your-idle-functions-around-before-a-cold-start-bf715d3b810>

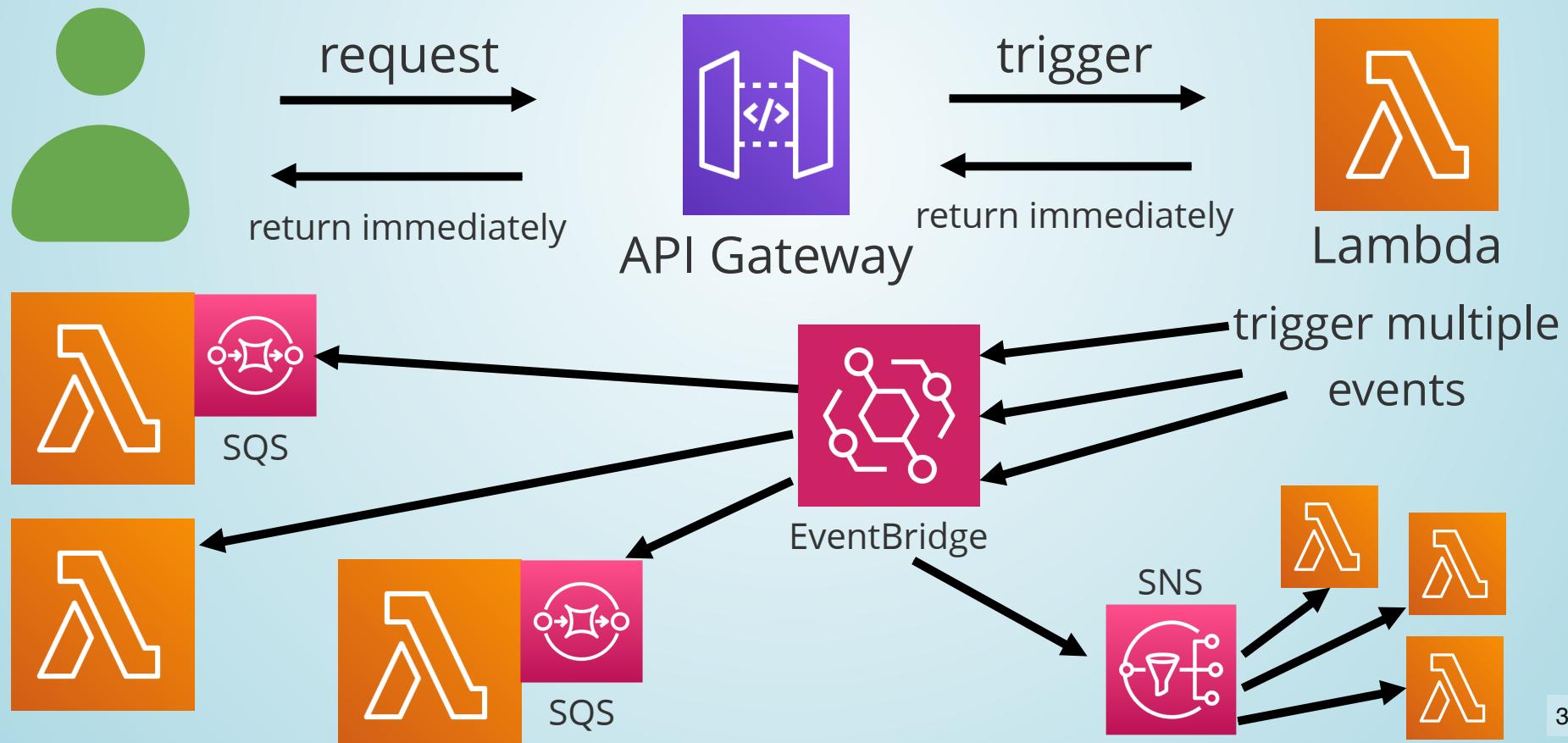


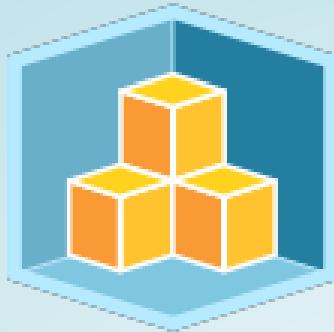
## Request-Response





## Event-Driven



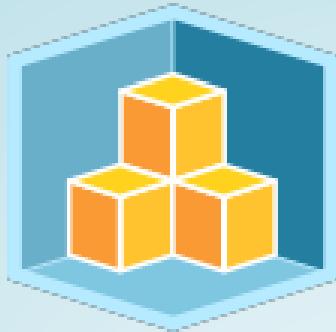


aws

Cloud  
Development  
Kit

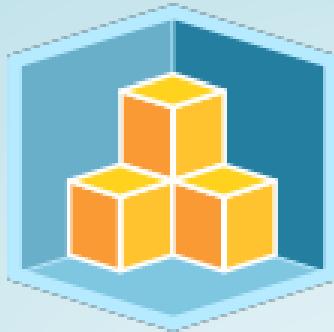
# Infrastructure as Code using AWS Cloud Development Kit

Infrastructure as *real* code



aws  
Cloud  
Development  
Kit

# Overview

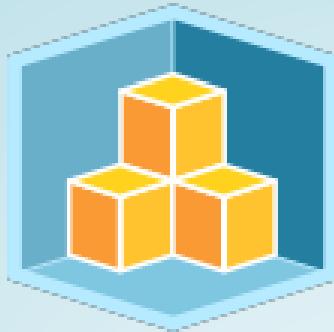


aws  
amazon

Cloud  
Development  
Kit

# Overview

- Introduction

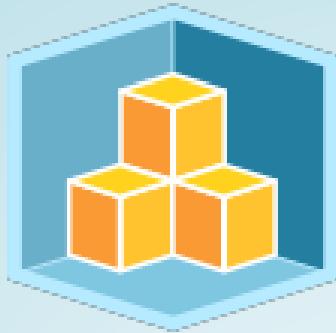


aws

Cloud  
Development  
Kit

# Overview

- Introduction
- Key Concepts

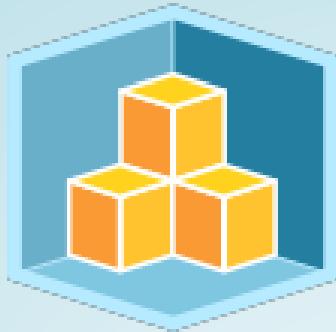


aws  
amazon

Cloud  
Development  
Kit

# Overview

- Introduction
- Key Concepts
- Getting started

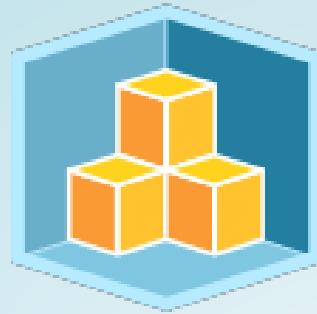


aws

Cloud  
Development  
Kit

# Overview

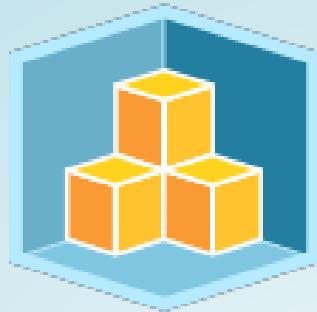
- Introduction
- Key Concepts
- Getting started
- Lab 1



aws

Cloud  
Development  
Kit

AWS CDK

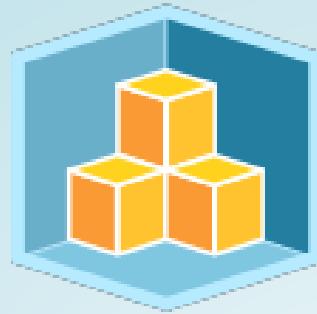


aws

Cloud  
Development  
Kit

## AWS CDK

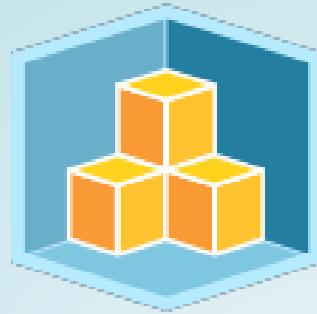
- Offers an imperative model for coding our infrastructure



aws  
Cloud  
Development  
Kit

## AWS CDK

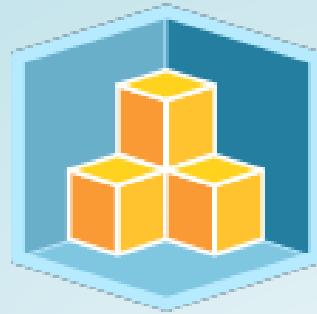
- Offers an imperative model for coding our infrastructure
- Can be coded using
  - JavaScript
  - TypeScript
  - Python
  - Java
  - C#
  - Go



aws  
Cloud  
Development  
Kit

## AWS CDK

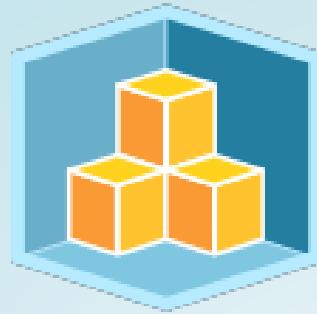
- Offers an imperative model for coding our infrastructure
- Can be coded using
  - JavaScript
  - TypeScript
  - Python
  - Java
  - C#
  - Go
- Developers write CDK code in a supported language



aws  
Cloud  
Development  
Kit

## AWS CDK

- Offers an imperative model for coding our infrastructure
- Can be coded using
  - JavaScript
  - TypeScript
  - Python
  - Java
  - C#
  - Go
- Developers write CDK code in a supported language
- That is "transpiled" into TypeScript

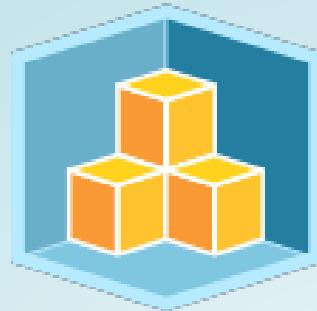


aws

Cloud  
Development  
Kit

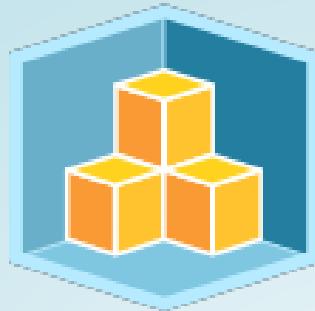
## AWS CDK

- Offers an imperative model for coding our infrastructure
- Can be coded using
  - JavaScript
  - TypeScript
  - Python
  - Java
  - C#
  - Go
- Developers write CDK code in a supported language
- That is "transpiled" into TypeScript
- That is "compiled" to CloudFormation Templates



aws  
Cloud  
Development  
Kit

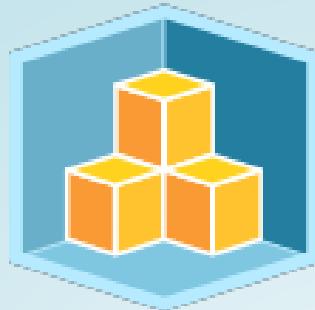
## AWS CDK Advantages



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

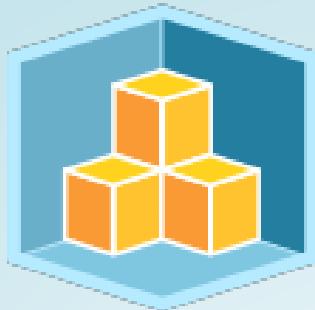
- Use logic (if statements, for-loops, etc)



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

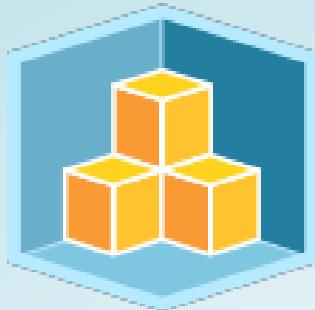
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

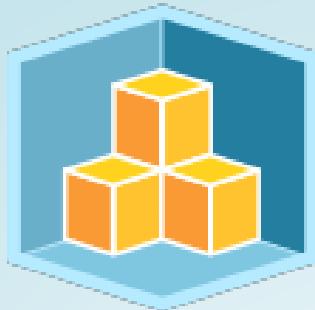
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

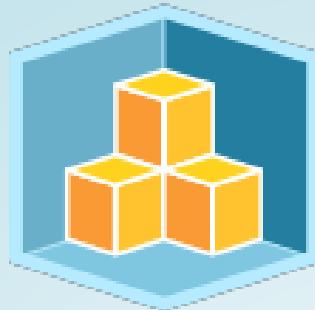
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community
- Organize your project into logical modules



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

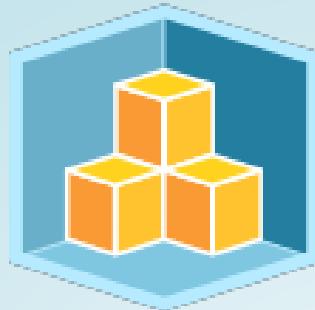
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community
- Organize your project into logical modules
- Share and reuse your infrastructure as a library



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

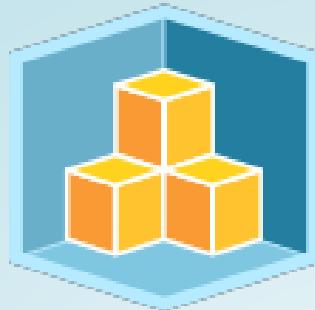
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community
- Organize your project into logical modules
- Share and reuse your infrastructure as a library
- Testing your infrastructure code using industry-standard protocols



aws  
Cloud  
Development  
Kit

## AWS CDK Advantages

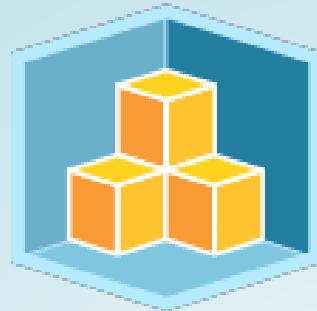
- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community
- Organize your project into logical modules
- Share and reuse your infrastructure as a library
- Testing your infrastructure code using industry-standard protocols
- Use your existing code review workflow



aws  
Cloud  
Development  
Kit

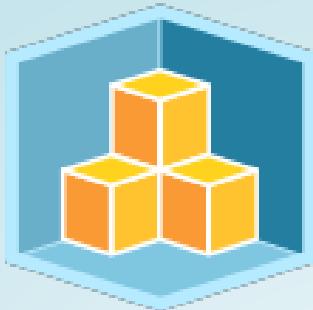
## AWS CDK Advantages

- Use logic (if statements, for-loops, etc)
- Use o-o techniques to create a model of your system
- Define high level abstractions, share them, and publish them to your team, company, or community
- Organize your project into logical modules
- Share and reuse your infrastructure as a library
- Testing your infrastructure code using industry-standard protocols
- Use your existing code review workflow
- Code completion within your IDE



aws  
Cloud  
Development  
Kit

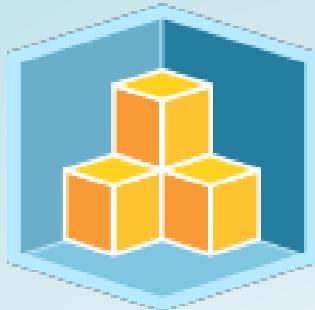
# Prerequisites



aws  
Cloud  
Development  
Kit

## Prerequisites

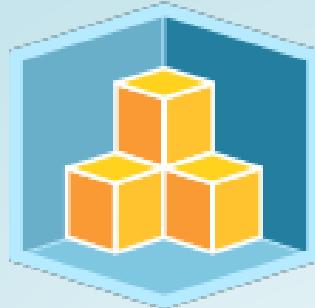
- General
  - Must have an AWS Account w/API access
  - Account needs all permissions you'll do with CDK (i.e **admin**)



aws  
Cloud  
Development  
Kit

## Prerequisites

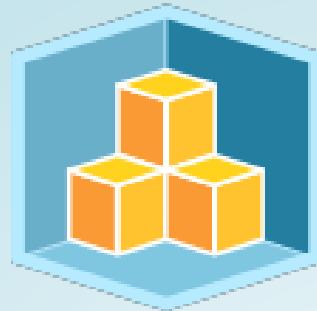
- General
  - Must have an AWS Account w/API access
  - Account needs all permissions you'll do with CDK (i.e **admin**)
- CLI Tools: Node.js >= 14.15.x



aws  
Cloud  
Development  
Kit

## Prerequisites

- General
  - Must have an AWS Account w/API access
  - Account needs all permissions you'll do with CDK (i.e **admin**)
- CLI Tools: Node.js >= 14.15.x
- Language
  - TypeScript: >= v3.8
  - JavaScript: None
  - Java: Maven 3.5+ (Can use Gradle though), JDK 8+
  - Python: >= 3.6
  - C#: .NET Core >= 3.1
  - Go: 1.18+



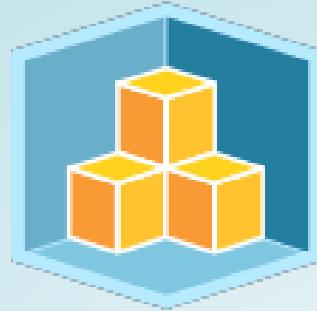
aws  
Cloud  
Development  
Kit

Project ▾

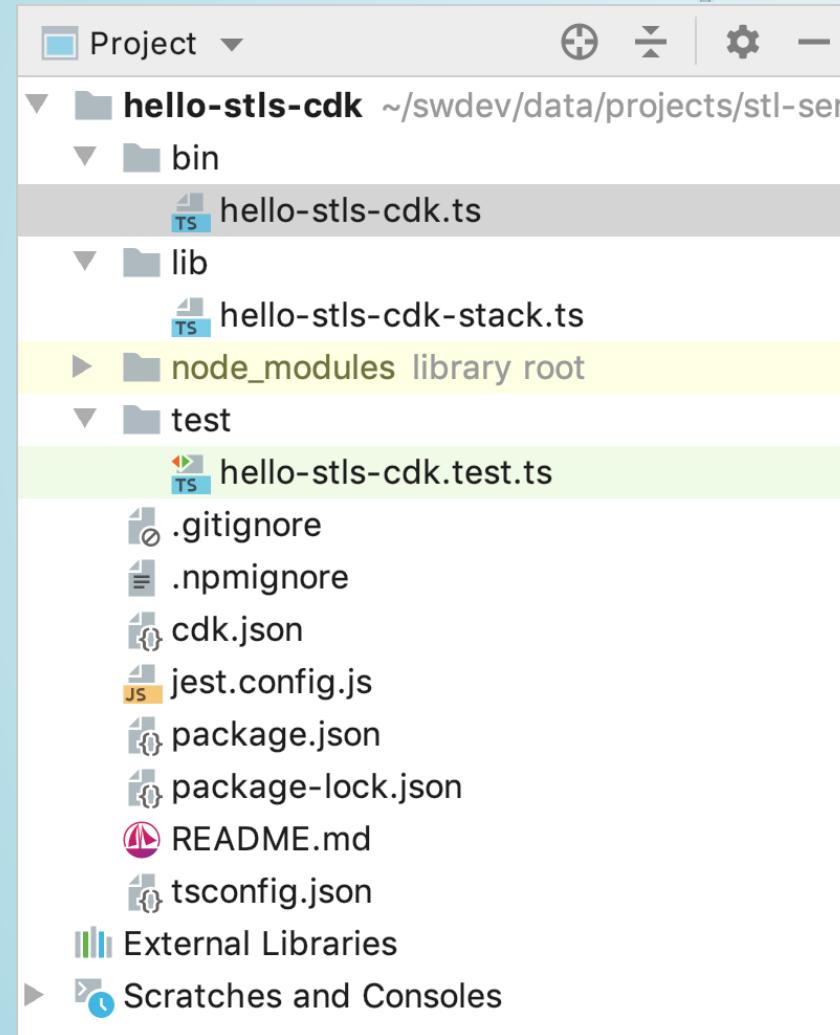
hello-stls-cdk ~/swdev/data/projects/stl-ser

- bin
- hello-stls-cdk.ts
- lib
- hello-stls-cdk-stack.ts
- node\_modules library root
- test
- hello-stls-cdk.test.ts
- .gitignore
- .npmignore
- cdk.json
- jest.config.js
- package.json
- package-lock.json
- README.md
- tsconfig.json
- External Libraries
- Scratches and Consoles

cdk init project result

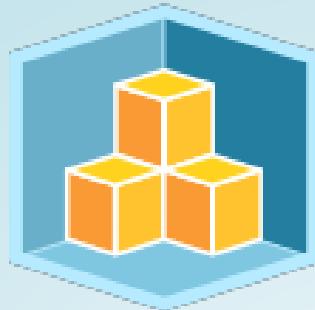


aws  
Cloud  
Development  
Kit



## cdk init project result

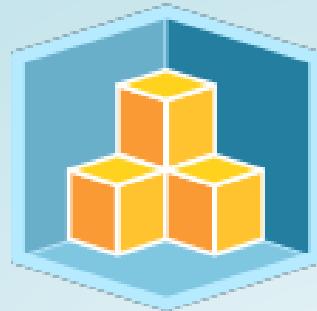
Note: A **.git** repo is created. You may want to delete it if adding a cdk folder in your business app and want to commit cdk code as part of that repo.



