AWS London Loft: CloudFormation Workshop

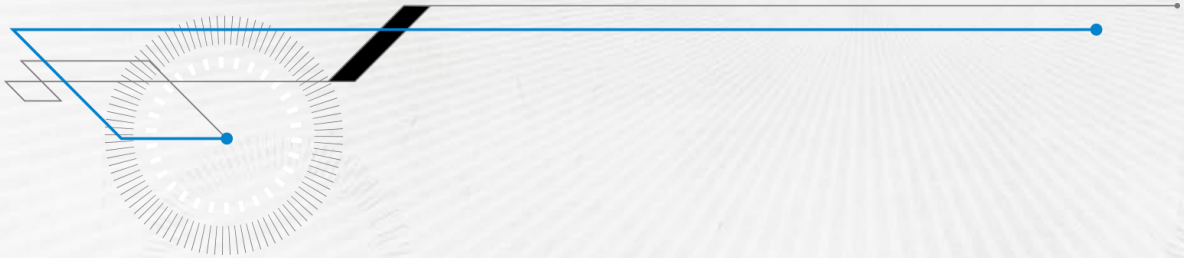
Templated AWS Resources

Tom Maddox
Solutions Architect
tmaddox@amazon.co.uk

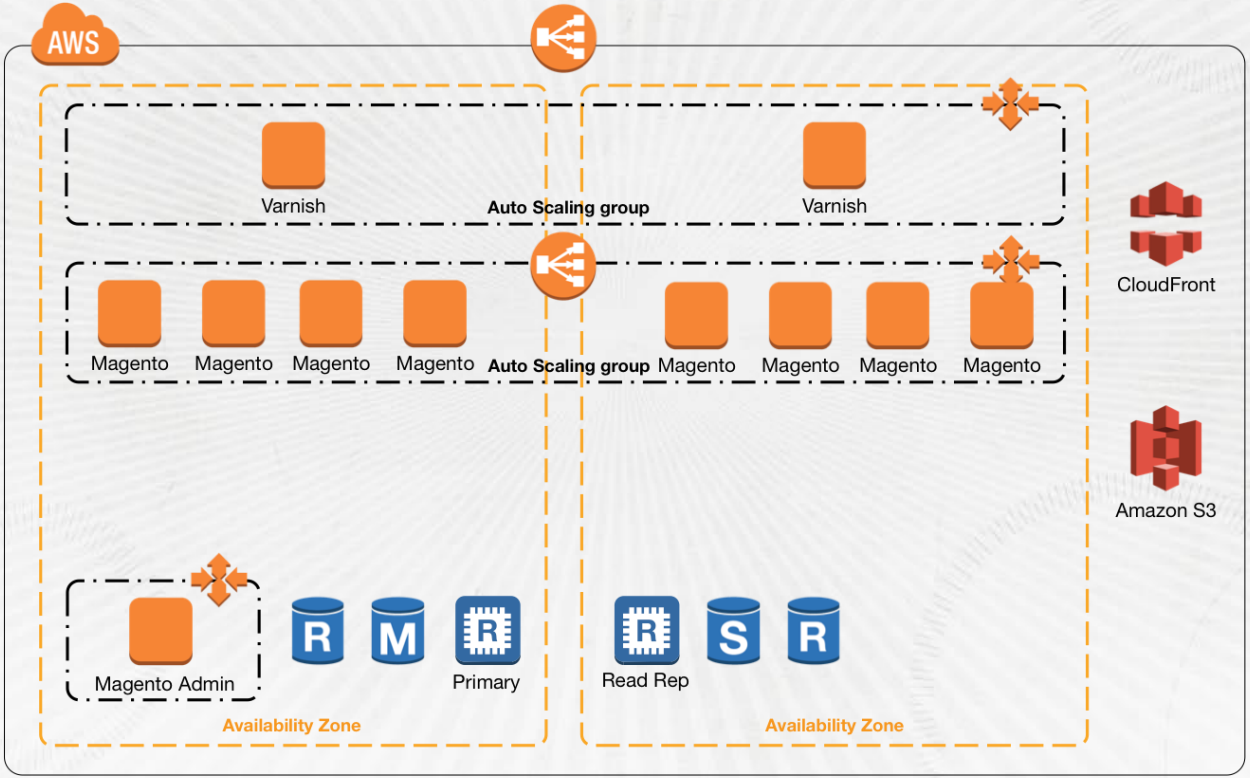
What will we cover?

- Introduction to CloudFormation
- Template anatomy
- Bootstrapping instances
- Managing stacks
- Best Practices
- Demo

Managing complex cloud architectures



Modern cloud architectures have lots of “bits”



Modern cloud architectures have lots of “bits”


EC2 instances


Security groups


IAM users, roles, and policies


VPCs


CloudFront distributions


Route 53 hosted zones


S3 buckets


CloudWatch alarms


RDS databases


Elastic IPs

Route tables


Auto Scaling groups

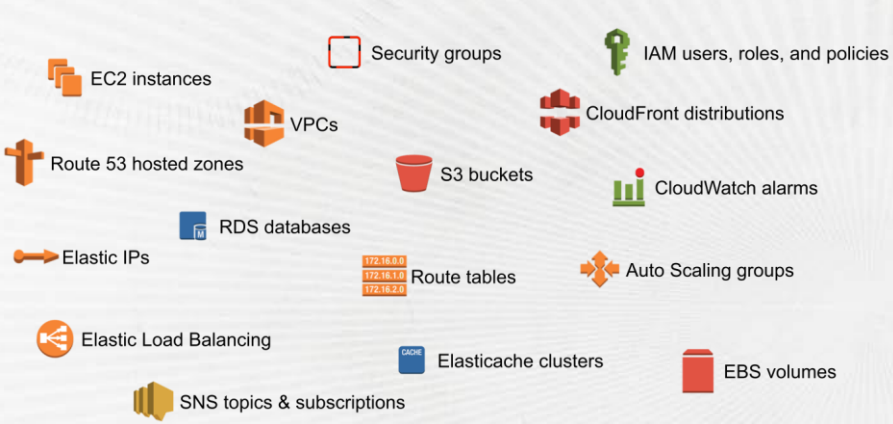

Elastic Load Balancing

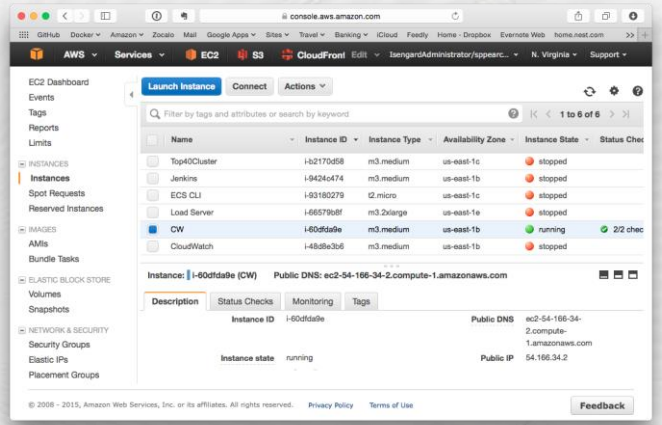
Elasticache clusters


EBS volumes


SNS topics & subscriptions

Use the console?





