

# Quick stack building, an introduction to AWS CDK and infrastructure as code.

Slides will be online later  
Leo Lapworth (@leolapworth)

Me: Leo Lapworth  
@leolapworth

# Me: Leo Lapworth

@leolapworth

- 20+ years developer / sysadmin / team lead

# Me: Leo Lapworth

## @leolapworth

- 20+ years developer / sysadmin / team lead
- Talk: @YAPC::EU 2007 - Evolving Architecture

# Me: Leo Lapworth

## @leolapworth

- 20+ years developer / sysadmin / team lead
- Talk: @YAPC::EU 2007 - Evolving Architecture
- Talk @ YAPC::EU 2008 - Evolving Architecture  
- further

# Me: Leo Lapworth

## @leolapworth

- 20+ years developer / sysadmin / team lead
- Talk: @YAPC::EU 2007 - Evolving Architecture
- Talk @ YAPC::EU 2008 - Evolving Architecture  
- further
- **Everything** has changed... even more!

# Exceptions

# Exceptions

- Cache invalidation

# Exceptions

- Cache invalidation
- Naming



**Corey Quinn**  
@QuinnyPig

How do they screw up their service names so badly? No idea, ask the AWS Systems Manager Marketing Manager.

# Topics

# Topics

- Infrastructure as Code

# Topics

- Infrastructure as Code
- CloudFormation

# Topics

- Infrastructure as Code
- CloudFormation
- CDK

# Topics

- Infrastructure as Code
- CloudFormation
- CDK
- Tips along the way

Context...

A history of...  
before...

`Infrastructure as Code`

( abridged )

# Version control (history)

# Version control (history)

```
> cp important_file.txt important_file.txt.bak
```

# Version control (history)

```
> cp important_file.txt important_file.txt.bak
```

```
> cp important_file.txt important_file.txt.bak2
```

# Version control (history)

> cp important\_file.txt important\_file.txt.bak

> cp important\_file.txt important\_file.txt.bak2

> cvs

# Version control (history)

> cp important\_file.txt important\_file.txt.bak

> cp important\_file.txt important\_file.txt.bak2

> cvs

> svn → subversion

# Version control (history)

> cp important\_file.txt important\_file.txt.bak

> cp important\_file.txt important\_file.txt.bak2

> cvs

> svn → subversion

> mercurial

# Version control (now)

# Version control (now)

> git

# Provisioning

# Provisioning

> Client “Hello”

# Provisioning

- > Client “Hello”
- > Hosting “Hello, how can we help?”

# Provisioning

- > Client “Hello”
- > Hosting “Hello, how can we help?”
- > Client “Can I have a new server please?”

# Provisioning

- > Client “Hello”
- > Hosting “Hello, how can we help?”
- > Client “Can I have a new server please?”
- > Hosting “Sure.. what are the specs and which Christmas do you need it by?”

# Configuration

# Configuration

```
> scp config/* root@new99.example.com:/
```

# Configuration

```
> scp config/* root@new99.example.com:/
```

```
> NEW_HOST=new99.example.com sh  
setup_script.sh
```

# Configuration

```
> scp config/* root@new99.example.com:/  
  
> NEW_HOST=new99.example.com sh  
setup_script.sh  
  
> ssh root@new99.example.com; apt-get  
install puppet; puppet run --master:  
puppet.example.com; make tea; puppet run
```

# Configuration

```
> scp config/* root@new99.example.com:/  
  
> NEW_HOST=new99.example.com sh  
setup_script.sh  
  
> ssh root@new99.example.com; apt-get  
install puppet; puppet run --master:  
puppet.example.com; make tea; puppet run  
  
> ansible new99.example.com <playbook>
```

Configuration?  
provisioning?

# Configuration? provisioning?

> docker

# Configuration? provisioning?

- > docker
- > docker-compose

# Configuration? provisioning?

- > docker
- > docker-compose
- > docker-swarm

# Configuration? provisioning?

- > docker
- > docker-compose
- > docker-swarm
- > Kubernetties something or other

History lesson  
over

# Provisioning AWS resources

<https://aws.amazon.com>

Amazon Web Services Sign-In

signin.aws.amazon.com/signin?client\_id=arn%3Aaws%3Aiam%3A%3A015428540659%3Auser%2Fhomepage&redirect\_uri=https%3A%2F%2Fcon...

**aws**

**Sign in ⓘ**

**Email address of your AWS account**

Or to sign in as an IAM user, enter your [account ID](#) or [account alias](#) instead.

**Next**

——— New to AWS? ———

**Create a new AWS account**

**About Amazon.com Sign In**

Amazon Web Services uses information from your Amazon.com account to identify you and allow access to Amazon Web Services. Your use of this site is governed by our [User Agreement](#) and [Privacy Notice](#).

AWS Management Console

eu-west-1.console.aws.amazon.com/console/home?region=eu-west-1#

AWS Services

Find Services You can enter names, keywords or acronyms. Example: Relational Database Service, database, RDS

Recently visited services

All services

- Compute
  - EC2
  - LightSail
  - ECR
  - ECS
  - EKS
  - Lambda
  - Batch
  - Elastic Beanstalk
  - Serverless Application Repository
- Storage
  - S3
  - EFS
  - Fsx
  - S3 Glacier
  - Storage Gateway
  - AWS Backup
- Database
  - RDS
  - DynamoDB
  - ElastiCache
  - Neptune
  - Amazon Redshift
  - Amazon QLDB
  - Amazon DocumentDB
- Migration & Transfer
  - AWS Migration Hub
  - Application Discovery Service
  - Database Migration Service
  - Server Migration Service
  - AWS Transfer for SFTP
  - Snowball
  - DataSync
- Networking & Content Delivery
  - VPC
  - CloudFront
  - Route 53
  - API Gateway
  - Direct Connect
  - AWS App Mesh
  - AWS Cloud Map
  - Global Accelerator
- Developer Tools
  - CodeStar
  - CodeCommit
  - CodeBuild
  - CodeDeploy
  - CodePipeline
  - Cloud9
  - X-Ray
- Robotics
  - AWS RoboMaker
- Blockchain
  - Amazon Managed Blockchain
- Satellite
  - Ground Station
- Management & Governance
  - AWS Organizations
  - CloudWatch
  - AWS Auto Scaling
  - CloudFormation
  - CloudTrail
  - Config
  - OpsWorks
  - Service Catalog
  - Systems Manager
  - Trusted Advisor
  - Managed Services
  - Control Tower
  - AWS License Manager
  - AWS Well-Architected Tool
  - Personal Health Dashboard
  - AWS Chatbot
- Media Services
  - Elastic Transcoder
  - Kinesis Video Streams
  - MediaConnect
  - MediaConvert
  - MediaLive
  - MediaPackage
  - MediaStore
  - MediaTailor
  - Elemental Appliances & Software
- Machine Learning
  - Amazon SageMaker
  - Amazon Comprehend
  - AWS DeepLens
  - Amazon Lex
  - Machine Learning
  - Amazon Polly
  - Rekognition
  - Amazon Transcribe
  - Amazon Translate
  - Amazon Personalize
  - Amazon Forecast
  - Amazon Textract
  - AWS DeepRacer
- Analytics
  - Athena
  - EMR
  - CloudSearch
  - Elasticsearch Service
  - Kinesis
  - QuickSight
  - Data Pipeline
  - AWS Glue
  - AWS Lake Formation
  - MSK
- Customer Engagement
  - Amazon Connect
  - Pinpoint
  - Simple Email Service
- Business Applications
  - Alexa for Business
  - Amazon Chime
  - WorkMail
- Security, Identity, & Compliance
  - IAM
  - Resource Access Manager
  - Cognito
  - Secrets Manager
  - GuardDuty
  - Inspector
  - Amazon Macie
  - AWS Single Sign-On
  - Certificate Manager
  - Key Management Service
  - CloudHSM
  - Directory Service
  - WAF & Shield
  - Artifact
  - Security Hub
- End User Computing
  - WorkSpaces
  - AppStream 2.0
  - WorkDocs
  - WorkLink
- Internet of Things
  - IoT Core
  - Amazon FreeRTOS
  - IoT 1-Click
  - IoT Analytics
  - IoT Device Defender
  - IoT Device Management
  - IoT Events
  - IoT Greengrass
  - IoT SiteWise
  - IoT Things Graph
- AWS Cost Management
  - AWS Cost Explorer
  - AWS Budgets
  - AWS Marketplace Subscriptions
- Game Development

Access resources on the go

Access the Management Console using the AWS Console Mobile App. Learn more

Explore AWS

Register for re:Invent

Join us in Las Vegas December 2 – 6 for 2,500+ sessions, bootcamps, hackathons, workshops, and chalk talks. View session catalog

CloudEndure Migration

Re-host a large number of machines to AWS without worrying about compatibility, performance disruption, or long cutover windows. Get started

EC2 Spot Instances

Run fault-tolerant workloads on Spot Instances and save up to 90% on compute. Learn more

Amazon RDS

Set up, operate, and scale your relational database in the cloud. Learn more

Have feedback?

Submit feedback to tell us about your experience with the AWS Management Console.

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

Services Resource Groups Leo Lapworth Global Support

Amazon S3

Buckets

Batch operations

Block public access (account settings)

Feature spotlight 2

AWS DataSync automates & accelerates moving data into or out of Amazon S3 & Amazon EFS. [Learn more »](#)

Documentation

S3 buckets

Discover the console

Search for buckets All access types

Create bucket Edit public access settings Empty Delete

0 Buckets 0 Regions

You do not have any buckets. Here is how to get started with Amazon S3.

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The screenshot shows the AWS S3 Management Console interface. On the left, there's a sidebar with 'Amazon S3' and several options: 'Buckets' (selected), 'Batch operations', 'Block public access (account settings)', and 'Feature spotlight' (with a '2' notification). The main content area has a banner about AWS DataSync. Below it, there's a search bar, a dropdown for 'All access types', and buttons for 'Create bucket', 'Edit public access settings', 'Empty', and 'Delete'. It displays '0 Buckets' and '0 Regions'. A message at the bottom says 'You do not have any buckets. Here is how to get started with Amazon S3.' At the bottom, there are links for 'Feedback', 'English (US)', and legal notices.

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

aws Services Resource Groups

Leo Lanworth Global

## Create bucket

① Name and region    ② Configure options    ③ Set permissions    ④ Review

Bucket name i

demo\_

Bucket name contains invalid characters '\_'

Region

Create    Cancel    Next

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

The screenshot shows the 'Create bucket' wizard in the AWS S3 Management Console. The first step, 'Name and region', is active. The user has entered 'demo\_' into the 'Bucket name' field. An error message 'Bucket name contains invalid characters '\_'' is displayed below the input field. The 'Create' button is available at the bottom left, while 'Cancel' and 'Next' buttons are at the bottom right. The overall interface is clean with a blue header and a white main content area.

