Wifi / Password

Wifi name:
TDPK-WIFI

Username:
TrueIDCAWS_On-boardingworkshop
Password
Welcome@2022

https://github.com/TIDC-PS-Inter/AWS-Workshop

Part 1

AWS Workshop Series

Day 8: Infrastructure as Code

Taking Enterprise Beyond the Cloud by TrueIDC

Mr. Niran Sohinkong

Professional Service Manager

Presented by

- Niran Sohinkong (Nueng)
- Professional Service Manager, TrueIDC
- AWS DevOps
- AWS SysOps / Architect
- niran.soh@ascendcorp.com

trueIDC

Agenda

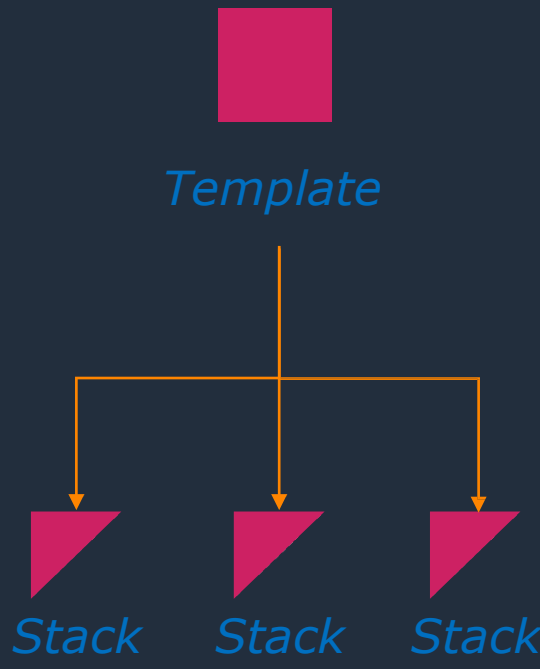
- Infrastructure as Code Overview
- What is CloudFormation
- CloudFormation Syntax
- CloudFormation Stack
- CloudFormation Designer
- Template Anatomy
- Lab

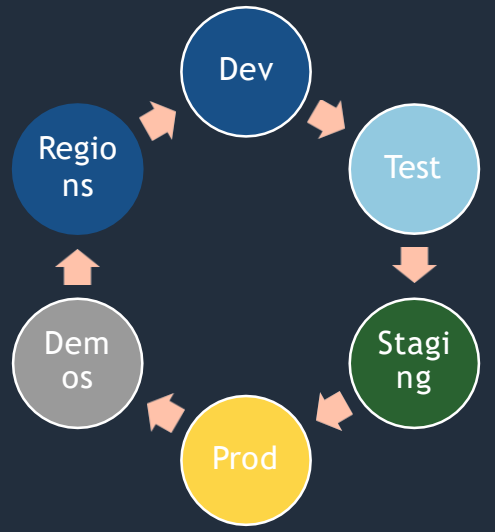
Infrastructure as code – Automation tools

aws

Infrastructure as code

- Single source of truth to deploy the whole stack
- Infrastructure that you can replicate, re-deploy, and re-purpose
- Control versioning on your infrastructure and your application together
- Service rolls back to the last good state on failures
- Build your infrastructure and run it through your CI/CD pipeline





