AWS CloudFormation

Transparent and open

Don’t reinvent the wheel

Declarative and flexible

AWS CloudFormation

No extra charge

Customizable (parameters)

Integration ready

aws

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

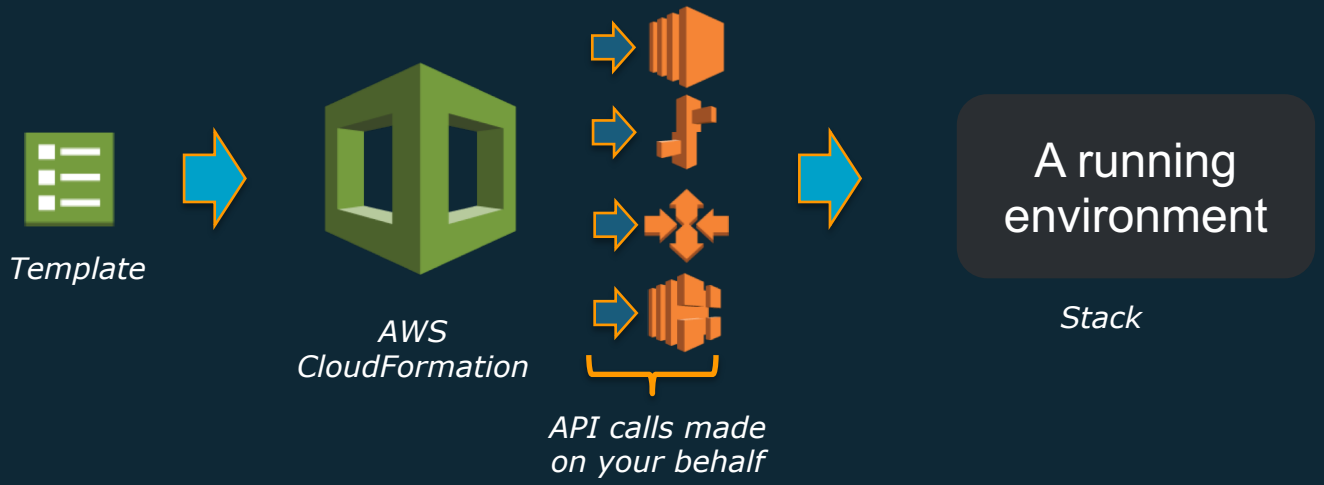
aws

Agenda

- AWS CloudFormation syntax: JSON vs. YAML
- Intrinsic and condition functions
- AWS CloudFormation Designer and AWS CloudFormer
- AWS CloudFormation Stack

aws

Overview



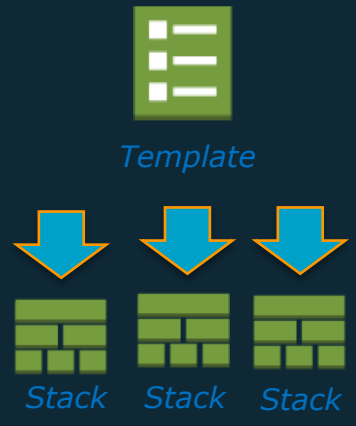
- JSON/YAML format template
- Presents template to AWS CloudFormation
- AWS CloudFormation translates it to an API request
- Forms a stack of resources

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

aws

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

AWS CloudFormation syntax

- JSON – JavaScript object notation
- Attribute-value pairs
- Similar to XML

```
{
"AWSTemplateFormatVersion" : "2010-09-09",
"Description" : "Create a Simple S3 bucket with parameter to choose own bucket name",
"Parameters": {
    "S3NameParam" : {
        "Type": "String",
        "Default" : "saurabh-dafaultbucket",
        "Description" : "Enter the Bucket Name",
        "MinLength" : "5",
        "MaxLength" : "30"
                }
            },

"Resources" : {
    "Bucket" : {
        "Type" : "AWS::S3::Bucket",
            "Properties" : {
                "AccessControl" : "PublicRead",
                "BucketName" : {"Ref" : "S3NameParam" },
                "Tags" : [ {"Key" : "Name" , "Value" : "MyBucket"} ]
                        }
                    }
        },

"Outputs" : {
    "BucketName" : {
        "Description" : "BucketName" ,
        "Value" : { "Ref" : "S3NameParam"}
                }
            }

}
```


