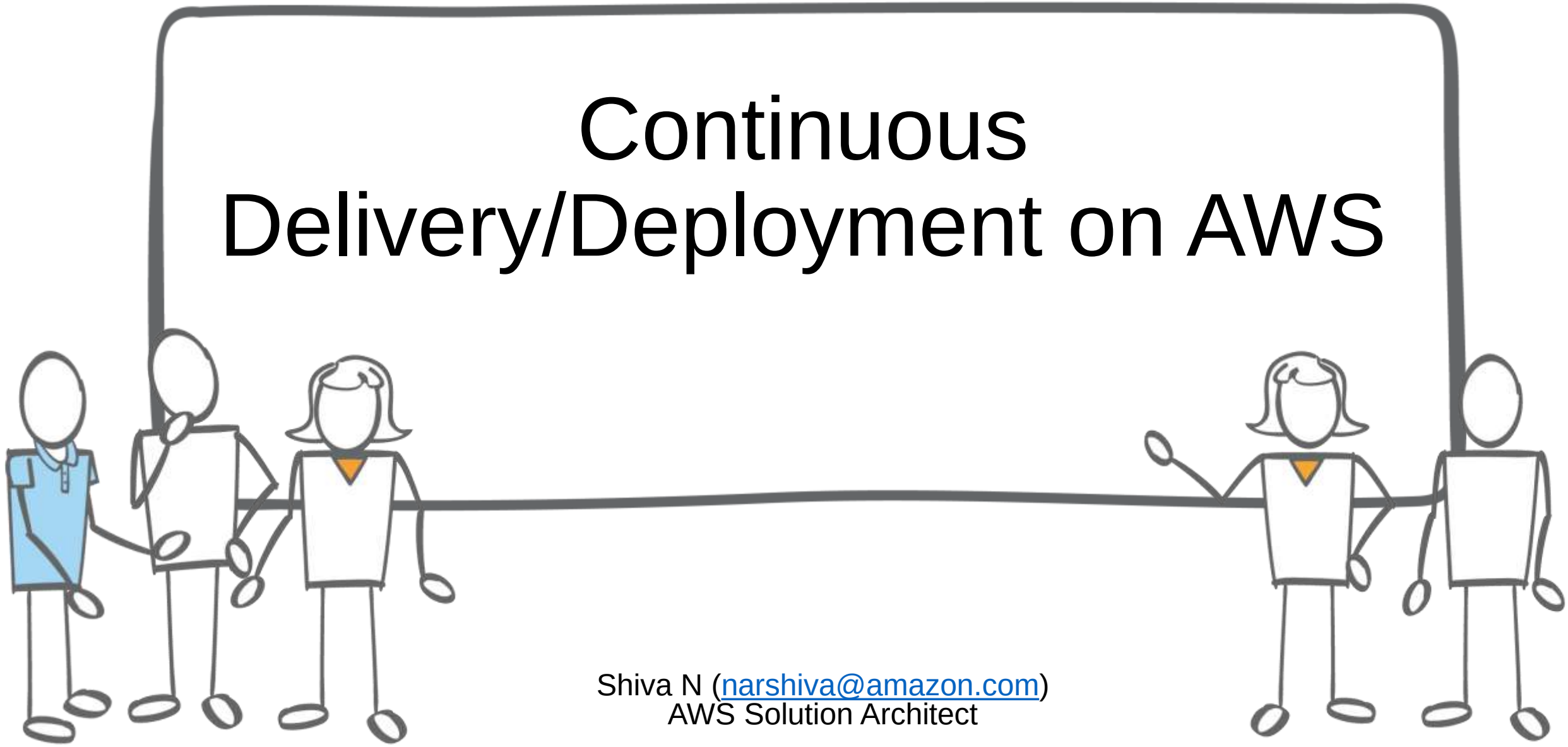
Continuous Delivery/Deployment on AWS



Shiva N (narshiva@amazon.com)
AWS Solution Architect

Version 1.0

DEPLOYMENTS AT AMAZON.COM

~11.6s	~1,079	~10,000	~30,000
Mean time between deployments (weekday)	Max number of deployments in a single hour	Mean number of hosts simultaneously receiving a deployment	Max number of hosts simultaneously receiving a deployment

Agenda

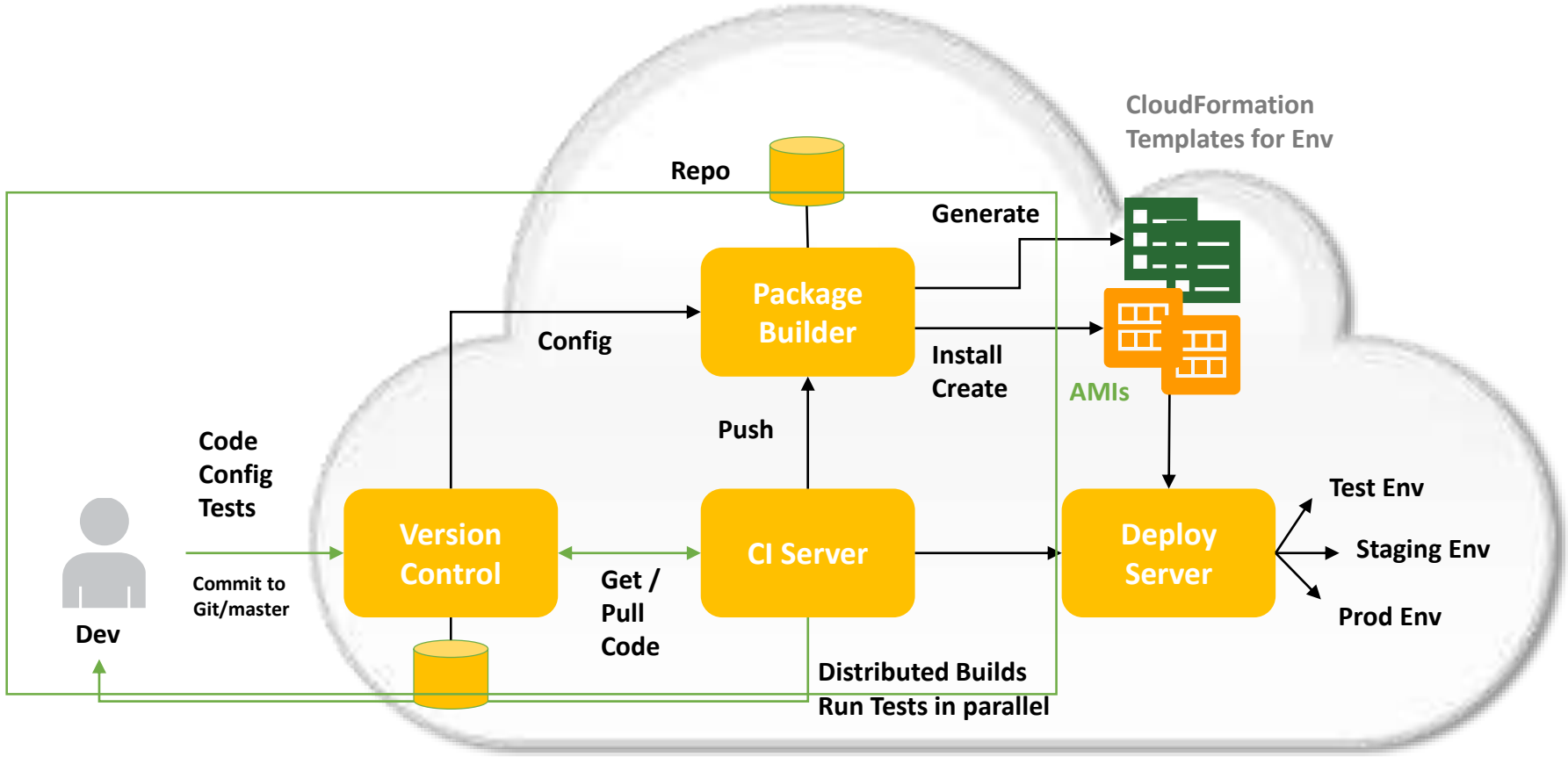
- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - Application Management
    - Elastic BeanStalk
    - Opsworks
    - Cloudformation
    - *EC2 Container Service (ECS)*
  - Application Lifecycle Management
    - *Code Commit*
    - *Code Pipeline*
    - *Code Deploy*

Agenda

- **Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)**
- CD Strategies
- CI-CD on AWS
  - Application Management
    - Elastic BeanStalk
    - Opsworks
    - Cloudformation
    - *EC2 Container Service (ECS)*
  - Application Lifecycle Management
    - *Code Commit*
    - *Code Pipeline*
    - *Code Deploy*

amazon web services

Continuous Integration



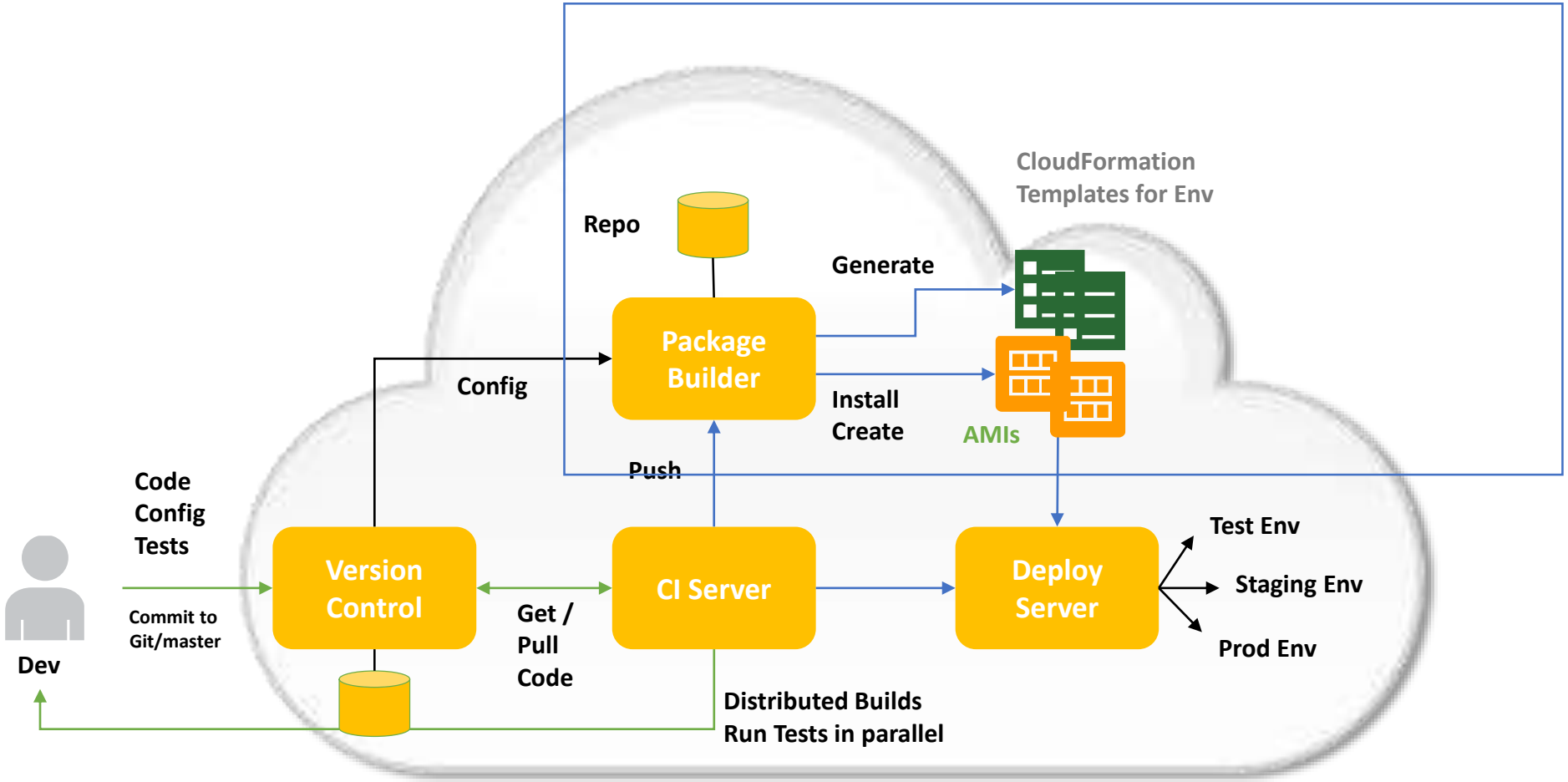
Send Build Report to Dev
Stop everything if build failed

amazon web services

What does CI give us?

- Test driven promotion (of development change)
- Increasing velocity of feedback cycle through iterative change
- Contain change to reduce risk
- Bugs are detected quickly
- Automated testing reduces size of testing effort

Continuous Delivery/Deployment



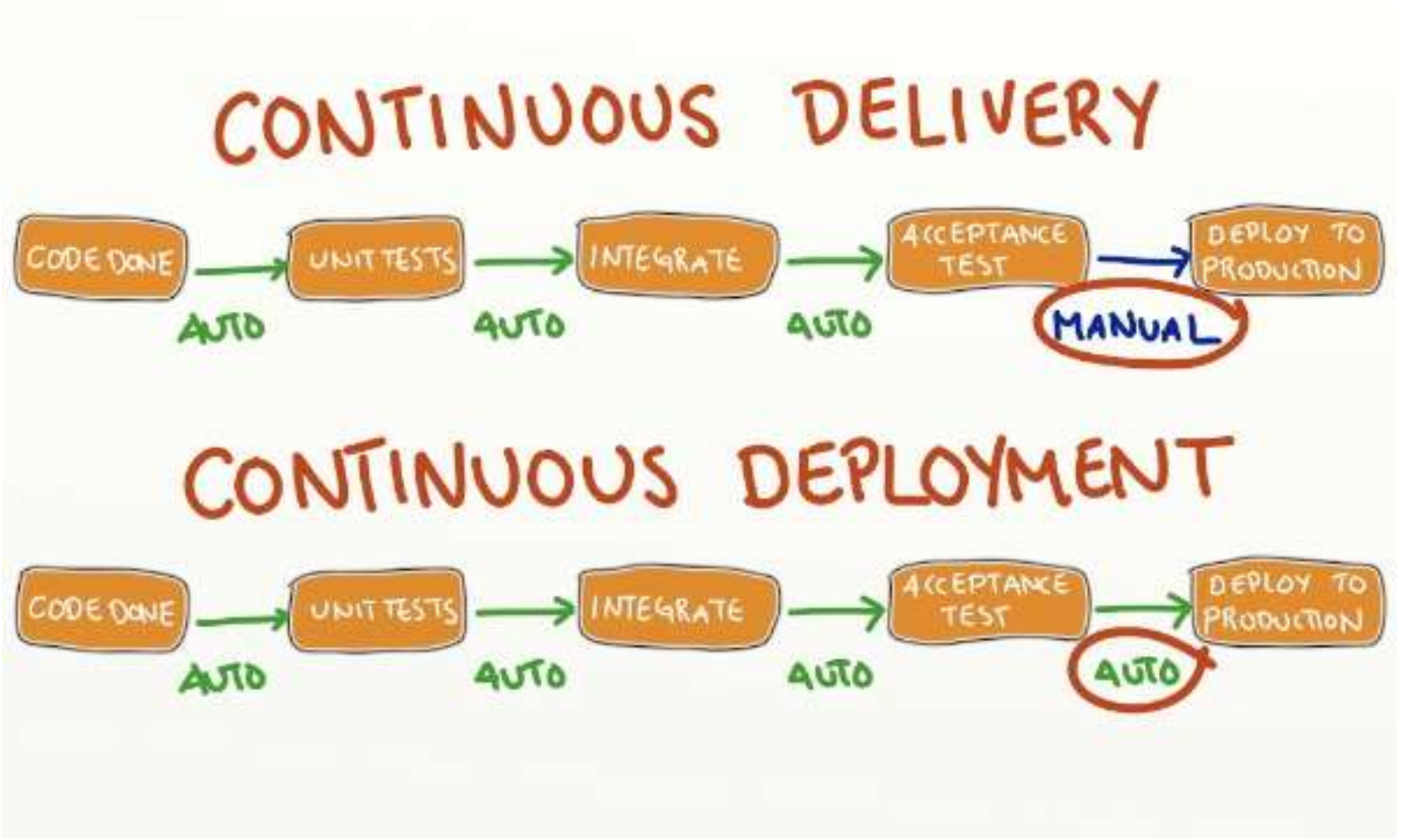
Send Build Report to Dev
Stop everything if build failed

amazon web services

What does CD give us?

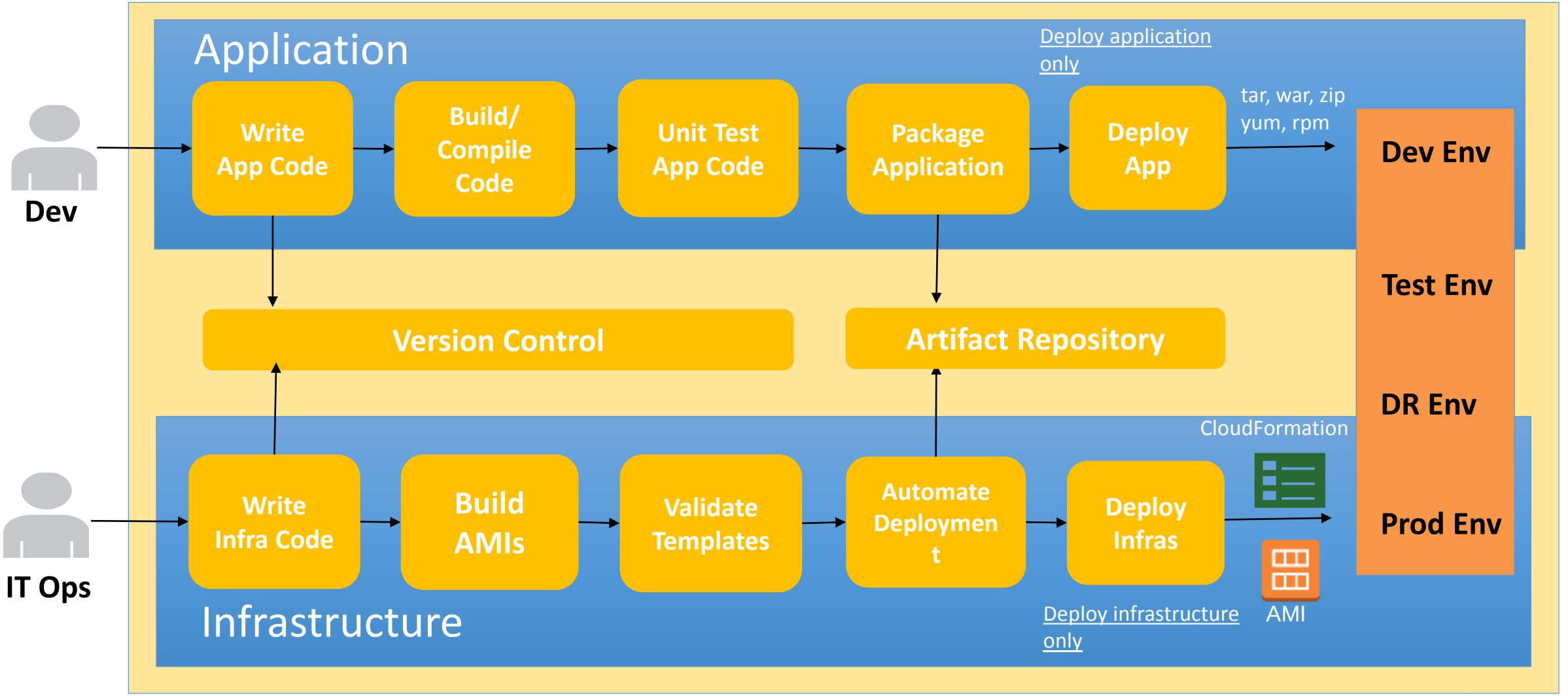
- Automated, repeatable process to push changes to production
- Hardens, de-risks the deployment process
- Immediate feedback from users
- Supports A/B testing or “We test customer reactions to features in production”
- Gives us a breadth of data points across our applications