AWS CloudFormation

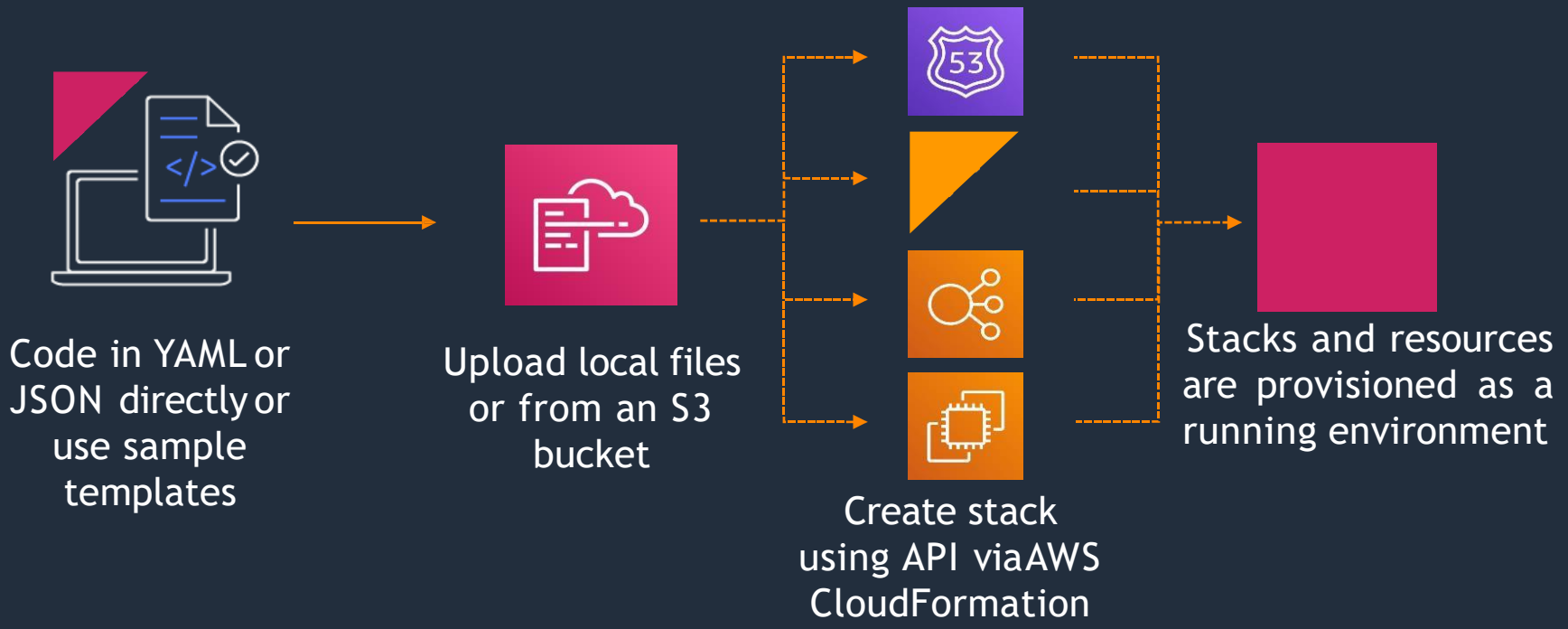
What is AWS CloudFormation?

- Simplified way to create and manage a collection of AWS resources
- Enables orderly and predictable provisioning and updating of resources
- Enables version control of your AWS infrastructure
- Deploy and update stacks using the AWS Management Console, the AWS Command Line Interface (CLI), or the AWS API
- Only pay for the resources you create

AWS CloudFormation

- Declarative and flexible
- Transparent and open
- Customizable (parameters)
- Integration ready
- Don’t reinvent the wheel
- No extra charge

CloudFormation Overview



- JSON/YAML format template
- Presents template to AWS CloudFormation
- AWS CloudFormation translates it to an API request
- Forms a stack of resources

- FREE – you only pay for resources
- All regions
- APIs are called in parallel
- Manages **dependencies/relationships**

AWS CloudFormation syntax

- JSON – JavaScript object notation
- Attribute-value pairs (Key:Value)
- Similar to XML