aws

AWS CloudFormation syntax

- YAML – Not a markup language
- YAML is a human friendly data serialization standard
- Comments - Use #
- ~~‘{‘ and ‘;’~~

```yaml
Resources:
  DB:
    Type: "AWS::RDS::DBInstance"
    Properties:
      AllocatedStorage: 5
      StorageType: gp2
      DBInstanceClass: db.t2.micro
      DBName: wordpress
      Engine: MySQL
      MasterUsername: wordpress
      MasterUserPassword: w0rdpr355
  EC2:
    Type: AWS::EC2::Instance
    Properties:
      ImageId: ami-c481fad3 # N.Virginia - Ama Sept'16
      InstanceType: t2.micro
  S3:
    Type: "AWS::S3::Bucket"
    Properties:
      BucketName: wp-xxxxxx # replace xxxxxx with random
```

High-level template structure

```
{
    "Description" : "A text description for the template usage",
    "Parameters": {
        // A set of inputs used to customize the template per deployment
    },
    "Resources" : {
        // The set of AWS resources and relationships between them
    },
    "Outputs" : {
        // A set of values to be made visible to the stack creator
    },
    "AWSTemplateFormatVersion" : "2010-09-09”
}
```

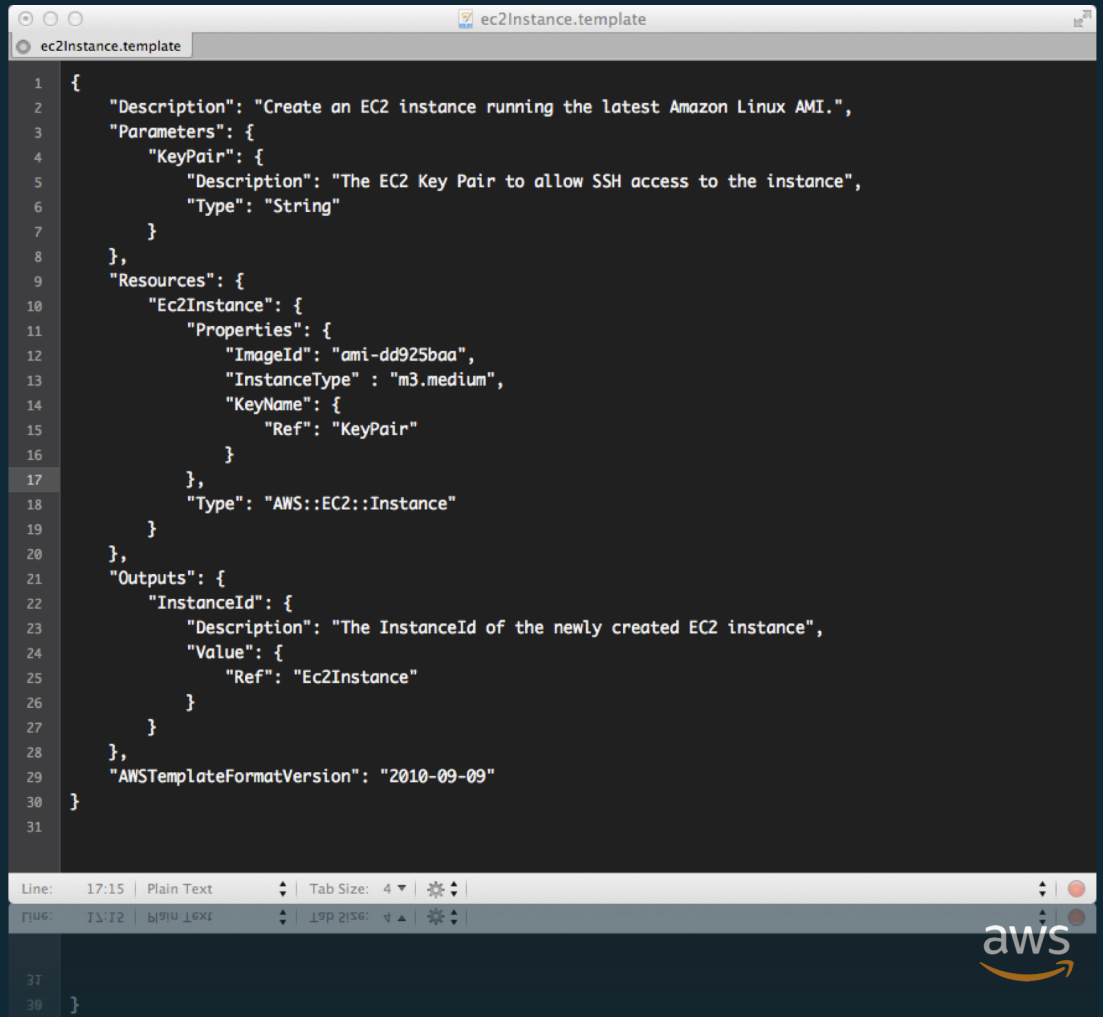

aws

Stack creation

You use a template to create and manage a stack

A stack is a collection of AWS resources that you can manage as a single unit

AWS CloudFormation ensures all stack resources are created or deleted as appropriate



Simple template – Create EC2 instance

```
{
    "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
    "Parameters": {
        "KeyPair": {
            "Description": "The EC2 Key Pair to allow SSH access to the instance",
            "Type": "String"
        }
    },
    "Resources": {
        "Ec2Instance": {
            "Properties": {
                "ImageId": "ami-9d23aeea",
                "InstanceType" : "m3.medium",
                "KeyName": {
                    "Ref": "KeyPair"
                }
            },
            "Type": "AWS::EC2::Instance"
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The InstanceId of the newly created EC2 instance",
            "Value": {
                "Ref": "Ec2Instance"
            }
        }
    },
       "AWSTemplateFormatVersion": "2010-09-09"
}
```

You enter values for these parameters when you create your stack

Simple template – Create EC2 instance

```
{
