

How This Course is Structured

CloudFormation

DevOps Tools - Run
Templates

CDK (Cloud
Development Kit)

Interview Guide

How This Course is Structured



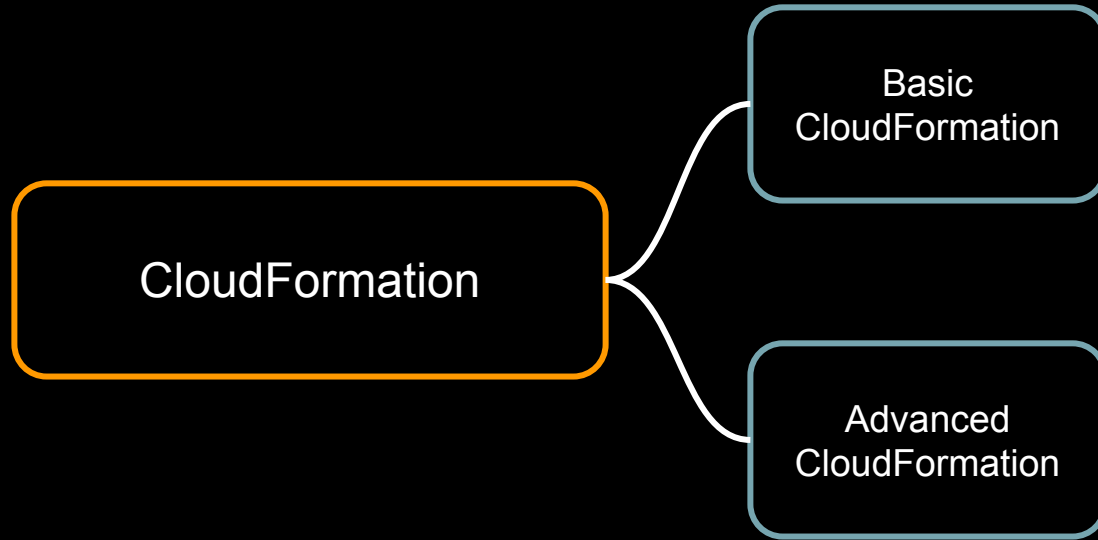
CloudFormation

DevOps Tools - Run
Templates

CDK (Cloud
Development Kit)

Interview Guide

CloudFormation



CloudFormation

CloudFormation

```
graph LR; A[CloudFormation] --- B[Basic CloudFormation]; A --- C[Advanced CloudFormation];
```

Basic
CloudFormation

Advanced
CloudFormation

Infrastructure as Code - What and Why
CloudFormation Intro
Cost Of CloudFormation Stack
JSON Vs YAML
Input Parameters
Intrinsic Functions
Pseudo Parameters
Mappings
Conditions
Resources
Outputs
CloudFormation Designer
Running CloudFormation from CLI
MUST WATCH: How to Code Any
CloudFormation

CloudFormation

CloudFormation

```
graph LR; A[CloudFormation] --- B[Basic CloudFormation]; A --- C[Advanced CloudFormation];
```

Basic
CloudFormation

Advanced
CloudFormation

CloudFormation And IAM

Drift Detection

Termination Protection

Stack Policy

Nested Stack

Change Sets

Lambda Primer

Custom Resources

StackSets And Hub & Spoke Model

Userdata

Dynamic Reference - Secrets Manager
And SSM

Helper Scripts (cfn-init, cfn-hup,
cfn-signal, cfn-get-metadata)

CloudFormer Tool

CloudFormation Best Practices

Creating Serverless Infrastructure Using
SAM (Serverless Application Model)

How This Course is Structured

CloudFormation

DevOps Tools - Run
Templates

CDK (Cloud
Development Kit)

Interview Guide

DevOps Tools

DevOps Tools - Run
Templates

Jenkins

Terraform

AWS
CodePipeline

DevOps Quick Primer
Why Automate CloudFormation
Install Jenkins Easy Way!
Run Cft From Git Using Jenkins Job
Run Cft Using Jenkins Pipeline
Terraform - What and Why
Run Terraform Templates
AWS CodePipeline - What and Why
Run Cft With AWS CodePipeline
Multiple Ways
CloudFormation Vs Terraform
AWS CodePipeline Vs Jenkins
MUST WATCH: AWS CodePipeline

How This Course is Structured

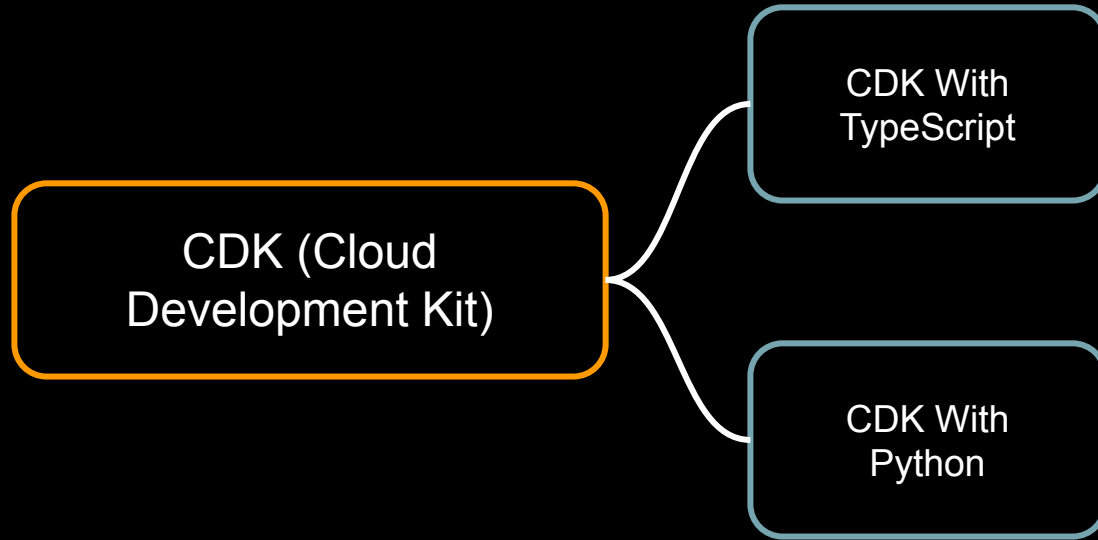
CloudFormation

DevOps Tools - Run
Templates

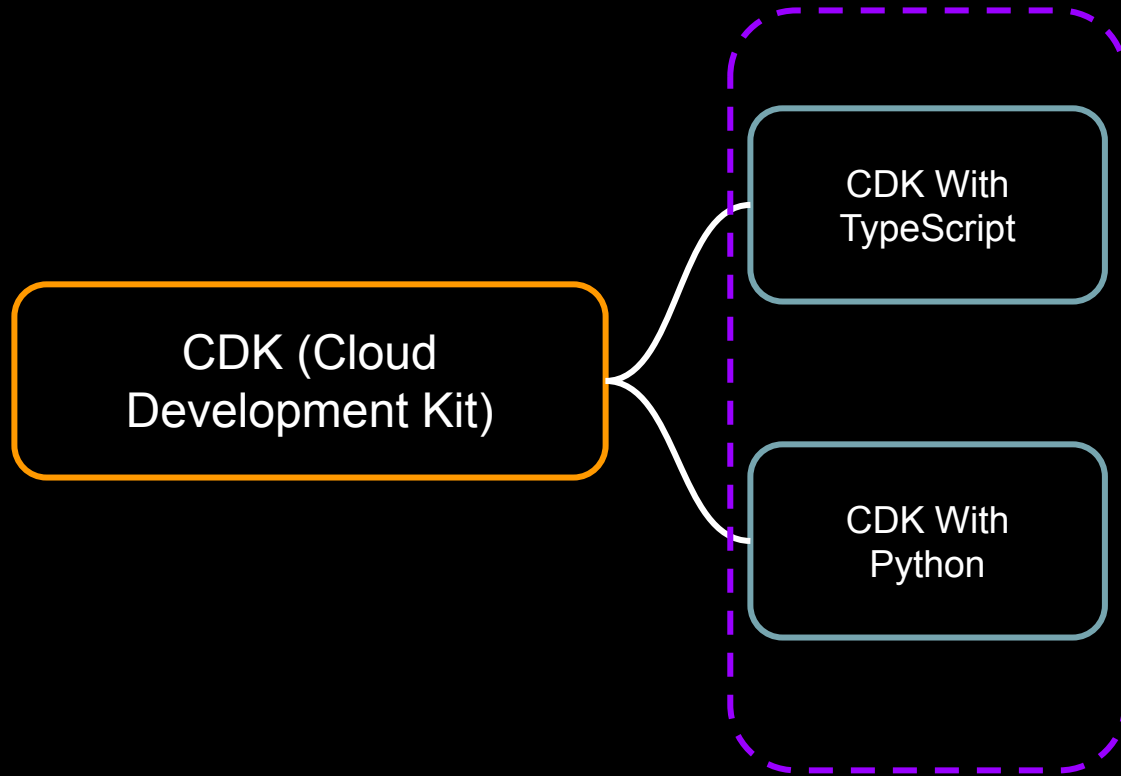
CDK (Cloud
Development Kit)

Interview Guide

AWS CDK



AWS CDK



CDK - What And Why?

CDK Vs CloudFormation

TypeScript - Initial Setup

TypeScript - Generate Cft And Deploy

TypeScript - Code Any TypeScript CDK!

Python - Initial Setup

Python - Generate Cft And Deploy

Python - Code Any Python CDK!

Example Infra - EC2, S3, Lambda, API

Gateway, SNS, SQS

MUST WATCH: [CDK CheatSheet For Coding Any CDK](#)

How This Course is Structured

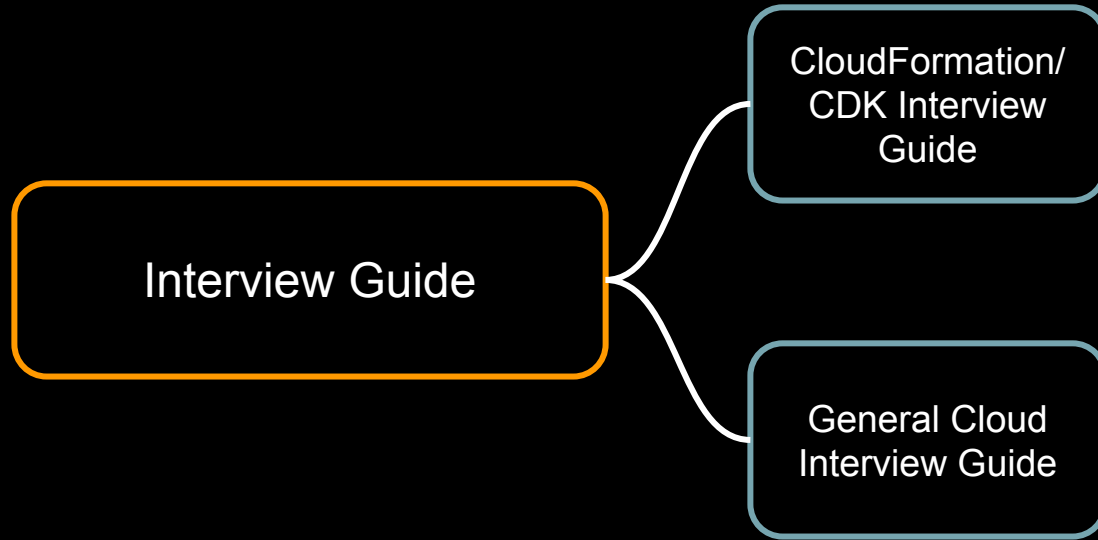
CloudFormation

DevOps Tools - Run
Templates

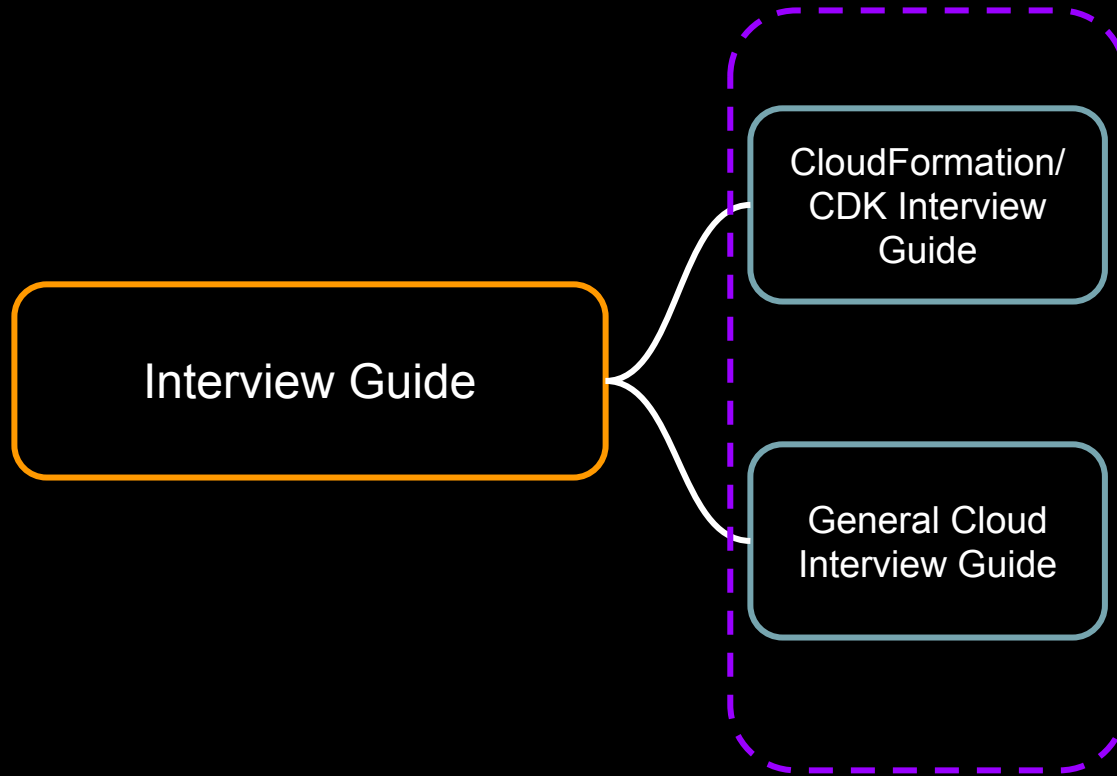
CDK (Cloud
Development Kit)

Interview Guide

Interview Guide



Interview Guide



CloudFormation Interviews - Main Areas Of Focus

Questions On Coding Template

Questions On DevOps

Questions On Learn And Adapt

General Cloud Interview Tips

Whiteboarding Interview Tips

MUST WATCH: [How I Switched My Career To AWS - Tips And Steps](#)

What is AWS CloudFormation

- Template to Spin Up AWS Infrastructure
 - Example - EC2, RDS, AutoScaling, three tier web app etc.
- Simplified Infrastructure Management
- Quickly Replicate Your Infrastructure
 - Dev, Stage, Prod, DR
- Easily Control and Track Changes to Infrastructure