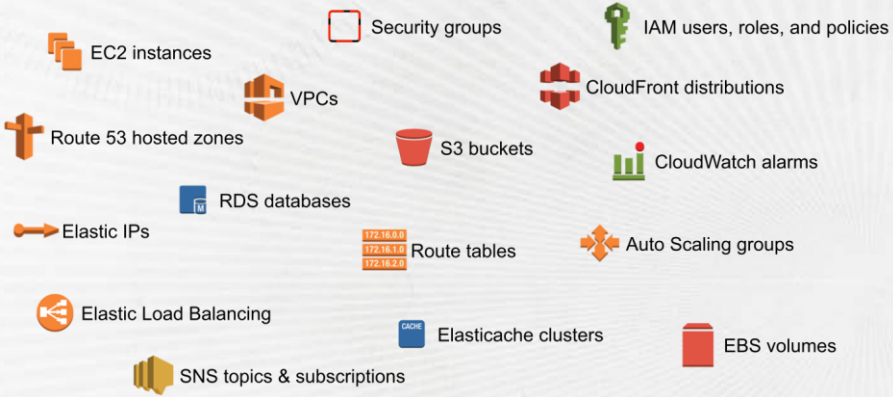
✗ Error prone

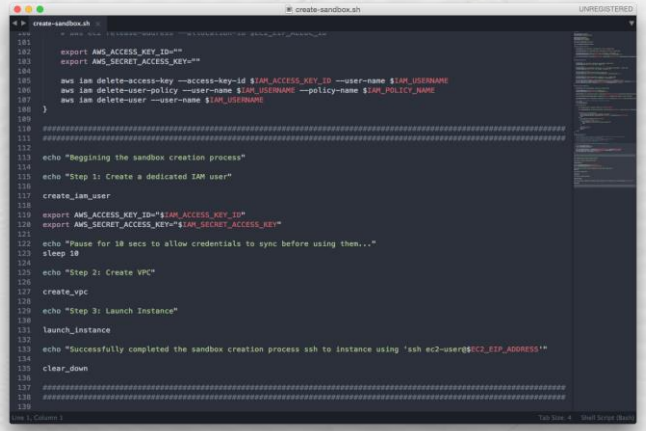
✗ Time consuming

✗ Not (easily) auditable

✗ Not repeatable

Write a script?





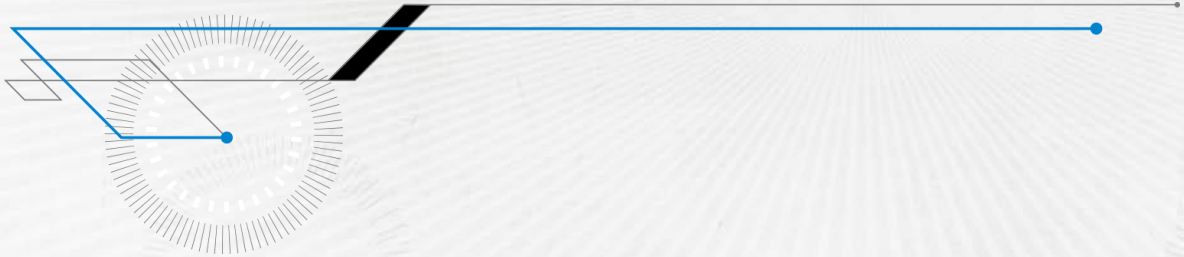
? What happens if an API call fails?

? How do I roll back?

? How can I update my environment?

? How long should I pause for?

Use CloudFormation?



AWS Application Management Services

Higher-level services

Do it yourself

Elastic Beanstalk

OpsWorks

CloudFormation

EC2

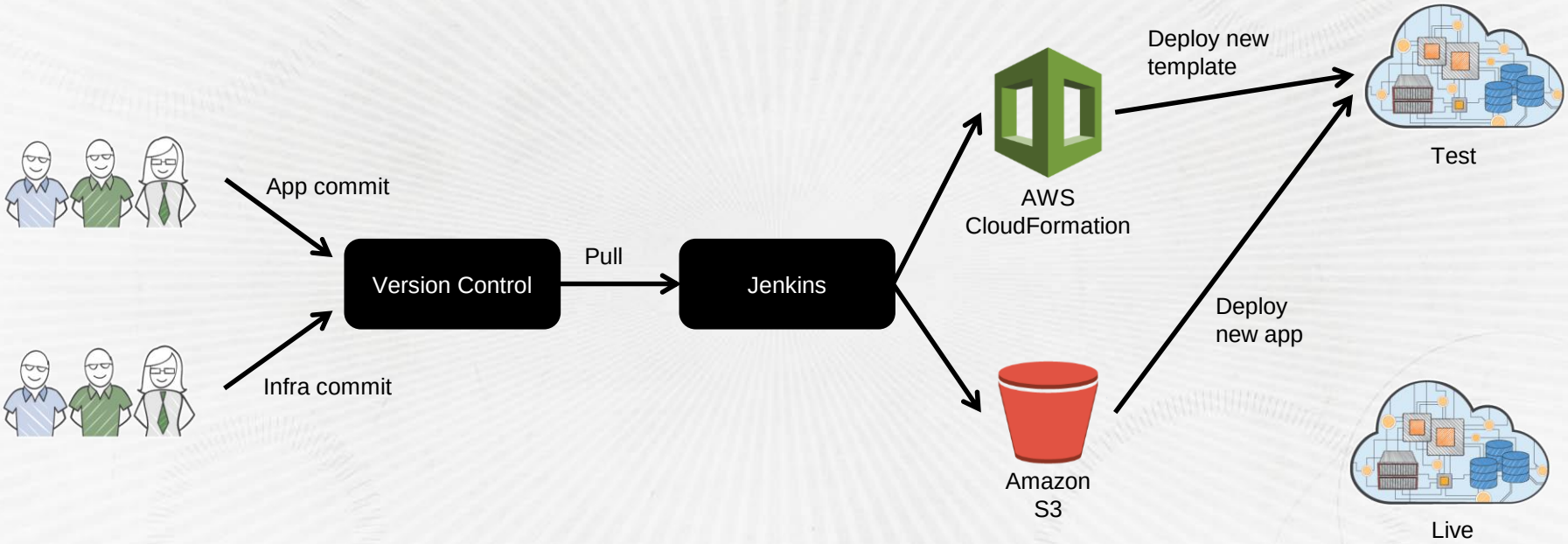
Convenience

Control

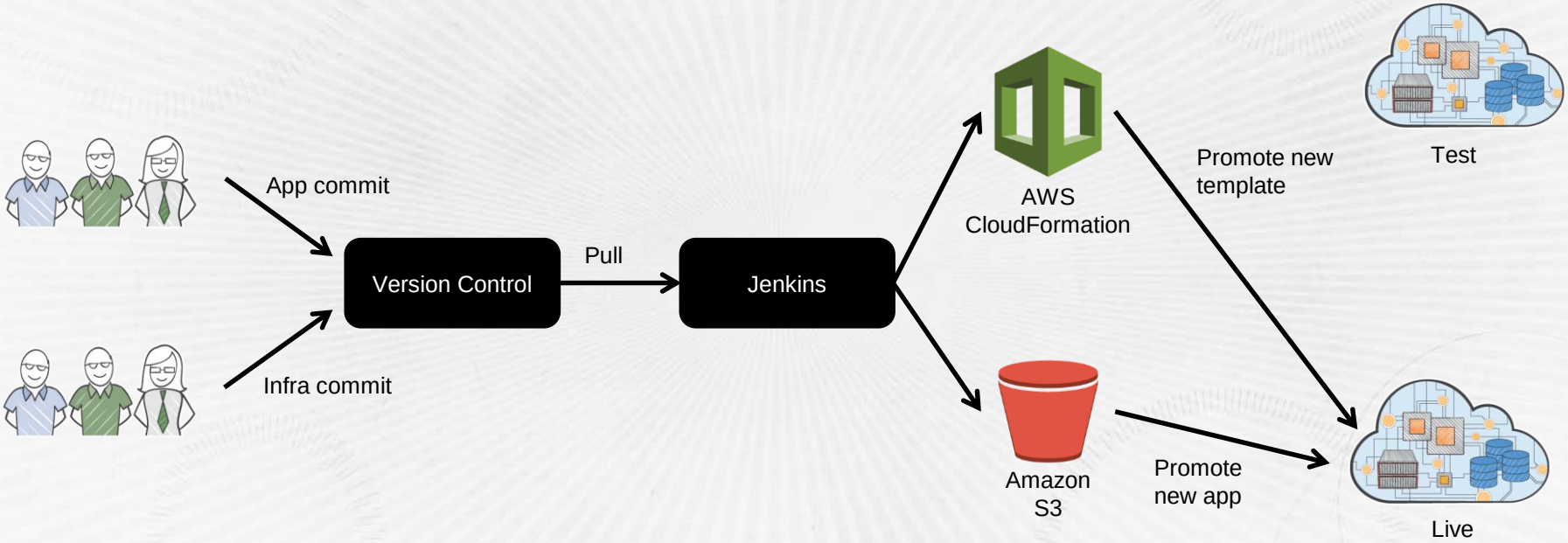
CloudFormation Overview

- Define & manage your infrastructure as code (JSON templates)
- Declarative approach with managed infrastructure dependencies
- Self documenting (no more spreadsheets of SG rules)
- Intelligent updating of resources and automated rollback capabilities
- AWS API interactions taken care for you
- Reproducibility
- Versioning