    "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
    "Parameters": {
        "KeyPair": {
            "Description": "The EC2 Key Pair to allow SSH access to the instance",
            "Type": "String"
        }
    },
    "Resources": {
        "Ec2Instance": {
            "Properties": {
                "ImageId": "ami-9d23aeea",
                "InstanceType" : "m3.medium",
                "KeyName": {
                    "Ref": "KeyPair"
                }
            },
            "Type": "AWS::EC2::Instance"
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The InstanceId of the newly created EC2 instance",
            "Value": {
                "Ref": "Ec2Instance"
            }
        }
    },
       "AWSTemplateFormatVersion": "2010-09-09"
}
```

Includes statically defined properties (ImageId and InstanceType) and a reference to the KeyPair parameter. ImageId is the AMI specific to the region that you want to launch this stack in (eu-west-1 region in this example)

Simple template – Create EC2 instance

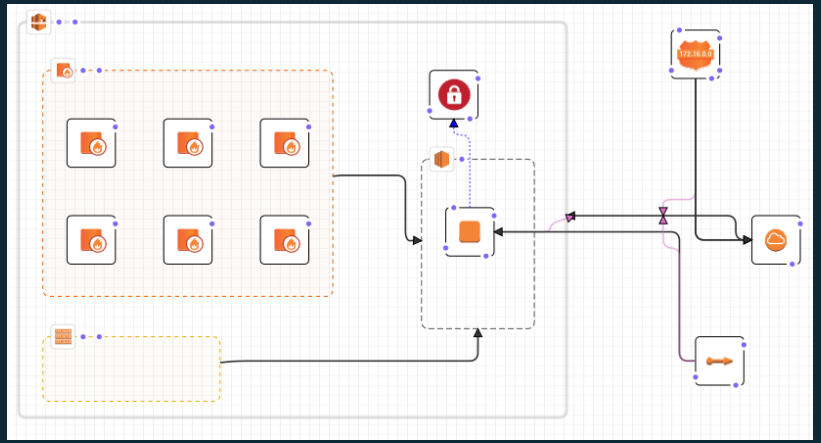
```
{
    "Description": "Create an EC2 instance running the latest Amazon Linux AMI.",
    "Parameters": {
        "KeyPair": {
            "Description": "The EC2 Key Pair to allow SSH access to the instance",
            "Type": "String"
        }
    },
    "Resources": {
        "Ec2Instance": {
            "Properties": {
                "ImageId": "ami-9d23aeea",
                "InstanceType" : "m3.medium",
                "KeyName": {
                    "Ref": "KeyPair"
                }
            },
            "Type": "AWS::EC2::Instance"
        }
    },
    "Outputs": {
        "InstanceId": {
            "Description": "The InstanceId of the newly created EC2 instance",
            "Value": {
                "Ref": "Ec2Instance"
            }
        }
    },
    "AWSTemplateFormatVersion": "2010-09-09"
}
```

These outputs are returned after the template has completed execution

aws

Create Stack Using AWS CloudFormation Designer

AWS CloudFormation Designer (Designer) is a graphic tool for creating, viewing, and modifying AWS CloudFormation templates. With Designer, you can diagram your template resources using a drag-and-drop interface, and then edit their details using the integrated JSON and YAML editor.