Continuous Delivery Vs Continuous Deployment

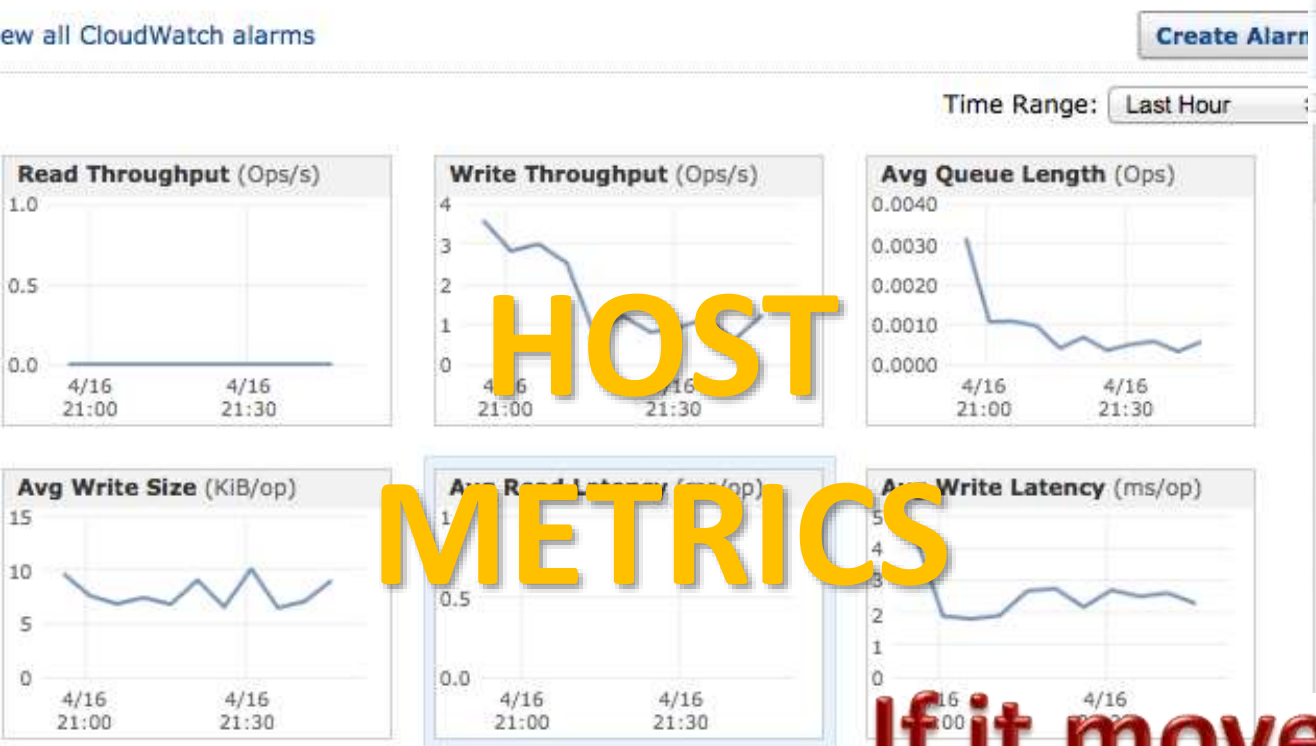


amazon web services

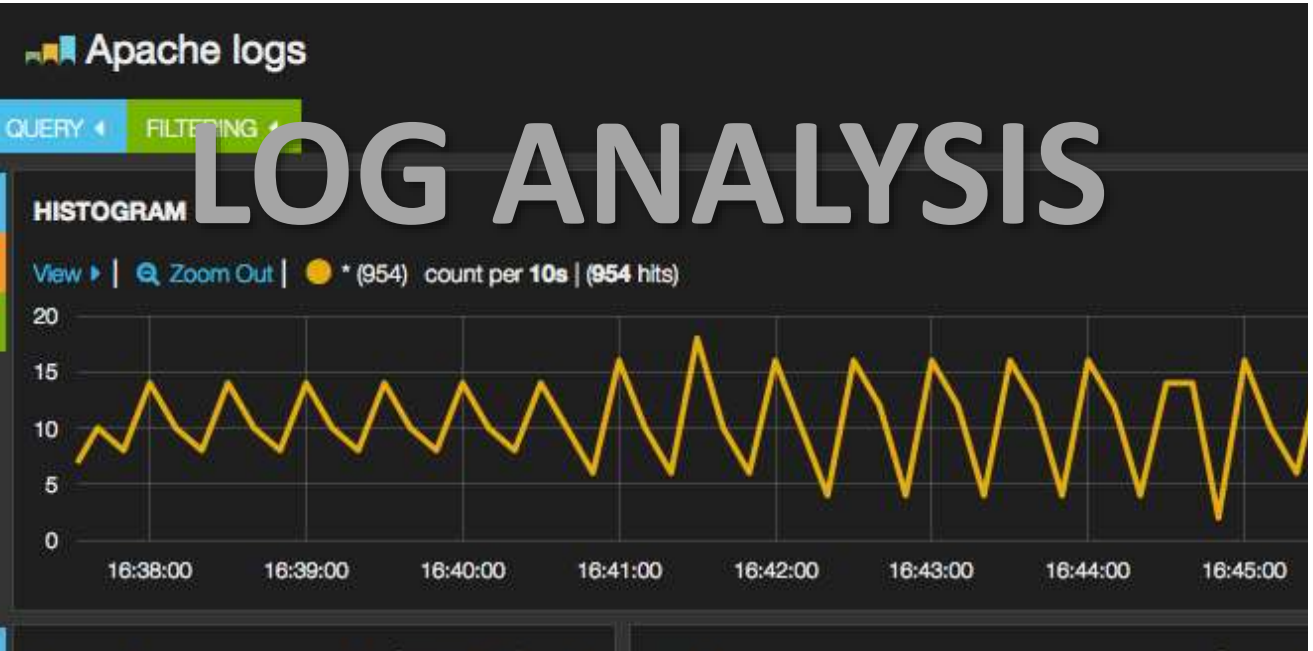
Example CI-CD Pipeline




https://secure.flickr.com/photos/jasoneppink/499531891



If it moves, plot it...



Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- **CD Strategies**
- CI-CD on AWS
  - Application Management
    - Elastic BeanStalk
    - Opsworks
    - Cloudformation
    - *EC2 Container Service (ECS)*
  - Application Lifecycle Management
    - *Code Commit*
    - *Code Pipeline*
    - *Code Deploy*

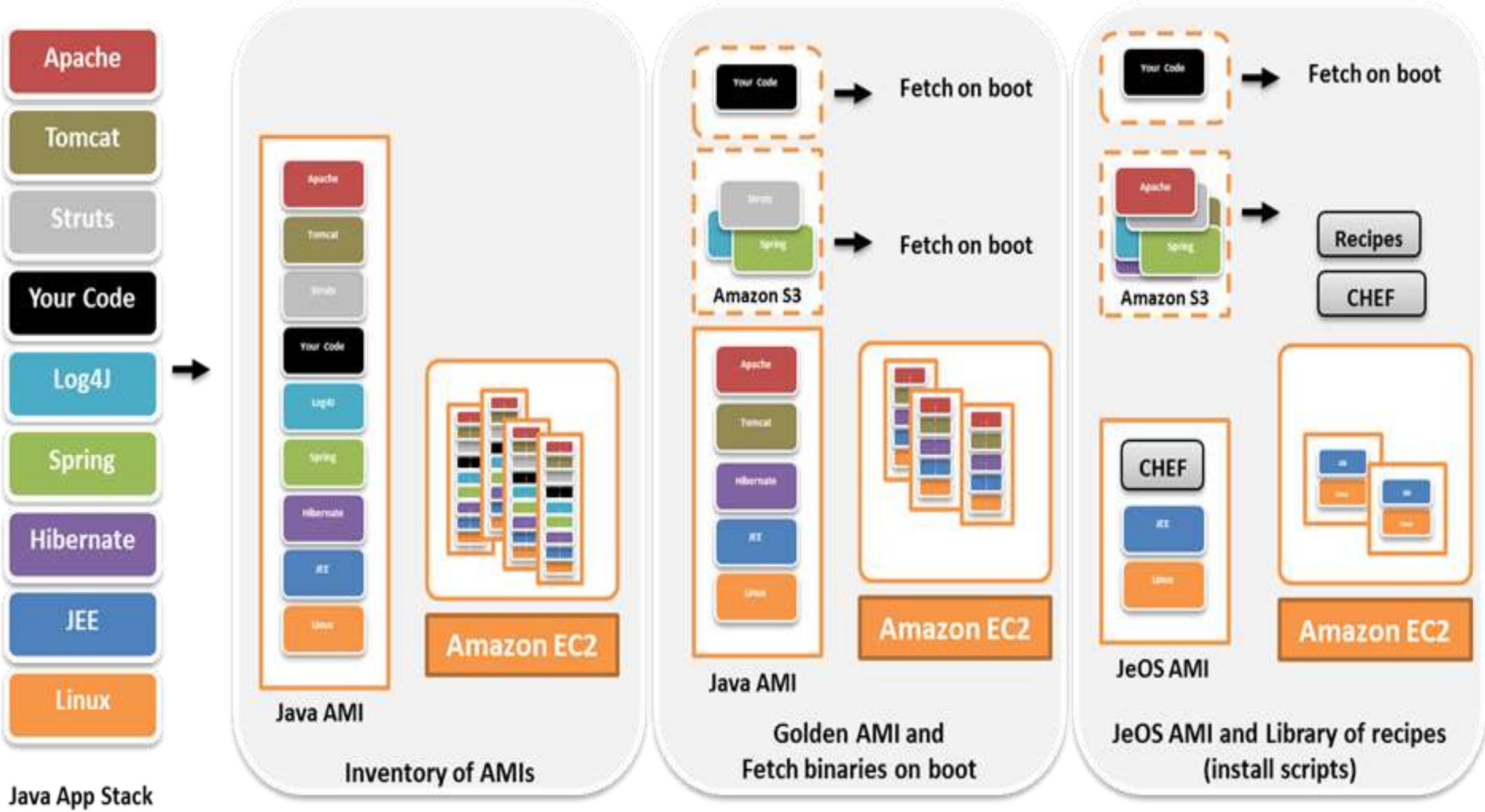
https://secure.flickr.com/photos/simononly/15386966677

Delivery approaches

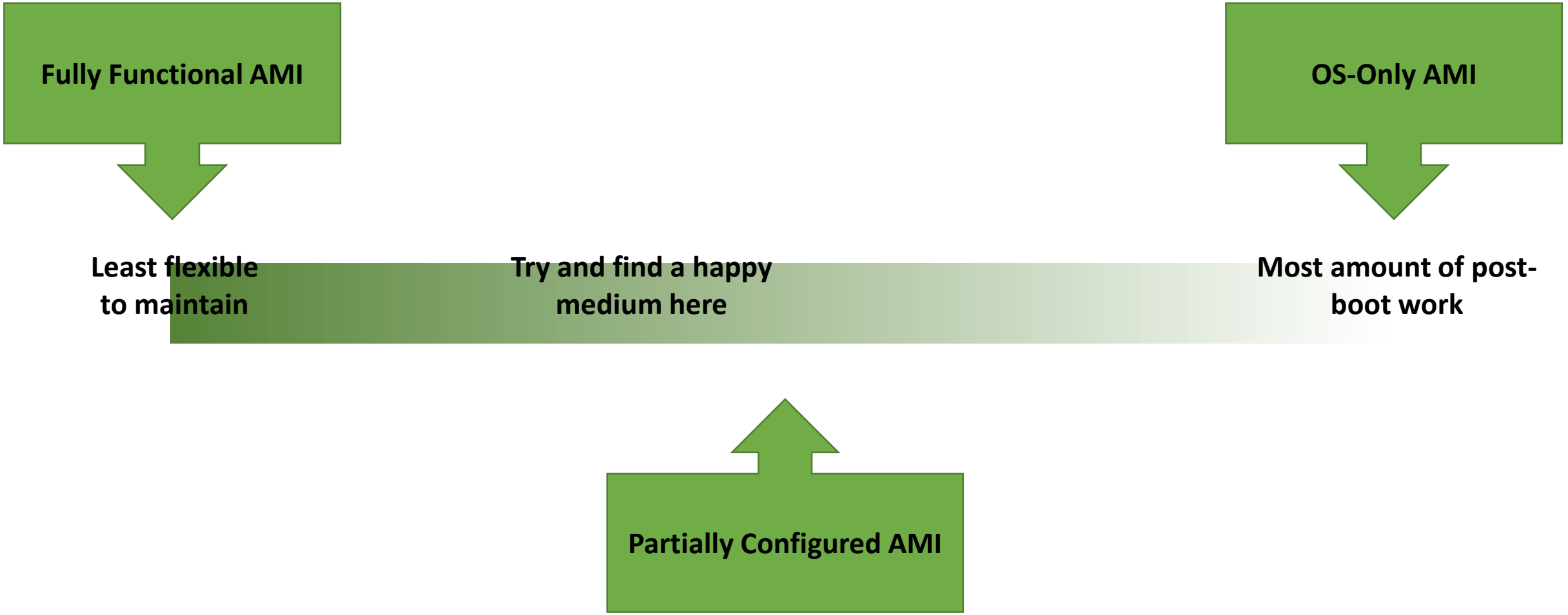
- How are we going to deliver our code?
  - File shipping:
    - Binaries (.rpm, .msi. .exe, .deb, .conf…)
  - As an AMI:
    - Bundle one or more of the above into an AMI
- Which method do you choose?
  - How fast do we need to do this?
  - Across how many instances?
  - How do we roll back (or forward)?

amazon web services

AMI Deployment Method - Building



Delivery approaches…

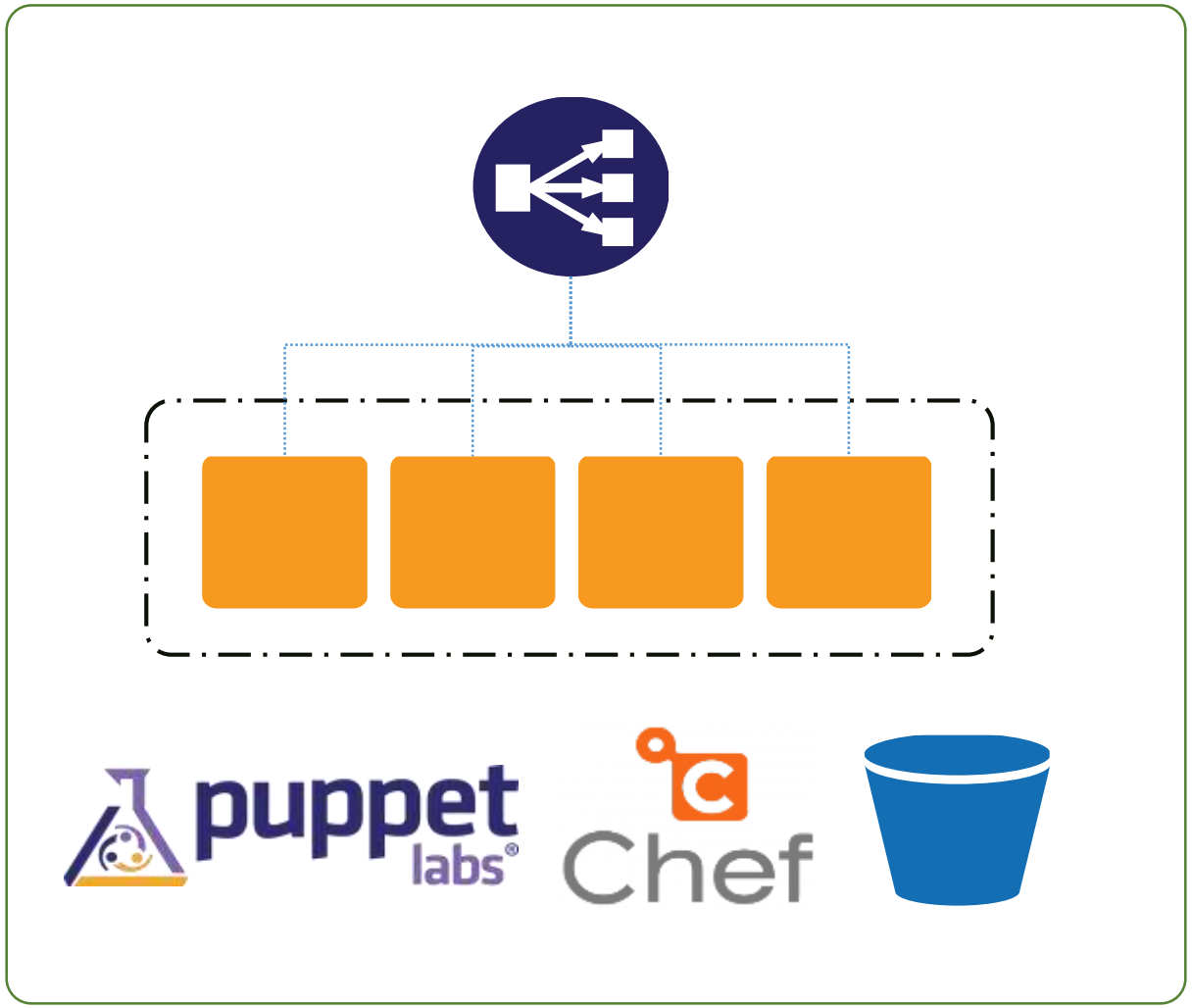


amazon web services

Deployment approaches

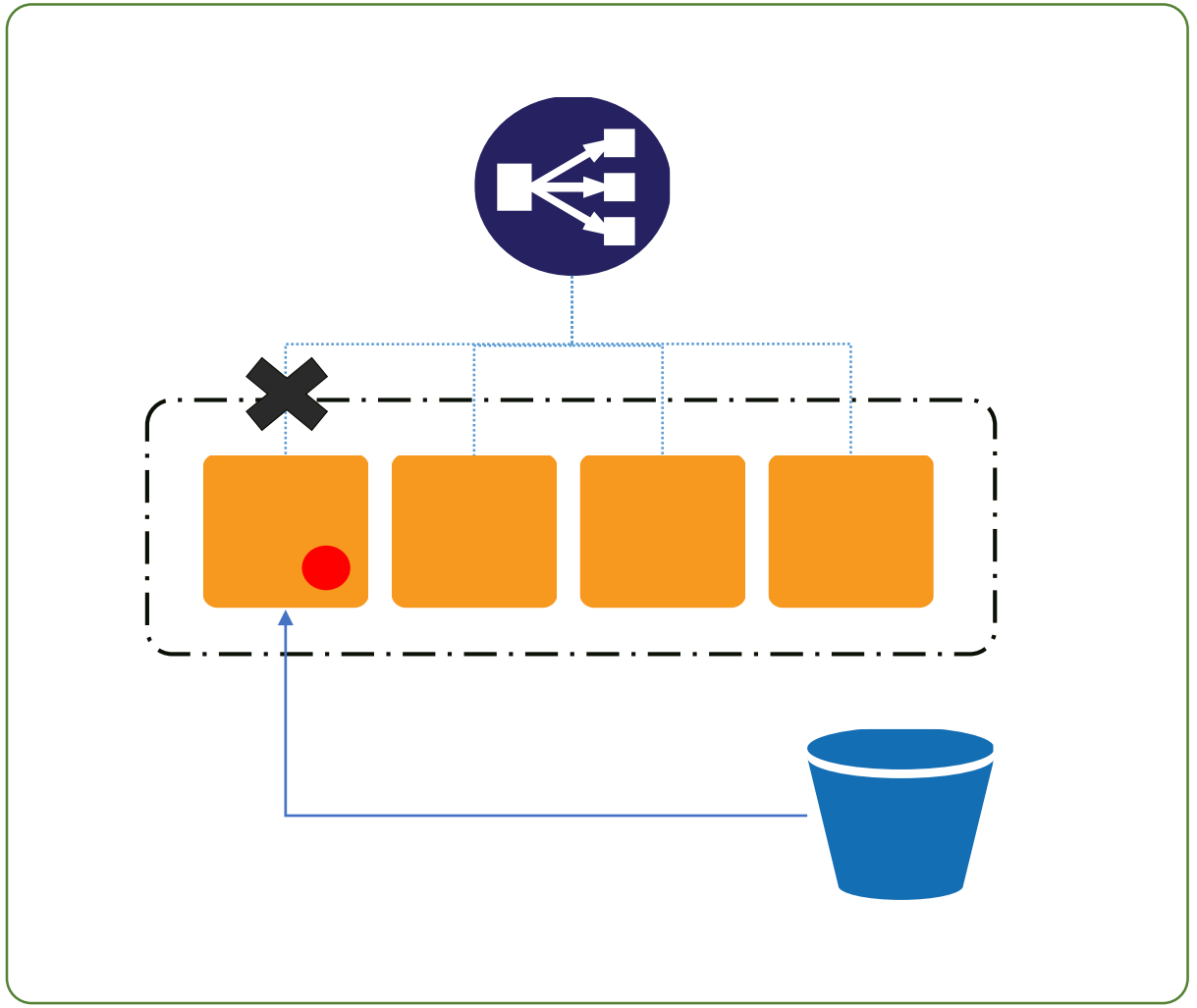
- Deploy in place
  - Deploy all at once (Service outage)
  - Rolling updates
- Blue-Green Deployment
  - Discrete environment
    - Multiple environments from branches
    - Support A/B testing
    - “Rolling DNS”
- Alternate Blue-Green (Red-Black?) deployment
    - Alternate auto scaling group
    - Avoid messing with DNS