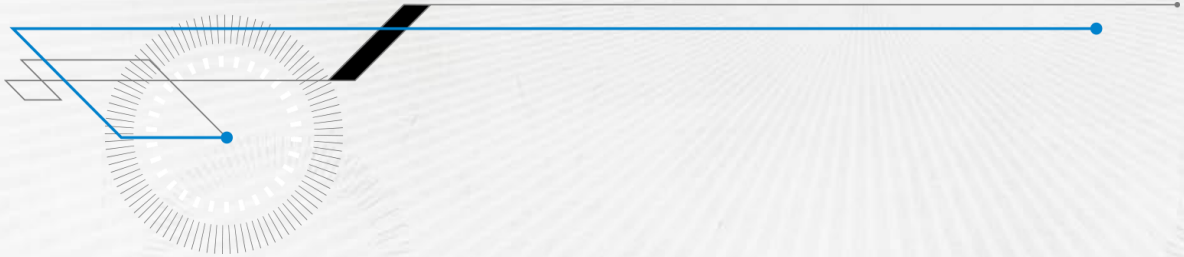
Continuous integration for your complete stack



Continuous integration for your complete stack



Template anatomy



Template structure

```
{
  "AWSTemplateFormatVersion" : "version date",

  "Description" : ”My lovely template",

  "Parameters" : {
    set of parameters
  },

  "Mappings" : {
    set of mappings
  },

  "Conditions" : {
    set of conditions
  },

  "Resources" : {
    set of resources
  },

  "Outputs" : {
    set of outputs
  }
}
```

Resources

- Describe detailed configuration of a resource in AWS
- Include, but not limited to:
  - IAM Policies, Users, Groups, Roles
  - VPCs, Subnets, NACLs, Security Groups
  - EC2 instances, AutoScaling Groups
  - RDS Databases, S3 Buckets
  - Elastic Load Balancers
  - CloudWatch Alarms
  - Lambda Functions
  - Logging (CloudTrail, CW Logs)

Resources (contd.)

- CloudFormation resources can automate other deployment services
  - OpsWorks: resource type “AWS::OpsWorks::Stack”

```
“AWS::OpsWorks::Stack”
```

  - Elastic Beanstalk

```
"AWS::ElasticBeanstalk::Environment"
```

  - CodeDeploy

```
“AWS::CodeDeploy::Application”
```

- What about things that CloudFormation doesn’t currently support?
  - Does it have an API?
    - Use Lambda: [create custom resources]

```
"AMIIDLookup": {
  "Type": "AWS::Lambda::Function",
  "Properties": {
    "Handler": "index.handler",
    "Role": { "Fn::GetAtt" : ["LambdaExecutionRole", "Arn"] },
    "Code": {
      "S3Bucket": "lambda-functions",
      "S3Key": "amilookup.zip"
    },
```

Nested Templates

Logical ID	Physical ID	Type	Status
stack1	arn:aws:cloudformation:us-east-1:979676883363:stack/GoldBase1-stack1-1IQK6Q0K6AZD5/8cc9fb90-78d6-11e5-ab62-5001ba48c2d2	AWS::CloudFormation::Stack	CREATE_COMPLETE
stack2	arn:aws:cloudformation:us-east-1:979676883363:stack/GoldBase1-stack2-32N9A77OO46U/8d192d00-78d6-11e5-a764-50e2416294a8	AWS::CloudFormation::Stack	CREATE_COMPLETE
stack3	arn:aws:cloudformation:us-east-1:979676883363:stack/GoldBase1-stack3-10QEXK61Z61LP/f46c9780-78d6-11e5-86e1-50e24162947c	AWS::CloudFormation::Stack	CREATE_COMPLETE
stack4	arn:aws:cloudformation:us-east-1:979676883363:stack/GoldBase1-stack4-1CIRM21C3IQ5G/2570cf40-78d7-11e5-abcb-507bb903ae0a	AWS::CloudFormation::Stack	CREATE_COMPLETE

- CloudFormation Stacks themselves can be resources

```
"AWS::CloudFormation::Stack"
```

- Useful for making reusable templates, segmenting resources, and avoiding template size limitations
- Launching a template with nested stacks will launch multiple sub-stacks
- Deleting the launching stack will, by default, delete all nested stacks

Parameters

- Used to pass in variables when launching a stack
- Use the “Ref” function to reference these variables in the Resources section of the template

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

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

Mappings

- Provides a set of custom named-value pairs
- Use for setting values based on different possible conditions (most notably, regions)
- Commonly used for mapping different AMI IDs to make template reusable across multiple AWS regions
- Use `Fn::FindInMap` when referencing in resources

```
"Mappings" : {
  "RegionMap" : {
    "us-east-1" : {
        "AMI" : "ami-76f0061f"
    },
    "us-west-1" : {
        "AMI" : "ami-655a0a20"
    },
    "eu-west-1" : {
        "AMI" : "ami-7fd4e10b"
    },
    "ap-southeast-1" : {
        "AMI" : "ami-72621c20"
    },
    "ap-northeast-1" : {
        "AMI" : "ami-8e08a38f"
    }
  }
},
```

```
"ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]}
```

Outputs

```
{
  "Description" : "Create an EC2 instance.”,
  "Parameters" : {
    “InstanceType” : {
      “Description” : “The EC2 Instance Type to launch.”,
      “Type” : “String”,
      “AllowedValues” : [“t1.micro”, “m1.small”, “m1.medium”]
    }
  },
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      ”Properties" : {
        "KeyName" : “my-key-pair”,
        "ImageId" : "ami-75g0061f”,
        “InstanceType” : { “Ref” : “InstanceType” }
      }
    }
  },
  "Outputs" : {
    "InstancePublicDnsName" : {
      "Description" : "The public DNS name of the newly created EC2 instance",
      "Value" : { ”Fn::GetAtt" : [ "Ec2Instance”, “PublicDnsName” ] }
    }
  }
}
```

Conditions

- Allow you to determine if a resource gets created or a property is defined
- The “Condition” attribute applied to any resource to specify a condition defined in the “Conditions” section of the template
- Condition must evaluate to true, otherwise the resource will not get created

```
"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"]}
},
```

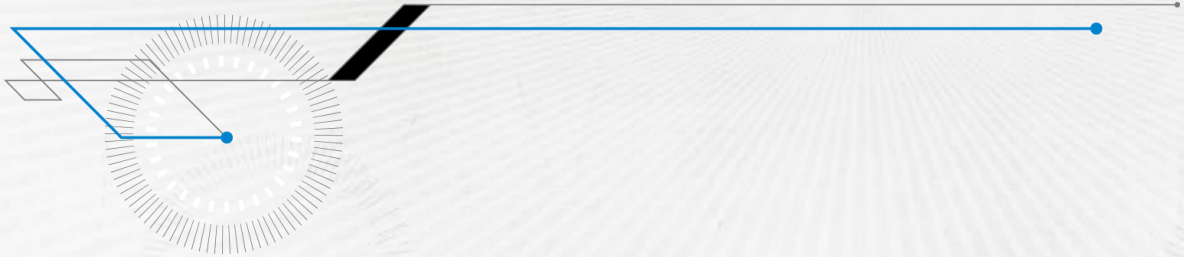
```
"MountPoint" :
 {
      "Type" : "AWS::EC2::VolumeAttachment",
      "Condition" : "CreateProdResources",
      "Properties" : {
              "InstanceId" : { "Ref" : "EC2Instance" },
              "VolumeId" : { "Ref" : "NewVolume" },
              "Device" : "/dev/sdh"
       }
 }
```