Templates – Create and manage a stack using the AWS CLI

Install the AWS CLI using installation guide

```
aws cloudformation create-stack
    --stack-name ec2InstanceCmdLineDemo
    --template-url https://s3.amazonaws.com/cf-templates-deloitte-
    workshop/Demo-1.json
    --parameters ParameterKey=KeyPair,ParameterValue=KeyName
```

Returns the details of the created stack, in the output format of your choice

```
arn:aws:cloudformation:us-east-1:496891363831:stack/t1/625f07c0-1fef-11e8-
a501-50d5ca63261e
```

Template anatomy

1. Format version
2. *Transform (new)*
3. Description
4. Metadata
5. Parameters
6. Mappings
7. Conditions
8. Resources* (required)
9. Outputs

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

aws

Template anatomy – Resources

- Only section that is **not optional**
- Define AWS resources to create/update
- Supports 164 resource types (and growing)
  - Refer to the CloudFormation User Guide for updated list

```
"Resources":{
  "Ec2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
      "KeyName" : { "Ref" : "KeyName" },
      "NetworkInterfaces": [ {
        "AssociatePublicIpAddress": "true",
        "DeviceIndex": "0",
        "GroupSet": [{ "Ref" : "myVPCEC2SecurityGroup" }],
        "SubnetId": { "Ref" : "PublicSubnet" }
      } ]
    }
  }
}
```

Template anatomy – Format version and description

Format version

- Currently only supports 1 value “2010-09-09”

Description

- JSON string where you provide a Description (optional)

```
{
  "AWSTemplateFormatVersion" : "2010-09-09"
  "Description" : "Here are some details about the template."
  .....
}
```


aws

Template Anatomy – Metadata

Arbitrary JSON objects that provide additional details about the template.

```
{
  .....
  "Metadata": {
    "Instances": {
      "Description": "Information about the instances"
    },
    "Databases": {
      "Description": "Information about the databases"
    }
  }
  .....
}
```


aws

Template anatomy – Parameters

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

```
{
  "Parameters" : {
    "InstanceTypeParameter" : {
      "Type" : "String",
      "Default" : "t1.micro",
      "AllowedValues" : ["t1.micro", "m1.small", "m1.large"],
      "Description" : "Enter t1.micro, m1.small, or m1.large
    }
  },
  ......
}
```

```
......
"Ec2Instance" : {
"Type" : "AWS::EC2::Instance",
"Properties" : {
  "InstanceType" : { "Ref" : "InstanceTypeParameter" },
  "ImageId" : "ami-2f726546"
  }
}
```

Template anatomy – AWS-specific parameters

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

```
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
List<AWS::EC2::Subnet::Id>
```


aws

Intrinsic functions and pseudo parameters

Intrinsic functions

Fn::Base64

Fn::FindInMap

Fn::GetAtt

Fn::GetAZs

Fn::Join

Fn::Select

Fn::Sub

Ref

Pseudo parameters

AWS::NotificationARNs

AWS::Region

AWS::StackId

AWS::StackName

aws

Template anatomy – Intrinsic functions

- `Fn::Select` returns a single object from a list of objects by index.
- `Fn::Join` appends a set of values into a single value, separated by the specified delimiter (preferred for large blocks of text like UserData).
- `Ref` returns the value of the specified parameter or resource.

```
                },
"MyCurrentAZ" : {
        "Description" : "My Current AZ",
        "Value" : {
                    "Fn::Select" : ["0", {"Fn::GetAZs" : {"Ref" : "AWS::Region"}}]
                   }
                },
"NextAZ" : {
        "Description" : "Next Available AZ" ,
        "Value" : {
                    "Fn::Select" : ["1", {"Fn::GetAZs" : {"Ref" : "AWS::Region"}}]
                   }
        },
 "JoinAZs" : {
           "Description" : "Join Value of Two AZs by colon" ,
           "Value" : {
                       "Fn::Join" : [":", ["MyCurrentAZ","NextAZ"]]
                      }
            }
}
```