CloudFormation Concepts

- Template
 - JSON and YAML file
 - Used as blueprints for building AWS resources

JSON Vs YAML

- NOT as BIG DEAL as some people make it to be

JSON Vs YAML

JSON

JavaScript Object Notation

Lightweight format for storing and transporting data

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters" : {
    "SNSEmail" : {
      "Type" : "String",
      "Description" : "Enter email for SNS notification"
    }
  },
  "Resources" : {
    "AlertSNSTopic" : {
      "Type" : "AWS::SNS::Topic",
      "Properties" : {
        "Subscription" : [{
          "Endpoint" : { "Ref" : "SNSEmail" },
          "Protocol" : "email"
        }]
      }
    }
  }
}
```

YAML

"YAML Ain't Markup Language"

human-readable data-serialization language

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  SNSEmail:
    Type: String
    Description: Enter email for SNS notification
Resources:
  AlertSNSTopic:
    Type: 'AWS::SNS::Topic'
    Properties:
      Subscription:
        -
          Endpoint:
            Ref: SNSEmail
          Protocol: email
```

JSON Vs YAML

- NOT as BIG DEAL as it is perceived
- One Click convert from JSON To YAML and vice versa
 - CloudFormation Console
 - Visual Code Editor
 - Hundreds of other addons/online converters
 - Free!
- JSON CloudFormation can call YAML CloudFormation (Nested Stack) and vice versa
- Code with what YOU are comfortable with and rock on

CloudFormation Concepts

- Template

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A sample template",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t2.micro",
        "KeyName" : "testkey",
        "BlockDeviceMappings" : [
          {
            "DeviceName" : "/dev/sdm",
            "Ebs" : {
              "VolumeType" : "io1",
              "Iops" : "200",
              "DeleteOnTermination" : "false",
              "VolumeSize" : "20"
            }
          }
        ]
      }
    }
  }
}
```

CloudFormation Concepts

- Template
 - JSON and YAML file
 - Used as blueprints for building AWS resources
- Stacks
 - Stacks are created by submitting templates
 - Stacks can be created, updated and deleted
- Change Sets
 - Summary of proposed changes in stack
 - How the change will impact resources
 - E.g - changing name of RDS database

CloudFormation Flow - Create Stack

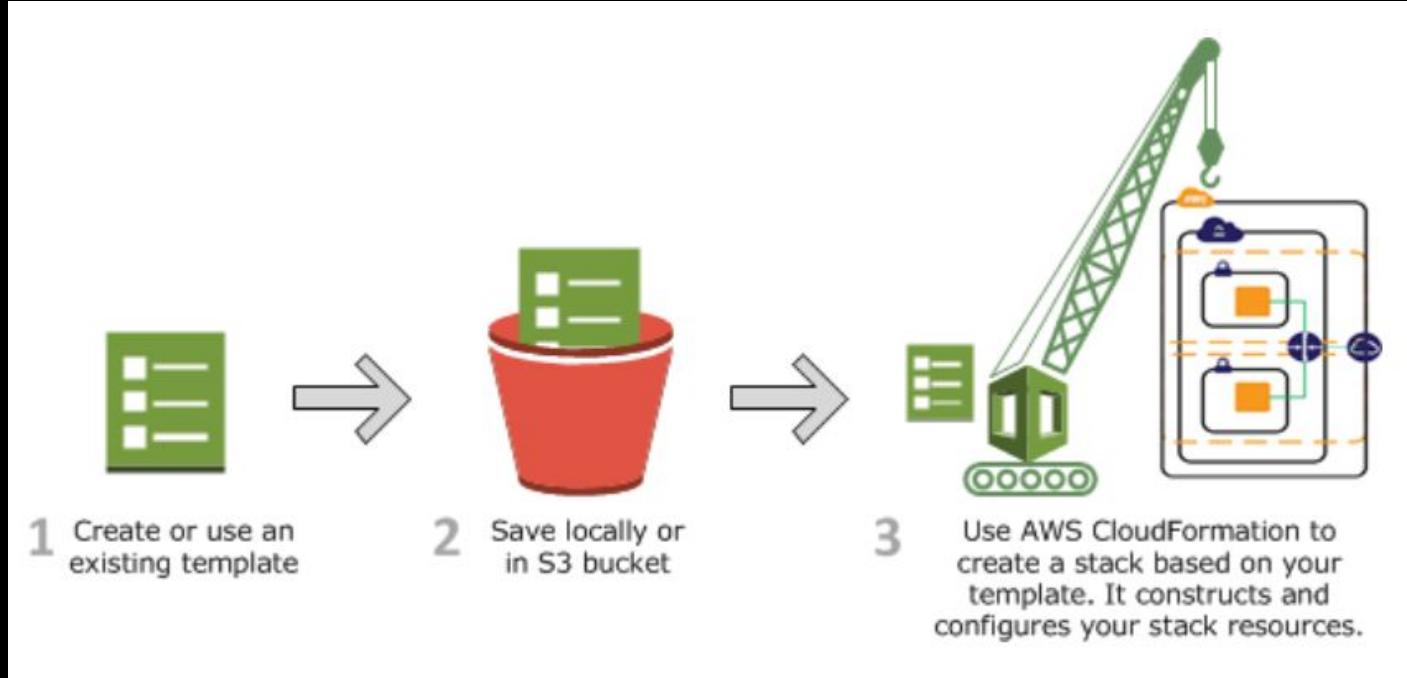
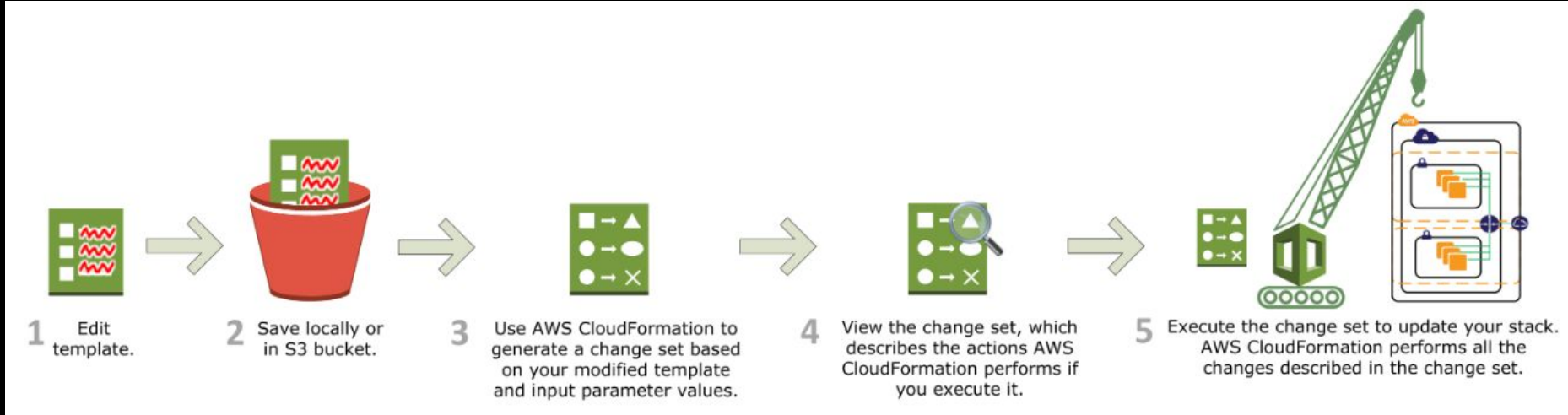


Image: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-what-is-howdoesitwork.html>

CloudFormation Flow - Change Sets



CloudFormation Demo - Create S3 Bucket



Cost of Your CloudFormation Stack



Cost of Your CloudFormation Stack

- CloudFormation itself is free
- You only pay for the resources spun up by CloudFormation (if not eligible by free tier)

Run CloudFormation From AWS CLI

- How to run any AWS CLI Commands!
- Demo using CloudFormation
 - Create Stack
 - Check Status of the Stack
 - Delete Stack

CloudFormation Template Anatomy

Template Anatomy

```
{
  "AWSTemplateFormatVersion" : "version date",
  "Description" : "JSON string",
  "Metadata" : {
    template metadata
  },
  "Parameters" : {
    set of parameters
  },
  "Mappings" : {
    set of mappings
  },
  "Conditions" : {
    set of conditions
  },
  "Transform" : {
    set of transforms
  },
  "Resources" : {
    set of resources
  },
  "Outputs" : {
    set of outputs
  }
}
```

REQUIRED

Template Version

- Not a Random Date, Use “2010-09-09”

```
{  
  "AWSTemplateFormatVersion" : "2010-09-09",  
  "Description" : "A simple EC2 instance",  
  "Resources" : {  
    "MyEC2Instance" : {  
      "Type" : "AWS::EC2::Instance",  
      "Properties" : {  
        "ImageId" : "ami-0ff8a91507f77f867",  
        "InstanceType" : "t1.micro"  
      }  
    }  
  }  
}
```

Description

- Free Form Description

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A simple EC2 instance",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t1.micro"
      }
    }
  }
}
```

Metadata

- Need to Study

Parameters

- Enables you to input custom values
- Will be used A LOT

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "A simple EC2 instance",
  "Resources" : {
    "MyEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : "ami-0ff8a91507f77f867",
        "InstanceType" : "t1.micro"
      }
    }
  }
}
```


Parameters

```
"Parameters" : {  
  "InstanceTypeParameter" : {  
    "Type" : "String",  
    "Default" : "t2.micro",  
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"],  
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro."  
  }  
}
```

```
"Ec2Instance" : {  
  "Type" : "AWS::EC2::Instance",  
  "Properties" : {  
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },  
    "ImageId" : "ami-0ff8a91507f77f867"  
  }  
}
```

Parameters

- More to Learn about Parameters

But Let's Do DEMO First!

Parameters - General Info

- Max of 60 Parameters
- Name must be Alphanumeric and Unique
- Each parameter must have a Type
 - String
 - Number (Integer or Float)
 - List <Number>
 - CommaDelimitedList ([“test”, “dev”, “prod”])
 - AWS-Specific Parameter Types
 - SSM Parameter Types
 - Parameter Converted to String for Referring elsewhere

Parameters - General Info Contd.

- Must have Value at Runtime
 - Using default value is a Good Practice

```
"Parameters" : {  
  "InstanceTypeParameter" : {  
    "Type" : "String",  
    "Default" : "t2.micro",  
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"],  
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro."  
  }  
}
```

Basic Parameter

- Lots of Checks and Properties Available

```
"Parameters" : {  
  "DBPort" : {  
    "Default" : "3306",  
    "Description" : "TCP/IP port for the database",  
    "Type" : "Number",  
    "MinValue" : "1150",  
    "MaxValue" : "65535"  
  },  
  "DBPwd" : {  
    "NoEcho" : "true",  
    "Description" : "The database admin account password",  
    "Type" : "String",  
    "MinLength" : "1",  
    "MaxLength" : "41",  
    "AllowedPattern" : "^[a-zA-Z0-9]*$"  
  }  
}
```

[Link to Full List of Properties in Resources Section](#)

AWS-Specific Parameter Types

- Predefined Parameters
- Example - AWS::EC2::KeyPair::KeyName
 - Validates Input against Existing Key pair Names

```
"Parameters" : {  
  "myKeyPair" : {  
    "Description" : "Amazon EC2 Key Pair",  
    "Type" : "AWS::EC2::KeyPair::KeyName"  
  },  
  "mySubnetIDs" : {  
    "Description" : "Subnet IDs",  
    "Type" : "List<AWS::EC2::Subnet::Id>"  
  }  
}
```

AWS-Specific Parameter Types

- `AWS::EC2::KeyPair::KeyName`
- `AWS::EC2::Subnet::Id`
- `AWS::EC2::VPC::Id`
- `AWS::EC2::Instance::Id`
- `AWS::EC2::Image::Id`
- `AWS::EC2::SecurityGroup::Id`

Quick Detour to Intrinsic Functions

```
"Parameters" : {  
  "InstanceTypeParameter" : {  
    "Type" : "String",  
    "Default" : "t2.micro",  
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"],  
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro."  
  }  
}
```

```
"Ec2Instance" : {  
  "Type" : "AWS::EC2::Instance",  
  "Properties" : {  
    "InstanceType" : { "Ref" : "InstanceTypeParameter" },  
    "ImageId" : "ami-0ff8a91507f77f867"  
  }  
}
```


Quick Detour to Intrinsic Functions

- Several Built-in Functions
- Used in Specific Parts of Template
 - Resource, Outputs, Metadata, Update Policy
Attributes

Quick Detour to Intrinsic Functions

- Fn::Base64
- Fn::Cidr
- Condition Functions
- Fn::FindInMap
- Fn::GetAtt
- Fn::GetAZs ({ "Fn::GetAZs" : "region" } outputs ["us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d"])
- Fn::ImportValue
- Fn::Join
- Fn::Select
- Fn::Split
- Fn::Sub
- Fn::Transform
- Ref

Will Be Covered In Detail - Now Back To Template Anatomy!

Pseudo Parameters

- Predefined by AWS CloudFormation
- Use intrinsic function Ref to retrieve values

JSON

```
"Outputs" : {  
  "StackRegion" : { "Value" : { "Ref" : "AWS::Region" } }  
}
```

YAML

```
Outputs:  
  StackRegion:  
    Value: !Ref "AWS::Region"
```

Popular Pseudo Parameters

- `AWS::Region`
- `AWS::AccountId`
- `AWS::NotificationARNs`
- `AWS::StackId`
- `AWS::StackName`

Mappings

- Key Value mapping
- Use intrinsic function Fn::FindInMap to retrieve values

```
"Mappings" : {  
  "RegionMap" : {  
    "us-east-1"      : { "RHEL" : "ami-012abc123"},  
    "us-west-1"      : { "RHEL" : "ami-012abc124"},  
    "eu-west-1"      : { "RHEL" : "ami-012abc125"}  
  }  
}
```

Fn::FindInMap

- Fn::Base64
- Fn::Cidr
- Condition Functions
- **Fn::FindInMap**
- Fn::GetAtt
- Fn::GetAZs ({ "Fn::GetAZs" : "region" } outputs ["us-east-1a", "us-east-1b", "us-east-1c", "us-east-1d"])
- Fn::ImportValue
- Fn::Join
- Fn::Select
- Fn::Split
- Fn::Sub
- Fn::Transform
- Ref

Fn::FindInMap

```
{ "Fn::FindInMap" : [ "MapName", "TopLevelKey", "SecondLevelKey" ] }
```

- MapName - Logical Name of Mapping
- TopLevelKey - Top-level key name
- SecondLevelKey - Second-level key name, set to one of the keys from the list assigned to TopLevelKey

Find Value from Map