aws  
Cloud  
Development  
Kit

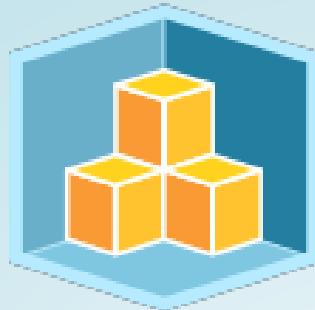
## AWS CDK Key Concepts

- Constructs
- Stacks
- Environments
- Apps
- Tokens
- Identifiers
- Resources
- Tags
- Assets
- Permissions
- Context
- Aspects
- Escape Hatches



aws  
Cloud  
Development  
Kit

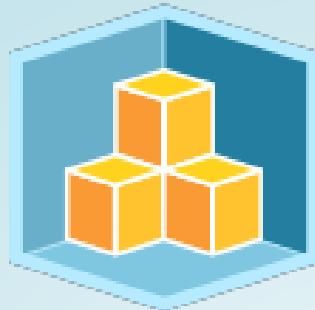
# CDK Constructs



aws  
Cloud  
Development  
Kit

## CDK Constructs

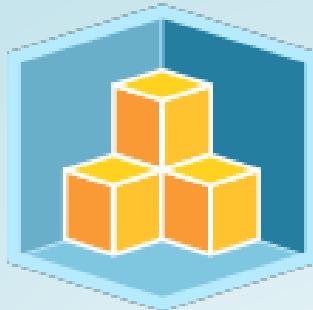
- Fundamental building blocks representing a cloud component



aws  
Cloud  
Development  
Kit

## CDK Constructs

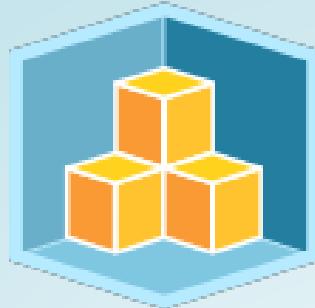
- Fundamental building blocks representing a cloud component
- Multiple levels of abstractions



aws  
Cloud  
Development  
Kit

## CDK Constructs

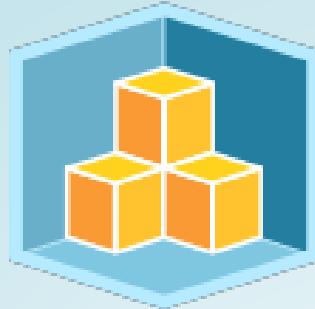
- Fundamental building blocks representing a cloud component
- Multiple levels of abstractions
- Level 1: Low-level CloudFormation resources; e.g.
  - `s3.CfnBucket` represents a AWS::S3::Bucket CFN Resource
  - Must configure completely



aws  
Cloud  
Development  
Kit

## CDK Constructs

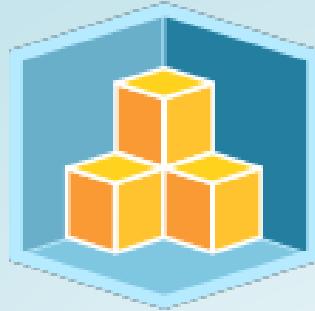
- Fundamental building blocks representing a cloud component
- Multiple levels of abstractions
- Level 1: Low-level CloudFormation resources; e.g.
  - `s3.CfnBucket` represents a AWS::S3::Bucket CFN Resource
  - Must configure completely
- Level 2: Medium-level, intention-based abstraction
  - Encapsulates details, boilerplate, & "glue" logic
  - Bakes in best practices; i.e. least privilege roles



aws  
Cloud  
Development  
Kit

## CDK Constructs

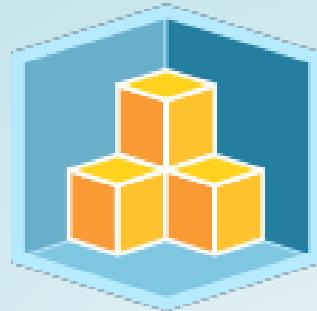
- Fundamental building blocks representing a cloud component
- Multiple levels of abstractions
- Level 1: Low-level CloudFormation resources; e.g.
  - `s3.CfnBucket` represents a AWS::S3::Bucket CFN Resource
  - Must configure completely
- Level 2: Medium-level, intention-based abstraction
  - Encapsulates details, boilerplate, & "glue" logic
  - Bakes in best practices; i.e. least privilege roles
- Level 3: High-level Patterns often multiple types of resources; e.g.
  - `aws-ecs-patterns.ApplicationLoadBalancedFargateService`



aws  
Cloud  
Development  
Kit

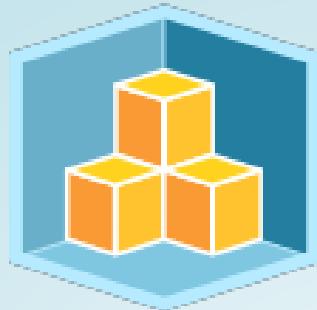
## Construct Example (ts)

```
1 // The bucket name is not "MyVersionedBucket"
2 // The bucket name will be auto-generated
3 new s3.Bucket(this, 'MyVersionedBucket', {
4   versioned: true
5 })
6
7 // class EncryptedBucket extends s3.Bucket {
8 new s3.Bucket(this, 'MyEncryptedBucket', {
9   encryption: s3.BucketEncryption.KMS,
10  websiteIndexDocument: 'index.html'
11 })
```



aws  
Cloud  
Development  
Kit

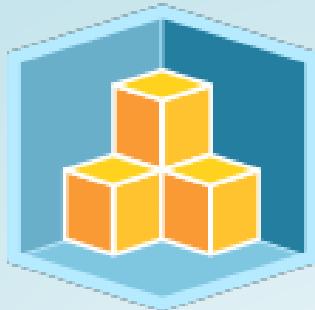
# CDK Stacks



aws  
Cloud  
Development  
Kit

## CDK Stacks

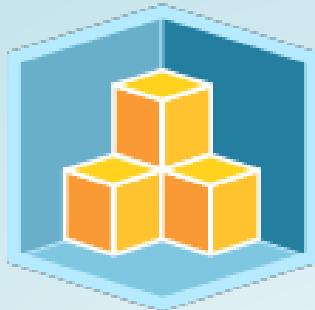
- A *stack* is a unit of deployment



aws  
Cloud  
Development  
Kit

## CDK Stacks

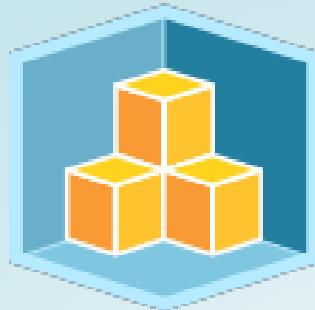
- A *stack* is a unit of deployment
- CDK stacks have same limitations as CloudFormation stacks



aws  
Cloud  
Development  
Kit

## CDK Stacks

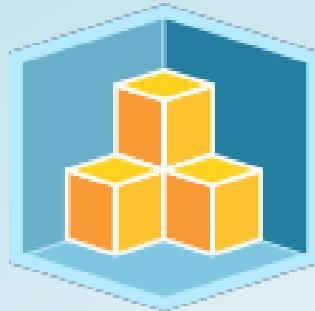
- A *stack* is a unit of deployment
- CDK stacks have same limitations as CloudFormation stacks
- A CDK stack has one or more Constructs



aws  
Cloud  
Development  
Kit

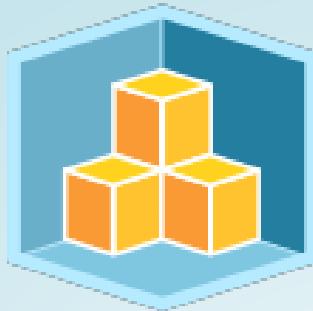
## CDK Stacks

- A *stack* is a unit of deployment
- CDK stacks have same limitations as CloudFormation stacks
- A CDK stack has one or more Constructs
- The same stack can be reused across *environments*



aws  
Cloud  
Development  
Kit

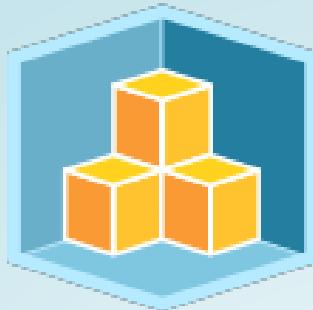
# CDK Environments



aws  
Cloud  
Development  
Kit

## CDK Environments

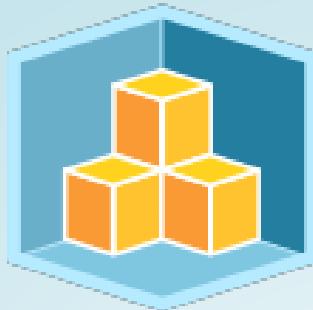
- A stack is implicitly or explicitly associated with one or more environments



aws  
Cloud  
Development  
Kit

## CDK Environments

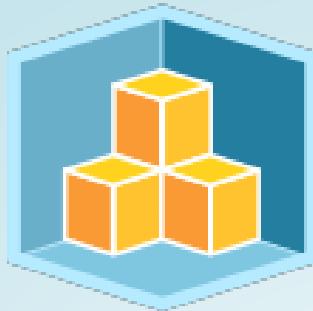
- A stack is implicitly or explicitly associated with one or more environments
- An environment is the target AWS account and region in which stack(s) need to be deployed



aws  
Cloud  
Development  
Kit

## CDK Environments

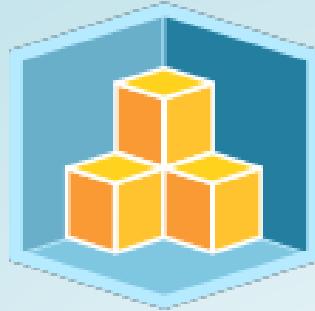
- A stack is implicitly or explicitly associated with one or more environments
- An environment is the target AWS account and region in which stack(s) need to be deployed
- If you don't specify an environment when you define a stack, the stack is *environment agnostic*



aws  
Cloud  
Development  
Kit

## CDK Environments

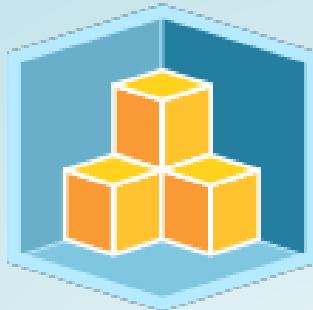
- A stack is implicitly or explicitly associated with one or more environments
- An environment is the target AWS account and region in which stack(s) need to be deployed
- If you don't specify an environment when you define a stack, the stack is *environment agnostic*
- The environment used for environment agnostic stacks is determined at deploy time



aws  
Cloud  
Development  
Kit

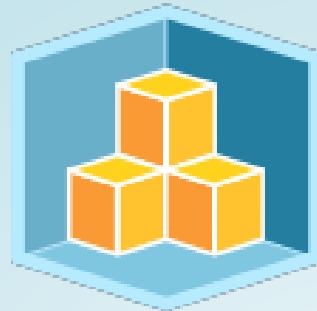
## CDK Environments

- A stack is implicitly or explicitly associated with one or more environments
- An environment is the target AWS account and region in which stack(s) need to be deployed
- If you don't specify an environment when you define a stack, the stack is *environment agnostic*
- The environment used for environment agnostic stacks is determined at deploy time
- Best production practice: Specify the environment for each stack



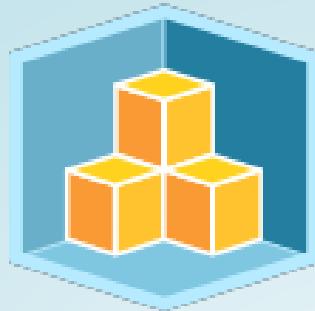
# Specifying Environments for Stacks

```
1 // development stack where each developer uses their own account and region
2 new MyDevStack(this, 'dev-stack', {
3   env: {
4     account: process.env.CDK_DEFAULT_ACCOUNT,
5     region: process.env.CDK_DEFAULT_REGION
6   }
7 })
8
9 // // production example
10 const envEU = { account: '2383838383', region: 'eu-west-1' }
11 const envUSA = { account: '8373873873', region: 'us-west-2' }
12
13 new MyFirstStack(this, 'first-stack-us', { env: envUSA, encryption: false })
14 new MyFirstStack(this, 'first-stack-eu', { env: envEU, encryption: true })
```



aws  
Cloud  
Development  
Kit

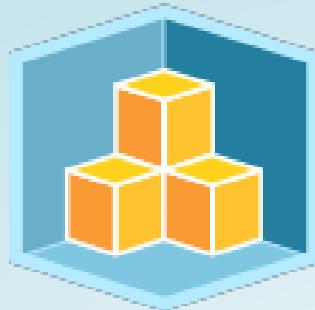
## CDK Apps



aws  
Cloud  
Development  
Kit

## CDK Apps

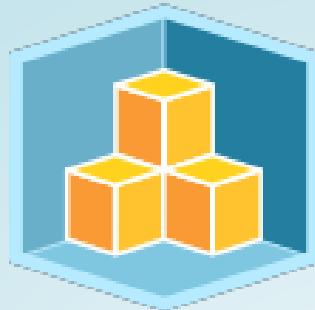
- A CDK App defines a *scope* a stack operates in



aws  
Cloud  
Development  
Kit

## CDK Apps

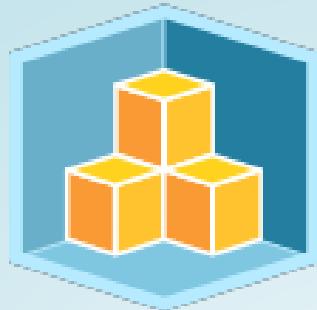
- A CDK App defines a *scope* a stack operates in
- It gives the Stack some context



aws  
Cloud  
Development  
Kit

## CDK Apps

- A CDK App defines a *scope* a stack operates in
- It gives the Stack some context
- An App has a definite lifecycle that is followed from instantiation to deployment

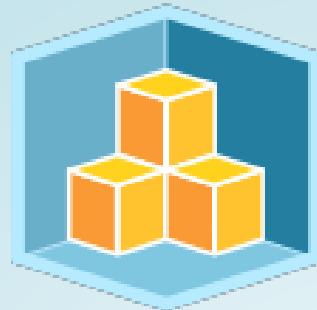


aws  
Cloud  
Development  
Kit

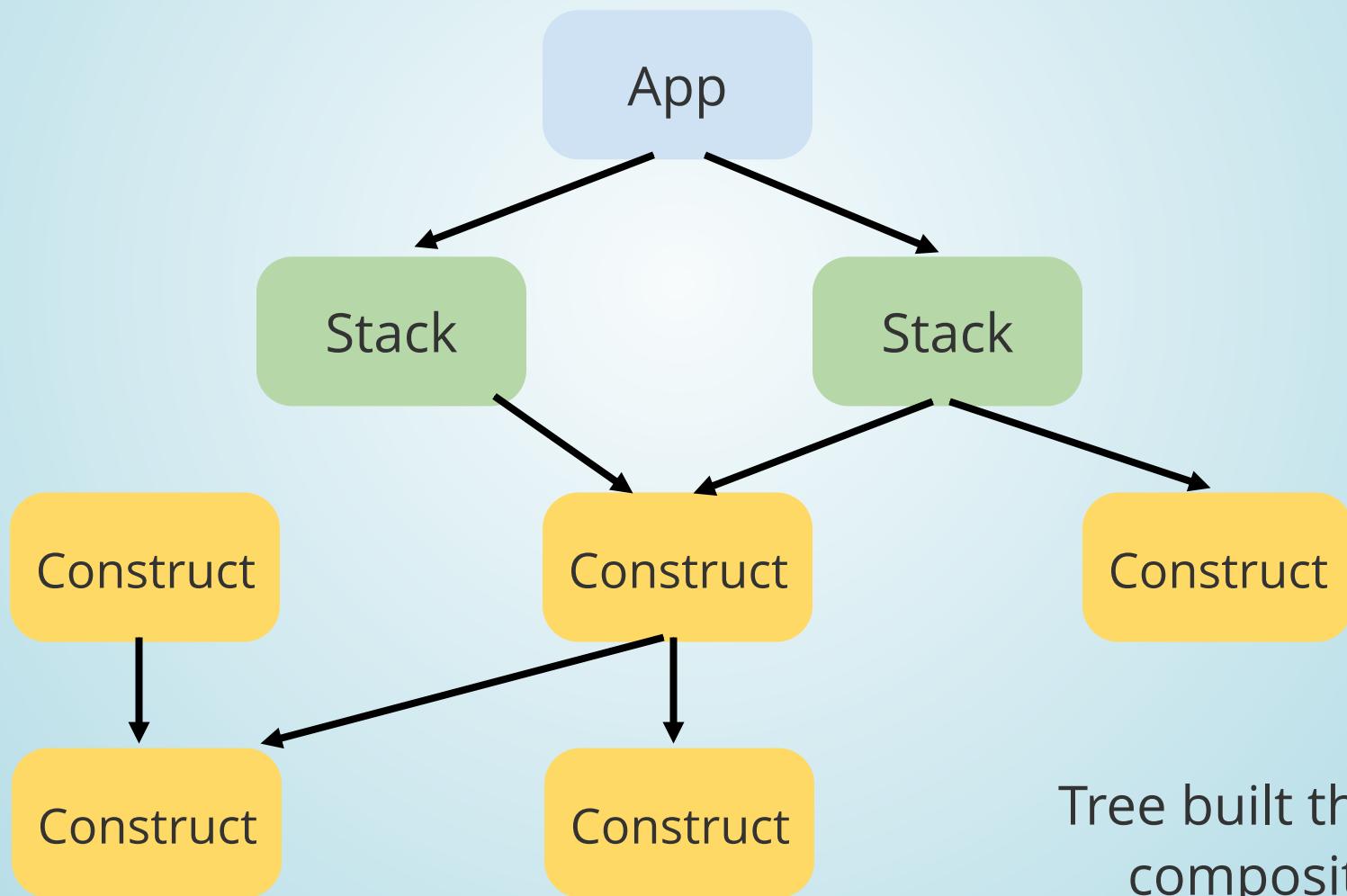
## CDK Apps

- A CDK App defines a *scope* a stack operates in
- It gives the Stack some context
- An App has a definite lifecycle that is followed from instantiation to deployment

```
1 class MyApp extends App {  
2     constructor() {  
3         new MyFirstStack(this, 'hello-cdk')  
4     }  
5 }  
6  
7 new MyApp().synth()
```