Deploy in place – Rolling update



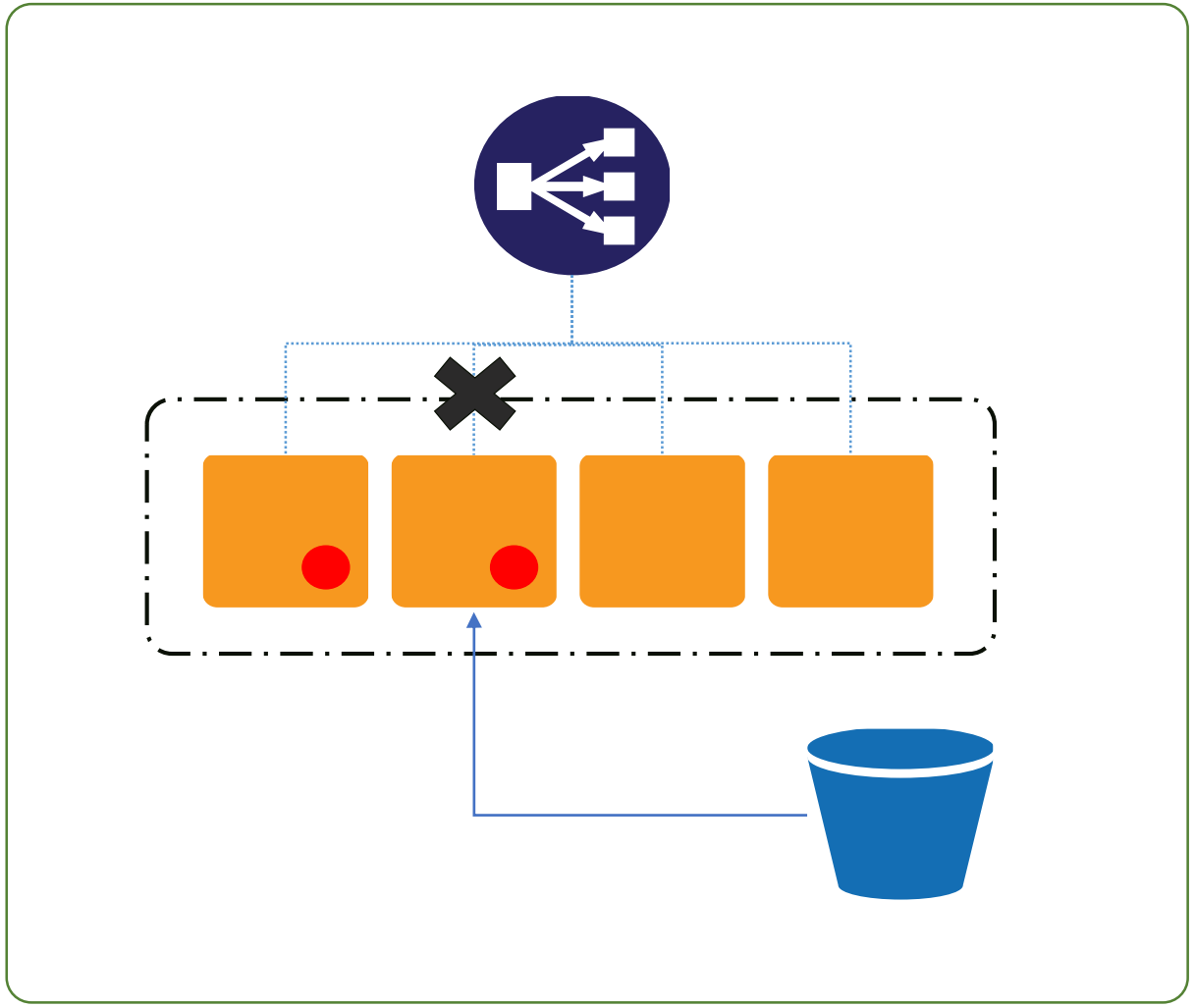
amazon web services

Deploy in place – Rolling update



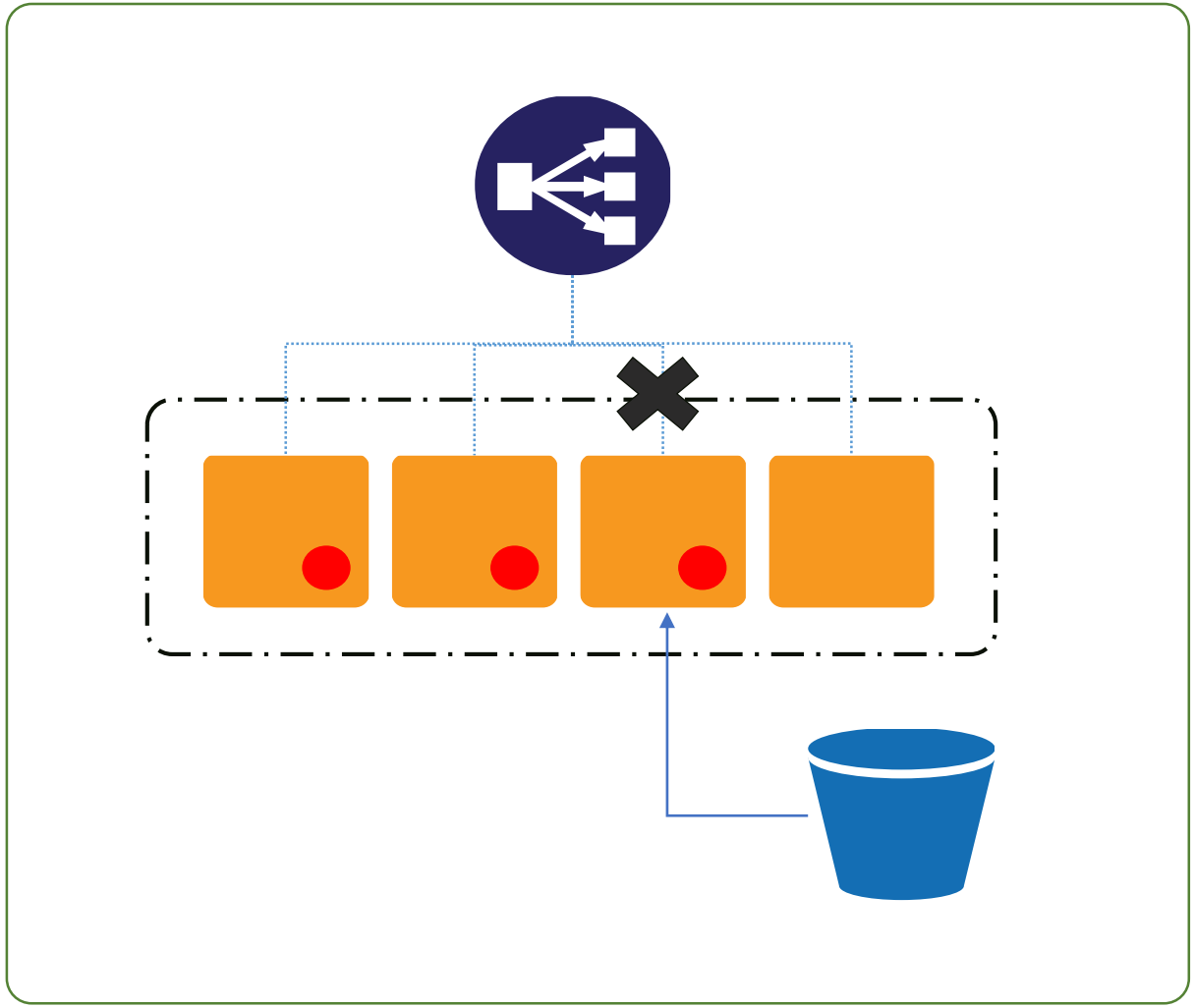
amazon web services

Deploy in place – Rolling update



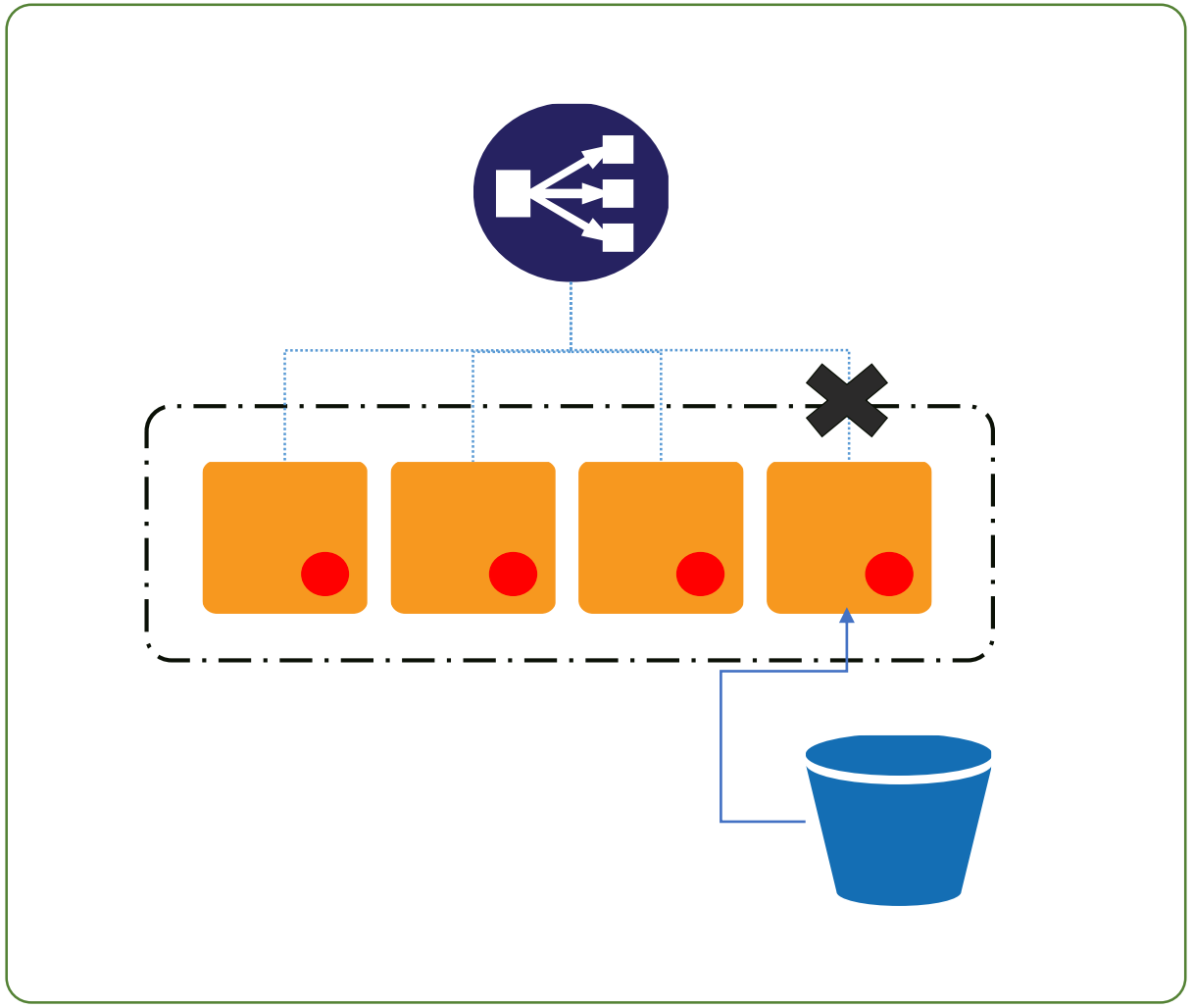
amazon web services

Deploy in place – Rolling update



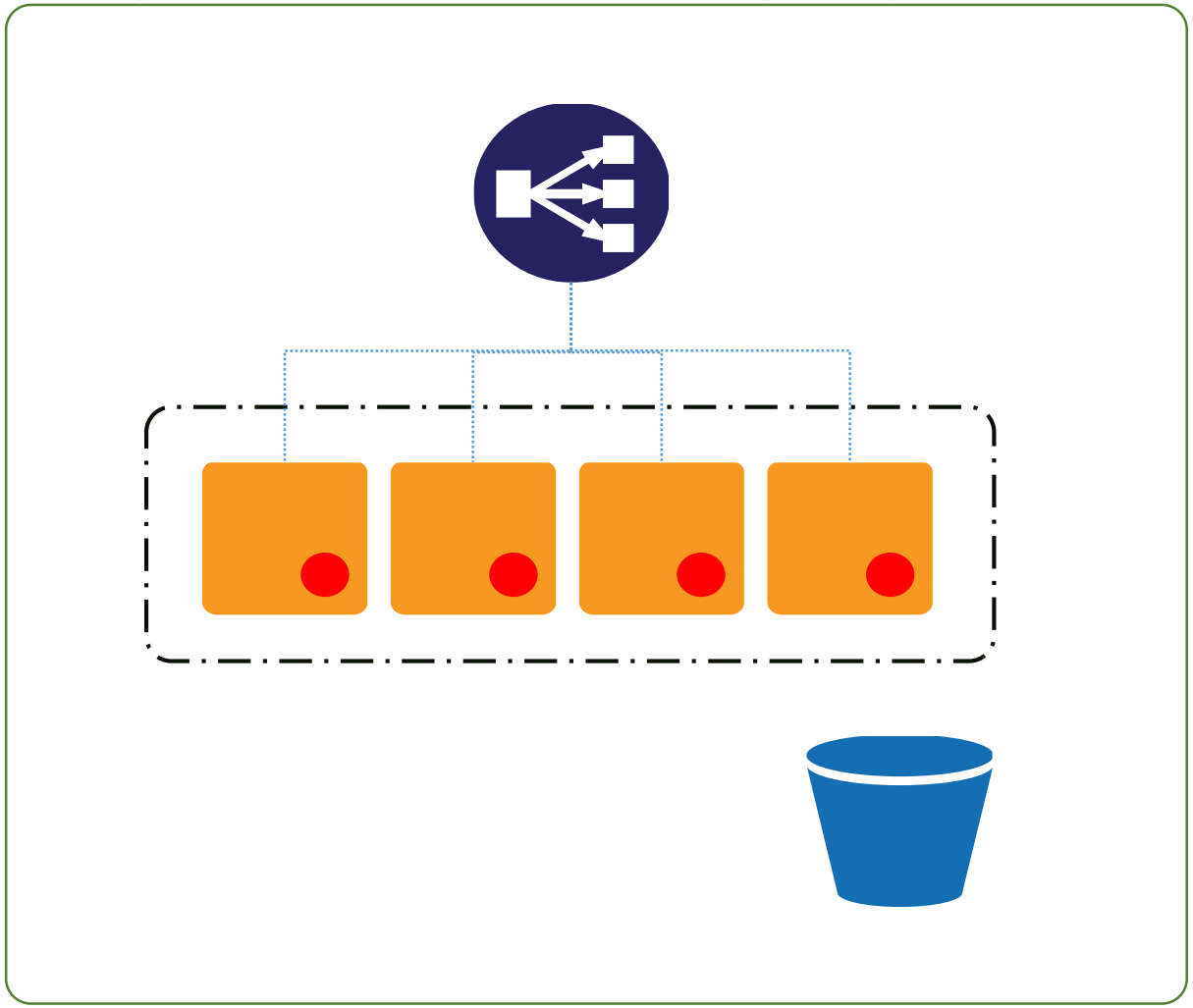
amazon web services

Deploy in place – Rolling update



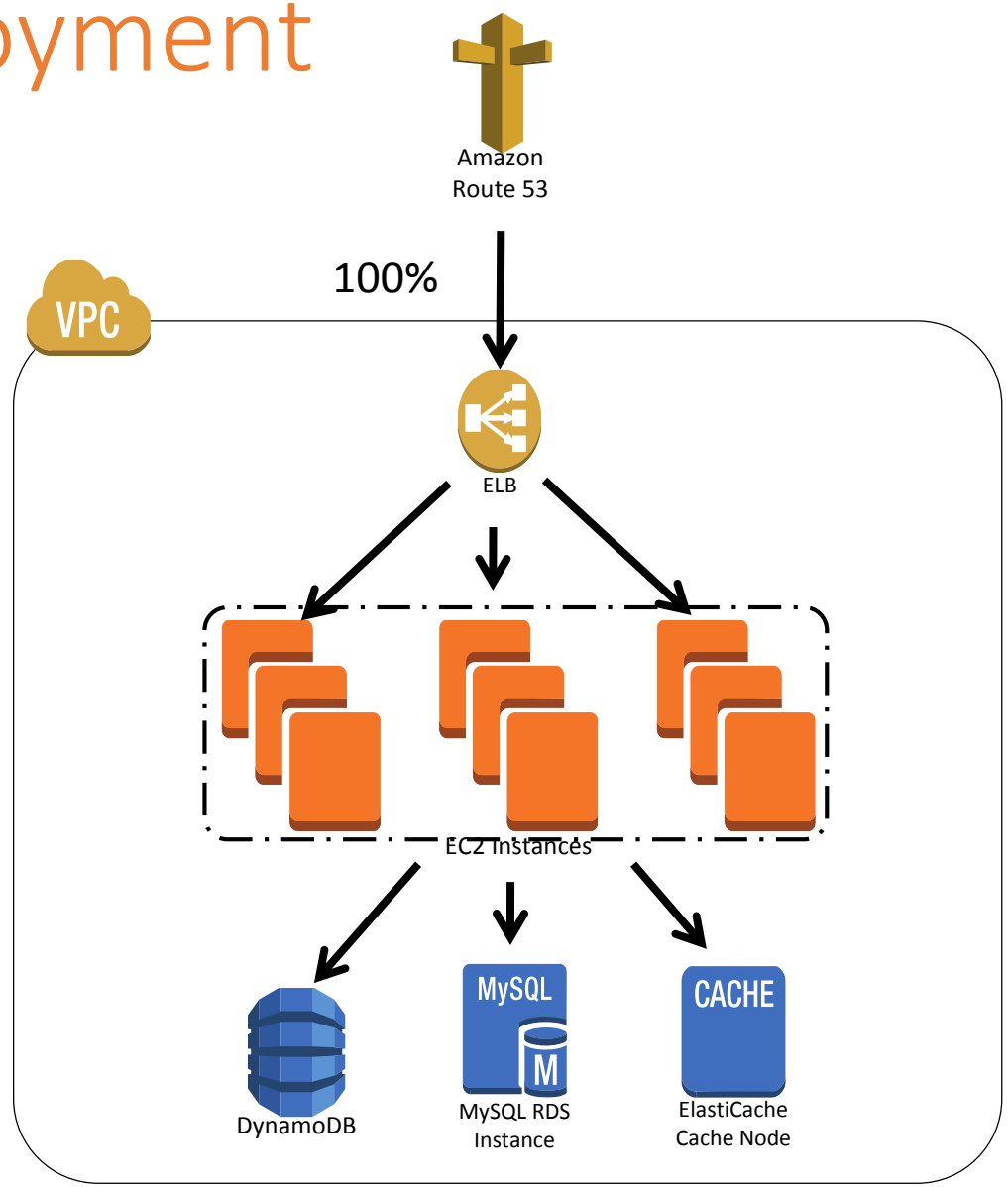
amazon web services

Deploy in place – Rolling update



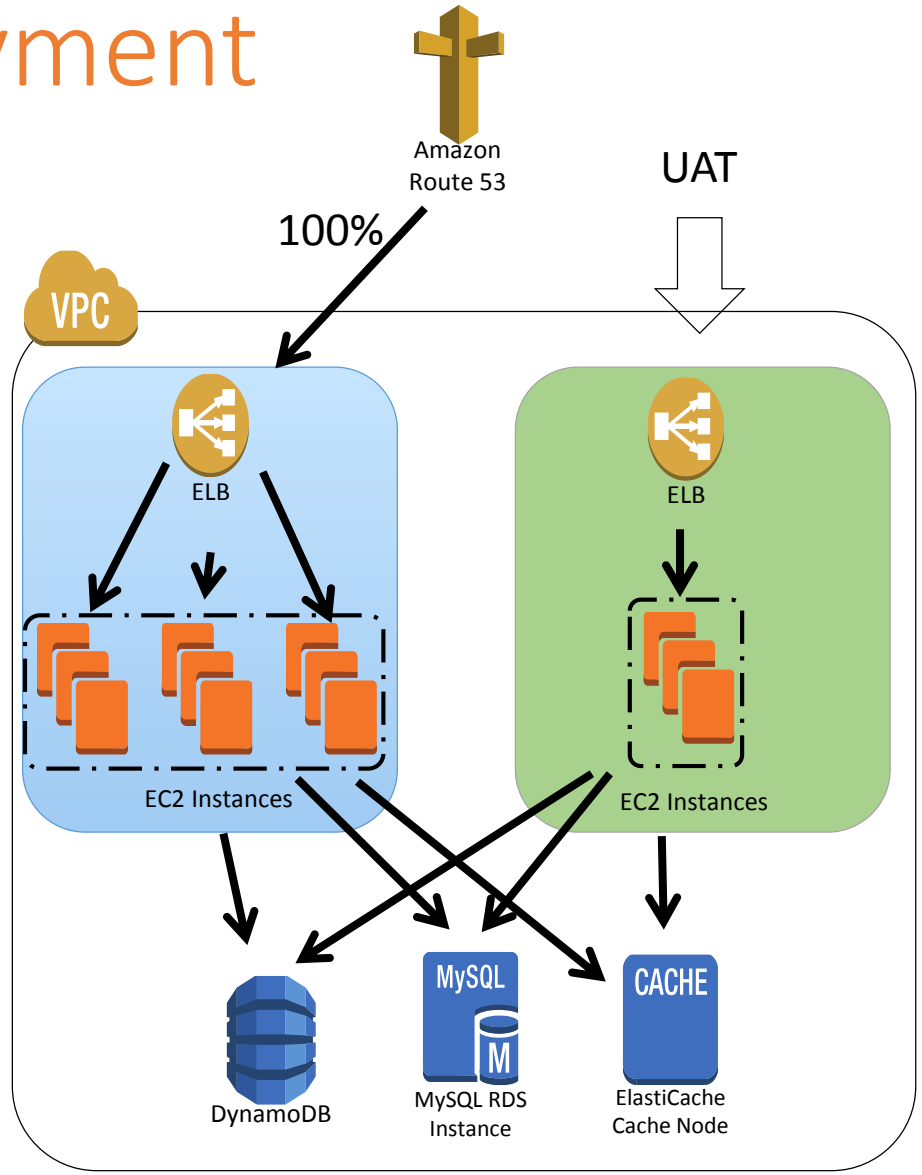
amazon web services

Blue-Green deployment



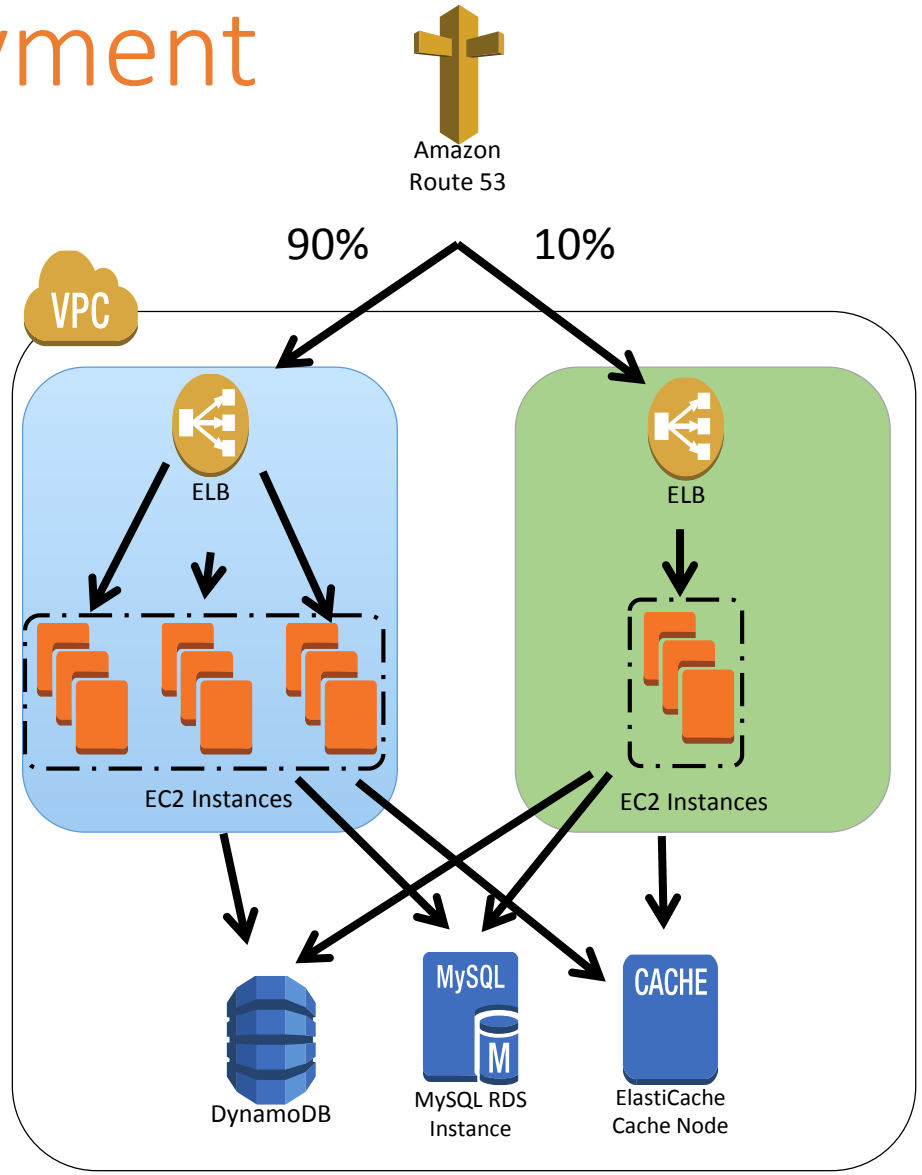
amazon web services

Blue-Green deployment



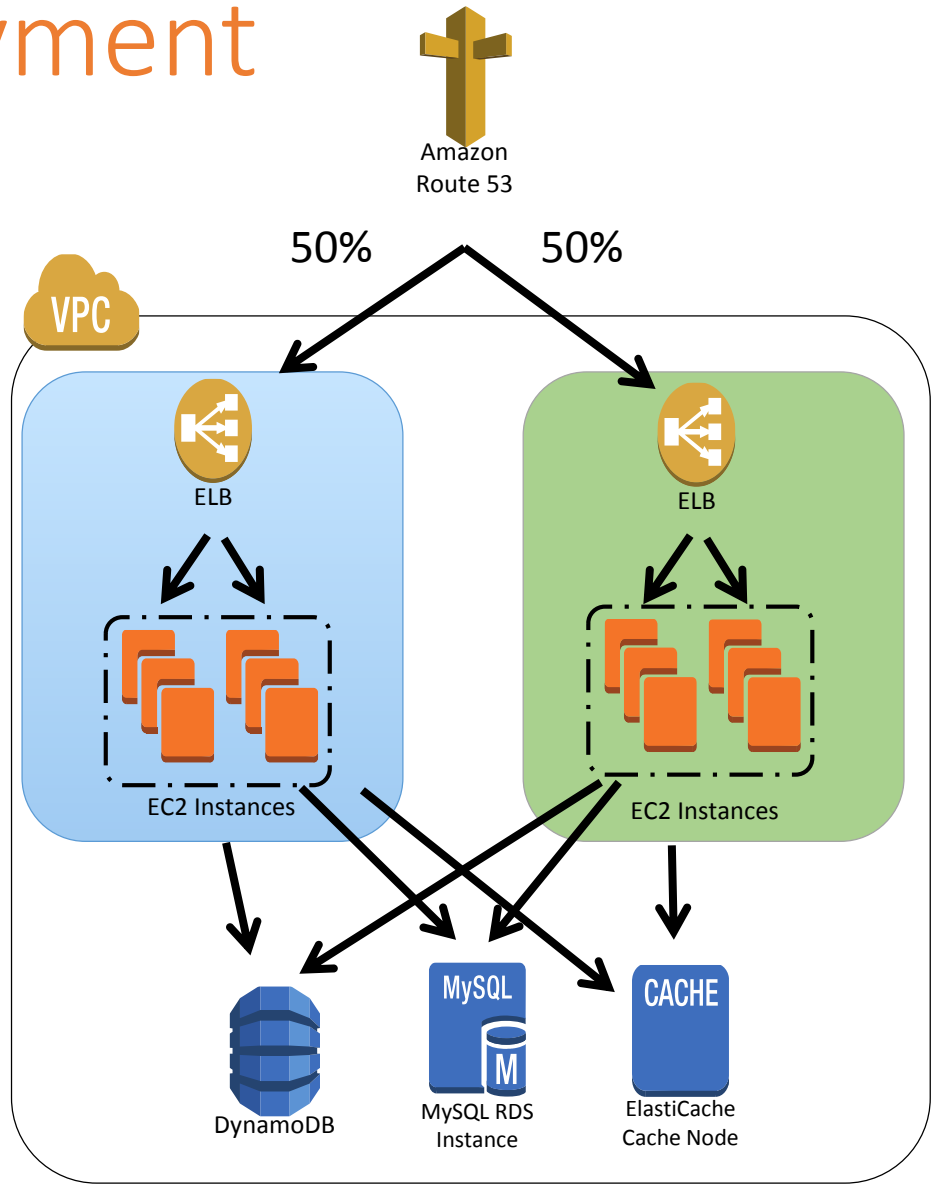
amazon web services

Blue-Green deployment



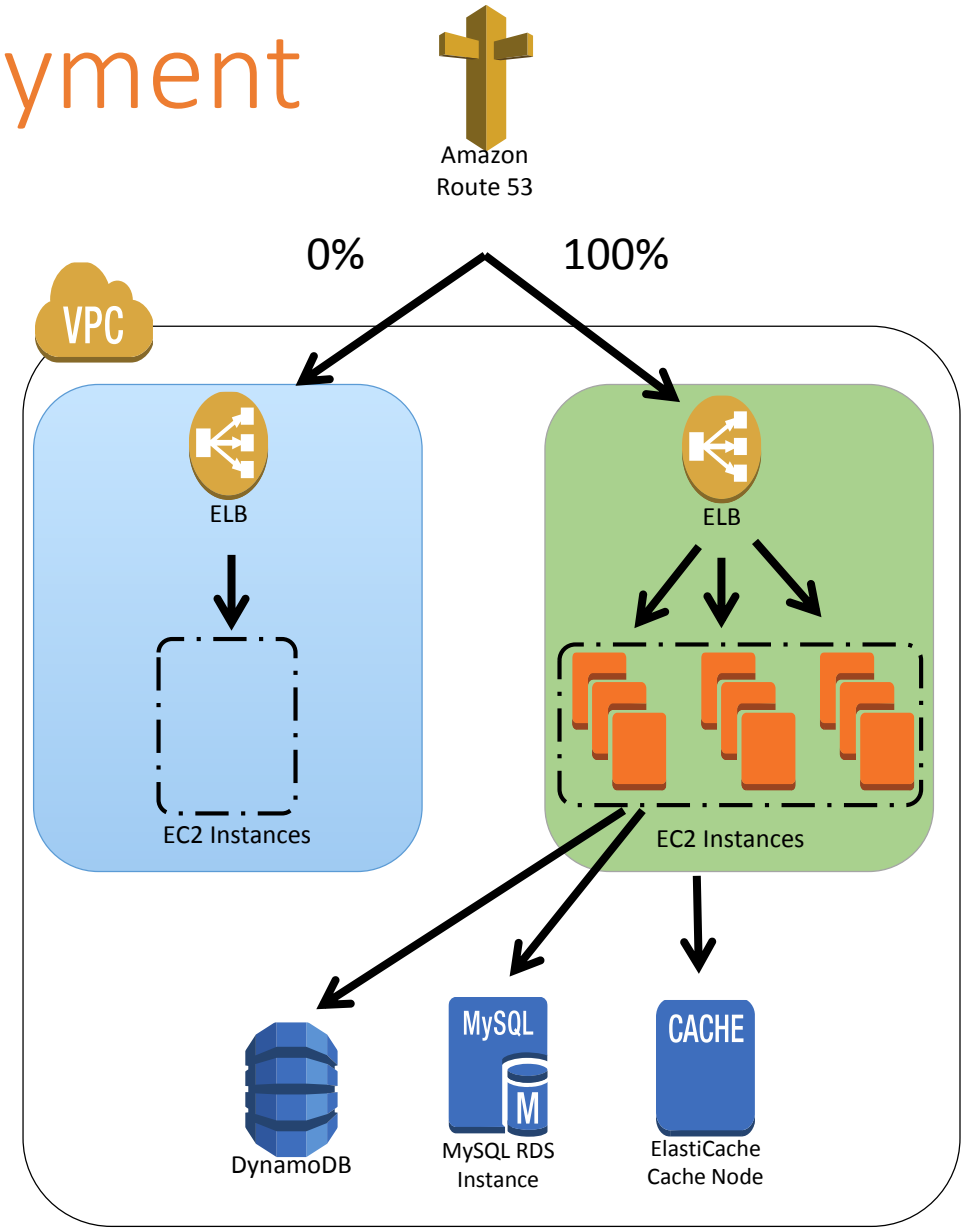
amazon web services

Blue-Green deployment



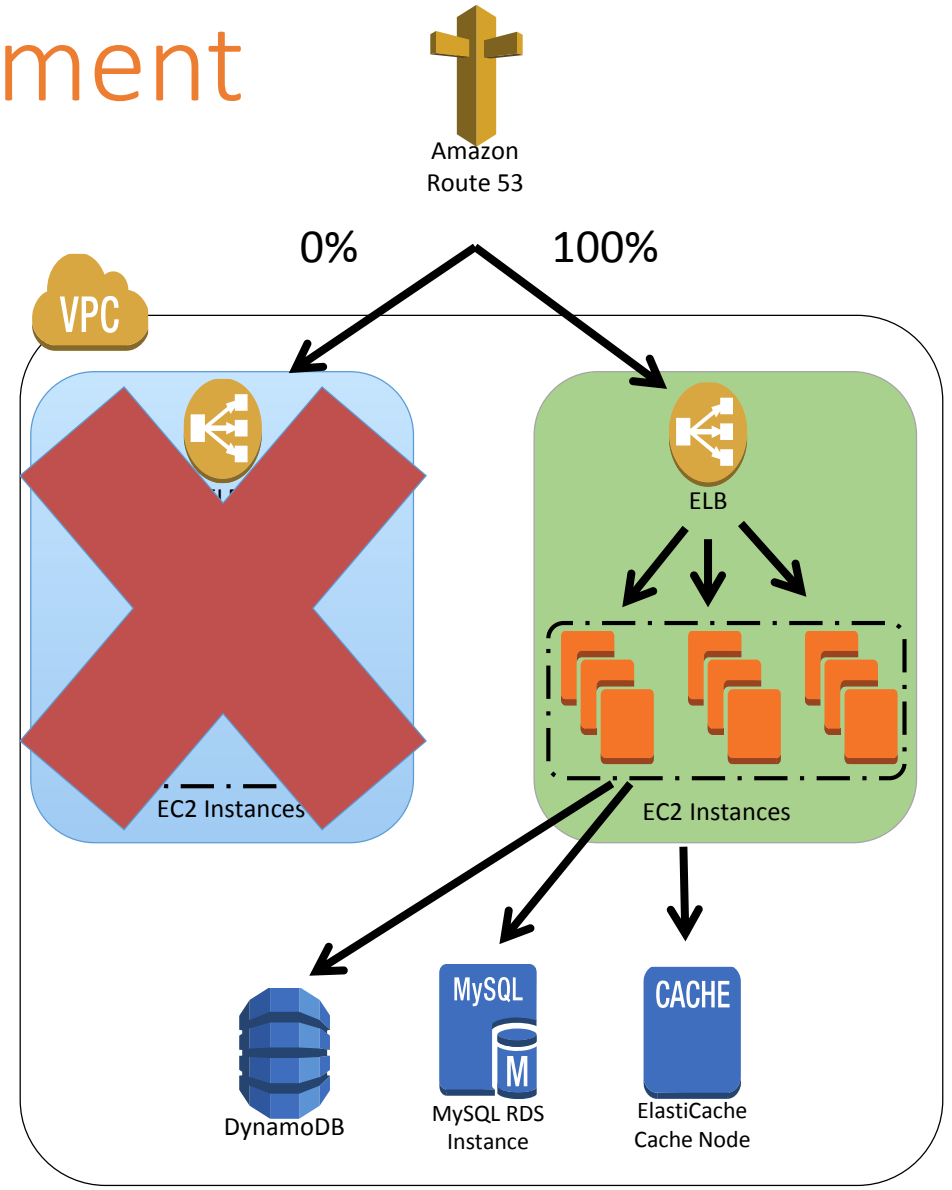
amazon web services

Blue-Green deployment



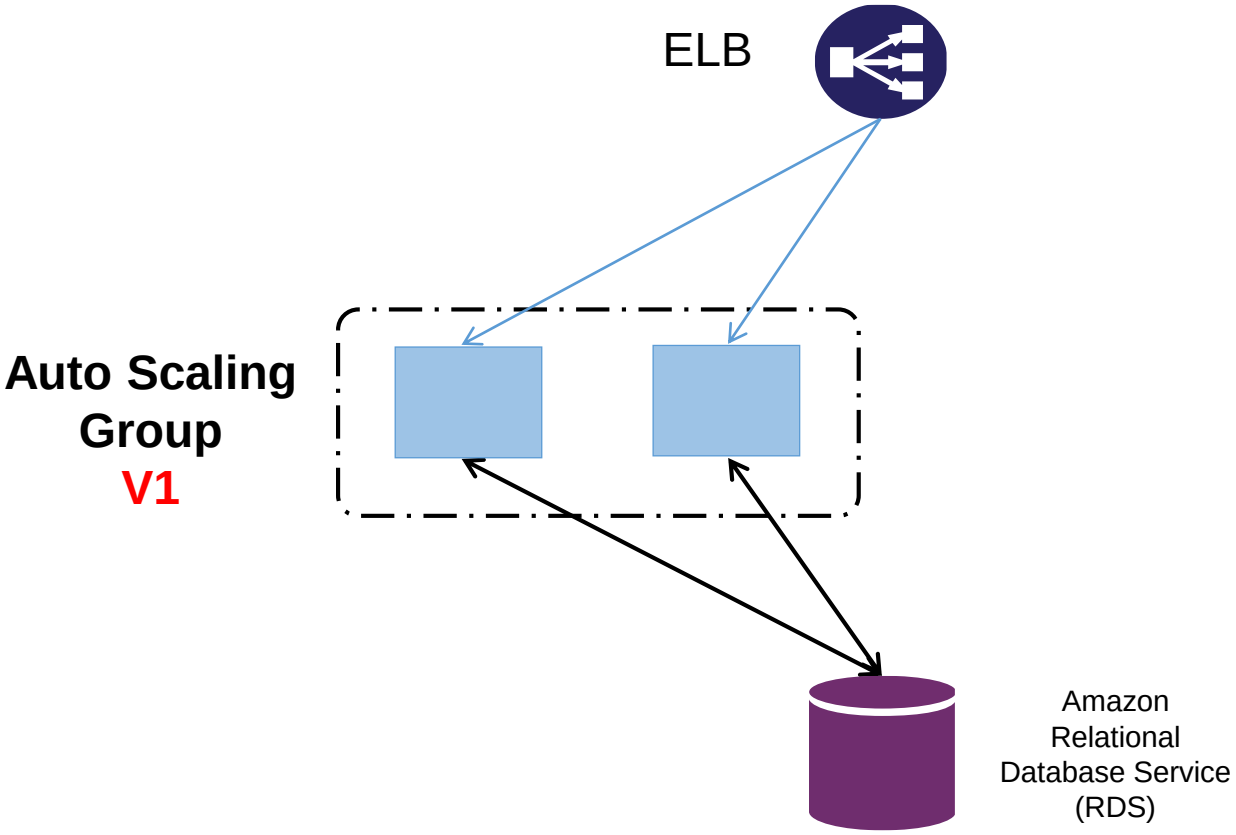
amazon web services

Blue-Green deployment



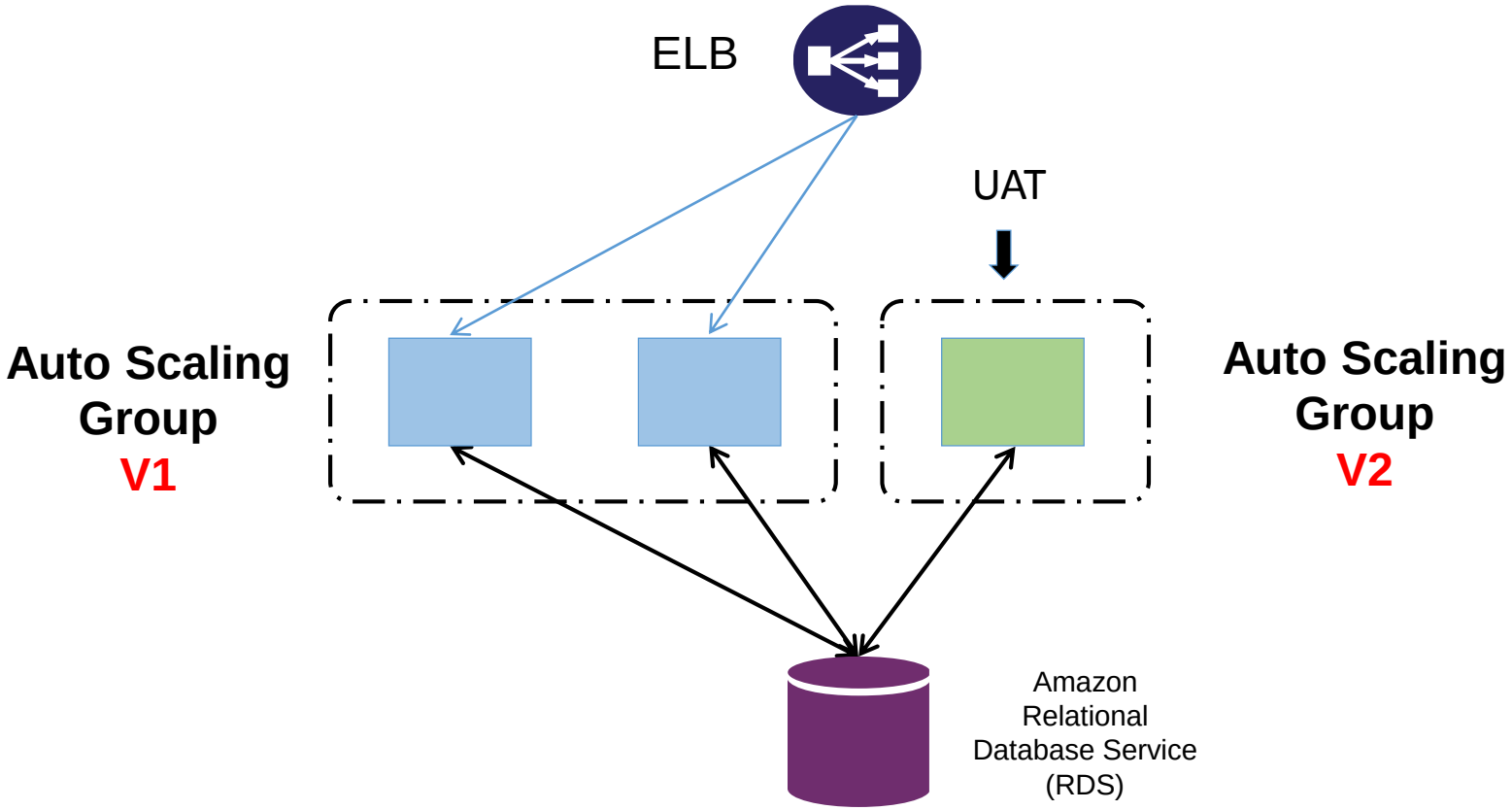
amazon web services

Red-Black Deployment



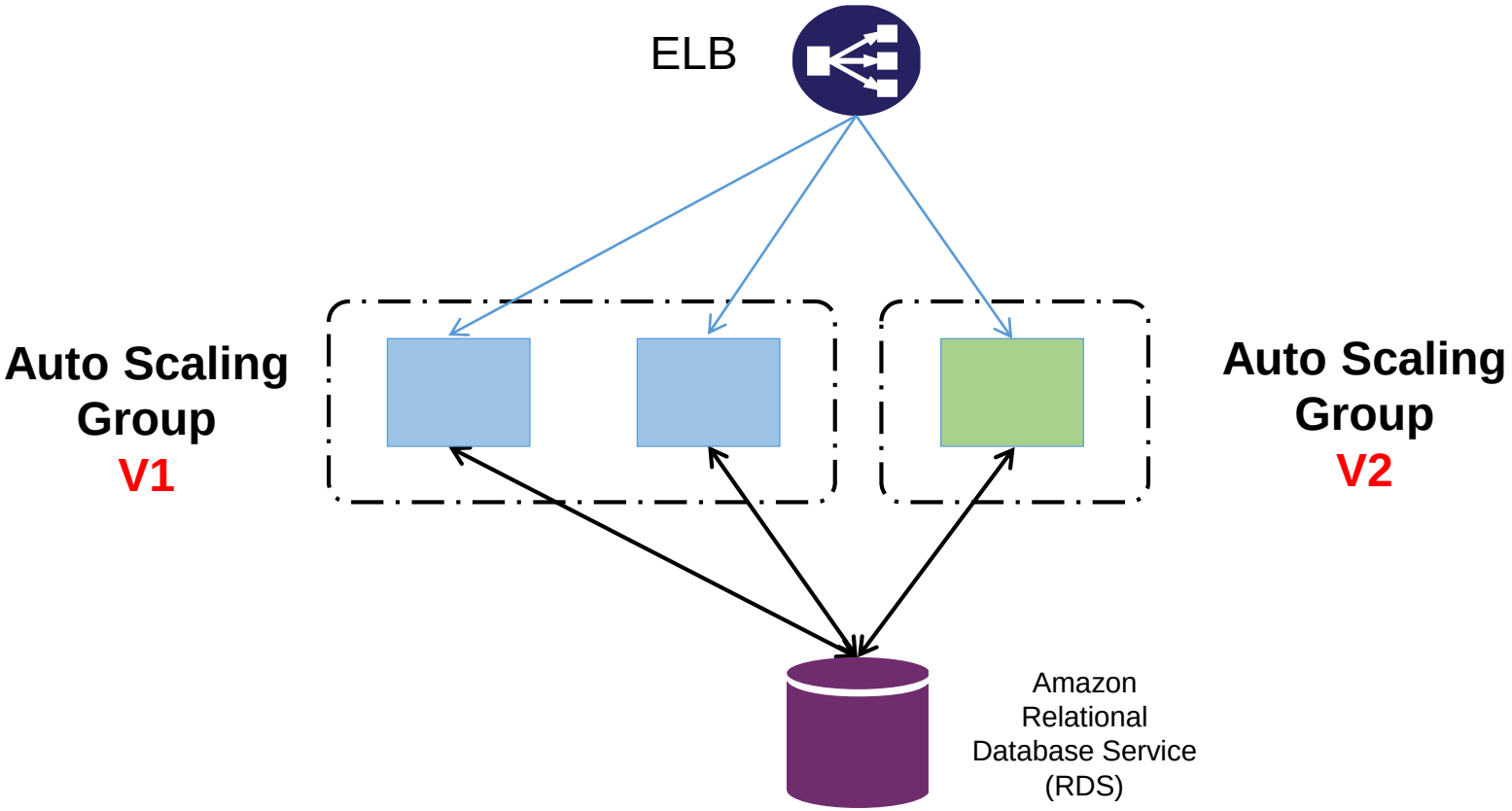
amazon web services

Red-Black Deployment



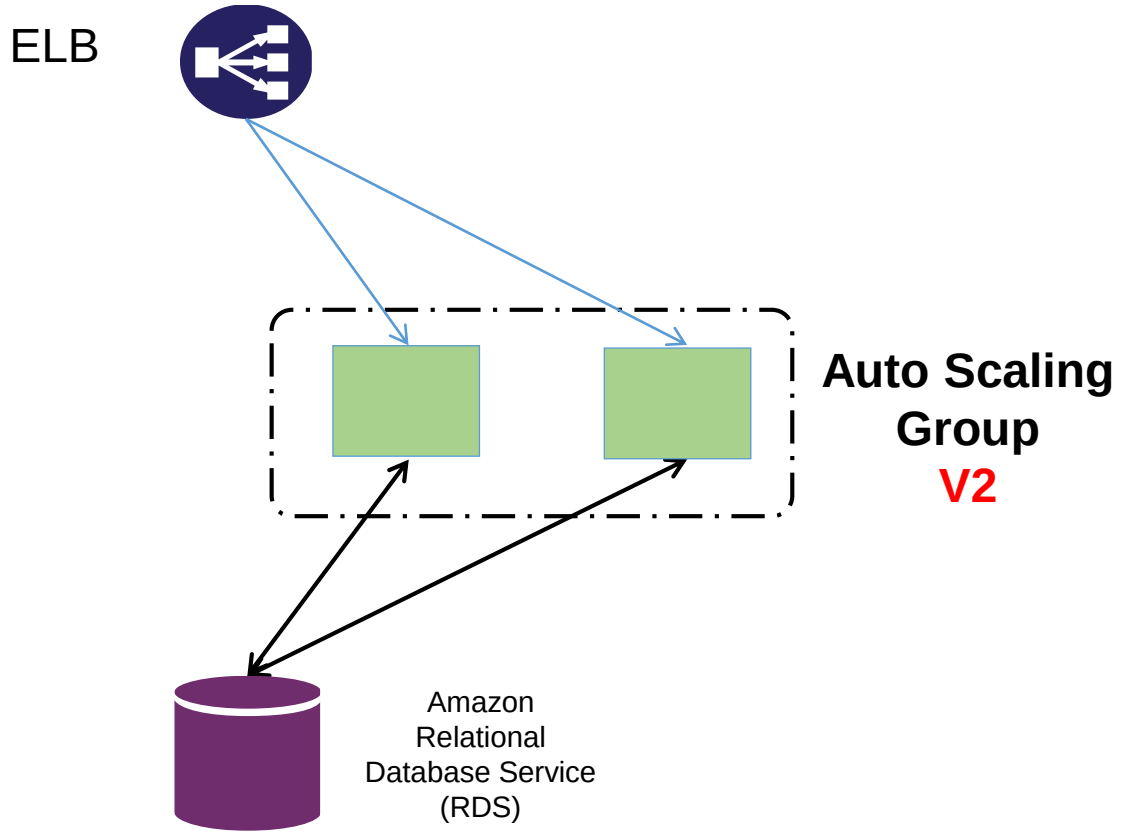
amazon web services

Red-Black Deployment

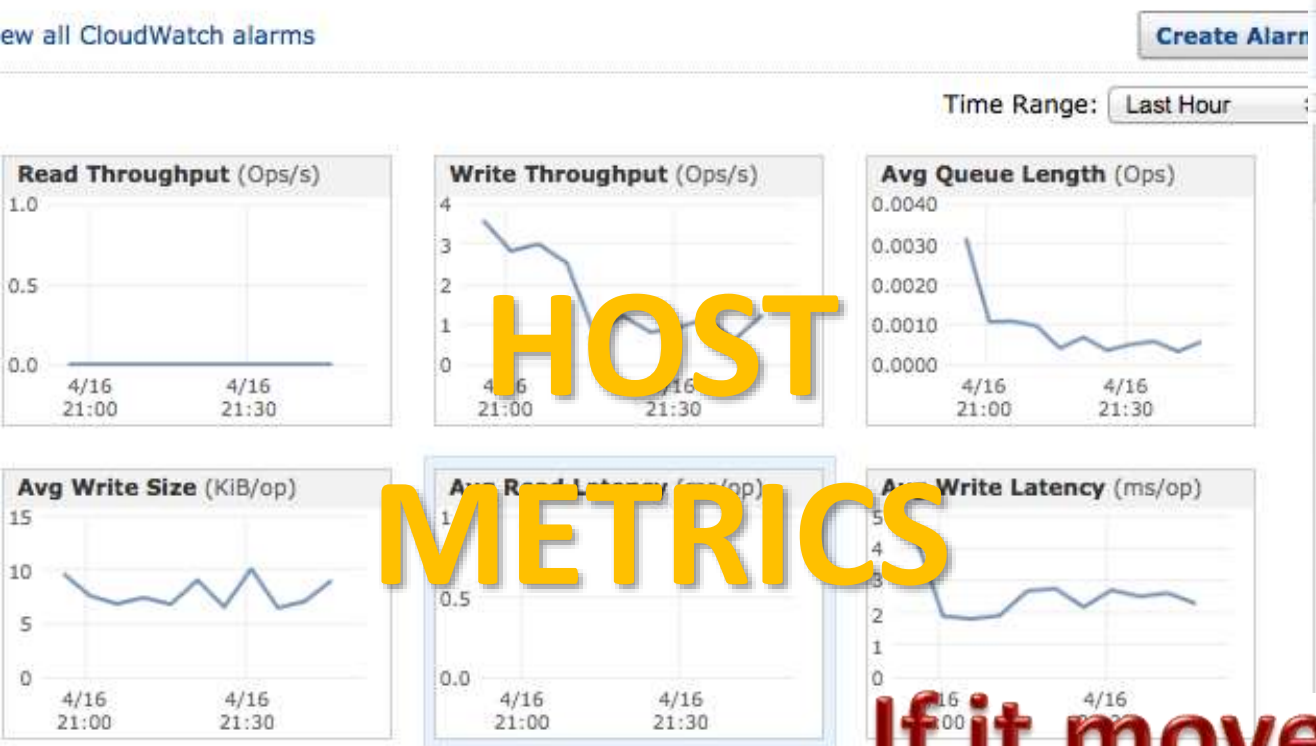