```
"Mappings" : {  
  "RegionMap" : {  
    "us-east-1" : { "RHEL" : "ami-012abc123"},  
    "us-west-1" : { "RHEL" : "ami-012abc124"},  
    "eu-west-1" : { "RHEL" : "ami-012abc125"}  
  }  
}
```

One of these

One of these

```
"Resources" : {  
  "myEC2Instance" : {  
    "Type" : "AWS::EC2::Instance",  
    "Properties" : {  
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "RHEL" ] },  
      "InstanceType" : "m1.small"  
    }  
  }  
}
```

Mapping Name

Top-level key (derived through Pseudo-parameter)

Second-level key

Mappings - Example 2

```
"Mappings" : {  
  "RegionMap" : {  
    "us-east-1"      : {"RHEL" : "ami-012abc123", "WIN" : "ami-012xyz123"},  
    "us-west-1"     : {"RHEL" : "ami-012abc124", "WIN" : "ami-012xyz124"},  
    "eu-west-1"     : {"RHEL" : "ami-012abc125", "WIN" : "ami-012xyz125"}  
  }  
}
```

- Can you determine the `Fn::FindInMap`?

Find Value from Map

```
"Mappings" : {  
  "RegionMap" : {  
    "us-east-1"      : { "RHEL" : "ami-012abc123", "WIN" : "ami-012xyz123" },  
    "us-west-1"     : { "RHEL" : "ami-012abc124", "WIN" : "ami-012xyz124" },  
    "eu-west-1"     : { "RHEL" : "ami-012abc125", "WIN" : "ami-012xyz125" }  
  }  
}
```

```
"Resources" : {  
  "myEC2Instance" : {  
    "Type" : "AWS::EC2::Instance",  
    "Properties" : {  
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "WIN" ] },  
      "InstanceType" : "m1.small"  
    }  
  }  
}
```

Mappings - with Input Parameter

```
{
  "Parameters" : {
    "EnvironmentType": {
      "Description": "The environment type",
      "Type": "String",
      "Default": "test",
      "AllowedValues": ["prod", "test"],
      "ConstraintDescription": "must be a prod or test"
    }
  },

  "Mappings" : {
    "RegionAndInstanceTypeToAMIID" : {
      "us-east-1": {
        "test": "ami-8ff710e2",
        "prod": "ami-f5f41398"
      },
      "us-west-2" : {
        "test" : "ami-eff1028f",
        "prod" : "ami-d0f506b0"
      },
    },
  },
}
```

```
"Outputs" : {
  "TestOutput" : {
    "Description": "Return the name of the AMI ID that matches the region and environment type keys",
    "Value" : { "Fn::FindInMap" : [ "RegionAndInstanceTypeToAMIID", { "Ref" : "AWS::Region" }, { "Ref" : "EnvironmentType" } ] }
  }
}
```

Fn::Join

- What is Fn::Join
- Use cases
- Better alternative (where applicable)

Fn::Join

- Appends set of values into a single value, separated by specified delimiter

Fn::Join Use Cases

```
"SNSTopicArn": {  
  "Fn::Join": [":", [  
    "arn:aws:sns",  
    { "Ref": "AWS::Region" },  
    { "Ref": "AWS::AccountId" },  
    "Topic1" ]  
  ]  
}
```

“arn:aws:sns:us-east-1:12345678901:Topic1”

Fn::Join

- Appends set of values into a single value
- Constructing fixed format names:
 - Arn of resources
 - IAM roles
 - API Gateway URIs
- CloudFormation Inline coding
 - Userdata
 - HTML Codes in CloudFormation

Fn::Join Use Cases

```
PolicyName:  
  !Join  
    - '-'  
    - - 'dev-env'  
      - !Ref 'AWS::Region'  
      - 'lambdaRole'
```

“dev-env-us-east-1-lambdarole”

Use Fn::Sub

```
PolicyName:  
  !Join  
    - '-'  
    - 'dev-env'  
    - !Ref 'AWS::Region'  
    - 'lambdaRole'
```



```
PolicyName:  
  !Sub 'dev-env-${AWS::Region}-lambdaRole'
```

Use Fn::Sub

```
SNSTopicArn:
```

```
!Sub 'arn:aws:sns:${AWS::Region}:${AWS::Account}:Topic1'
```

Fn::Join in Cft Inline Codes

Conditions

- Define conditions based on which entities are created or configured
- E.g - Create a resource when certain condition is true

JSON

```
"Conditions" : {  
  "Logical ID" : {Intrinsic function}  
}
```

YAML

```
Conditions:  
  Logical ID:  
    Intrinsic function
```

Conditions

- Fn::And
- Fn::Equals
- Fn::If
- Fn::Not
- Fn::Or

Conditions

- `Fn::Equals`

How to find the parameters of Conditions?
(Teaching how to fish instead of giving
you the fish!)

Conditions

```
"Parameters" : {  
  "EnvType" : {  
    "Description" : "Environment type",  
    "Default" : "test",  
    "Type" : "String",  
    "AllowedValues" : ["prod", "test"],  
    "ConstraintDescription" : "must specify prod or test"  
  }  
},  
  
"Conditions" : {  
  "CreateProdResources" : {"Fn::Equals" : [{"Ref" : "EnvType"}, "prod"]}  
},  
  
"Resources" : {  
  "EC2Instance" : {  
    "Type" : "AWS::EC2::Instance",  
    "Condition" : "CreateProdResources",  
    "Properties" : {  
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ] }  
    }  
  }  
}
```

Resources

- Required section
- Declares the AWS Resources such as EC2, S3 etc.

JSON

```
Resources" : {  
  "Logical ID" : {  
    "Type" : "Resource type",  
    "Properties" : {  
      Set of properties  
    }  
  }  
}
```

YAML

```
Resources:  
  Logical ID:  
    Type: Resource type  
    Properties:  
      Set of properties
```


Resources

- Required fields in Resources

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-template-resource-type-ref.html>

Resources

```
"Resources" : {  
  "MyEC2Instance" : {  
    "Type" : "AWS::EC2::Instance",  
    "Properties" : {  
      "ImageId" : "ami-012abc124"  
    }  
  }  
}
```

Resources

```
AWSTemplateFormatVersion: '2010-09-09'
```

```
Resources:
```

```
  S3Bucket:
```

```
    Type: AWS::S3::Bucket
```

```
Outputs:
```

```
  BucketName:
```

```
    Value: !Ref 'S3Bucket'
```

```
    Description: Name of the sample Amazon S3 bucket.
```

Outputs

- Outputs (Prints) Certain Attributes of Resources
 - E.g - S3 bucket name, SNS Topic Name etc.

JSON

```
"Outputs" : {  
  "Logical ID" : {  
    "Description" : "Information  
about the value",  
    "Value" : "Value to return",  
    "Export" : {  
      "Name" : "Value to export"  
    }  
  }  
}
```

YAML

```
Outputs:  
  Logical ID:  
    Description: Information about the value  
    Value: Value to return  
    Export:  
      Name: Value to export
```

Outputs

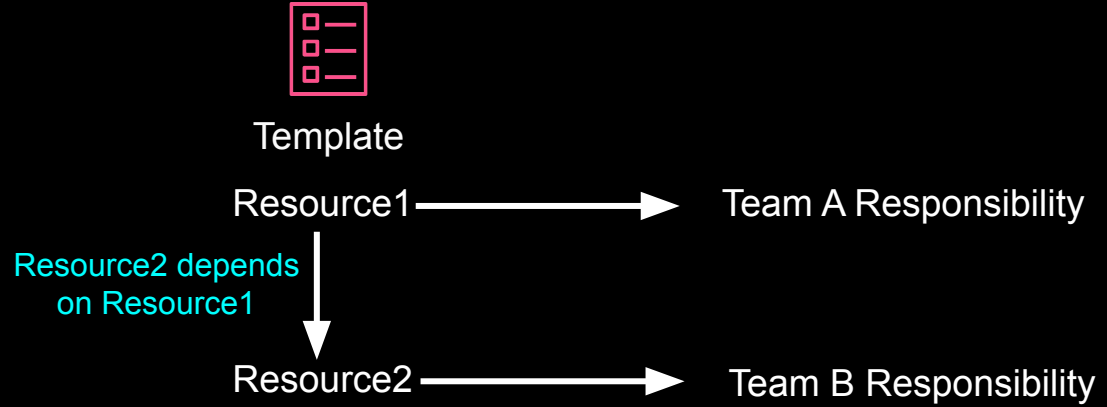
```
"Outputs" : {  
  "BackupLoadBalancerDNSName" : {  
    "Description": "The DNSName of the backup load balancer",  
    "Value" : { "Fn::GetAtt" : [ "BackupLoadBalancer", "DNSName" ] },  
    "Condition" : "CreateProdResources"  
  },  
  "InstanceID" : {  
    "Description": "The Instance ID",  
    "Value" : { "Ref" : "EC2Instance" }  
  }  
}
```

Outputs - Cross Stack

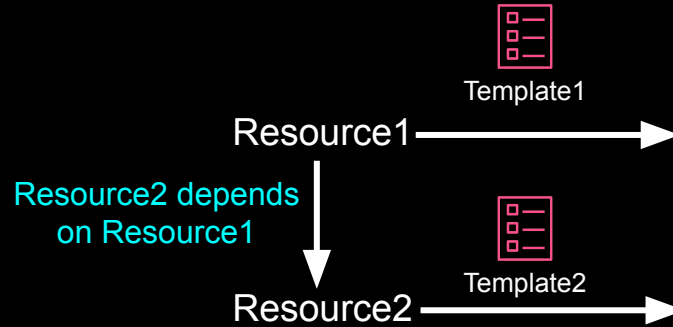
```
"Outputs" : {  
  "StackVPC" : {  
    "Description" : "The ID of the VPC",  
    "Value" : { "Ref" : "MyVPC" },  
    "Export" : {  
      "Name" : { "Fn::Sub": "${AWS::StackName}-VPCID" }  
    }  
  }  
}
```

Example: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/outputs-section-structure.html>

Outputs - Cross Stack

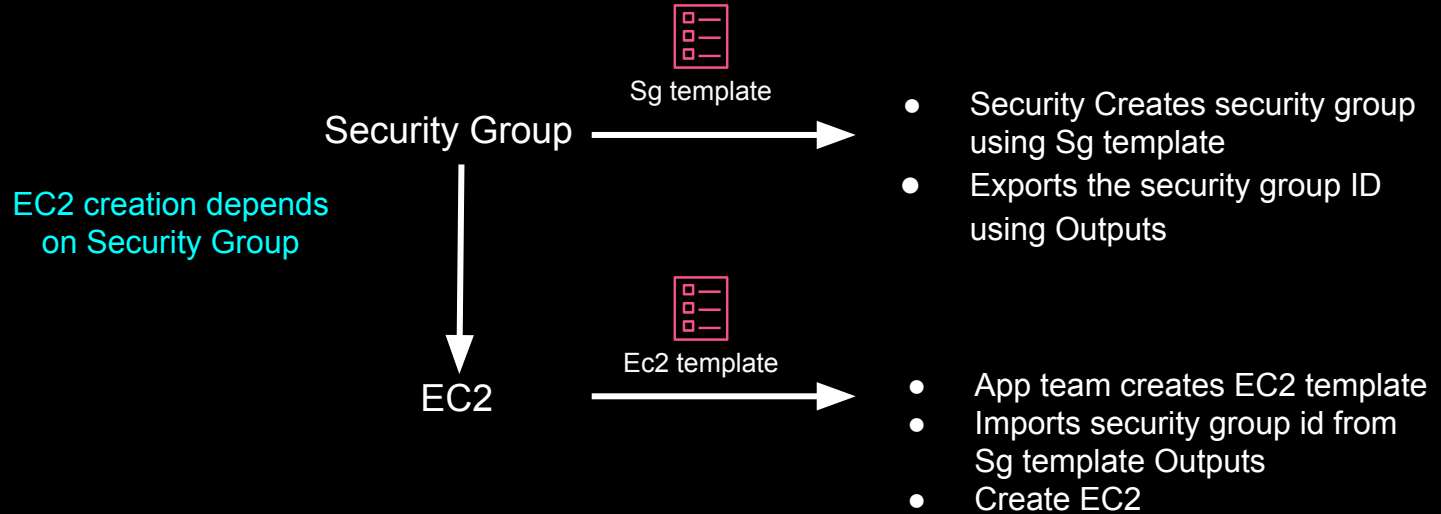


Outputs - Cross Stack

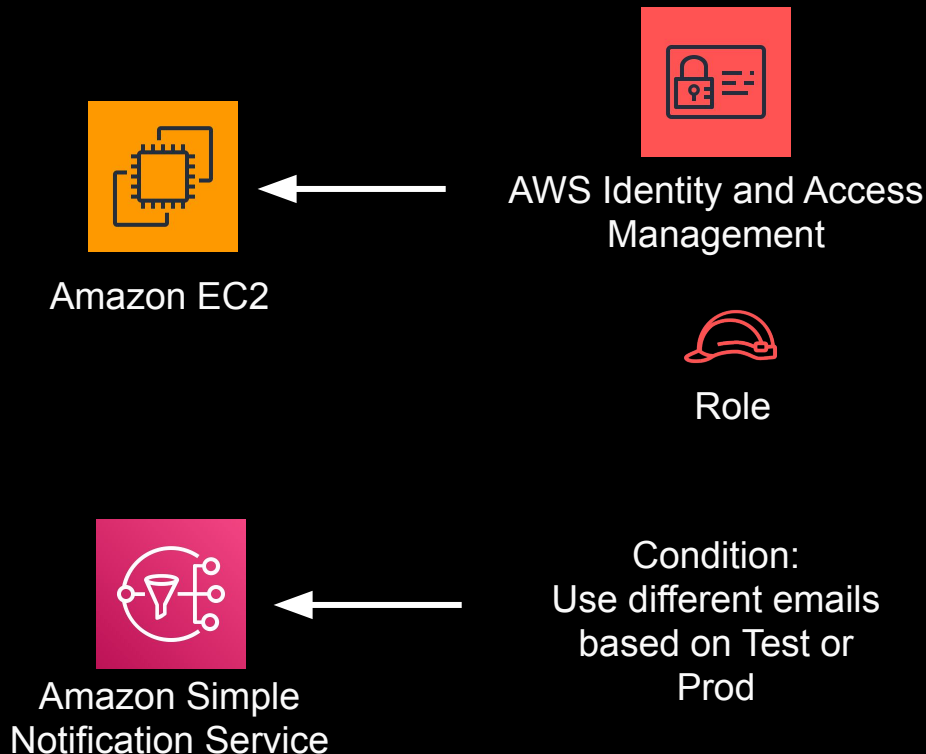


- Team A Creates Resource1 using Template1
- Exports the required value using Outputs
- Team B creates Template2
- Imports reqd value from Template1 Outputs
- Create Resource2

Outputs - Cross Stack



Code Any CloudFormation



Involves:

Mappings
Parameter
Pseudo Parameters
Intrinsic Functions
Conditions
Outputs
Resource

First step in learning Programming

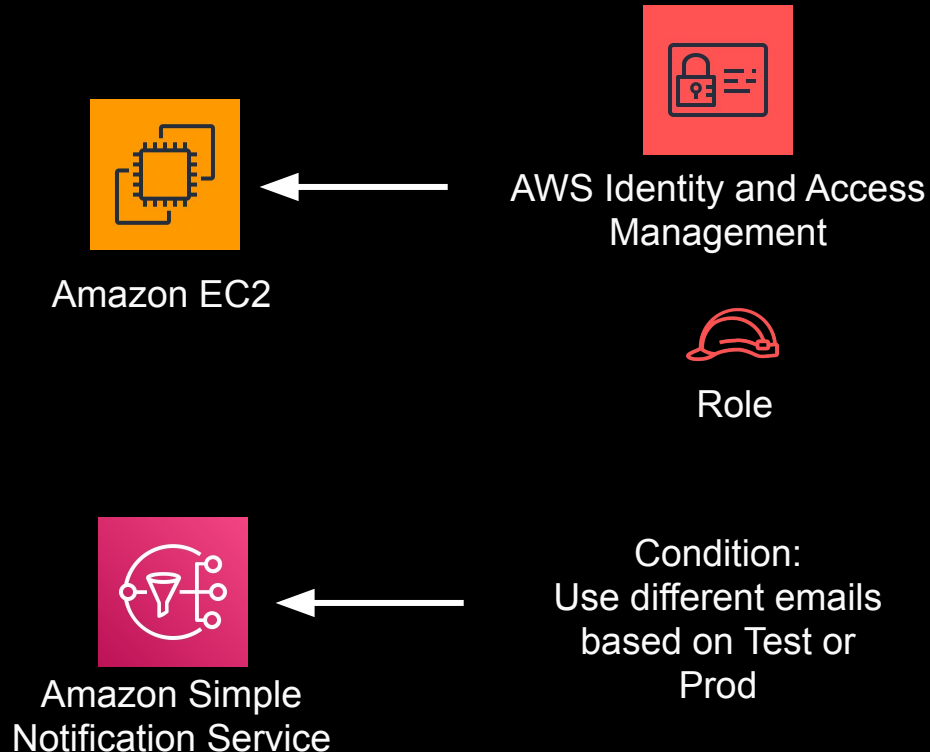


Learn Basic
Syntax,
Data Types and
Variables.



Learn how to
Google.

How Do I Code Any CloudFormation?



Involves:

- Mappings
- Parameter
- Pseudo Parameters
- Intrinsic Functions
- Conditions
- Outputs
- Resource

We Will Learn:

- How to navigate reference guides
- How to get a base template
- How to code any infrastructure
- Tips and tricks

Helper Scripts

- Python scripts to install software and start services on an EC2 instance that is created in your CloudFormation Stack
 - cfn-init
 - cfn-signal
 - cfn-get-metadata
 - cfn-hup
- *From Experience - NOT used extensively in Enterprises*
 - Spinning up infra and installing software have separate lifecycles
 - Makes CloudFormation hard to read
 - Better alternatives available
 - Install software through Ansible, SSM etc.
 - Create AMI, use that AMI in template

Helper Scripts - cfn-init

- Cfn-init helper script reads template metadata from AWS::CloudFormation::Init key and acts accordingly to:
 - Fetch and parse metadata from Cft
 - Install packages
 - Write files to disk
 - Enable/Disable and start/stop services

Syntax