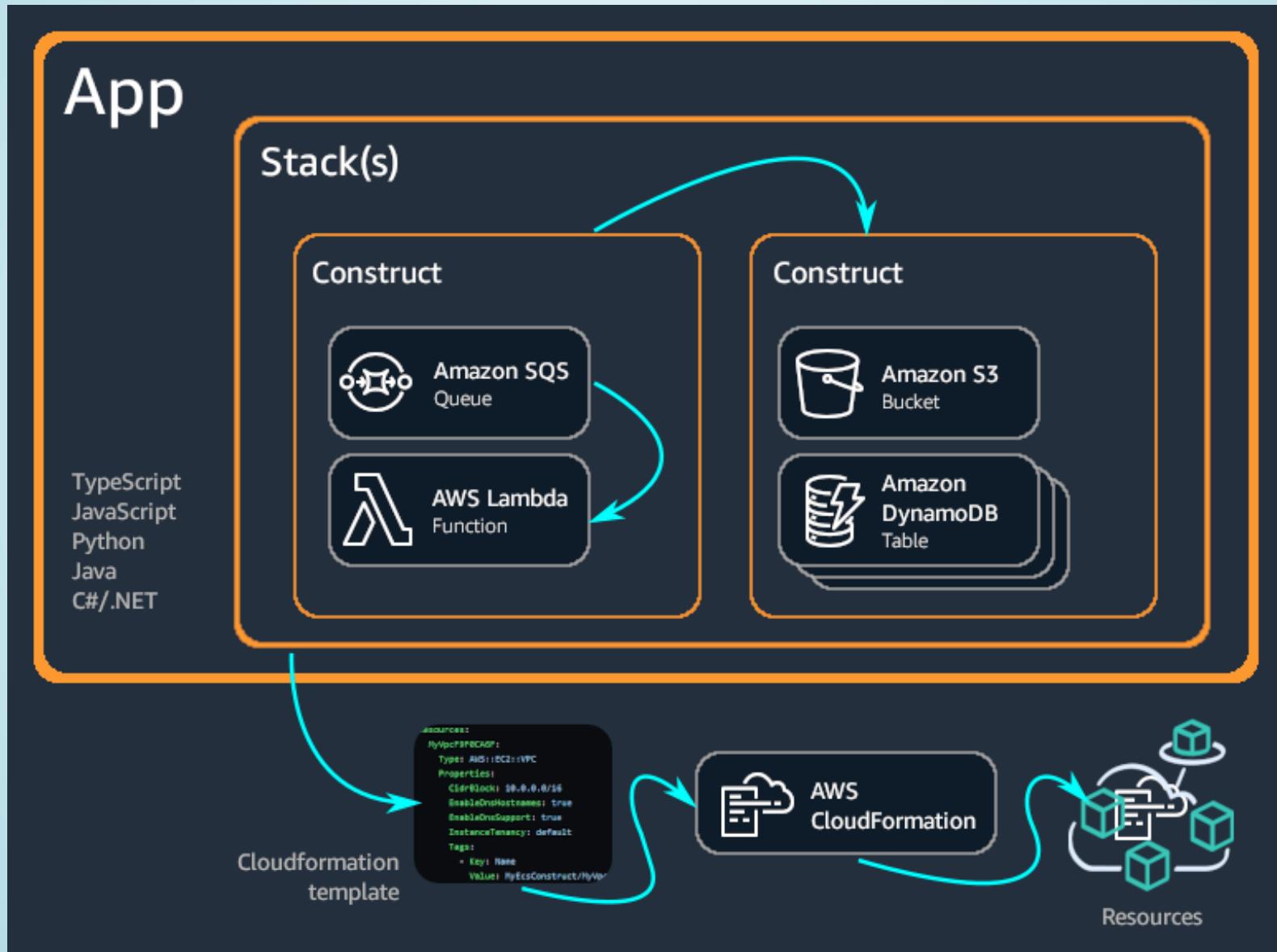


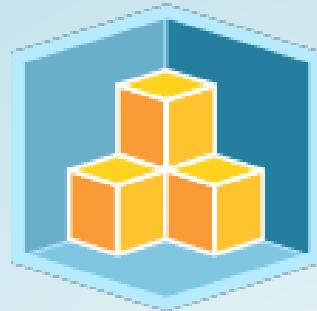
aws  
Cloud  
Development  
Kit



Tree built through  
composition

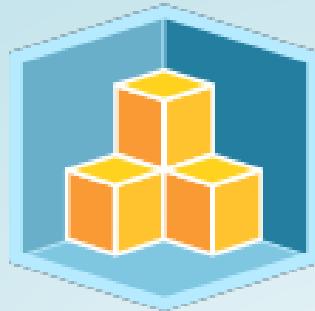
# CDK Apps





aws  
Cloud  
Development  
Kit

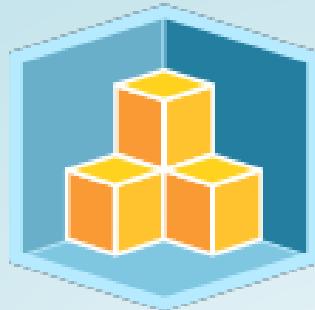
# CDK App Lifecycle



aws  
Cloud  
Development  
Kit

## CDK App Lifecycle

1. Construction: Instantiate Construct tree and Stacks

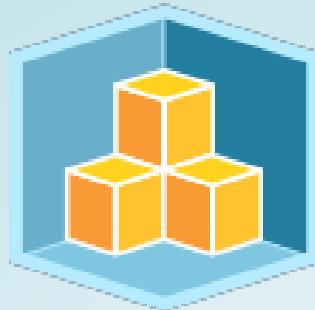


aws  
Cloud  
Development  
Kit

## CDK App Lifecycle

1. Construction: Instantiate Construct tree and Stacks
2. Preparation

- Construct's `prepare()` method called
- It's not recommended for Constructs to implement *prepare*



aws  
Cloud  
Development  
Kit

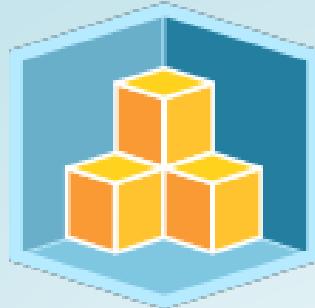
## CDK App Lifecycle

1. Construction: Instantiate Construct tree and Stacks

2. Preparation

- Construct's `prepare()` method called
- It's not recommended for Constructs to implement *prepare*

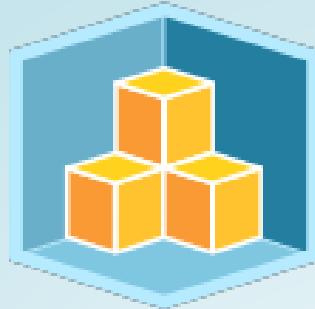
3. Validation: Construct's `validate()` method called



aws  
Cloud  
Development  
Kit

## CDK App Lifecycle

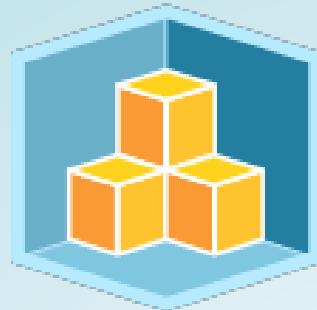
1. Construction: Instantiate Construct tree and Stacks
2. Preparation
  - Construct's `prepare()` method called
  - It's not recommended for Constructs to implement *prepare*
3. Validation: Construct's `validate()` method called
4. Synthesis: Construct's `synthesize()` method called



aws  
Cloud  
Development  
Kit

## CDK App Lifecycle

1. Construction: Instantiate Construct tree and Stacks
2. Preparation
  - Construct's `prepare()` method called
  - It's not recommended for Constructs to implement *prepare*
3. Validation: Construct's `validate()` method called
4. Synthesis: Construct's `synthesize()` method called
5. Deployment: Deploys assets and starts CloudFormation deployment



aws  
Cloud  
Development  
Kit

## Lab 1 - A taste of CDK



# AWS Lambda



# Lambda Basics



# Lambda Basics

- No servers to manage



# Lambda Basics

- No servers to manage
- Continuous scaling



# Lambda Basics

- No servers to manage
- Continuous scaling
- Cost-optimized with millisecond metering



# Lambda Basics

- No servers to manage
- Continuous scaling
- Cost-optimized with millisecond metering
- Consistent performance at any scale



# Lambda Basics

- No servers to manage
- Continuous scaling
- Cost-optimized with millisecond metering
- Consistent performance at any scale
- 1,000,000 free invocations a month



# Lambda as Part of a Serverless Platform



## Lambda as Part of a Serverless Platform

You focus on your app. AWS worries about the infrastructure



## Lambda as Part of a Serverless Platform

You focus on your app. AWS worries about the infrastructure

- Orchestration & State Management



## Lambda as Part of a Serverless Platform

You focus on your app. AWS worries about the infrastructure

- Orchestration & State Management
- Reliability & Performance



## Lambda as Part of a Serverless Platform

You focus on your app. AWS worries about the infrastructure

- Orchestration & State Management
- Reliability & Performance
- Security & Access Control



## Lambda as Part of a Serverless Platform

You focus on your app. AWS worries about the infrastructure

- Orchestration & State Management
- Reliability & Performance
- Security & Access Control
- Global Scale



# Sample Use Cases

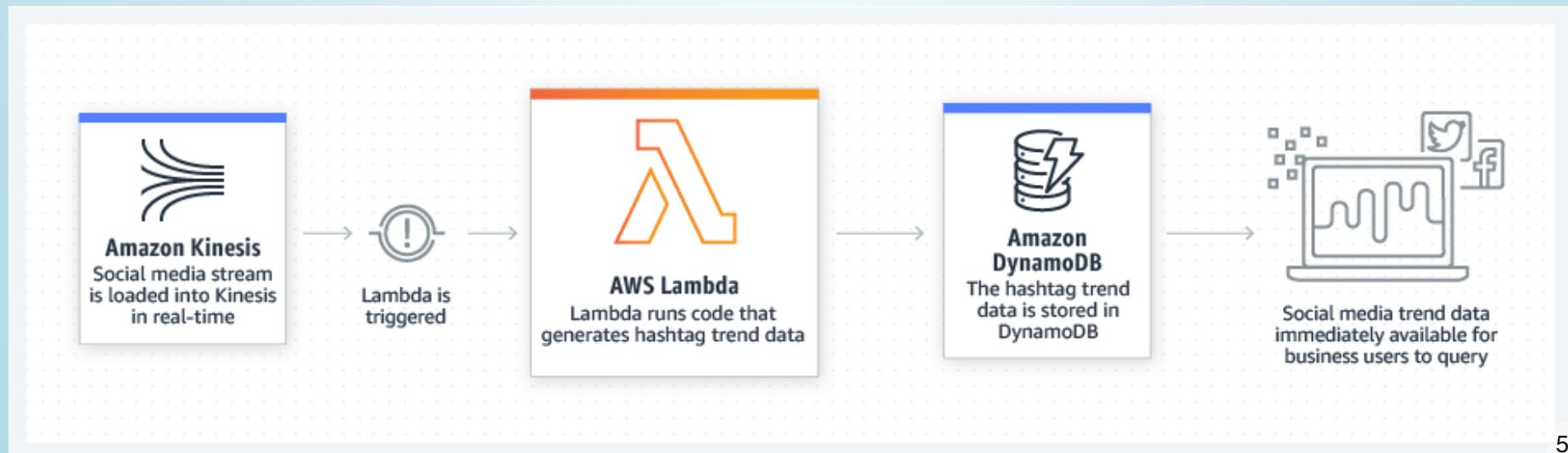
## File Transformations





# Sample Use Cases

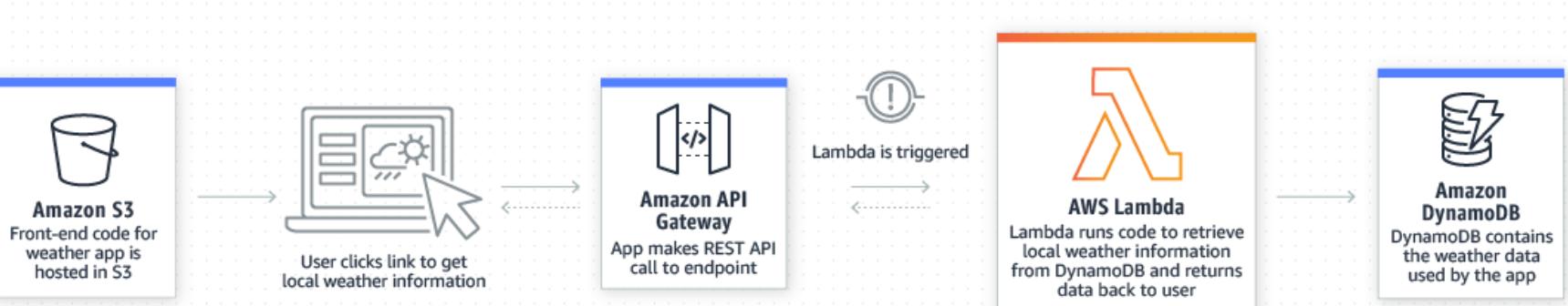
## Streaming Data Analytics





# Sample Use Cases

## Web App API





# Lambda Cold Start Performance



# Lambda Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request



# Lambda Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request
- Different languages and frameworks have different cold start penalties



# Lambda Cold Start Performance

- Cold start time is the time between when a request is received and the handler starts processing the request
- Different languages and frameworks have different cold start penalties
- Available memory dramatically affects cold start performance



# 2014 - re:Invent

## Lambda Concepts

The most important Lambda concept is the Lambda function, or function for short. You write your functions in [Node.js](#) (an event-driven, server side implementation of [JavaScript](#)).

You upload your code

```
1 console.log('Loading event');
2 var aws = require('aws-sdk');
3 var s3 = new aws.S3({apiVersion: '2006-03-01'});
4
5 exports.handler = function(event, context) {
6     console.log('Received event:');
7     console.log(event);
8     // Get the object from the event and show its
9     s3.getObject({Bucket:event.Bucket, Key:event.
10        function(err,data) {
11            if (err) {
12                console.log('error getting object');
```



## 2014 - re:Invent

### Lambda Concepts

The most important Lambda concept is the Lambda function, or function for short. You write your functions in [Node.js](#) (an event-driven, server side implementation of [JavaScript](#)).

You upload your code

```
1 console.log('Loading event');
2 var aws = require('aws-sdk');
3 var s3 = new aws.S3({apiVersion: '2006-03-01'});
4
5 exports.handler = function(event, context) {
6     console.log('Received event:');
7     console.log(event);
8     // Get the object from the event and show its
9     s3.getObject({Bucket:event.Bucket, Key:event.
10        function(err,data) {
11            if (err) {
12                console.log('error getting object');
```

- Javascript / Node.js only



## 2014 - re:Invent

### Lambda Concepts

The most important Lambda concept is the Lambda function, or function for short. You write your functions in [Node.js](#) (an event-driven, server side implementation of [JavaScript](#)).

You upload your code

```
1 console.log('Loading event');
2 var aws = require('aws-sdk');
3 var s3 = new aws.S3({apiVersion: '2006-03-01'});
4
5 exports.handler = function(event, context) {
6     console.log('Received event:');
7     console.log(event);
8     // Get the object from the event and show its
9     s3.getObject({Bucket:event.Bucket, Key:event.
10        function(err,data) {
11            if (err) {
12                console.log('error getting object');
```

- Javascript / Node.js only
- 60-second execution limit



## 2014 - re:Invent

### Lambda Concepts

The most important Lambda concept is the Lambda function, or function for short. You write your functions in [Node.js](#) (an event-driven, server side implementation of [JavaScript](#)).

```
1 console.log('Loading event');
2 var aws = require('aws-sdk');
3 var s3 = new aws.S3({apiVersion: '2006-03-01'});
4
5 exports.handler = function(event, context) {
6     console.log('Received event:');
7     console.log(event);
8     // Get the object from the event and show its
9     s3.getObject({Bucket:event.Bucket, Key:event.
10        function(err,data) {
11            if (err) {
12                console.log('error getting object');
```

You upload your code

- Javascript / Node.js only
- 60-second execution limit
- Three event sources: S3; DynamoDB; Kinesis Streams



2015



2015

- Python support added



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)



# 2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail
  - SES



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail
  - SES
  - Cognito



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail
  - SES
  - Cognito
- **API Gateway**



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail
  - SES
  - Cognito
  - **API Gateway**
  - SNS



2015

- Python support added
- Function execution duration increased from 60 seconds to 300 seconds
- Function Versioning and Aliasing
- Scheduled Functions (Cron) (initially via console only)
- More service integrations
  - CloudFormation
  - Simple Workflow (SWF)
  - CloudTrail
  - SES
  - Cognito
  - **API Gateway**
  - SNS
  - Cloudwatch Logs



2016



2016

- Environment Variables for Lambdas



2016

- Environment Variables for Lambdas
- DotNet support (C#)



# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ



2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)



2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas



# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas
- AWS X-Ray Preview



# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas
- AWS X-Ray Preview
- RDS/Aurora Lambda Triggers



# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas
- AWS X-Ray Preview
- RDS/Aurora Lambda Triggers
- Lambdas in VPCs



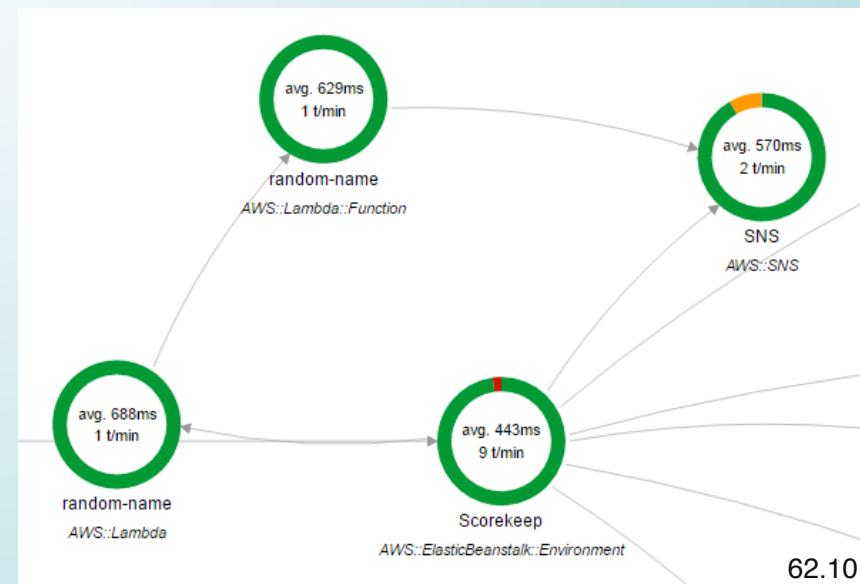
# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas
- AWS X-Ray Preview
- RDS/Aurora Lambda Triggers
- Lambdas in VPCs
- Lambda@Edge Preview



# 2016

- Environment Variables for Lambdas
- DotNet support (C#)
- Per-function DLQ
- AWS SAM (Serverless Application Model)
- AWS Step Functions with Lambdas
- AWS X-Ray Preview
- RDS/Aurora Lambda Triggers
- Lambdas in VPCs
- Lambda@Edge Preview





2017



2017

- Doubled Lambda memory limits to 3GB
  - This yields increases in CPU and network



2017

- Doubled Lambda memory limits to 3GB
  - This yields increases in CPU and network
- Traffic shifting to allow canary and Blue/Green Deployments



2017

- Doubled Lambda memory limits to 3GB
  - This yields increases in CPU and network
- Traffic shifting to allow canary and Blue/Green Deployments
- Added ability to set Lambda concurrency limits



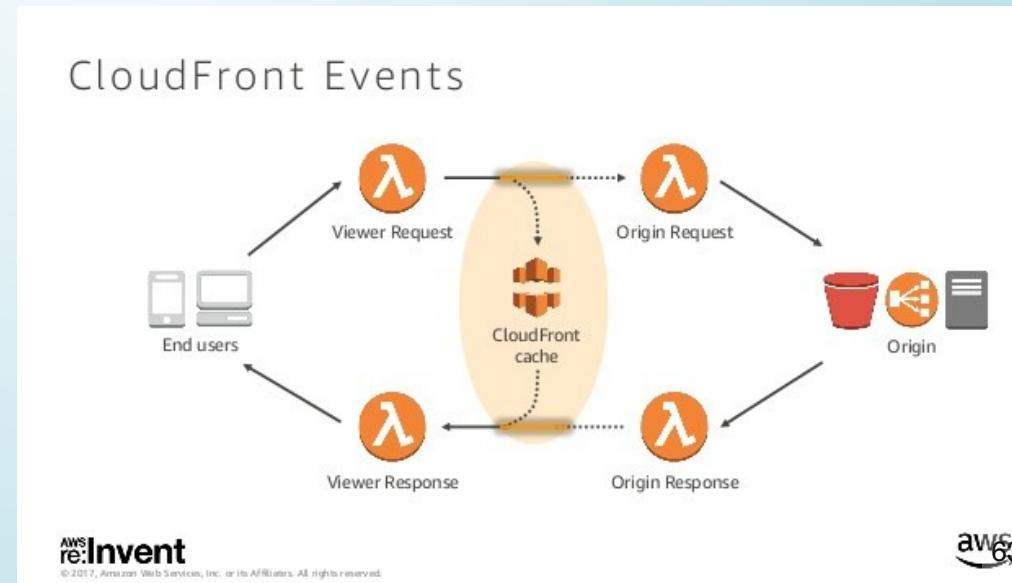
2017

- Doubled Lambda memory limits to 3GB
  - This yields increases in CPU and network
- Traffic shifting to allow canary and Blue/Green Deployments
- Added ability to set Lambda concurrency limits
- Lambda@Edge GA



# 2017

- Doubled Lambda memory limits to 3GB
  - This yields increases in CPU and network
- Traffic shifting to allow canary and Blue/Green Deployments
- Added ability to set Lambda concurrency limits
- Lambda@Edge GA





2018



2018

- re:Invent 2018 was often called, The Serverless re:Invent



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers
- Lambda Custom Runtimes



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers
- Lambda Custom Runtimes
- ALB Support for Lambdas



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers
- Lambda Custom Runtimes
- ALB Support for Lambdas
- AURORA Serverless



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers
- Lambda Custom Runtimes
- ALB Support for Lambdas
- AURORA Serverless
- WebSocket Support for API Gateway



2018

- re:Invent 2018 was often called, The Serverless re:Invent
- Lambda Layers
- Lambda Custom Runtimes
- ALB Support for Lambdas
- AURORA Serverless
- WebSocket Support for API Gateway
- Ruby added as a supported language



2018



2018

- Lambda Layers
  - Shareable artifacts between Lambdas
  - Analogous to Docker image layers
  - Limit 5
  - Layers count in 250MB Lambda unzip



2018

- Lambda Layers
  - Shareable artifacts between Lambdas
  - Analogous to Docker image layers
  - Limit 5
  - Layers count in 250MB Lambda unzip
- Lambda custom runtimes
  - Bring your own language!
  - AWS provided runtimes for C++ and Rust



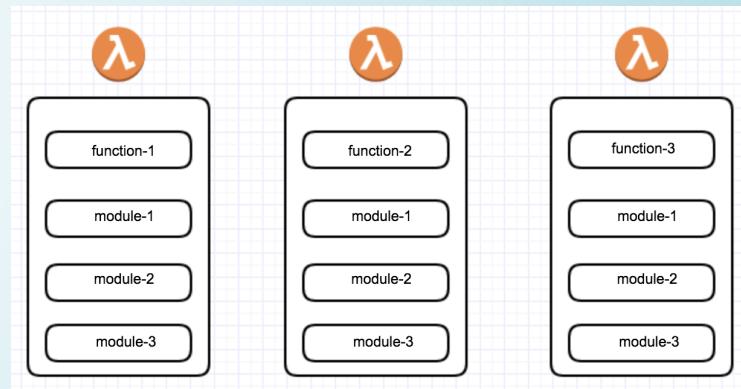
2018

- Lambda Layers

- Shareable artifacts between Lambdas
- Analogous to Docker image layers
- Limit 5
- Layers count in 250MB Lambda unzip

- Lambda custom runtimes

- Bring your own language!
- AWS provided runtimes for C++ and Rust





2019



2019

- RDS Proxy instead of Connection-per-Lambda



2019

- RDS Proxy instead of Connection-per-Lambda
- Provisioned Concurrency
  - (more cold start mitigation)



2019

- RDS Proxy instead of Connection-per-Lambda
- Provisioned Concurrency
  - (more cold start mitigation)
- Lambda Destinations



# 2019

- RDS Proxy instead of Connection-per-Lambda
- Provisioned Concurrency
  - (more cold start mitigation)
- Lambda Destinations
- New Lambda event sources
  - SQS FIFO Queues
  - EventBridge



2019

# Provisioned Concurrency



2019

## Provisioned Concurrency

- Cold starts are a big issue in Lambda functions
  - Especially for those written in C# and Java



2019

## Provisioned Concurrency

- Cold starts are a big issue in Lambda functions
  - Especially for those written in C# and Java
- Provisioned concurrency is a way to keep and pre-warm Lambdas using auto-scaling



2019

## Provisioned Concurrency

- Cold starts are a big issue in Lambda functions
  - Especially for those written in C# and Java
- Provisioned concurrency is a way to keep and pre-warm Lambdas using auto-scaling
- It definitely is a step away from serverless
  - You pay by the hour whether you need it or not



2019

## Provisioned Concurrency

- Cold starts are a big issue in Lambda functions
  - Especially for those written in C# and Java
- Provisioned concurrency is a way to keep and pre-warm Lambdas using auto-scaling
- It definitely is a step away from serverless
  - You pay by the hour whether you need it or not
- There are better solutions in JVM space around GraalVM
  - Micronaut, Quarkus today; Spring Native



2019

# Lambda Destinations



2019

## Lambda Destinations

- Used with asynchronous function executions



2019

## Lambda Destinations

- Used with asynchronous function executions
- When executing `async`, you get an immediate response
  - Did the function succeed?
  - Did it fail?



