AWS re:Invent

DEV372

Infrastructure Is Code with the **AWS Cloud Development Kit**

Elad Ben-Israel
Principal Engineer
AWS Developer Tools

Jason Fulghum
Development Manager
AWS Developer Tools

AWS re:Invent

aws

Agenda

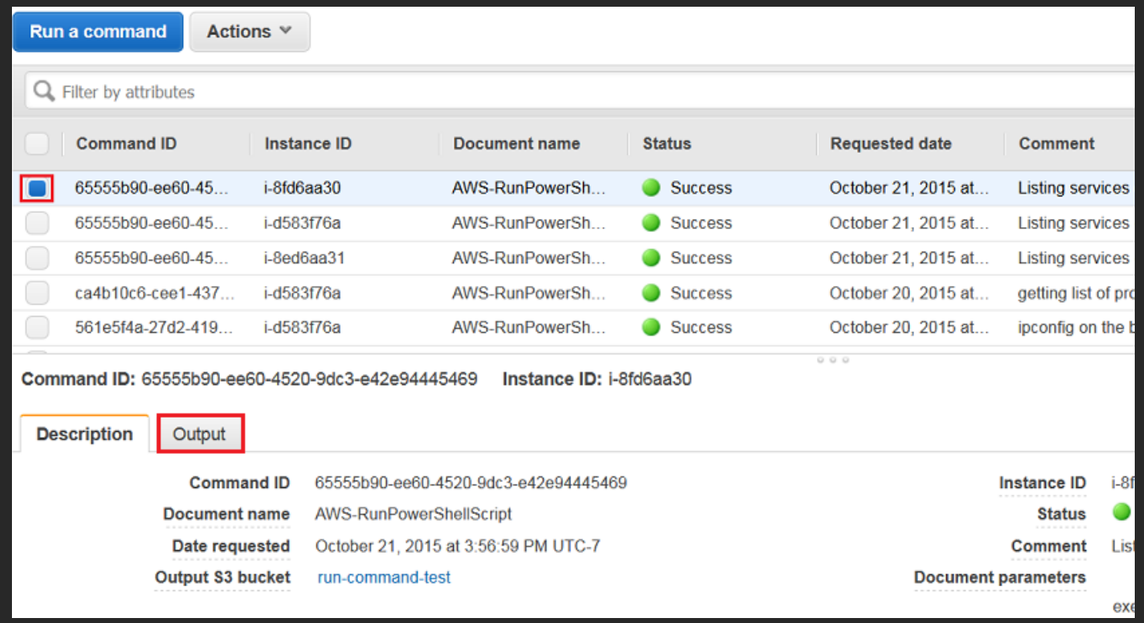
- Our infrastructure management journey
- The AWS Cloud Development Kit (AWS CDK)
- Demo: Build an AWS CDK app
- A lot more to explore

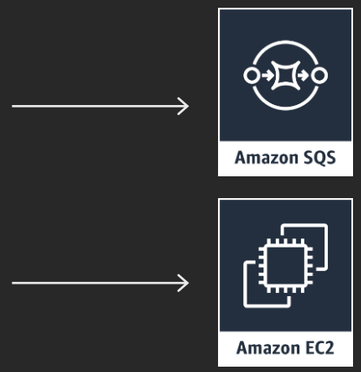
Our infrastructure management journey

AWS re:Invent

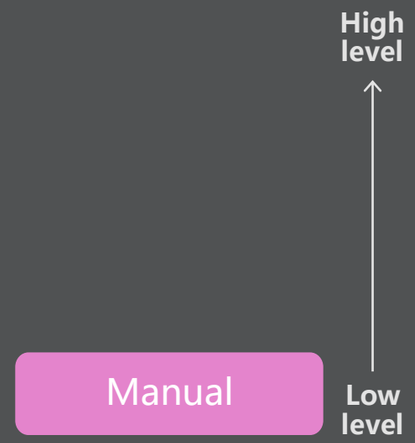
aws

Manual



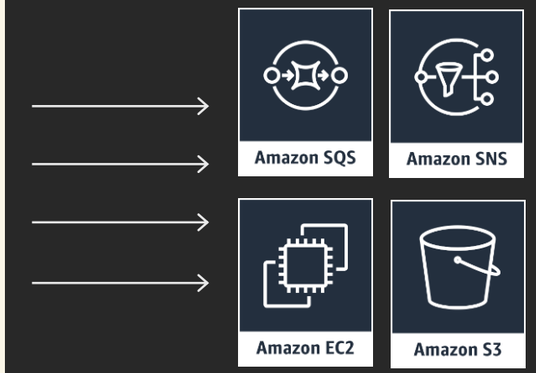


Easy to get started
Not reproducible
Error prone
Time consuming

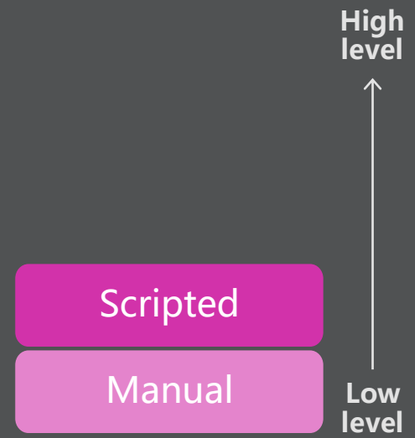


Scripted

```ruby
require 'aws-sdk-ec2'

ec2 = Aws::EC2::Resource.new(region: 'us-west-2')

instance = ec2.create_instances({
  image_id: 'IMAGE_ID',
  min_count: 1,
  max_count: 1,
  key_name: 'MyGroovyKeyPair',
  security_group_ids: ['SECURITY_GROUP_ID'],
  instance_type: 't2.micro',
  placement: {
    availability_zone: 'us-west-2a'
  },
  subnet_id: 'SUBNET_ID',
  iam_instance_profile: {
    arn: 'arn:aws:iam::' + 'ACCOUNT_ID' + ':instance-profile/aws-opsworks-ec2-role'
  }
})
```