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

Services Resource Groups

Leo Lanworth Global

## Create bucket

① Name and region    ② Configure options    ③ Set permissions    ④ Review

**Bucket name** 

demo-for-scl-2019-w

**Region**

EU (Ireland) 

---

**Create** **Cancel** **Next**

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

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

Services Resource Groups Leo Lapworth Global Support

## Create bucket

① Name and region ② Configure options ③ Set permissions ④ Review

**Properties**

**Versioning**

Keep all versions of an object in the same bucket. [Learn more ↗](#)

**Server access logging**

Log requests for access to your bucket. [Learn more ↗](#)

**Tags**

You can use tags to track project costs. [Learn more ↗](#)

Key Value

3.

Previous Next

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

Services Resource Groups Leo Lapworth Global Support

## Create bucket

3. Set permissions

4. Review

Note: You can grant access to specific users after you create the bucket.

### Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block *all* public access. These settings apply only to this bucket. AWS recommends that you turn on Block *all* public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

3.

Previous Next

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

S3 Management Console    +

console.aws.amazon.com/s3/home?region=eu-west-1

aws Services Resource Groups Leo Lapworth Global Support

## Create bucket

Name and region  Configure options  Set permissions  Review

Name and region Edit

**Bucket name** demo-for-scl-2019-w **Region** EU (Ireland)

Options Edit

**Versioning** Disabled  
**Server access logging** Disabled  
**Tagging** 0 Tags  
**Object-level logging** Disabled

Previous Create bucket

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

S3 Management Console

console.aws.amazon.com/s3/home?region=eu-west-1

Services Resource Groups Leo Lapworth Global Support

Amazon S3

Buckets

Batch operations

Block public access (account settings)

Feature spotlight 2

Amazon S3's newest storage class S3 Intelligent-Tiering auto-tiers your data to deliver cost savings. [Learn more »](#)

Documentation

S3 buckets [Discover the console](#)

Search for buckets All access types

+ Create bucket Edit public access settings Empty Delete

1 Buckets 1 Regions

<input type="checkbox"/> Bucket name	Access	Region	Date created
<input type="checkbox"/> demo-for-scl-2019-w	Bucket and objects not public	EU (Ireland)	Sep 17, 2019 9:29:31 PM GMT+0100

Feedback English (US) © 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The screenshot shows the AWS S3 Management Console interface. On the left sidebar, under the 'Amazon S3' section, the 'Buckets' option is selected. The main content area displays a banner about S3 Intelligent-Tiering and a search bar. Below the search bar are buttons for '+ Create bucket', 'Edit public access settings', 'Empty', and 'Delete'. A summary indicates there is 1 Bucket and 1 Region. The table below lists the single bucket: 'demo-for-scl-2019-w' is listed with 'Bucket and objects not public' access, in the 'EU (Ireland)' region, and was created on Sep 17, 2019, at 9:29:31 PM GMT+0100. At the bottom, there are links for Feedback, Language selection (English (US)), and legal notices.

# Provisioning AWS

# Provisioning AWS

```
> aws sqs create-queue
```

# Provisioning AWS

```
> aws sqs create-queue
```

```
> aws ec2 run-instances --image-id ami-xxxxxxxx --count 1  
--instance-type t2.micro --key-name MyKeyPair --security-  
group-ids sg-903004f8 --subnet-id subnet-6e7f829e
```

# Provisioning AWS

```
> aws sqs create-queue  
  
> aws ec2 run-instances --image-id ami-xxxxxxxx --count 1  
--instance-type t2.micro --key-name MyKeyPair --security-  
group-ids sg-903004f8 --subnet-id subnet-6e7f829e  
  
> aws s3api create-bucket –bucket demo-for-scl-2019
```



What about

What about

What about

Version control?

What about

Version control?

What about

Version control?

Reproducibility?

What about

Version control?

Reproducibility?

What about

Version control?

Reproducibility?

Speed of provisioning?

That's what  
Infrastructure as code  
gives you!

# Some approaches to IaC

# Some approaches to IaC

- Serverless framework <https://serverless.com/>

# Some approaches to IaC

- Serverless framework <https://serverless.com/>
- Arc <https://arc.codes/>

# Some approaches to IaC

- Serverless framework <https://serverless.com/>
- Arc <https://arc.codes>
- Terraform <https://www.terraform.io/>

# Some approaches to IaC

- Serverless framework <https://serverless.com/>
- Arc <https://arc.codes>
- Terraform <https://www.terraform.io/>
- AWS SAM (Serverless Application Model)

# Some approaches to IaC

- Serverless framework <https://serverless.com/>
- Arc <https://arc.codes>
- Terraform <https://www.terraform.io/>
- AWS SAM (Serverless Application Model)
- AWS Native cloudformation

# How do they work? (for AWS)

# How do they work? (for AWS)

- Interface with AWS service APIs

# How do they work? (for AWS)

- Interface with AWS service APIs
- AWS CloudFormation

What are we trying to  
define?

