AWS ENTERPRISE SUMMIT

DevOps on AWS

Deep Dive on Infrastructure as Code

Hubert Cheung, Solutions Architect

September 2016

Why are we here today?

https://secure.flickr.com/photos/mgifford/4525333972

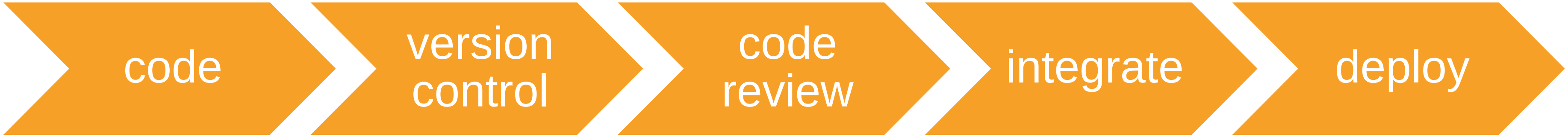
Why are we here today?

Moving to cloud based infrastructure opens doors to building and managing infrastructure in completely new ways:

- Infrastructure can be provisioned in seconds
- Scale can be achieved without complicated capacity planning
- Being API driven means we can interact with our infrastructure via languages typically used in our applications

Infrastructure as Code is a practice by where traditional infrastructure management techniques are supplemented and often replaced by using code based tools and software development techniques.

Infrastructure as Code workflow



code

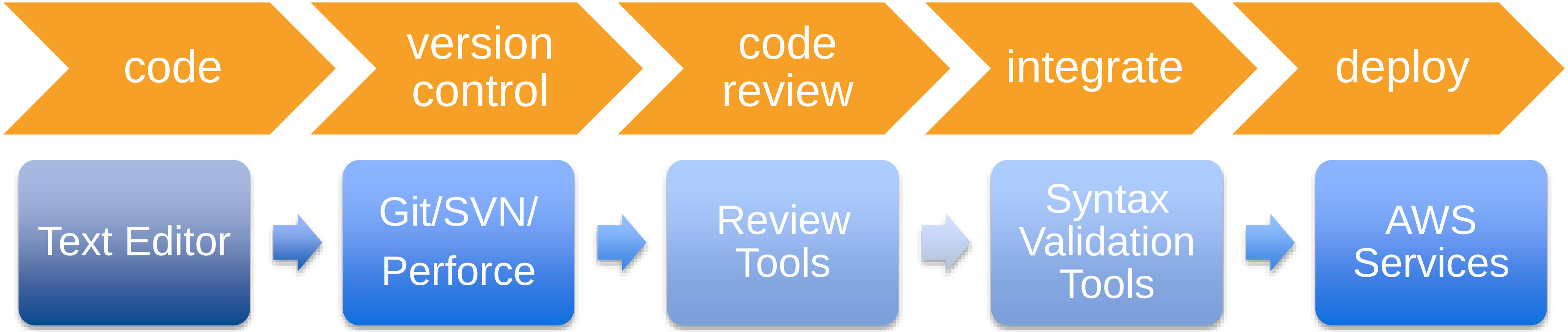
version control

code review

integrate

deploy

Infrastructure as Code workflow



code

version control

code review

integrate

deploy

Text Editor

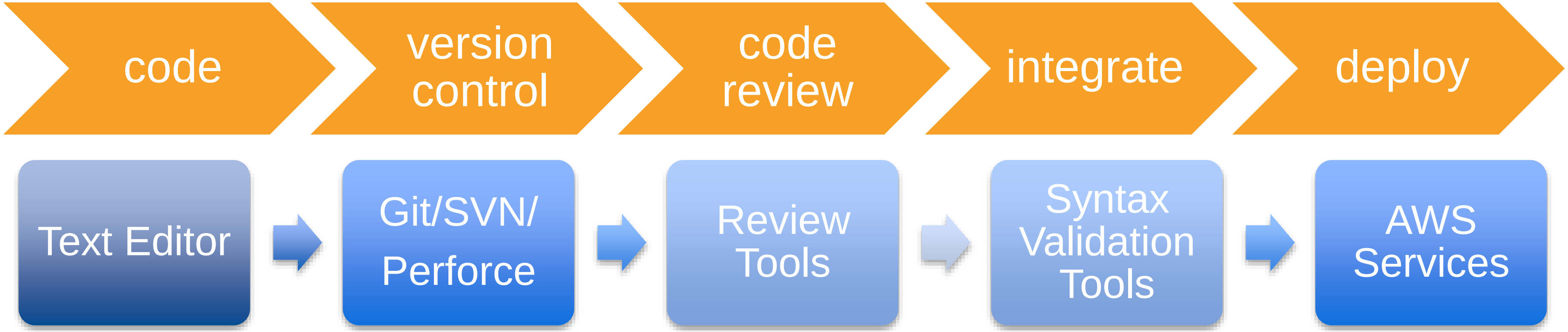
Git/SVN/ Perforce

Review Tools

Syntax Validation Tools

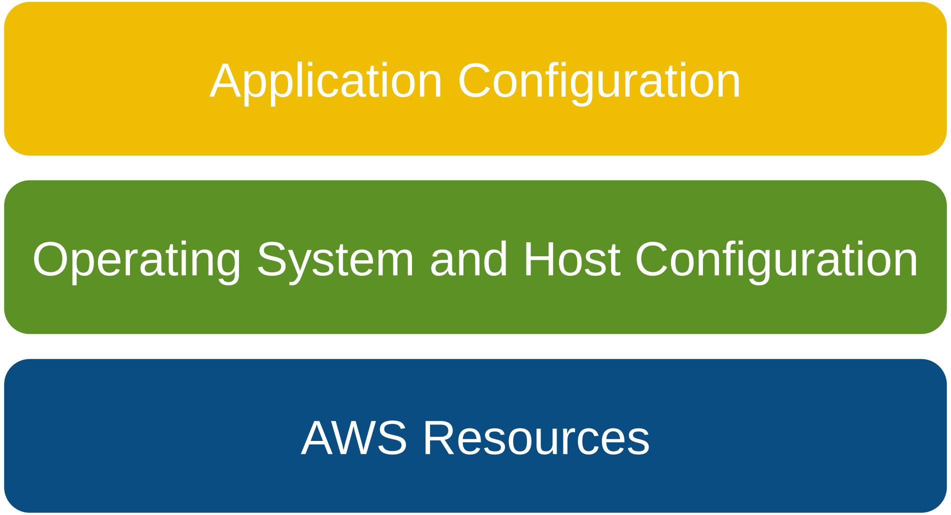
AWS Services

Infrastructure as Code workflow



“It’s all software”

“It’s all software”