aws

Template anatomy – Intrinsic functions

- `Fn::GetAtt` returns the value of an attribute from a resource in the template.
- `Fn::Base64` returns the Base64 representation of the input string. Typically used to pass encoded data to Amazon EC2 instances using the UserData property.

```
"Resources" : {
    "MyEC2Ins" : {
         "Type" : "AWS::EC2::Instance" ,
         "Properties" : {
         "ImageId" : {"Fn::FindInMap" : ["AMIRegionMap" , {"Ref" : "AWS::Region"} , "AMI"] },
         "InstanceType" : "t2.micro",
         "UserData" : {"Fn::Base64" : "yum upgrade -y"}
                    }
               }
          },
"Outputs" : {
   "EC2PrivateIP" : {
            "Description" : "EC2 Instance Private IP Address",
            "Value" : {
                       "Fn::GetAtt" : ["MyEC2Ins" , "PrivateIp" ]
                      }
                    },
```


aws

Template anatomy - Mappings

- Reference table, matches a key to a corresponding set of named values.
- Use `Fn::FindInMap` intrinsic function to retrieve values in a map.

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

  "Mappings" : {
    "RegionMap" : {
      "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
      "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
      "eu-west-1" : { "32" : "ami-37c2
      "ap-southeast-1" : { "32" : "ami
      "ap-northeast-1" : { "32" : "ami
    }
  },
```

```
  "Resources" : {
    "myEC2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "32"]},
        "InstanceType" : "m1.small"
      }
    }
  }
}
```

Template anatomy – Conditions

- Resource creation can depend on logical conditions.
- Used in conjunctions with Intrinsic Functions:
  `Fn::If`, `Fn::Equals`, and `Fn::Not`

```
{
  ....
  "Conditions" : {
    "CreateProdResources" : {"Fn::Equals" : [{"Ref
  },

  "Resources" : {
    .....
    "MountPoint" : {
      "Type" : "AWS::EC2::VolumeAttachment",
      "Condition" : "CreateProdResources",
      "Properties" : {
        "InstanceId" : { "Ref" : "EC2Instance" },
        "VolumeId"  : { "Ref" : "NewVolume" },
        "Device" : "/dev/sdh"
      }
    },
    ......
  }
}
```

```
{
  ....
    "Outputs" : {
      "ProdEnvironment" : {
        "Description" : "Environment Type",
        "Value" : {
          "Fn::If" : [
            "CreateProdResources",
            "Production Environment",
            "Development Environment"
          ]
        }
      }
    }
}
```


aws

Template anatomy - Outputs

- Output values to view from the console
  OR
  Values returned from stack call
- Used with nested stack and cross stack references
  - Pass parameter values from one stack to another