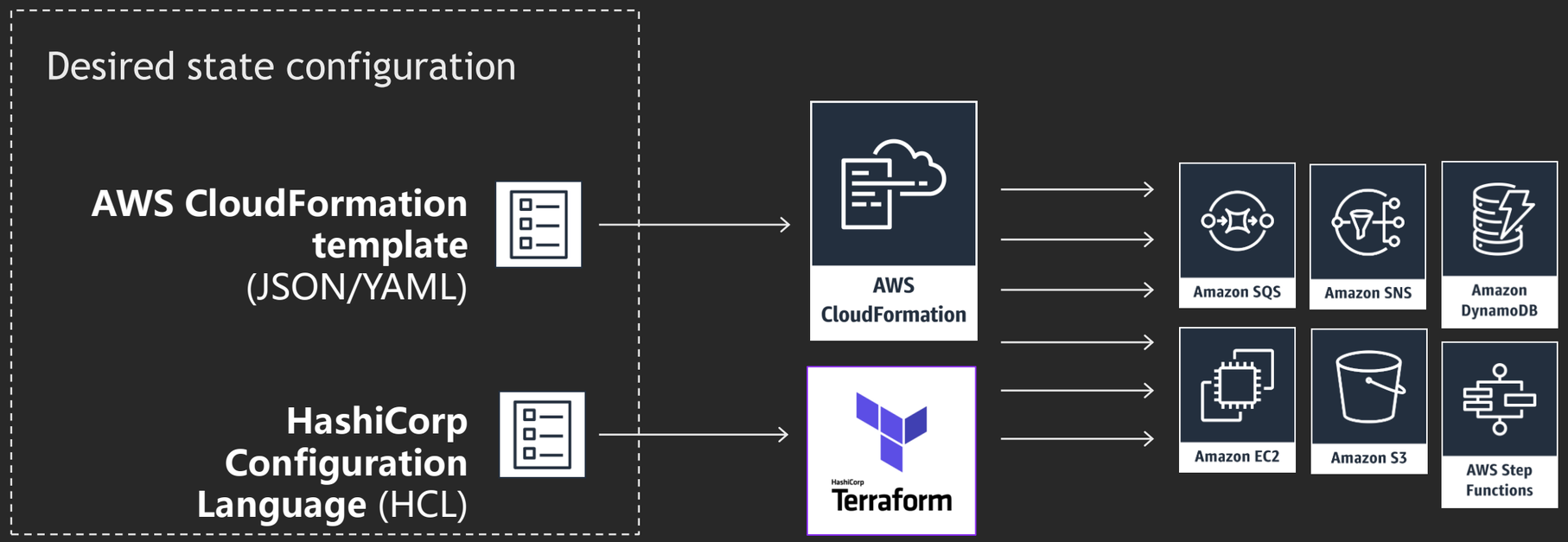


What happens if an API call fails?
How do I make updates?
How do I know a resource is ready?
How do I roll back?

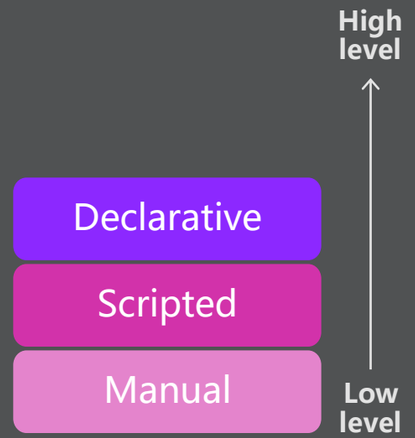


aws

Resource provisioning engines



Easy to automate
Reproducible
Configuration syntax
No abstraction, lots of details

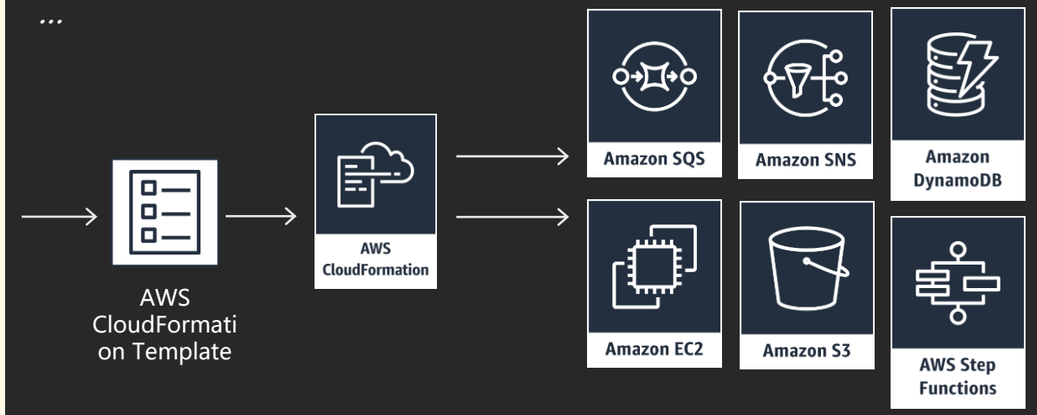


aws

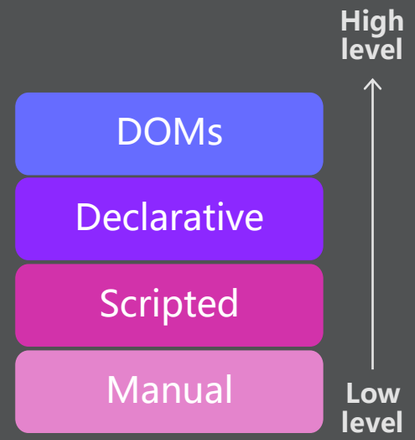
Document Object Models (DOMs)

```python
from troposphere import Template
from troposphere.ec2 import VPC, Subnet, InternetGateway

t = Template()

VPC = t.add_resource(
    VPC(
        'VPC',
        CidrBlock='10.0.0.0/16',
        Tags=Tags(
            Application=ref_stack_id)))

subnet = t.add_resource(
    Subnet(
        'Subnet',
        CidrBlock='10.0.0.0/24',
        VpcId=Ref(VPC),
        Tags=Tags(
            Application=ref_stack_id)))
```

Troposphere *Python*
SparkleFormation *Ruby*
GoFormation *Go*
...



Real code ♥ *if statements, for loops, IDE benefits*
Desired state
Abstraction is not built-in *Ex: 218 lines of Troposphere for a VPC*