```
cfn-init --stack|-s stack.name.or.id \  
  --resource|-r Logical.resource.id \  
  --region region \  
  --access-key access.key \  
  --secret-key secret.key \  
  --role rolename \  
  --credential-file|-f credential.file \  
  --configsets|-c config.sets \  
  --url|-u service.url \  
  --http-proxy HTTP.proxy \  
  --https-proxy HTTPS.proxy \  
  --verbose|-v
```

Helper Scripts - cfn-signal

- cfn-init is going and installing packages
- cfn-signal checks if all the installations are done
- Can check if all installations are success or errored
- Also sends data to Events part of the Stack in console

```
cfn-signal --success|-s signal.to.send \  
  --access-key access.key \  
  --credential-file|-f credential.file \  
  --exit-code|-e exit.code \  
  --http-proxy HTTP.proxy \  
  --https-proxy HTTPS.proxy \  
  --id|-i unique.id \  
  --region AWS.region \  
  --resource resource.logical.ID \  
  --role IAM.role.name \  
  --secret-key secret.key \  
  --stack stack.name.or.stack.ID \  
  --url AWS CloudFormation.endpoint
```

Helper Scripts - cfn-get-metadata

- Retrieves and prints the metadata of cft
- All of cfn-init installations are from metadata
- If installations go wrong, get metadata using cfn-signal
 - Pretty Painful to debug using metadata
 - Ansible, SSM preferred

```
cfn-get-metadata --access-key access.key \  
                 --secret-key secret.key \  
                 --credential-file|f credential.file \  
                 --key|k key \  
                 --stack|-s stack.name.or.id \  
                 --resource|-r logical.resource.id \  
                 --role IAM.role.name \  
                 --url|-u service.url \  
                 --region region
```

Helper Scripts - cfn-hup

- Run as Daemon or one time
- Detect updates to metadata
- Runs user-specified actions when changes are detected