Application Configuration

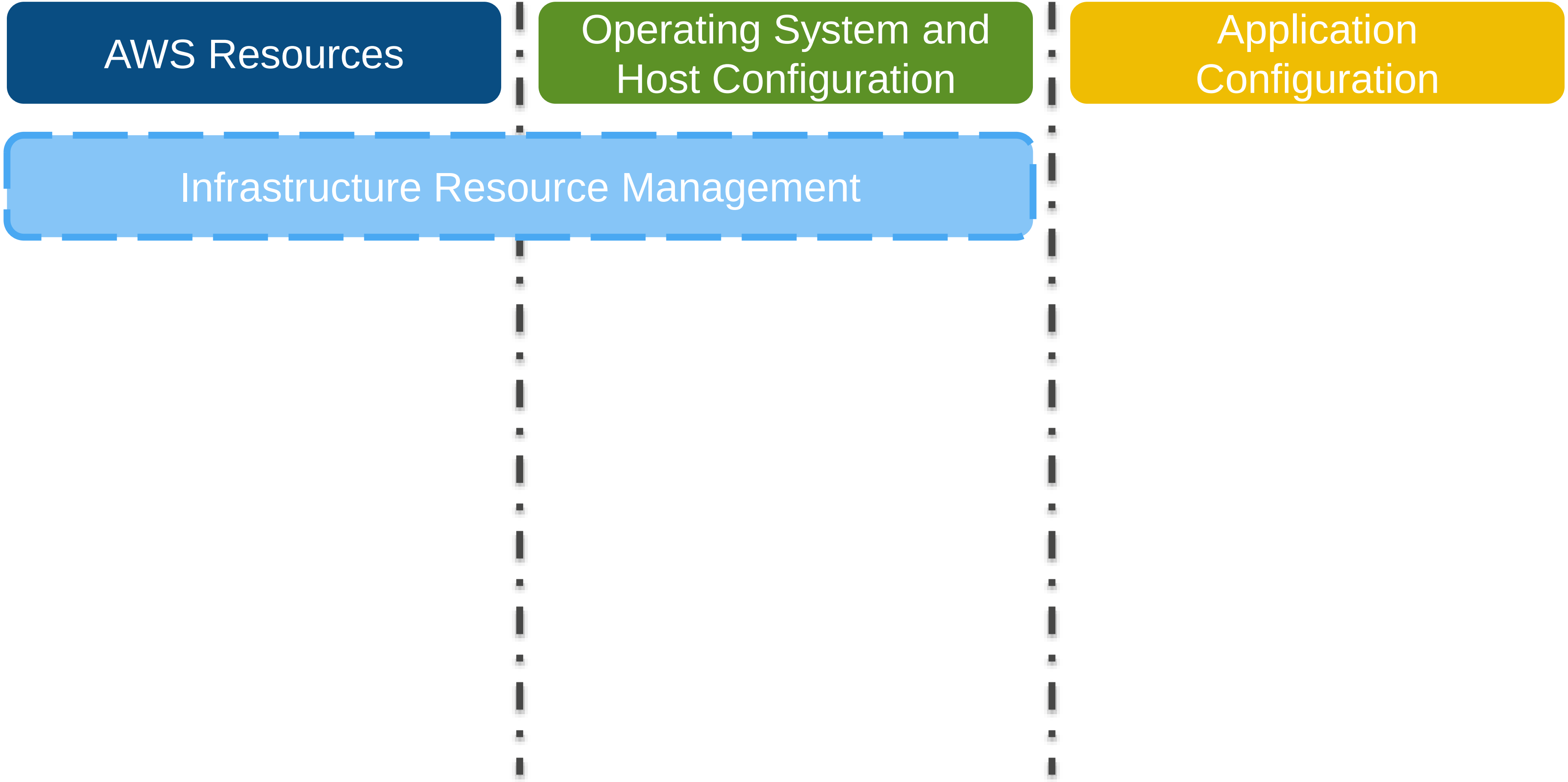
Operating System and Host Configuration

AWS Resources

AWS Resources

Operating System and Host Configuration

Application Configuration

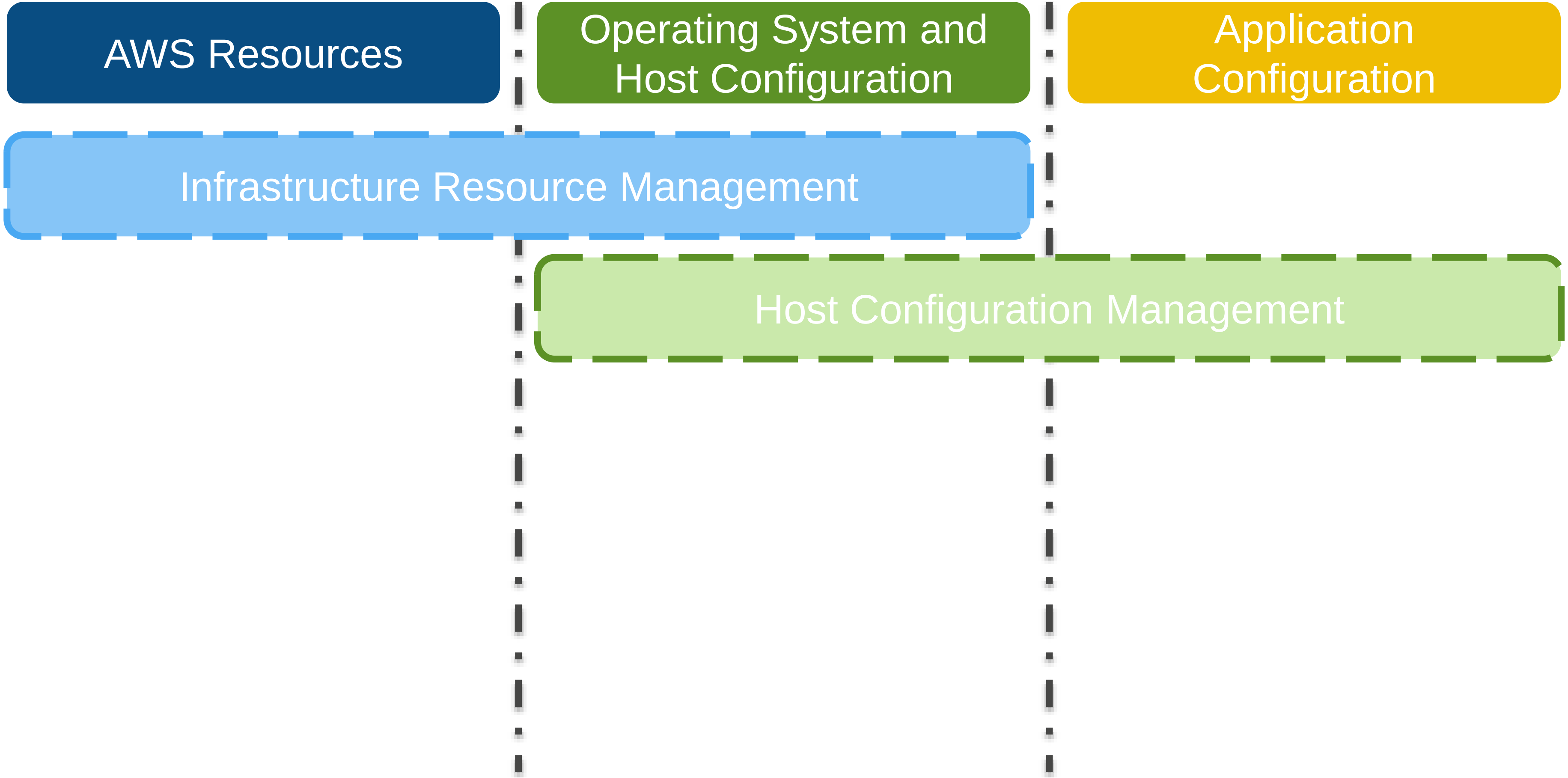


AWS Resources

Operating System and Host Configuration

Application Configuration

Infrastructure Resource Management



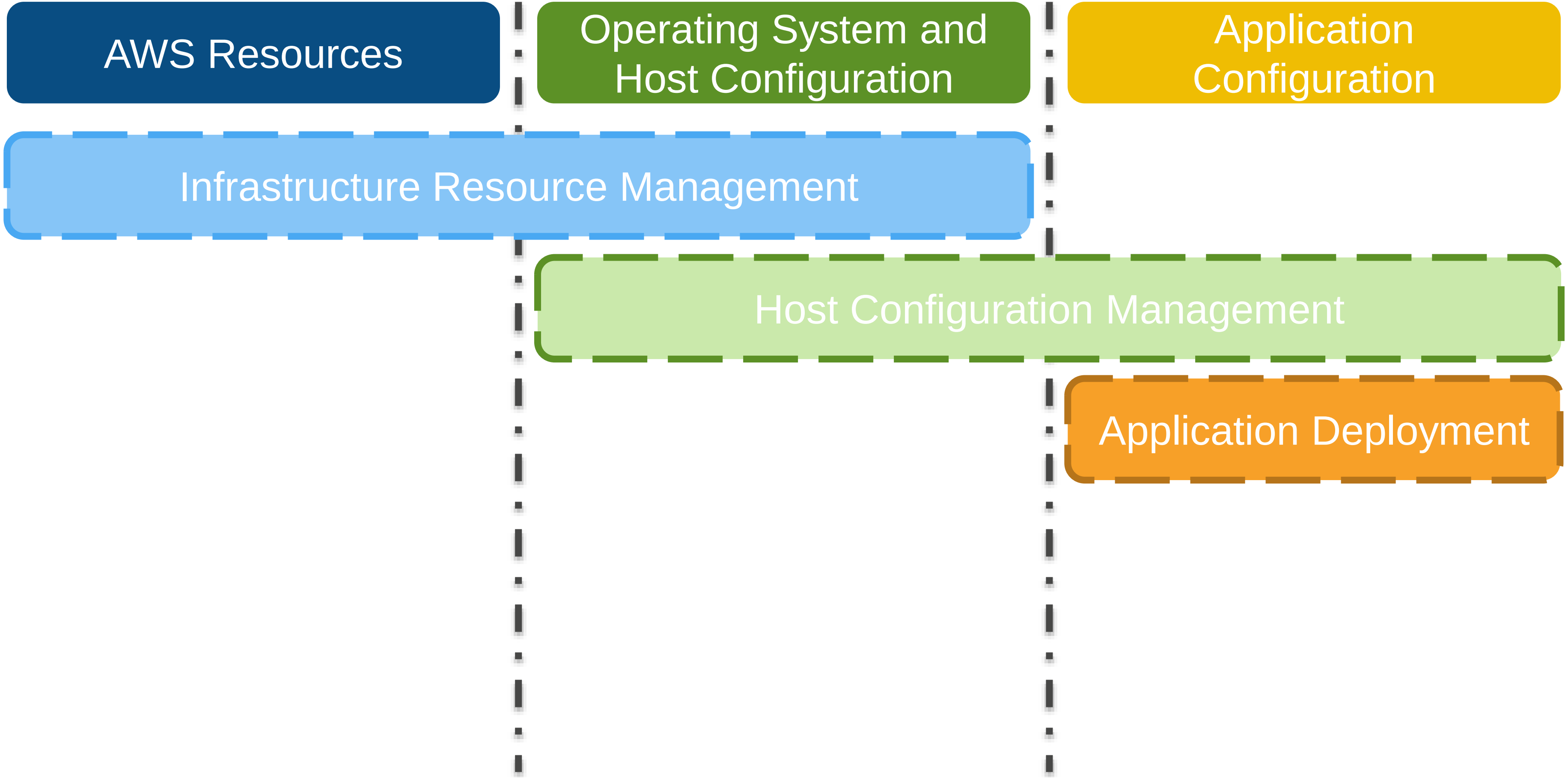
AWS Resources

Operating System and Host Configuration

Application Configuration

Infrastructure Resource Management

Host Configuration Management



AWS Resources

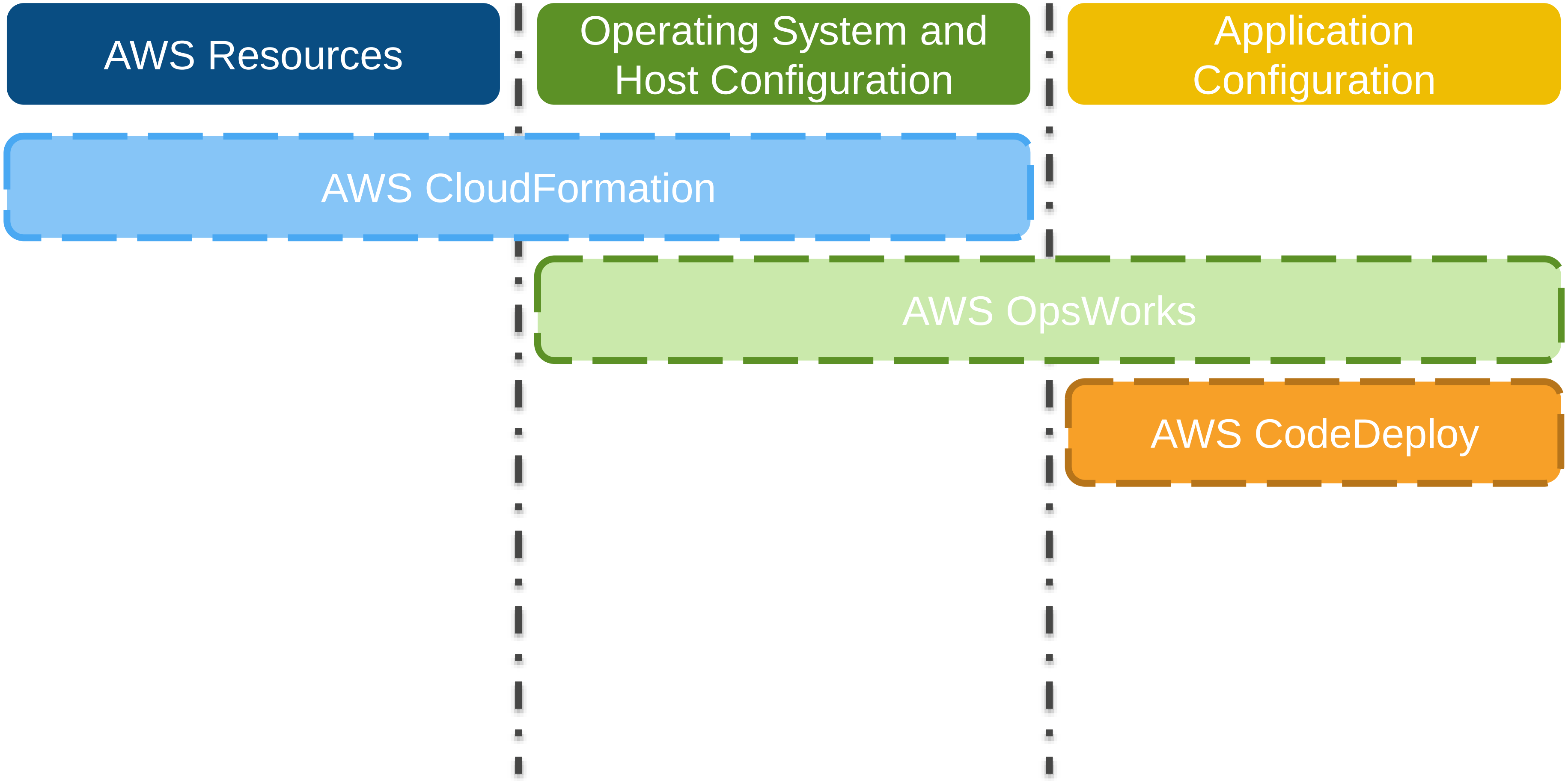
Operating System and Host Configuration

Application Configuration

Infrastructure Resource Management

Host Configuration Management

Application Deployment



AWS Resources

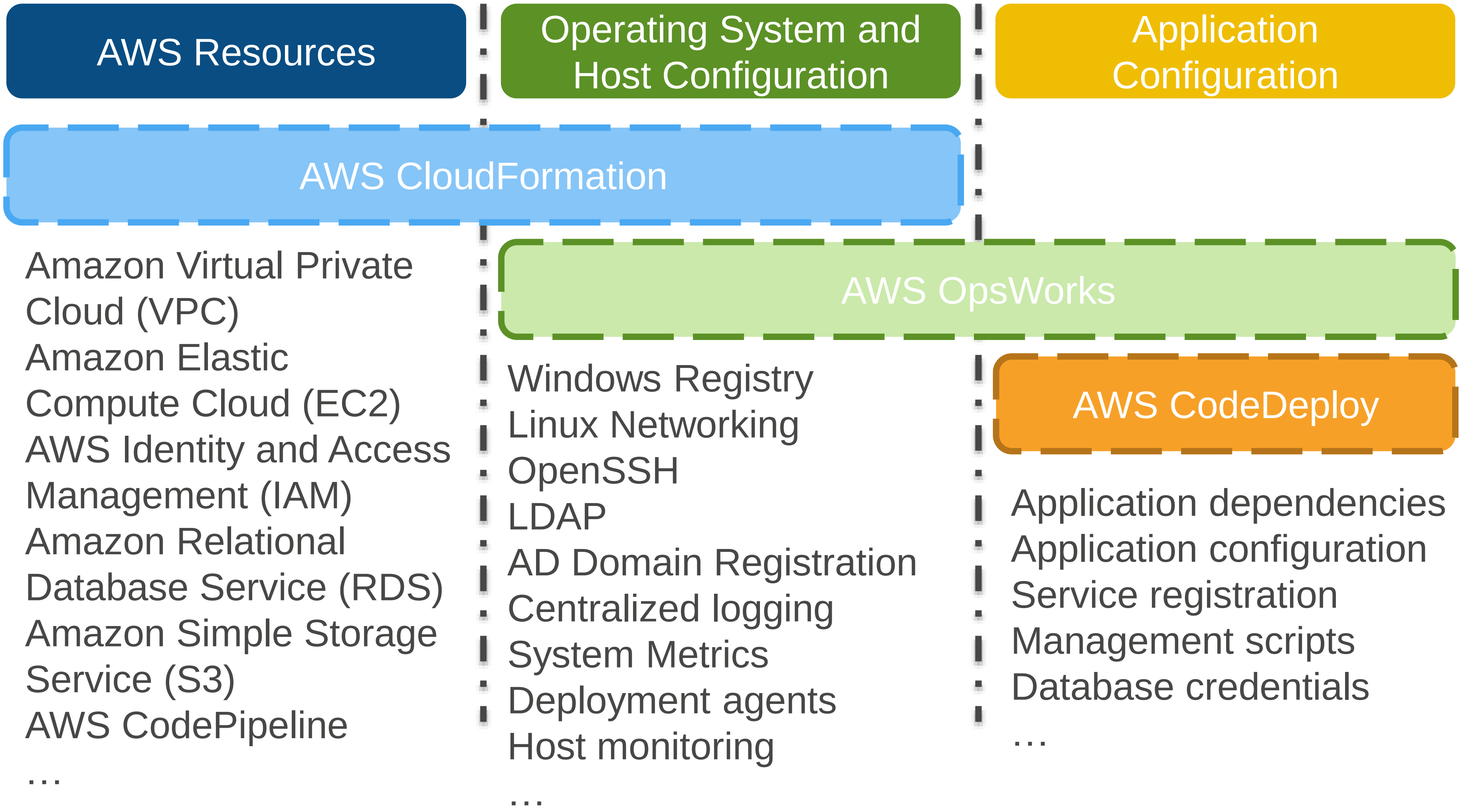
Operating System and Host Configuration

Application Configuration

AWS CloudFormation

AWS OpsWorks

AWS CodeDeploy



AWS Resources

Operating System and Host Configuration

Application Configuration

AWS CloudFormation

AWS OpsWorks

AWS CodeDeploy

Amazon Virtual Private Cloud (VPC)
Amazon Elastic Compute Cloud (EC2)
AWS Identity and Access Management (IAM)
Amazon Relational Database Service (RDS)
Amazon Simple Storage Service (S3)
AWS CodePipeline
…

Windows Registry
Linux Networking
OpenSSH
LDAP
AD Domain Registration
Centralized logging
System Metrics
Deployment agents
Host monitoring
…

Application dependencies
Application configuration
Service registration
Management scripts
Database credentials
…

allOfThis == $Code

https://secure.flickr.com/photos/wscullin/3770015991

AWS CloudFormation

Create templates of your infrastructure

CloudFormation provisions AWS resources based on dependency needs