amazon web services

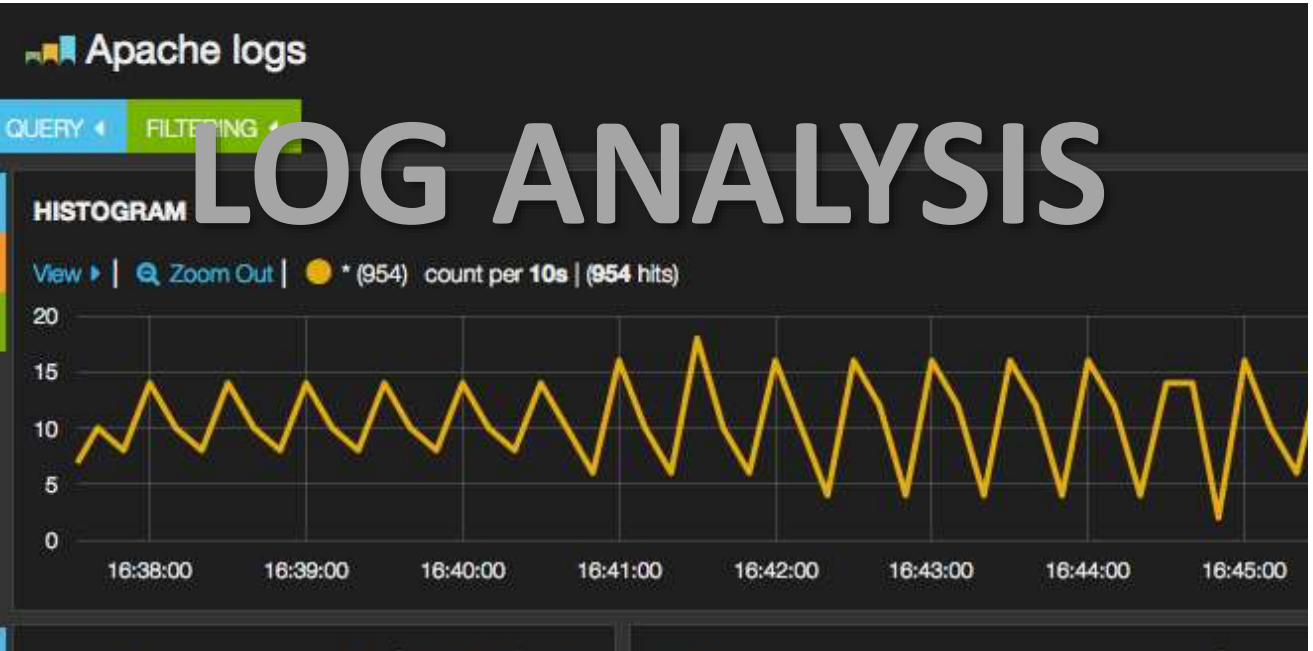
Red-Black Deployment



amazon web services



If it moves, plot it...



Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - **Application Management**
    - Elastic BeanStalk
    - Opsworks
    - Cloudformation
    - EC2 Container Service (ECS)
  - Application Lifecycle Management
    - Code Commit
    - Code Pipeline
    - Code Deploy

Deployment and Management

AWS Elastic Beanstalk

Automated resource management – web apps made easy

AWS OpsWorks

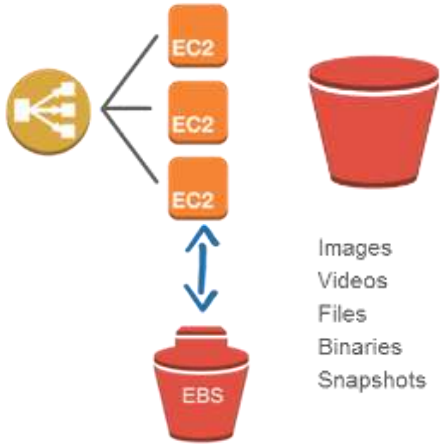
DevOps framework for application lifecycle management and automation

AWS CloudFormation

Templates to deploy & update infrastructure as code

DIY / On Demand

DIY, on demand resources: EC2, S3, custom AMI's, etc.



Convenience ⟷ Control

amazon web services

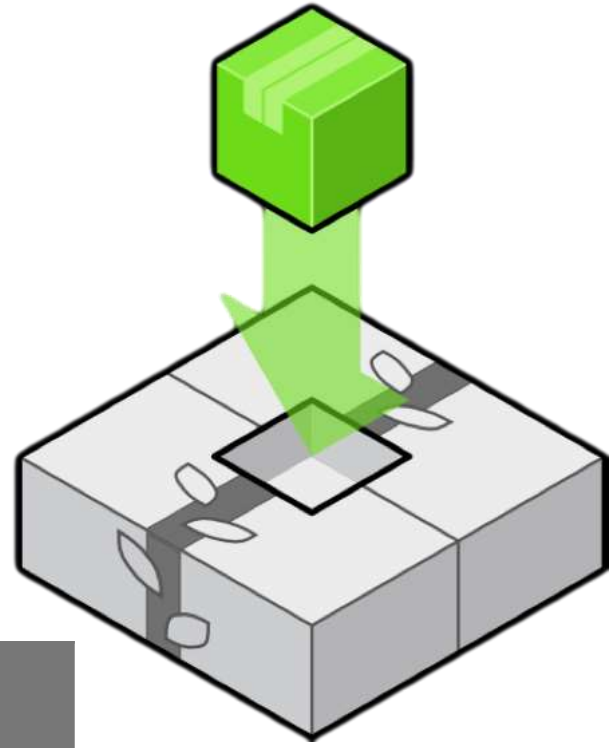
Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - **Application Management**
    - **Elastic BeanStalk**
    - Opsworks
    - Cloudformation
    - EC2 Container Service (ECS)
  - Application Lifecycle Management
    - Code Commit
    - Code Pipeline
    - Code Deploy

amazon web services

AWS Elastic Beanstalk (EB)

- Easily deploy, monitor, and scale three-tier web applications and services.
- Infrastructure provisioned and managed by EB – but you maintain complete control.