aws

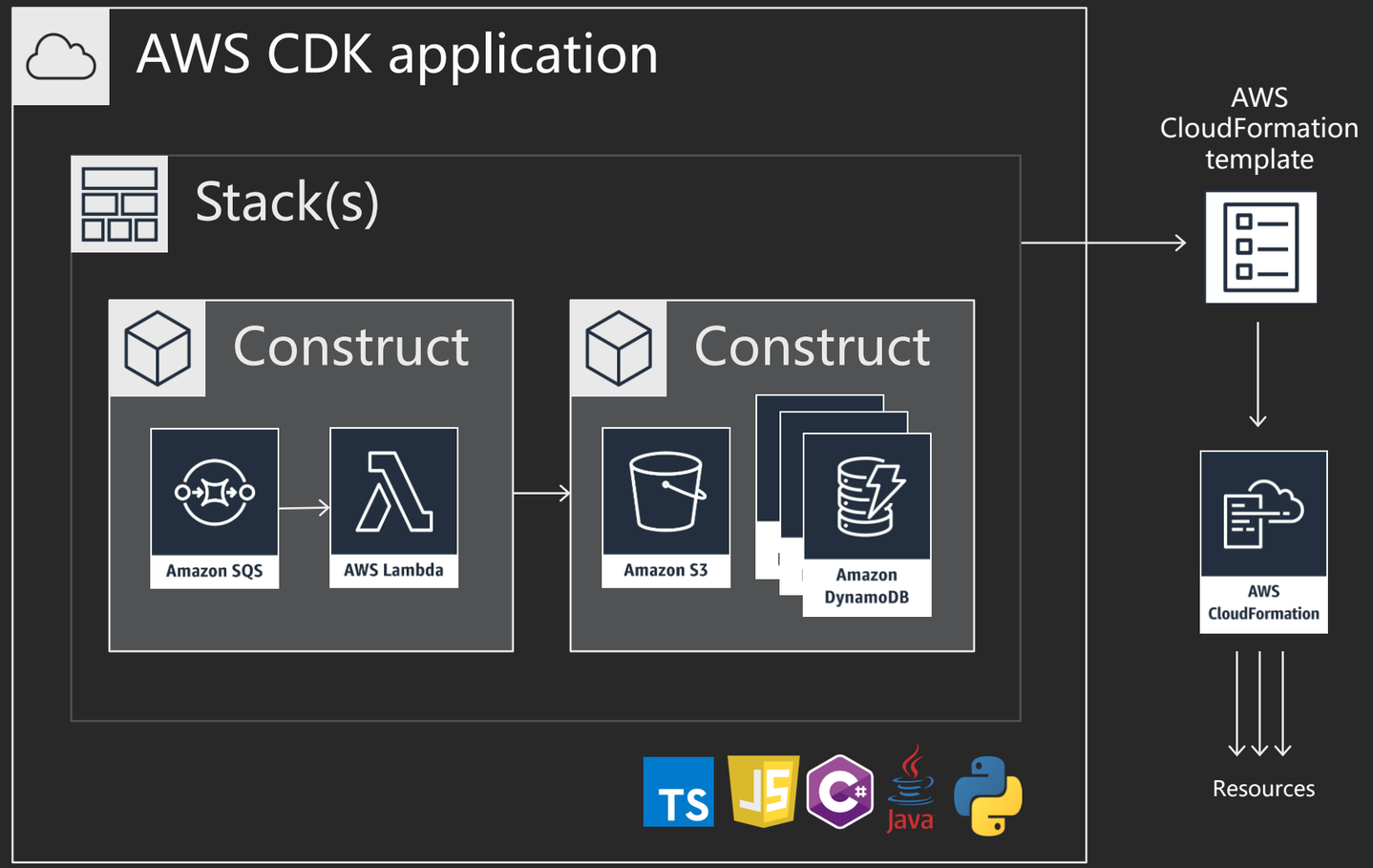
AWS Cloud Development Ki
(AWS CDK)