```
cfn-hup --config/-c config.dir \  
        --no-daemon \  
        --verbose/-v
```


CloudFormation And IAM User

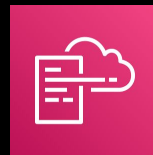


IAM User

Q - What can IAM User do with CloudFormation?



A - Depends on the policies attached to the IAM User

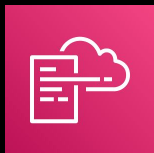


AWS CloudFormation

Filter policies <input type="text" value="cloudformation"/>				
	Policy name	Type	Used as	Description
<input type="radio"/>	AWSCloudFormationFullAccess	AWS managed	Permissions policy (2)	Provides full access to AWS CloudFormation.
<input type="radio"/>	AWSCloudFormationReadOnlyAccess	AWS managed	None	Provides access to AWS CloudFormation via the AWS Management Console.
<input type="radio"/>	AWSDeepRacerCloudFormationAccessPolicy	AWS managed	None	Allows CloudFormation to create and manage AWS stacks and resources on your behalf.
<input type="radio"/>	CloudFormationStackSetsOrgAdminServiceRolePolicy	AWS managed	None	Service Role for CloudFormation StackSets (Organization Master Account)
<input type="radio"/>	CloudFormationStackSetsOrgMemberServiceRolePolicy	AWS managed	None	Service Role for CloudFormation StackSets (Organization Member Account)

IMPORTANT - This does NOT define what services CloudFormation have access to

CloudFormation IAM Service Role



AWS CloudFormation

Q - What AWS Resources can
CloudFormation
create/update/delete?



A - Depends on the policies
attached to the Role for
CloudFormation.

If no Service Role given, inherits
IAM User Policies.



Template



Stack

CloudFormation and IAM

DEMO

CLOUDFORMATION ADVANCED CONCEPTS

Change Sets

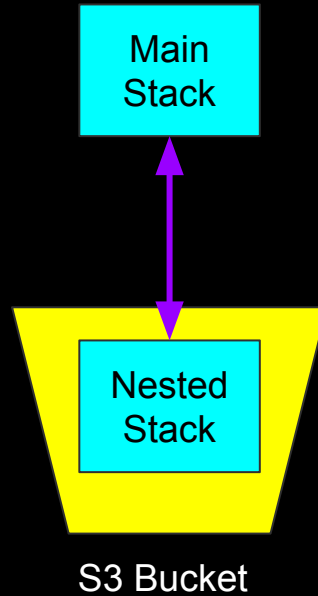
- Shows you impact to running resources with proposed changes
- Shown BEFORE you execute the change
- NOT an indicator that Cft will run fine! Cloudformation can still fail during execution

DEMO - Change Sets

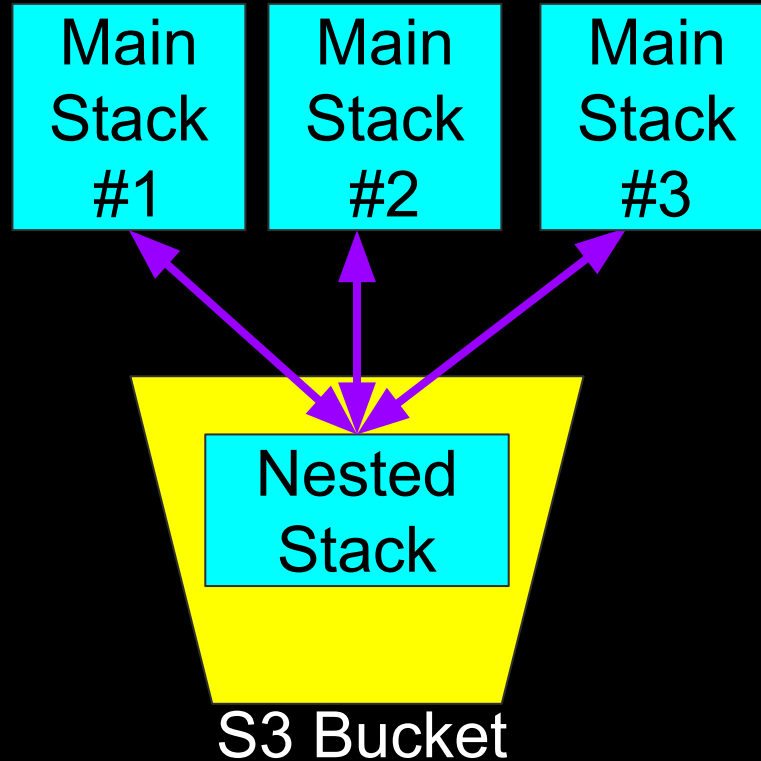
Nested Stacks

- Reuses one template in other stacks
 - Example - template to create web server EC2, reference it from other templates instead of copy pasting same code
- Nested stacks can contain other nested stacks
- Using nested stacks to declare common components is considered Best Practice

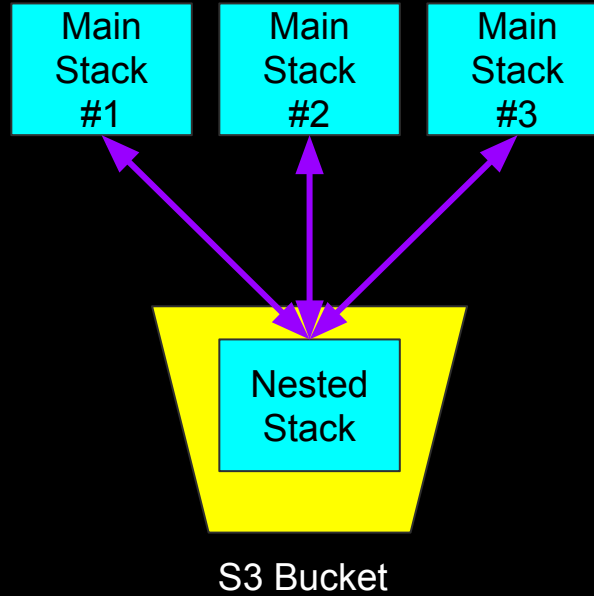
Nested Stacks



NESTED STACKS



Nested Stacks



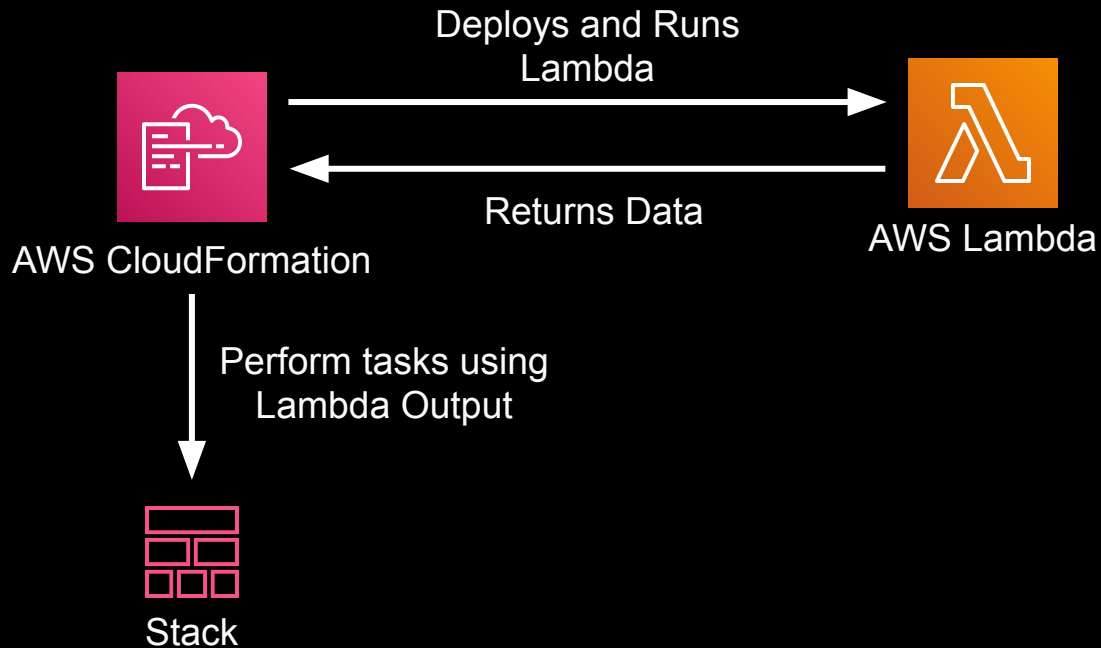
DEMO - Nested Stacks

- Simple Nested Stack
- Passing Parameters to Nested Stack

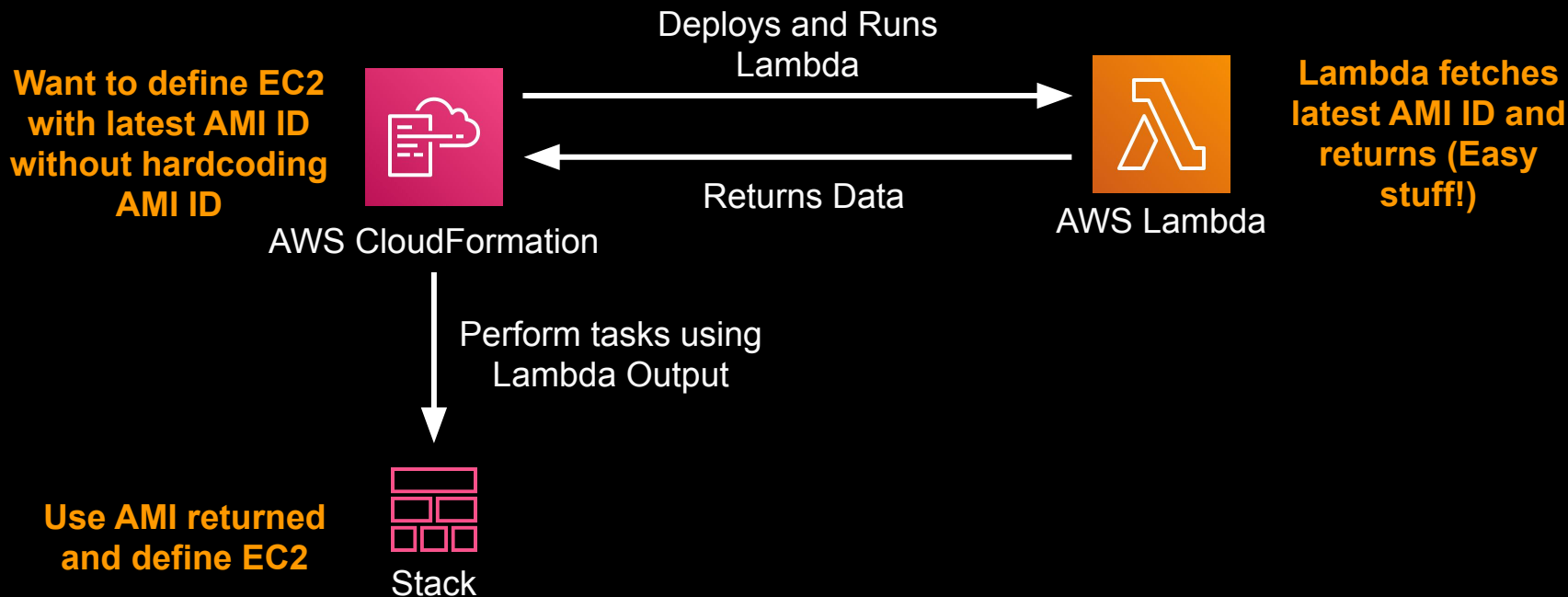
Custom Resources

- Enable writing custom provisioning logic in template
 - Perform tasks not included in CloudFormation Resource Types
 - Enables you to call Lambda (Do you feel powerful yet?)
- Let's take a look at the flow

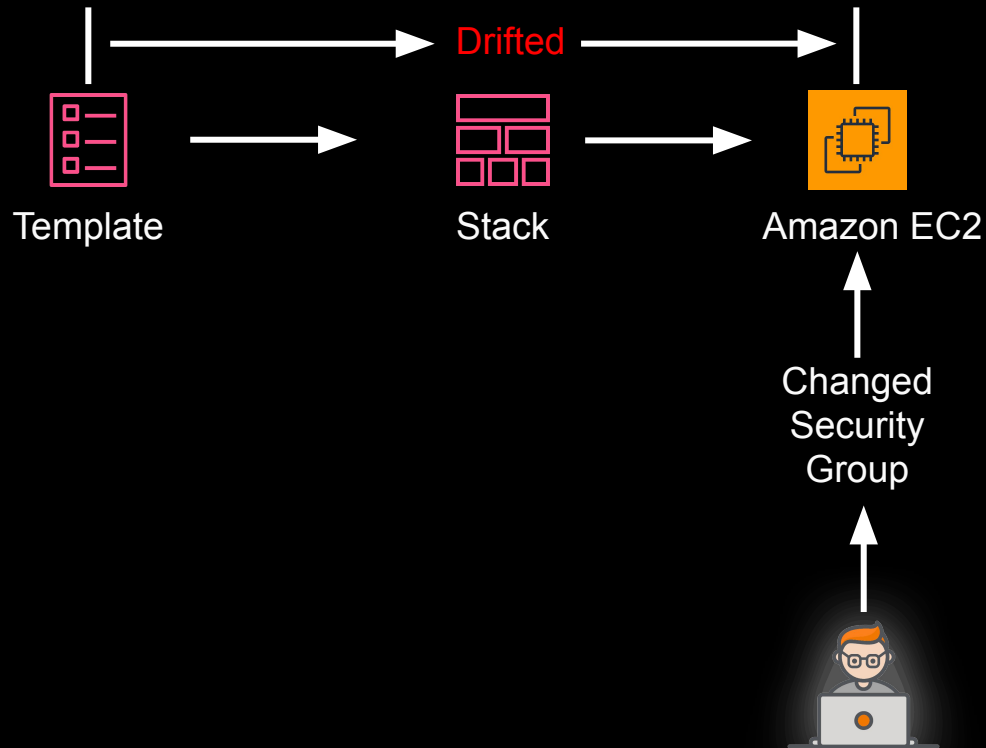
Custom Resources



Custom Resources - Example



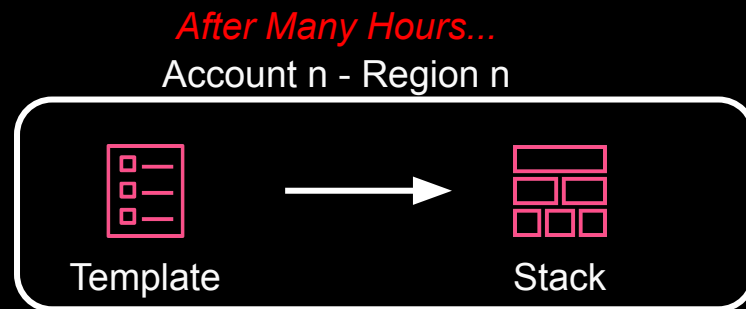
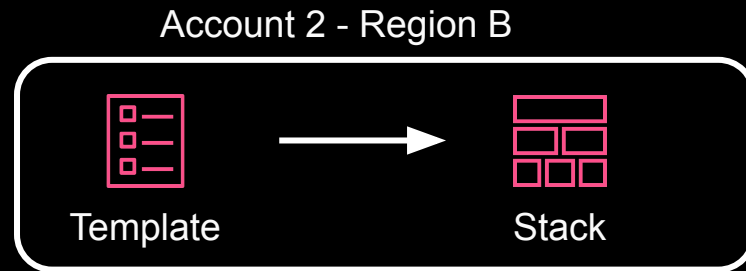
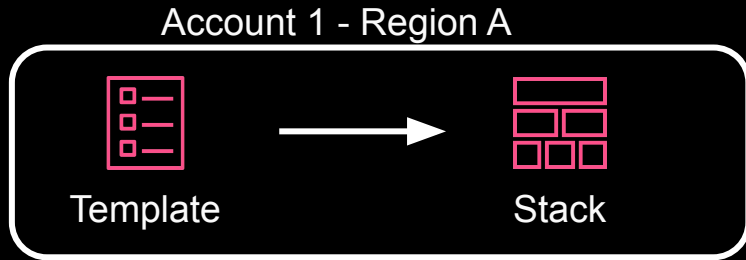
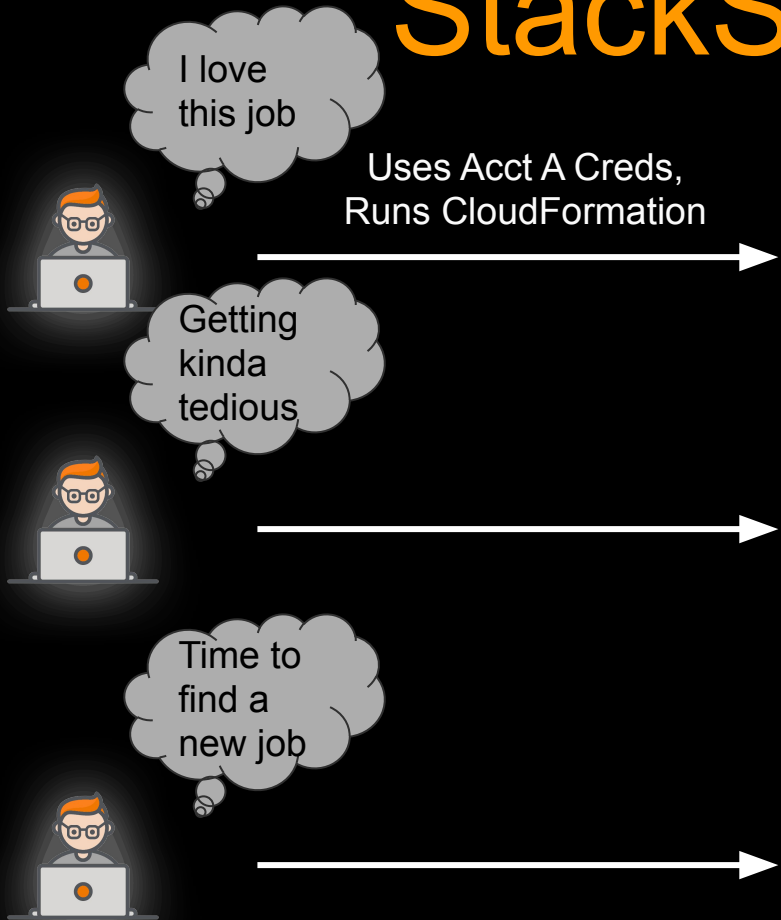
Drift Detection



Drift Detection - Why?

- Manual change of AWS Infra
 - Can't be reproduced in Prod (No console Update access)
 - Time intensive
 - Human error prone
- Change the Template instead
 - Refer Drift Details and adjust template
 - Implement through Change Set

StackSets - WHY?



Problem With This Approach

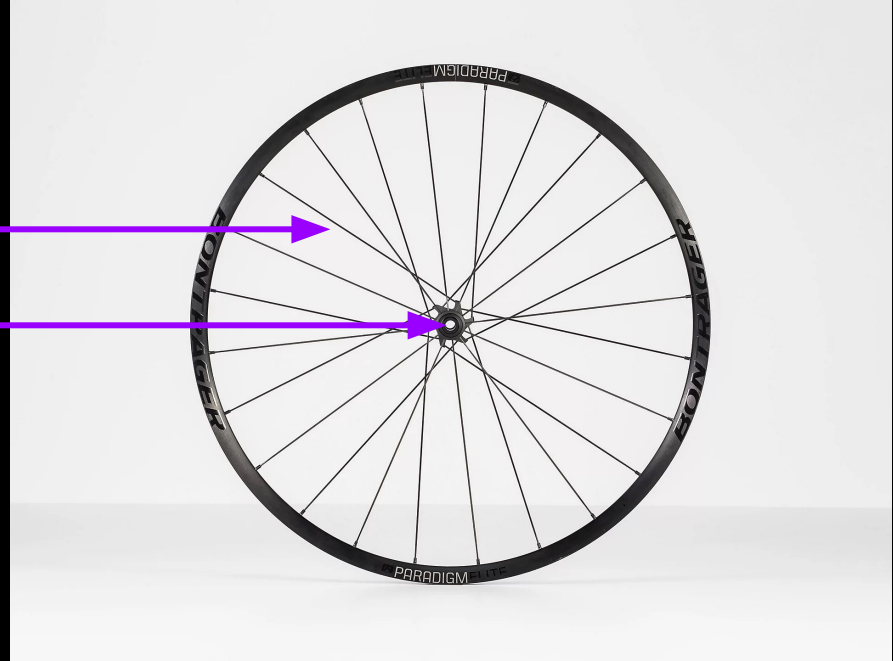
- CloudFormation is Regional
 - Very tedious to run same template in multi account/region
 - Slow and error prone
 - Even more tedious for Template changes!
- Security is BIGGER Issue
 - User (or Tool) has to maintain admin credentials for ALL Accounts
 - Multiple accounts to audit/track for access

In Real World, Hub and Spoke Model is Used instead!

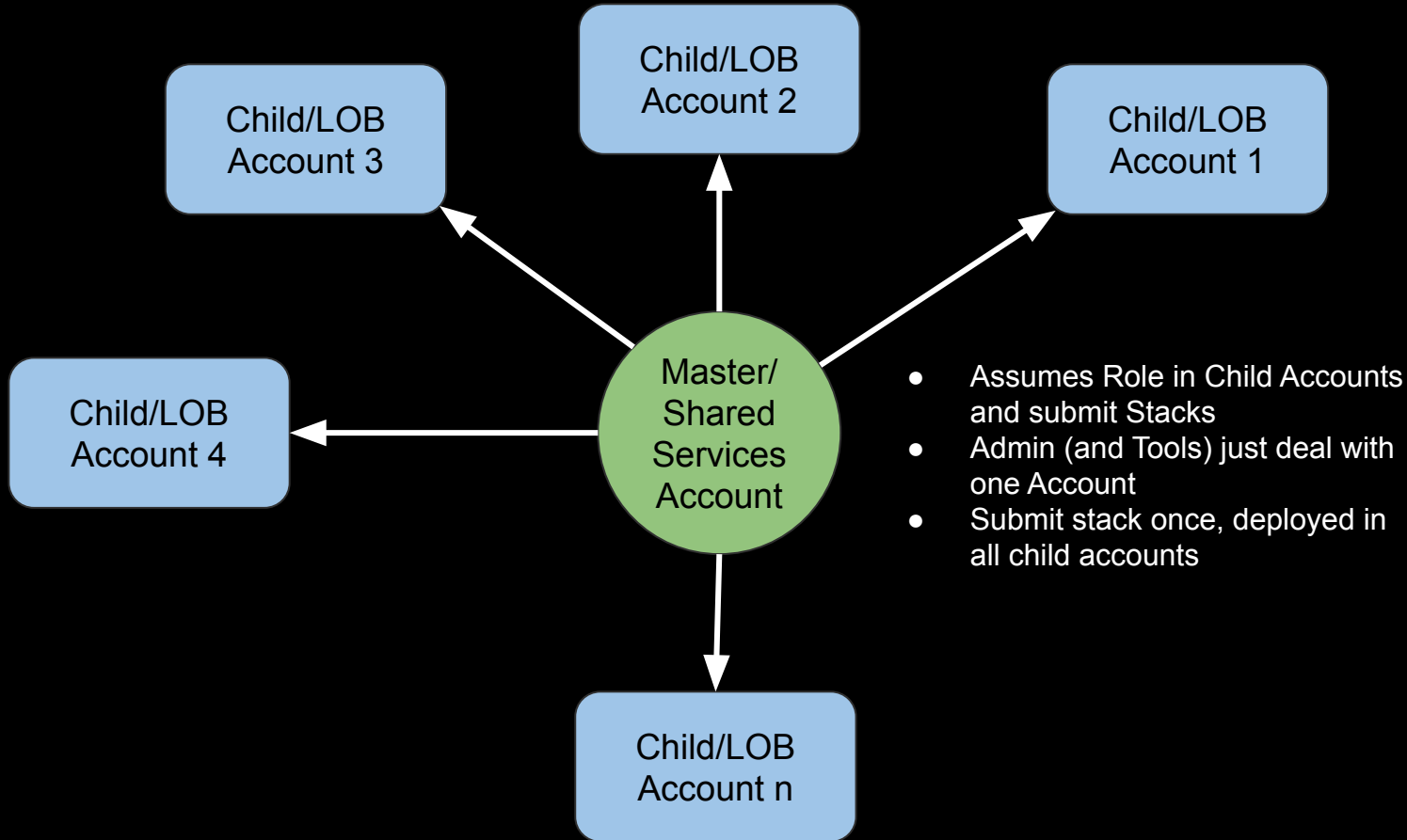
Hub And Spoke Model

Spoke

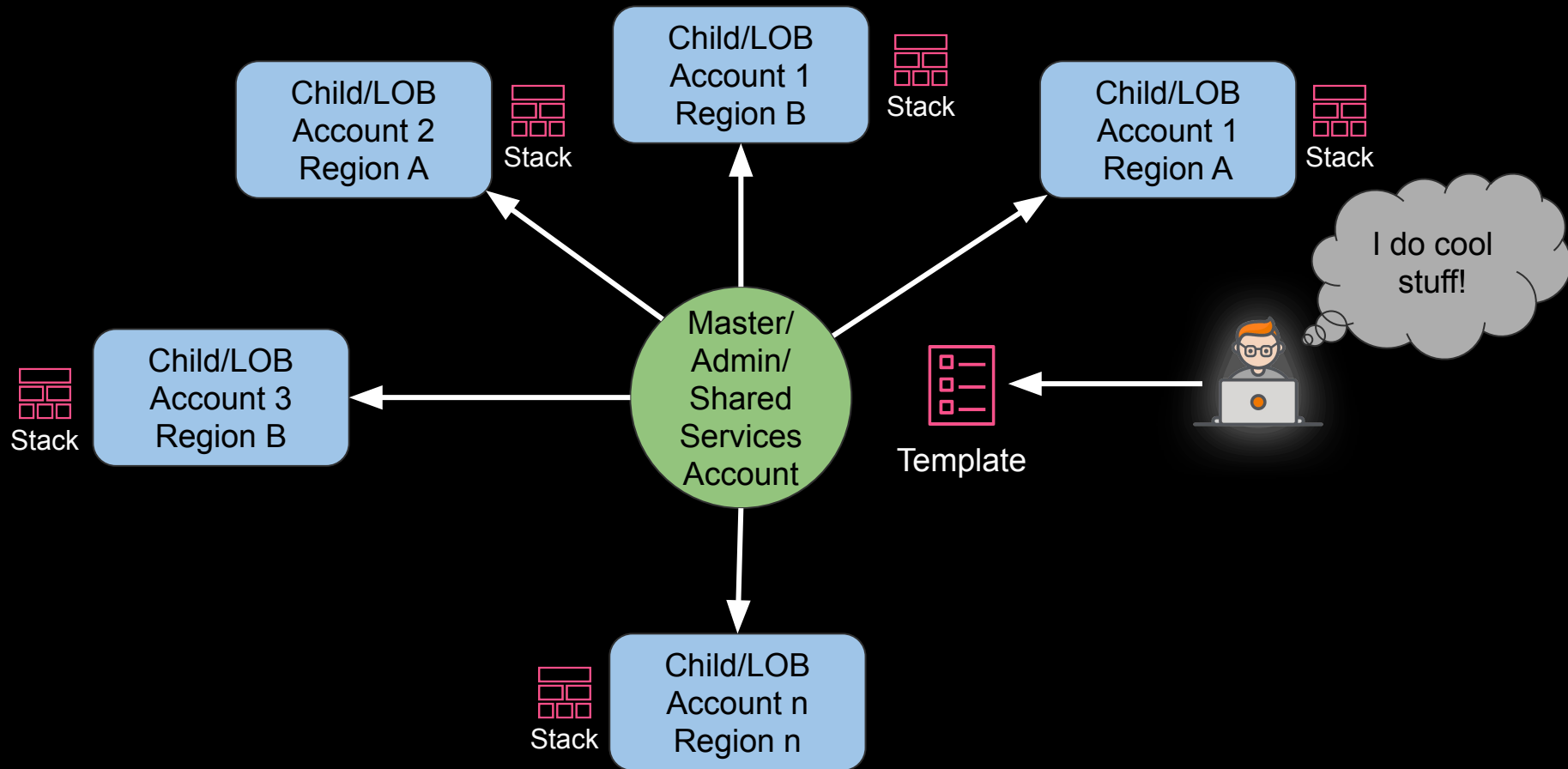
Hub



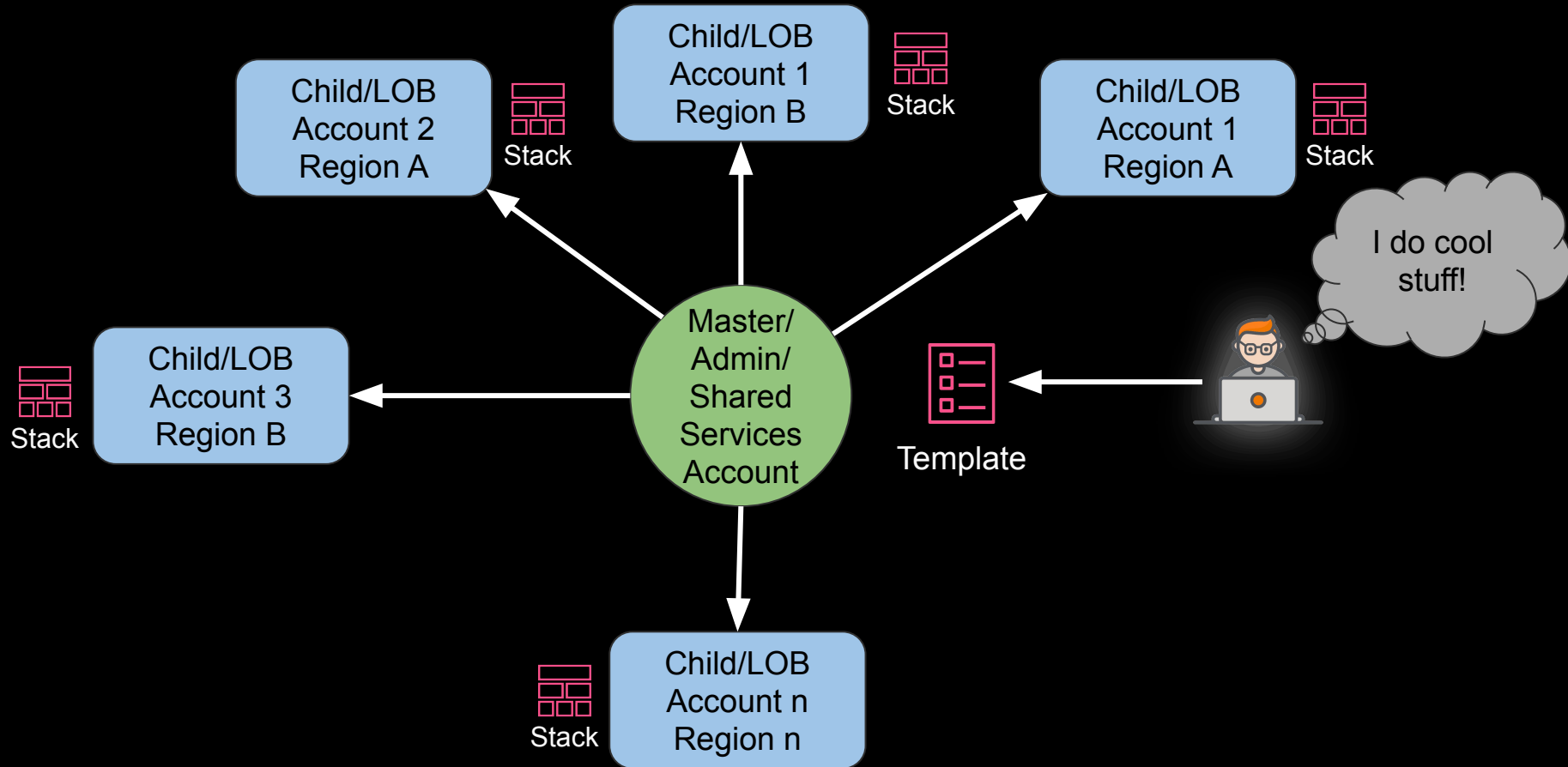
Hub And Spoke Model



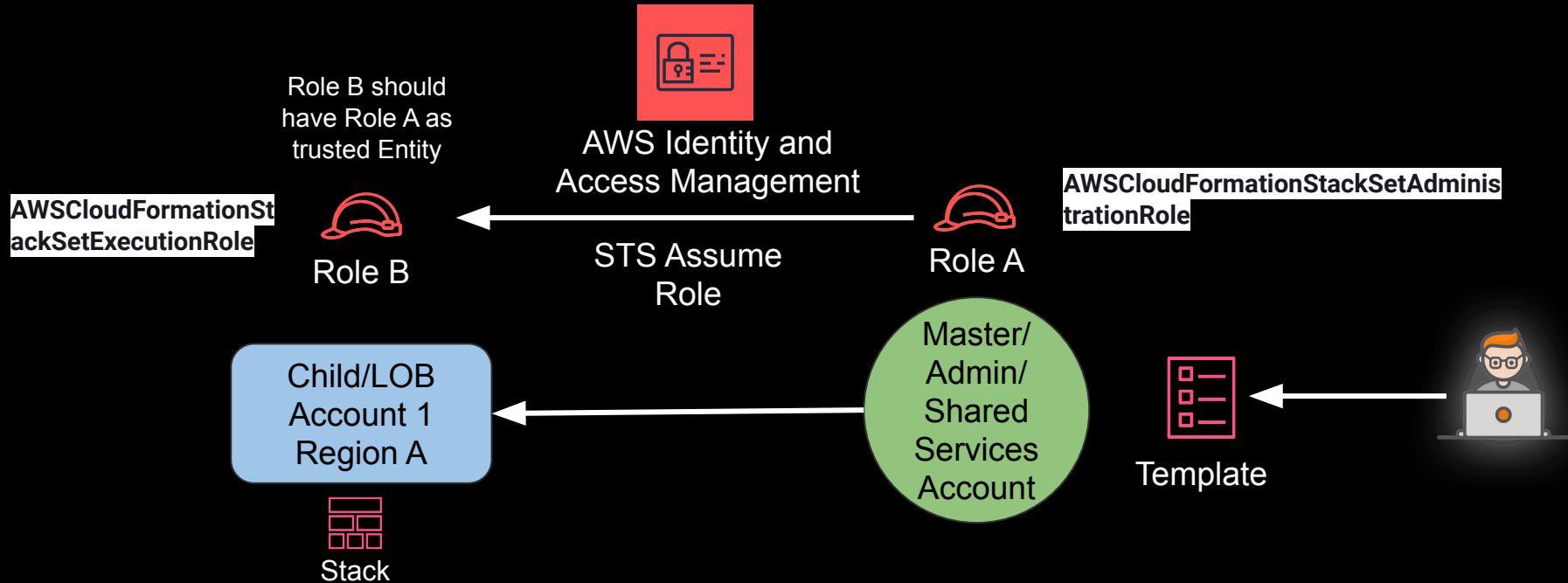
CloudFormation StackSet



DEMO - CloudFormation StackSet



DEMO - CloudFormation StackSet