Helpful Links

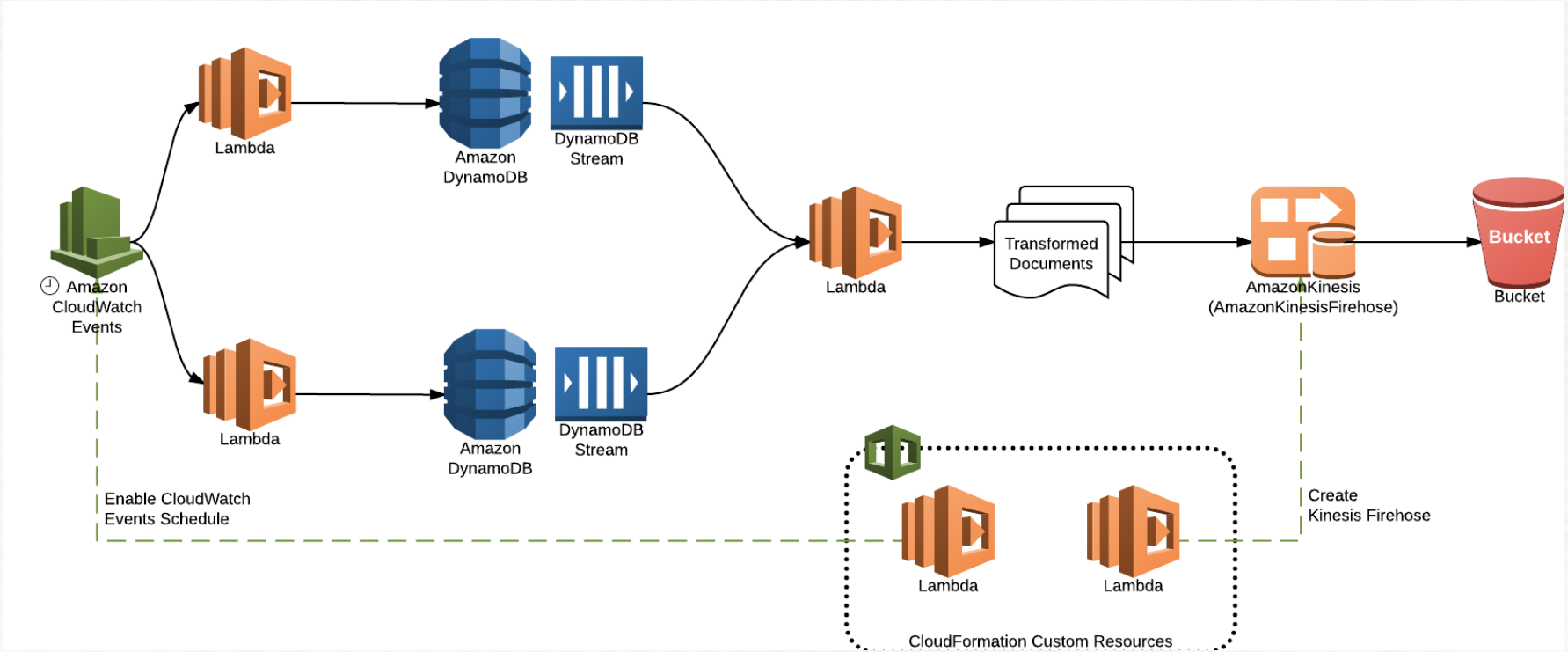
- http://aws.amazon.com/cloudformation/aws-cloudformation-templates/
  - Search: “AWS CloudFormation Templates”
  - Helpful solution-based templates (eg: Active Directory domain forest, or LAMP stack

- http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/CHAP_TemplateQuickRef.html
  - Search: “AWS CloudFormation Snippets”
  - Handy snippets for common AWS services (eg: Autoscaling group)

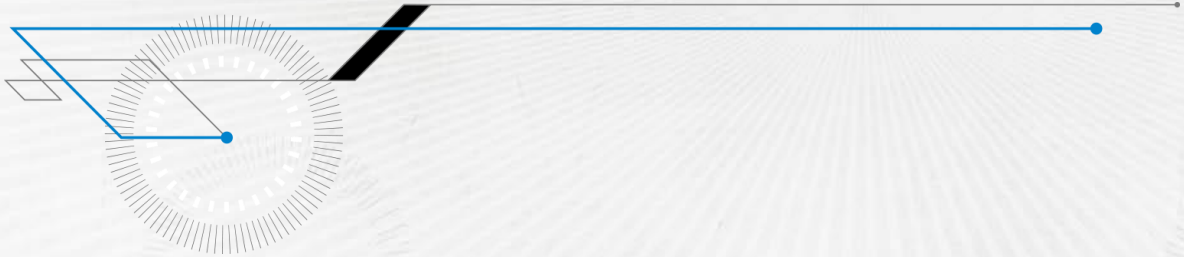
Demo!



Demo Architecture Diagram



Bootstrapping instances



Use AWS::CloudFormation::Init

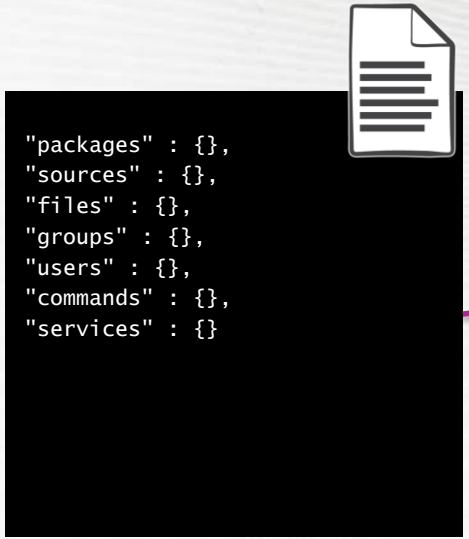
- **Declarative –** Specify the end state, not how to get there
- **Debugable –** cfn-init.log can integrate into CloudWatch Logs
- **Secure –** Supports IAM roles
- **Updatable** - Supported as part of the stack update process
- **Bring your own tools** – Use cfn-init to bootstrap Chef, Puppet etc.

Use AWS::CloudFormation::Init

```
{
  "Description" : "Create an EC2 instance.”,
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "AWS::CloudFormation::Init" : {
          "webapp-config": {
            "packages" : {},
            "sources" : {},
            "files" : {},
            "groups" : {},
            "users" : {},
            "commands" : {},
            "services" : {}
          }
        }
      },
      ”Properties" : {
        "KeyName" : “my-key-pair”,
        "ImageId" : "ami-66f28c34",
        "InstanceType" : “m3.medium”
      }
    }
  }
}
```

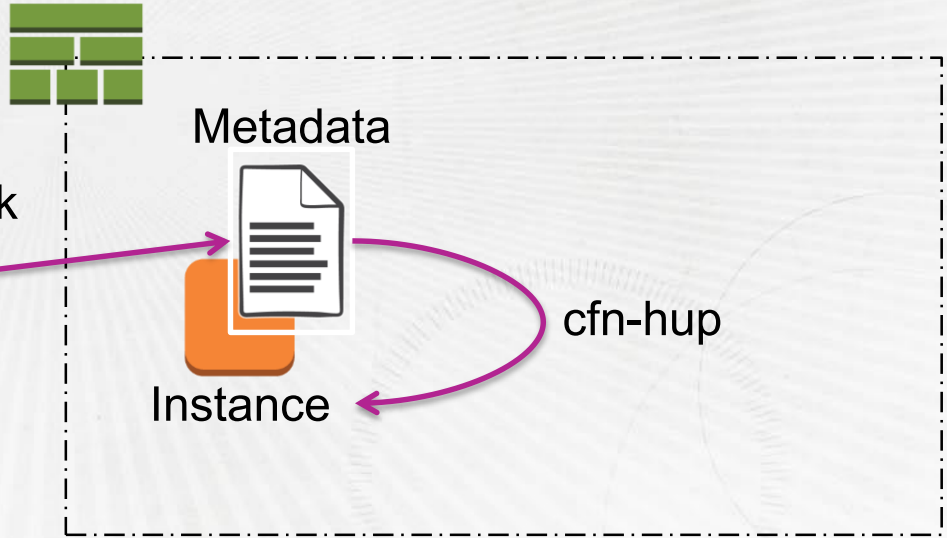
Using CloudFormation::Init to update instances