# What are we trying to define?

Resources

# What are we trying to define?

Resources

Parameters

# What are we trying to define?

Resources

Parameters

Permissions

# What are we trying to define?

Resources

Parameters

Permissions

Triggers

# What are we trying to define?

Resources

Parameters

Permissions

Triggers

Dependencies

# What are we trying to define?

Resources

Parameters

Permissions

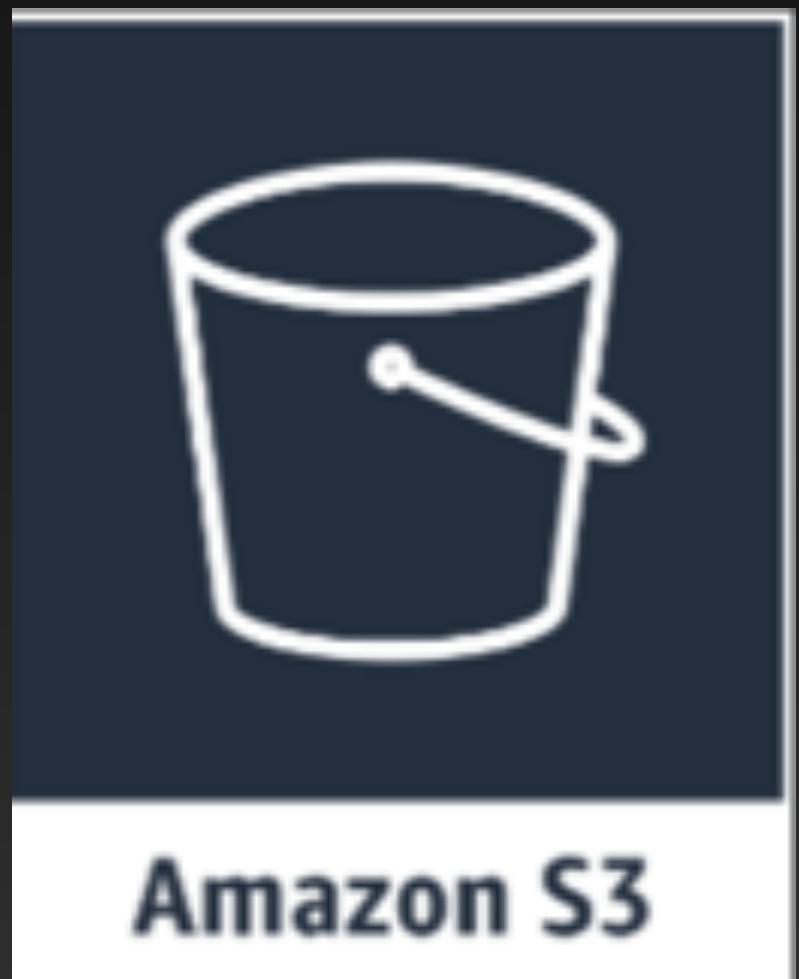
Triggers

Dependencies

Outputs

# Infrastructure graphs

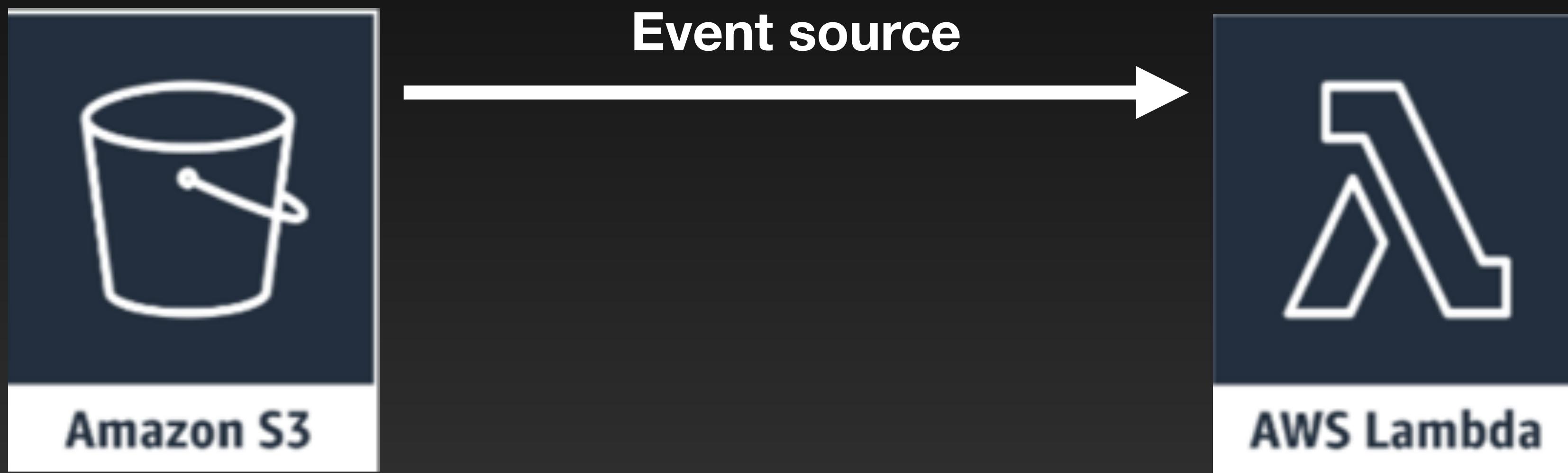
# Infrastructure graph



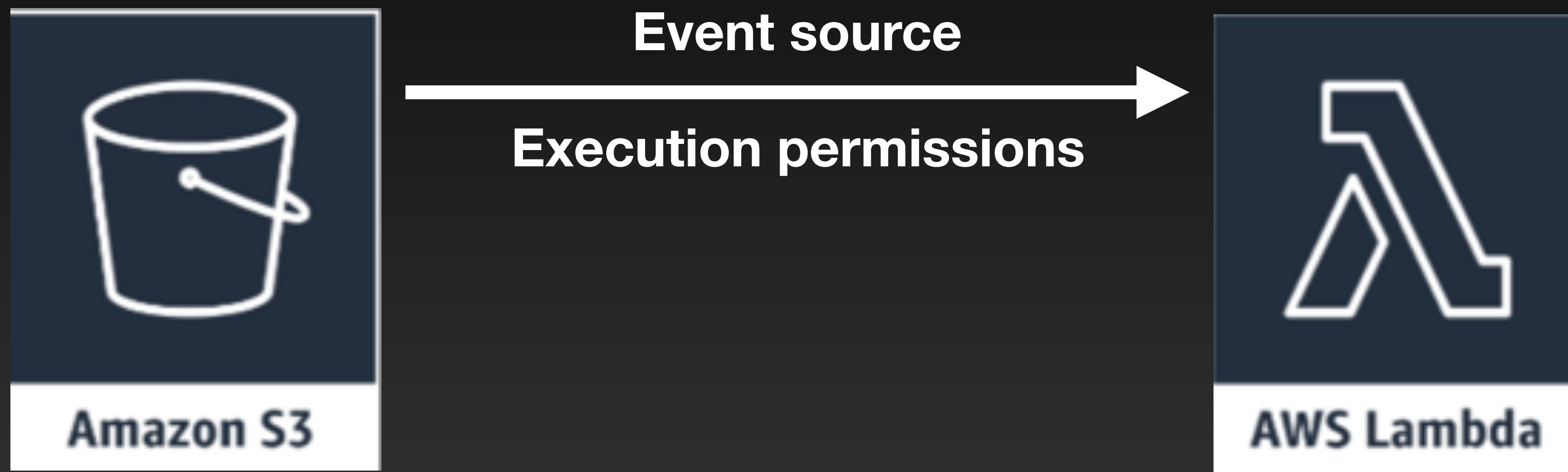
# Infrastructure graph



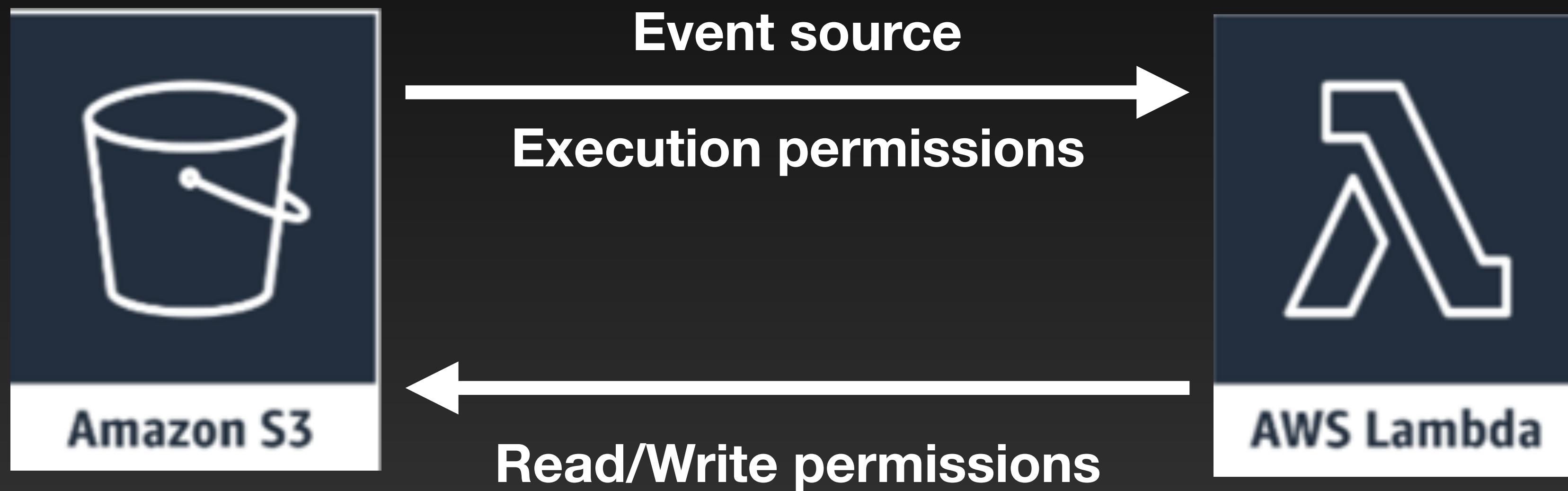
# Infrastructure graph



# Infrastructure graph

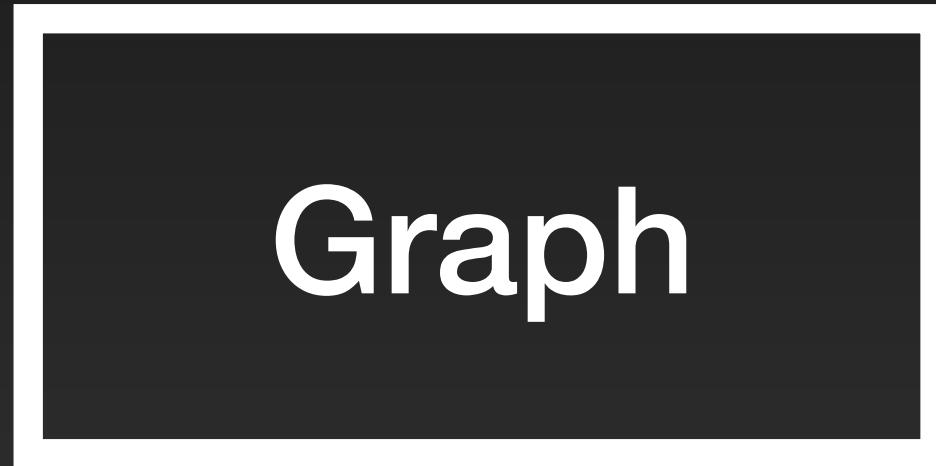


# Infrastructure graph

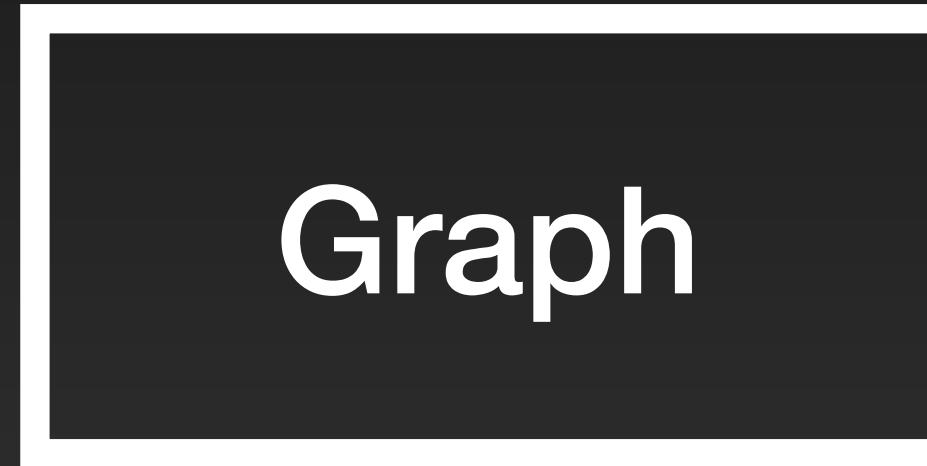


# Infrastructure graph management

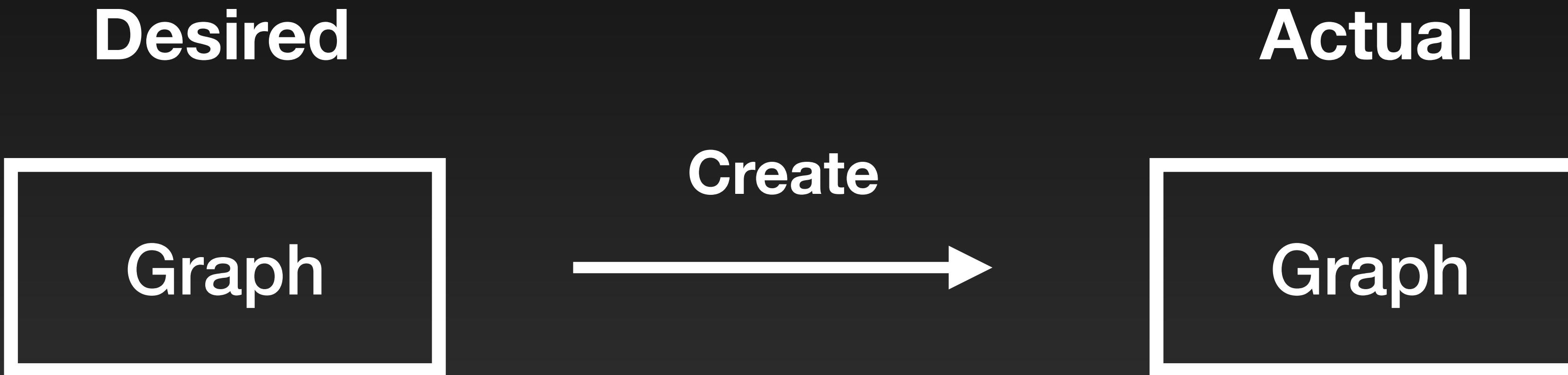
Desired



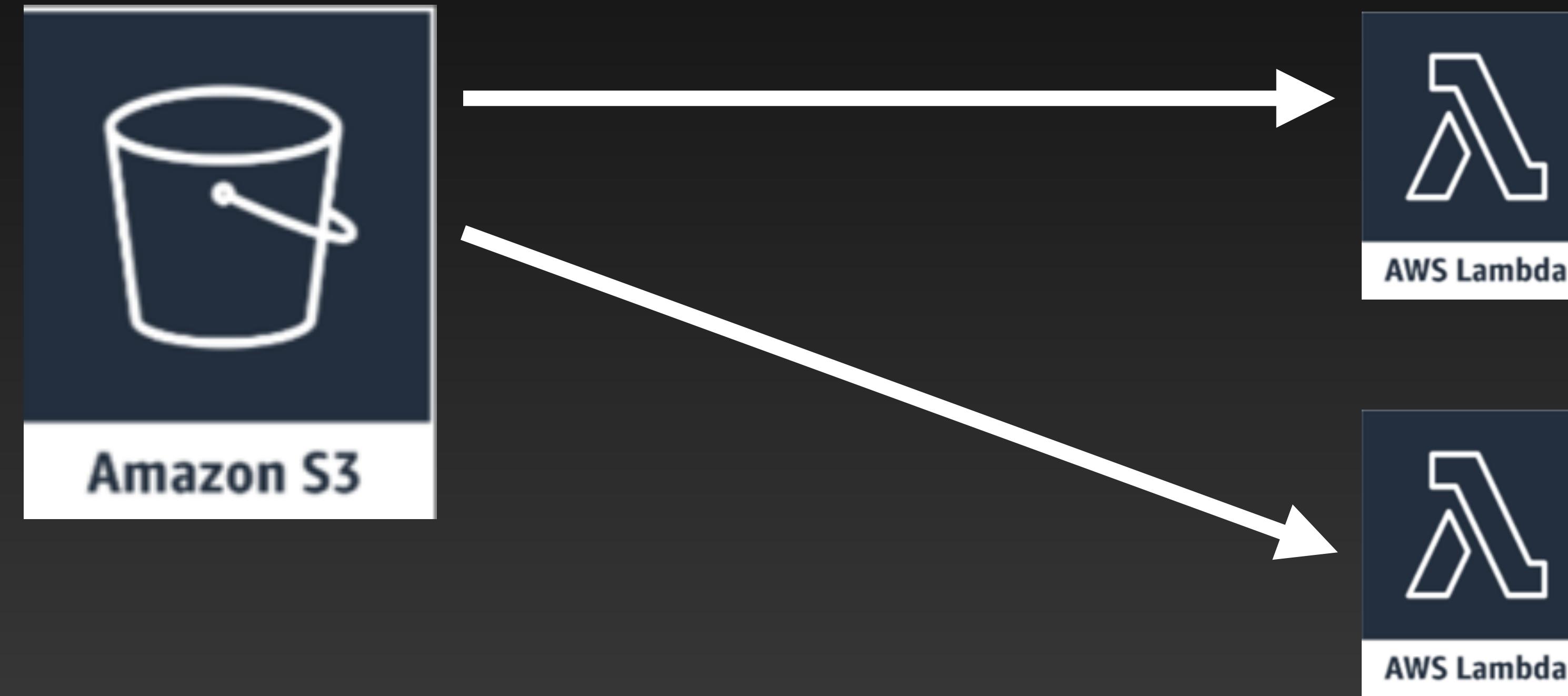
Actual



# Infrastructure graph management

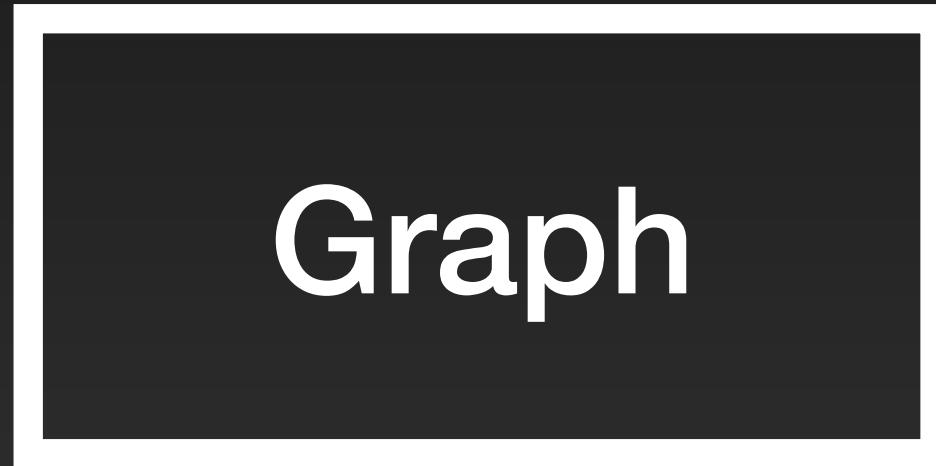


# Infrastructure graph management

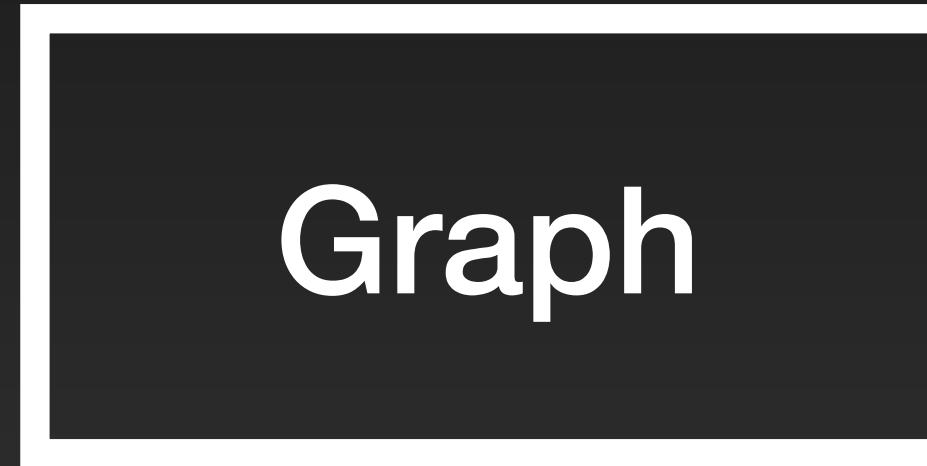


# Infrastructure graph management

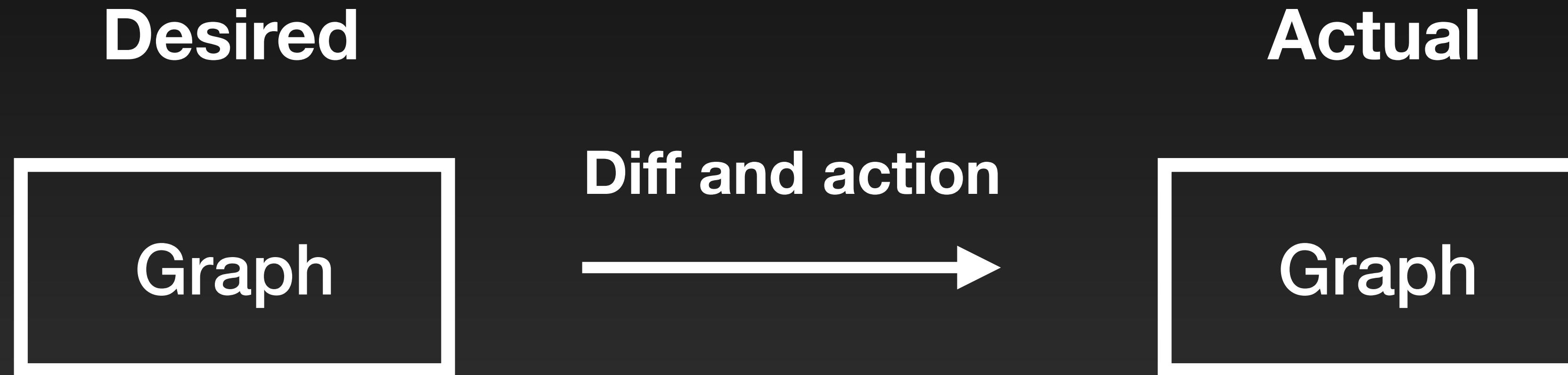
Desired



Actual



# Infrastructure graph management





CloudFormation

```
> cat template.yaml
```

```
Resources:  
  MyDemoBucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      BucketName: demo-for-scl-2019
```

```
> cat template.yaml
```

```
Resources:  
  MyDemoBucket:  
    Type: AWS::S3::Bucket  
    Properties:  
      BucketName: demo-for-scl-2019
```

```
> aws cloudformation deploy --template-file  
  template.yaml --stack-name DemoStack3
```



Waiting for change-set to be created..

Waiting for change-set to be created..

Waiting for stack create/update to complete

Waiting for change-set to be created..

Waiting for stack create/update to complete

Successfully created/updated stack - DemoStack3

# Why DemoStack3?

# Why DemoStack3?

**BucketName:** dem\_for\_scl\_2019

# Why DemoStack3?

**BucketName:** dem\_for\_scl\_2019

# Why DemoStack3?

**BucketName: dem\_for\_scl\_2019**

**Failed to create/update the stack. Run the following command**

# Why DemoStack3?

**BucketName: dem\_for\_scl\_2019**

**Failed to create/update the stack. Run the following command  
to fetch the list of events leading up to the failure**

# Why DemoStack3?

**BucketName: dem\_for\_scl\_2019**

**Failed to create/update the stack. Run the following command  
to fetch the list of events leading up to the failure**

# Why DemoStack3?

**BucketName: dem\_for\_scl\_2019**

**Failed to create/update the stack. Run the following command  
to fetch the list of events leading up to the failure**