REAL WORLD USE CASE

<pick from walkthroughs in
developer guide>

CloudFormation Best Practices

There is a LOT, no need to memorize everything

- Reuse Templates to Replicate Stacks in Multiple Environments
 - Same template can be used for dev, test and prod
 - Use Parameters, mappings, conditions
- Use Nested Stacks to Reuse Common Template Patterns
- Do Not Embed Credentials in Your Template
 - Input Parameters (with NoEcho) for sensitive info
 - Dynamic Parameters for Systems Manager/Secrets Manager

CloudFormation Best Practices

- Use AWS-Specific Parameter Types
 - Reduces user error
 - Gives drop-down with valid values
- Use Parameter Constraints
- Validate Templates Before Using Them
 - Console validation
 - AWS CLI - “aws cloudformation validate-template”
 - CloudFormation API - ValidateTemplate
- Don't Let the Stack Drift
 - No Manual Change of Resources

CloudFormation Best Practices

- Utilize Change Sets
- Use Code Reviews and Revision Controls for Cft
 - Save Cfts in Code Repo
 - Treat this as any other Production code (Infrastructure as CODE)

AWS CDK

Constructs

- Basic Building Block of CDK apps
- Represent a single resource, such as S3 bucket
- Think of it as CloudFormation Resources

Constructs

- Constructs can be higher-level, intent-based API
- Example - S3 Bucket class with additional properties and methods, such as lifecycle rules

Constructs

- Constructs creating multiple resources - patterns
- Example - AWS Fargate cluster creating Application Load Balancer, Fargate

Easier to understand with a demo

CDK DEMO

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources

Note - CDK Demos will be shown in Cloud9

CDK DEMO

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources

CDK DEMO

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources

CDK DEMO

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources

CDK DEMO - PYTHON

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources
 - S3
 - EC2
 - **IMPORTANT** - HOW TO SPIN UP ANY
INFRA READING THE MANUAL

CDK DEMO - PYTHON

- CDK Boilerplate
 - TypeScript
 - Python
- Add Additional Resources
 - S3
 - LAMBDA + API GATEWAY
 - **IMPORTANT** - HOW TO SPIN UP ANY INFRA READING THE MANUAL

CDK STEPS - TS & PYTHON

- Boilerplate - `cdk init` & (`npm watch` OR `virtualenv`) & `cdk bootstrap`
- Include package for the construct
 - *e.g. - `import s3 = require('@aws-cdk/aws-s3');`*
 - *e.g. - `from aws_cdk import (aws_s3 as s3)`*
- Install the package
 - *e.g. - `npm install @aws-cdk/aws-s3`*
 - *e.g. - `pip install aws-cdk.aws-s3`*
- Read Construct Reference
 - <https://docs.aws.amazon.com/cdk/api/latest/docs/aws-construct-library.html>
- Code!
- Deploy
 - `cdk diff` - Show what's changing
 - `cdk synth` - Show CloudFormation
 - `cdk deploy` - Deploy the App

CDK STEPS - TS/PYTHON

1. Boilerplate (One Time Setup) - `cdk init` & (`npm watch` OR `virtualenv`) & `cdk bootstrap`
2. Read Construct Reference
 - a. <https://docs.aws.amazon.com/cdk/api/latest/docs/aws-construct-library.html>
3. Include package for the construct
 - a. e.g. - `import s3 = require('@aws-cdk/aws-s3');`
 - b. e.g. - `from aws_cdk import (aws_s3 as s3)`
4. Install the package
 - a. e.g. - `npm install @aws-cdk/aws-s3`
 - b. e.g. - `pip install aws-cdk.aws-s3`
5. Code!
6. Deploy
 - a. `cdk diff` - Show what's changing
 - b. `cdk synth` - Show CloudFormation
 - c. `cdk deploy` - Deploy the App

CDK STEPS

- Boilerplate - `cdk init & npm watch & cdk bootstrap`
- Include package for the construct
 - *E.g- `import s3 = require('@aws-cdk/aws-s3');`*
- Install the package
 - *E.g - `npm install @aws-cdk/aws-s3`*
- Read Construct Reference
 - <https://docs.aws.amazon.com/cdk/api/latest/docs/aws-construct-library.html>
- Code!
 - *E.g - `const bucket = new s3.Bucket(this, 'cdkbucket', {versioned:true})`*
- Deploy
 - **cdk diff** - Stackset with differences
 - **cdk synth** - Show CloudFormation
 - **cdk deploy** - Deploy the App