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {
    "S3NameParam":{  "Type":
      "String",  "Default":
      "mybucket",
      "Description": "Namefor your AWSS3bucket",
      "MinLength": 5,
      "MaxLength": 30
    }
  },
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {
        "AccessControl": "PublicRead",
        "BucketName": {"Ref" : "S3NameParam"}
      },
      "DeletionPolicy": "Retain"
    }
  },
  "Outputs": {
    "Bucketname": {
      "Description": "Nameof AWSS3 Bucket"
    }
  }
}
```

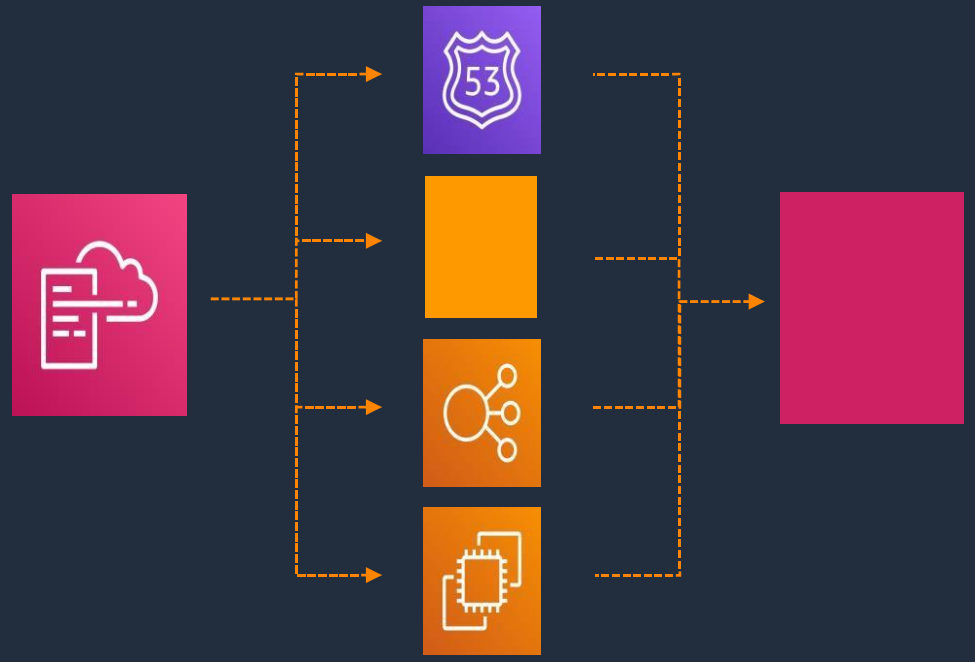
AWS CloudFormation syntax

- YAML – Not a markup language
- YAML is a human friendly data serialization standard
- Comments - Use #