**aws cloudformation describe-stack-events --stack-name DemoStack3**

# Why DemoStack3?

**BucketName: dem\_for\_scl\_2019**

**Failed to create/update the stack. Run the following command  
to fetch the list of events leading up to the failure**

```
aws cloudformation describe-stack-events --stack-name DemoStack3
```

```
{
  "StackEvents": [
    {
      "StackId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
      "EventId": "1959c120-d8ba-11e9-bb16-0abc432e4032",
      "StackName": "DemoStack2",
      "LogicalResourceId": "DemoStack2",
      "PhysicalResourceId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
      "ResourceType": "AWS::CloudFormation::Stack",
      "Timestamp": "2019-09-16T19:42:21.223Z",
      "ResourceStatus": "ROLLBACK_COMPLETE"
    },
    {
      "StackId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
      "EventId": "MyFirstBucket-DELETE_COMPLETE-2019-09-16T19:42:20.814Z",
      "StackName": "DemoStack2",
      "LogicalResourceId": "MyFirstBucket",
      "PhysicalResourceId": "",
      "ResourceType": "AWS::S3::Bucket",
      "Timestamp": "2019-09-16T19:42:20.814Z",
      "ResourceStatus": "DELETE_COMPLETE"
    }
  ]
}
```

```
  "StackName": "DemoStack2",
  "LogicalResourceId": "DemoStack2",
  "PhysicalResourceId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
  "ResourceType": "AWS::CloudFormation::Stack",
  "Timestamp": "2019-09-16T19:41:52.401Z",
  "ResourceStatus": "ROLLBACK_IN_PROGRESS",
  "ResourceStatusReason": "The following resource(s) failed to create: [MyFirstBucket]. Requested by user."
},
{
  "StackId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
  "EventId": "MyFirstBucket-CREATE FAILED-2019-09-16T19:41:52.005Z",
  "StackName": "DemoStack2",
  "LogicalResourceId": "MyFirstBucket",
  "PhysicalResourceId": "",
  "ResourceType": "AWS::S3::Bucket",
  "Timestamp": "2019-09-16T19:41:52.005Z",
  "ResourceStatus": "CREATE FAILED",
  "ResourceStatusReason": "Bucket name should not contain '_'",
  "ResourceProperties": "{\"BucketName\":\"dem_for_scl_2019\"}"
},
{
  "StackId": "arn:aws:cloudformation:eu-west-1:374733882471:stack/DemoStack2/0363ce60-d8ba-11e9-b8ea-069f6d7e1780",
  "EventId": "MyFirstBucket-CREATE_IN_PROGRESS-2019-09-16T19:41:51.590Z",
  "StackName": "DemoStack2",
  "LogicalResourceId": "MyFirstBucket",
  "PhysicalResourceId": "",
  "ResourceType": "AWS::S3::Bucket",
  "Timestamp": "2019-09-16T19:41:51.590Z",
  "ResourceStatus": "CREATE_IN_PROGRESS",
  "ResourceProperties": "{\"BucketName\":\"dem_for_scl_2019\"}"
```

I thought this was a talk  
about CDK?