Version control/replicate/update templates like code

Integrates with development, CI/CD, management tools

Launched in 2010

CloudFormation – Components & Technology

Template

JSON formatted file

Parameter definition

Resource creation

Configuration actions

CloudFormation

Framework

Stack creation

Stack updates

Error detection and rollback

Stack

Configured AWS resources

Comprehensive service support

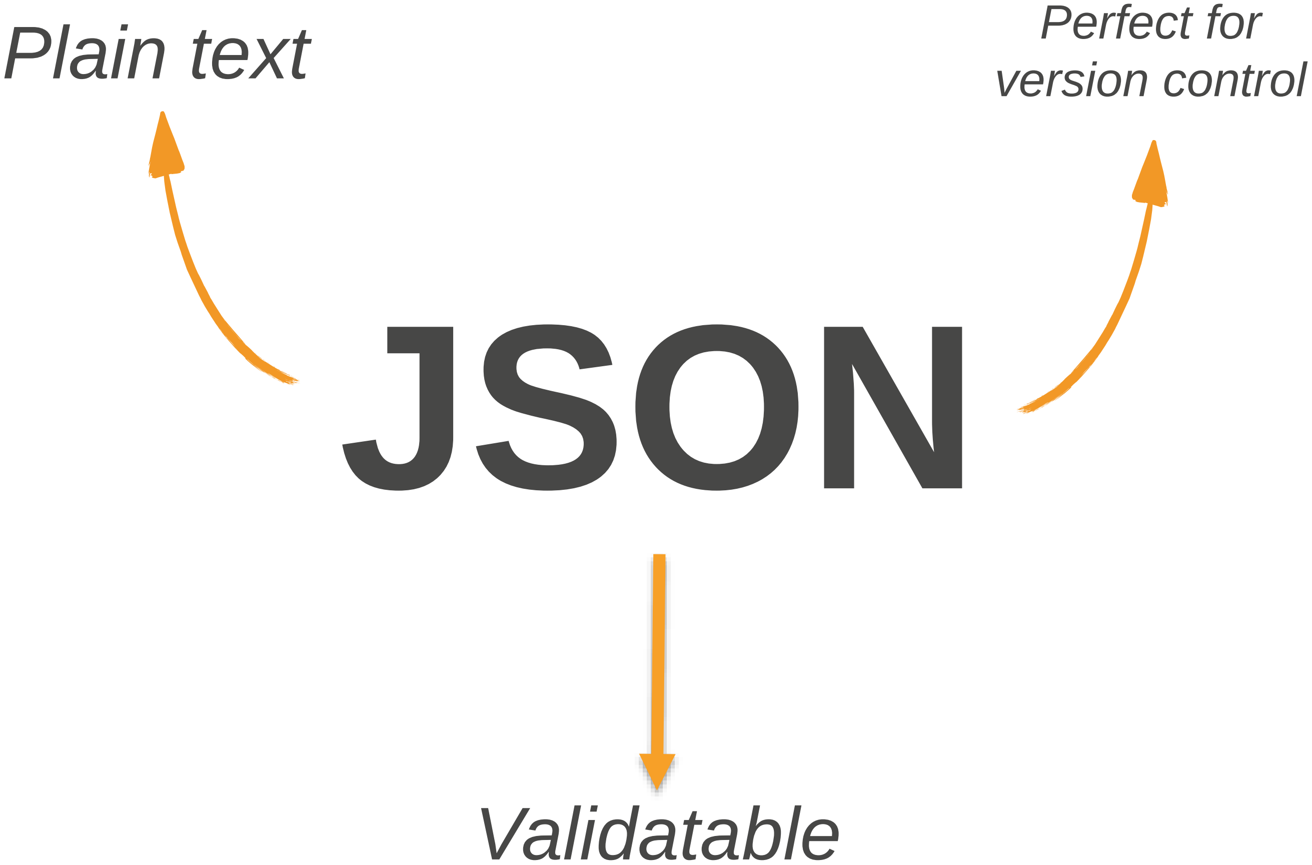
Service event aware

Customizable

Anatomy of a template

Declarative language

JSON



Plain text

Perfect for
version control

JSON

Validatable

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

  "Description" : "AWS CloudFormation Sample Template EC2InstanceSample: **WARNING** This template an Amazon EC2 instances. You will be billed for the AWS resources used if
you create a stack from this template.",

  "Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",
      "Type" : "String"
    },

     "Environment": {
       "Type" : "String",
       "Default" : ”Dev",
       "AllowedValues" : [”Mgmt", "Dev", ”Staging", "Prod"],
        "Description" : "Environment that the instances will run in.”
      }
  },

  "Mappings" : {
    "RegionMap" : {
      "us-east-1"      : { "AMI" : "ami-7f418316" },
      "us-west-2"      : { "AMI" : "ami-16fd7026" }
    }
  },

 "Conditions" : {
     ”EnableEBSOptimized" : {"Fn::Equals" : [{"Ref" : " Environment "}, ”Prod"]},
 },

  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "EbsOptimized " : {"Fn::If": [ " EnableEBSOptimized ", {“true”}, {“false”}]]},
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
        "UserData" : { "Fn::Base64" : "80" }
      }
    }
  },