```
AWSTemplateFormatVersion: '2010-09-09'
Parameters:
  S3NameParam:
    Type: String
    Default: mybucket
    Description: Name for your AWS S3 bucket
    MinLength: 5
    MaxLength: 30
Resources:
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      AccessControl: PublicRead
      BucketName:
        Ref: S3NameParam
    DeletionPolicy: Retain
Outputs:
  Bucketname:
    Description: Name of AWS S3 Bucket
```

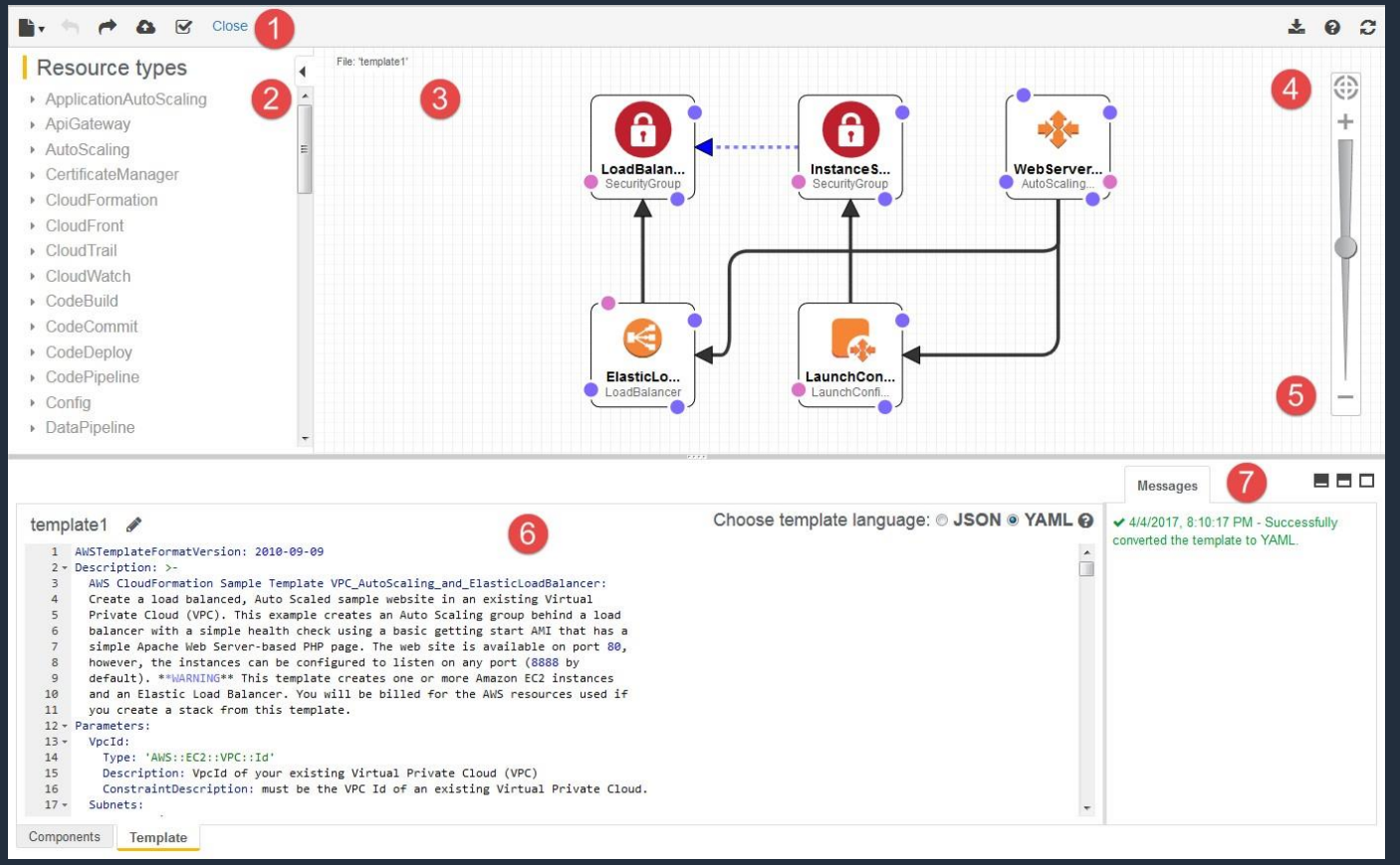
CloudFormation Stack

- A stack is a collection of AWS resources that you can manage as a single unit, or a template
- AWS CloudFormation ensures all stack resources are created or deleted as appropriate
- You can work with stacks by using the AWS CloudFormation console, API, or AWS CLI.
- *Nested stacks* are stacks created as part of other stacks. You create a nested stack within another stack by using the AWS::CloudFormation::Stack resource



AWS CloudFormation Designer

- Graphical tool for creating, viewing, and modifying CloudFormation templates.
- Drag-and-drop interface
- Integrated JSON and YAML editor.



https://console.aws.amazon.com/cloudformation/designer

Template Anatomy

- Use templates to create and manage stacks
- JSON or YAML-formatted text files that describe your AWS infrastructure
- AWS CloudFormation JSON template structure and sections

```
{
   "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
   }
}
```

Template Anatomy

- Format version
- *Transform*
- Description
- Metadata
- Parameters
- Mappings
- Conditions
- Resources* (required)
- Outputs

Reference: http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/crpg-ref.html

Template Anatomy - Resources (required)

- Only section that is **not optional**
- Define AWS resources to create/update
- Supports 459 resource types (and growing)
- Refer to the CloudFormation User Guide for updated list