2019

## Lambda Destinations

- Used with asynchronous function executions
- When executing `async`, you get an immediate response
  - Did the function succeed?
  - Did it fail?
- Now you can send a message to success / error destination



2019

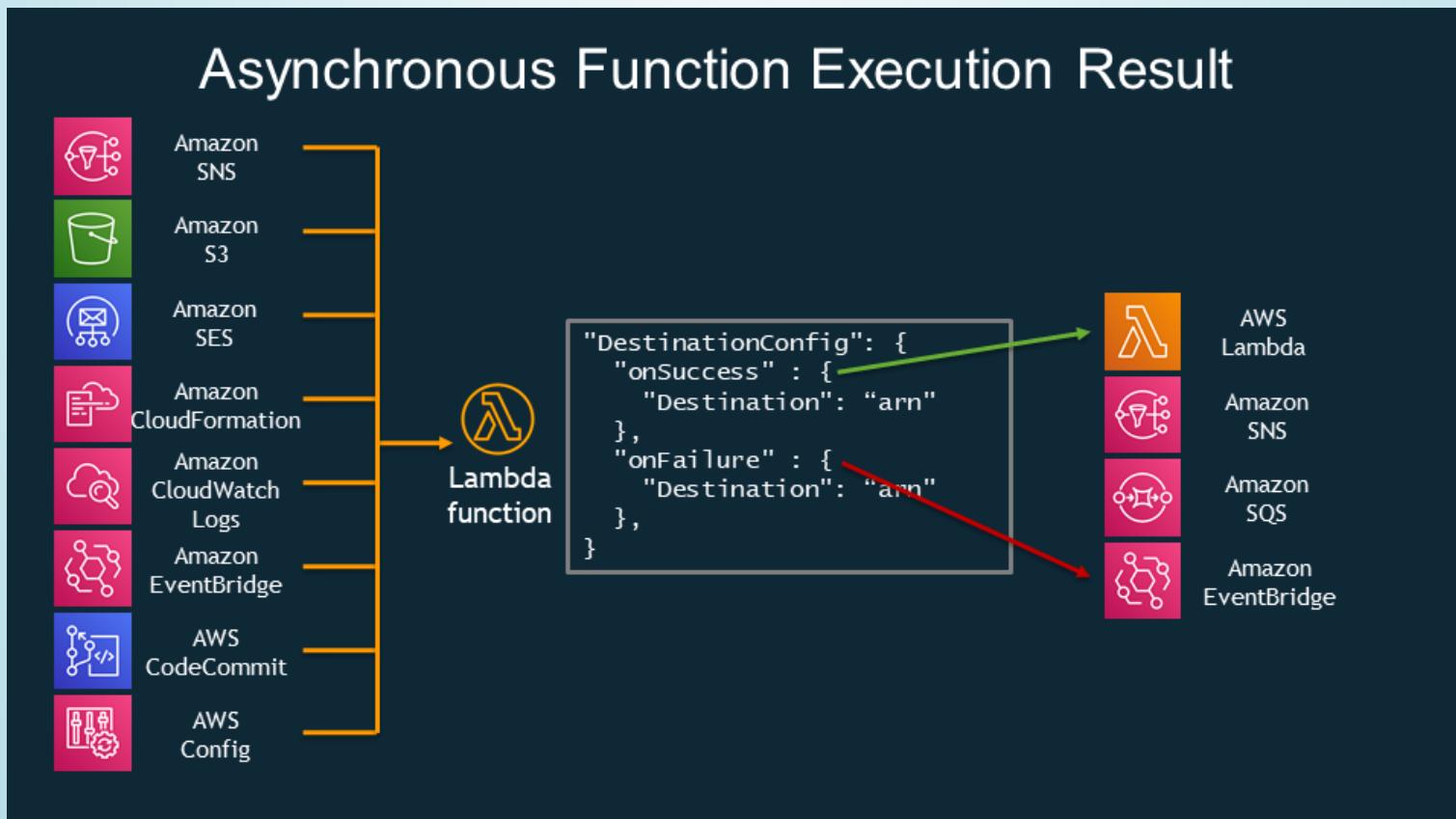
## Lambda Destinations

- Used with asynchronous function executions
- When executing async, you get an immediate response
  - Did the function succeed?
  - Did it fail?
- Now you can send a message to success / error destination
- Much lighter weight than using Step Functions to choreograph



2019

## Lambda Destinations





2020

# GraalVM + Lambda Containers



2020

## GraalVM + Lambda Containers

- With GraalVM AOT compiler, we can turn a JVM app into an OS executable



2020

## GraalVM + Lambda Containers

- With GraalVM AOT compiler, we can turn a JVM app into an OS executable
- With Lambda supporting containers, we can package that into a container image



2020

## GraalVM + Lambda Containers

- With GraalVM AOT compiler, we can turn a JVM app into an OS executable
- With Lambda supporting containers, we can package that into a container image
- We can achieve <100ms application starts



2020

## GraalVM + Lambda Containers

- With GraalVM AOT compiler, we can turn a JVM app into an OS executable
- With Lambda supporting containers, we can package that into a container image
- We can achieve <100ms application starts
- Three frameworks exist today
  - Micronaut (Object Computing, Inc.)
  - Quarkus (Red Hat)
  - Spring Native



2021

## Lambda Updates



2021

## Lambda Updates

- Cross-account ECR (Elastic Container Registry) Access



2021

## Lambda Updates

- Cross-account ECR (Elastic Container Registry) Access
- Partial batch responses for SQS



2021

## Lambda Updates

- Cross-account ECR (Elastic Container Registry) Access
- Partial batch responses for SQS
- Event filtering for SQS, DynamoDB, Kinesis



2022

# Lambda Updates



2022

## Lambda Updates

- AWS Inspect now scans Lambda



2022

## Lambda Updates

- AWS Inspect now scans Lambda
- SnapStart (Java-only)



2022

## Lambda Updates

- AWS Inspect now scans Lambda
- SnapStart (Java-only)
- AWS Application Composer



# Application Composer

Connected mode

learning-serverless/template.yaml

Menu ▾

List Resources

Search for a resource

- API Gateway
- Cognito UserPool
- Cognito UserPoolClient
- DynamoDB Table
- EventBridge Event rule
- EventBridge Schedule
- Kinesis Stream
- Lambda Function
- Lambda Layer
- S3 Bucket
- SNS Topic

Canvas Template Arrange

API Compose

API-Gateway API

Items

ListItems

CreateItem

GetItem

UpdateItem

DeleteItem

Items

This screenshot shows the AWS Application Composer interface for creating serverless applications. The top navigation bar includes 'Connected mode', the file path 'learning-serverless/template.yaml', and a 'Menu'. Below the header, there are tabs for 'List', 'Resources' (which is selected), 'Canvas', 'Template', and 'Arrange'. A search bar is also present. On the left, a sidebar lists various AWS services with icons: API Gateway, Cognito UserPool, Cognito UserPoolClient, DynamoDB Table, EventBridge Event rule, EventBridge Schedule, Kinesis Stream, Lambda Function, Lambda Layer, S3 Bucket, and SNS Topic. The main canvas area displays a diagram of an API endpoint named 'API-Gateway API'. This API has five methods: GET /items, POST /items, GET /items/{id}, PUT /items/{id}, and DELETE /items/{id}. Each method is connected to a corresponding Lambda Function (ListItems, CreateItem, GetItem, UpdateItem, DeleteItem). These Lambda Functions are grouped under a 'API Compose' container. An external 'Items' resource from a 'DynamoDB Table' is also connected to the Lambda Functions. The 'Template' tab is currently active, showing the template.yaml file content.

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

73



# Lab 2: Lambdas



# Synchronous Triggers

- Elastic Load Balancing (Application Load Balancer)
- Amazon Cognito
- Amazon Lex
- Amazon Alexa
- Amazon API Gateway
- Amazon CloudFront (Lambda@Edge)
- Amazon Kinesis Data Firehose



# Asynchronous Triggers

- [Amazon Simple Storage Service](#)
- [Amazon Simple Notification Service](#)
- [Amazon Simple Email Service](#)
- [AWS CloudFormation](#)
- [Amazon CloudWatch Logs](#)
- [Amazon CloudWatch Events](#)
- [AWS CodeCommit](#)
- [AWS Config](#)

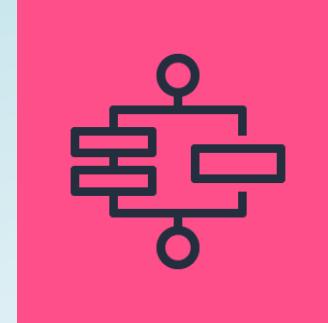


## Polling Triggers

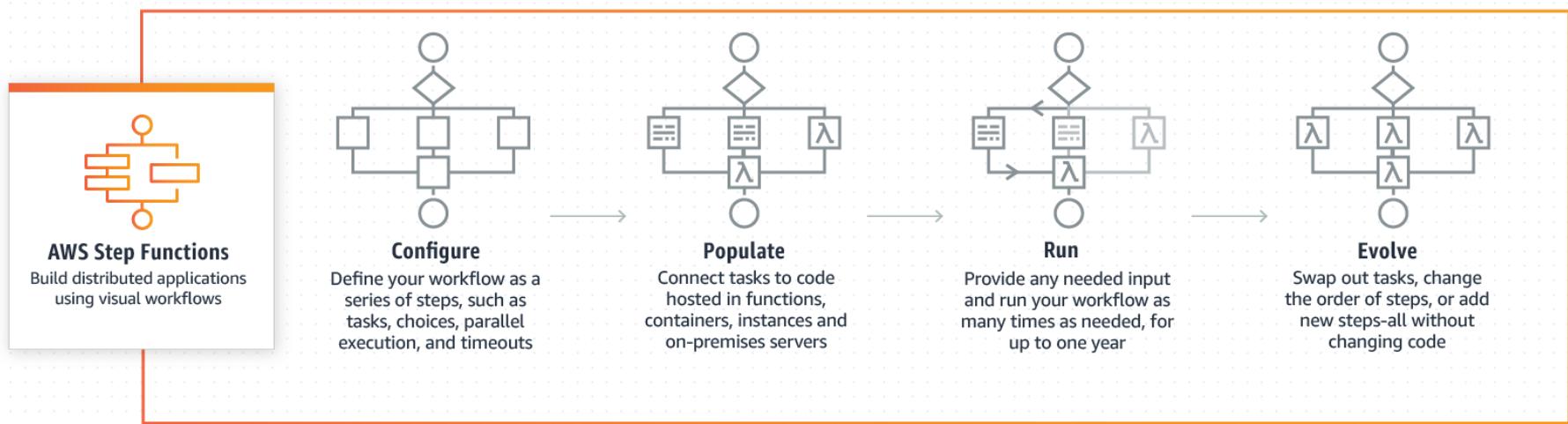
- [Amazon Kinesis](#)
- [Amazon SQS](#)
- [Amazon DynamoDB Streams](#)