1. Update instance metadata in the template

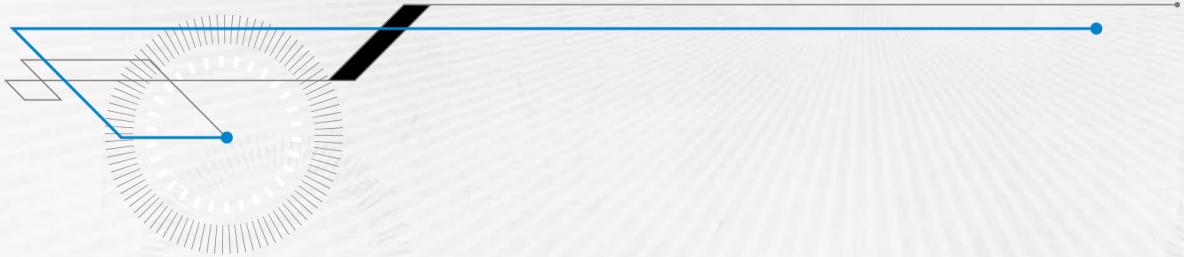


2. UpdateStack

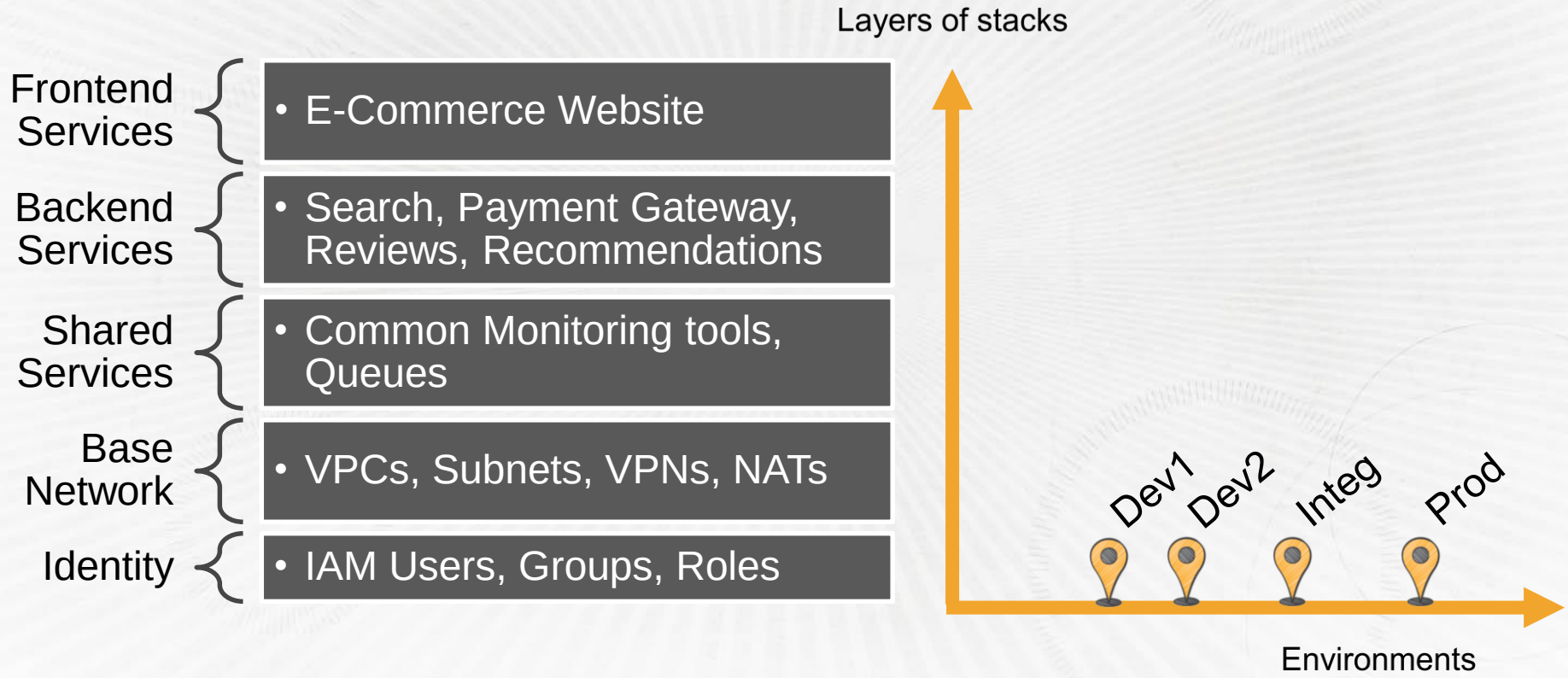
3. AWS CloudFormation daemon updates configuration



