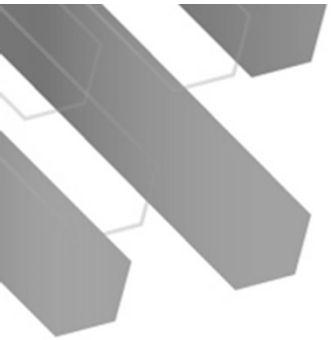


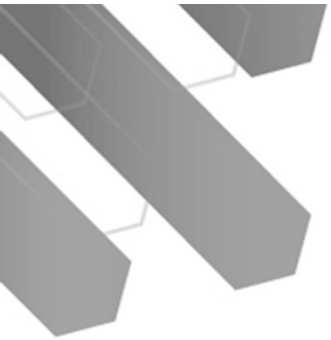
Deploying Applications on AWS

Module overview

- Part 1: Introducing DevOps
- Part 2: Using AWS code services for Continuous integration/continuous delivery *(*CI/CD)
- Part 3: Introducing deployment and testing strategies
- Part 4: Deploying applications with AWS Elastic Beanstalk
- Part 5: Deploying applications with AWS CloudFormation
- Part 6: Deploying serverless applications with the AWS Serverless Application Model (SAM)

Module objectives

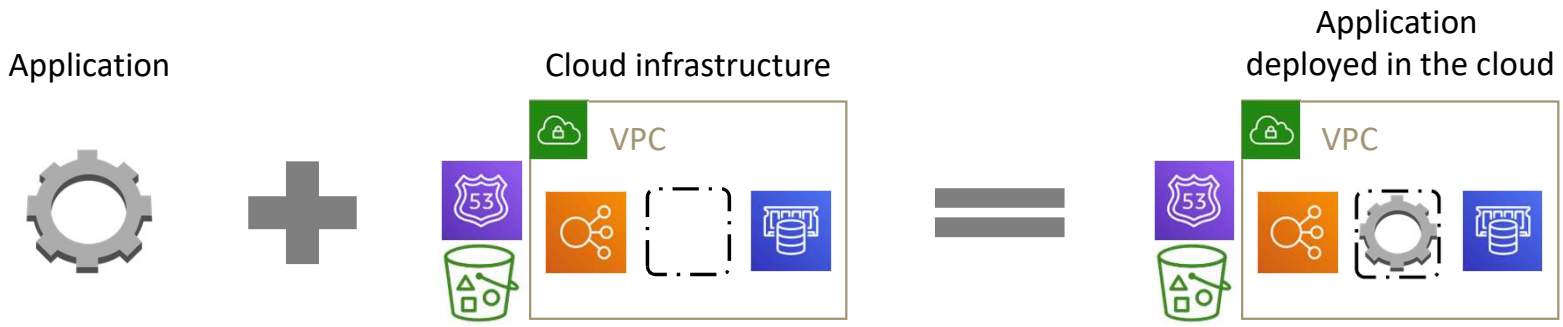
- Describe DevOps
- Recognize AWS code services for CI/CD
- Summarize deployment strategies
- Describe how AWS Elastic Beanstalk is used to deploy applications
- Describe how AWS CloudFormation is used to deploy applications
- Describe how AWS SAM is used to deploy serverless applications



Part 1: Introducing DevOps

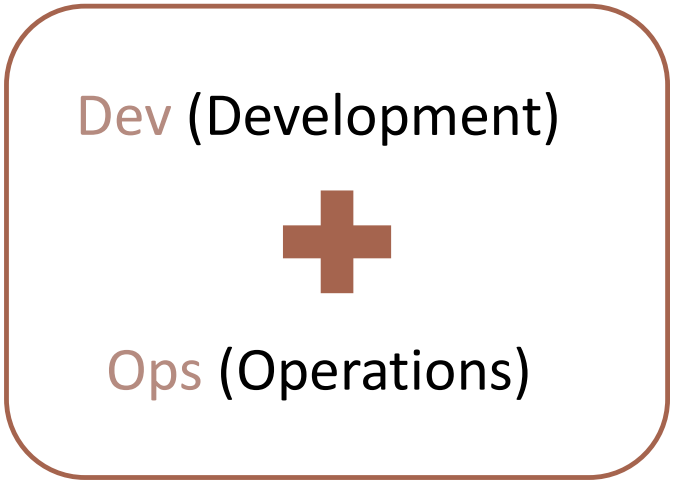
Deploying applications + infrastructure

- In the cloud, your application is not just your application. Your application is the application plus all the associated infrastructure.
- This way of approaching applications is supported by DevOps.
- DevOps is a combination of cultural philosophies, practices, and tools.