- Preconfigured application containers that are easily customizable.
- Support for these platforms:



Jav PHP Python Ruby .NET Node.js docker

amazon web services

Elastic Beanstalk object model

Application

Environments

- Infrastructure resources (such as EC2 instances, ELB load balancers, and Auto Scaling groups)
- Runs a single application version at a time for better scalability
- An application can have many environments (such as staging and production)

Application versions

- Application code
- Stored in Amazon S3
- An application can have many application versions (easy to rollback to previous versions)

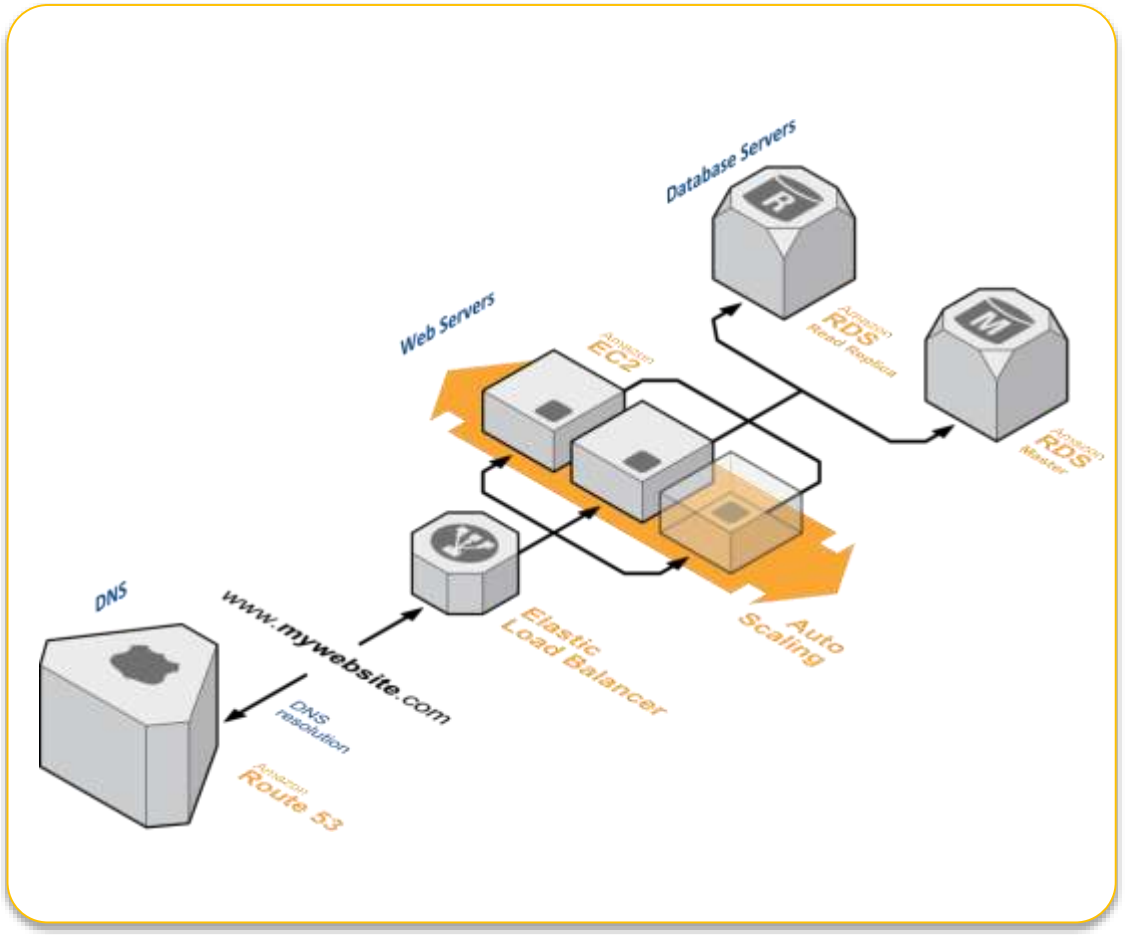
Saved configurations

- Configuration that defines how an environment and its resources behave
- Can be used to launch new environments quickly or roll-back configuration
- An application can have many saved configurations

Elastic Beanstalk environment

- Two types:
  - Single instance
  - Load balancing, auto scaling
- Two tiers (web server and worker)
- Elastic Beanstalk provisions necessary infrastructure resources such as load balancers, auto-scaling groups, security groups, and databases (optional)
- Configures Amazon Route 53 and gives you a unique domain name

  (For example: yourapp.elasticbeanstalk.com)