  "Outputs" : {
    "InstanceId" : {
      "Description" : "InstanceId of the newly created EC2 instance",
      "Value" : { "Ref" : "Ec2Instance" }
    },
     "PublicDNS" : {
      "Description" : "Public DNSName of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }
    }
  }
}
```

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

 "Description" : "AWS CloudFormation Sample Template EC2InstanceSample: **WARNING** This template an Amazon EC2 instances. You will be billed for the AWS resources used if
you create a stack from this template.",
 "Parameters" : {
  "KeyName" : {
   "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",
   "Type" : "String"
  },

  "Environment": {
   "Type" : "String",
   "Default" : ”Dev",
   "AllowedValues" : [”Mgmt", "Dev", ”Staging", "Prod"],
    "Description" : "Environment that the instances will run in.”
   }
 },
 "Mappings" : {
  "RegionMap" : {
   "us-east-1"      : { "AMI" : "ami-7f418316" },
   "us-west-2"      : { "AMI" : "ami-16fd7026" }
  }
 },
"Conditions" : {
    ”EnableEBSOptimized" : {"Fn::Equals" : [{"Ref" : " Environment "}, ”Prod"]},
},
 "Resources" : {
  "Ec2Instance" : {
   "Type" : "AWS::EC2::Instance",
   "Properties" : {
    "KeyName" : { "Ref" : "KeyName" },
    "EbsOptimized " : {"Fn::If": [ " EnableEBSOptimized ", {“true”}, {“false”}]},
    "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
    "UserData" : { "Fn::Base64" : "80" }
   }
  }
 },
 "Outputs" : {
  "InstanceId" : {
   "Description" : "InstanceId of the newly created EC2 instance",
   "Value" : { "Ref" : "Ec2Instance" }
  },
  "PublicDNS" : {
   "Description" : "Public DNSName of the newly created EC2 instance",
   "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }
  }
 }
}
```

HEADERS

PARAMETERS

MAPPINGS

CONDITIONALS

RESOURCES

OUTPUTS

HEADERS

Description of what your stack does, contains, etc

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

  "Description" : "AWS CloudFormation Sample Template EC2InstanceSample: **WARNING** This template an Amazon EC2 instances. You will be billed for the AWS resources used if
you create a stack from this template.",
```

PARAMETERS

Provision time values that add structured flexibility and customization

```
  "Parameters" : {
    "KeyName" : {
      "Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",
      "Type" : "String"
    },
    "Environment": {
      "Type" : "String",
      "Default" : "Dev",
      "AllowedValues" : ["Mgmt", "Dev", "Staging", "Prod"],
       "Description" : "Environment that the instances will run in."
    }
  },
```

MAPPINGS

Pre-defined conditional case statements

```
  "Mappings" : {
    "RegionMap" : {
      "us-east-1"      : { "AMI" : "ami-7f418316" },
      "us-west-2"      : { "AMI" : "ami-16fd7026" }
    }
  },
```

CONDITIONALS

Conditional values set via evaluations of passed references

```
  "Conditions" : {
    "EnableEBSOptimized" : {"Fn::Equals" : [{"Ref" : "Environment"}, "Prod"]},
  },
```

RESOURCES

AWS resource definitions

```
  "Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "EbsOptimized " : {"Fn::If": [ " EnableEBSOptimized", {"true"}, {"false"}]},
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
        "UserData" : { "Fn::Base64" : "80" }
      }
    }
  },
```

OUTPUTS

Resulting attributes of stack resource creation

```
  "Outputs" : {
    "InstanceId" : {
      "Description" : "InstanceId of the newly created EC2 instance",
      "Value" : { "Ref" : "Ec2Instance" }
    },
    "PublicDNS" : {
      "Description" : "Public DNSName of the newly created EC2 instance",
      "Value" : { "Fn::GetAtt" : [ "Ec2Instance", "PublicDnsName" ] }
    }
  }
}
```

AWS CloudFormation

Templates (in action):

```
"ImageId" : { "Fn::FindInMap" : [ "AWSRegionVirt2AMI", { "Ref" : "AWS::Region" },
{"Fn::FindInMap": ["AWSInstanceType2Virt", { "Ref" : "myInstanceType" }, "Virt"]} ]},
```

AWS CloudFormation

“AWSRegionVirt2AMI” Map

Templates (in action):

```
"ImageId" : { "Fn::FindInMap" : [ "AWSRegionVirt2AMI", { "Ref" : "AWS::Region" },
{"Fn::FindInMap": ["AWSInstanceType2Virt", { "Ref" : "myInstanceType" }, "Virt"]} ]},
```

AWS CloudFormation

“AWSRegionVirt2AMI” Map

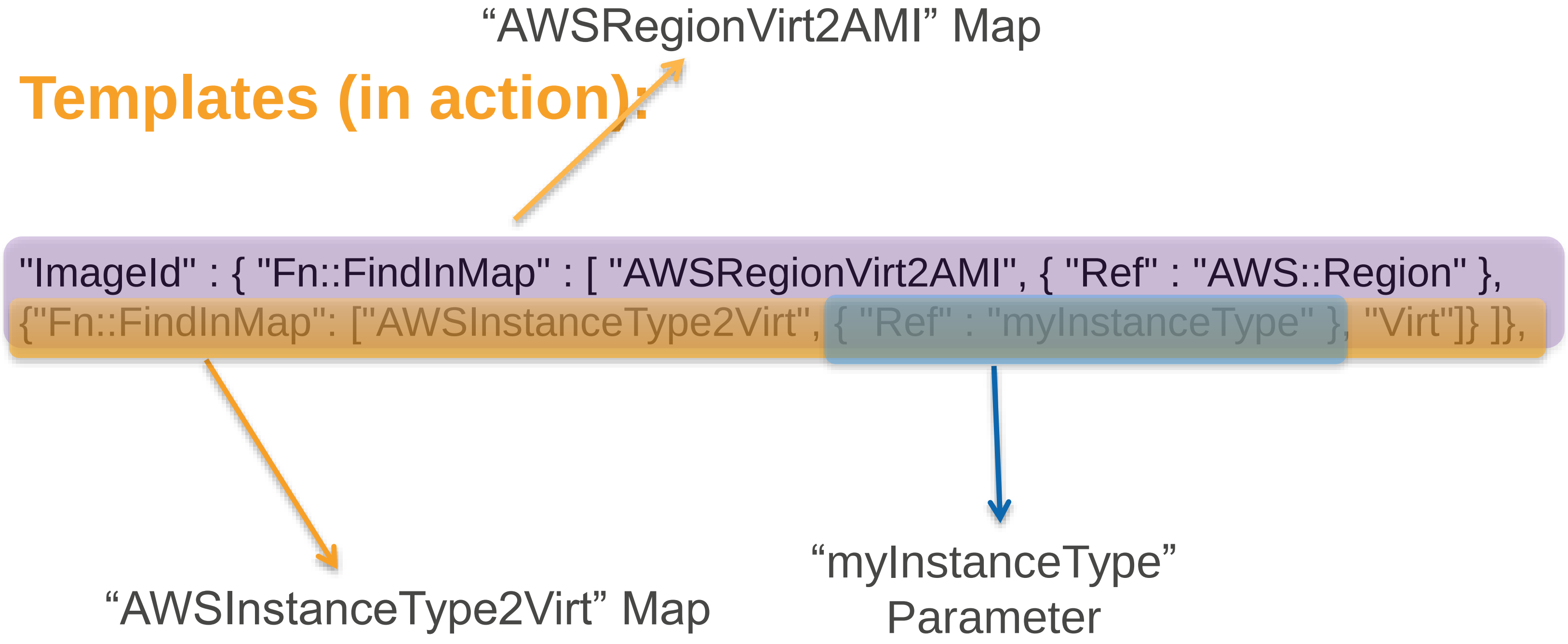
Templates (in action):

```
"ImageId" : { "Fn::FindInMap" : [ "AWSRegionVirt2AMI", { "Ref" : "AWS::Region" },
{"Fn::FindInMap": ["AWSInstanceType2Virt", { "Ref" : "myInstanceType" }, "Virt"]} ]},
```

“AWSInstanceType2Virt” Map

AWS CloudFormation

Templates (in action):



```
"ImageId" : { "Fn::FindInMap" : [ "AWSRegionVirt2AMI", { "Ref" : "AWS::Region" },
{"Fn::FindInMap": ["AWSInstanceType2Virt", { "Ref" : "myInstanceType" }, "Virt"]} ]},
```

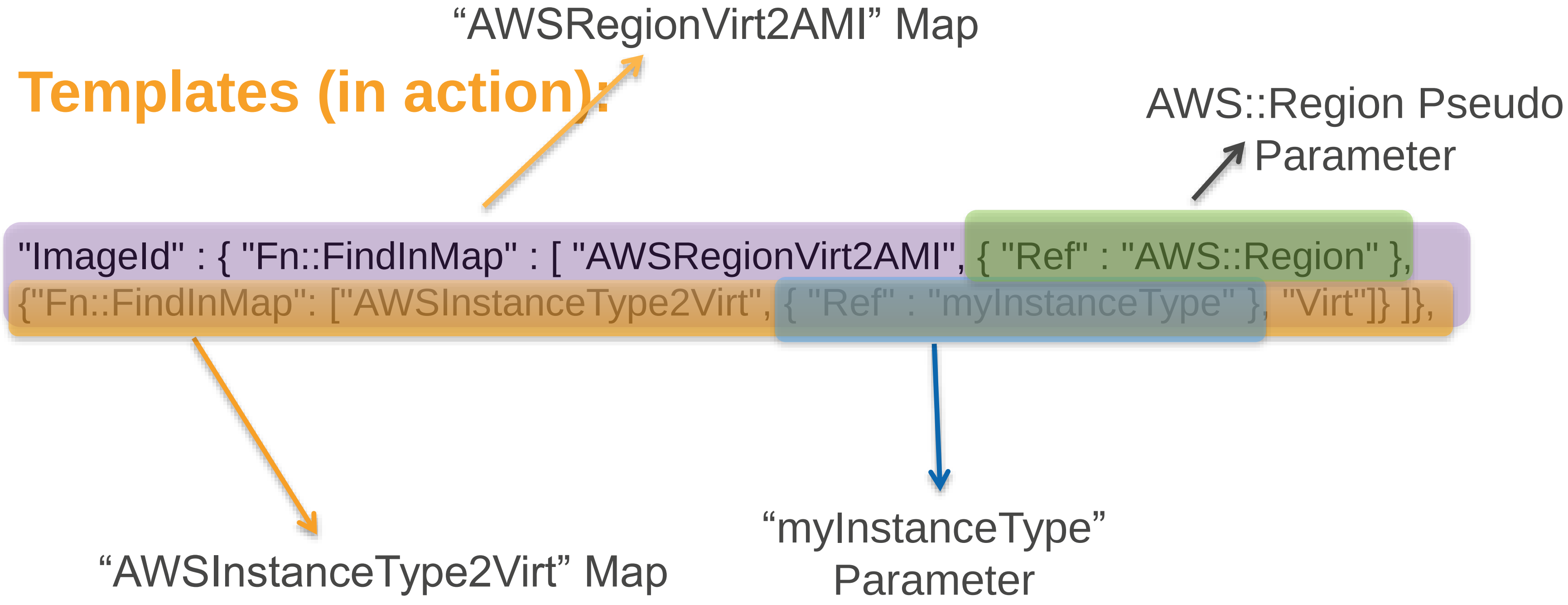
"AWSRegionVirt2AMI" Map

"AWSInstanceType2Virt" Map

"myInstanceType"
Parameter

AWS CloudFormation

Templates (in action):