**Give a man a fish and you feed
him for a day; teach a man to fish
and you feed him for a lifetime.**

Maimonides

CI/CD Tools

Why Automate Cloudformation?



- Time Consuming
- Error Prone
- Higher Cost (talented developer's time can be used elsewhere!)
- Not-Cool for Developers!

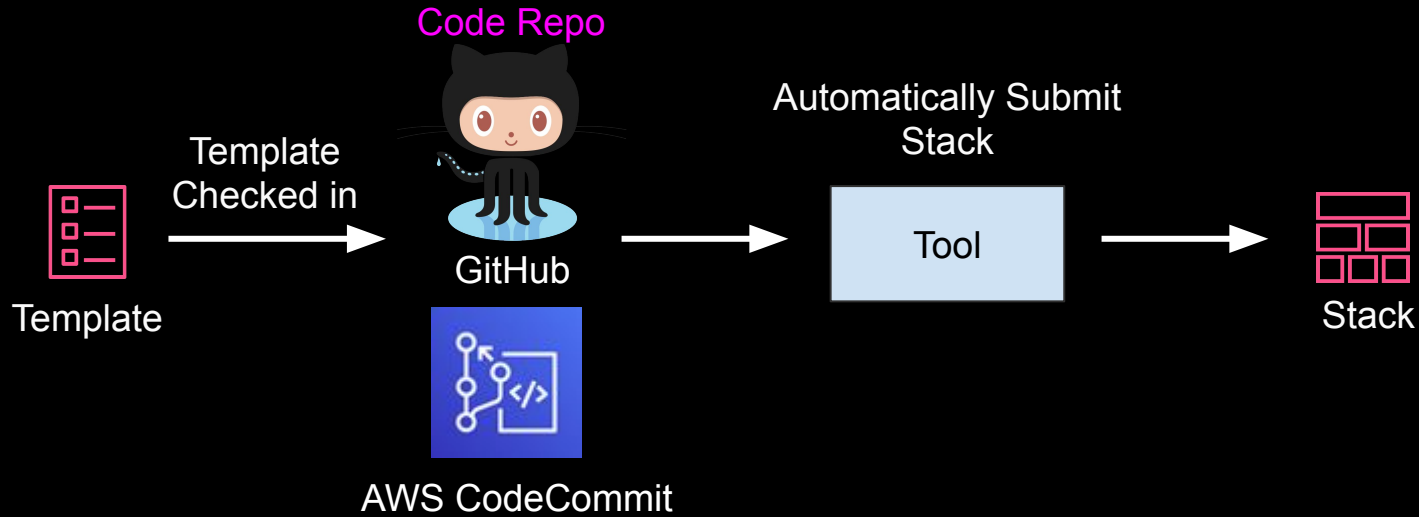
Running CloudFormation From DevOps Tools

A Quick Word!

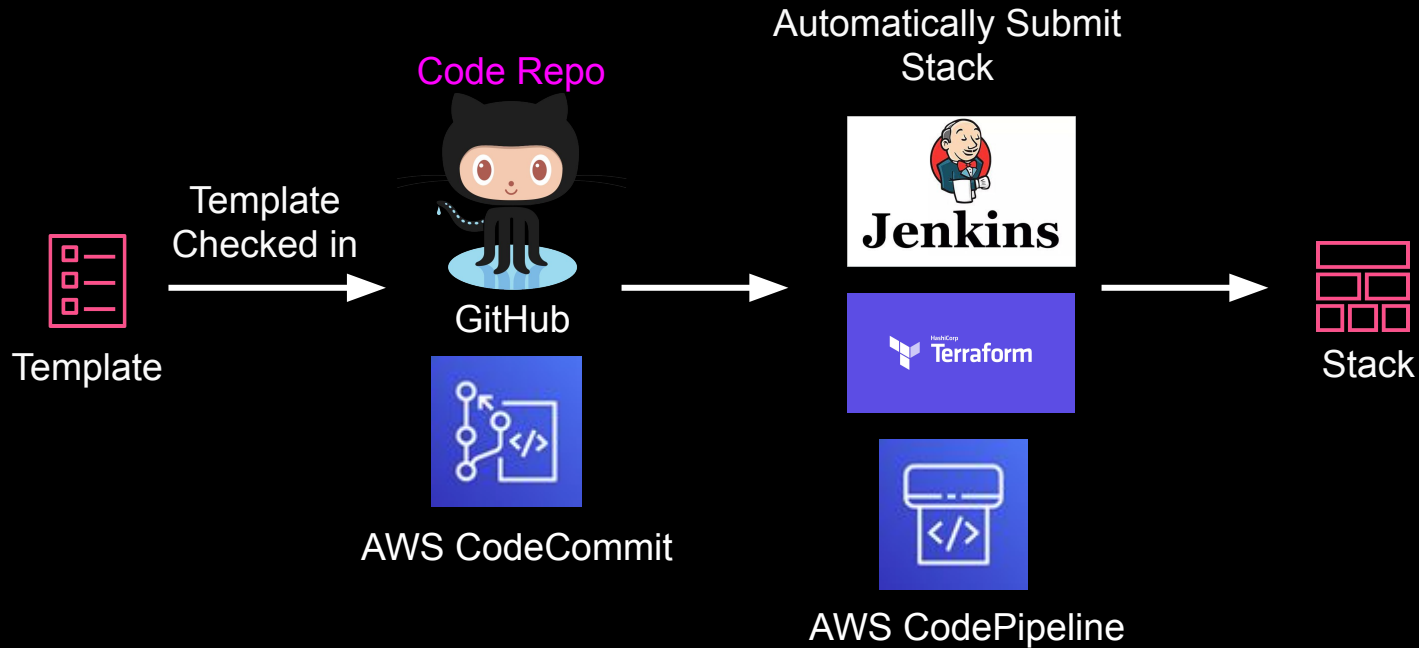
Ways to Run CloudFormation

- AWS CloudFormation Console
- CloudFormation API
 - Anything that can call API - Lambda, EC2, AWS CLI etc.
- DevOps Tools
 - Jenkins
 - CloudFormation Plugin
 - Pipeline
 - Terraform
 - AWS CodePipeline
- Million other tools/services - coz API!

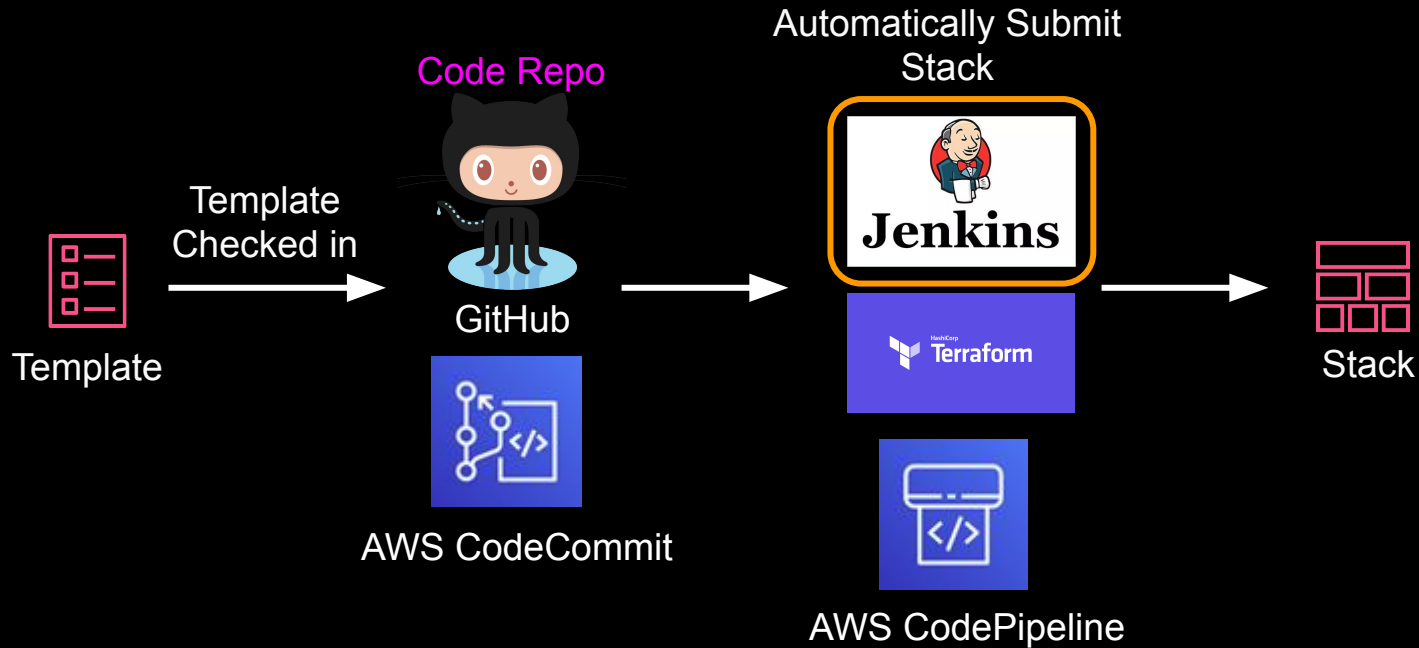
Tools - Conceptual View



Tools - Conceptual View



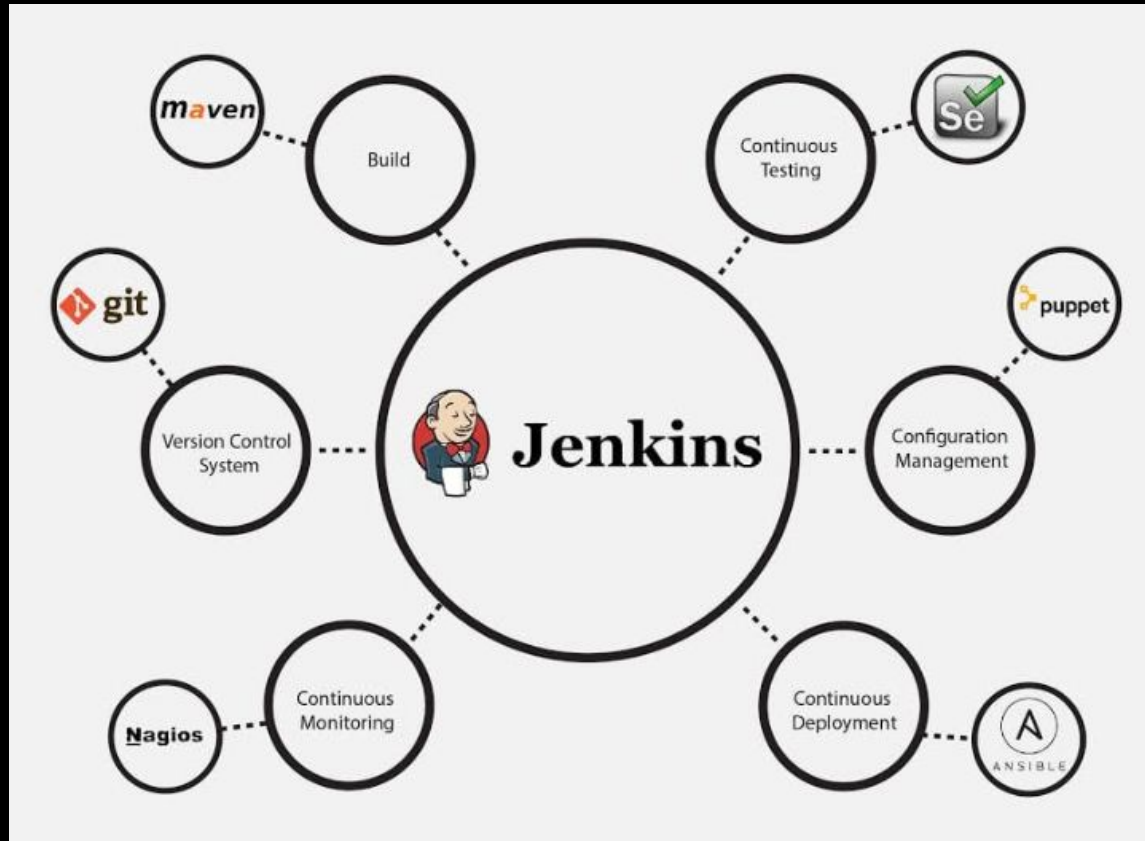
Tools - Conceptual View



Jenkins

- Open source CI/CD Tool
- Utilize Plugin for tasks - build, test, and CloudFormation etc.
- 1400+ Plugins - new plugin can be written and shared
- The most popular open source tool

Jenkins



Jenkins & CloudFormation

- CloudFormation Plugin
- Using Pipeline, run AWS CLI
CloudFormation Command

Jenkins Demo

- Install Jenkins
 - Marketplace AMI (Free!)



Jenkins Demo

- Install Jenkins
 - Marketplace AMI (Free!)
- Install CloudFormation Plugin
- Run Cft using Cft plugin
- Run Cft using Jenkins pipeline
 - Uses AWS CLI Commands



Jenkins Tutorial

Part 1

- Install Jenkins
 - Marketplace AMI (Free!)
- Install CloudFormation Plugin
- Run Cft using Cft plugin

Part 2

- Run Cft using Jenkins pipeline
 - Uses AWS CLI Commands



Jenkins Demo

- Install Jenkins
 - Marketplace AMI (Free!)
- Install CloudFormation Plugin
- Run Cft using Cft plugin
- Run Cft using Jenkins pipeline
 - Uses AWS CLI Commands

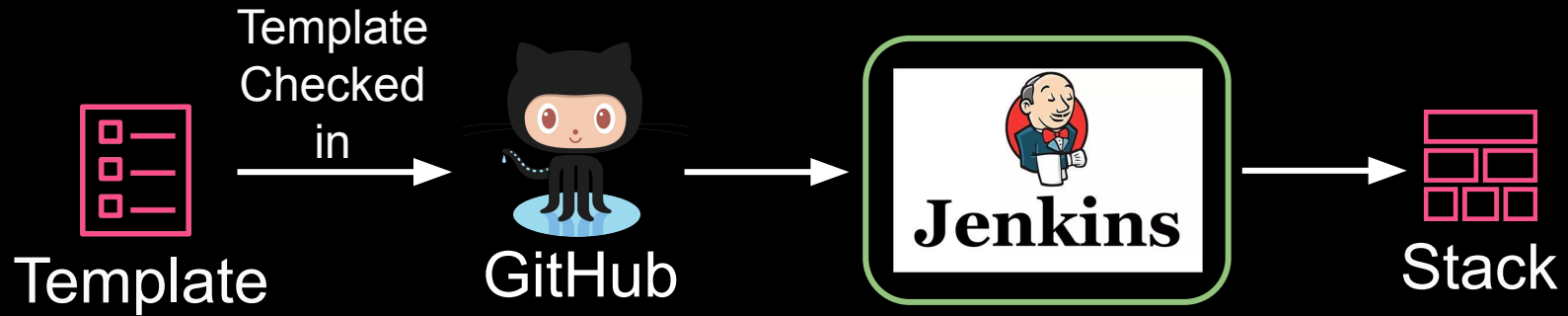


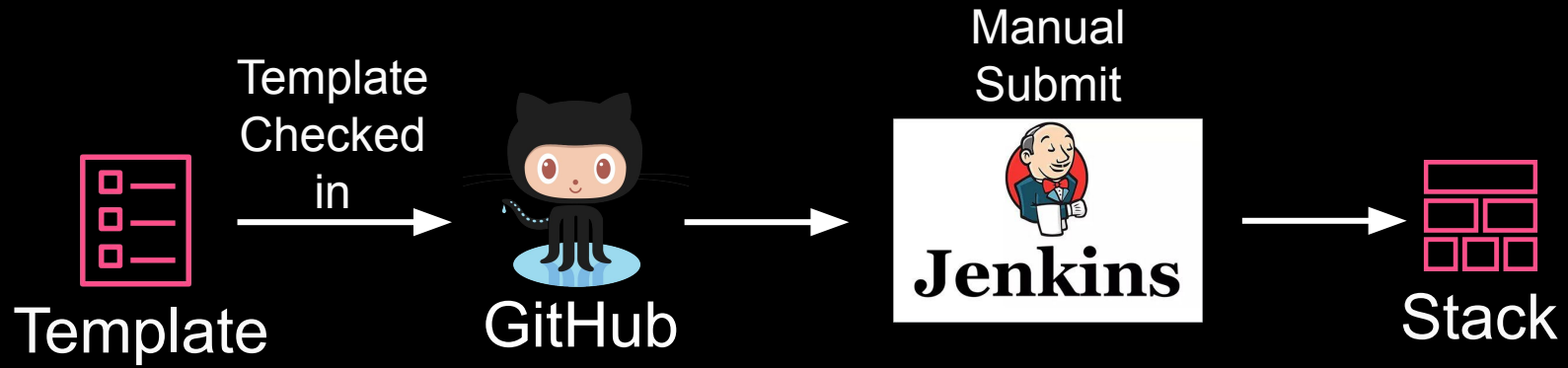
Jenkins Demo

- Install AWS CLI
 - SSH and Install AWS CLI
 - Use SSM Session Manager to Install without any keys
- Jenkins File
- Enhance EC2 Role
- Run Cft using Jenkins pipeline
 - Uses AWS CLI Commands

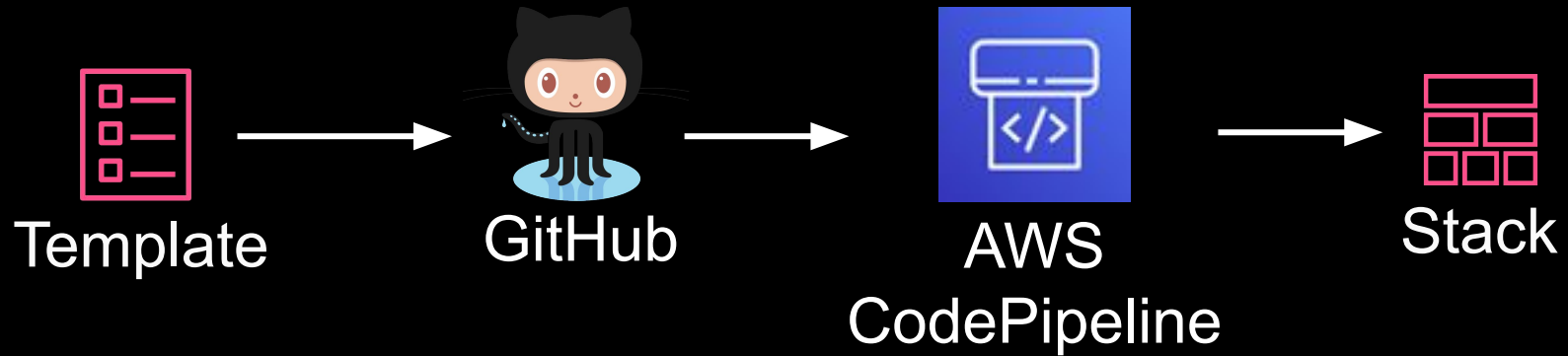
INSTALL JENKINS ON EC2 (EASY WAY)

RUN CLOUDFORMATION FROM GIT

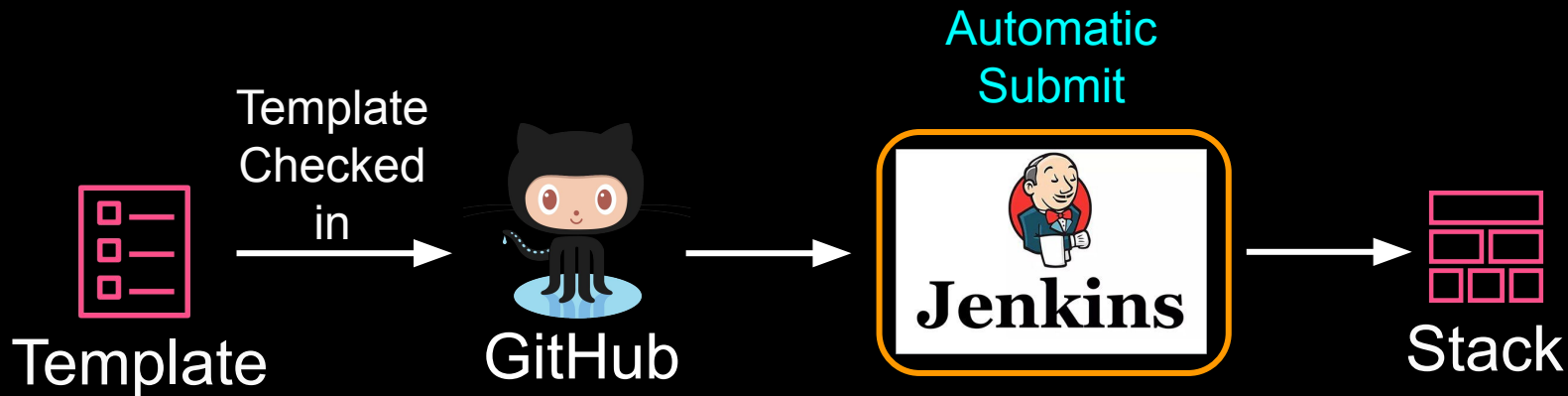




CLOUDFORMATION PARAMETERS CODEPIPELINE



AUTOMATICALLY TRIGGER JENKINS JOB FROM GITHUB



Jenkins Demo

- Install Cft Plugin
- Run Cft
 - With Parameters
- Webhooks

Jenkins Demo

- Run Cft using Pipeline

Jenkins vs CodePipeline

Jenkins

Need to maintain Master and Worker Nodes in VM

Pay for idle resources

You take care of availability and scalability

Very mature plugin ecosystem

Can use CodeBuild as worker node

CodePipeline

No need to provision and manage any Server

Pay as you go

Fully managed and scales automatically

Have some prepackaged environments

Inherent integration with other AWS Services

CloudFormation

Works to provision AWS
Services only

Free, included within AWS Support

Manages state by stack

Can have wait conditions

Not possible to bring existing
resources into stack

Terraform

Cloud Agnostic

Open source, Enterprise version
costs \$ and includes support

State stored in local disk by
default, can use remote state

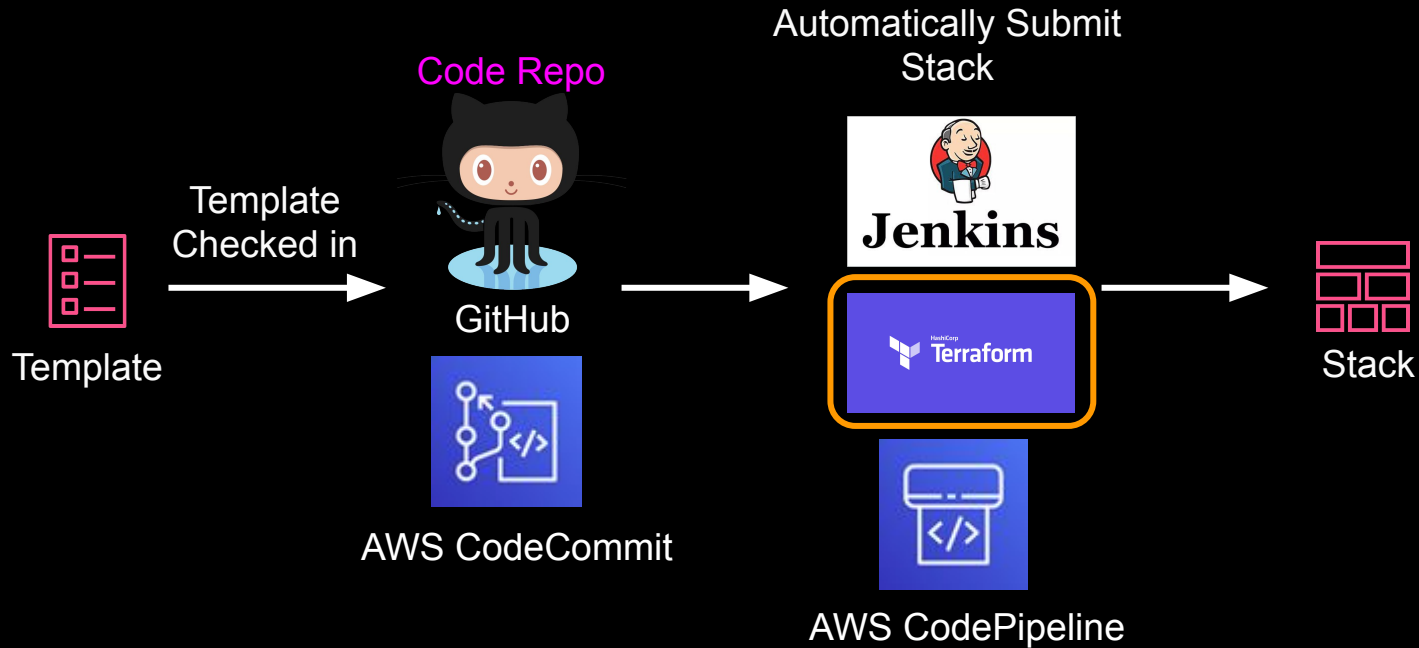
Terraform does not have wait
conditions

Import existing resources

All Opinions are my own



Tools - Conceptual View



Terraform

- Open source tool for building, changing, and versioning infrastructure
 - Enterprise version has more features
- Works with multi-cloud
- Uses HCL (HashiCorp Configuration Language)



Terraform Demo

- *Prerequisite - AWS CLI*
- Install Terraform
 - Include in Path
- Write HCL for EC2
 - Visual Studio Code Plugin
- Spin Up EC2

SafeGuard Options

- Termination Protection
- Deletion Policy
- IAM Policy
- Stack Policy

Deletion Policy

- Resource Level Safeguard in Cft

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" :
  {
    "myS3Bucket" : {
      "Type" : "AWS::S3::Bucket",
      "DeletionPolicy" : "Retain"
    }
  }
}
```