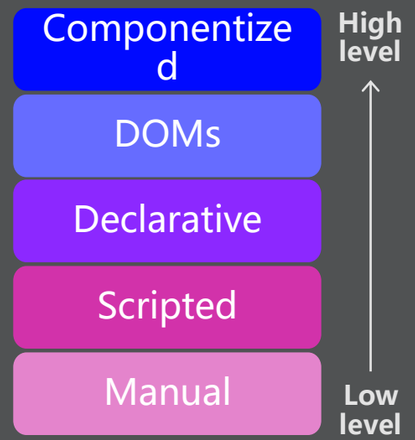
AWS re:Invent

aws

AWS CDK

Developer preview

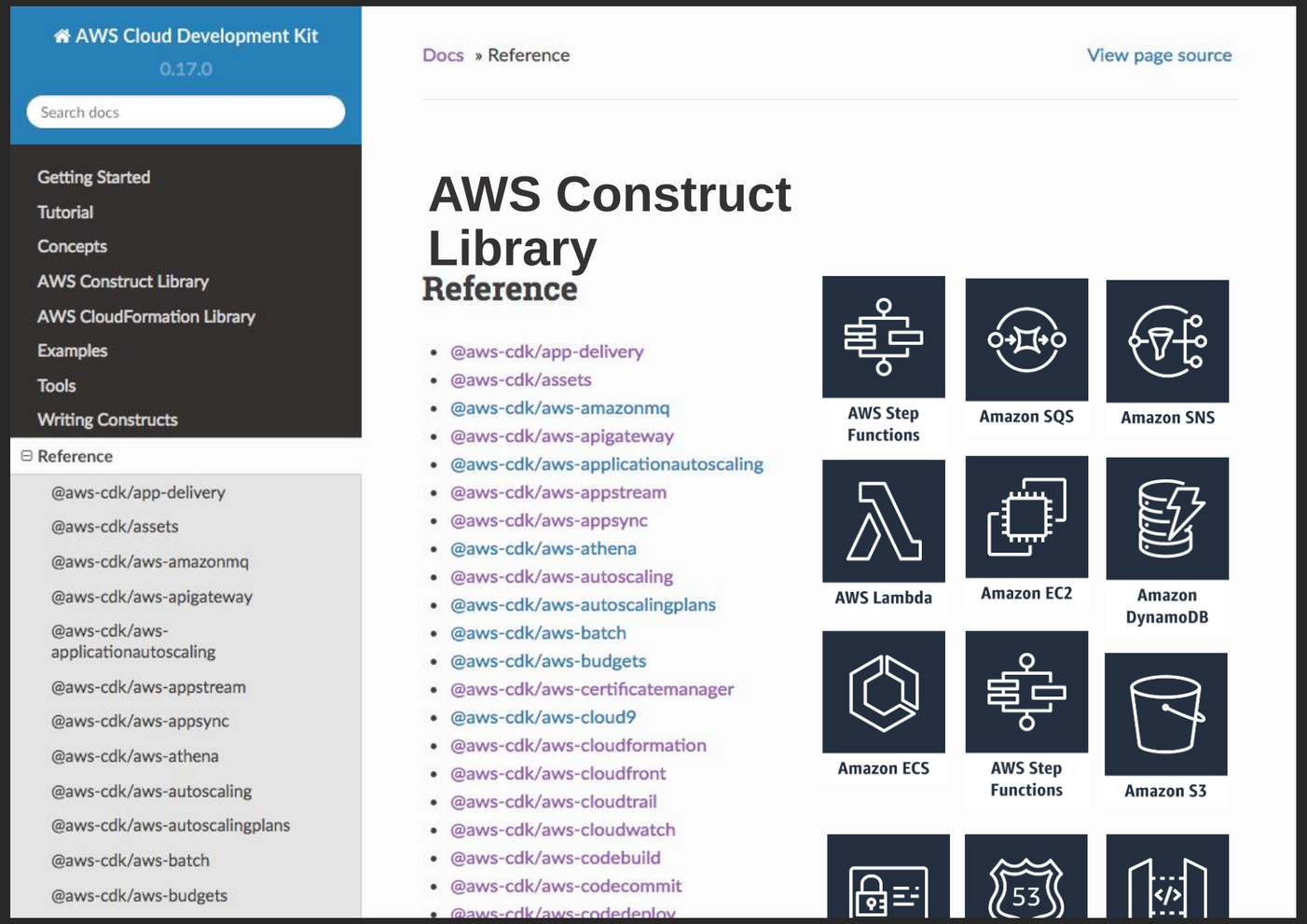


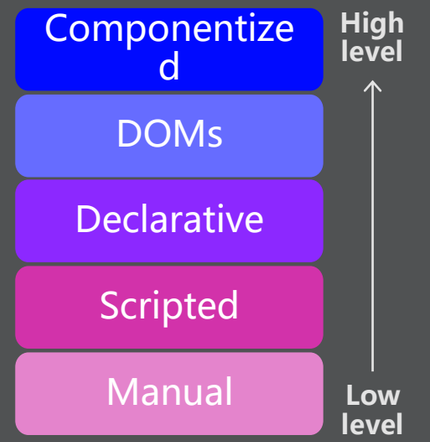


aws

AWS CDK

Developer preview





aws

AWS CDK

Developer preview

README.m

Fork me on GitHub

S Cloud Development Kit (AWS CDK)

AWS CodeBuild passing | build passing | npm package 0.13.0 | chat on gitter

The **AWS Cloud Development Kit (AWS CDK)** is an open-source software development framework to define cloud infrastructure in code and provision it through AWS CloudFormation. The CDK integrates fully with AWS services and offers a higher level object-oriented abstraction to define AWS resources imperatively. Using the CDK's library of infrastructure constructs, you can easily encapsulate AWS best practices in your infrastructure definition and share it without worrying about boilerplate logic. The CDK improves the end-to-end development experience because you get to use the power of modern programming languages to define your AWS infrastructure in a predictable and efficient manner.

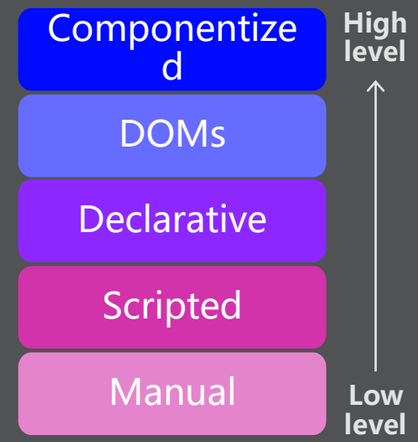
The following languages are currently supported:

- Java
- JavaScript, TypeScript
- .NET

The CDK is currently in developer preview and we look forward to community feedback and collaboration!

User Guide | Getting Started | API Reference | Getting Help

```
huijbers ~/aws-cdk> npm install -g aws-cdk
/usr/local/bin/cdk -> /usr/local/lib/node_modules/aws-cdk/bin/cdk
+ aws-cdk@0.7.4-beta
updated 1 package in 16.203s
```