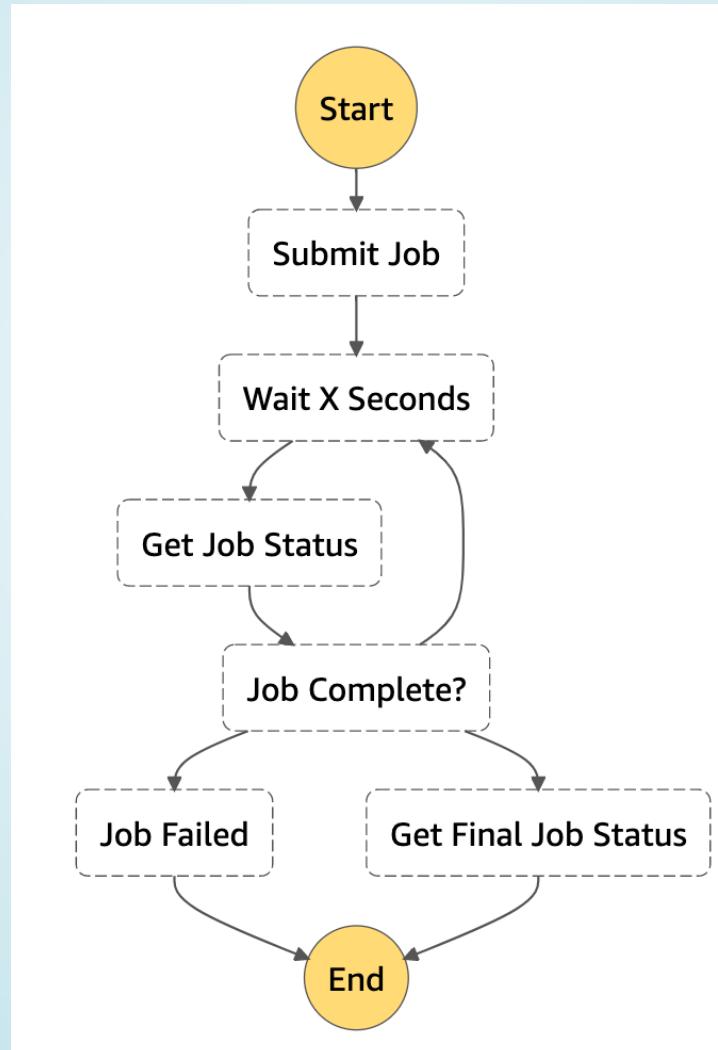
# Lab 3: Lambda Triggers



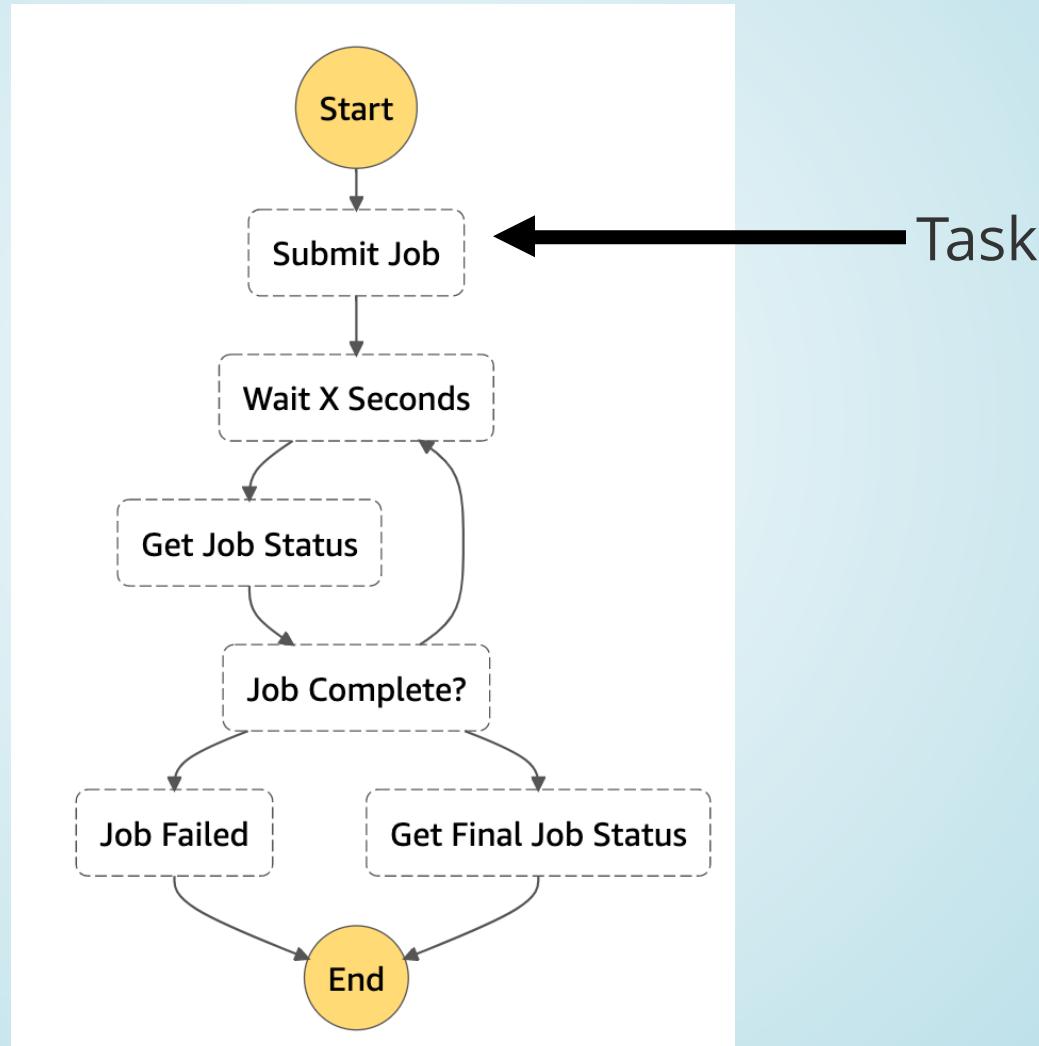
# Step Functions



# Example - Batch Job Workflow

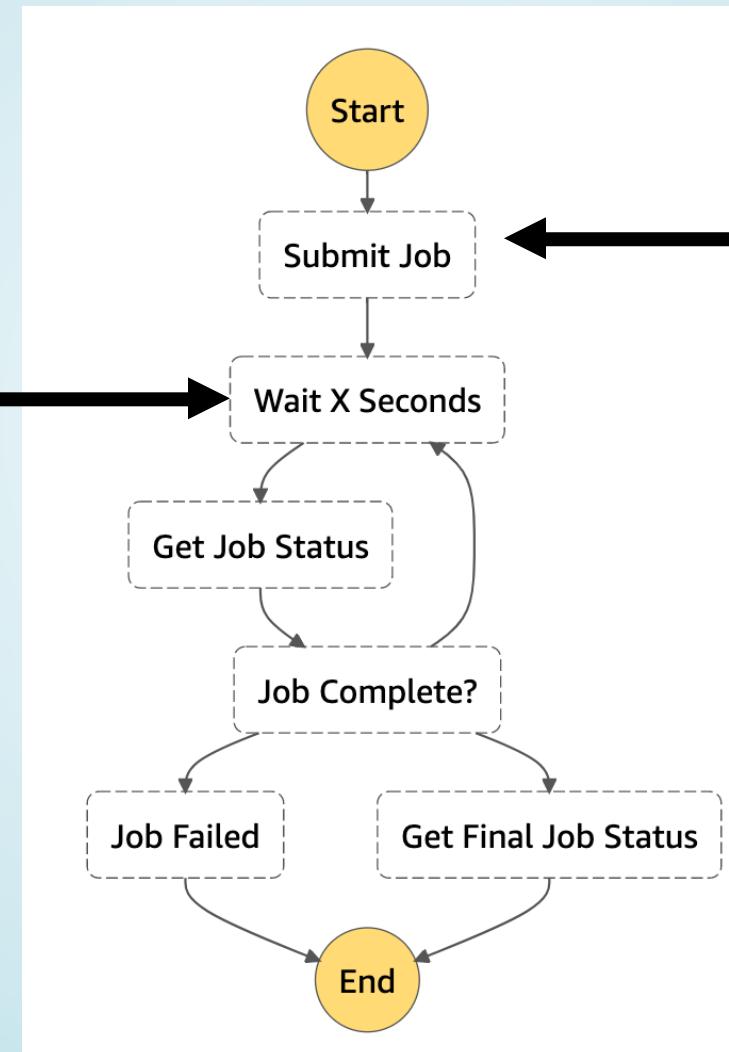


# Example - Batch Job Workflow



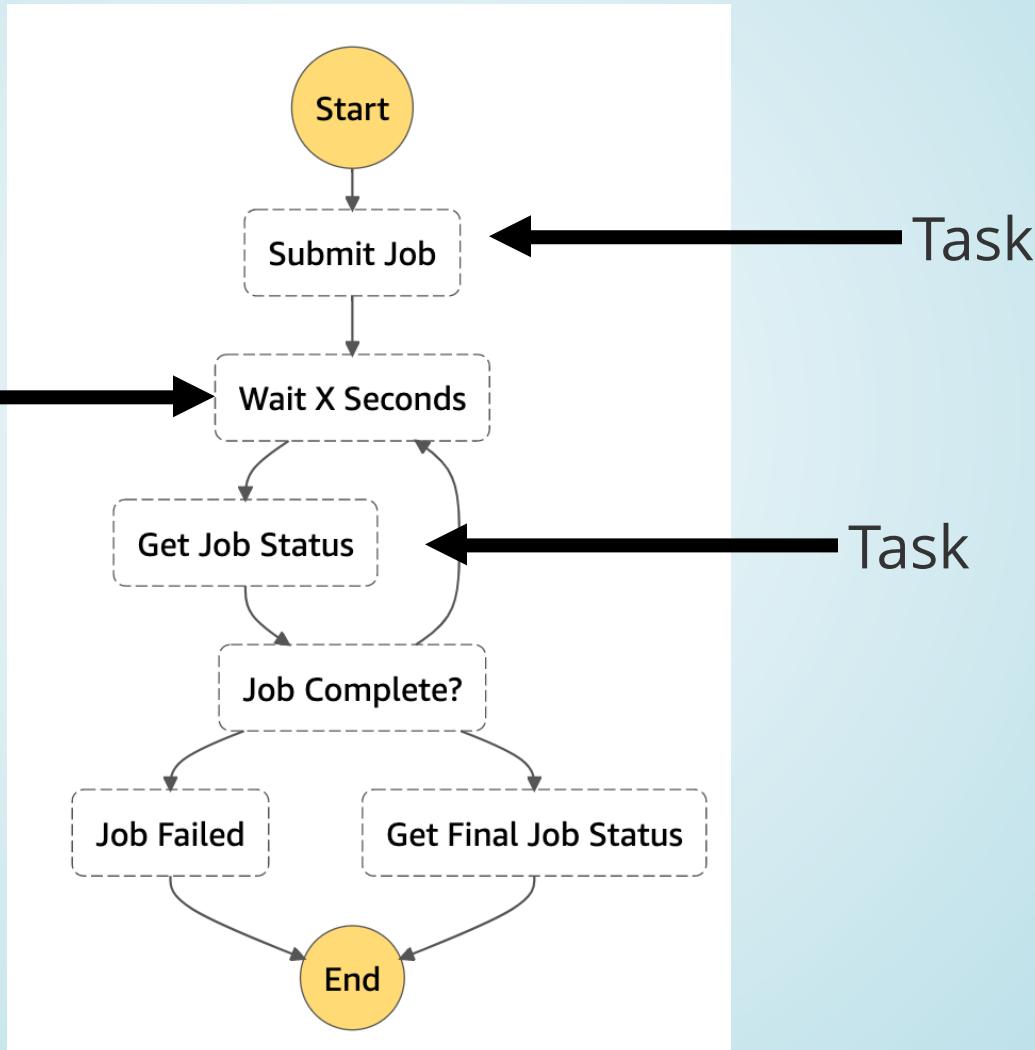
# Example - Batch Job Workflow

Wait ————— Task

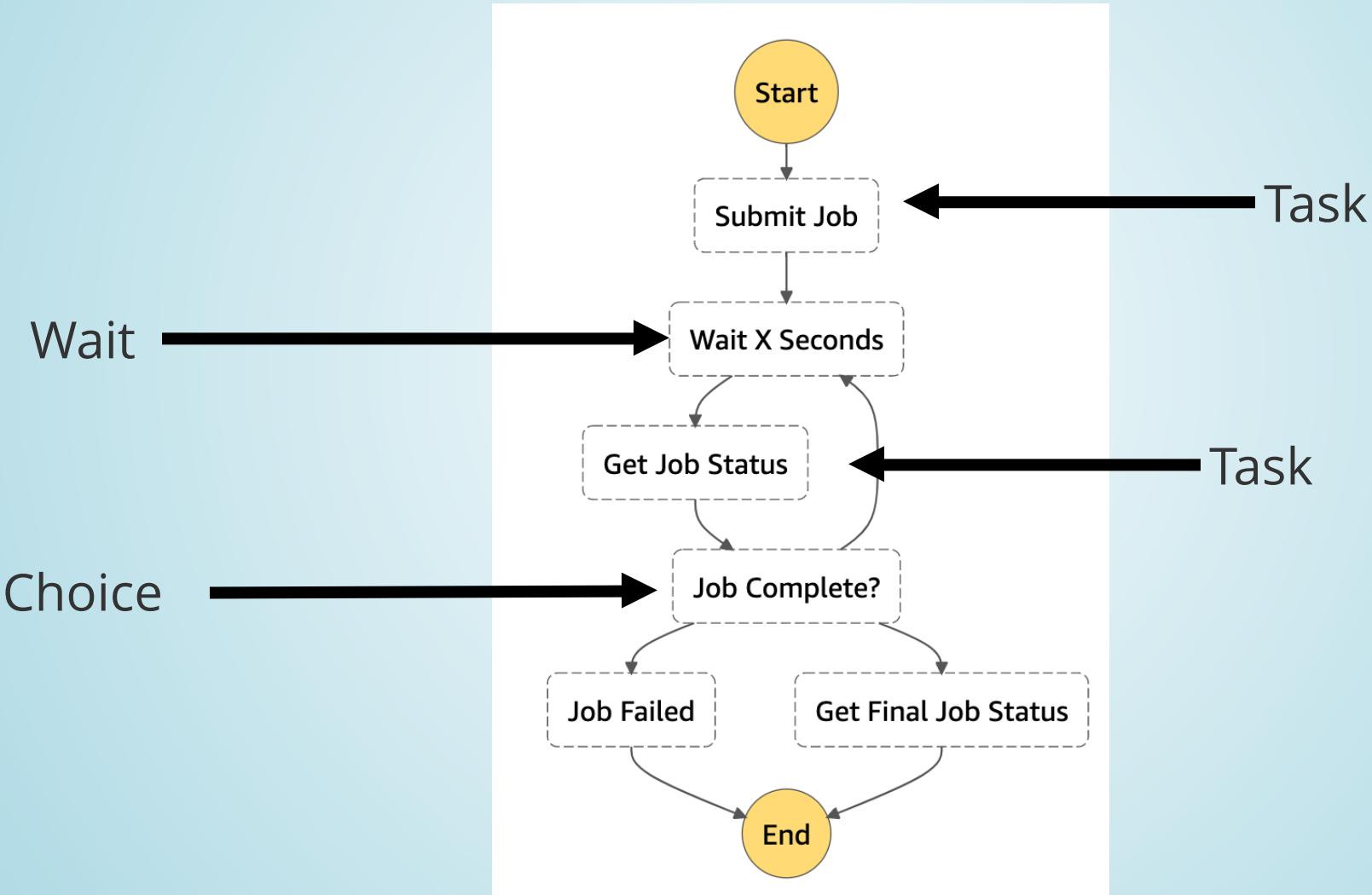


# Example - Batch Job Workflow

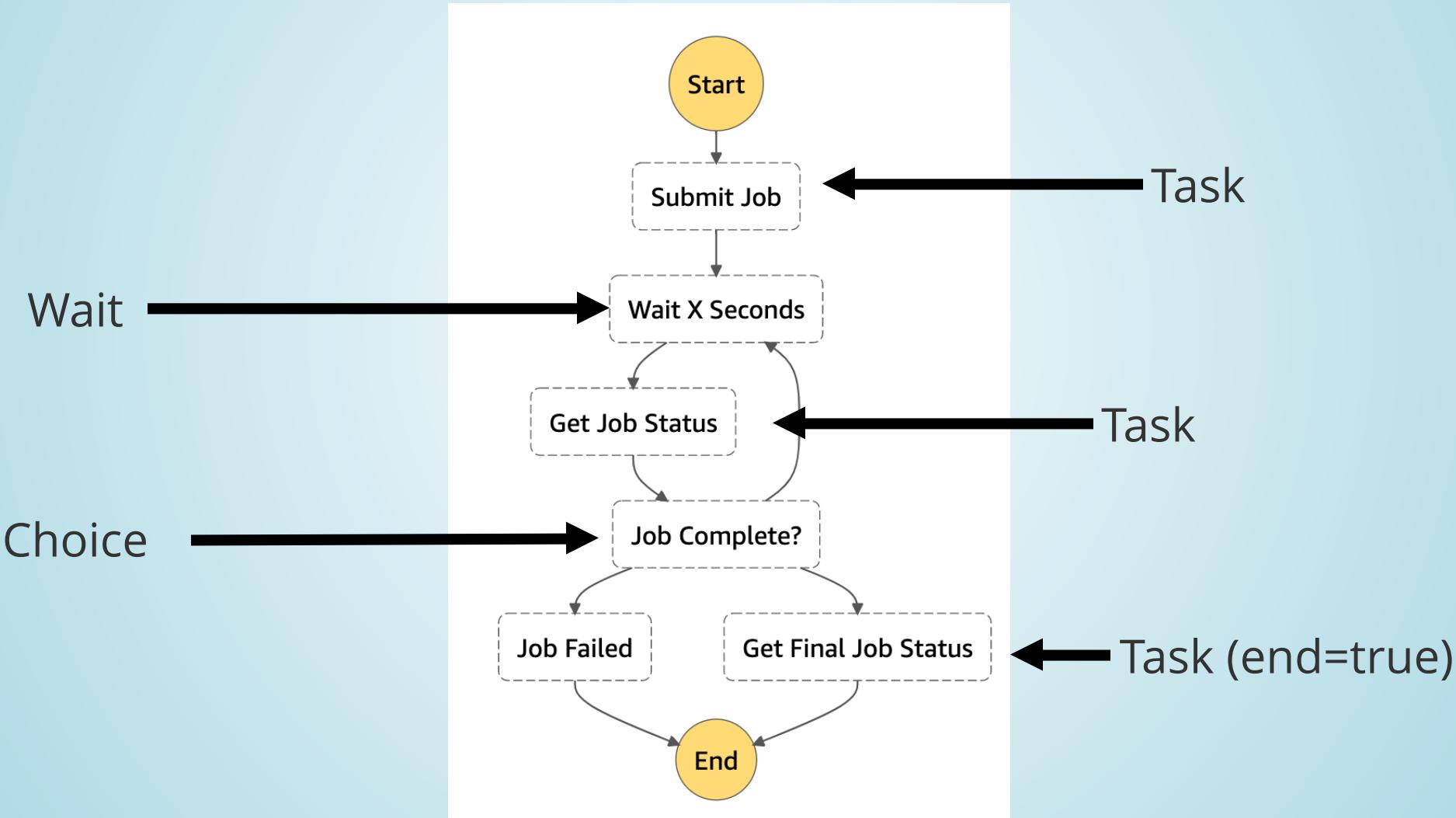
Wait



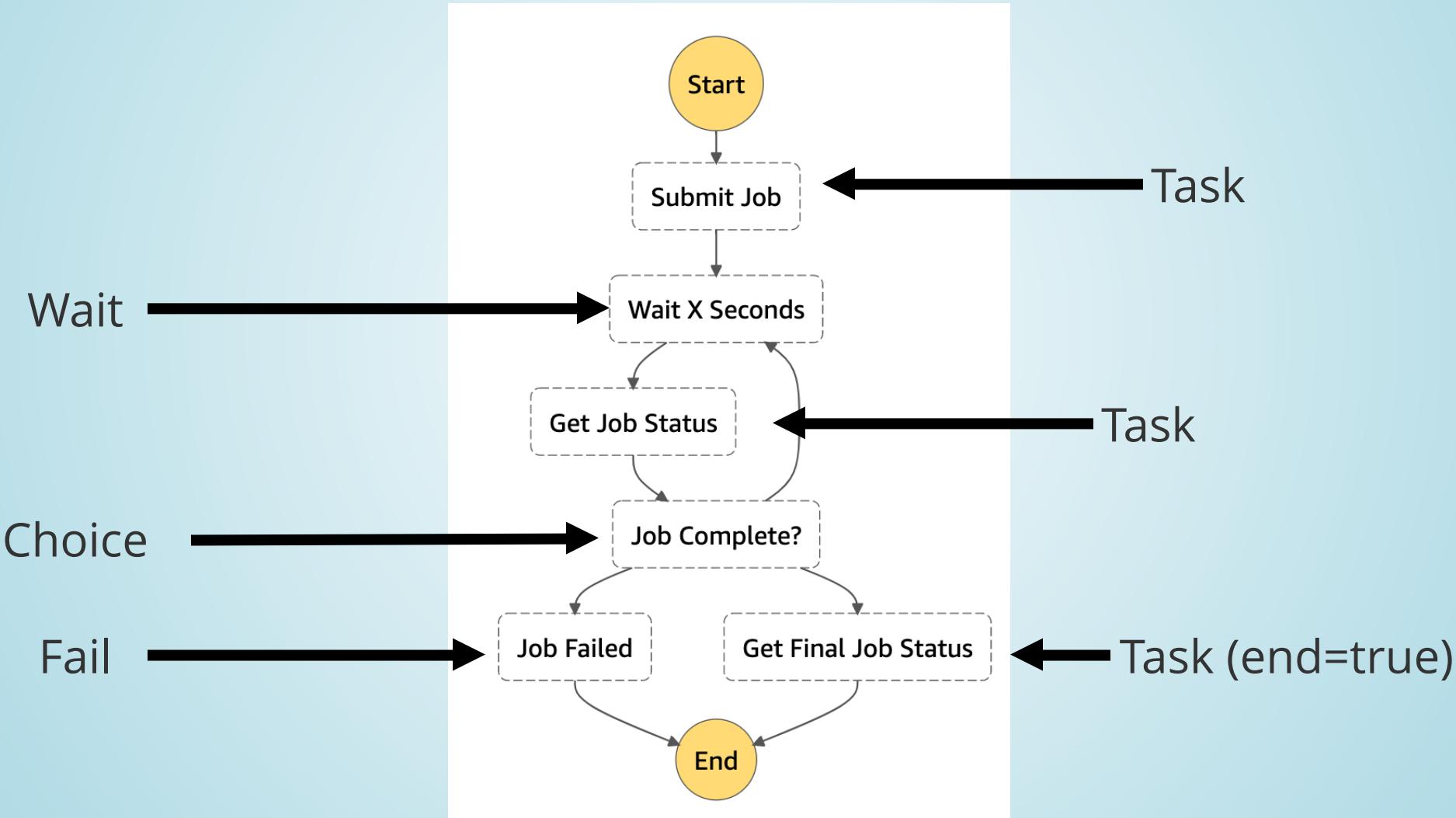
# Example - Batch Job Workflow

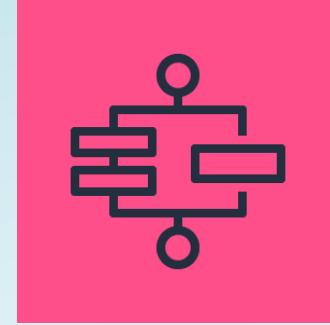


# Example - Batch Job Workflow

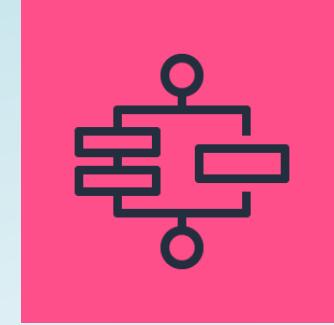


# Example - Batch Job Workflow



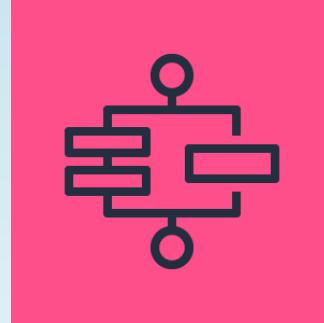


# States



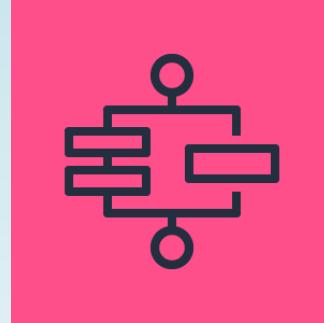
# States

- States are defined in the top-level "States" object



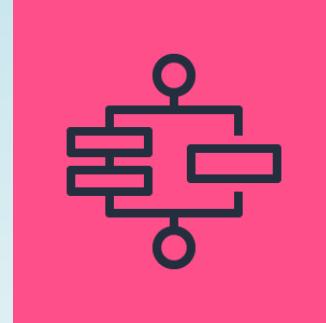
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control



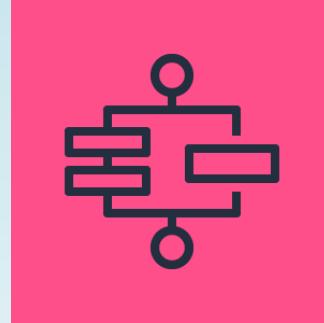
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine



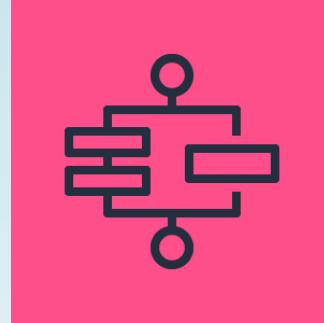
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters



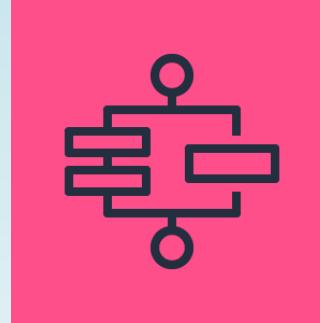
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters
- Each state must have a "Type" field



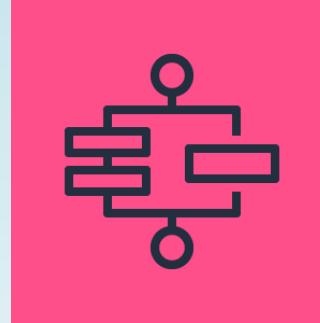
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters
- Each state must have a "Type" field
- Each state may have a "Comment" field



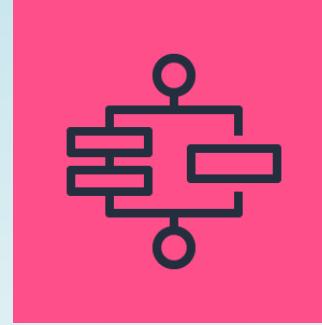
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters
- Each state must have a "Type" field
- Each state may have a "Comment" field
- Most state types have additional requirements



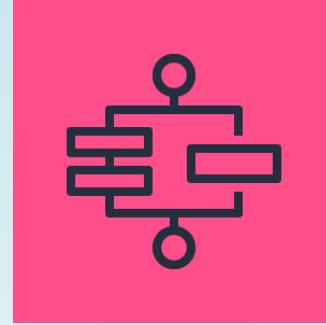
# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters
- Each state must have a "Type" field
- Each state may have a "Comment" field
- Most state types have additional requirements
- Any state other than types **Choice**, **Succeed**, and **Fail** may have a boolean field named "End"

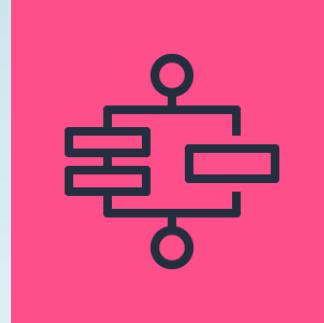


# States

- States are defined in the top-level "States" object
- States describe tasks ("units of work") or flow control
- State name must be unique in scope of state machine
- State name must <= 128 Unicode characters
- Each state must have a "Type" field
- Each state may have a "Comment" field
- Most state types have additional requirements
- Any state other than types **Choice**, **Succeed**, and **Fail** may have a boolean field named "End"
- Terminal states have {"End": true} or are of type **Succeed** or **Fail**

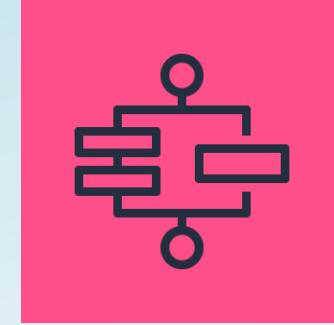


# Simple State Example

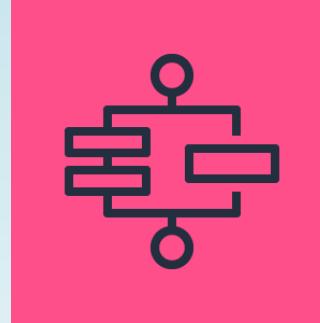


# Simple State Example

```
"HelloWorld": {  
    "Type": "Task",  
    "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloWorld",  
    "Next": "NextState",  
    "Comment": "Executes the HelloWorld Lambda function"  
}
```

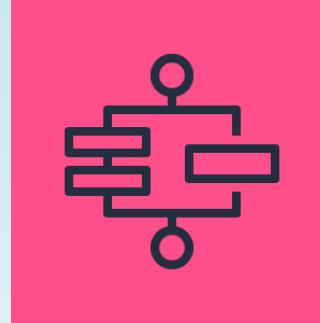


# State Transitions



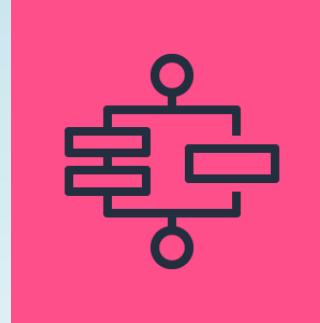
# State Transitions

- After executing the action in a non-terminal state, the state machine transitions to the state specified in "Next" field