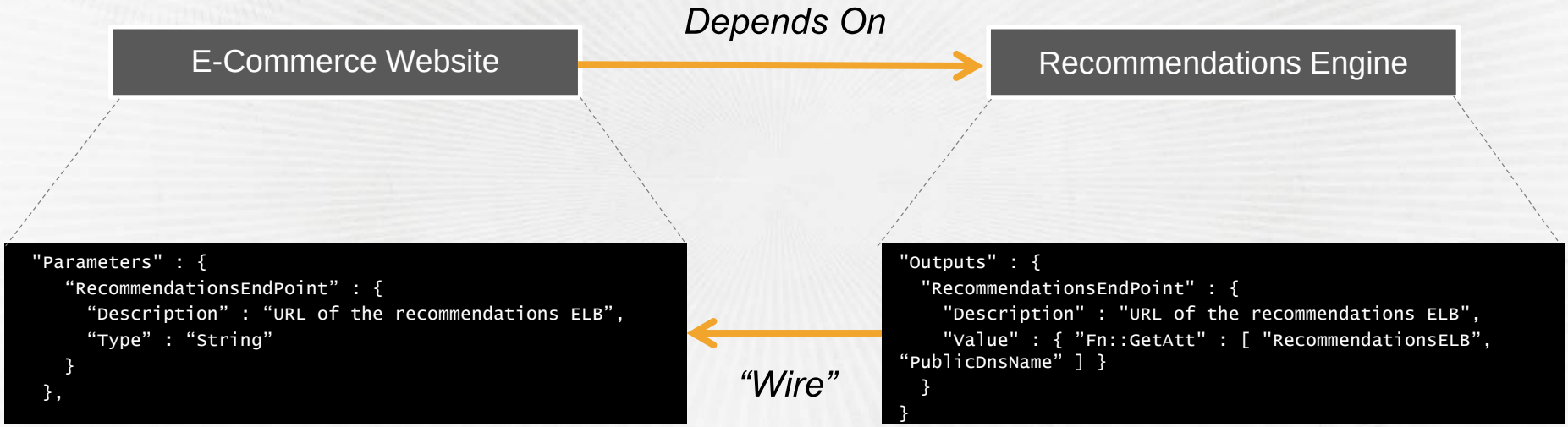
Managing your stacks



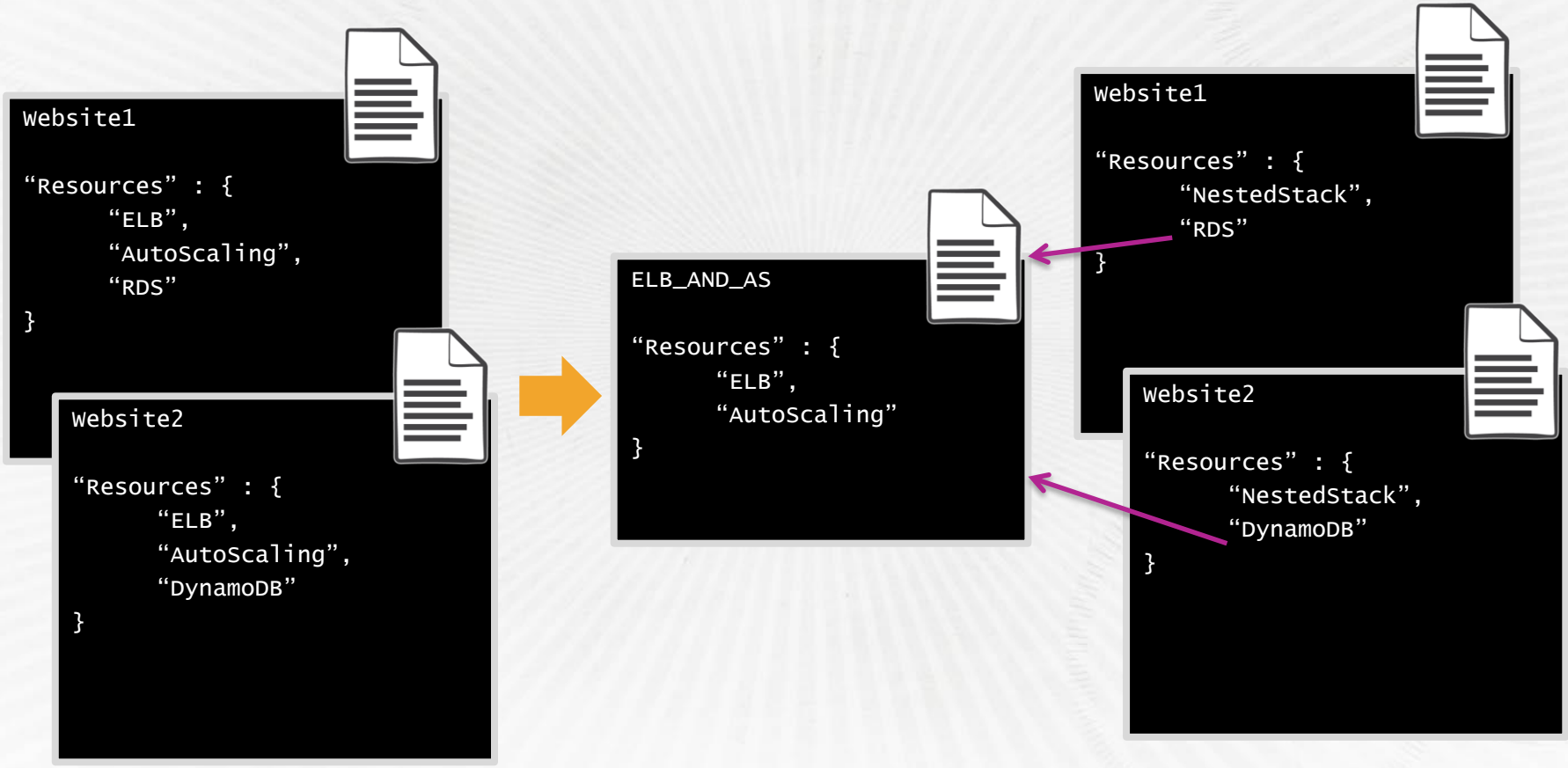
Organize by layers & environments



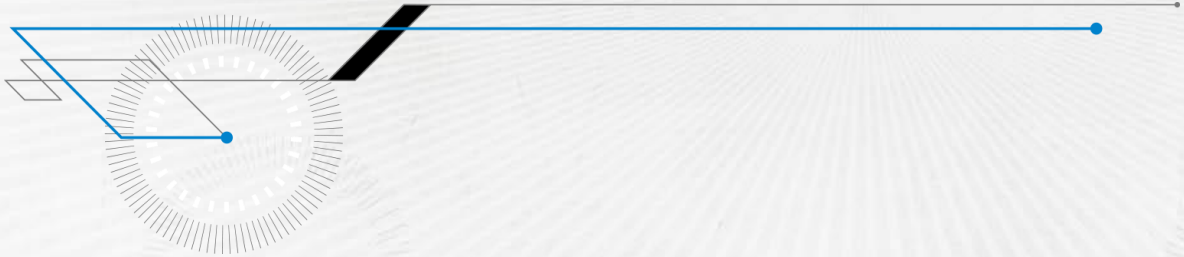
Apply SOA principles



Nested stacks for reusability



Deploying Applications



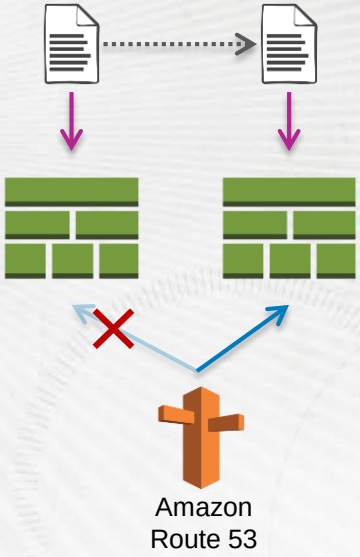
Choose a deployment style

In-place

Templates

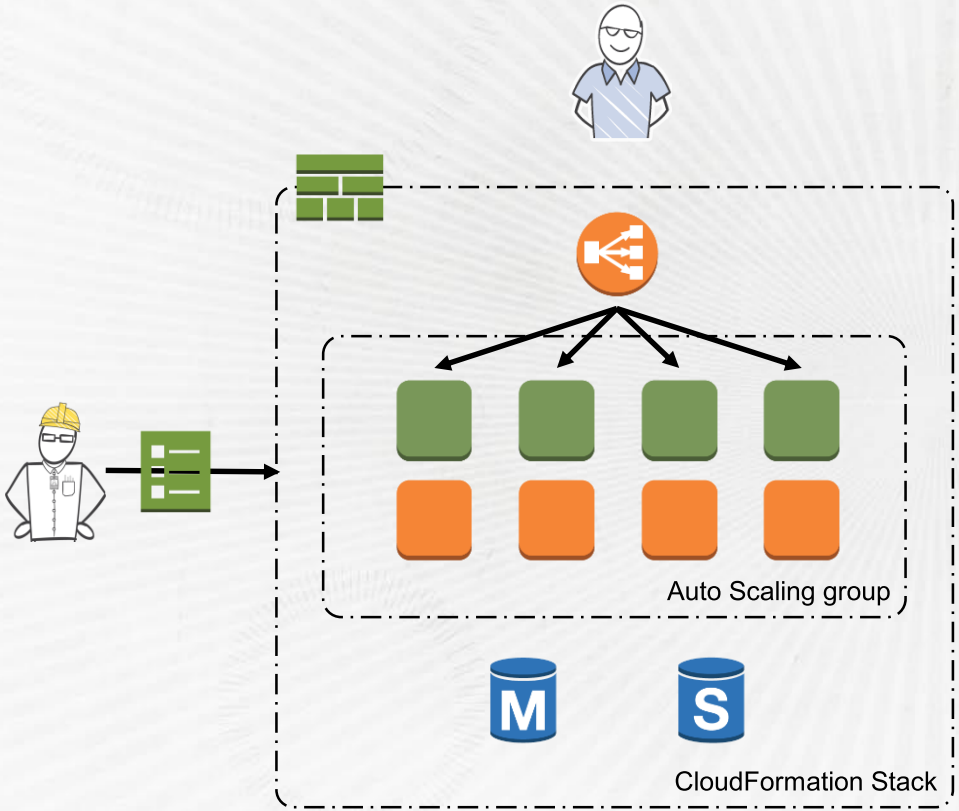
Stacks

Blue-Green



Use AutoScalingRollingUpdate for zero downtime

```
"WebServerGroup" : {
  "Type" : "AWS::AutoScaling::AutoScalingGroup",
  "Properties" : {
    "LaunchConfigurationName" : { "Ref" : "LaunchConfig" },
    ...
  },
  "UpdatePolicy" : {
    "AutoScalingRollingUpdate" : {
      "MaxBatchSize" : “2”,
      "MinInstancesInService" : ”2”,
      "PauseTime" : “PT20M”
    }
  }
}
```



Use AutoScalingRollingUpdate for zero downtime

- Developer deploys new template containing change to Launch Configuration
- CloudFormation replaces a single batch of EC2 instances, the others remain in service
- ELB health check confirms new instances are healthy (otherwise rollback)
- Next batch of EC2 instances are replaced