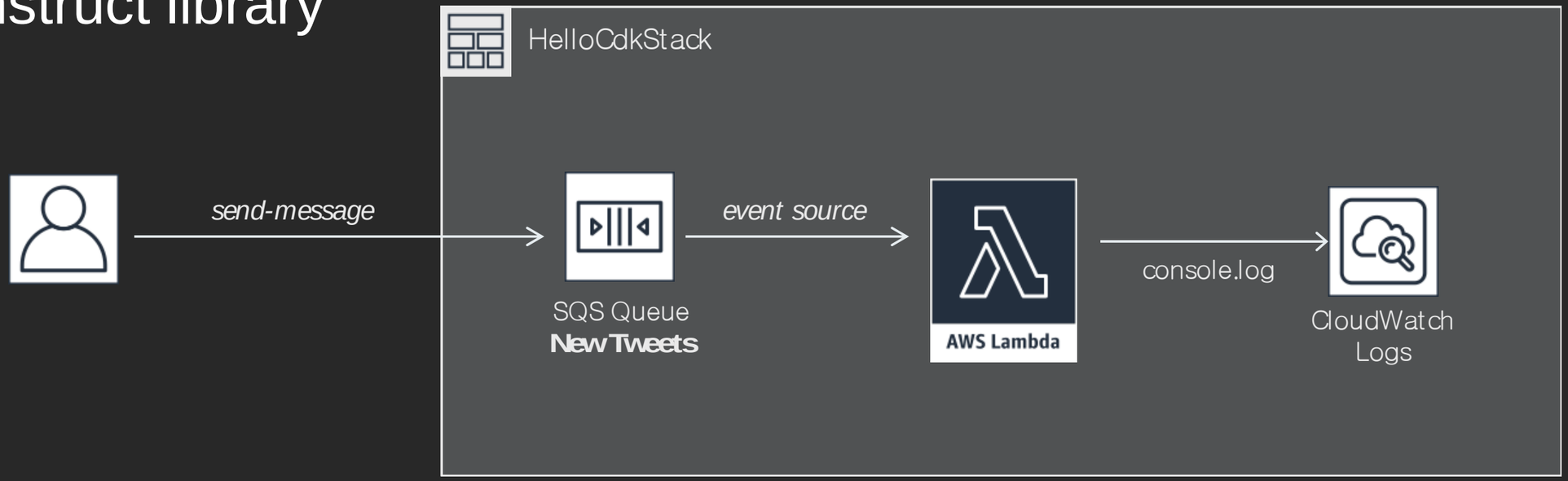
Demo: Build an AWS CDK app

AWS re:Invent

aws

Demo – Goals

- CDK concepts
- CDK CLI workflow
- AWS construct library

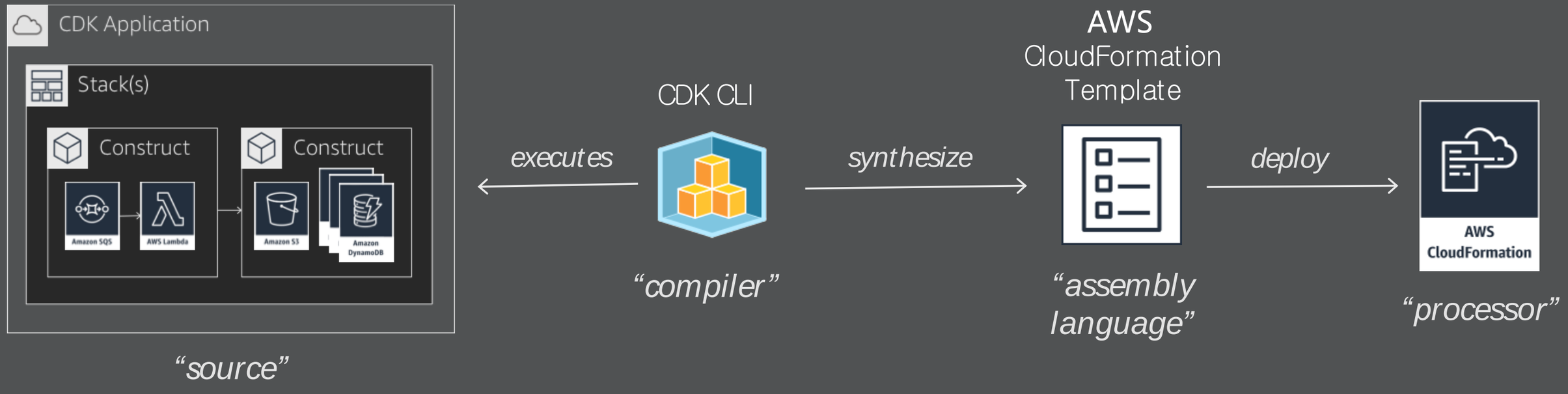


Demo

Demo recap

Get started:

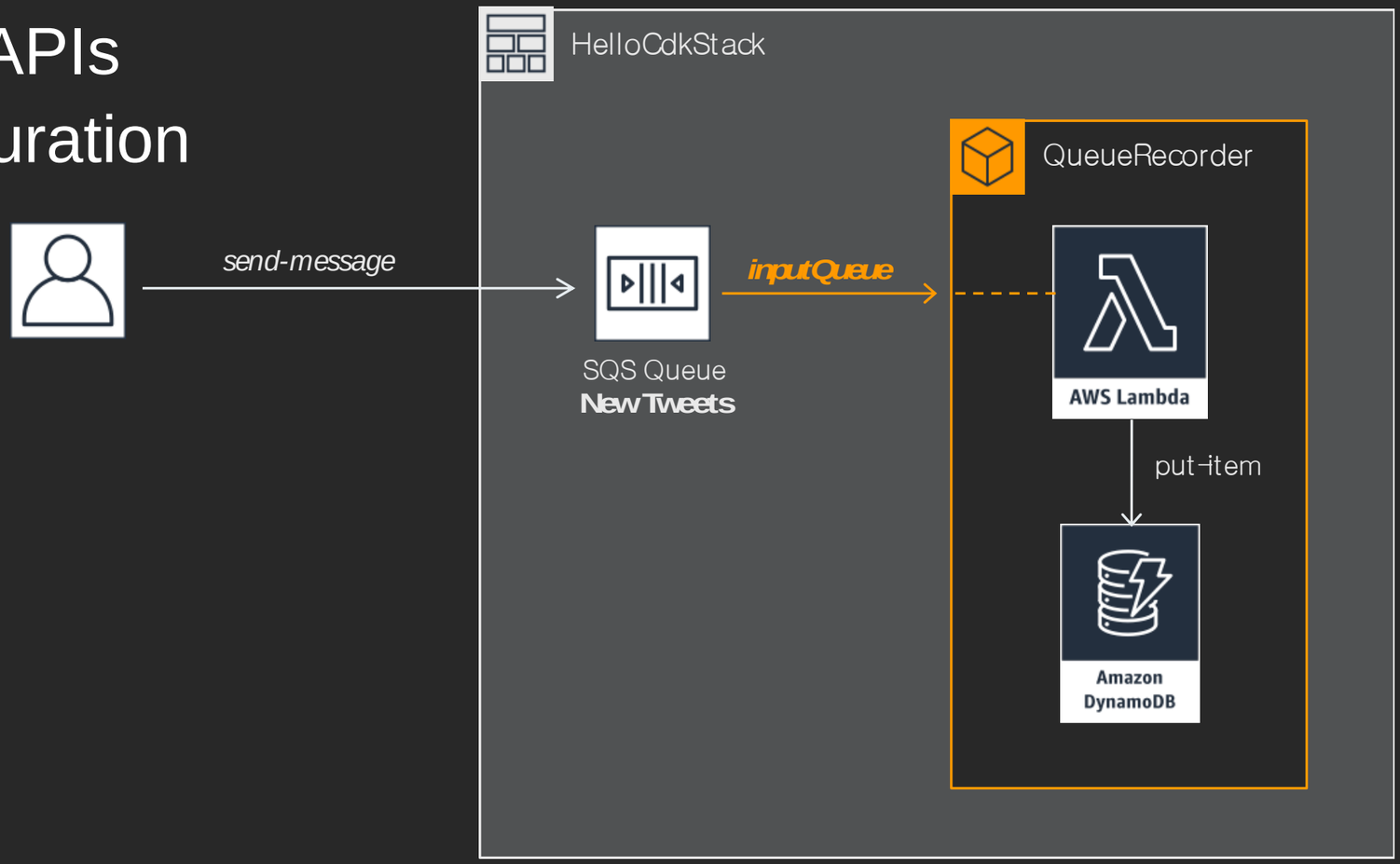
```
npm install -g aws-cdk
cdk init --language  <typescript | java | csharp | ...>
```