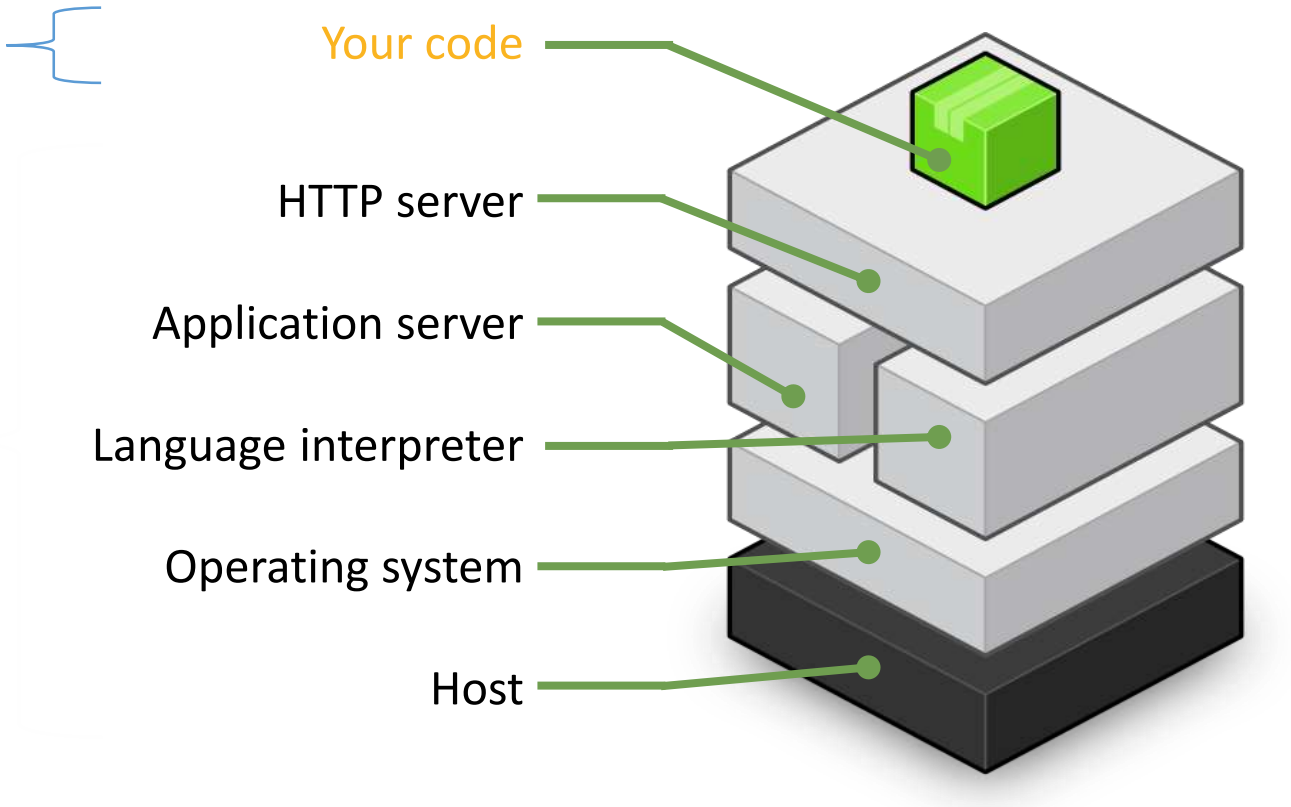


amazon web services

On-instance configuration

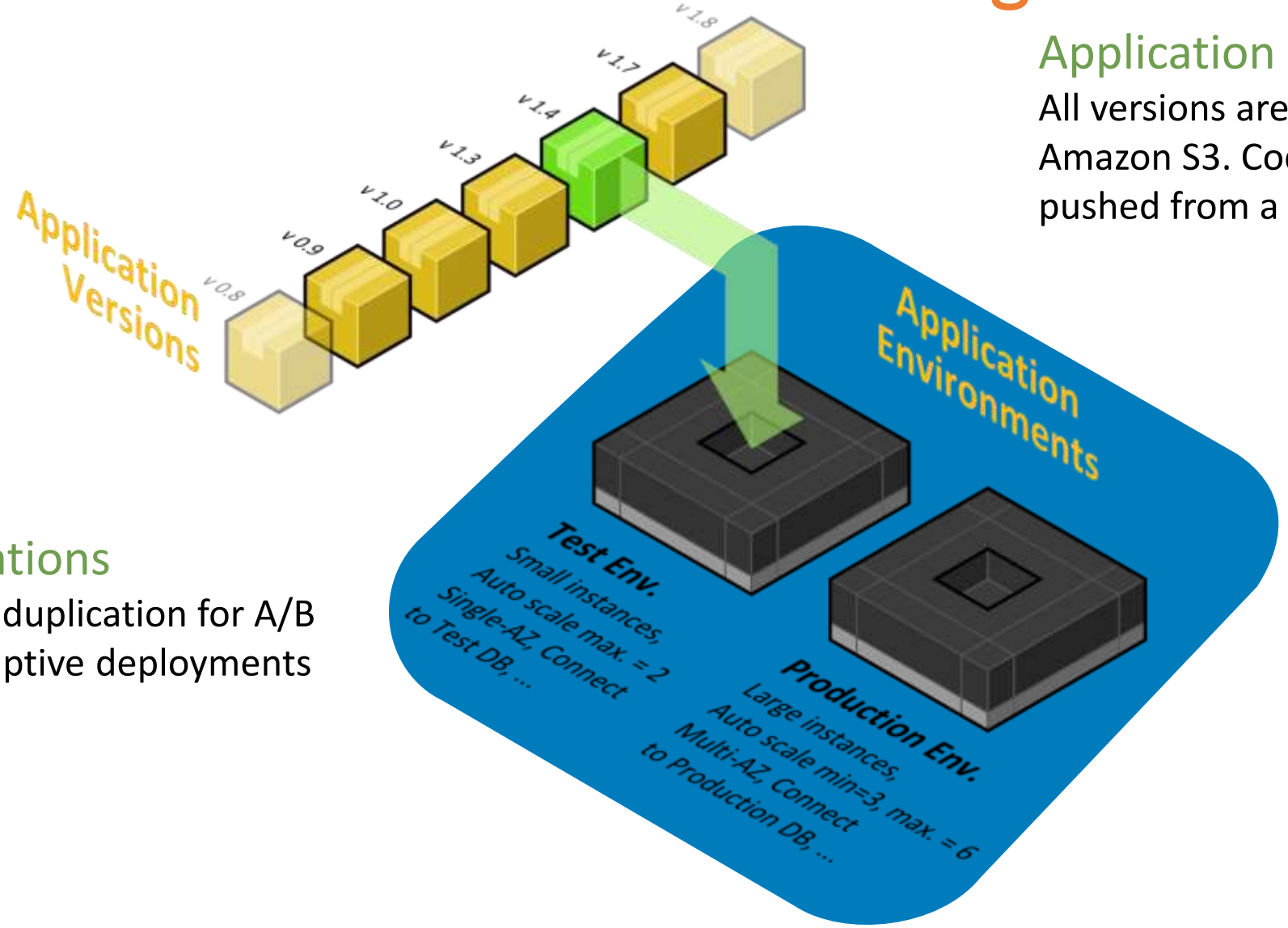
Focus on building your application

- Elastic Beanstalk configures each EC2 instance in your environment with the components necessary to run applications for the selected platform
- No more worrying about logging into instances to install and configure your application stack



amazon web services

Application versions and saved configurations



Application versions

All versions are stored durably in Amazon S3. Code can also be pushed from a Git repository!

Saved configurations

Save these for easy duplication for A/B testing or non-disruptive deployments

amazon web services

Deployment options

1. Via the AWS Management Console

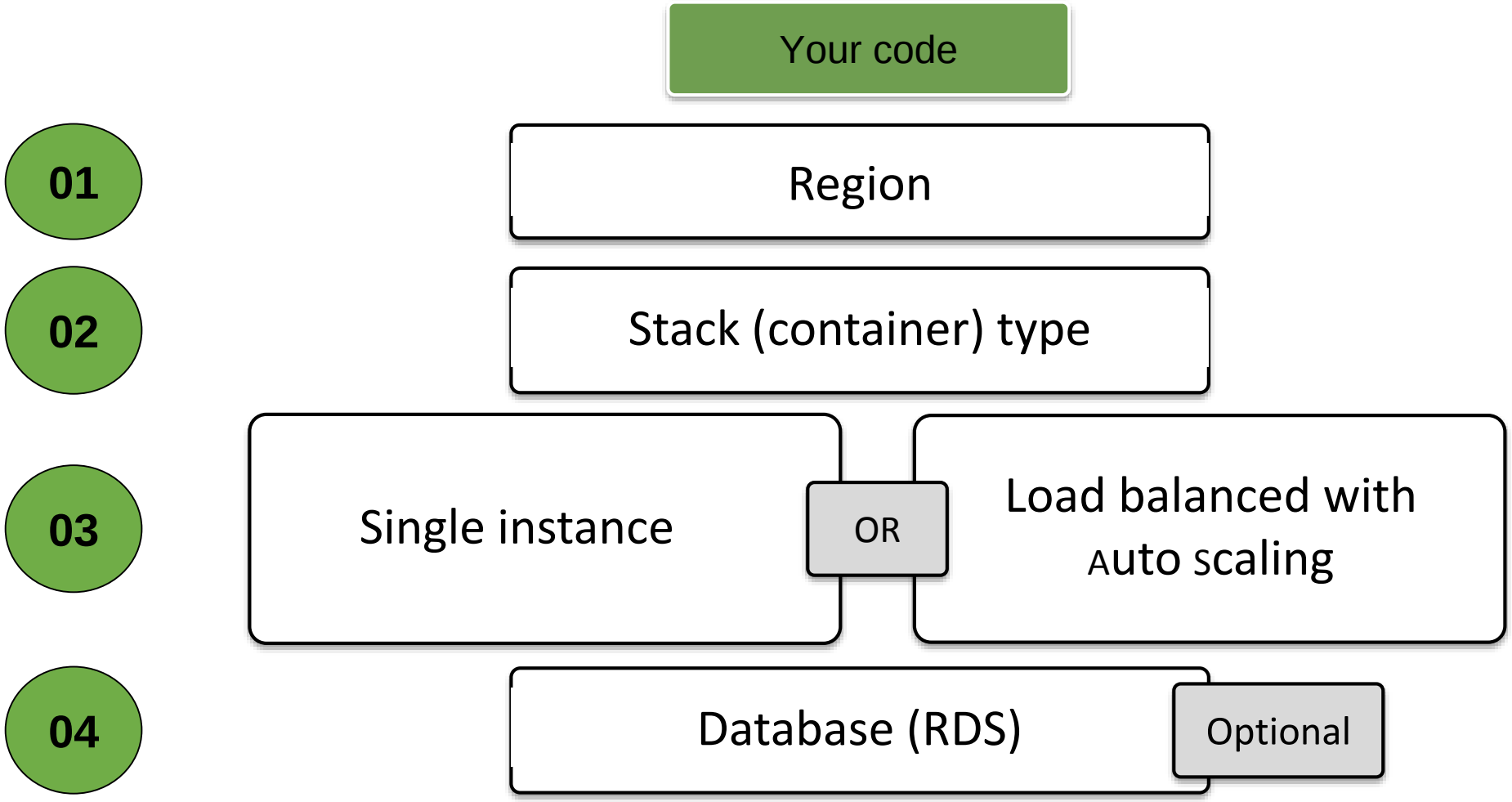
2. Via Git / EB CLI

```
$ git aws.push
```

3. Via the AWS Toolkit for Eclipse and the Visual Studio IDE

amazon web services

Deployment configuration



amazon web services

Example: CLI workflow

Initial app deployment:

01 Initialize your Git repository

```
$ git init .
```

02 Create your Elastic Beanstalk app

```
$ eb init
```

03 *Follow the prompts to configure the environment*

04 Add your code

```
$ git add .
```

05 Commit

```
$ git commit –m “v1.0”
```

06 Create the resources and launch the application

```
$ eb create
```

Example: CLI workflow

Update your app:

01 Update your code

02 Push the new code

```
$ git add .
$ git commit -m “v2.0”
$ eb deploy
```

03 Monitor the deployment progress

```
$ eb status
```

amazon web services

Customize application containers

Add custom software to your environment using ebextensions:

```
packages:
  yum:
    newrelic-sysmond: []
  rpm:
    newrelic: http://yum.newrelic.com/pub/newrelic/el5/i386/newrelic-repo-5-3.noarch.rpm

commands:
  0_newrelic_command:
    command: "touch /tmp/$(date '+%F.%T.%N').newrelic_command_0"
  1_configure_new_relic_key:
    command: nrsysmond-config --set license_key=<Your key here>
  1a_newrelic_command:
    command: "touch /tmp/$(date '+%F.%T.%N').newrelic_command_1a"
  2_start_new_relic:
    command: "/etc/init.d/newrelic-sysmond start"
  2a_newrelic_command:
    command: "touch /tmp/$(date '+%F.%T.%N').newrelic_command_2a"
```

Iterate on application architecture

Add additional resources to your environments using ebextensions:

Add other components such as:

- In-memory caching (Amazon ElastiCache Redis and Memcached)
- Amazon SQS
- Amazon CloudFront

```
Resources:
  MyElastiCache:
    Type: AWS::ElastiCache::CacheCluster
    Properties:
      CacheNodeType:
         Fn::GetOptionSetting:
             OptionName : CacheNodeType
             DefaultValue: cache.m1.small
      NumCacheNodes:
           Fn::GetOptionSetting:
             OptionName : NumCacheNodes
             DefaultValue: 1
      Engine:
           Fn::GetOptionSetting:
             OptionName : Engine
             DefaultValue: memcached
```

amazon web services

Zero-downtime deployments

Swap URLs

1. Create a new environment for an existing application

2. Deploy your updated application code to the new environment

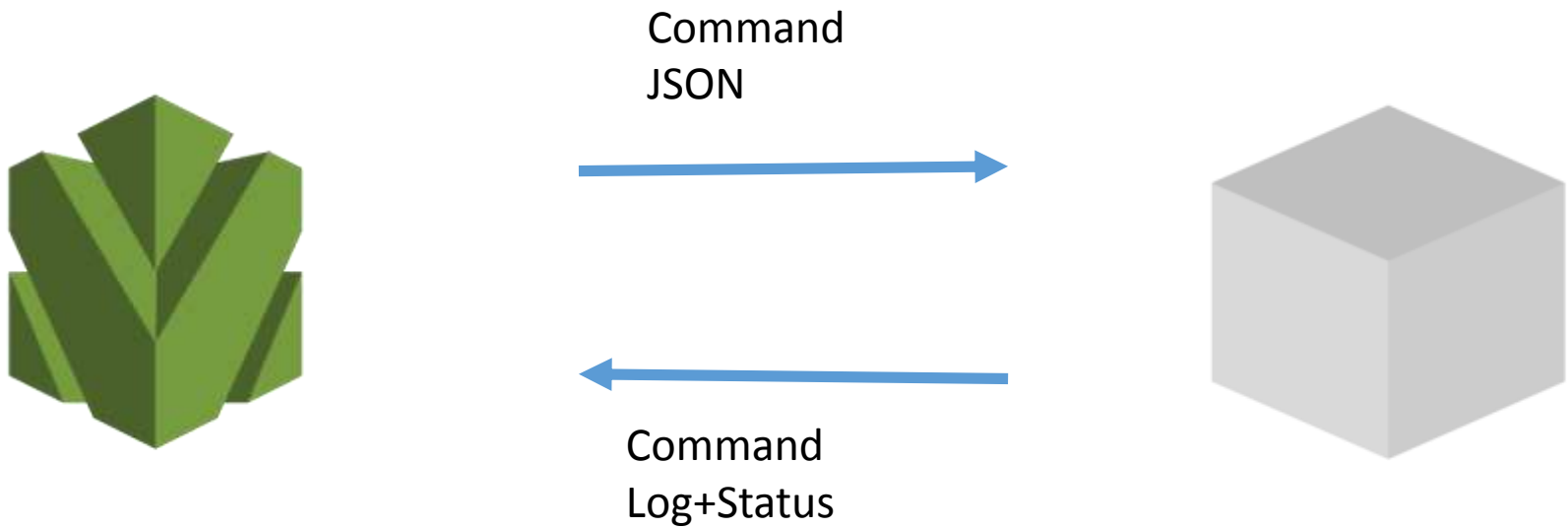
3. Use the “Swap URLs” feature to transition users to the new production environment

Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - **Application Management**
    - Elastic BeanStalk
    - **Opsworks**
    - Cloudformation
    - EC2 Container Service (ECS)
  - Application Lifecycle Management
    - Code Commit
    - Code Pipeline
    - Code Deploy

amazon web services

AWS OpsWorks architecture



Amazon EC2, Amazon EBS, EIP,
Amazon VPC, Elastic Load Balancing….
Auto-Scaling, Auto-Healing….

On-instance execution via
Chef client/zero

amazon web services

The heart of AWS OpsWorks

Agent on each
EC2 instance

understands a set of commands that are triggered by OpsWorks.
The agent then runs a Chef solo run.

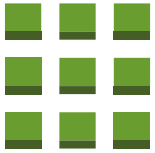
Chef integration

- Supports Chef 11.10
- Built-in convenience cookbooks / bring your own
- Chef run is triggered by lifecycle event firing:
  push vs. pull
- Event comes with stack state JSON

Opsworks components

Stack is basically a container for AWS resources—Amazon EC2 instances, Amazon EBS volumes, Elastic IP addresses, and so on—that have a common purpose and would be logically managed together.

A layer is basically a blueprint that specifies how to configure a set of Amazon EC2 instances for a particular purpose, such as serving applications or hosting a database server. Eg Java App server layer, PHP layer, RDS layer, MySQL Layer, HAProxy layer etc



An instance represents an Amazon EC2 instance and defines its basic configuration, such as operating system and size. Each layer has an associated set of Chef recipes that AWS OpsWorks runs on the layer's instances at key points in an instance's life cycle.

Each application is represented by an app, which specifies the application type and contains the information that AWS OpsWorks needs to deploy the application from the repository to your instances.

amazon web services

Opsworks components

instances

Scalability
- Auto healing
- Auto scaling
- Load balancing
- Scaling – time
- Scaling - load

applications

Configure Application
- Source of packages
- Git, svn, S3

layers

Application Architecture
- Load balancers
- Web layer
- Elastic IP's
- Security groups
- Database layer

deployments

Deployment
- Environments
- Dev, Test, Prod

stack

Infrastructure Provisioning
- Region
- Availability Zone
- Operating system
- Keys

AWS OpsWorks

monitoring

Monitoring
- Logs
- Monitor

amazon web services

Instance lifecycle commands

setup

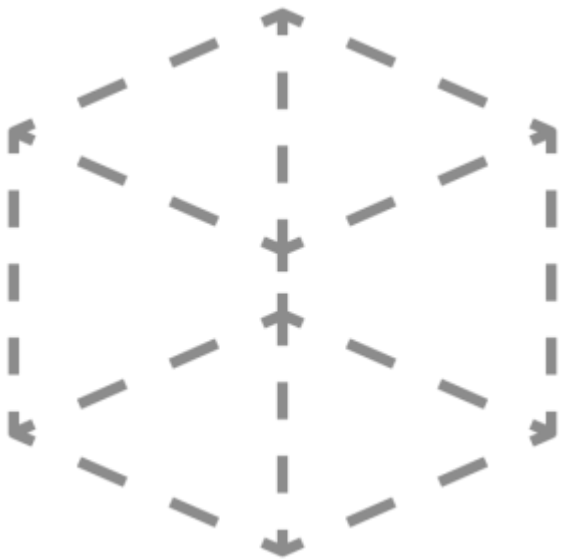
configure

deploy

undeploy

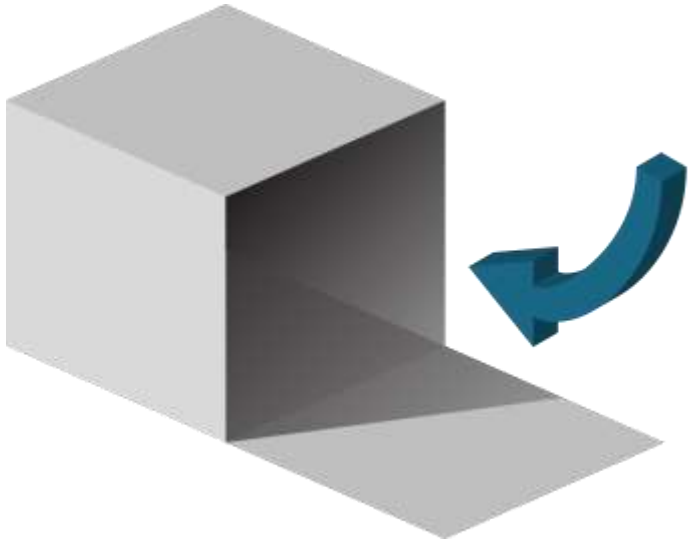
shutdown

Setup event



- Sent when instance boots
- Includes **deploy** event
- Use for initial installation of software & services

Configure event



- Sent to all instances when any instance enters or leaves online state
- Use for making sure the configuration is up-to-date

Deploy event

- Sent when you deploy via UI/API; part of each setup.
- Use for custom deployment

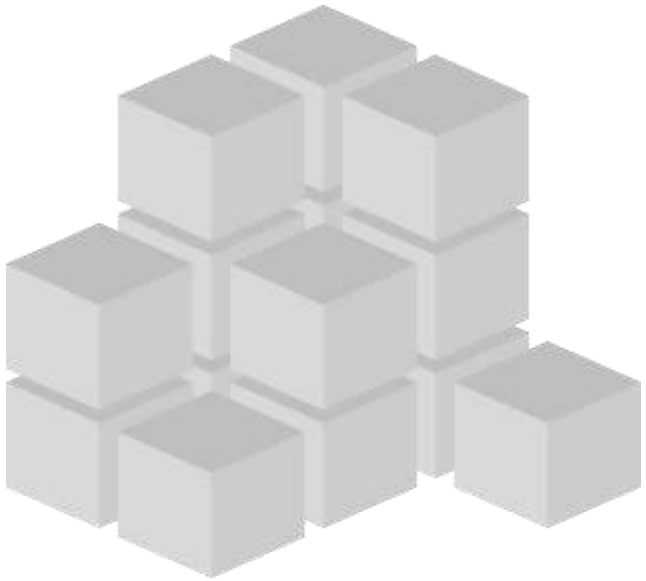
amazon web services

Undeploy event

- Sent via UI/API when apps are deleted
- Use to remove apps from running instances

amazon web services

Shutdown event



- Sent when an instance is shut down
- ~45s to execute
- Use for clean shutdown

Automation good!

https://secure.flickr.com/photos/macwagen/94975613

Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - **Application Management**
    - Elastic BeanStalk
    - Opsworks
    - **Cloudformation**
    - EC2 Container Service (ECS)
  - Application Lifecycle Management
    - Code Commit
    - Code Pipeline
    - Code Deploy

amazon web services

Amazon CloudFormation

AWS CloudFormation

- Infrastructure as Code
- Integrates with version control
- JSON format
- Templates
- Stacks
- Supports all AWS resource types

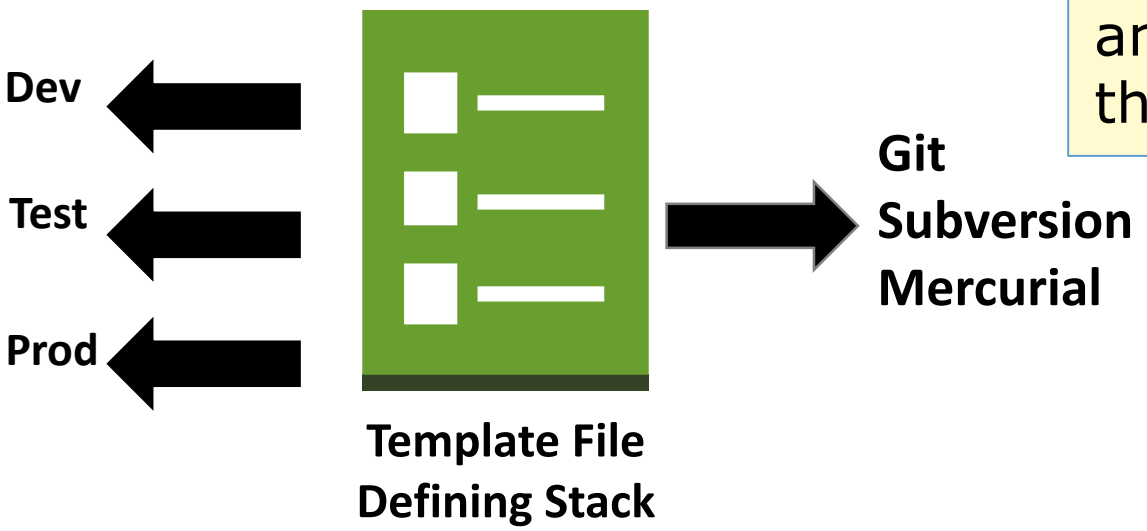
amazon web services

AWS CloudFormation: Model Your App

- Document, version control, and share your applications and infrastructure as a JSON document
- Provision app and other AWS resources (VPC, DynamoDB, etc) from a template
- Repeatable, reliable deployments for test/dev/prod in any AWS Region