```
Resources:
  EC2Instance:
    Type: AWS::EC2::Instance
    Properties:
    InstanceType:
       Ref: InstanceType
     SecurityGroups:
     -Ref: InstanceSecurityGroup
     KeyName:
       Ref: KeyName
     ImageId:
     Fn::FindInMap:
      - AWSRegionArch2AMI
      - Ref: AWS::Region
      - Fn::FindInMap:
        - AWSInstanceType2Arch
        - Ref: InstanceType
        - Arch
```

Template Anatomy - Format Version and Description

- Format version
  - Currently only supports 1 value “2010-09-09”
- Description
  - JSON/YAML string where you provide a Description (optional)

```
{
  "AWSTemplateFormatVersion": "version date",
  "Description": "JSON string",
}
```

Template Anatomy - Metadata

Arbitrary JSON/YAML objects that provide additional details about the template

```
{
   "AWSTemplateFormatVersion": "version date",
   "Description": "JSON string",
   "Metadata": {
      "Instances": {
         "Description": "Information about the instances"
      },
      "Databases": {
         "Description": "Information about the databases"
      }
   },
   ....
}
```

Template Anatomy - Parameters

- Enable you to input custom values to your template each time you create or update a stack
- Supports parameter types: String, Number, List<Number>, CommaDelimitedList, AWS-Specific types and SSM types
- Use the Ref intrinsic function to reference a parameter

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Parameters": {




    }
  },
  "Resources": {
    "S3Bucket": {
      "Type": "AWS::S3::Bucket",
      "Properties": {
        "AccessControl": "PublicRead",


        }
      },
      "DeletionPolicy": "Retain"
    }
  },
  "Outputs": {
    "Bucketname": {
      "Description": "Name of AWS S3 Bucket"
    }
  }
}
```

Template Anatomy - AWS-Specific Parameter Types

- Validates parameter values against existing values in users' AWS accounts
- Catches invalid values when you start creating or updating a stack
- AWS::CloudFormation::Interface – metadata key that defines how parameters are grouped and sorted in the AWS CloudFormation console

```
Metadata:
  AWS::CloudFormation::Interface:
    ParameterGroups:
      - ParameterGroup
    ParameterLabels:
      ParameterLabel
```

AWS::EC2::AvailabilityZone::Name
AWS::EC2::Image::Id
AWS::EC2::Instance::Id
AWS::EC2::KeyPair::KeyName
AWS::EC2::SecurityGroup::GroupName
AWS::EC2::SecurityGroup::Id
AWS::EC2::Subnet::Id
AWS::EC2::Volume::Id
AWS::EC2::VPC::Id
AWS::Route53::HostedZone::Id
List<AWS::EC2::AvailabilityZone::Name>
List<AWS::EC2::Image::Id>
List<AWS::EC2::Instance::Id>
List<AWS::EC2::SecurityGroup::GroupName>
List<AWS::EC2::SecurityGroup::Id>
List<AWS::EC2::Subnet::Id>
List<AWS::EC2::Volume::Id>
List<AWS::EC2::VPC::Id>
List<AWS::Route53::HostedZone::Id>

See AWS-Specific Parameter Types

Template Anatomy - Intrinsic Function

Intrinsic

- Fn::Base64
- Fn::Cidr
- Fn::FindInMap
- Fn::GetAtt
- Fn::GetAZs
- Fn::ImportValue
- Fn::Join
- Fn::Select
- Fn::Split
- Fn::Sub
- Fn::Transform
- Ref

Intrinsic (Conditionals)

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

Pseudo

- AWS::AccountId
- AWS::NotificationARNs
- AWS::NoValue
- AWS::Partition
- AWS::Region
- AWS::StackId
- AWS::StackName
- AWS::URLSuffix

Note

You can use intrinsic functions only in specific parts of a template. Currently, you can use intrinsic functions in resource properties, outputs, metadata attributes, and update policy attributes. You can also use intrinsic functions to conditionally create stack resources.