Demo – Writing constructs

Demo goals

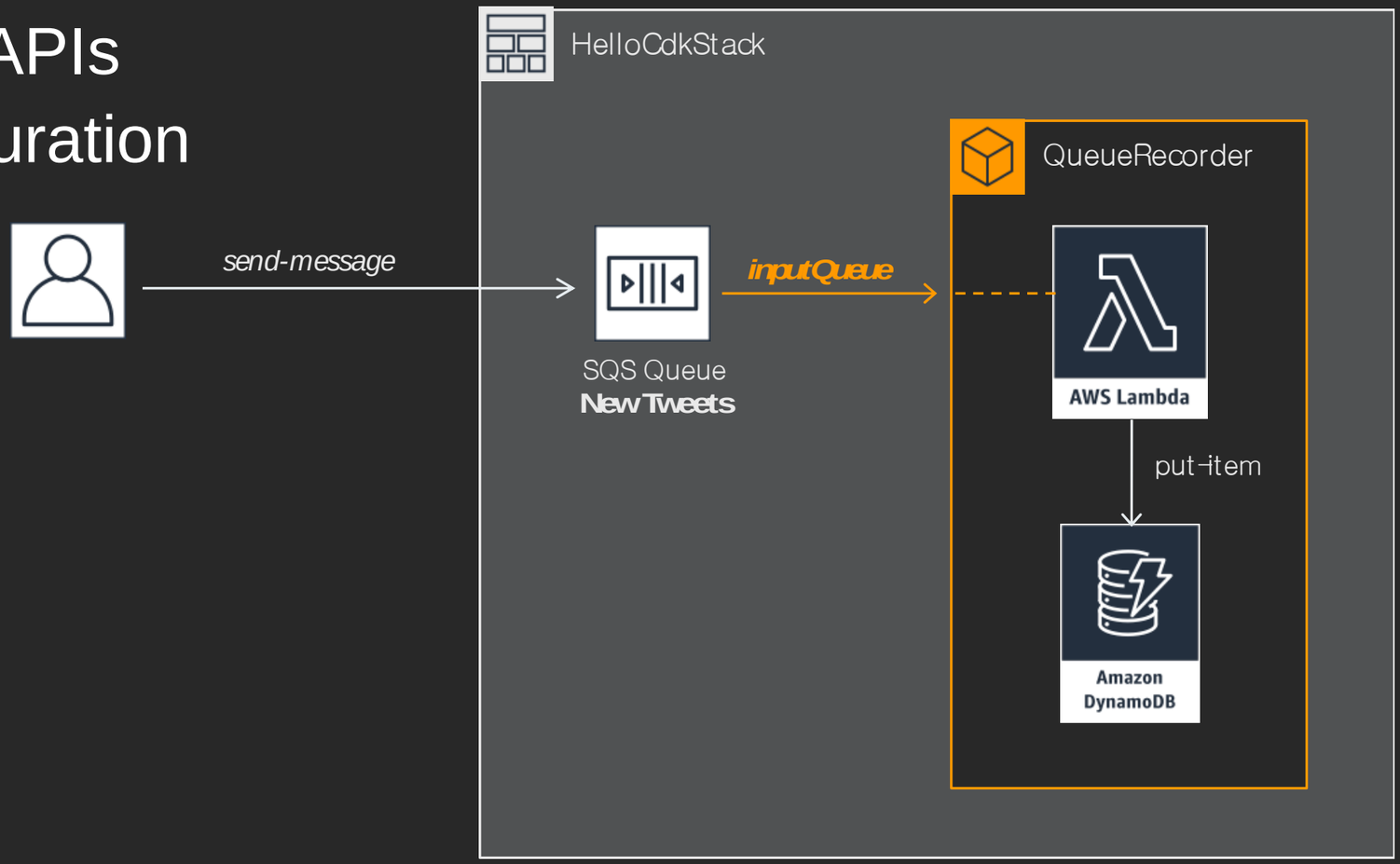
- Thinking in constructs
- Permissions – Grant APIs
- Wiring runtime configuration



Demo

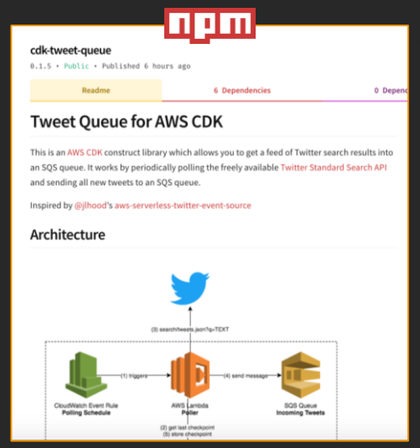
Recap

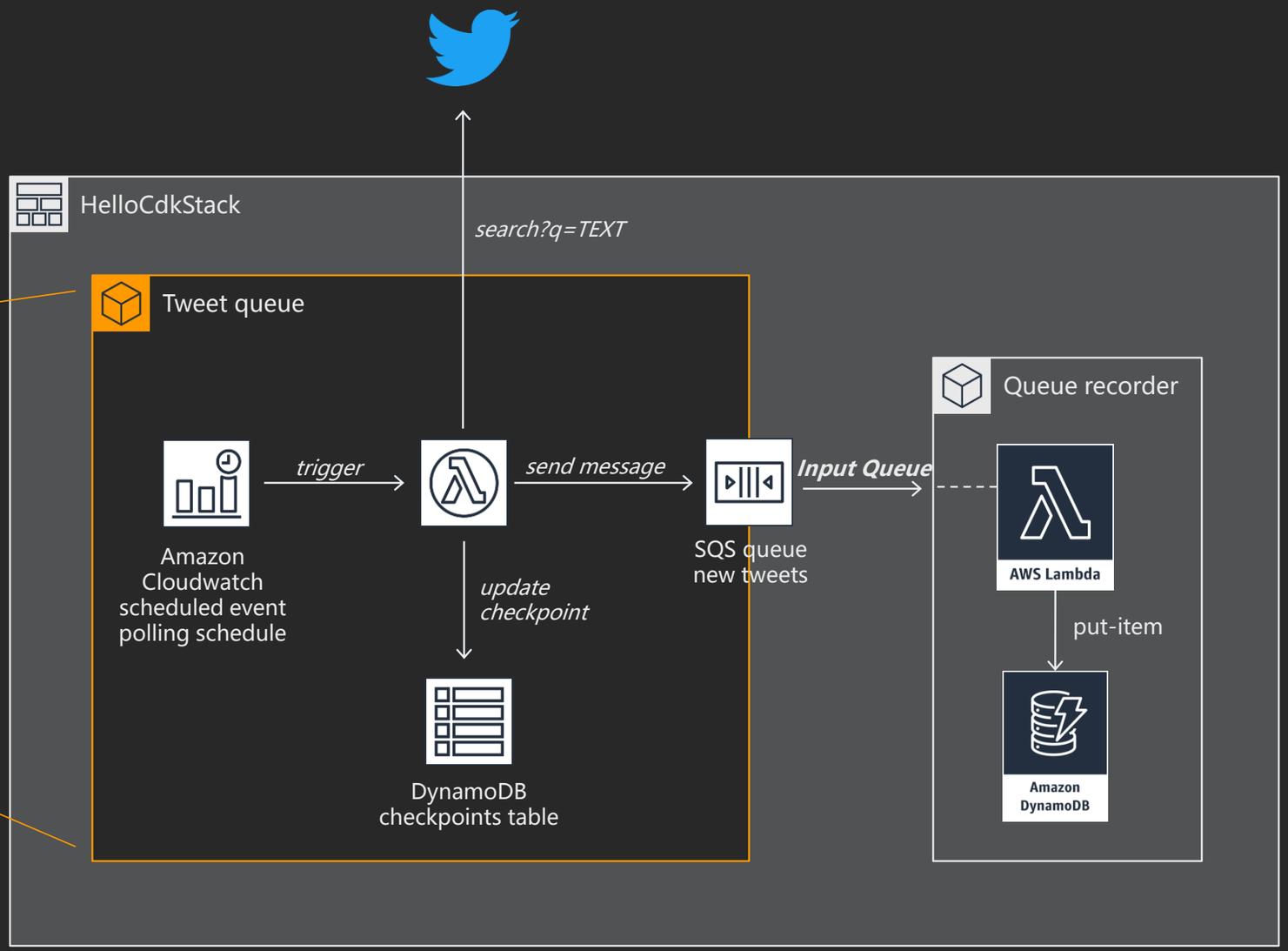
- Thinking in constructs
- Permissions – Grant APIs
- Wiring runtime configuration