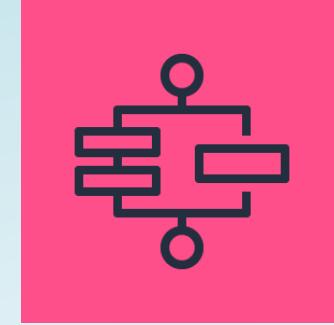
# State Transitions

- After executing the action in a non-terminal state, the state machine transitions to the state specified in "Next" field
- The state transitioned to from a Choice state type is determined by the logic in the Choice state

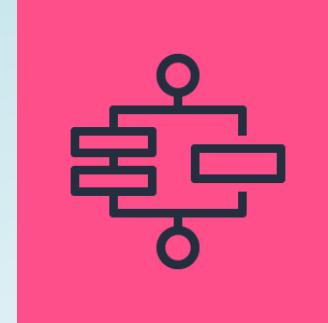


# State Transitions

- After executing the action in a non-terminal state, the state machine transitions to the state specified in "Next" field
- The state transitioned to from a Choice state type is determined by the logic in the Choice state
- A state can have multiple incoming transitions from other states

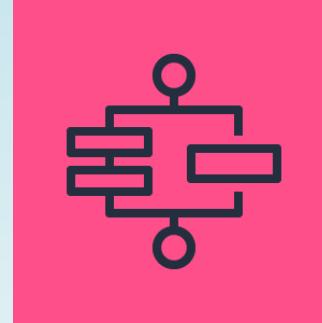


# States Data



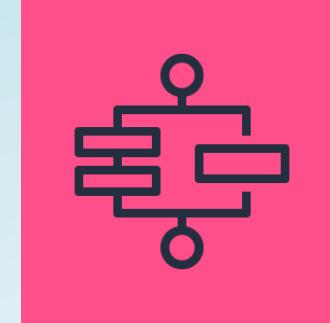
# States Data

- Interpreter passes data between states



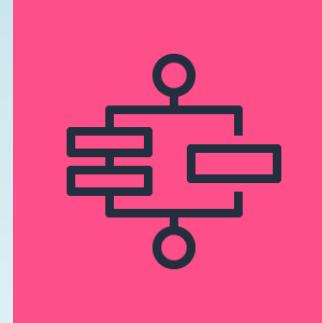
# States Data

- Interpreter passes data between states
- All data must be in JSON format



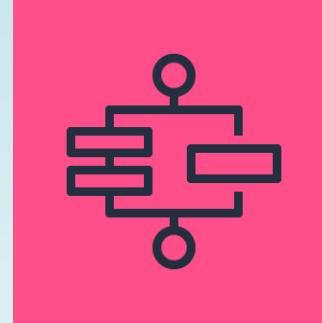
# States Data

- Interpreter passes data between states
- All data must be in JSON format
- Initial data may be provided to the start state



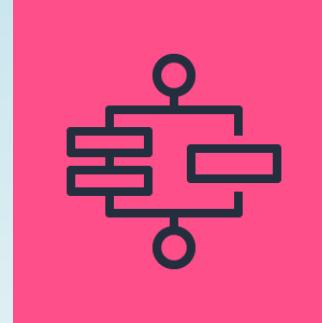
# States Data

- Interpreter passes data between states
- All data must be in JSON format
- Initial data may be provided to the start state
- If no data provided an empty JSON object is passed; i.e. {}



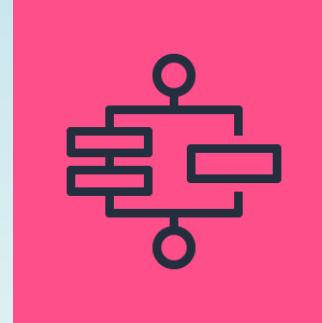
# States Data

- Interpreter passes data between states
- All data must be in JSON format
- Initial data may be provided to the start state
- If no data provided an empty JSON object is passed; i.e. { }
- A state can create output data which must be JSON



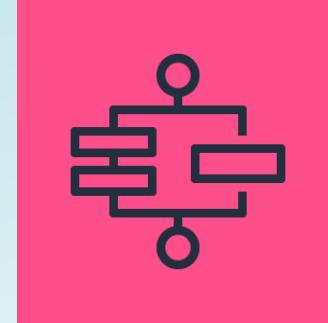
# States Data

- Interpreter passes data between states
- All data must be in JSON format
- Initial data may be provided to the start state
- If no data provided an empty JSON object is passed; i.e. { }
- A state can create output data which must be JSON
- Numbers generally conform to JavaScript double precision, IEEE-854 values

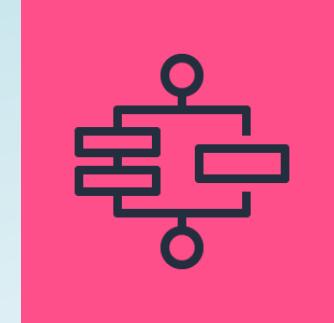


# States Data

- Interpreter passes data between states
- All data must be in JSON format
- Initial data may be provided to the start state
- If no data provided an empty JSON object is passed; i.e. { }
- A state can create output data which must be JSON
- Numbers generally conform to JavaScript double precision, IEEE-854 values
- Strings, booleans, and numbers are valid JSON texts

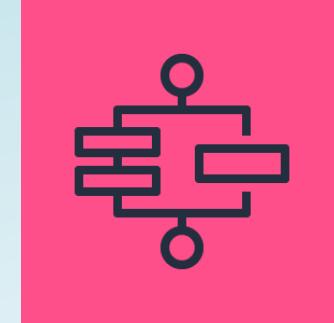


# States Data - Path Expressions



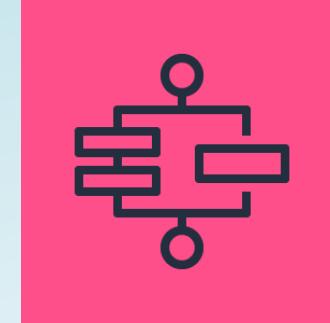
# States Data - Path Expressions

- When states need to access specific fields, they can use JsonPath expressions



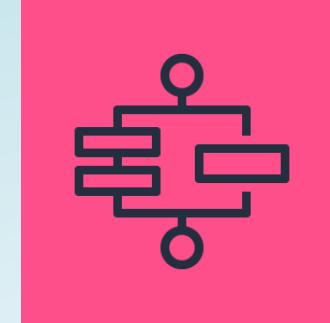
# States Data - Path Expressions

- When states need to access specific fields, they can use JsonPath expressions
- A Path expression starts with a \$



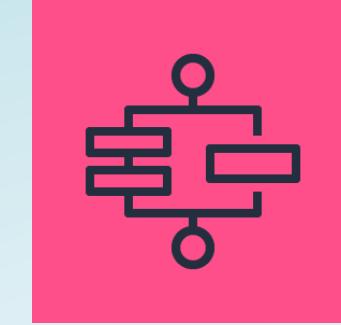
# States Data - Path Expressions

- When states need to access specific fields, they can use JsonPath expressions
- A Path expression starts with a \$
- \$\$ path expression means path is taken from *context* object



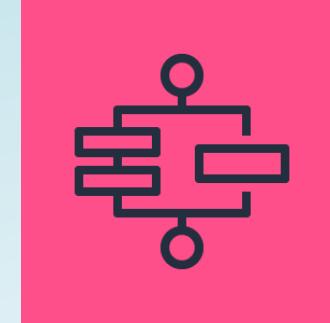
# States Data - Path Expressions

- When states need to access specific fields, they can use JsonPath expressions
- A Path expression starts with a \$
- \$\$ path expression means path is taken from *context* object
- A Reference Path is a Path that resolves to a single node in the JSON data



# States Data - Path Expressions

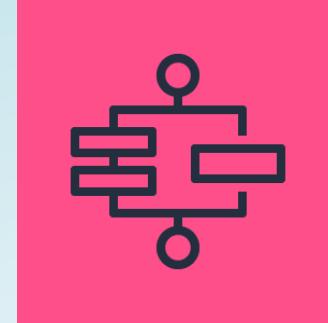
- When states need to access specific fields, they can use JsonPath expressions
- A Path expression starts with a \$
- \$\$ path expression means path is taken from *context* object
- A Reference Path is a Path that resolves to a single node in the JSON data
- The operators "@", ",", ":", and "?" are not supported



# States Data - Path Expressions

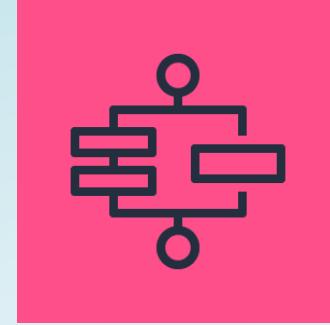
- When states need to access specific fields, they can use JsonPath expressions
- A Path expression starts with a \$
- \$\$ path expression means path is taken from *context* object
- A Reference Path is a Path that resolves to a single node in the JSON data
- The operators "@", "\", ".", and "?" are not supported

<https://github.com/json-path/JsonPath>

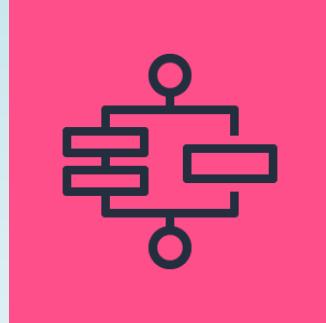


# States Data - Example

```
{  
    "foo": 123,  
    "bar": ["a", "b", "c"],  
    "car": {  
        "cdr": true  
    }  
}  
  
$.foo => 123  
$.bar => ["a", "b", "c"]  
$.car.cdr => true
```

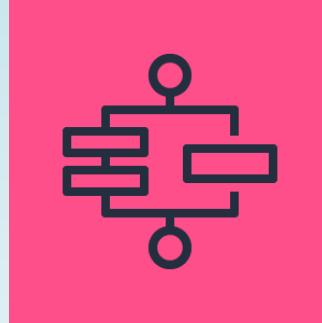


# Handling Errors



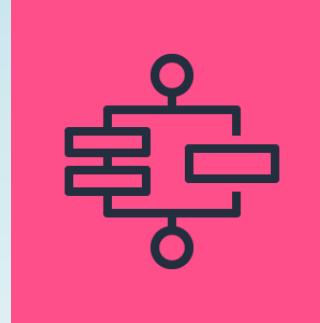
## Handling Errors

- Runtime errors are identified by case-sensitive strings, called Error Names



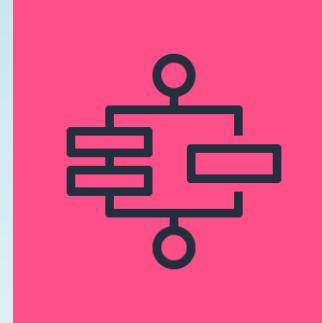
## Handling Errors

- Runtime errors are identified by case-sensitive strings, called Error Names
- The States Language has some reserved Error Names that all begin with "States."



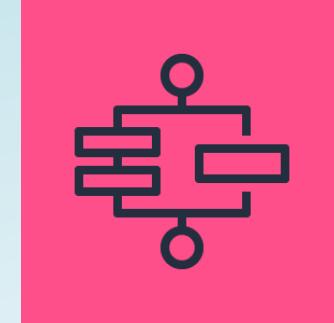
## Handling Errors

- Runtime errors are identified by case-sensitive strings, called Error Names
- The States Language has some reserved Error Names that all begin with "States."
- Custom error names must **not** start with "States."

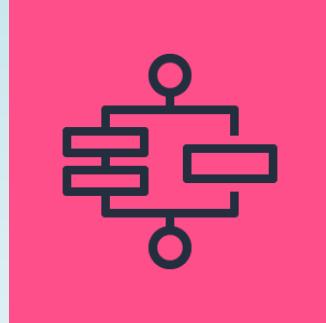


## Handling Errors

- Runtime errors are identified by case-sensitive strings, called Error Names
- The States Language has some reserved Error Names that all begin with "States."
- Custom error names must **not** start with "States."
- Error handling has two flavors, which can be used together
  - Retriers
  - Catchers

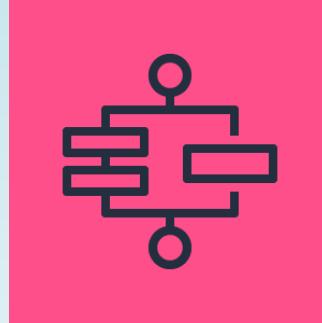


## States - Retriever



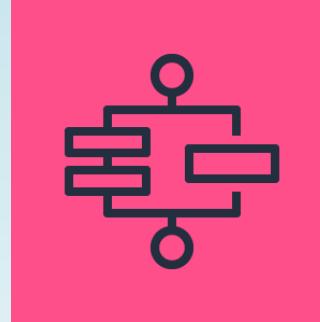
## States - Retriever

- Task and Parallel State types may include a "Retry" field that defines an array of *Retriever*s



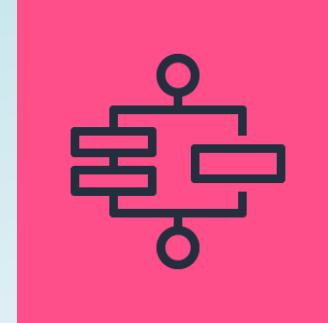
## States - Retriers

- Task and Parallel State types may include a "Retry" field that defines an array of *Retriers*
- Required Retrier fields:
  - "ErrorEquals" array - specifies error names handled

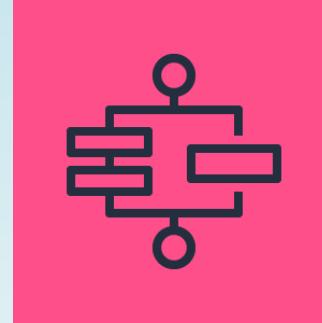


## States - Retriever

- Task and Parallel State types may include a "Retry" field that defines an array of *Retriever*s
- Required Retriever fields:
  - "ErrorEquals" array - specifies error names handled
- Optional Retriever fields...

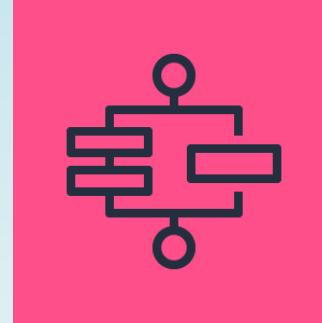


## States - Optional Retriever Fields



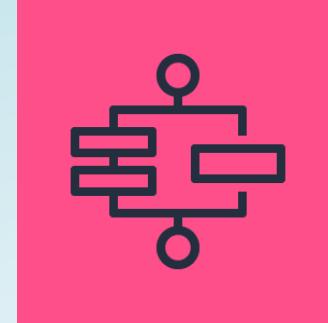
## States - Optional Retrier Fields

- "MaxAttempts"
  - Non-negative, integer field (0 means never)
  - Default = 3



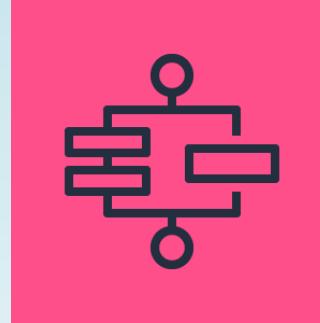
## States - Optional Retrier Fields

- "MaxAttempts"
  - Non-negative, integer field (0 means never)
  - Default = 3
- "IntervalSeconds"
  - Positive, integer field representing delay after error before first retry until
  - Default = 1



## States - Optional Retrier Fields

- "MaxAttempts"
  - Non-negative, integer field (0 means never)
  - Default = 3
- "IntervalSeconds"
  - Positive, integer field representing delay after error before first retry until
  - Default = 1
- "BackoffRate"
  - Multiplier of retry interval applied after every retry
  - Default = 2.0



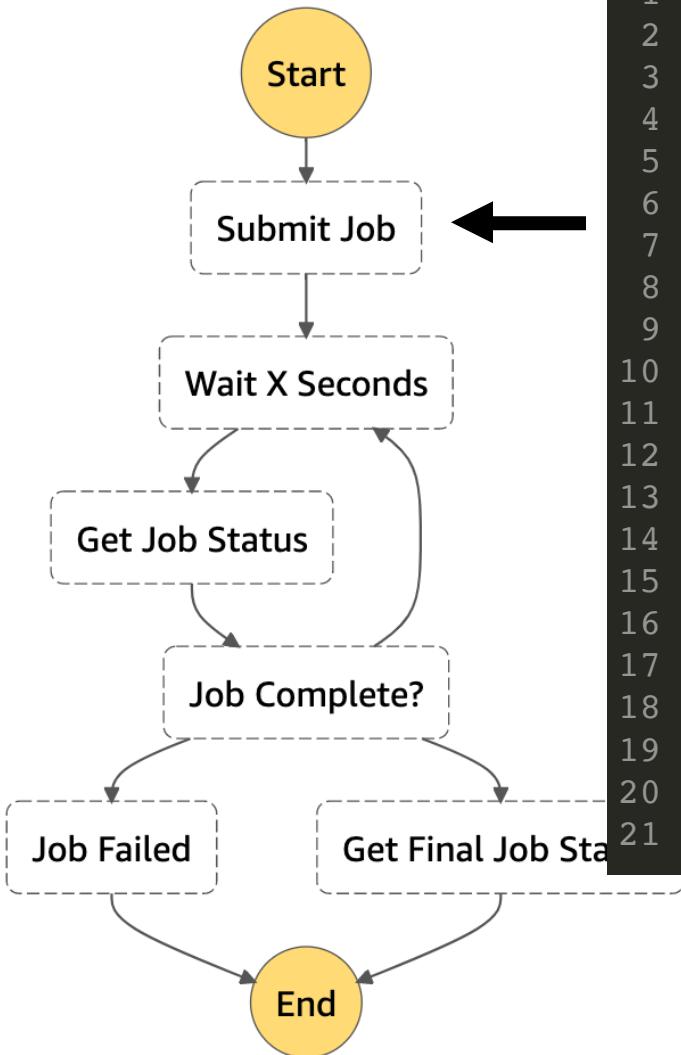
## States - Retrier Example

```
"Retry" : [ {  
    "ErrorEquals": [ "States.Timeout" ],  
    "IntervalSeconds": 3,  
    "MaxAttempts": 3,  
    "BackoffRate": 2.0  
} ]
```

### Interpretation:

- If a timeout error occurs, wait 3 seconds, then retry
- If a timeout occurs again, wait 3.0s x 2.0 (6.0 seconds), then retry
- If a timeout occurs again, wait 6.0s x 2.0 (12.0 seconds), then retry

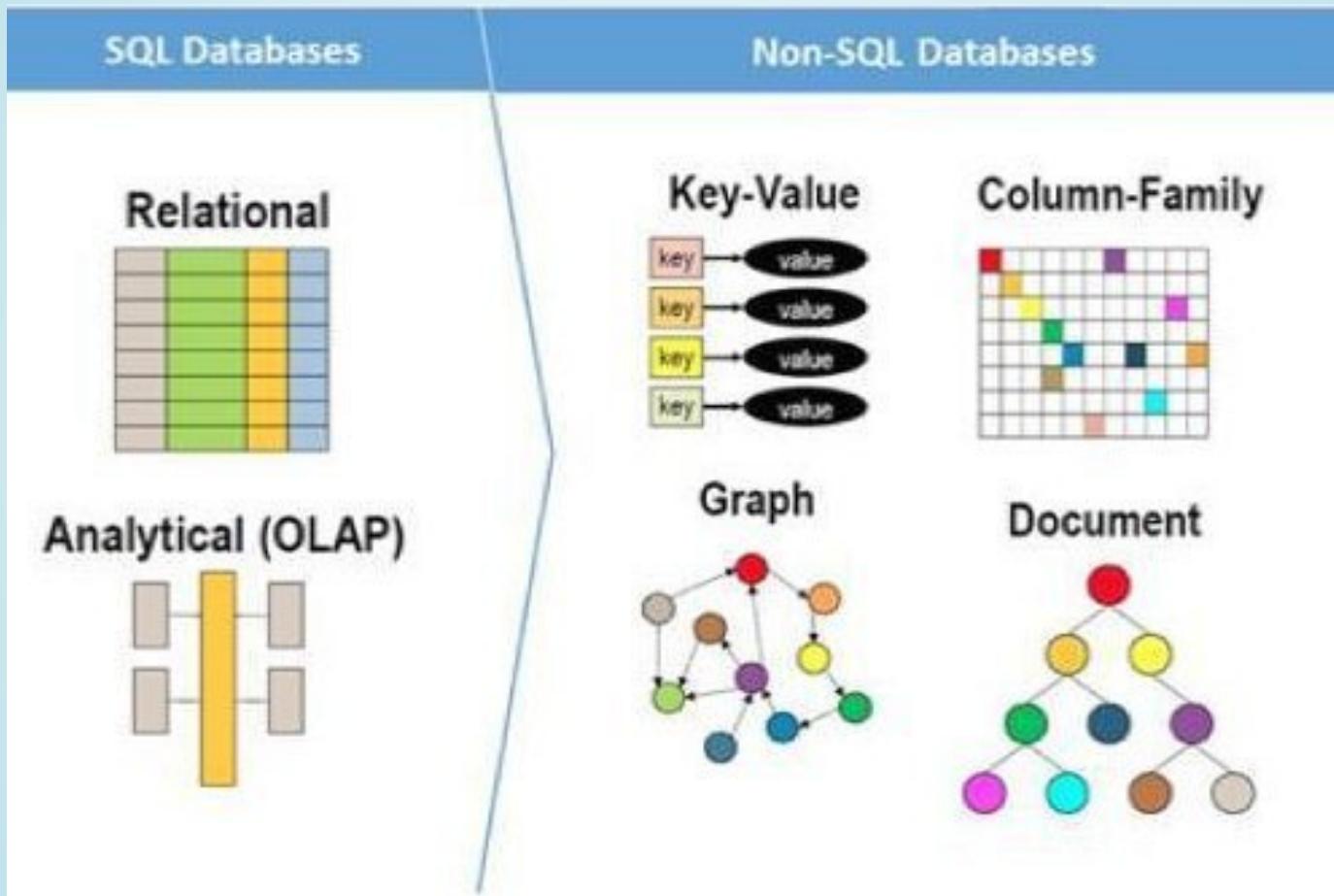
# A Simple State Machine



```
1  {
2      "Comment": "A simple AWS Batch workflow",
3      "StartAt": "Submit Job",
4      "States": {
5          "Submit Job": {
6              "Type": "Task",
7              "Resource":
8                  "arn:<PARTITION>:lambda:::function:SubmitJob",
9              "ResultPath": "$.guid",
10             "Next": "Wait X Seconds",
11             "Retry": [
12                 {
13                     "ErrorEquals": ["States.ALL"],
14                     "IntervalSeconds": 1,
15                     "MaxAttempts": 3,
16                     "BackoffRate": 2
17                 }
18             ]
19         },
20         ...
21     }
```