```
"ImageId" : { "Fn::FindInMap" : [ "AWSRegionVirt2AMI", { "Ref" : "AWS::Region" },
{"Fn::FindInMap": ["AWSInstanceType2Virt", { "Ref" : "myInstanceType" }, "Virt"]} ]},
```

“AWSRegionVirt2AMI” Map

AWS::Region Pseudo Parameter

“AWSInstanceType2Virt” Map

“myInstanceType” Parameter

AWS CloudFormation

Parameters:

```
 "myInstanceType" : {
    "Type" : "String",
    "Default" : "t2.large",
    "AllowedValues" : ["t2.micro", "t2.small",
"t2.medium", "t2.large"],
    "Description" : "Instance type for instances created,
must be in the t2 family."
}
```

Mappings:

```
"AWSInstanceType2Virt": {
      "t2.micro": {"Virt": "HVM"},
      "t2.small": {"Virt": "HVM"},
      "t2.medium": {"Virt": "HVM"},
      "t2.large": {"Virt": "HVM"},
 }
```

Mappings:

```
"AWSRegionVirt2AMI": {
      "us-east-1": {
        "PVM": "ami-50842d38",
        "HVM": "ami-08842d60"
      },
      "us-west-2": {
        "PVM": "ami-af86c69f",
        "HVM": "ami-8786c6b7"
      },
      "us-west-1": {
        "PVM": "ami-c7a8a182",
        "HVM": "ami-cfa8a18a"
      }
    }
```

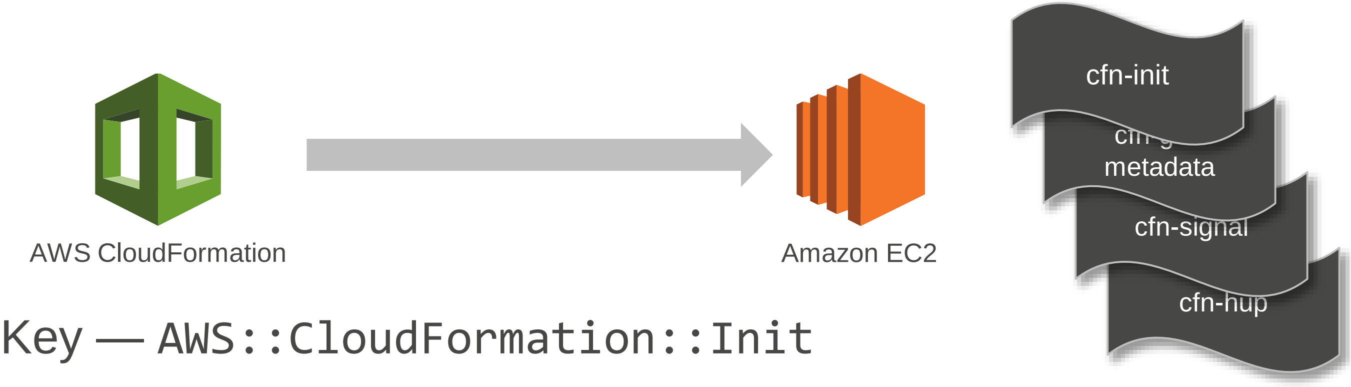
Bootstrapping Applications & Handling Updates

Option 1: Use EC2 UserData, which is available as a property of AWS::EC2::Instance resources

```
"Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyName" },
        "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "AMI" ]},
        "UserData" : { "Fn::Base64" : { "Fn::Join" : ["",[
            "#!/bin/bash -ex","\n",
            "yum -y install gcc-c++ make","\n",
            "yum -y install mysql-devel sqlite-devel","\n",
            "yum -y install ruby-rdoc rubygems ruby-mysql ruby-devel","\n",
            "gem install --no-ri --no-rdoc rails","\n",
            "gem install --no-ri --no-rdoc mysql","\n",
            "gem install --no-ri --no-rdoc sqlite3","\n",
            "rails new myapp","\n",
            "cd myapp","\n",
            "rails server -d","\n"]]}}
      }
    }
```

Bootstrapping Applications & Handling Updates

Option 2: AWS CloudFormation provides helper scripts for deployment within your EC2 instances



Metadata Key — `AWS::CloudFormation::Init`

Cfn-init reads this metadata key and installs the packages listed in this key (e.g., httpd, mysql, and php). Cfn-init also retrieves and expands files listed as sources.

AWS CloudFormation

Use AWS::CloudFormation::Init with cfn-init to help bootstrap instances:

```
"Metadata": {
  "AWS::CloudFormation::Init" : {
        "config" : {
          "packages" : {
          },
          "sources" : {
          },
          "commands" : {
          },
          "files" : {
          },
          "services" : {
          },
          "users" : {
          },
          "groups" : {
          }
        }
      }
```

AWS CloudFormation

Install packages with the native package management tool:

```
“WebAppHost" : {
      "Type" : "AWS::EC2::Instance",
      "Metadata" : {
        "AWS:CloudFormation::Init" : {
            "config" : {
                "packages" : {
                    "yum" : {
                    "gcc" : [],
                    "gcc-c++" : [],
                    "make" : [],
                    "automake" : [],
```

Manage a wide range of AWS services & resources

- Amazon EC2
- Amazon EC2 Container Service
- Amazon EC2 Simple Systems Manager (**New**)
- AWS Lambda (including event sources)
- Auto Scaling (including Spot Fleet)

- Amazon VPC
- Elastic Load Balancing
- Amazon Route 53
- Amazon CloudFront
- AWS WAF (**New**)

- Amazon RDS
- Amazon Redshift
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon RDS (including Aurora)
- Amazon S3

- Amazon IAM (including managed policies)
- Amazon Simple AD / Microsoft AD (**New**)

- Amazon Kinesis
- Amazon SNS
- Amazon SQS

- AWS CloudTrail
- Amazon CloudWatch
- AWS Config (**New**)
- AWS Key Management Service (**New**)

- AWS Data Pipeline
- AWS Elastic Beanstalk
- AWS OpsWorks
- AWS CodeDeploy
- AWS CodePipeline (**New**)

- Amazon Workspaces

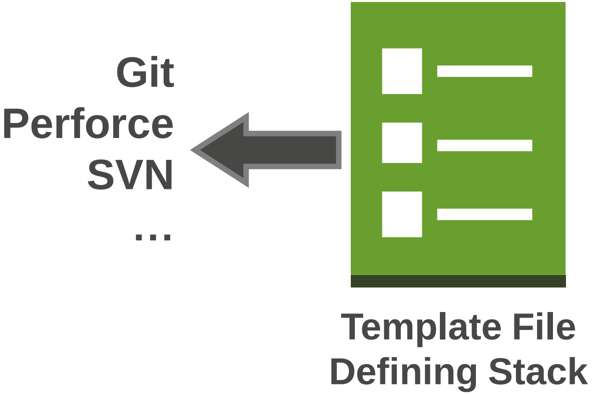
Many Stacks & Environments from One Template

Template File
Defining Stack

The entire infrastructure can be represented in an AWS CloudFormation template.

Many Stacks & Environments from One Template

Use the version control system of your choice to store and track changes to this template



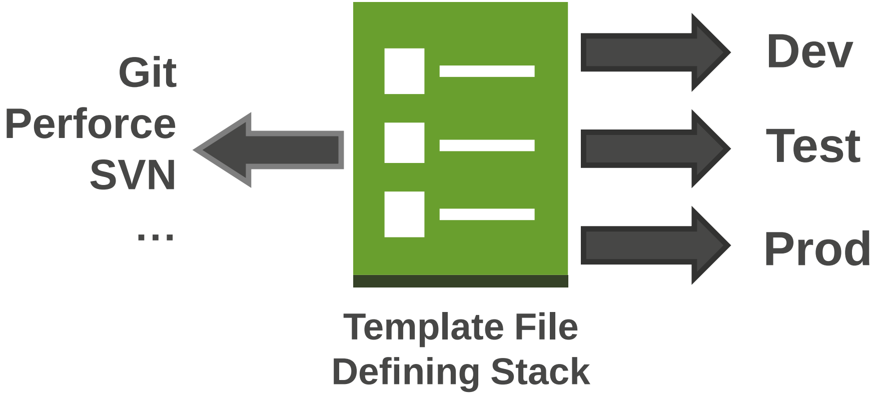
Git
Perforce
SVN
…

Template File Defining Stack

The entire infrastructure can be represented in an AWS CloudFormation template.

Many Stacks & Environments from One Template

Use the version control system of your choice to store and track changes to this template



Git
Perforce
SVN
…

Template File
Defining Stack

Dev
Test
Prod

Build out multiple environments, such as for Development, Test, Production and even DR using the same template

The entire infrastructure can be represented in an AWS CloudFormation template.

AWS CloudFormation

Versioning!

You track changes within your code

Do it within your infrastructure!

- What is changing?
- Who made that change?
- When was it made?
- Why was it made?(tied to ticket/bug/project systems?)

Self imposed, but you need to be doing this!

AWS CloudFormation

Testing:

- Validate via API/CLI
  - $ aws cloudformation validate-template – confirm CF syntax
  - Use something like Jsonlint (http://jsonlint.com/) to find JSON issues like missing commas, brackets!

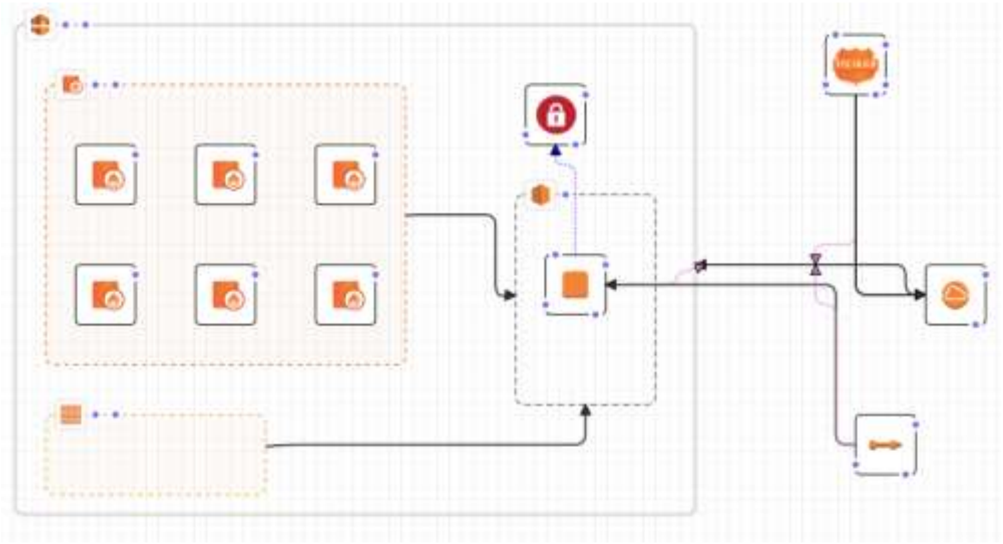
- Throw this into your testing/continuous integration pipelines!

AWS CloudFormation Designer

- Visualize template resources

- Modify template with drag-drop gestures

- Customize sample templates



AWS CloudFormation

Deploy & update via console or API/command line:

- Just a couple of clicks

OR