Optimizing the ASG UpdatePolicy

- On test environments you can **optimize for speed** and replace all instances at once
- Once live, you should update the ASG in batches making sure you don’t have downtime

…for example

For a service with an ASG with 6 instances…

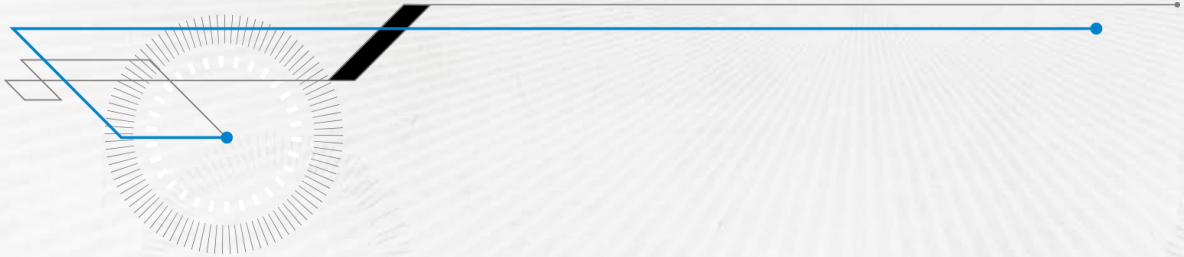
TEST

```
"UpdatePolicy": {
  "AutoScalingRollingUpdate": {
    "PauseTime": "PT0S",
    "MaxBatchSize": “6",
    "MinInstancesInService": "0"
  }
}
```

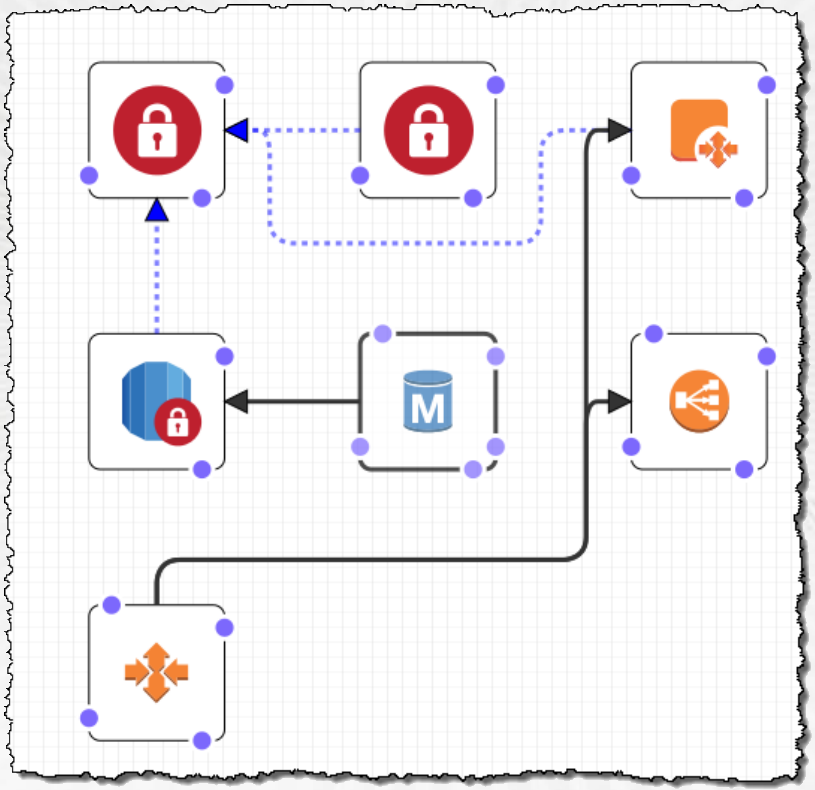
LIVE

```
"UpdatePolicy": {
  "AutoScalingRollingUpdate": {
    "PauseTime": "PT15S",
    "MaxBatchSize": "2",
    "MinInstancesInService": "2"
  }
}
```

What’s New



Template Designer



Support for CloudWatch Events, API Gateway and WAF

```
{
  "Type" : "AWS::Events::Rule",
  "Properties" : {
    "Description" : String,
    "EventPattern" : JSON object,
    "Name" : String,
    "RoleArn" : String,
    "ScheduleExpression" : String,
    "State" : String,
    "Targets" : [ Target, ... ]
  }
}
```

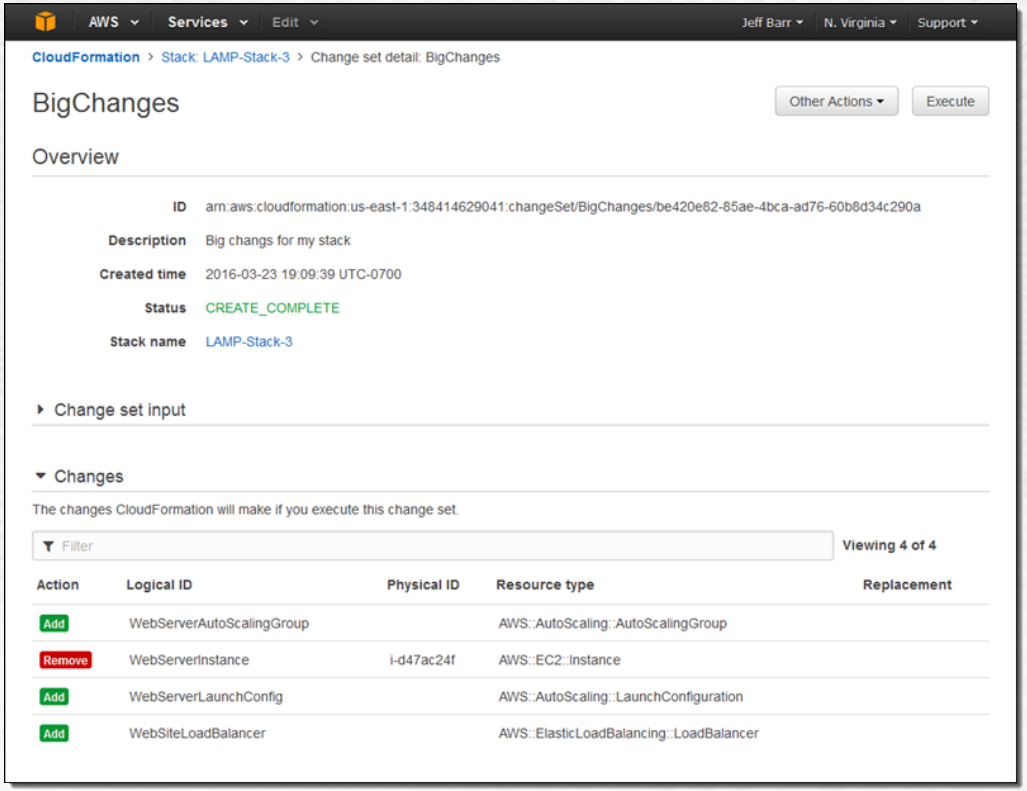
Support for CloudWatch Events, API Gateway and WAF

```
{
  "Type" : "AWS::ApiGateway::Method",
  "Properties" : {
    "ApiKeyRequired" : Boolean,
    "AuthorizationType" : String,
    "AuthorizerId" : String,
    "HttpMethod" : String,
    "Integration" : Integration,
    "MethodResponses" : [ MethodResponse, ... ],
    "RequestModels" : { String:String, ... },
    "RequestParameters" : { String:Boolean, ... },
    "ResourceId" : String,
    "RestApiId" : String
  }
}
```