Demo – Reusability and sharing

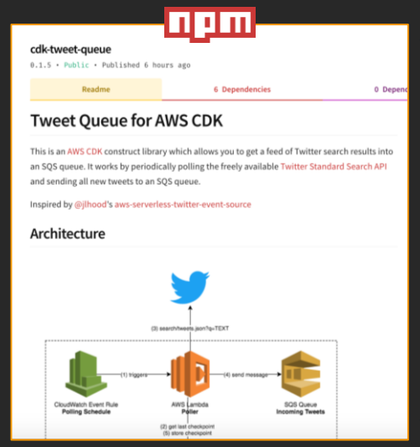
Demo goals

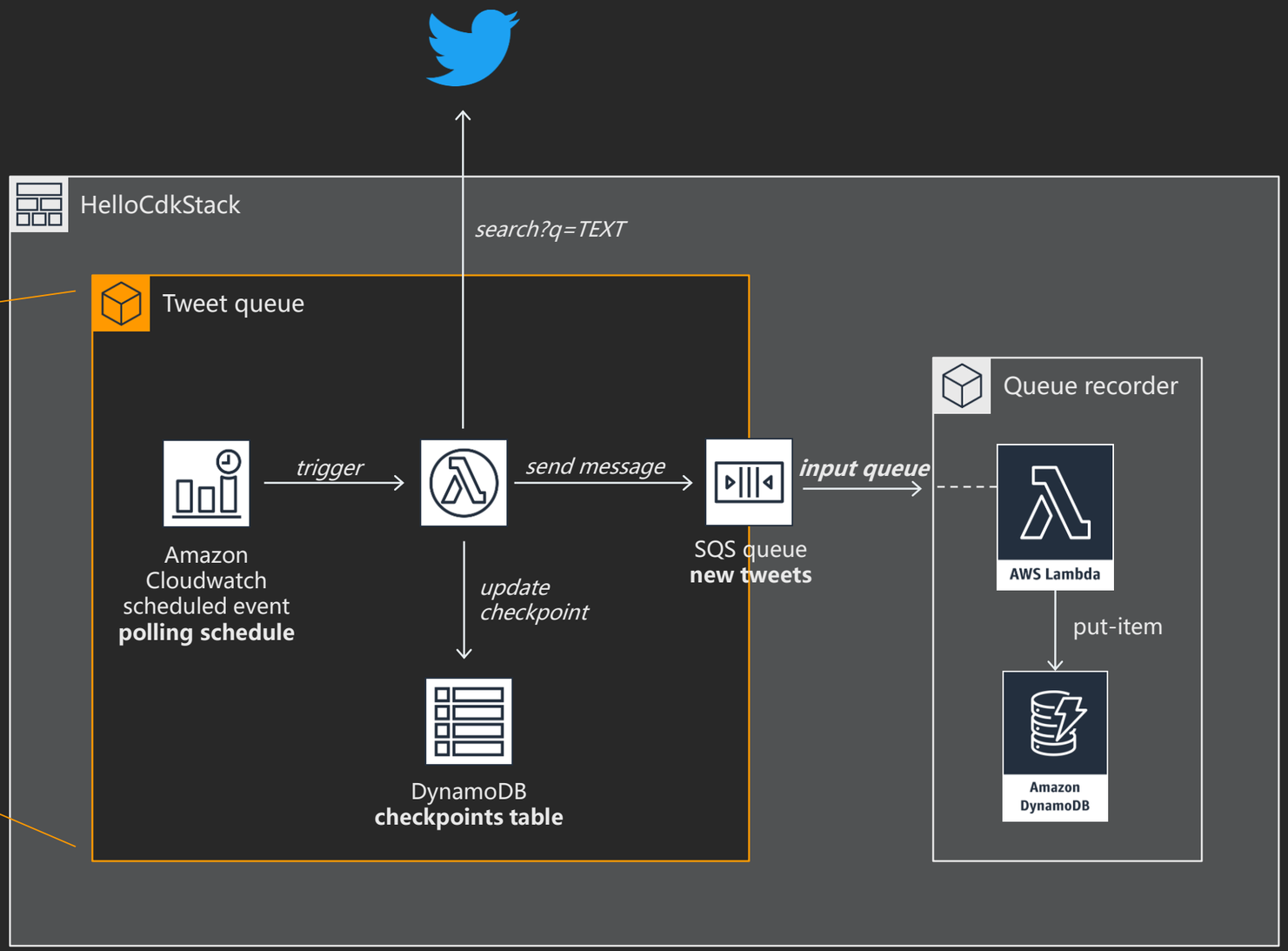




Demo

Recap





#awscdk

More to explore

AWS re:Invent

aws

AWS construct library

#awscdk

- Amazon Elastic Container Service (Amazon ECS)
- AWS Lambda
- Amazon API Gateway
- Amazon Elastic Cloud Compute (Amazon EC2)
  - VPCs, security groups, Auto Scaling
- Amazon CloudWatch
  - Metrics, alarms, dashboards
- AWS Step Functions
- AWS CodePipeline
- AWS Auto Scaling

README.md

AWS Elastic Container Service (ECS) Construct Library

This package contains constructs for working with **AWS Elastic Container Service** (ECS). The simplest example of using this library looks like this:

```
// Create an ECS cluster
const cluster = new ecs.Cluster(this, 'Cluster', {
  vpc,
});

// Add capacity to it
cluster.addDefaultAutoScalingGroupCapacity({
  instanceType: new ec2.InstanceType("t2.xlarge"),
  instanceCount: 3,
});

// Instantiate ECS Service with an automatic load balancer
const ecsService = new ecs.LoadBalancedEc2Service(this, 'Service', {
  cluster,
  memoryLimitMiB: 512,
  image: ecs.ContainerImage.fromDockerHub("amazon/amazon-ecs-sample"),
});
```

Fargate vs ECS

There are two sets of constructs in this library; one to run tasks on ECS and one to run Tasks on Fargate.