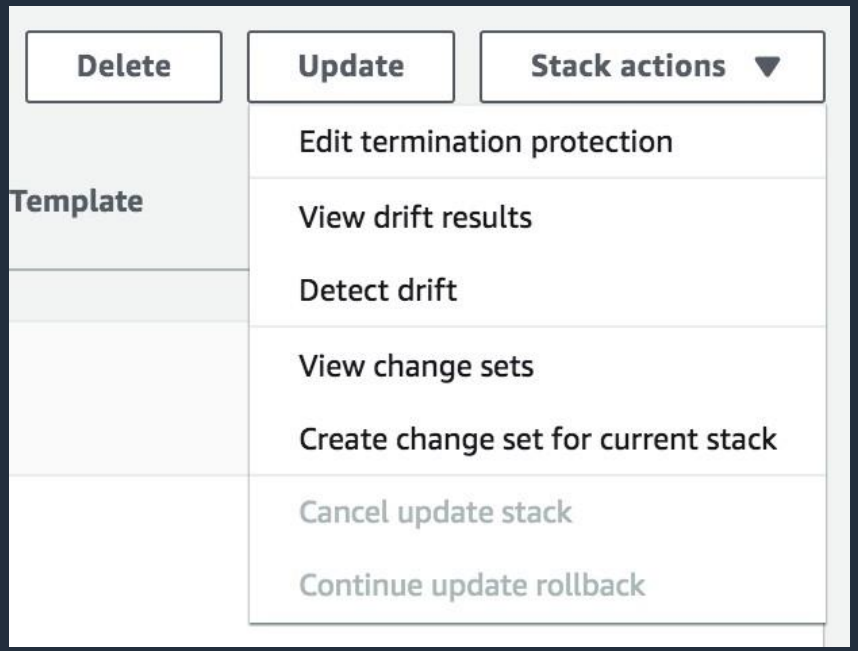
Template Anatomy - Outputs

- The optional Outputs section declares output values that you can import into other stacks to create cross-stack references
- Return in response (to describe stack calls), or view on the AWS CloudFormation console

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cfn-templates-us-east-1/S3_Bucket.template
      TimeoutInMinutes: "60"
Outputs:
  StackRef:
    Value: !Ref myStack
  OutputFromNestedStack:
    Value: !GetAtt myStack.Outputs.BucketName
```

AWS CloudFormation- Stack Updates

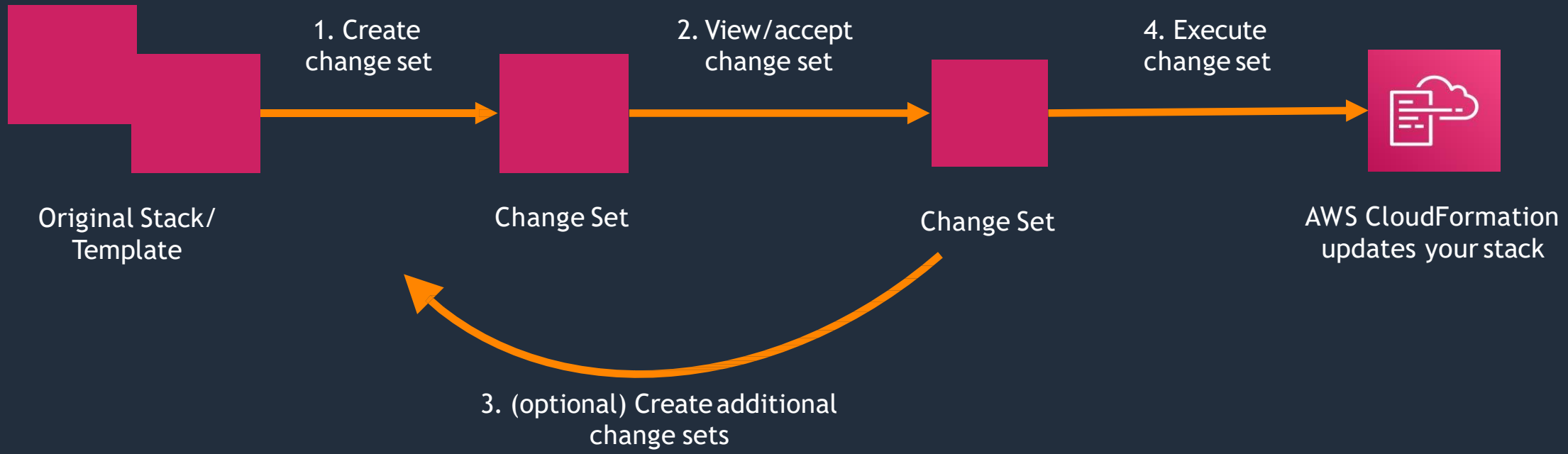
- Make changes to a stack's settings or change its resources by updating stack
- When you update a stack, you submit changes to AWS CloudFormation
- Two methods for updating stacks: *direct update* or *change sets* (you create and execute)