# Using CDK to generate CloudFormation

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');
```

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');
```

```
new s3.Bucket(this, 'MyFirstBucket', {
  bucketName: "demo_for_scl_2019",
});
```

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');

export class HelloCdkStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string) {
    super(scope, id);

    new s3.Bucket(this, 'MyFirstBucket', {
      bucketName: "demo_for_scl_2019",
    });
  }
}
```

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');

export class HelloCdkStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string) {
    super(scope, id);

    new s3.Bucket(this, 'MyFirstBucket', {
      bucketName: "demo_for_scl_2019",
    });
  }
}

const app = new cdk.App();
new HelloCdkStack(app, 'StoreStack');
app.synth();
```

# CDK synth

# CDK synth

> cdk synth

# CDK synth

> cdk synth

```
@aws-cdk/aws-s3/lib/bucket.js:539
    throw new Error(`Invalid S3 bucket name (value: ${bucketName})${os_1.EOL}${
{errors.join(os_1.EOL)})`);
```

**Error: Invalid S3 bucket name (value: demo\_for\_scl\_2019)**

**Bucket name must only contain lowercase characters and the symbols, period (.) and dash (-) (offset: 3)**

at ....

# CDK synth

> cdk synth

```
@aws-cdk/aws-s3/lib/bucket.js:539
    throw new Error(`Invalid S3 bucket name (value: ${bucketName})${os_1.EOL}${
{errors.join(os_1.EOL)})`);
```

→ **Error: Invalid S3 bucket name (value: demo\_for\_scl\_2019)**

**Bucket name must only contain lowercase characters and the symbols, period (.) and dash (-) (offset: 3)**

at ....

# CDK synth

> cdk synth

```
@aws-cdk/aws-s3/lib/bucket.js:539
    throw new Error(`Invalid S3 bucket name (value: ${bucketName})${os_1.EOL}${
{errors.join(os_1.EOL)})`);
```

Error: Invalid S3 bucket name (value: demo\_for\_scl\_2019)

Bucket name must only contain lowercase characters and the symbols, period (.) and dash (-) (offset: 3)

at ....

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');

export class HelloCdkStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string) {
    super(scope, id);

    new s3.Bucket(this, 'MyFirstBucket', {
      bucketName: 'demo_for_scl_2019',
    });
  }
}

const app = new cdk.App();
new HelloCdkStack(app, 'StoreStack');
app.synth();
```

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');

export class HelloCdkStack extends cdk.Stack {
  constructor(scope: cdk.App, id: string) {
    super(scope, id);

    new s3.Bucket(this, 'MyFirstBucket', {
      bucketName: 'demo-for-scl-2019',
    });
  }
}

const app = new cdk.App();
new HelloCdkStack(app, 'StoreStack');
app.synth();
```



> **cdk synth**

```
> cdk synth
```

Resources:

MyFirstBucketB8884501:

Type: AWS::S3::Bucket

Properties:

BucketName: demo-for-scl-2019

**Resources:**

MyFirstBucketB8884501:

Type: AWS::S3::Bucket

Properties:

BucketName: demo-for-scl-2019

UpdateReplacePolicy: Retain

DeletionPolicy: Retain



## CDK synth

Resources:

MyFirstBucketB8884501:

Type: AWS::S3::Bucket

Properties:

BucketName: demo-for-scl-2019

UpdateReplacePolicy: Retain

DeletionPolicy: Retain

Metadata:

aws:cdk:path: StoreStack/MyFirstBucket/Resources

CDKMetadata:

Type: AWS::CDK::Metadata

Properties: .....



# CDK structure

(bottom up)

# CDK structure

(bottom up)

- CloudFormation (Cfn)

# CDK structure

(bottom up)

- CloudFormation (Cfn)
- Resource Constructs

# CDK structure

## (bottom up)

- CloudFormation (Cfn)
- Resource Constructs
- Constructs (reusable)

# CDK structure

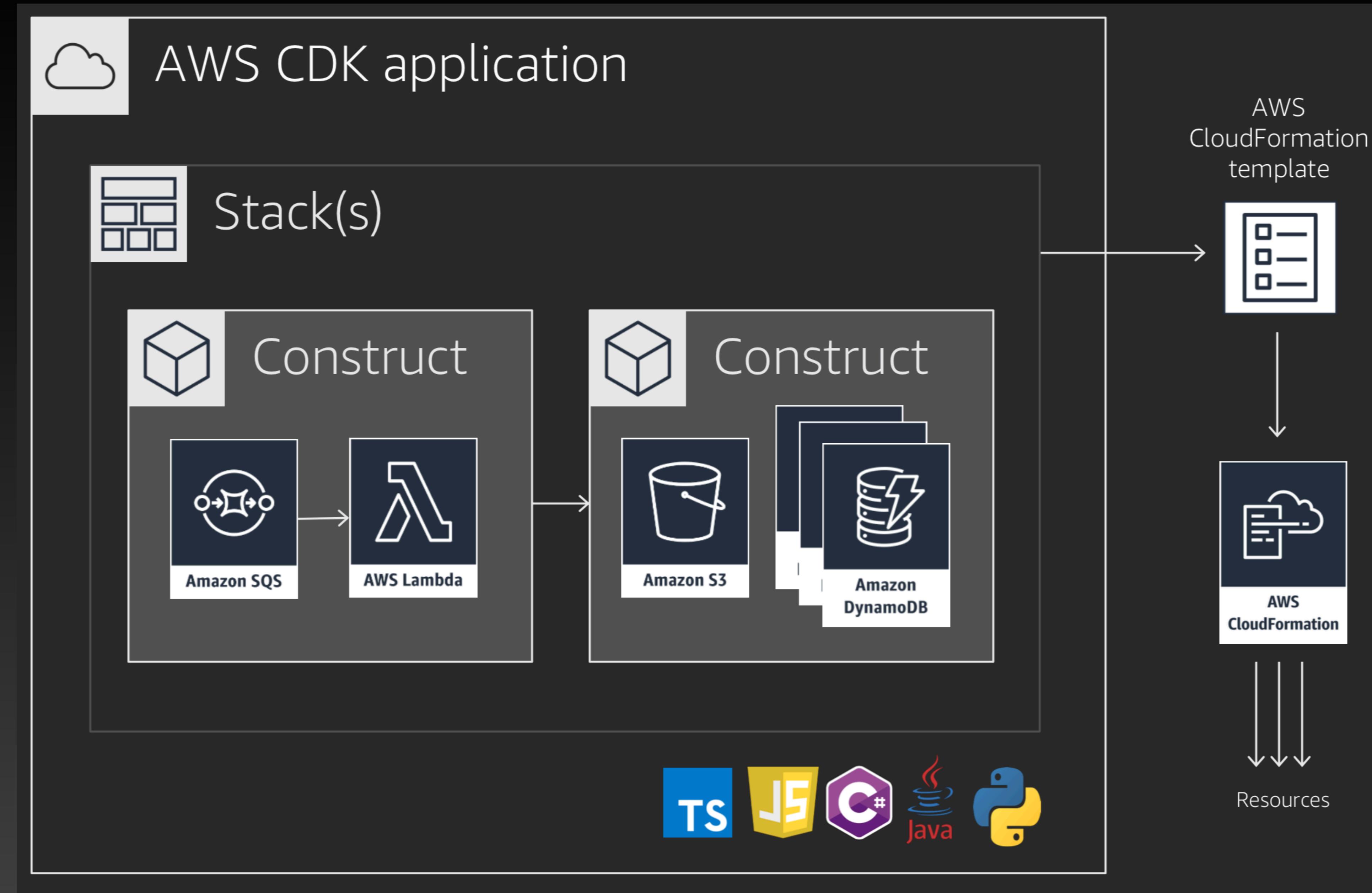
(bottom up)

- CloudFormation (Cfn)
- Resource Constructs
- Constructs (reusable)
- Stacks

# CDK structure

## (bottom up)

- CloudFormation (Cfn)
- Resource Constructs
- Constructs (reusable)
- Stacks
- Applications



# Basic stack demo

# CDK constructor interface (3 params)

# CDK constructor interface (3 params)

- Scope ( *app* for stacks, *this* for constructs )

# CDK constructor interface (3 params)

- Scope ( *app* for stacks, *this* for constructs )
- Id (unique within this scope)

# CDK constructor interface (3 params)

- Scope ( *app* for stacks, *this* for constructs )
- Id (unique within this scope)
- Properties (arguments)

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App,  
        id: string,  
        props?: props  
    ) {  
        super(scope, id);  
  
        new s3.Bucket(  
            this,  
            'MyFirstBucket',  
            {  
                bucketName: "demo-for-scl-2019",  
            }  
        );  
    }  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string,  
        props?: props  
    ) {  
        super(scope, id);
```

```
        new s3.Bucket(  
            this,  
            'MyFirstBucket',  
            {  
                bucketName: "demo-for-scl-2019",  
            }  
        );  
    }  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string, ← Id  
        props?: props  
    ) {  
        super(scope, id);
```

```
        new s3.Bucket(  
            this, ←  
            'MyFirstBucket',  
            {  
                bucketName: "demo-for-scl-2019",  
            }  
        );  
    }  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string, ← Id  
        props?: props ← Properties  
    ) {  
        super(scope, id);  
    }  
}
```

```
new s3.Bucket(  
    this, 'MyFirstBucket',  
    {  
        bucketName: "demo-for-scl-2019",  
    }  
);  
}  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string, ← Id  
        props?: props ← Properties  
    ) {  
        super(scope, id);  
    }  
}
```