- Use the `Ec2TaskDefinition` and `Ec2Service` constructs to run tasks on EC2 instances running in your account.
- Use the `FargateTaskDefinition` and `FargateService` constructs to run tasks on instances that are managed for you by AWS.

Here are the main differences:

- **EC2**: instances are under your control. Complete control of task to host allocation. Required to specify at least a memory reseration or limit for every container. Can use Host, Bridge and AwsVpc networking modes. Can attach Classic Load Balancer. Can share volumes between container and host.
- **Fargate**: tasks run on AWS-managed instances, AWS manages task to host allocation for you. Requires specification of memory and cpu sizes at the taskdefinition level. Only supports AwsVpc networking modes and Application/Network

aws

Multi-language support

#awscdk

Coming soon!

README.md

jsii

AWS CodeBuild passing

jsii allows code in any language to naturally interact with JavaScript classes.

For example, consider the following TypeScript class:

```
export class HelloJsii {
    public sayHello(name: string) {
        return `Hello, ${name}!`
    }
}
```


aws

Advanced topics

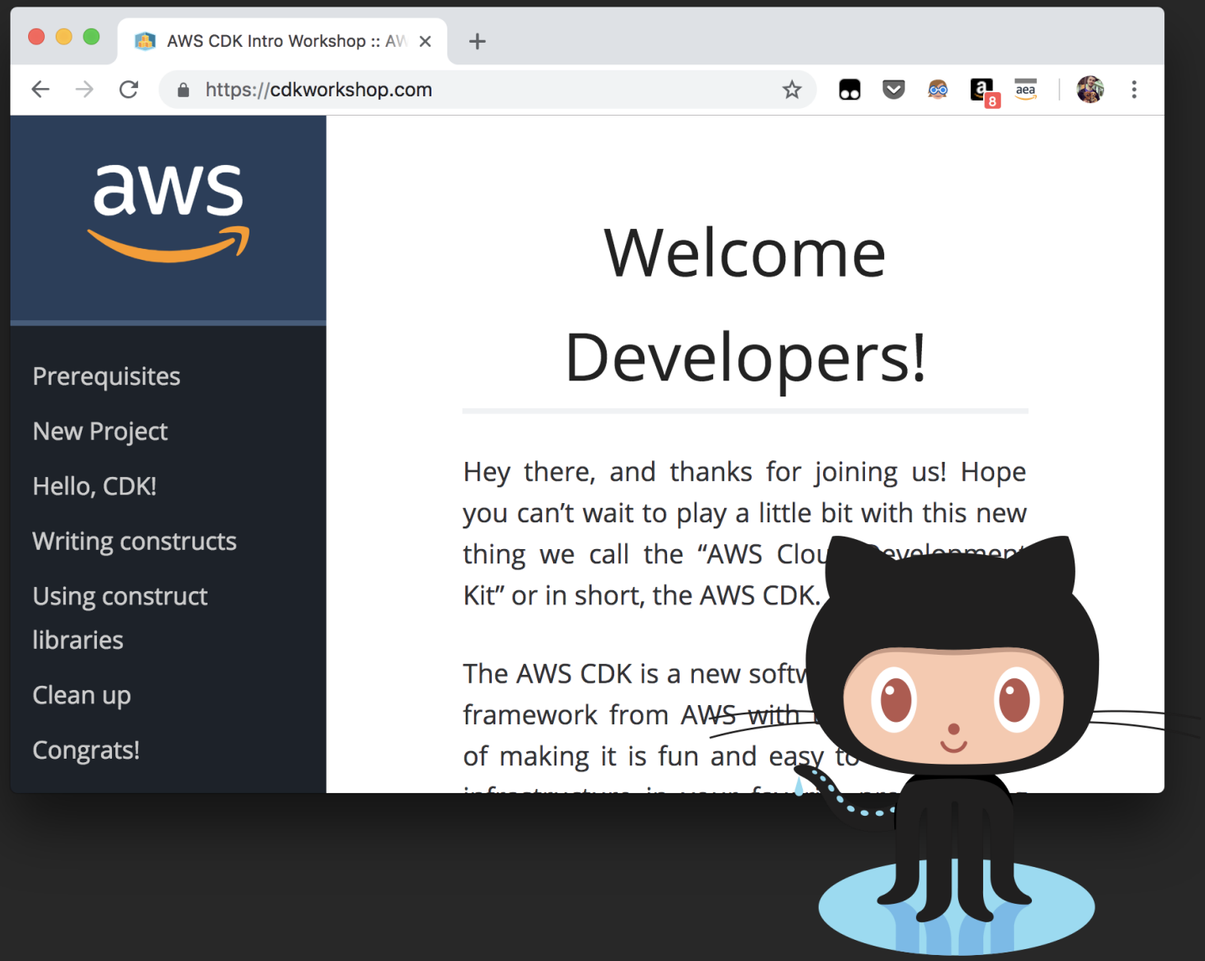
#awscdk

- Multi-stack and cross-region applications
- Declarative constructs with AWS CDK applets
- Working with the AWS CloudFormation DOM
- Embedding existing AWS CloudFormation templates
- Importing resources from other stacks
- Embedding code bundles, docker files, and others into constructs

Next steps

#awscdk

- Try out the CDK
  - https://cdkworkshop.com
  - https://awslabs.github.io/aws-cdk
  - https://github.com/awslabs/cdk-reinvent
- Engage with us
  - https://github.com/awslabs/aws-cdk
  - https://gitter.im/awslabs/aws-cdk
- Check out related sessions



aws

Other CDK sessions

#awscdk

DEV411: Infrastructure as Code: AWS Best Practices

Tuesday, Nov 27, 6:15 p.m. - 7:15 p.m. Aria West, Level 3, Starvine 2
Wednesday, Nov 28, 1:45 p.m. - 2:45 p.m. Mirage, Antigua B
Thursday, Nov 29, 12:15 p.m. - 1:15 p.m. Venetian, Level 2, Veronese 2402

DEV327: Beyond the Basics: Advanced Infrastructure as Code Programming on AWS

Monday, Nov 26, 11:30 a.m. – 12:30 p.m. Aria East, Level 2, Mariposa 5
Thursday, Nov 29, 4:00 p.m. - 5:00 p.m. Aria West, Level 3, Ironwood 5

DEV309: CI/CD for Serverless and Containerized Applications

Tuesday, Nov 27, 7:00 p.m. - 8:00 p.m. Venetian, Level 2, Venetian E
Thursday, Nov 29, 2:30 p.m. - 3:30 p.m. Bellagio, Level 1, Grand Ballroom 2

AWS re:Invent

aws

Thank you!

Elad Ben-Israel
@emeshbi

Jason Fulghum
@jason_fulghum

AWS re:Invent

aws

Please complete the session survey in the mobile app.