```
aws cloudformation update-stack --stack-name mystack --use-previous-template --notification-arns "arn:aws:sns:us-east-1:12345678912:mytopic" "arn:aws:sns:us-east-1:12345678912:mytopic2"
```

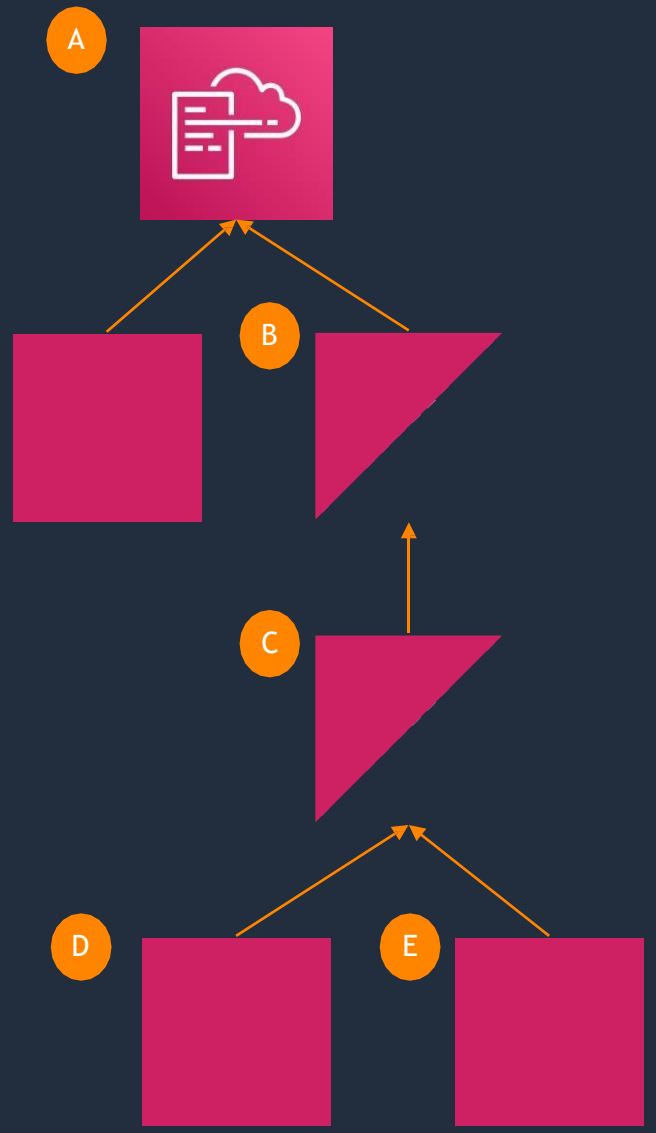
AWS CloudFormation – Change Sets

- Change sets enable you to preview how proposed changes to a stack might impact your running resources
- AWS CloudFormation makes the changes to your stack only when you decide to execute the change set



AWS CloudFormation – Nested Stacks

- Monolithic to Modular: common patterns can emerge in which you declare the same components in multiple templates
- Root (A) AWS CloudFormation is the root stack for all the other, nested, stacks in the hierarchy
- Nested stack templates must be placed in Amazon S3
- Broad permissions required tocreate a stack
- Blast radius – Takes one parent stack to destroy them all
- Using nested stacks to declare common components is considered a best practice



AWS CloudFormation – Nested Stacks

- You can use outputs from one stack in the nested stack group as inputs to another stack in the group. This differs from exporting values
- Outputs values of the child stack can be referenced by the parent stack or other nested stacks