IAM Policy

- IAM level policy to control who can delete stack

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": "cloudformation:DeleteStack",
    "Resource": "arn:aws:cloudformation:us-east-1:123456789012:stack/MyProductionStack/*"
  }]
}
```

Stack Policy

- Stack level JSON to control update of particular resources in the stack


```
{ "Statement": [
  {
    "Effect": "Deny",
    "Action": [ "Update:Delete", "Update:Replace" ],
    "Principal": "*",
    "Resource": "*",
    "Condition": {
      "StringEquals": {
        "ResourceType": [ "AWS::RDS::DBInstance" ]
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": "Update:*",
    "Principal": "*",
    "Resource": "*"
  }
]
```


Infrastructure As Code

Interview Questions



I should have taken
Rajdeep's course



Should have
watched
Raj's video

What are
DevOps
Phases, CI and
CD?

General Cloud Interview Guidance

1. Have a personal github repo ready
2. Practice whiteboarding at home
3. Demonstrate that you know recent announcements - major service change, acquisitions etc.
4. Adjust answers based on the job title
5. Try to take part in hackathon and events - connections!

Main Interview Topics

- Can YOU code CloudFormation?
- Can YOU automate CloudFormation deployment?
- Do YOU keep up with changes?

Can YOU code CloudFormation?

- Parameters (MUST!)
 - How do you accept parameters from user?
 - How do you restrict user to choose predefined values?
 - Different types of Parameters
 - What Intrinsic Functions do you use with Parameters?
 - Can you write an example (in whiteboard/paper) of a Cft with parameters?

Can YOU code CloudFormation?

- Mappings (MUST!)
 - Scenario based question - many accounts, many VPCs, many AMIs - how do you select automatically?
 - Advanced Follow Up - I don't want to change the mapping everytime anything changes, any alternative? Custom Resources
 - Max number of Mappings in a Cft - 100
 - Advanced Follow Up - Use Nested Stacks to work around
 - What Intrinsic Functions do you use with mappings?
 - Can you write an example (in whiteboard/paper) of a Cft with mappings?

Can YOU code CloudFormation?

- How do you estimate cost for CloudFormation?
 - Follow up - How do you control costs for CloudFormation?
 - Call additional API from CI/CD Tool to get Cft cost and budget. If exceeded, don't submit cft, send notification
 - Resource Constraints with Service Catalog (Out of Scope)
 - Standardize template for good hygiene - check EC2 size, EC2 "up all the time" tag etc.
- Best practices of CloudFormation

CloudFormation CI/CD

- This is your chance to impress!
 - Most folks just know the cft template
 - Mention different deployment tools, pros cons to set yourself apart
 - My favorite topic - sets apart from Pen and Paper Architect



CloudFormation CI/CD

- Why do we need DevOps for CloudFormation?
- What DevOps tools are you familiar with for Infrastructure as Code?
 - Make sure you mention at least one cloud native (Code PipeLine) and one cloud agnostic tool (Jenkins/terraform)
 - Good chance to ask back what tools does the company use and adjust future answers
 - Jenkins is the most popular tool
- How do you use Cft with Jenkins?
- Hub and Spoke Model (Mention StackSet)

CloudFormation CI/CD

- Pros/Cons of DevOps Tools(refer to Jenkins Vs Terraform Vs CodePipeline video)
- How do you create DR environment?
 - Always use infra as code
 - Utilize mappings for different environments
 - Mention different DR strategies (Link in description) and dive deep as needed

Ready To Learn and Adopt

- Do you know about CDK or Infrastructure as Code?
- Can you mention any recent announcements that impacted your CloudFormation?
 - Change Set
 - Drift Detection
 - CloudFormer
 - Look up any recent changes
- Other Cloud News
 - Recent mergers
 - Major Service Change