-
  ```
  aws cloudformation create-stack --stack-name myteststack --template-body file:////home//local//test//sampletemplate.json --parameters ParameterKey=string,ParameterValue=string
  ```

But what do we do once things are up and running?

Ongoing Management

- Updates/patches?
- New software?
- New configurations?
- New code deploys?
- Pool specific changes?
- Environment specific changes?
- Run commands across all hosts?
- Be on top of all running resources?

Could we do this with AWS CloudFormation?

Sure! But potentially tricky to do at scale:

- Try changing a vhost configuration on every web server across multiple environments (dev, stage, prod)
- Install a package on certain hosts, but not others to test out newer versions
- Need to change LDAP config on every running Amazon EC2 Linux host, but they are across 25 different AWS CloudFormation templates?

We need a tool to interact with each host that we manage, to make our lives easier doing the previously mentioned tasks

AWS OpsWorks

Configuration management solution for automating operational tasks

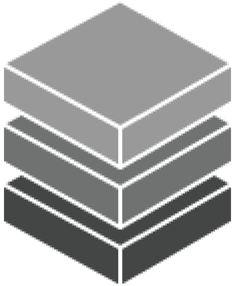
Model, control and automate applications of nearly any scale and complexity

Manage Linux and Windows the same way

Supports both AWS and on-premises

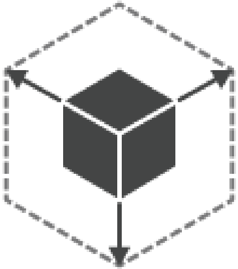
Launched in 2013

AWS OpsWorks

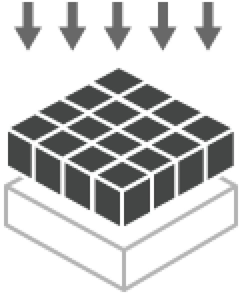


A **stack** represents the cloud infrastructure and applications that you want to manage together.

A **layer** defines how to setup and configure a set of instances and related resources.

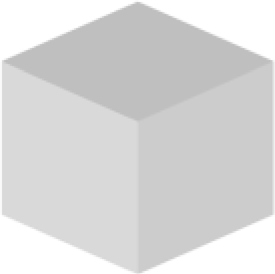


Decide how to scale: manually, with **24/7** instances, or automatically, with **load-based** or **time-based** instances.



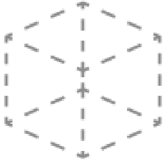
Then deploy your app to specific instances and customize the deployment with Chef recipes.

AWS OpsWorks Instance Lifecycle

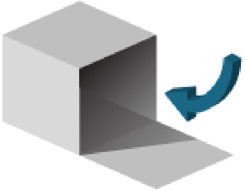


Agent on each instance understands a set of commands that are triggered by OpsWorks. The agent then runs Chef.

Setup



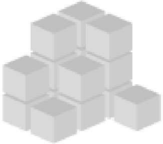
Configure



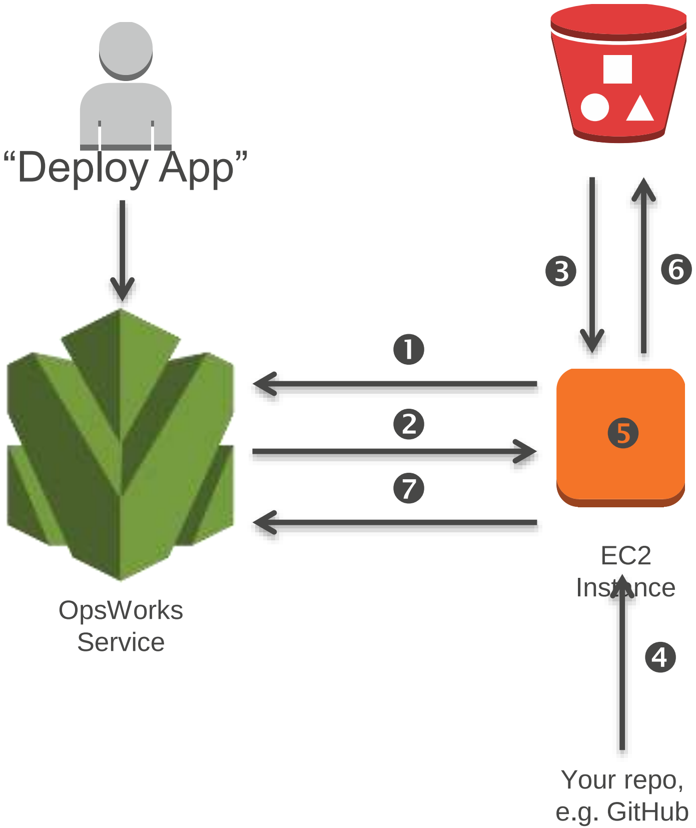
Deploy

Undeploy

Shutdown



OpsWorks Agent Communication



1. Instance connects with OpsWorks service to send keep alive heartbeat and receive lifecycle events
2. OpsWorks sends lifecycle event with pointer to configuration JSON (metadata, recipes) in S3 bucket
3. Download configuration JSON
4. Pull cookbooks and other build assets from your repo
5. Execute recipe
6. Upload Chef log
7. Report Chef run status

How OpsWorks Bootstraps the EC2 instance

Instance is started with IAM role

- UserData passed with instance private key, OpsWorks public key
- Instance downloads and installs OpsWorks agent

Agent connects to instance service, gets run info

- Authenticate instance using instance’s IAM role
- Pick-up configuration JSON from the OpsWorks instance queue
- Decrypt & verify message, run Chef recipes
- Upload Chef log, return Chef run status

Agent polls instance service for more messages

AWS OpsWorks + Chef

OpsWorks uses Chef to configure the software on the instance

OpsWorks provides many Chef Server functions to users.

- Associate cookbooks with instances
- Dynamic metadata that describes each registered node in the infrastructure

Supports "Push" Command and Control Client Runs

Support for community cookbooks

Let’s Start Cooking

https://secure.flickr.com/photos/75001512@N00/8675972995/

Working with Chef

Similar to CloudFormation and application code:

- Mixture of Json and a Ruby DSL
- Tools exist to do linting and syntax checking
- Versioning
  - Built in cookbook versioning
  - Some manual/processes scripted abilities
  - But still use source control!
- Tie in with continuous integration systems just like AWS CloudFormation and the rest of your code!

Working with Chef

Basics:

- Nodes
    - Roles
- Cookbooks
    - Recipes
- Attributes
- Data bags
- Environments

Host Configuration Management with Chef

Examples:

```
package "ntp" do
  action :install
end

service "ntpd" do
  supports :status => true, :restart => true,
:reload => true
  action [ :enable, :start ]
end

cookbook_file "/etc/ntp.conf" do
    source "ntp.conf"
    owner "root"
    group "root"
    mode 00644
    # Restart ntp.conf if /etc/ntp.conf changes
    notifies :restart, resources(:service =>
"ntpd")
end
```

```
group "ganglia" do
  gid 499
end

user "ganglia" do
  home "/var/lib/ganglia"
  shell "/sbin/nologin"
  uid 499
  gid "ganglia"
end

directory "/etc/ganglia" do
  action :create
end
```

Host Configuration Management with Chef

Examples:

```
template "/etc/ganglia/gmond.conf" do
  source "gmond.conf.erb"
  owner "root"
  group "root"
  mode 00644
  notifies :restart, resources(:service => "gmond")
  variables(
    :gmetad_name => node[:ganglia][:gmetad_name],
    :cluster_name => node[:ganglia][:cluster_name]
  )
end

cron "all-gmetrics" do
  command "for FILE in `ls /opt/bin/gmetric-*`; do command $FILE; done >/dev/null 2>&1"
end
```

Host Configuration Management with Chef

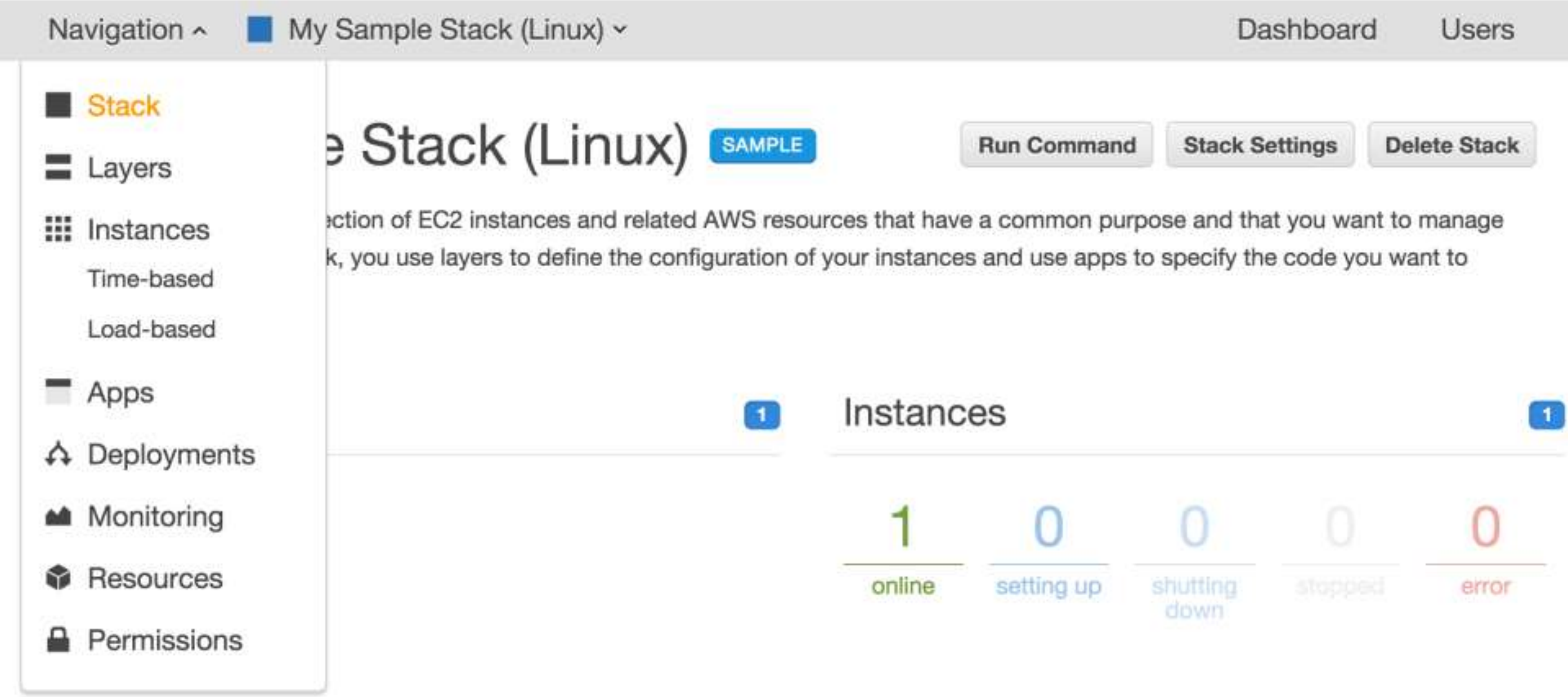
```
{
  "opsworks": {
    "data_bags": {
      "myapp": {
        "mysql": {
          "username": "default-user",
          "password": "default-pass"
        }
      }
    }
  }
}
```

Custom JSON