```
{
  ....
  "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" }
    }
  }
}
```

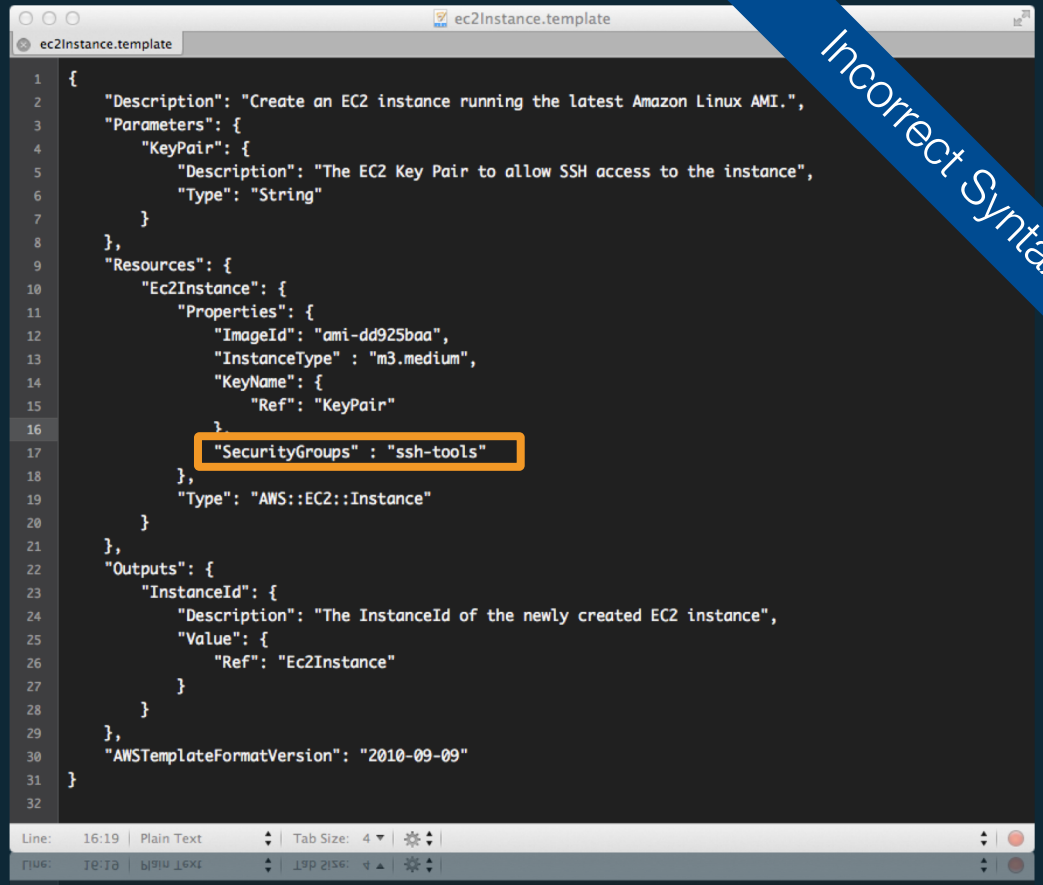

aws

Updating a stack

When you update a stack, you submit changes, such as new input parameter values or an updated template.

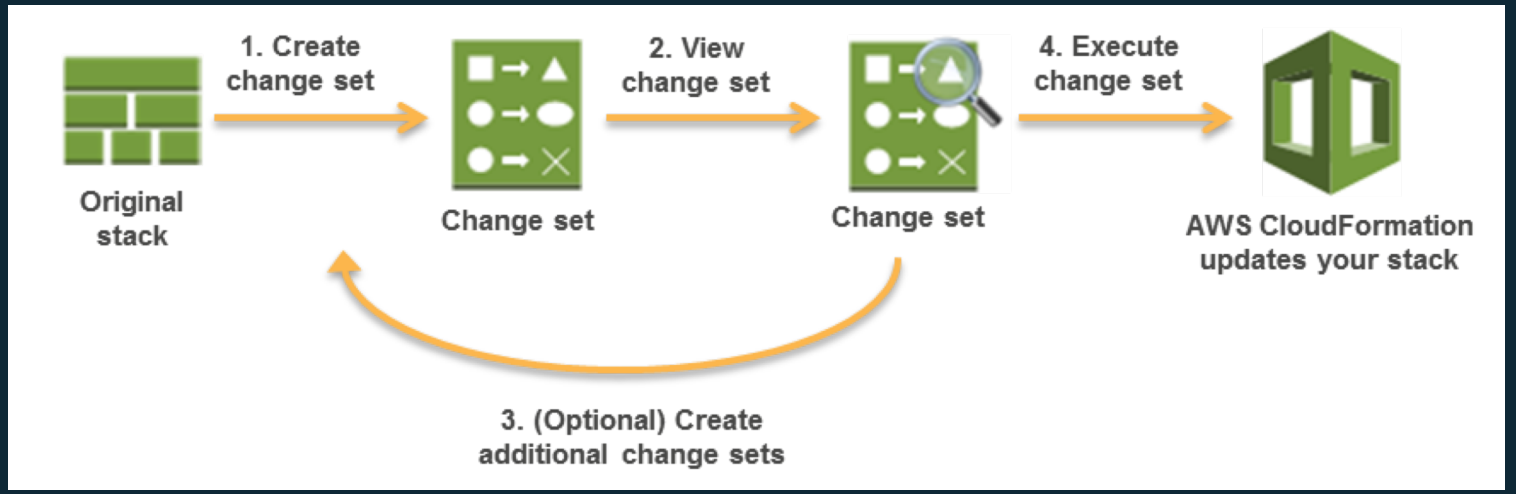
AWS CloudFormation compares the changes you submit with the current state of your stack and updates only the changed resources.

Two methods for updating stacks: *direct update* or *change sets* (you create and execute).



Stack change set

- 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



Nested stacks

- Breaks down a monolithic architecture into a modularized architecture
- AWS CloudFormation creates a stack from within another stack
- Templates must be placed in Amazon S3
- Outputs values of the child stack that are referenced by the parent stack
- Broad permissions required to create a stack
- Blast radius – Takes one parent stack to destroy them all

aws

Nested stacks

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "S3Bucket" : {
      "Type" : "AWS::S3::Bucket",
      "Properties" : {
        "AccessControl" : "PublicRead"
      }
    }
  },

  "Outputs" : {
    "BucketName" : {
      "Value" : { "Ref" : "S3Bucket" },
      "Description" : "Name of S3 bucket to hold website content"
    }
  }
}
```

```
{
    "AWSTemplateFormatVersion" : "2010-09
    "Resources" : {
        "myStack" : {
         "Type" : "AWS::CloudFormation::S
         "Properties" : {
            "TemplateURL" : "https://s3.a
             "TimeoutInMinutes" : "60"
         }
        }
    },
    "Outputs": {
      "StackRef": {"Value": { "Ref" : "myStack"}},
      "OutputFromNestedStack" : {
            "Value" : { "Fn::GetAtt" : [ "myStack", "Outputs.BucketName" ] }
      }
    }
}
```