```
AWSTemplateFormatVersion: "2010-09-09"
Resources:
  myStack:
    Type: AWS::CloudFormation::Stack
    Properties:
      TemplateURL: https://s3.amazonaws.com/cfn-templates-us-east-1/S3_Bucket.template
      TimeoutInMinutes: "60"
Outputs:
  StackRef:
    Value: !Ref myStack
  OutputFromNestedStack:
    Value: !GetAtt myStack.Outputs.BucketName
```

AWS CloudFormation - Cross-stack Reference (Layers)

- You can use outputs from one stack in the nested stack group as inputs to another stack in the group. This differs from exporting values
- Outputs values of the child stack can referenced by the parent stack or other nested stacks

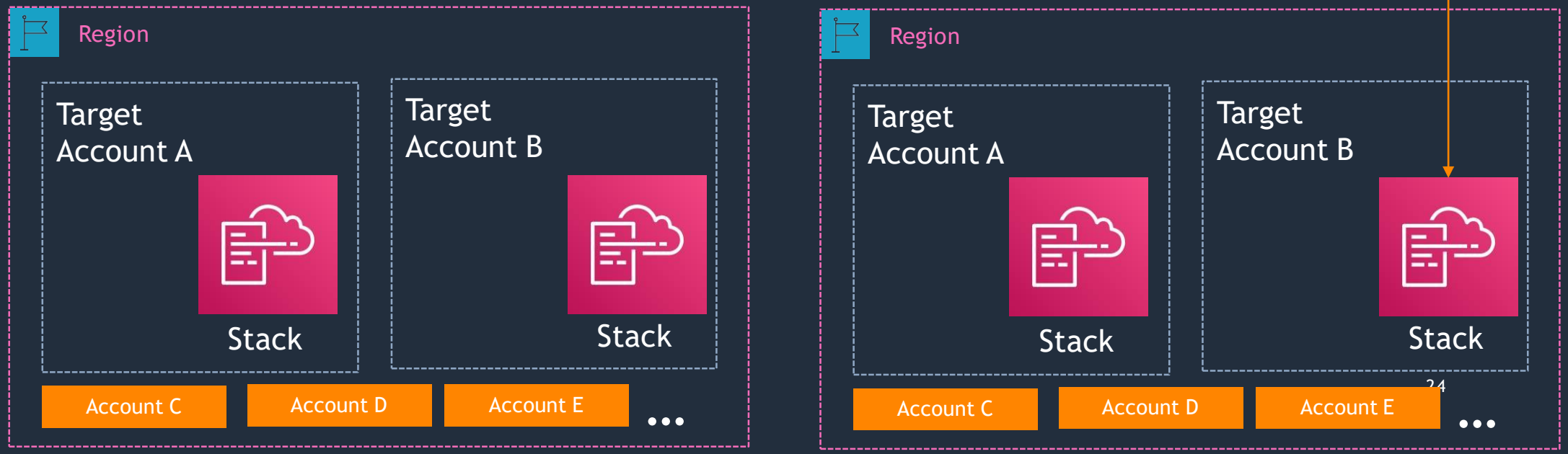
```
Outputs:
  PublicSubnet:
    Description: The subnet ID to use for public web servers
    Value:
      Ref: PublicSubnet
    Export:
      Name:
        Fn::Sub: "${AWS::StackName}-SubnetID"
```

```
Resources:
  ElasticLoadBalancer:
    Type: AWS::ElasticLoadBalancer
    Properties:
      Subnets:
      - Fn::ImportValue:
          Fn::Sub: "${NetworkStackName}-SubnetID"
      SecurityGroups:
      - Ref: ELBSecurityGroup
      CrossZone: 'true'
      ...
```

AWS CloudFormation - StackSets

Administration Account

AWS CloudFormation StackSets extend the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and regions with a single operation.



LAB Time

- **Lab for AWS CloudFormation**
  - WordPress Basic
  - WordPress Advance

Thank you!