```
mything = data_bag_item("myapp", "mysql")
Chef::Log.info("username: #{mything['username']}")
```

Recipe

AWS OpsWorks



e Stack (Linux)

ection of EC2 instances and related AWS resources that have a common purpose and that you want to manage k, you use layers to define the configuration of your instances and use apps to specify the code you want to

1	0	0	0	0
online	setting up	shutting down	stopped	error

Deploying your applications

https://secure.flickr.com/photos/simononly/15386966677

AWS CodeDeploy

Rolling deploys with zero downtime

Perform host management as part of a deploy

Deploy applications in any language on any Operating System

Deploy to any instance: AWS or on-premises

Treat all environments the same

Launched in 2014

appspec.yml Example

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: “*.html”
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  ApplicationStart:
    - location: scripts/start_httpd.sh
  ValidateService:
    - location: scripts/test_site.sh
    - location: scripts/register_with_elb.sh
```

appspec.yml Example

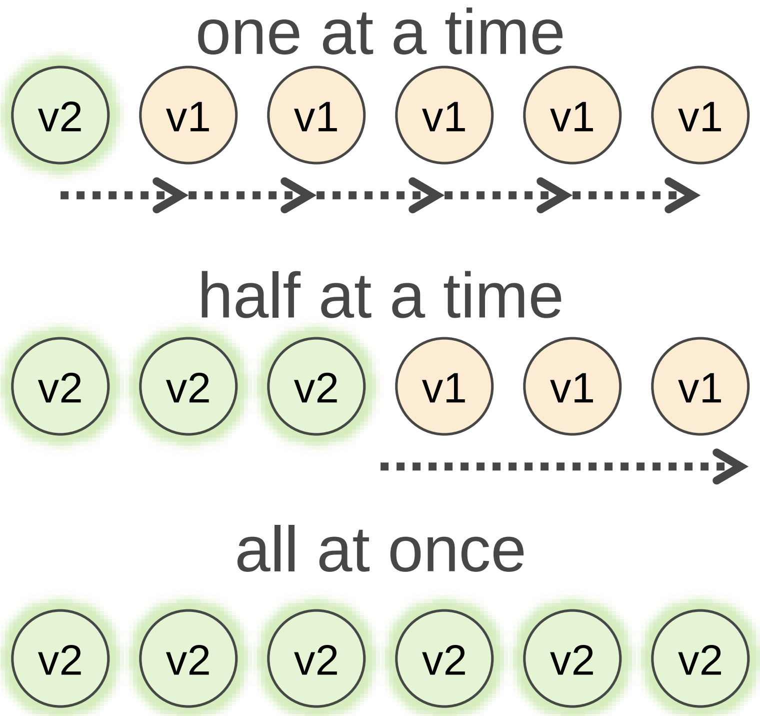
```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html
permissions:
  - object: /var/www/html
    pattern: “*.html”
    owner: root
    group: root
    mode: 755
hooks:
  ApplicationStop:
    - location: scripts/deregister_from_elb.sh
  BeforeInstall:
    - location: scripts/install_dependencies.sh
  ApplicationStart:
    - location: scripts/start_httpd.sh
  ValidateService:
    - location: scripts/test_site.sh
    - location: scripts/register_with_elb.sh
```

- Send application files to one directory and configuration files to another

- Set specific permissions on specific directories & files

- Remove/Add instance to ELB
- Install dependency packages
- Start Apache
- Confirm successful deploy
- More!

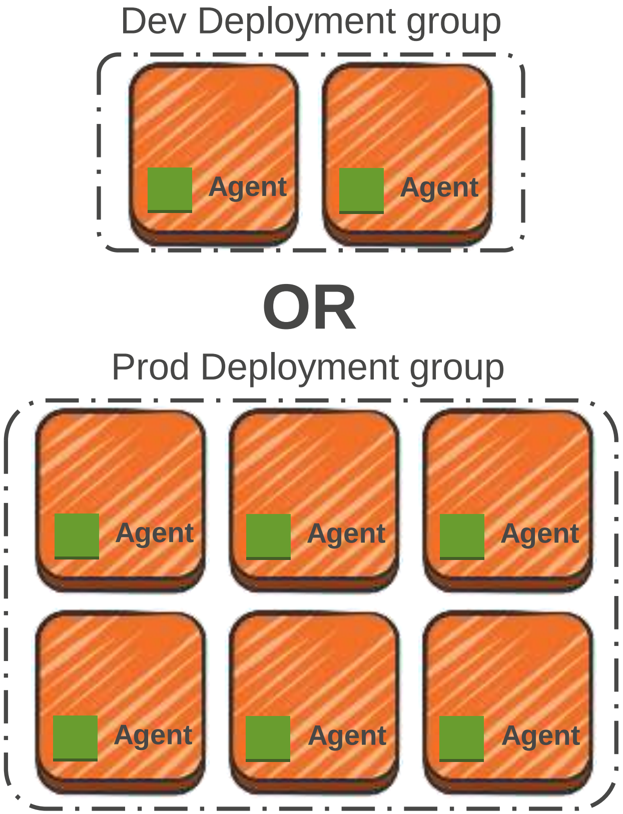
Choose deployment speed & group



one at a time

half at a time

all at once



Dev Deployment group

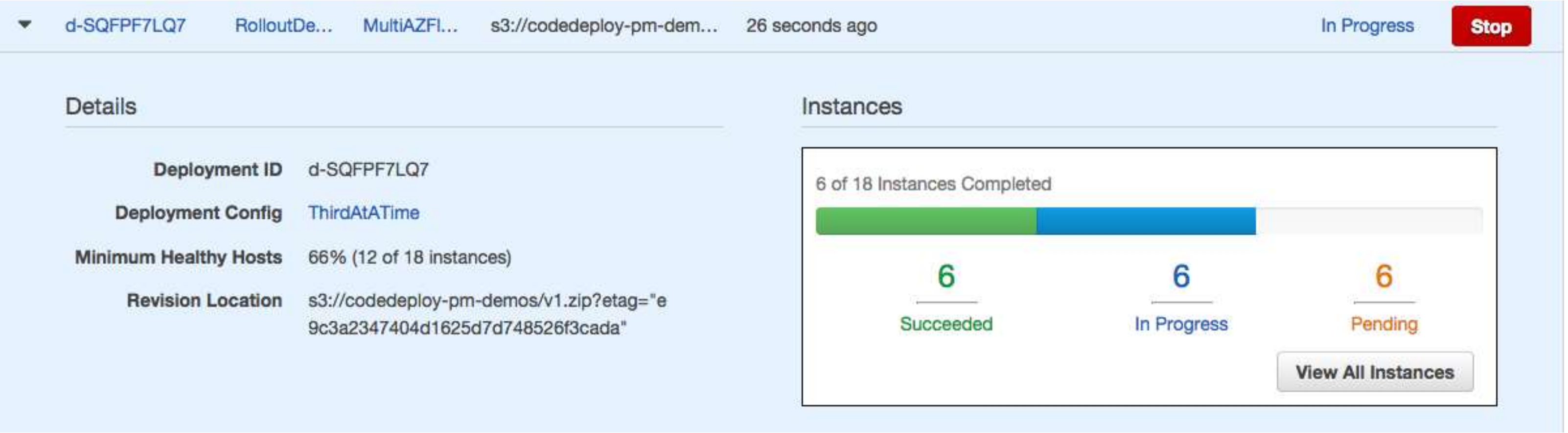
OR

Prod Deployment group

Deploy!

AWS CLI & SDKs
AWS Console
AWS CodePipeline
CI / CD Partners
GitHub

```
aws deploy create-deployment  \
--application-name MyApp  \
--deployment-group-name TargetGroup  \
--s3-location bucket=MyBucket,key=MyApp.zip
```



d-SQFPF7LQ7 RolloutDe... MultiAZFl... s3://codedeploy-pm-dem... 26 seconds ago In Progress

Details

Deployment ID d-SQFPF7LQ7

Deployment Config ThirdAtATime

Minimum Healthy Hosts 66% (12 of 18 instances)

Revision Location s3://codedeploy-pm-demos/v1.zip?etag="e9c3a2347404d1625d7d748526f3cada"