aws

Cross-stack reference – Layers

- Export values from one stack and use them in another stack.
- Build infrastructure in layers without nesting them, and instead of declaring everything in one stack
- `Export` output field and the `Fn::ImportValue` intrinsic function
- Regionally scoped, visible in console.

```
"PublicSubnet1": {
  "Description": "Public Subnet 1",
  "Value": {
    "Ref": "PublicSubnet1"
  },
  "Export" : { "Name" : {"Fn::Sub": "${AWS::StackName}-PublicSubnet1" }}
},
"PublicSu
```

```
"ElasticLoadBalancer" : {
  "Type" : "AWS::ElasticLoadBalancing::LoadBalancer",
  "Properties" : {
    "Subnets" : [ { "Fn::ImportValue" :  {"Fn::Sub": "${NetworkStack}-PublicSubnet1" } },
    "SecurityGroups": [ { "Ref" : "ELBSecurityGroup"} ],
    "CrossZone" : "true",
    "LBCookieStickinessPolicy" : [ {
      "PolicyName" : "CookieBasedPolicy"
```

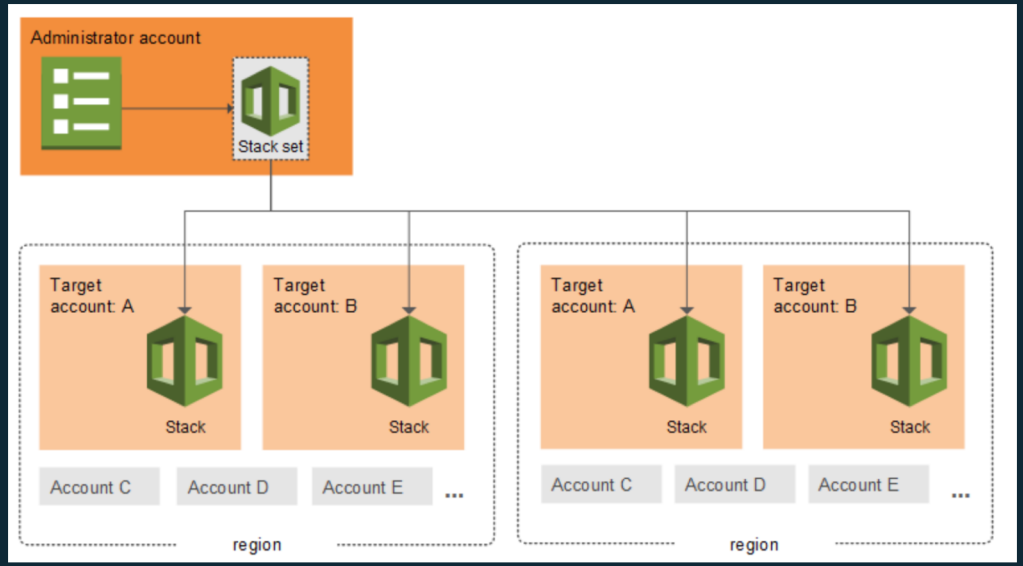

aws

AWS CloudFormer

CloudFormer is a template creation tool that creates an AWS CloudFormation template from existing AWS resources in your account. Its reverse engineering .

You select any supported AWS resources that are running in your account, and CloudFormer creates a template in an Amazon S3 bucket.

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



Resource: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

LAB