```
new s3.Bucket(  
    this, ← Scope  
    'MyFirstBucket',  
    {  
        bucketName: "demo-for-scl-2019",  
    }  
);  
}  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string, ← Id  
        props?: props ← Properties  
    ) {  
        super(scope, id);  
    }  
}
```

```
new s3.Bucket(  
    this, ← Scope  
    'MyFirstBucket', ← Id  
    {  
        bucketName: "demo-for-scl-2019",  
    }  
);  
}  
}
```

```
export class HelloCdkStack extends cdk.Stack {  
    constructor(  
        scope: cdk.App, ← Scope  
        id: string, ← Id  
        props?: props ← Properties  
    ) {  
        super(scope, id);  
    }  
}
```

```
new s3.Bucket(  
    this, ← Scope  
    'MyFirstBucket', ← Id  
    { ← Properties  
        bucketName: "demo-for-scl-2019",  
    }  
);  
}  
}
```

API Reference  
( [https://docs.aws.amazon.com/  
cdk/api/latest/](https://docs.aws.amazon.com/cdk/api/latest/) )

AWS CDK · AWS CDK Reference Documentation

docs.aws.amazon.com/cdk/api/latest/

AWS CDK 1.14.0

API Reference Python TypeScript Java .NET Developer Guide Examples

# AWS CDK

## AWS CDK Reference Documentation

[API REFERENCE](#) [DEVELOPER GUIDE](#)

**Docs**

AWS Construct Library

**Community**

Stack Overflow  
Project Chat (Gitter)

**More**

GitHub

Copyright © 2019 Amazon Web Services

aws-s3 module · AWS CDK

docs.aws.amazon.com/cdk/api/latest/docs/aws-s3-readme.html

## AWS CDK 1.14.0

API Reference Python TypeScript Java .NET Developer Guide Examples

### aws-s3

- Overview
- Constructs
  - Bucket
  - BucketPolicy
- Classes
  - BlockPublicAccess
  - ReplaceKey
  - StorageClass
- Structs
  - BlockPublicAccessOptions
  - BucketAttributes
  - BucketMetrics
  - BucketNotificationDestinationConfig
  - BucketPolicyProps
  - BucketProps
  - CorsRule
  - LifecycleRule
  - Location
  - NoncurrentVersionTransition
  - NotificationKeyFilter
  - OnCloudTrailBucketEventOptions
  - RedirectTarget

# aws-s3 module

Language	Package
Python	aws_cdk.aws_s3
Java	software.amazon.awscdk.services.
.NET	Amazon.CDK.AWS.S3
TypeScript	@aws-cdk/aws-s3

## Amazon S3 Construct Library

STABILITY STABLE

Define an unencrypted S3 bucket.

```
new Bucket(this, 'MyFirstBucket');
```

(Example not in your language? Click here.)

Bucket constructs expose the following deploy-time attributes:

[https://docs.aws.amazon.com/cdk/api/latest/docs/@aws-cdk\\_aws-s3.BucketMetrics.html](https://docs.aws.amazon.com/cdk/api/latest/docs/@aws-cdk_aws-s3.BucketMetrics.html)

2019\_0911-CO....pdf

Show All

Amazon S3 Construct Library

- Encryption
- Permissions
- Sharing buckets between stacks
- Importing existing buckets
- Bucket Notifications
- Block Public Access
- Website redirection
- Filling the bucket as part of deployment

aws-s3 module · AWS CDK

docs.aws.amazon.com/cdk/api/latest/docs/aws-s3-readme.html

## AWS CDK 1.14.0

### aws-s3

Overview

Constructs

- Bucket
- BucketPolicy

Classes

- BlockPublicAccess
- ReplaceKey
- StorageClass

Structs

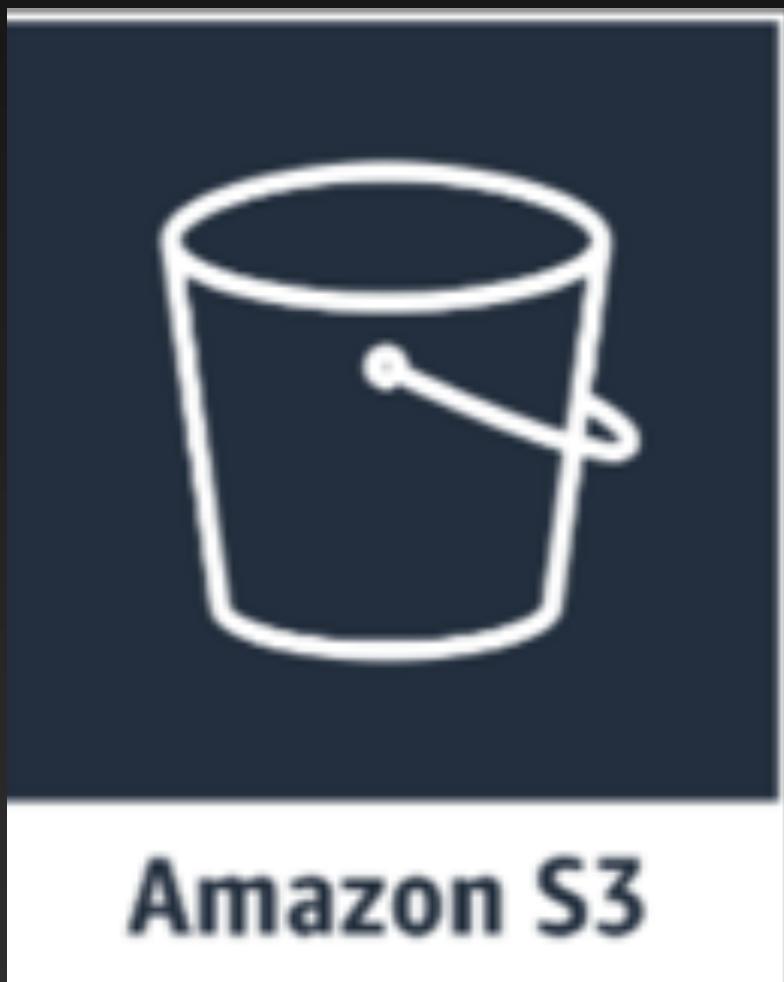
- BlockPublicAccessOptions

# aws-s3 module

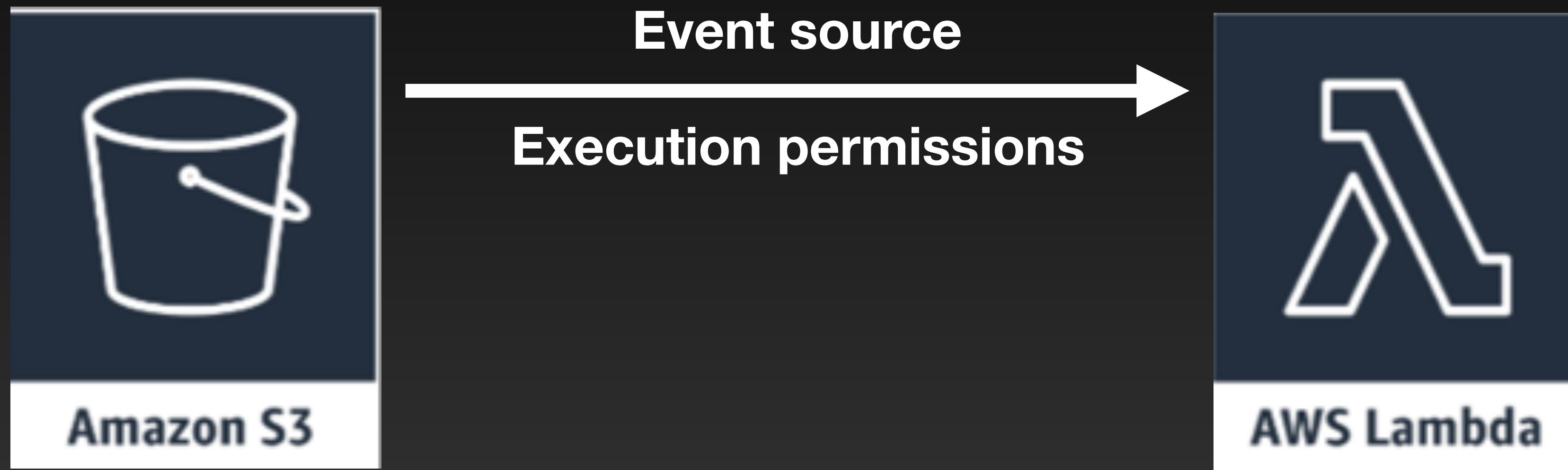
Language	Package
Python	aws_cdk.aws_s3
Java	software.amazon.awscdk.services.s3
.NET	Amazon.CDK.AWS.S3
TypeScript	@aws-cdk/aws-s3

# Events & Permissions

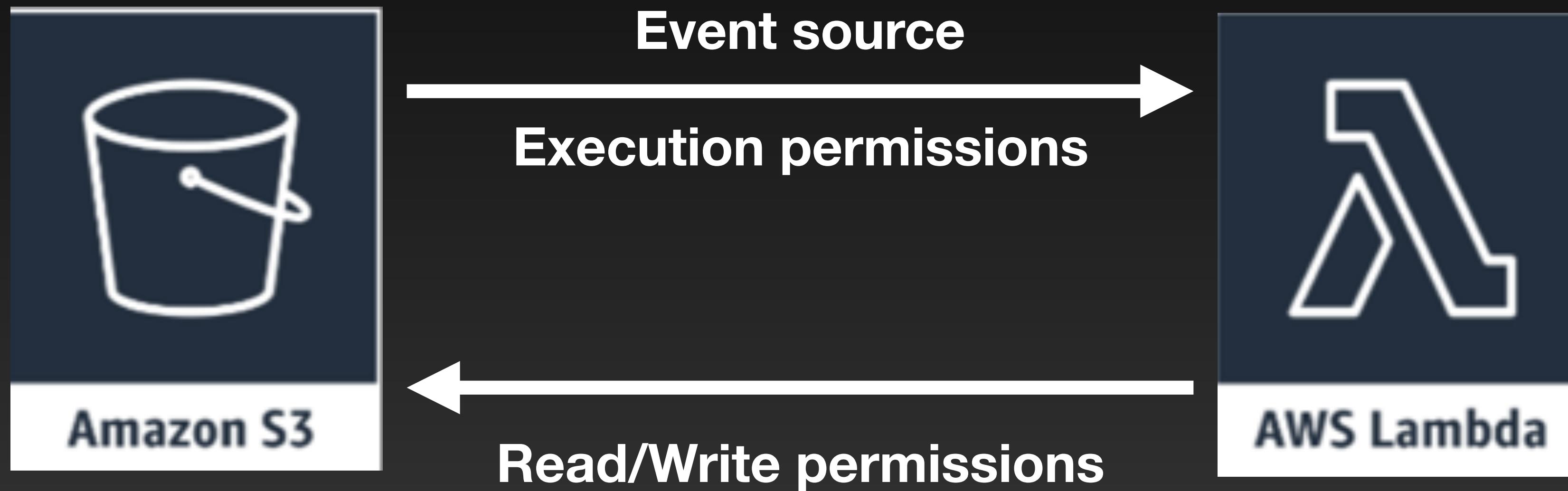
# S3 event source



# S3 event source



# S3 event source