Instances

6 of 18 Instances Completed

6 Succeeded

6 In Progress

6 Pending

Your infrastructure assembly line

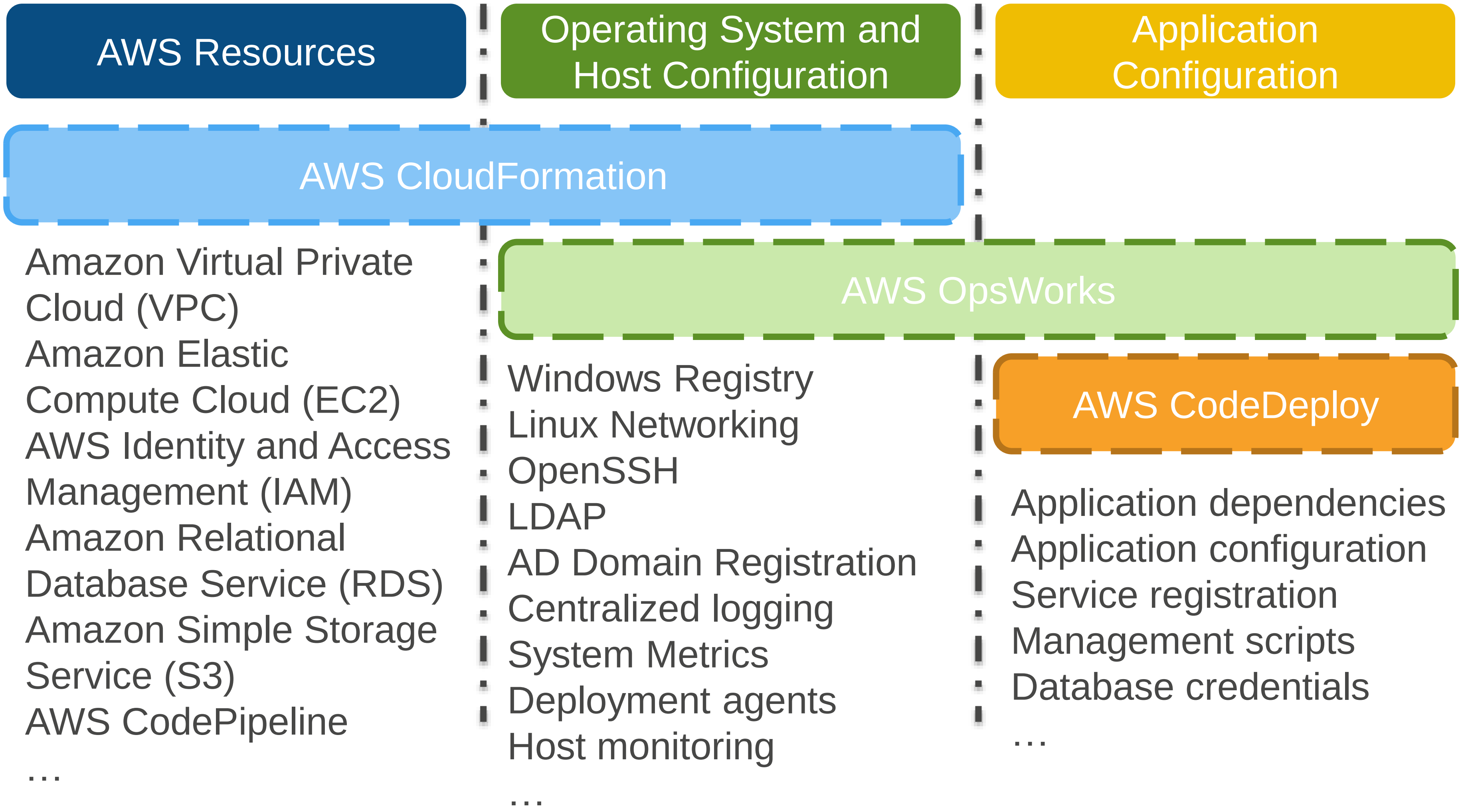
https://secure.flickr.com/photos/spenceyc/7481166880

CloudFormation + OpsWorks + CodeDeploy!

- Create/update/manage AWS resources configuration and properties with **CloudFormation**
  - You can configure OpsWorks and CodeDeploy via CloudFormation
- Use **OpsWorks** for ongoing tweaks to software/configuration of host based applications and the operating system
  - You can configure and deploy CodeDeploy’s agent with OpsWorks
- Use **CodeDeploy** to deploy your applications and their configurations

CloudFormation + OpsWorks + CodeDeploy!

- Your CloudFormation **templates** and Chef **cookbooks** should go in their own **repositories**
- Include **appspec.yml** file and related **scripts** in your application's code **repositories**
- **Every commit** should cause an execution of your **continuous delivery pipeline** to **lint, validate and/or test**
- Use each related service's CLI/console/APIs to update or deploy as necessary



AWS Resources

Operating System and Host Configuration

Application Configuration

AWS CloudFormation

AWS OpsWorks

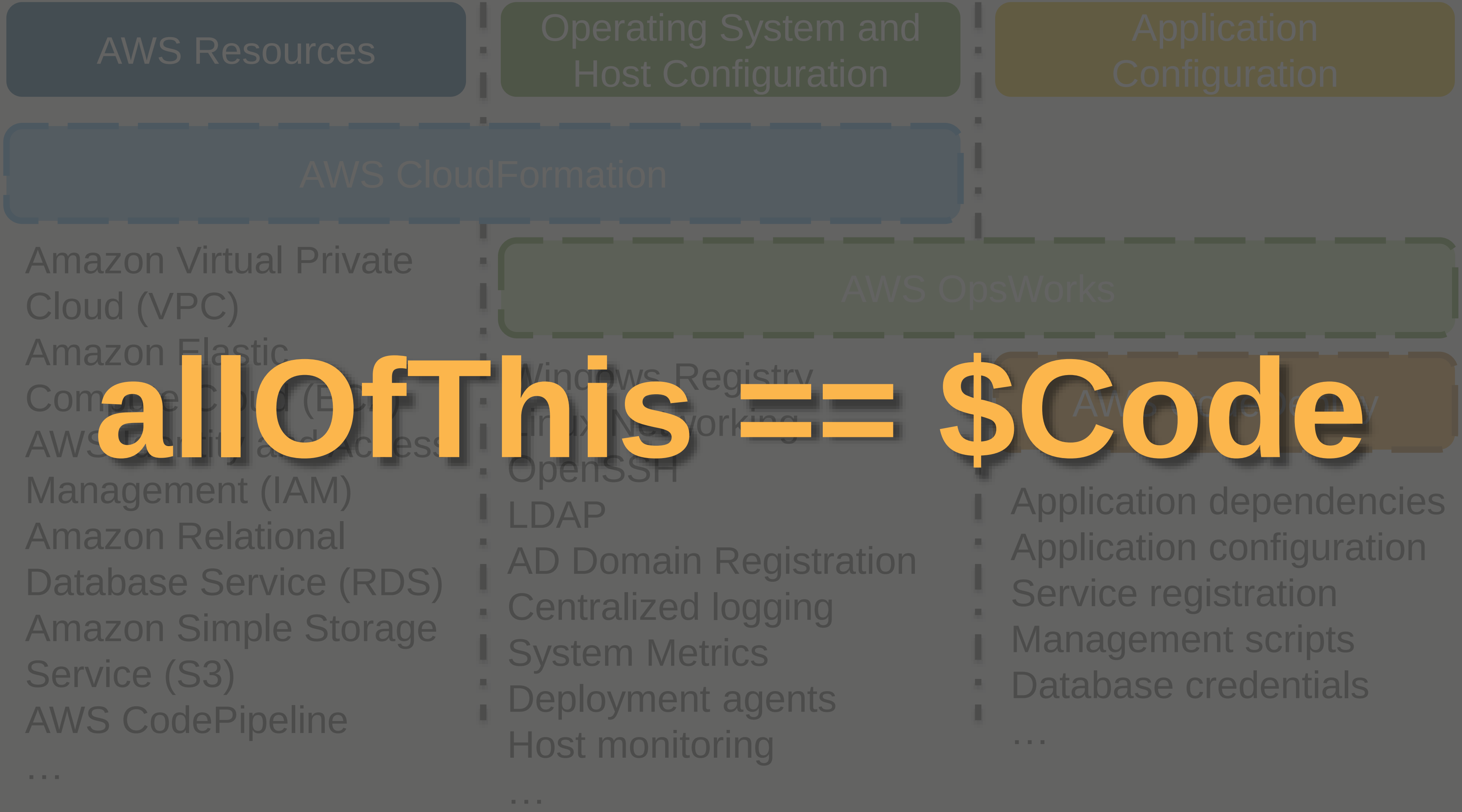
AWS CodeDeploy

Amazon Virtual Private Cloud (VPC)
Amazon Elastic Compute Cloud (EC2)
AWS Identity and Access Management (IAM)
Amazon Relational Database Service (RDS)
Amazon Simple Storage Service (S3)
AWS CodePipeline
…

Windows Registry
Linux Networking
OpenSSH
LDAP
AD Domain Registration
Centralized logging
System Metrics
Deployment agents
Host monitoring
…

Application dependencies
Application configuration
Service registration
Management scripts
Database credentials
…

allOfThis == $Code



AWS Resources

Operating System and Host Configuration

Application Configuration

AWS CloudFormation

AWS OpsWorks

AWS CodeDeploy

Amazon Virtual Private Cloud (VPC)
Amazon Elastic Compute Cloud (EC2)
AWS Identity and Access Management (IAM)
Amazon Relational Database Service (RDS)
Amazon Simple Storage Service (S3)
AWS CodePipeline
…

Windows Registry
Linux Networking
OpenSSH
LDAP
AD Domain Registration
Centralized logging
System Metrics
Deployment agents
Host monitoring
…

Application dependencies
Application configuration
Service registration
Management scripts
Database credentials
…

Who is iTMethods?

iTMethods.

- Founded in 2005 by Paul Goldman
  - ✓ Founder of Arqana Technologies. Built from 0 to $75M. Acquired by TELUS (NYSE: TU)
- Customer Centric Partnership Approach
- Helping organizations execute their Digital Strategies
  - ✓ Deploy new, refactor and optimize existing applications on AWS
  - ✓ Streamline outcomes with Automation & DevOps
  - ✓ Certified Deploy/Migrate, Automate & Managed AWS Services
- Strategic Partnership with Amazon Web Services (AWS)
  - ✓ Certified AWS Managed Service Partner
  - ✓ Advanced Consulting Partner
  - ✓ Authorized Channel Reseller
  - ✓ Authorized Government Reseller, AWS Direct Connect Partner

amazon web services™

User Story – An EHR Healthcare Provider

Customer Requirements:

Rapid Transition from on-premises infrastructure into AWS while leveraging HIPAA-eligible services

- On-premises infrastructure consists of:
  - 5+ critical application workloads
  - 30+ bare-metal Microsoft SQL Servers
  - Managed virtualized platform
- Managed entirely on a per-server basis (no fleet-management)
- Not currently using any automated software deployment mechanisms

- AWS Implementation
  - Designed for HIPAA compliance using secure design principles
- Deployed environment using Infrastructure as Code (Ansible/CloudFormation)
  - Leveraged HIPAA-eligible services, employing encryption and dedicated tenancy while enforcing strict change management processes
- CloudFormation invoked by Ansible to deploy redundant Microsoft SQL Enterprise Clusters
- Deployed entire Windows environment, including Active Directory and supporting components using Ansible

Value Provided by iTMethods

Achieved full infrastructure implementation and automation

- Leveraged Ansible and Cloudformation
- Full stack management of infrastructure and applications

Consolidated entire Infrastructure as Code deployment and management into single package

- Reduced the overhead for ongoing maintenance and changes
- Modular design of IaC package to enable distributed team-based efforts

Provide DevOps automation expertise for automated service deployment & management

But wait, there’s more!

Resources to learn more:

- CloudFormation
  - https://aws.amazon.com/cloudformation/
  - https://aws.amazon.com/documentation/cloudformation/
  - https://aws.amazon.com/cloudformation/aws-cloudformation-templates/
- OpsWorks
  - https://aws.amazon.com/opsworks/
  - https://aws.amazon.com/documentation/opsworks/
  - https://github.com/aws/opsworks-cookbooks
- CodeDeploy
  - https://aws.amazon.com/codedeploy/
  - https://aws.amazon.com/documentation/codedeploy/
  - https://github.com/awslabs/aws-codedeploy-samples

AWS

ENTERPRISE

SUMMIT

Thank you!

amazon web services