# Serverless Databases

# Types of Databases



<https://www.kdnuggets.com/2016/07/seven-steps-understanding-nosql-databases.html>

# Scaling Relational Databases

To scale, we scale up\*

Machines can only grow  
so big in CPU, memory



# Scaling Relational Databases

To scale, we scale up\*

Machines can only grow  
so big in CPU, memory



\* Different vendors implement sharding tricks to scale horizontally, but they're complex, maintenance-intensive, and a bit hacky

# Scaling NoSQL Databases

To scale, we scale out



# It's not an either-or situation

Our serverless applications can use both,  
but doing so makes things more complex

NoSQL



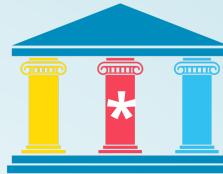
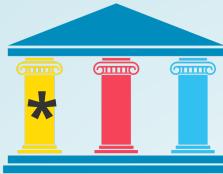
SQL



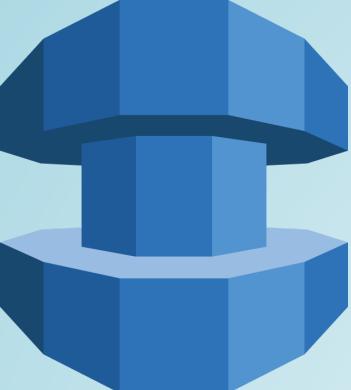
# Amazon Web Services (AWS)



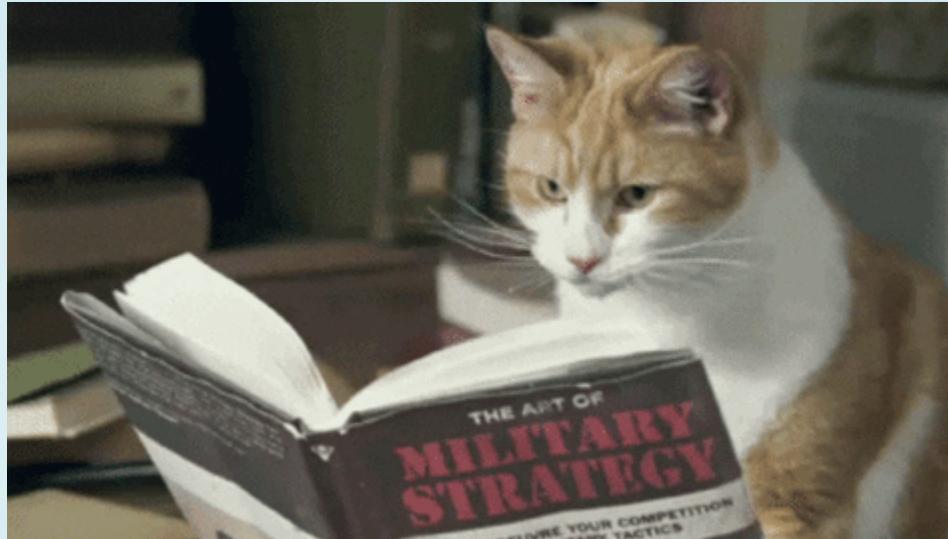
# AWS



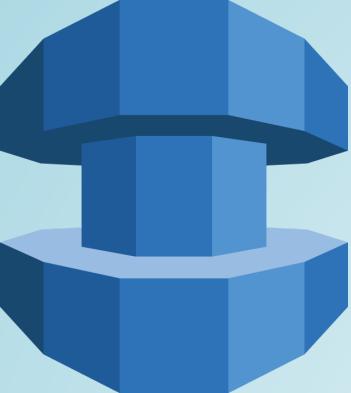
<b>Database</b>	<b>Not managing servers</b>	<b>Autoscales + / - to 0</b>	<b>Pay for what you use</b>
ElastiCache	<b>Yes</b>	<b>No</b>	<b>Yes*</b> \$/hr cache size
RDS	<b>Yes</b>	<b>No</b>	<b>No</b>
Aurora Serverless	<b>Yes</b>	<b>Yes</b>	<b>Yes</b> \$ per cpu-sec \$ per GB/month \$ per MReq
DynamoDB	<b>Yes</b>	<b>Yes*</b> On-Demand	<b>Yes*</b>



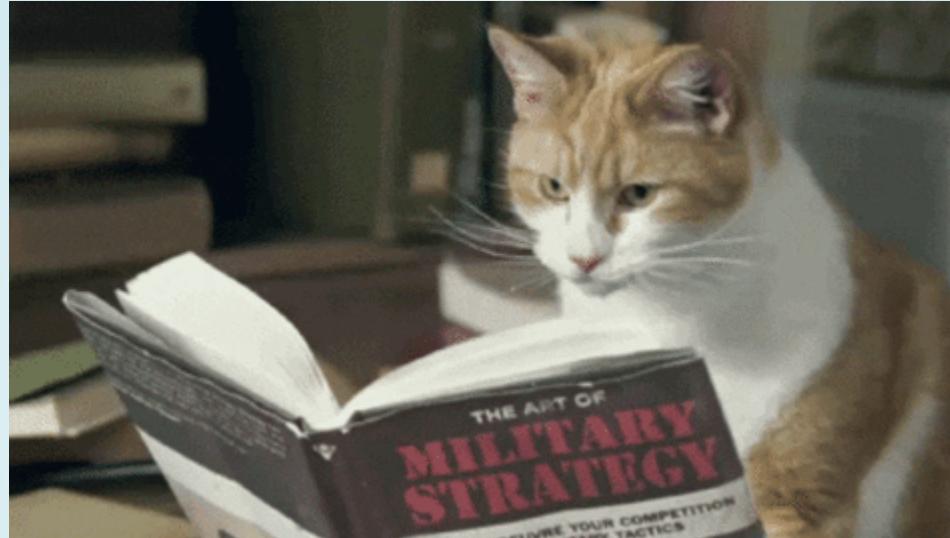
# AWS Aurora Serverless



## High Performance

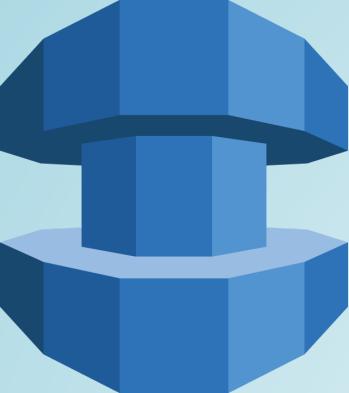


# AWS Aurora Serverless



## High Performance

- 5x standard MySQL

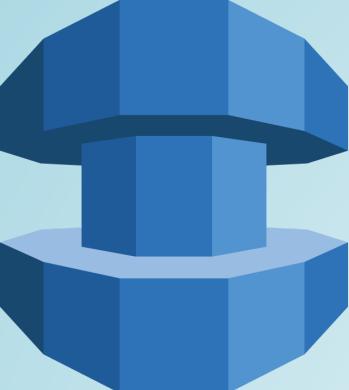


# AWS Aurora Serverless



## High Performance

- 5x standard MySQL
- 3x standard PostgreSQL

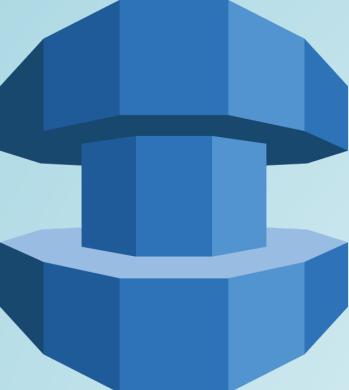


# AWS Aurora Serverless

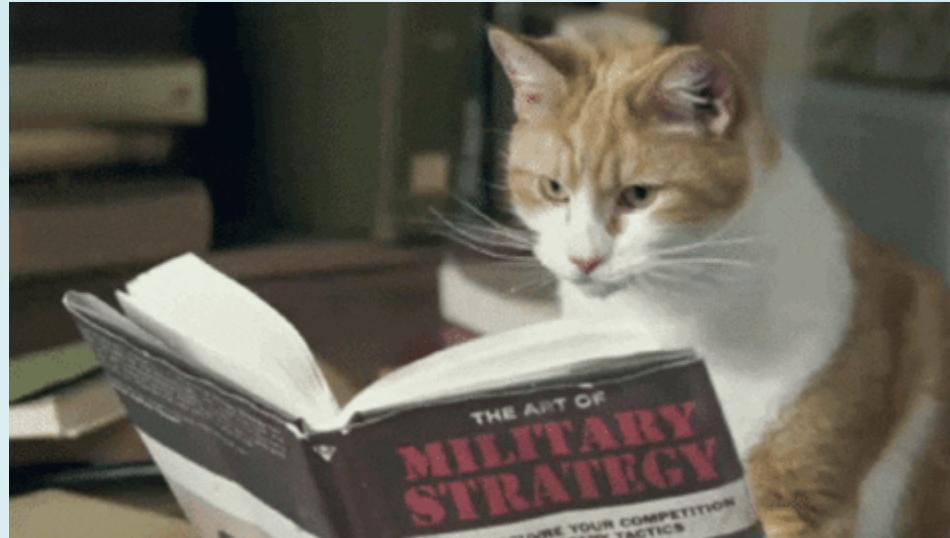


## High Performance

- 5x standard MySQL
- 3x standard PostgreSQL
- Up to 15 read replicas across 3 AZs



# AWS Aurora Serverless



## High Performance

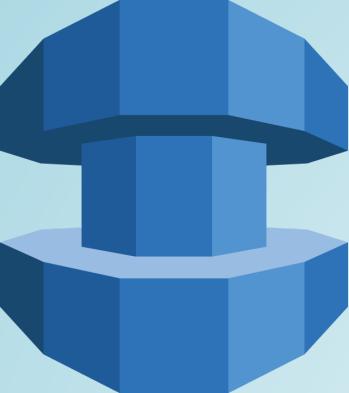
- 5x standard MySQL
- 3x standard PostgreSQL
- Up to 15 read replicas across 3 AZs
- Up to 64TB per DB instance



# AWS Aurora Serverless



## High Availability

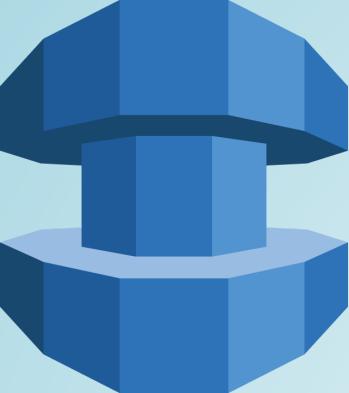


# AWS Aurora Serverless



## High Availability

- Replicates 6 copies across 3 AZs

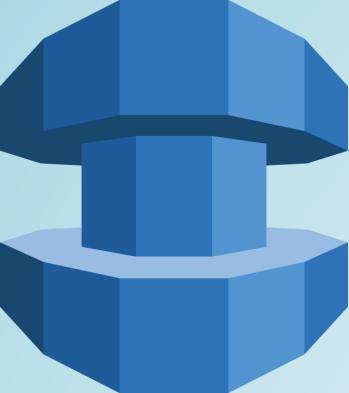


# AWS Aurora Serverless



## High Availability

- Replicates 6 copies across 3 AZs
- 99.99% availability

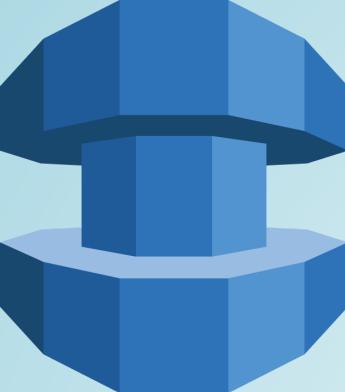


# AWS Aurora Serverless



## High Availability

- Replicates 6 copies across 3 AZs
- 99.99% availability
- Continuous, automatic backup to S3

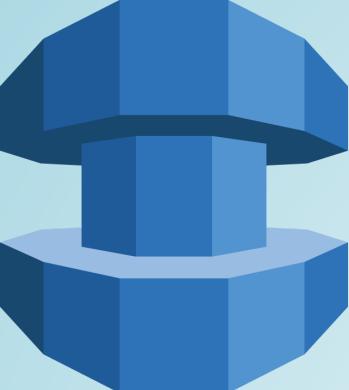


# AWS Aurora Serverless



## High Availability

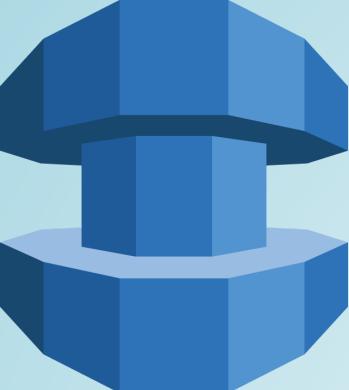
- Replicates 6 copies across 3 AZs
- 99.99% availability
- Continuous, automatic backup to S3
- Global Database allows spanning regions



# AWS Aurora Serverless



## Highly Secure

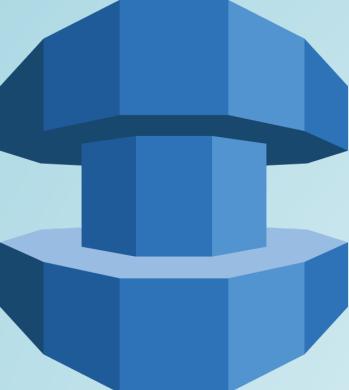


# AWS Aurora Serverless



## Highly Secure

- Network isolation in VPC

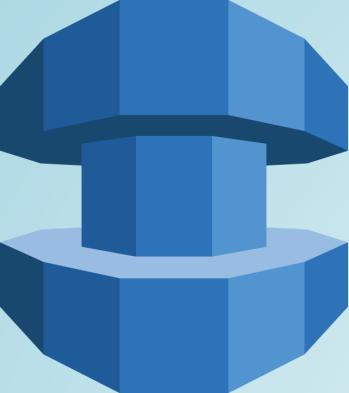


# AWS Aurora Serverless



## Highly Secure

- Network isolation in VPC
- Encryption at rest for storage, backups, snapshots, replicas

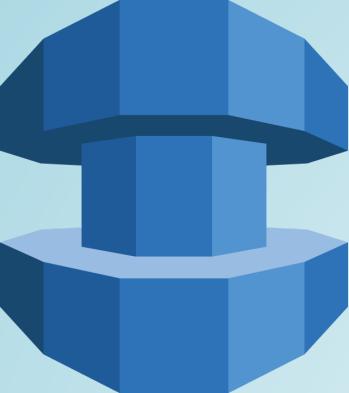


# AWS Aurora Serverless

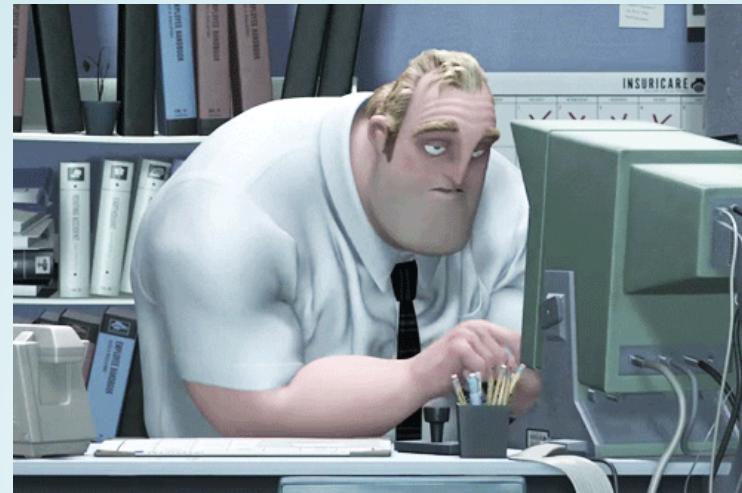


## Highly Secure

- Network isolation in VPC
- Encryption at rest for storage, backups, snapshots, replicas
- You control the encryption keys



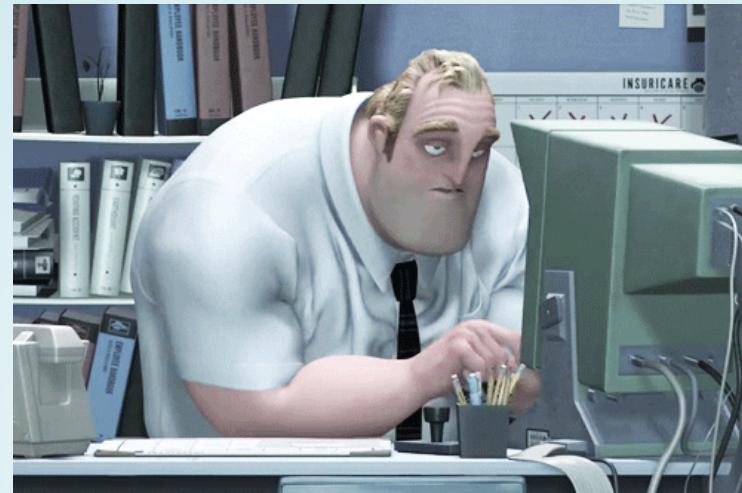
# AWS Aurora Serverless



Fully Managed

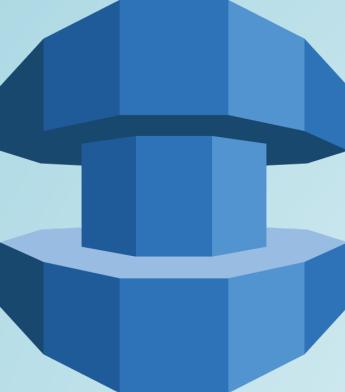


# AWS Aurora Serverless

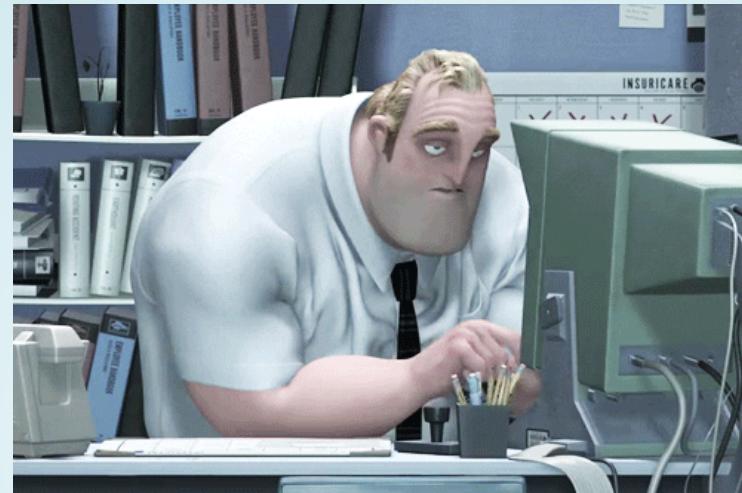


## Fully Managed

- Managed by AWS RDS

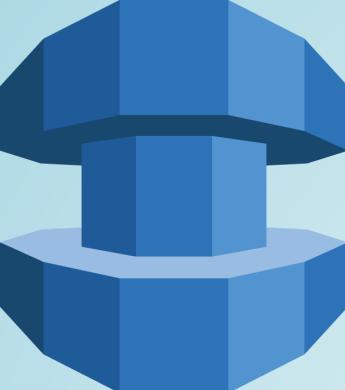


# AWS Aurora Serverless

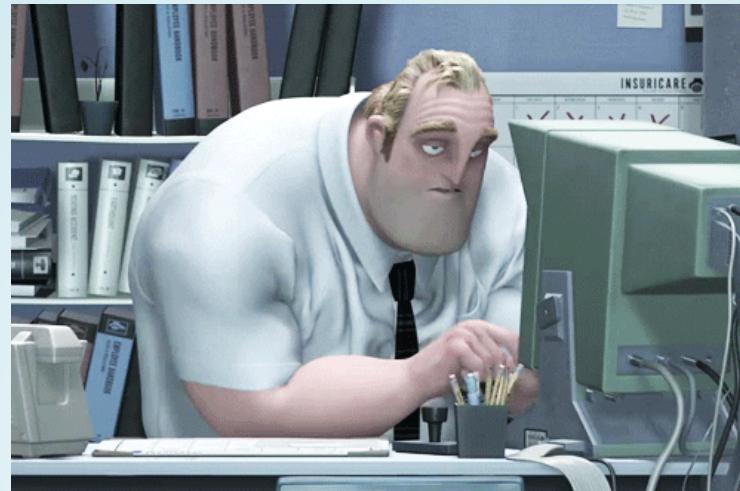


## Fully Managed

- Managed by AWS RDS
- Amazon handles all patching



# AWS Aurora Serverless

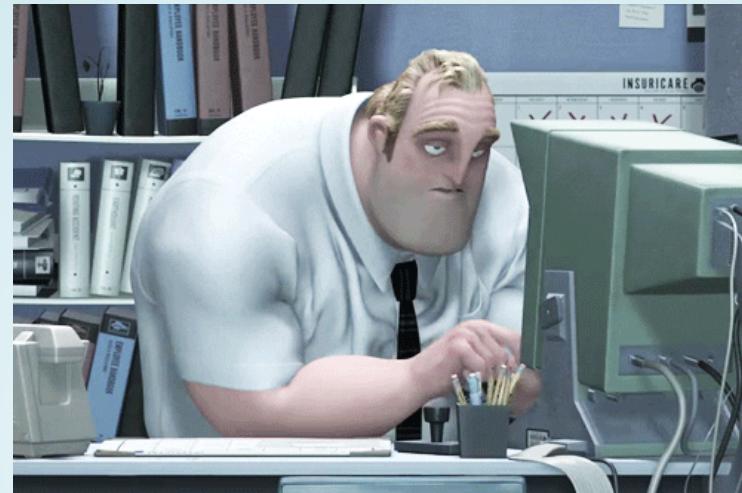


## Fully Managed

- Managed by AWS RDS
- Amazon handles all patching
- Automatic backups

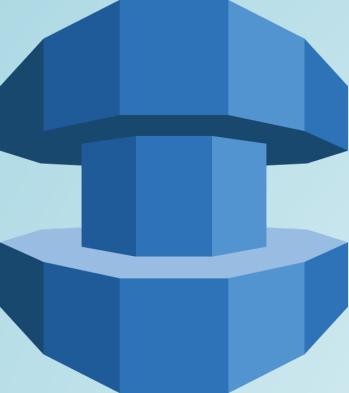


# AWS Aurora Serverless



## Fully Managed

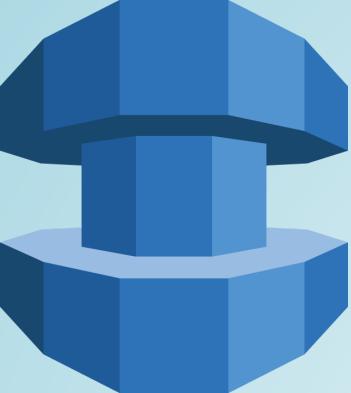
- Managed by AWS RDS
- Amazon handles all patching
- Automatic backups
- Monitoring through CloudWatch,  
Performance Insights