AWS CloudFormation: Application stack example (continue)

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



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

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

Template Anatomy

```
{
"Description" : "Create an EC2 instance.”,
"Resources" : {
   "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
         "KeyName" : “my-key-pair”,
         "ImageId" : "ami-75g0061f”,
         “InstanceType” : “m1.medium”
      }
   }
}
}
```

amazon web services

Template Anatomy

```
{
"Description" : "Create an EC2 instance.”,
"Parameters" : {
   "UserKeyName" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "String"
   }
},
"Resources" : {
   "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
         "KeyName" : { “Ref” : “UserKeyName”},
         "ImageId" : "ami-75g0061f”,
         “InstanceType” : “m1.medium”
      }
   }
}
}
```

amazon web services

Template Anatomy

```
{
"Description" : "Create an EC2 instance.”,
"Parameters" : {
   "UserKeyName" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "String"
   },
  “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" : { “Ref” : “UserKeyName”},
         "ImageId" : "ami-75g0061f”,
         “InstanceType” : { “Ref” : “InstanceType” }
      }
   }
},
"Outputs" : {
   "InstancePublicDnsName" : {
      "Description" : "The public DNS name of the newly created EC2 instance",
      "Value" : { ”Fn::GetAtt" : [ "Ec2Instance”, “PublicDnsName” ] }
   }
}
}
```

amazon web services

Application Deployment - User Data

```
"UserData": {
          "Fn::Base64": {
            "Fn::Join": [
              "",
              [
                "#!/bin/bash -ex\n",
                "yum -y install git-core\n",
                "yum -y install php-pear\n",
                "pear install Crypt_HMAC2-1.0.0\n",
                "pear install HTTP_Request-1.4.4\n",
        "pear install aws/sdk\n",
```

Application Deployment - cfn-init

```
"Ec2Instance": {
  "Metadata": {
    "AWS::CloudFormation::Init": {
      "config": {
        "sources" : {
          "/usr/local/bin/s3cmd" : "https://github.com/s3tools/s3cmd”
        },
        "packages": {
          "yum": { "git": [] }
        }
    }
  }
}
```

3rd Party Tools

- Easily integrate with existing configuration management tools
- Simply use User-Data or cfn-init to configure agents

Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - **Application Management**
    - Cloudformation
    - Elastic BeanStalk
    - Opsworks
    - **EC2 Container Service (ECS)**
  - Application Lifecycle Management
    - Code Commit
    - Code Pipeline
    - Code Deploy

amazon web services

Preview

EC2 Container Service (ECS)

- Cluster Management Made Easy
- Flexible Scheduling
- High Performance
- Resource Efficiency
- Extensible
- Security
- Programmatic Control
- Docker Compatibility
- Monitoring
- AWS Integration

amazon web services

Preview

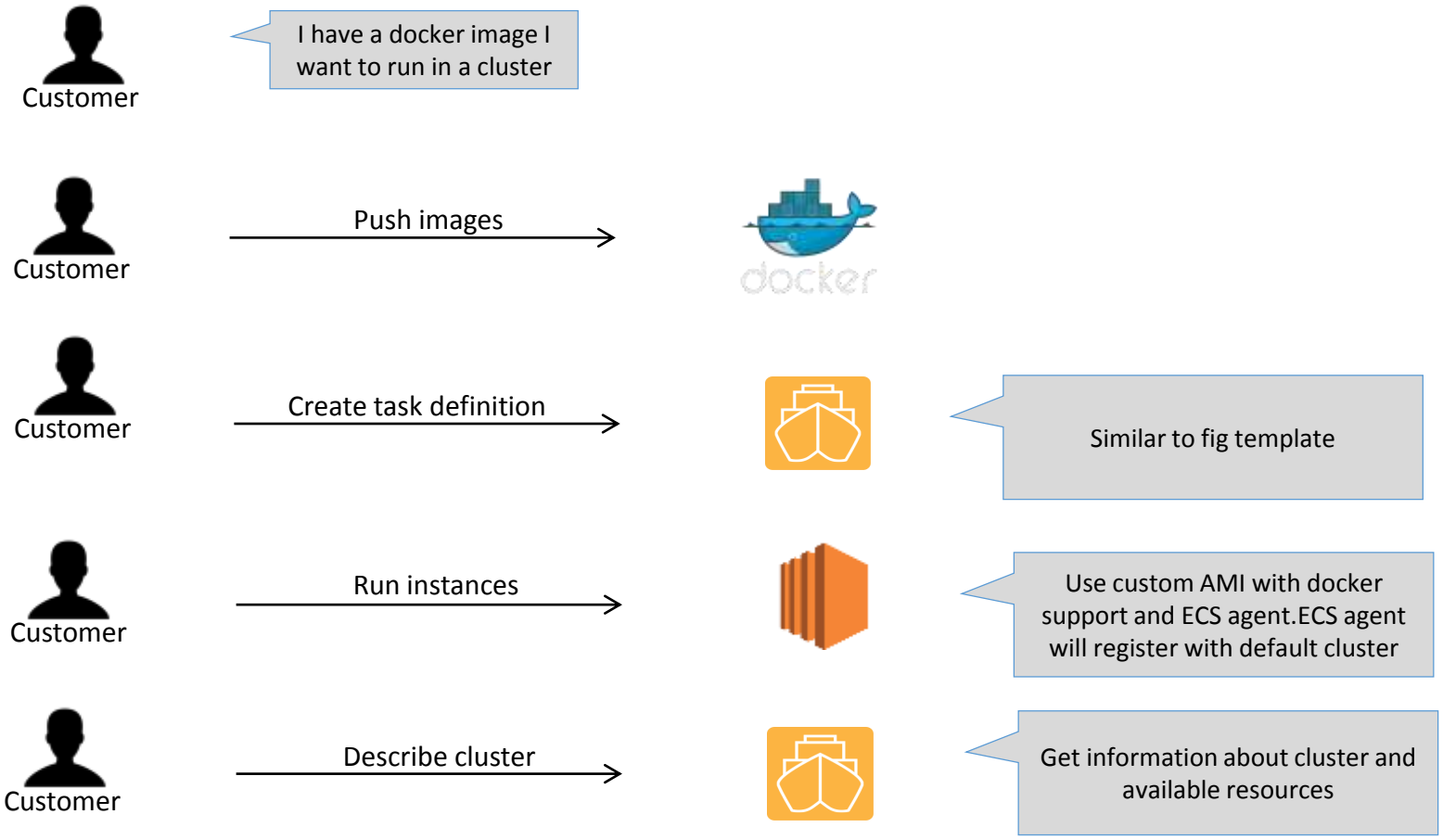
ECS Components

- Containers
  - Names and identifies your image
  - Includes default runtime attributes for your container (Environment Variables, Port Mappings, Container entry point and commands, Resource constraints…)
- Tasks
  - A group of related containers
- Container Instances
  - An instance on which Tasks are scheduled
  - Runs AMI with ECS Agent installed
  - Registers into cluster on launch
- Clusters
  - Provides a pool of resources for your Tasks
  - A grouping of Container Instances
  - Starts empty, dynamically scalable

amazon web services

Preview

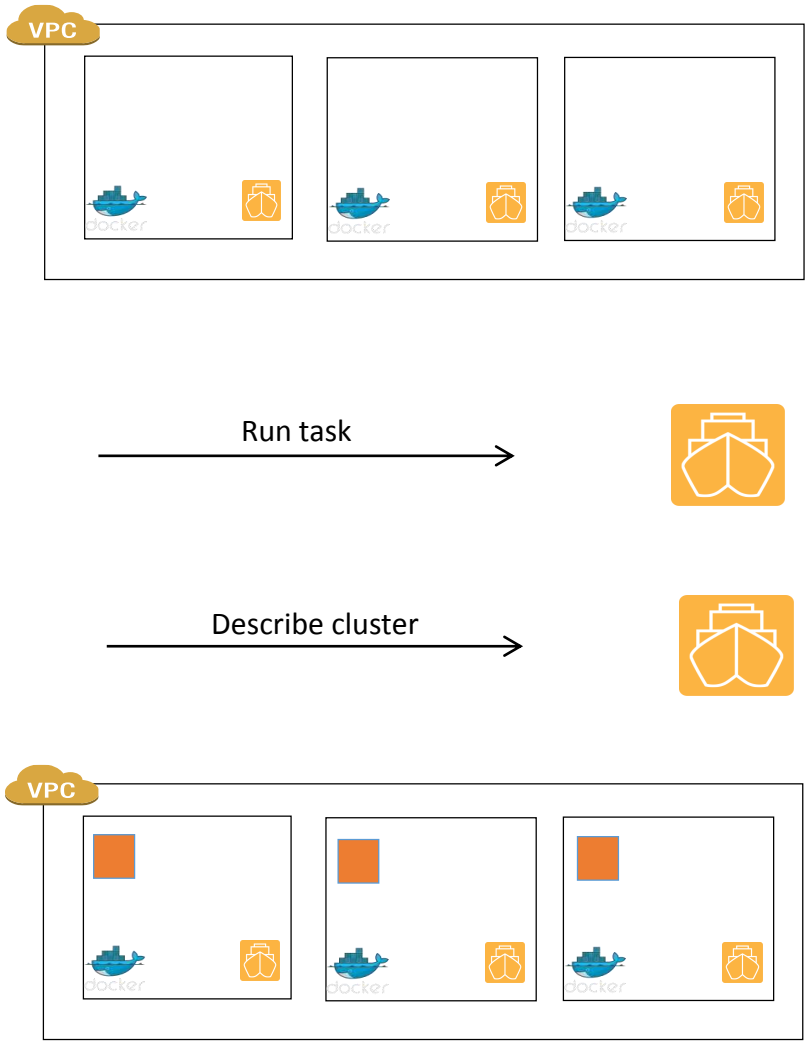
User Workflow



amazon web services

Preview

User Workflow



Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - Application Management
    - Cloudformation
    - Elastic BeanStalk
    - Opsworks
    - EC2 Container Service (ECS)
  - **Application Lifecycle Management**
    - **Code Commit**
    - Code Pipeline
    - Code Deploy

Announced

ALM | What is CodeCommit?

A secure, highly scalable, managed source control service that hosts private Git repositories.

Eliminates the need to operate your own source control system or worry about scaling its infrastructure.

Basically, managed Git

amazon web services

Announced

ALM | What is CodeCommit?

Fully managed service source control service for hosting private Git repositories

Automatically scales to meet the needs of your project
Stores any type of file (source, images, videos, libraries etc.) with no limit on repository size.

Fully integrated with AWS **CodePipeline** and AWS **CodeDeploy** to streamline development and release processes.

amazon web services

Announced

ALM | What is CodeCommit?

Only transfers incremental changes – not the entire application

CodeCommit supports all Git commands and works with your existing Git-based tools (e.g., continuous integration/continuous delivery systems, and graphical clients).

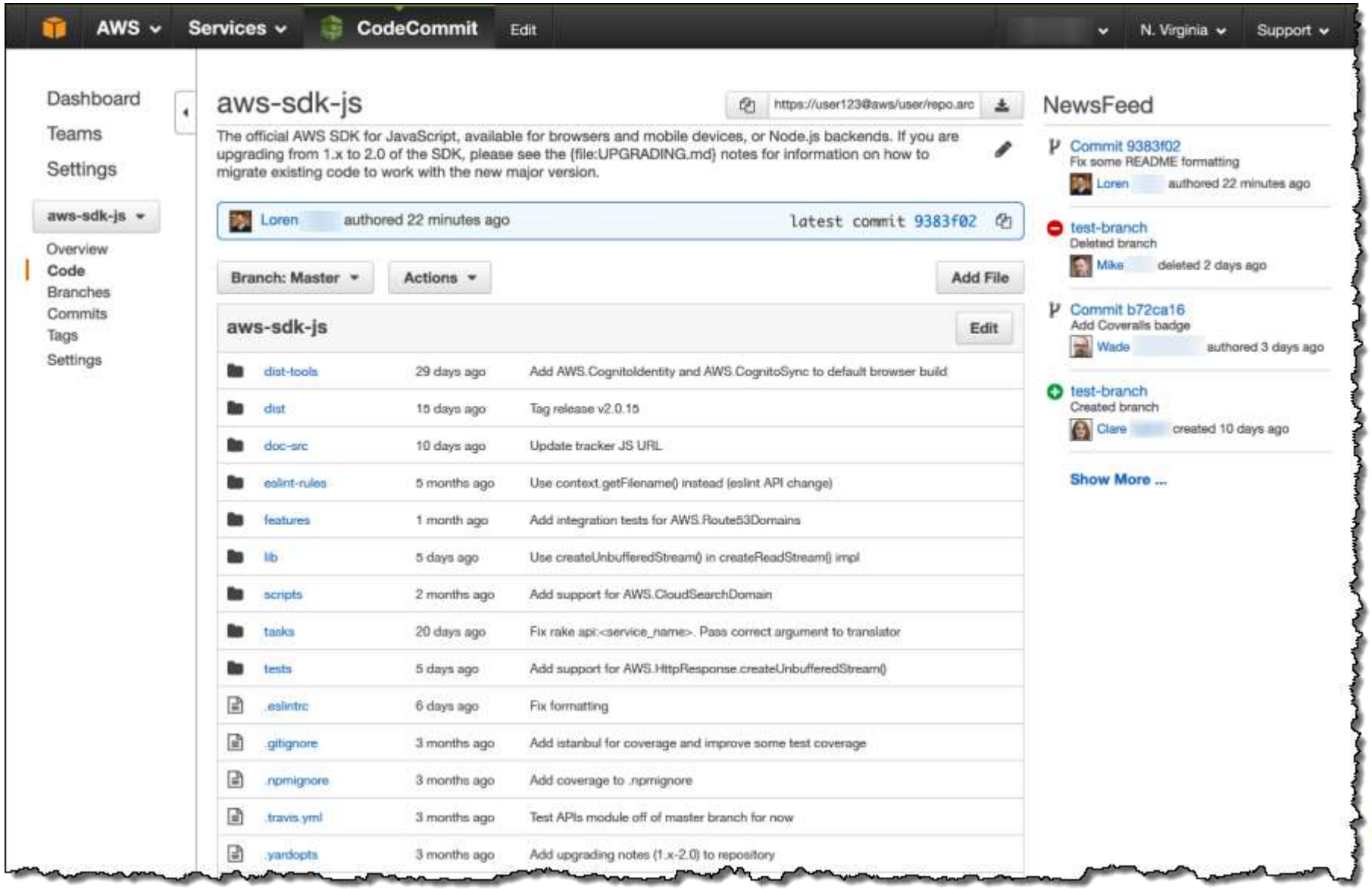
Built-in encryption support

Fully integrated with AWS Identity and Access Management (IAM)

amazon web services

Announced

ALM | Preliminary look at CodeCommit console



amazon web services

Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - Application Management
    - Cloudformation
    - Elastic BeanStalk
    - Opsworks
    - EC2 Container Service (ECS)
  - **Application Lifecycle Management**
    - Code Commit
    - **Code Pipeline**
    - Code Deploy

Announced

ALM | What is CodePipeline?

A continuous delivery and release automation service that aids smooth deployments.

You can design your development workflow for **checking in** code, **building** the code, **deploying** your application into staging, testing it, and releasing it to production

Similar to Bamboo or Jenkins

amazon web services

Announced

ALM | What is CodePipeline?

CodePipeline standardizes and automates the software release process, allowing you to rapidly release new features to users

Provides the capability to set up configurable gates between each stage such as time-based rules or manual approvals

Workflows can be created to run unit and integration tests before deploying to production

amazon web services

Announced

ALM | What is CodePipeline?

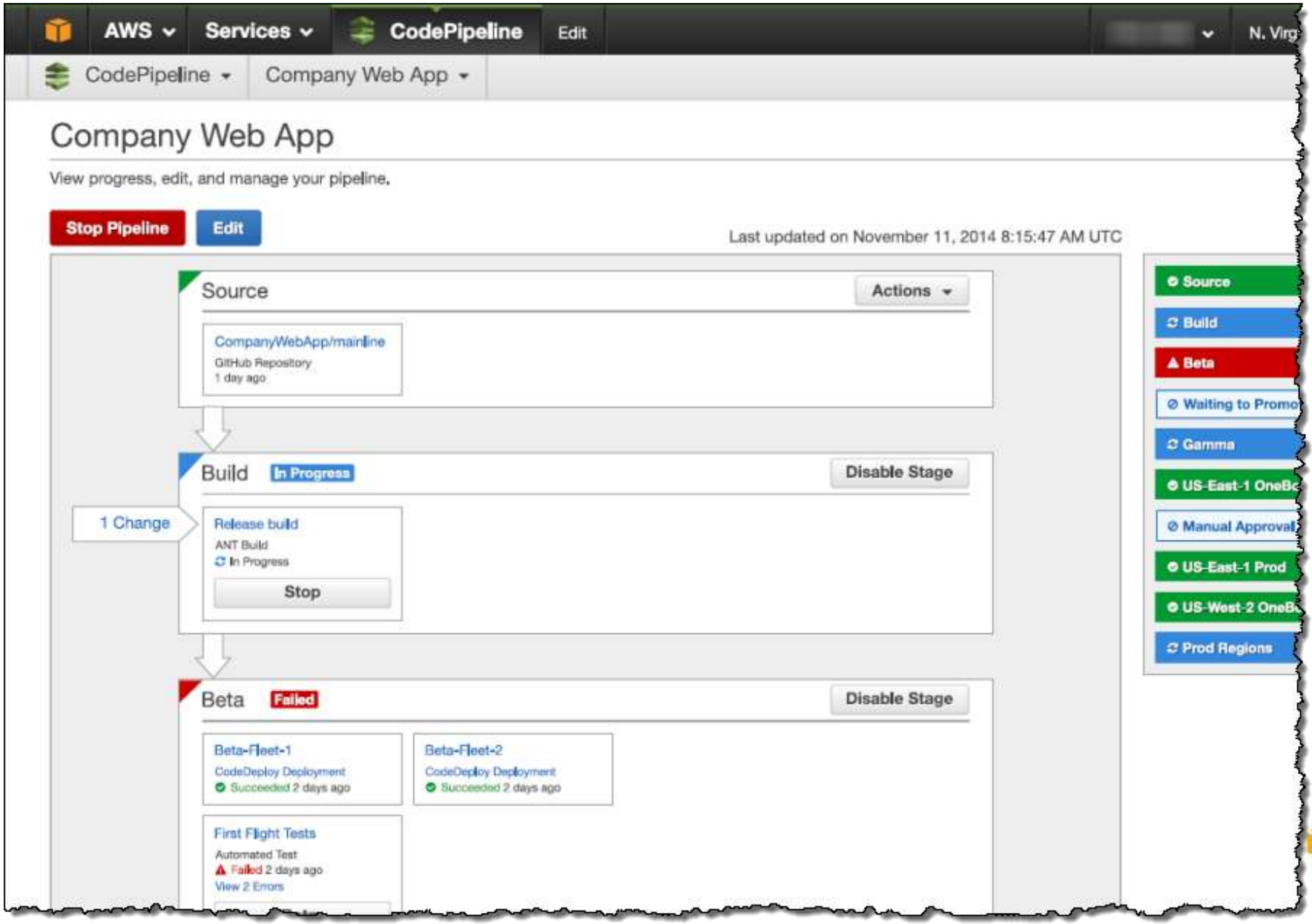
IMPORTANT:

Able to be used stand-alone as an end-to-end solution, or can be integrated with your existing source control system, test framework or build tools (like Bamboo, Jenkins, etc)

amazon web services

Announced

ALM | Preliminary look at the console



amazon web services

Agenda

- Intro to Continuous Integration and Continuous Deployment/Delivery (CI-CD)
- CD Strategies
- CI-CD on AWS
  - Application Management
    - Cloudformation
    - Elastic BeanStalk
    - Opsworks
    - EC2 Container Service (ECS)
  - **Application Lifecycle Management**
    - Code Commit
    - Code Pipeline
    - **Code Deploy**

amazon web services

Limited regions

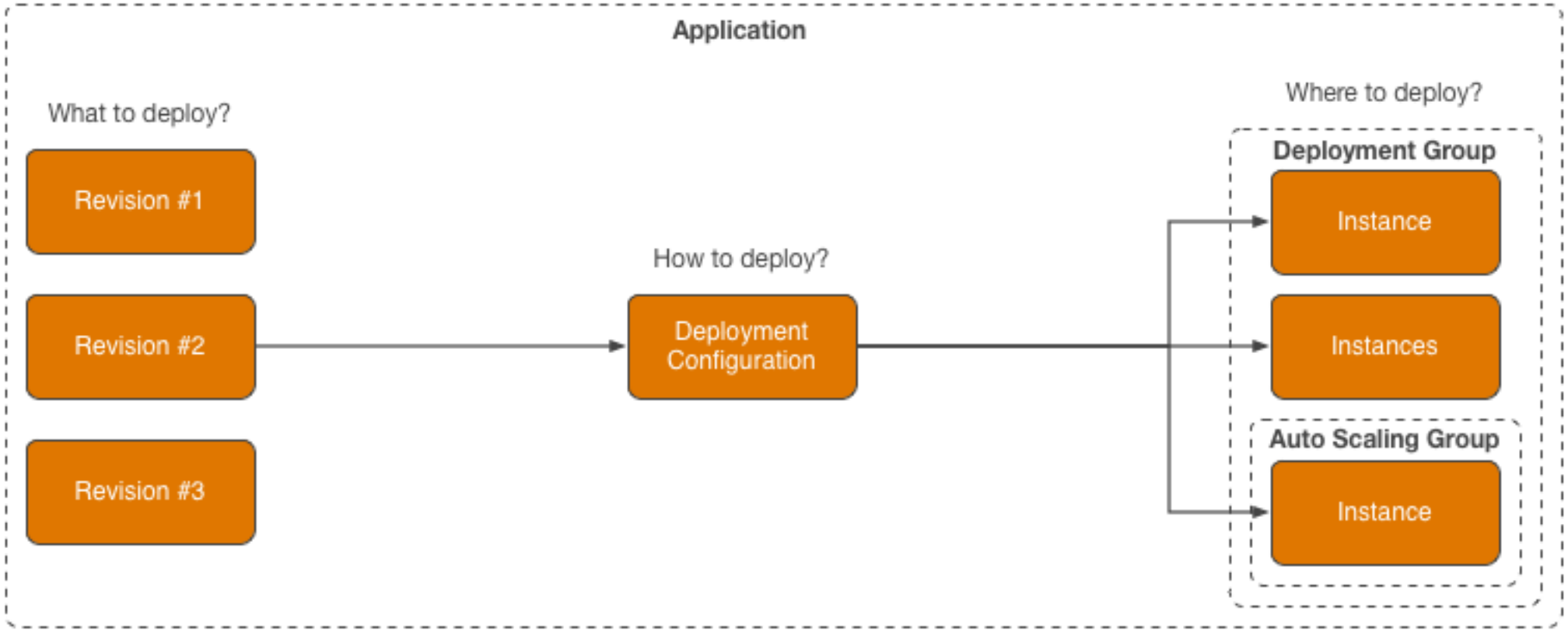
Code Deploy

- Deploys your released code to a "fleet" of EC2 instances
- Accommodate fleets that range in size from one instance all the way up to tens of thousands of instances
- Automatically schedules updates across multiple Availability Zones in order to maintain high availability during the deployment
- Application and Deployment groups described in YAML-formatted files
- Deployment groups identify EC2 instances by tags & can also reference Auto Scaling Groups
- Managed via AWS Management Console, CLI or APIs
- Can be used in conjunction with Chef recipes or Puppet scripts

amazon web services

Limited regions

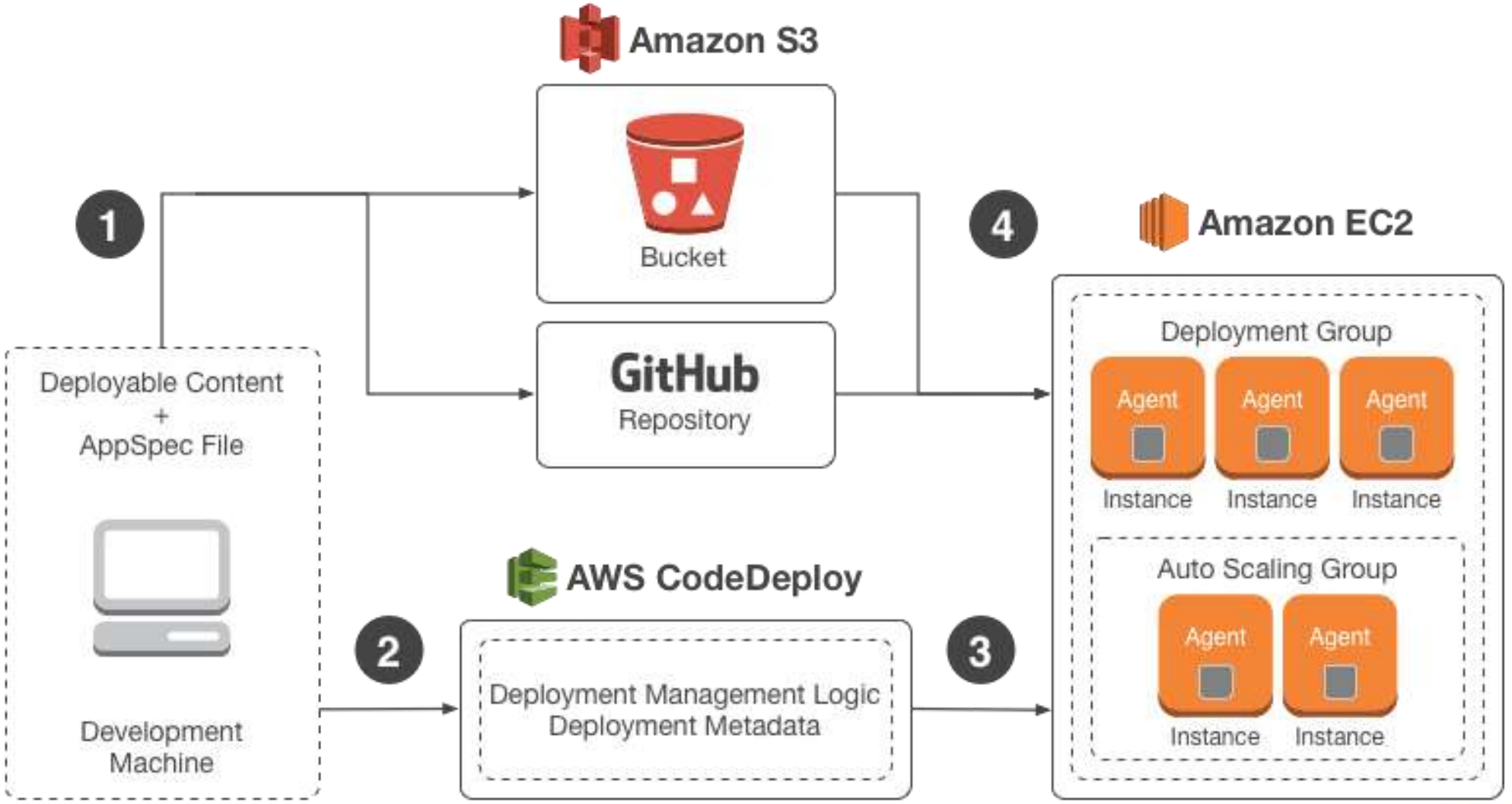
Code Deploy components



amazon web services

Limited regions

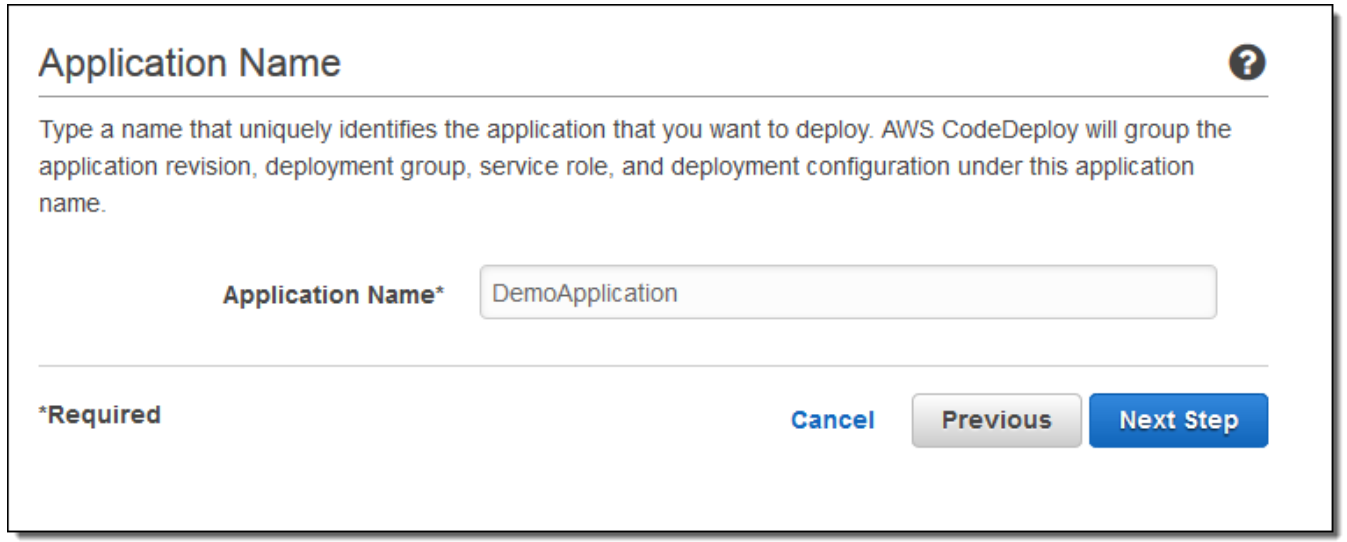
Code Deploy Workflow



amazon web services

Limited regions

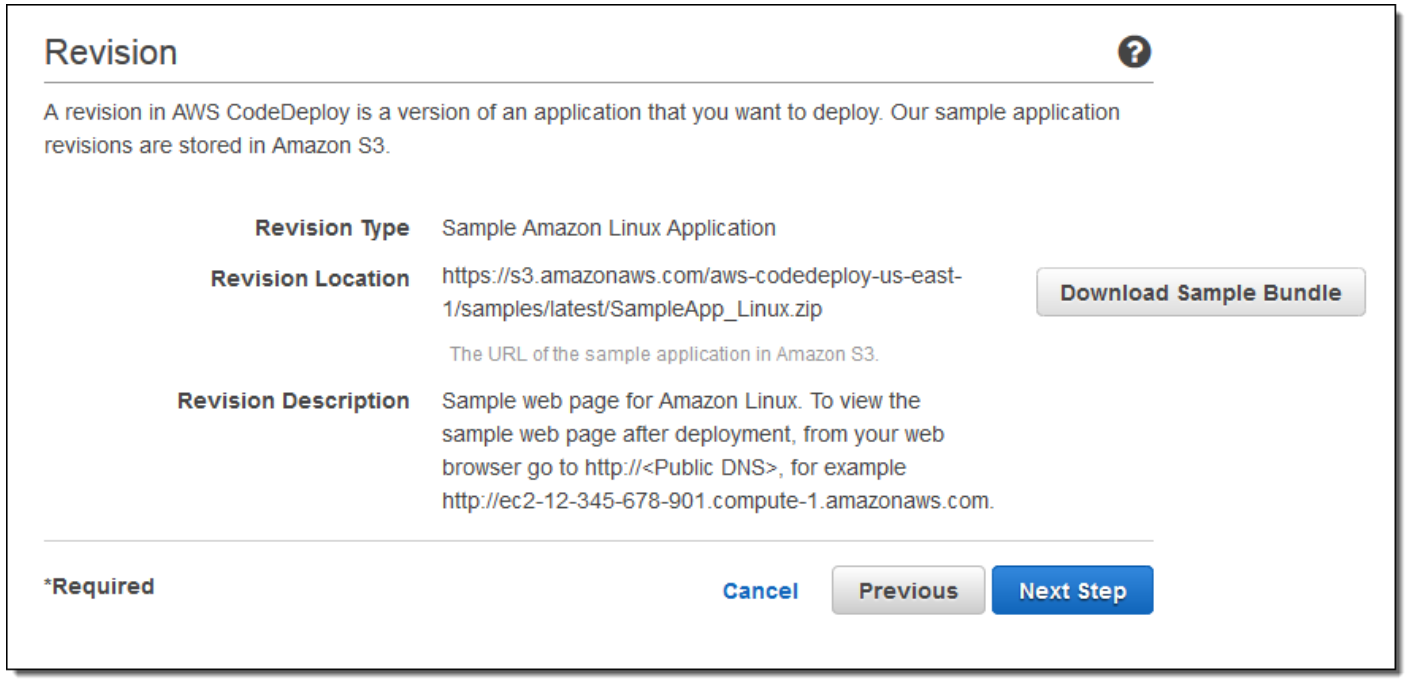
Using AWS CodeDeploy



- Begin by defining an Application

Limited regions

Using AWS CodeDeploy

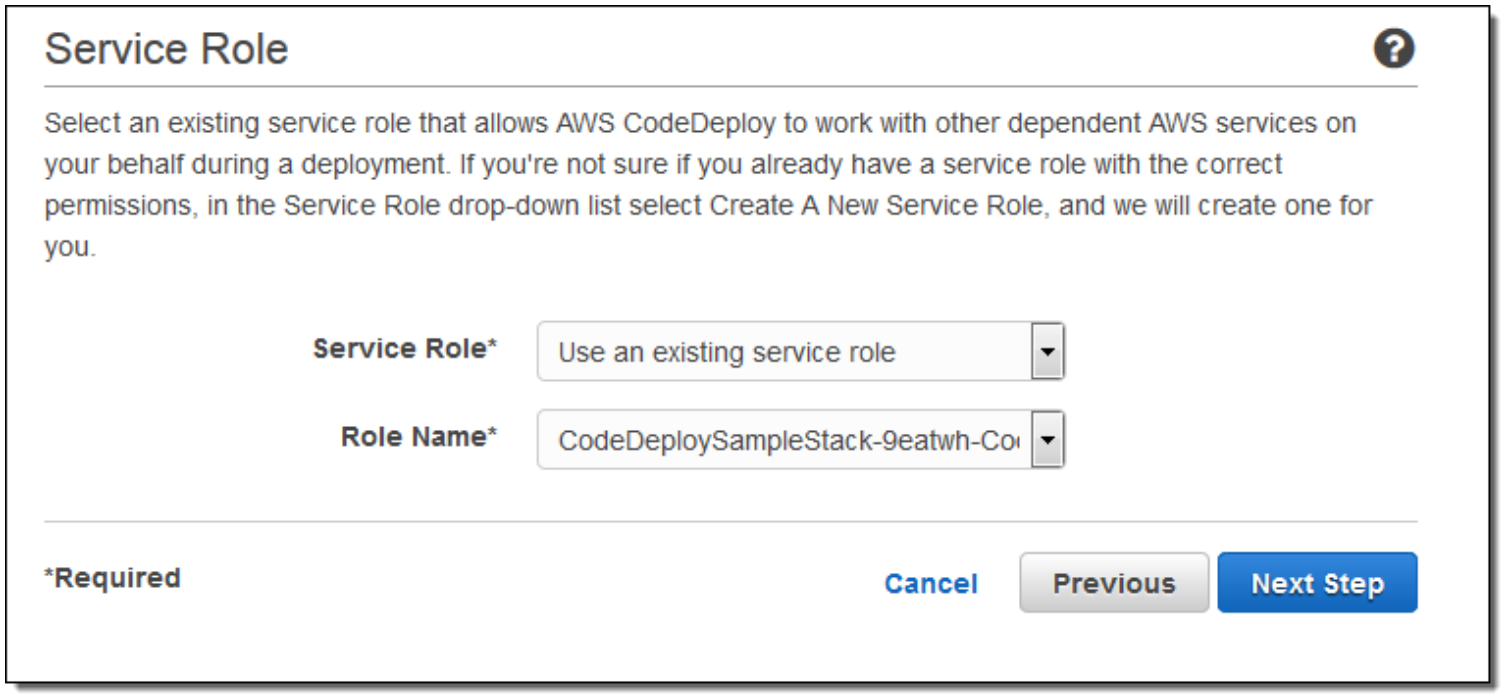


- Create a versioned revision for deployment.

  In this example the revision is stored in S3 but it could also come from CodeCommit or GitHub

Limited regions

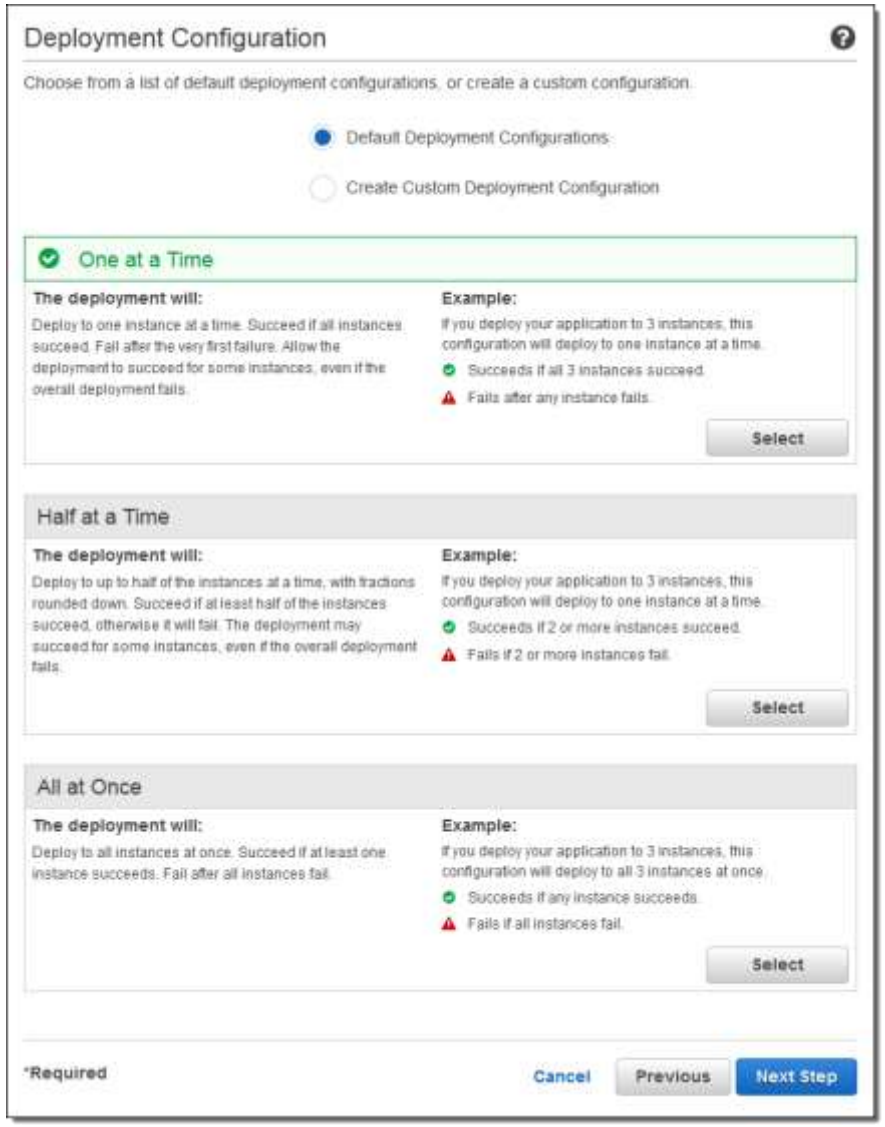
Using AWS CodeDeploy



- Define the IAM role to be used when interacting with other AWS services such as EC2 or Auto Scaling

Limited regions

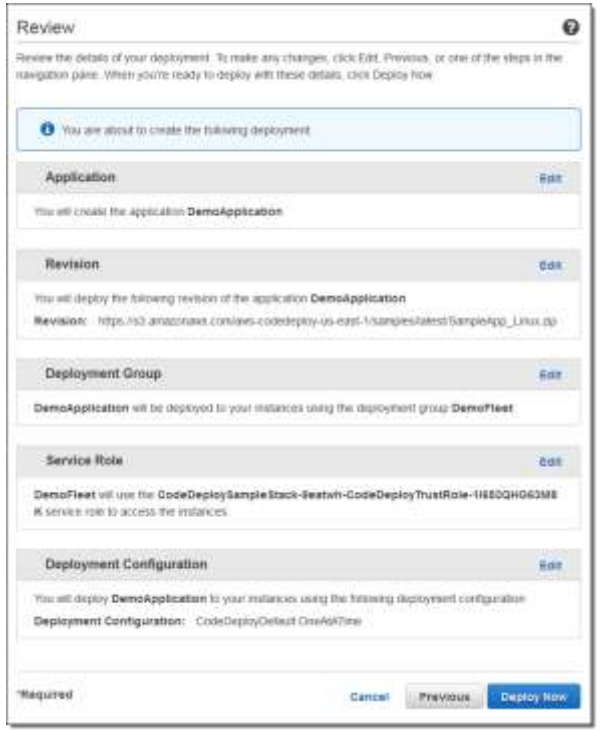
Using AWS CodeDeploy



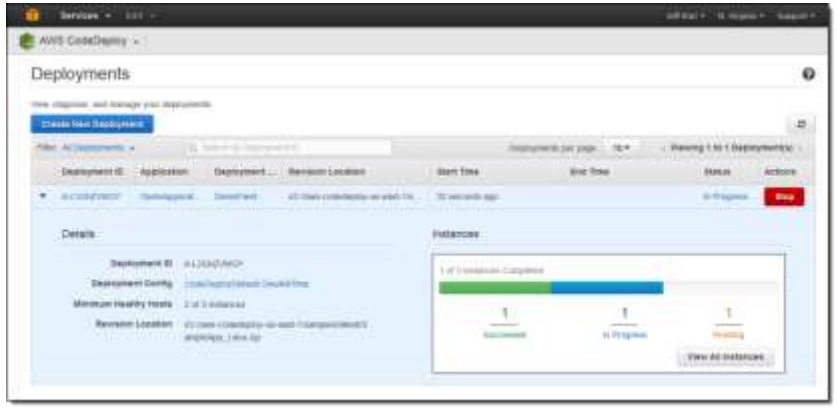
- Create a new Deployment Configuration or select from one of the defaults.

Limited regions

Using AWS CodeDeploy



- Review your settings and deploy.



- Deployment progress will be displayed in the AWS Management Console.

https://secure.flickr.com/photos/dullhunk/202872717/