```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');
import lambda = require('@aws-cdk/aws-lambda');
import { S3EventSource } from '@aws-cdk/aws-lambda-event-sources';

export class EventS3LambdaStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // object store
    const myBucket = new s3.Bucket(this, 'aBucket', {});

    // create an event source for new objects
    let bucketEvents = new S3EventSource(myBucket, {
      events: [ s3.EventType.OBJECT_CREATED ],
    });

    // A function to process the S3 event
    const fn = new lambda.Function(this, 'EchoS3', {
      runtime: lambda.Runtime.NODEJS_8_10,
      handler: 'index.handler',
      code: lambda.Code.fromAsset('./lib/funcs/echo_s3/'),
      events: [bucketEvents],
    });

    // Remember to let the function read/write to the
    // bucket if it needs more than just the object key
    myBucket.grantReadWrite(fn);
  }
}
```



```
import cdk =
```

```
import cdk =  
  require( '@aws-cdk/core' );
```

```
import cdk =  
  require( '@aws-cdk/core' );
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =  
  require('@aws-cdk/aws-lambda');
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =  
  require('@aws-cdk/aws-lambda');
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =  
  require('@aws-cdk/aws-lambda');
```

```
import { S3EventSource }
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =  
  require('@aws-cdk/aws-lambda');
```

```
import { S3EventSource }  
from
```

```
import cdk =  
  require('@aws-cdk/core');
```

```
import s3 =  
  require('@aws-cdk/aws-s3');
```

```
import lambda =  
  require('@aws-cdk/aws-lambda');
```

```
import { S3EventSource }  
from  
  '@aws-cdk/aws-lambda-event-sources';
```





```
// object store  
const myBucket = new s3.Bucket(
```

```
// object store
const myBucket = new s3.Bucket(
  this,
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
  {}
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
  {}
) ;
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
  {}
) ;
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
  {}
) ;
```

```
// object store
const myBucket = new s3.Bucket(
  this,
  'aBucket',
  {}
);
```

```
// create an event source
let bucketEvents = new S3EventSource(
  myBucket, {
    events: [
      s3.EventType.OBJECT_CREATED
    ],
  }
);
```





```
// A function to process the S3 event
const fn = new lambda.Function(
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
{
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
{
  runtime: lambda.Runtime.NODEJS_8_10,
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
    handler: 'index.handler',
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
    handler: 'index.handler',
    events: [bucketEvents],
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
    handler: 'index.handler',
    events: [bucketEvents],
  });
});
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
    handler: 'index.handler',
    events: [bucketEvents],
  });
});
```

```
// A function to process the S3 event
const fn = new lambda.Function(
  this,
  'EchoS3',
  {
    runtime: lambda.Runtime.NODEJS_8_10,
    code: lambda.Code.fromAsset(
      './lib/funcs/echo_s3/'),
    handler: 'index.handler',
    events: [bucketEvents],
  });

```

```
// Give function read/write to the
// bucket if it is needed
myBucket.grantReadWrite(fn);
```

```
import cdk = require('@aws-cdk/core');
import s3 = require('@aws-cdk/aws-s3');
import lambda = require('@aws-cdk/aws-lambda');
import { S3EventSource } from '@aws-cdk/aws-lambda-event-sources';

export class EventS3LambdaStack extends cdk.Stack {
  constructor(scope: cdk.Construct, id: string, props?: cdk.StackProps) {
    super(scope, id, props);

    // object store
    const myBucket = new s3.Bucket(this, 'LaBucket', {});

    // create an event source for new objects
    let bucketEvent = new S3EventSource(myBucket, {
      events: [ s3.EventType.OBJECT_CREATED ],
    });

    // A function to process the S3 event
    const fn = new lambda.Function(this, 'EchoS3', {
      runtime: lambda.Runtime.NODEJS_8_10,
      handler: 'index.handler',
      code: lambda.Code.fromAsset('./lib/funcs/echo_s3/'),
      events: [bucketEvent],
    });

    // Remember to let the function read/write to the
    // bucket if it needs more than just the object key
    myBucket.grantReadWrite(fn);
  }
}
```

30 lines of code

CDK synth



```
Resources:
  aBucket43B84104:
    Type: AWS::S3::Bucket
    UpdateReplacePolicy: Retain
    DeletionPolicy: Retain
    Metadata:
      aws:cdk:path: Events$LambdaStack/aBucket/Resource
  aBucketNotificationsC5696022:
    Type: Custom::S3BucketNotifications
    Properties:
      ServiceToken:
        Fn::GetAtt:
          - BucketNotificationsHandler050a0587b7544547bf325f094a
          - Arn
      BucketName:
        Ref: aBucket43B84104
      NotificationConfiguration:
        LambdaFunctionConfigurations:
          - Events:
              - s3:ObjectCreated:*
```

CloudFormation YAML  
250 lines

# Full s3/lambda demo

# Other Lambda event sources?

# Other Lambda event sources?

- SQS

# Other Lambda event sources?

- SQS
- SNS

# Other Lambda event sources?

- SQS
- SNS
- DynamoDB Streams

# Other Lambda event sources?

- SQS
- SNS
- DynamoDB Streams
- Kinesis

# Other Lambda event sources?

- SQS
- SNS
- DynamoDB Streams
- Kinesis
- More planned

# HTTP event source with ApiGateway

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");  
  
const api = new apigateway.RestApi(this, "itemsApi", {
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");  
  
const api = new apigateway.RestApi(this, "itemsApi", {  
  restApiName: "Items Service"
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");

const lambdaIntergration
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");

const lambdaIntergration
  = new apigateway.LambdaIntegration(aLambdaObject);
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");

const lambdaIntergration
  = new apigateway.LambdaIntegration(aLambdaObject);
```

## HTTP event source with ApiGateway

```
const apigateway = require("@aws-cdk/aws-apigateway");

const api = new apigateway.RestApi(this, "itemsApi", {
  restApiName: "Items Service"
});

const items = api.root.addResource("items");

const lambdaIntergration
  = new apigateway.LambdaIntegration(aLambdaObject);

items.addMethod("GET", lambdaIntergration);
```

# Tips & tricks

# Stack base class

```
import cdk = require('@aws-cdk/core');

export interface IEnv {
  region: string,
  account: string,
}

export interface IBaseStack {
  env: IEnv,
  stackName: string,
}

export class BaseStack extends cdk.Stack {

  constructor(scope: cdk.Construct, id: string, props: IBaseStack) {
    let superProps = {
      env: props.env,
      stackName: props.stackName
    }
    super(scope, id, superProps);

    // Tag everything in the stack with the stack name
    cdk.Tag.add(this, 'stack', props.stackName);
  }
}
```

```
import cdk = require('@aws-cdk/core');

export interface IEnv {
  region: string,
  account: string,
}

export interface IBaseStack {
  env: IEnv,
  stackName: string,
}

export class BaseStack extends cdk.Stack {

  constructor(scope: cdk.Construct, id: string, props: IBaseStack) {
    let superProps = {
      env: props.env,
      stackName: props.stackName
    }
    super(scope, id, superProps);

    // Tag everything in the stack with the stack name
    cdk.Tag.add(this, 'stack', props.stackName);
  }
}
```



```
import cdk = require('@aws-cdk/core');

export interface IEnv {
  region: string,
  account: string,
}

export interface IBaseStack {
  env: IEnv,
  stackName: string,
}

export class BaseStack extends cdk.Stack {

  constructor(scope: cdk.Construct, id: string, props: IBaseStack) {
    let superProps = {
      env: props.env,
      stackName: props.stackName
    }
    super(scope, id, superProps);

    // Tag everything in the stack with the stack name
    cdk.Tag.add(this, 'stack', props.stackName);
  }
}
```

# Importing existing resources



```
const bucket = s3.Bucket.fromBucketAttributes(
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
);
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
);
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
) ;
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
) ;
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this, "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
) ;
```

```
// now you can just call methods on the bucket
```

```
const bucket = s3.Bucket.fromBucketAttributes(  
  this,  
  "ImportedBucket",  
  {  
    bucketArn: "arn:aws:s3:::my-bucket"  
  }  
) ;
```

```
// now you can just call methods on the bucket  
bucket.grantReadWrite(lambdaFunction);
```

# Misc

# Misc

- **Stack.of(construct)** - access stack at any point (useful if you need stack.region)

# Misc

- **Stack.of(construct)** - access stack at any point (useful if you need stack.region)
- Do not edit *anything* through web UI

# Local development (using SAM)

# Serverless Application Model (SAM)

# Serverless Application Model (SAM)

- Serverless YAML shortcuts for CloudFormation YAML (not as nice as CDK IMHO!)