# AWS Aurora Serverless



## Use Cases

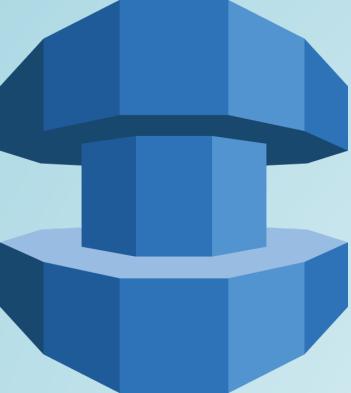


# AWS Aurora Serverless



## Use Cases

- New apps where load is uncertain

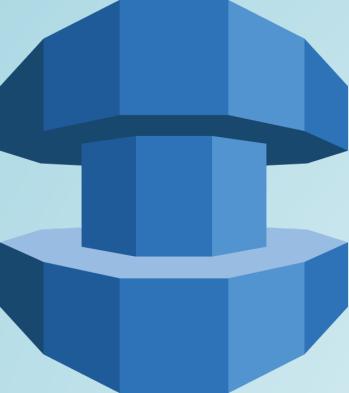


# AWS Aurora Serverless



## Use Cases

- New apps where load is uncertain
- Variable & "spiky" workloads

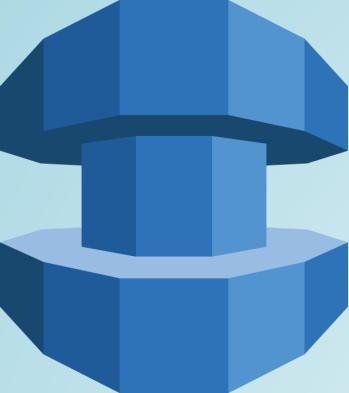


# AWS Aurora Serverless



## Use Cases

- New apps where load is uncertain
- Variable & "spiky" workloads
- Development and test environments

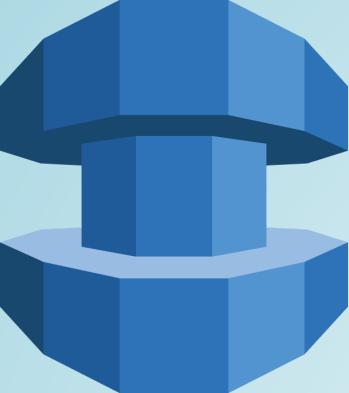


# AWS Aurora Serverless



## Use Cases

- New apps where load is uncertain
- Variable & "spiky" workloads
- Development and test environments
- Multi-tenant apps with varying loads each with their own DB

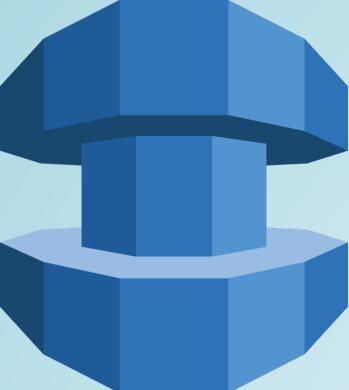


# AWS Aurora Serverless



## Use Cases

- New apps where load is uncertain
- Variable & "spiky" workloads
- Development and test environments
- Multi-tenant apps with varying loads each with their own DB
- Infrequently used apps (e.g. month end billing)



# AWS Aurora Serverless



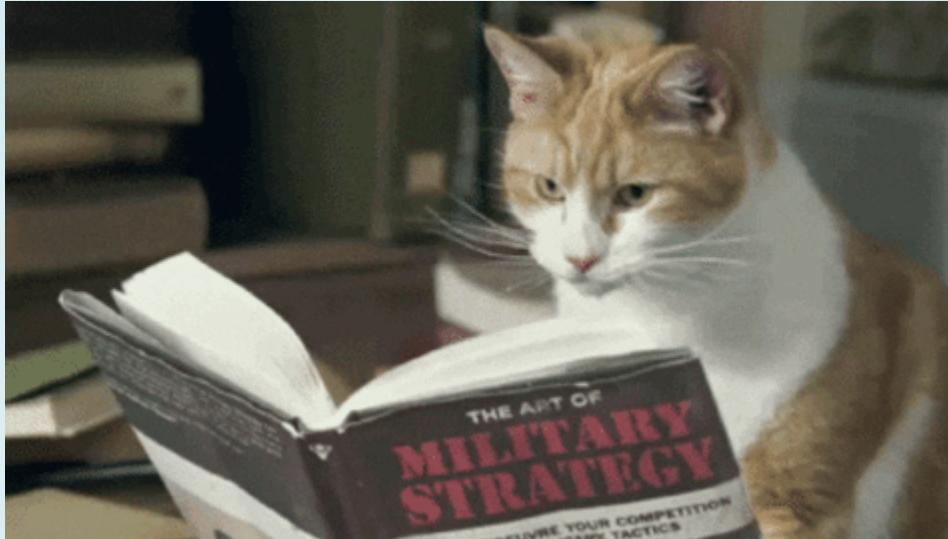
## Use Cases

- New apps where load is uncertain
- Variable & "spiky" workloads
- Development and test environments
- Multi-tenant apps with varying loads each with their own DB
- Infrequently used apps (e.g. month end billing)

**~30 secs from  
0 to start**



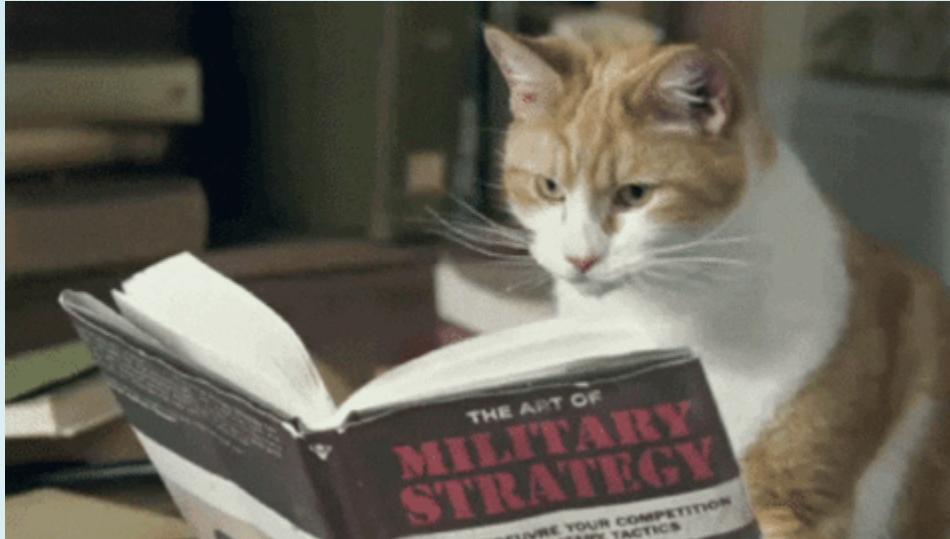
# AWS DynamoDB



## High Performance



# AWS DynamoDB



## High Performance

- Single-digit millisecond response at any scale



# AWS DynamoDB

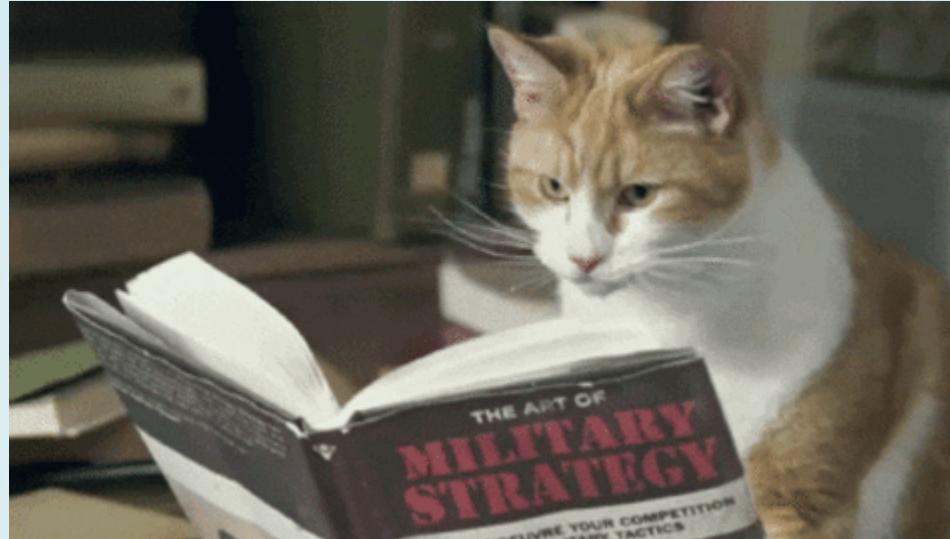


## High Performance

- Single-digit millisecond response at any scale
- Global replication allows more local access



# AWS DynamoDB



## High Performance

- Single-digit millisecond response at any scale
- Global replication allows more local access
- DynamoDB Accelerator (DAX) supplemental in-memory cache
  - Microsecond response times



# AWS DynamoDB



# High Availability



# AWS DynamoDB



## High Availability

- SLA: Standard tables - 99.99%



# AWS DynamoDB



## High Availability

- SLA: Standard tables - 99.99%
- SLA: Global tables - 99.999%



# AWS DynamoDB



Highly Secure



# AWS DynamoDB



## Highly Secure

- SOC, PCI, FedRAMP, and HIPAA Compliant



# AWS DynamoDB



## Highly Secure

- SOC, PCI, FedRAMP, and HIPAA Compliant
- All data is encrypted at rest, include tables, indexes, replications, and DAX clusters



# AWS DynamoDB



## Highly Secure

- SOC, PCI, FedRAMP, and HIPAA Compliant
- All data is encrypted at rest, include tables, indexes, replications, and DAX clusters
- Use AWS or your own encryption keys



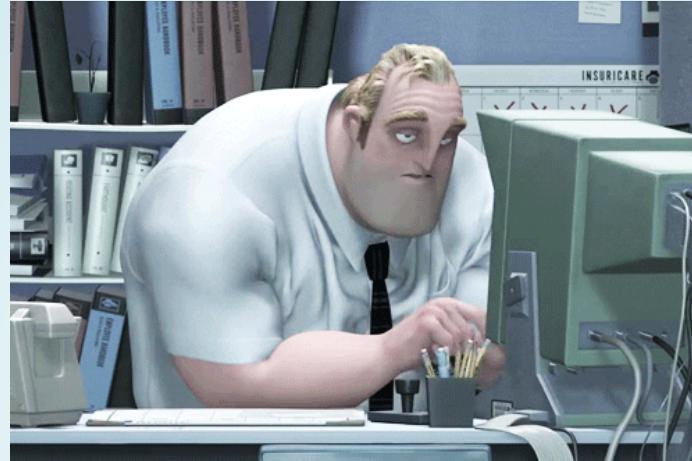
# AWS DynamoDB



Fully Managed



# AWS DynamoDB



## Fully Managed

- No servers to patch or maintain



# AWS DynamoDB

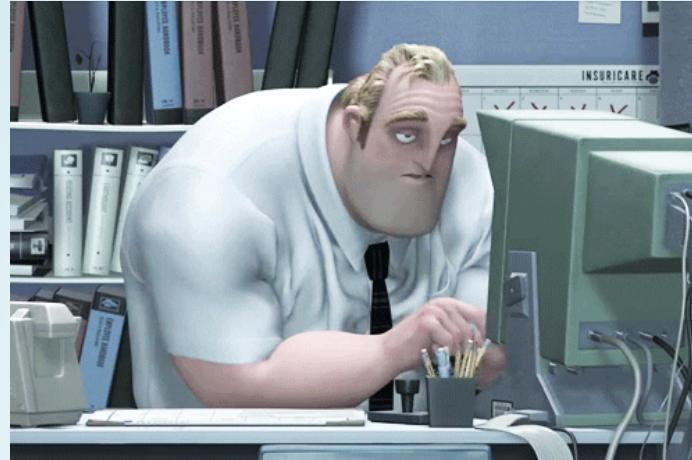


## Fully Managed

- No servers to patch or maintain
- Automatic scaling



# AWS DynamoDB

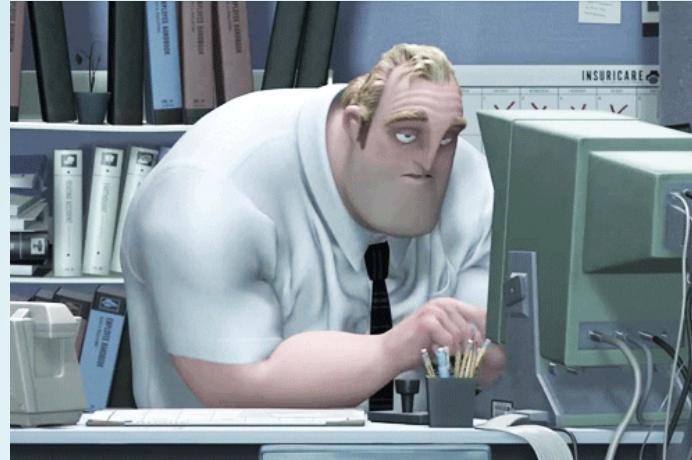


## Fully Managed

- No servers to patch or maintain
- Automatic scaling
- Provisioned or On-Demand capacity



# AWS DynamoDB

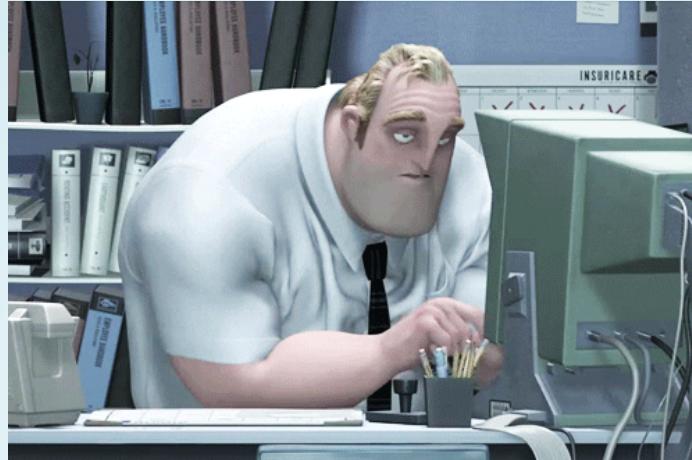


## Fully Managed

- No servers to patch or maintain
- Automatic scaling
- Provisioned or On-Demand capacity
- Automatic backups



# AWS DynamoDB



## Fully Managed

- No servers to patch or maintain
- Automatic scaling
- Provisioned or On-Demand capacity
- Automatic backups
- Monitoring through CloudWatch, CloudTrail



# AWS DynamoDB

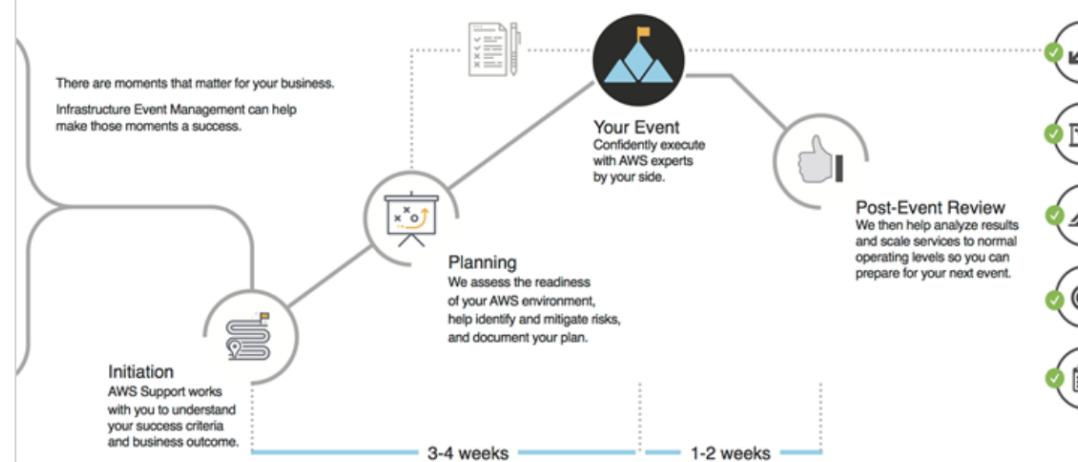


Nick Walsh  
@TheNickWalsh

Replying to [@dabit3](#)

+Amazon retail - [@dynamodb](#) had 7.11 trillion API calls (45.4mil reqs/s peak) from prime day alone this year

## view of Infrastructure Event Management



Amazon Prime Day 2019 – Powered by AWS | Amazon Web Services

What did you buy for Prime Day? I bought a 34" Alienware Gaming Monitor and used it to replace a pair of 25" monitors that had served me well for the past s...

[aws.amazon.com](#)

# Lab 4 - Serverless Databases



# Summary



# Summary

- Serverless is the right architecture for many workloads, but not all



# Summary

- Serverless is the right architecture for many workloads, but not all
- AS CDK gives us a rich SDK in which to do *Infrastructure as Code* for AWS
  - CDK8s extends that for Kubernetes
  - CDK for Terraform extends that to other providers



# Summary

- Serverless is the right architecture for many workloads, but not all
- AS CDK gives us a rich SDK in which to do *Infrastructure as Code* for AWS
  - CDK8s extends that for Kubernetes
  - CDK for Terraform extends that to other providers
- Lambdas are a serverless workhorse, but are most useful when integrated with other services



# Summary

- Serverless is the right architecture for many workloads, but not all
- AS CDK gives us a rich SDK in which to do *Infrastructure as Code* for AWS
  - CDK8s extends that for Kubernetes
  - CDK for Terraform extends that to other providers
- Lambdas are a serverless workhorse, but are most useful when integrated with other services
- Do asynchronous, event-driven serverless



# Summary

- Serverless is the right architecture for many workloads, but not all
- AS CDK gives us a rich SDK in which to do *Infrastructure as Code* for AWS
  - CDK8s extends that for Kubernetes
  - CDK for Terraform extends that to other providers
- Lambdas are a serverless workhorse, but are most useful when integrated with other services
- Do asynchronous, event-driven serverless
- Use serverless databases if you can



# A few resources

- AWS Serverless: <https://aws.amazon.com/serverless>
- AWS CDK:  
[docs.aws.amazon.com/cdk/v2/guide/getting\\_started.html](docs.aws.amazon.com/cdk/v2/guide/getting_started.html)
- Construct Hub: <constructs.dev>
- CDK Examples:  
<https://github.com/aws-samples/aws-cdk-examples>



# Thanks for attending!



When questions come up...

Post in our Meetup or on our Slack workspace