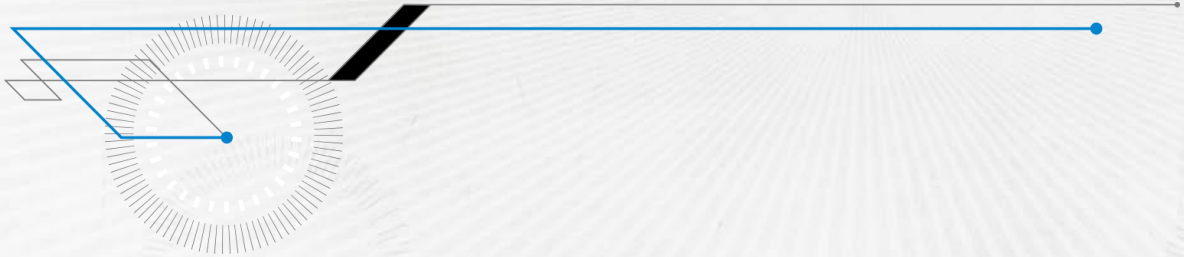
Support for CloudWatch Events, API Gateway and WAF

```
{
  "Type" : "AWS::WAF::XssMatchSet",
  "Properties" : {
    "Name" : String,
    "XssMatchTuples" : [ XssMatchTuple, ... ]
  }
}
```

Change Sets


AWS
Services
Edit
Jeff Barr
N. Virginia
Support
CloudFormation > Stack: LAMP-Stack-3 > Change set detail: BigChanges
BigChanges
Other Actions
Execute
Overview
ID
arn:aws:cloudformation:us-east-1:348414629041:changeSet/BigChanges/be420e82-85ae-4bca-ad76-60b8d34c290a
Description
Big changs for my stack
Created time
2016-03-23 19:09:39 UTC-0700
Status
CREATE_COMPLETE
Stack name
LAMP-Stack-3
Change set input
Changes
The changes CloudFormation will make if you execute this change set.
Filter
Viewing 4 of 4
Action
Logical ID
Physical ID
Resource type
Replacement
Add
WebServerAutoScalingGroup
AWS::AutoScaling::AutoScalingGroup
Remove
WebServerInstance
i-d47ac24f
AWS::EC2::Instance
Add
WebServerLaunchConfig
AWS::AutoScaling::LaunchConfiguration
Add
WebSiteLoadBalancer
AWS::ElasticLoadBalancing::LoadBalancer

Best Practices



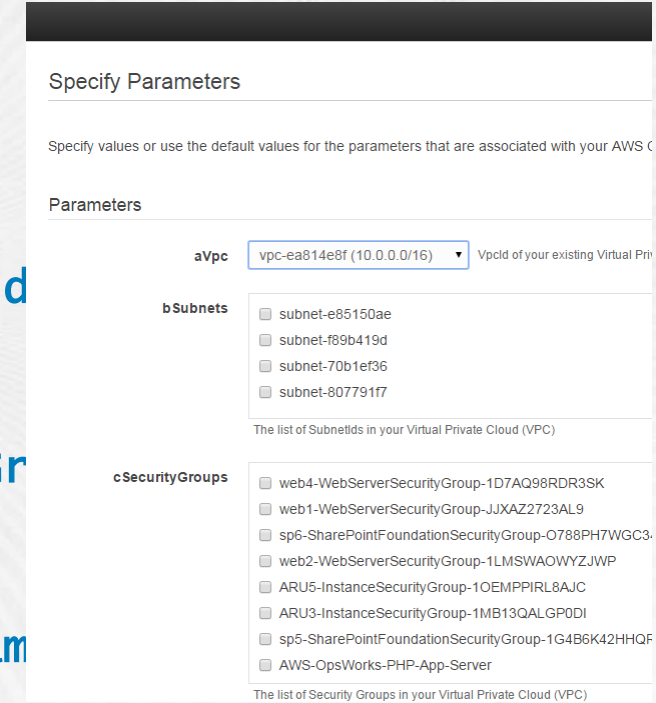
Validate your templates

ValidateTemplate API validates:

- JSON syntax
- Absence of circular dependencies
- Template structure

Use parameter types

```
"Parameters" : {
    “VpcId" : {
        "Type" : "AWS::EC2::VPC::Id"
    },
    “SubnetIds" : {
        "Type" : "List<AWS::EC2::Subnet::Id
    },
    “SecurityGroups" : {
        "Type" : "List<AWS::EC2::SecurityGr
    },
    “KeyPair" : {
        "Type" : "AWS::EC2::KeyPair::KeyNam
    }
```



Use Deletion Policies to protect resources

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myS3Bucket" : {
      "Type" : "AWS::S3::Bucket",
      "DeletionPolicy" : "Retain"
      "Properties" : {
        "BucketName" : “MyBucket”
      }
    }
  }
}
```

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Resources" : {
    "myVolume" : {
      "Type":"AWS::EC2::Volume",
      "DeletionPolicy" : "Snapshot”
      "Properties" : {
        "AvailabilityZone" :”us-east-1a”,
        "Size” : “100”
      }
    }
  }
}
```

On stack deletion:

MyBucket

myVolume

Snapshot

Use stack policies as update guardrails

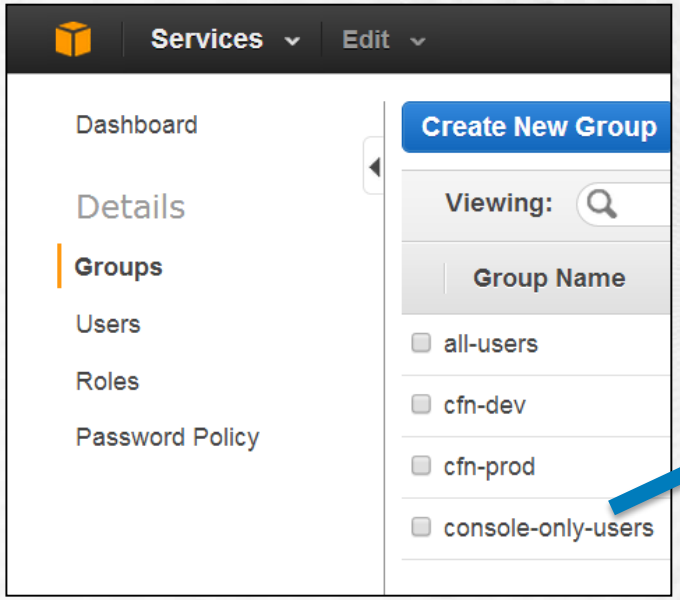
“Do not update the databases”

```
"Effect" : "Deny",
"Principal" : "*",
"Action" : "Update:*",
"Resource" : "*",
"Condition" : {
  "StringEquals" : {
    "ResourceType” : [
      "AWS::RDS::DBInstance”,
      "AWS::Redshift::Cluster”
    ]
  }
}
```

“Okay to update, unless the update requires replacement”

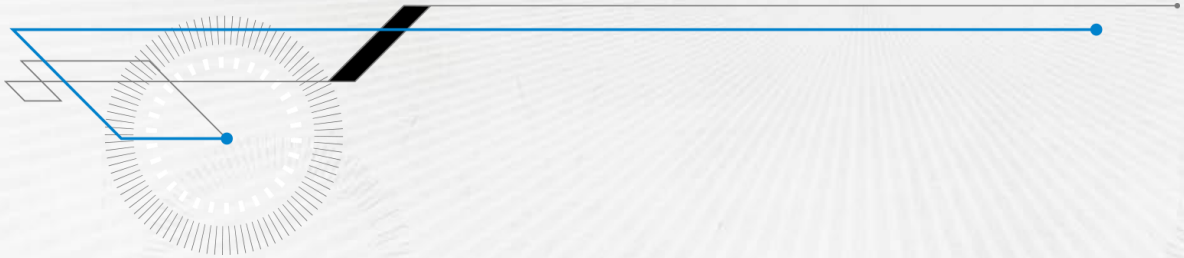
```
"Effect" : "Deny",
"Principal": "*",
"Action" : "Update:Replace",
"Resource" : "LogicalResourceId/MyInstance"
```

Stop “configuration drift” with IAM and tags




```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": ”Deny",
      "Action": “*",
      "Resource": "*"
    },
    {
      "Effect" : ”Allow",
      "Action" : [
        "Action": "ec2:Describe*”
      ],
      "Condition": {
        "Null": { "ec2:ResourceTag/*cloudformation*" : "true" }
      },
      "Resource" : "*"
    }
  ]
}
```

Back to the Demo!



Next steps

- Load your existing templates into the template designer for inspection
- Migrate scripts from UserData to CFN-Init directives
- Review your existing permissions/guardrails
- Start using specific parameter types