# Serverless Application Model (SAM)

- Serverless YAML shortcuts for CloudFormation YAML (not as nice as CDK IMHO!)
- Local testing and debugging

# Setup for local testing

# Setup for local testing

- `cdk synth --no-staging > template.yaml`

# Setup for local testing

- `cdk synth --no-staging > template.yaml`
- Find lambda resource name

# Setup for local testing

- `cdk synth --no-staging > template.yaml`
- Find lambda resource name
- Create **payload.json** (`sam local generate-event s3`) and **env.json**

# Setup for local testing

- **cdk synth –no-staging > template.yaml**
- Find lambda resource name
- Create **payload.json (sam local generate-event s3)** and **env.json**
- **sam local invoke –env-var env.json –event payload.json lambdaResourceName**

# Local invoke demo

# Lambda Layers

# Layer

# Layer

- Runtime (OS + Language)

# Layer

- Runtime (OS + Language)
- Your code (**with all** dependencies)

# Layer

# Layer

- Runtime (OS + Language)

# Layer

- Runtime (OS + Language)
- Shared dependencies (in a Layer)

# Layer

- Runtime (OS + Language)
- Shared dependencies (in a Layer)
- Your code

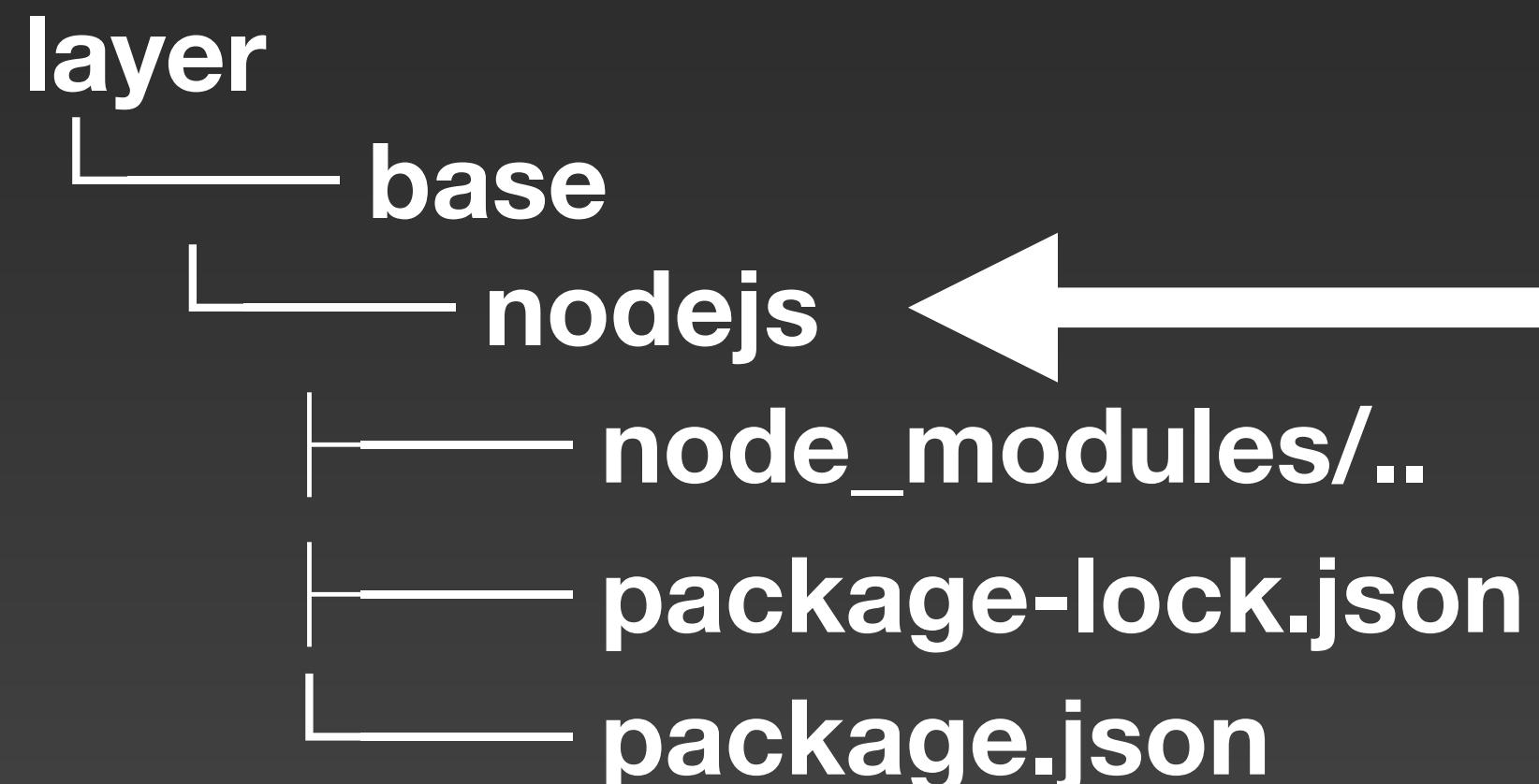
# Creating a layer

```
// Create a layer
const baseLayer = new lambda.LayerVersion(
  this,
  'MyBaseLayer',
  {
    compatibleRuntimes: [
      lambda.Runtime.NODEJS_8_10
    ],
    code:
      lambda.Code.fromAsset('..../layer/base')
  }
);
```

```
// Create a layer
const baseLayer = new lambda.LayerVersion(
  this,
  'MyBaseLayer',
  {
    compatibleRuntimes: [
      lambda.Runtime.NODEJS_8_10
    ],
    code:
      lambda.Code.fromAsset('../layer/base')
  }
);
```

```
layer
└── base
  └── nodejs
    ├── node_modules/..
    ├── package-lock.json
    └── package.json
```

```
// Create a layer
const baseLayer = new lambda.LayerVersion(
  this,
  'MyBaseLayer',
  {
    compatibleRuntimes: [
      lambda.Runtime.NODEJS_8_10
    ],
    code:
      lambda.Code.fromAsset('../layer/base')
  }
);
```



# CDK: Chicken & Egg

- Layer has ARN (Amazon Resource Name)
- Lambda depends on ARN
- CDK: Updating layer deletes old ARN...  
which breaks dependency across stacks

# ParamStore to the rescue







```
// Record the versionArn into SSM
```

```
const layerParamName =
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    description: 'The ARN of the Lambda Layer Version to use'  
  })
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,  
    stringValue: [REDACTED]
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,  
    stringValue:  
      baseLayer.layerVersionArn,
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,  
    stringValue:  
      baseLayer.layerVersionArn,  
  } );
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,  
    stringValue:  
      baseLayer.layerVersionArn,  
  } );
```

```
// Record the versionArn into SSM
```

```
const layerParamName =  
  '/layers/baseLayer';
```

```
new ssm.StringParameter(  
  this,  
  'VersionArn',  
  {  
    parameterName: layerParamName,  
    stringValue:  
      baseLayer.layerVersionArn,  
  } );
```



```
import cdk = require('@aws-cdk/core');
import lambda = require('@aws-cdk/aws-lambda');
import ssm = require('@aws-cdk/aws-ssm');
```

```
const layerParamName = '/layers/baseLayer';
```

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

```
// Create a layer
```

```
const baseLayer = new lambda.LayerVersion(
```

```
this,
```

```
'MyBaseLayer',
```

```
{
```

```
  compatibleRuntimes: [
```

```
    lambda.Runtime.NODEJS_8_10
  ],
  code: lambda.Code.fromAsset('../layer/base')
```

```
}
```

```
);
```

```
// Record the versionArn into SSM
```

```
new ssm.StringParameter(this, 'VersionArn', {
```

```
  parameterName: layerParamName,
```

```
  stringValue: baseLayer.layerVersionArn,
```

```
});
```

```
}
```

```
import cdk = require('@aws-cdk/core');
import lambda = require('@aws-cdk/aws-lambda');
import ssm = require('@aws-cdk/aws-ssm');
```

```
const layerParamName = '/layers/baseLayer';
```

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

```
// Create a layer
```

```
const baseLayer = new lambda.LayerVersion(
```

```
this,
```

```
' MyBaseLayer ',
```

```
{
```

```
compatibleRuntimes: [
```

```
lambda.Runtime.NODEJS_8_10
```

```
],
```

```
code: lambda.Code.fromAsset('../layer/base')
```

```
}
```

```
);
```

```
// Record the versionArn into SSM
```

```
new ssm.StringParameter(this, 'VersionArn', {
```

```
parameterName: layerParamName,
```

```
stringValue: baseLayer.layerVersionArn,
```

```
});
```

```
}
```

Using the layer  
(in another stack)



```
const layerParamName = "/layers/baseLayer";
```

```
const layerParamName = "/layers/baseLayer";
```

```
// fetch the Arn from param store
const baseLayerArn =
  ssm.StringParameter.valueForStringParameter(
    this,
    layerParamName
  );
```

```
const layerParamName = "/layers/baseLayer";
```

```
// fetch the Arn from param store
const baseLayerArn =
  ssm.StringParameter.valueForStringParameter(
    this,
    layerParamName
  );
```

```
// generate layer version from Arn
const layer1 =
  lambda.LayerVersion.fromLayerVersionArn(
    this,
    "BaseLayerFromArn",
    baseLayerArn
  );
```

```
// Then supply when you create a lambda
new lambda.Function(this, "SomeName", {
    runtime: lambda.Runtime.NODEJS_8_10,
    code:
        lambda.Code.fromAsset(
            "../lambda/aFunction"),
    handler: "index.handler",
    layers: [layer1]
}) ;
```

```
// Then supply when you create a lambda
new lambda.Function(this, "SomeName", {
    runtime: lambda.Runtime.NODEJS_8_10,
    code:
        lambda.Code.fromAsset(
            "../lambda/aFunction"),
    handler: "index.handler",
    layers: [layer1]
}) ;
```



Supply the Lambda function with the code and handler information.



**Ben Kehoe**  
@ben11kehoe



This is interesting CFN behavior. While there are paths to fixing it, I think it furthers the point that layers should only be used when their lifecycles are independent of the functions that use them. Deploying a function and a layer it uses in the same stack is a code smell.

# Balance



**Ben Kehoe** @ben11kehoe · Oct 12

Not understanding the resource graph you are creating is a recipe for increased operations burden.





**Ben Kehoe** @ben11kehoe · Oct 25



Replies to [@rchrdbdyd](#) and [@thomasphorton](#)

Before diving into the CDK, I would recommend watching my recent talk on why it is not the panacea it may appear to be. Approach it with a thoughtful mindset!



[Serverlessconf New York 2019: YAML Is Better than ...](#)

Serverless is service-full, which means you've got more complicated cloud infrastructure graphs to ...

🔗 [acloud.guru](#)



1



2



4



# YAML Is Better than Your Favorite Language: Fightin' words about Infrastructure as code | Ben Kehoe

[https://acloud.guru/series/  
serverlessconf-nyc-2019/view/  
yaml-better](https://acloud.guru/series/serverlessconf-nyc-2019/view/yaml-better)

# Review

# Review

# Review

- IaC: **repeatable, versionable, speed of deploy**

# Review

- IaC: repeatable, versionable, speed of deploy
- CloudFormation: Infrastructure Graph Engine
  - learn to read it

# Review

- **IaC: repeatable, versionable, speed of deploy**
- **CloudFormation: Infrastructure Graph Engine**
  - learn to read it
- **CDK: apps, stacks, constructs**

# Review

- **IaC: repeatable, versionable, speed of deploy**
- **CloudFormation: Infrastructure Graph Engine**
  - learn to read it
- **CDK: apps, stacks, constructs**
- **Local dev and Lambda Layers**

# Review

- **IaC: repeatable, versionable, speed of deploy**
- **CloudFormation: Infrastructure Graph Engine**
  - learn to read it
- **CDK: apps, stacks, constructs**
- **Local dev and Lambda Layers**
- **A bunch of other bits that I hope are useful**

# The end

- Slides online later
- Contact / follow: @leolapworth
- Thank you