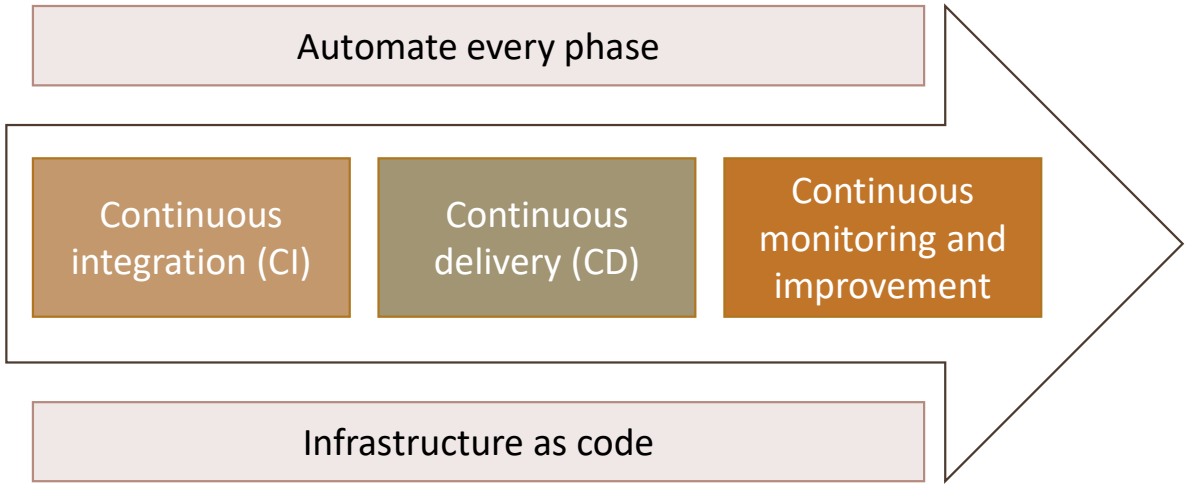
DevOps cultural philosophies

- Motto: People over Process over Tools
- Remove barrier between development and operations
- Shared responsibility



DevOps practices

- Microservice architecture
- Continuous integration/ continuous delivery (CI/CD)
- Continuous monitoring and improvement
- Automate everything
- Infrastructure as code



DevOps tools

CI/CD

- AWS CodeCommit
- AWS CodePipeline
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodeStar

Microservices

- Amazon Elastic Container Service
- AWS Lambda

Platform as a Service

- AWS Elastic Beanstalk

Infrastructure as Code

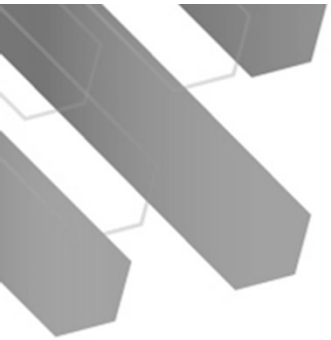
- AWS CloudFormation
- AWS OpsWorks
- AWS Systems Manager

Monitoring and logging

- Amazon CloudWatch
- AWS CloudTrail
- AWS X-Ray
- AWS Config

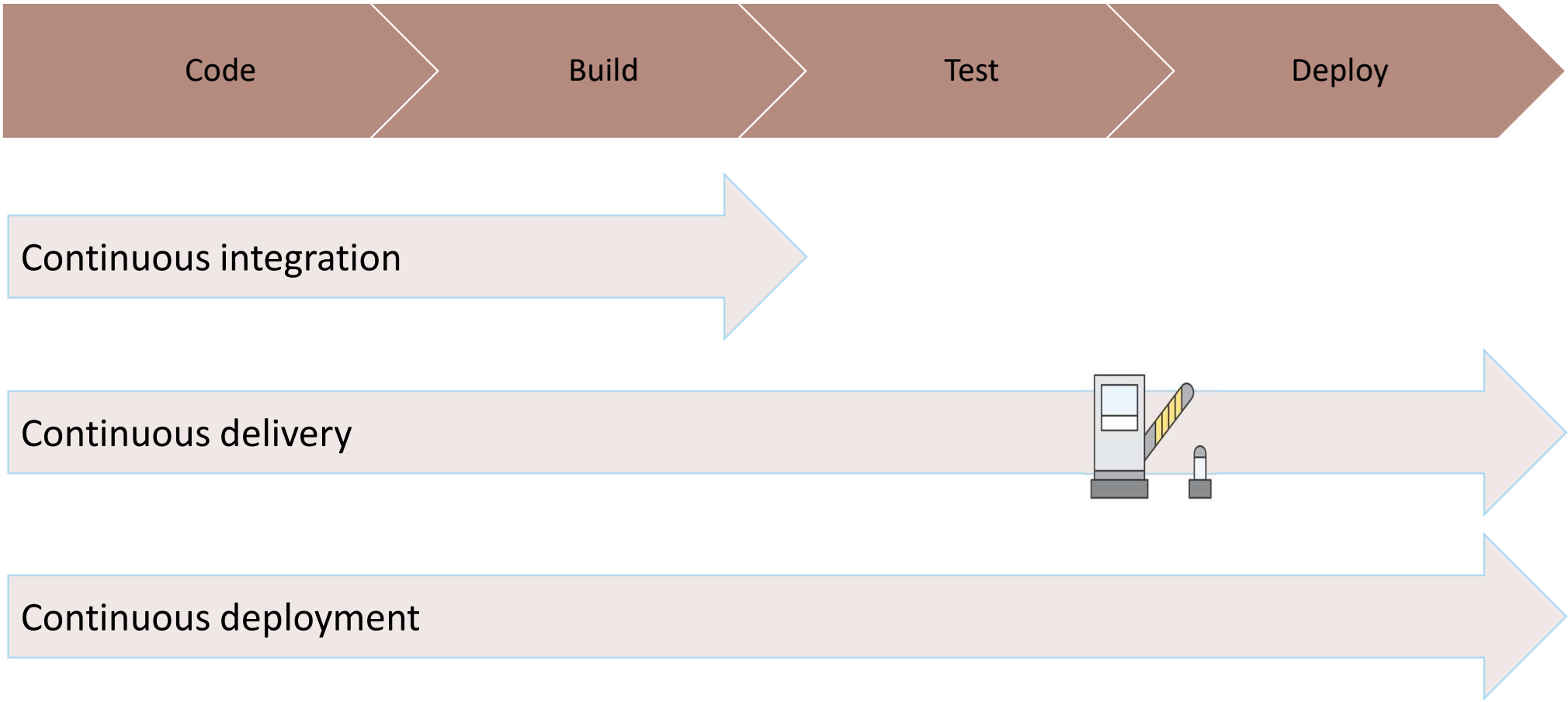
Benefits of DevOps

Improved collaboration	Rapid delivery	Scalable
Secure	Reliable	Maintainable

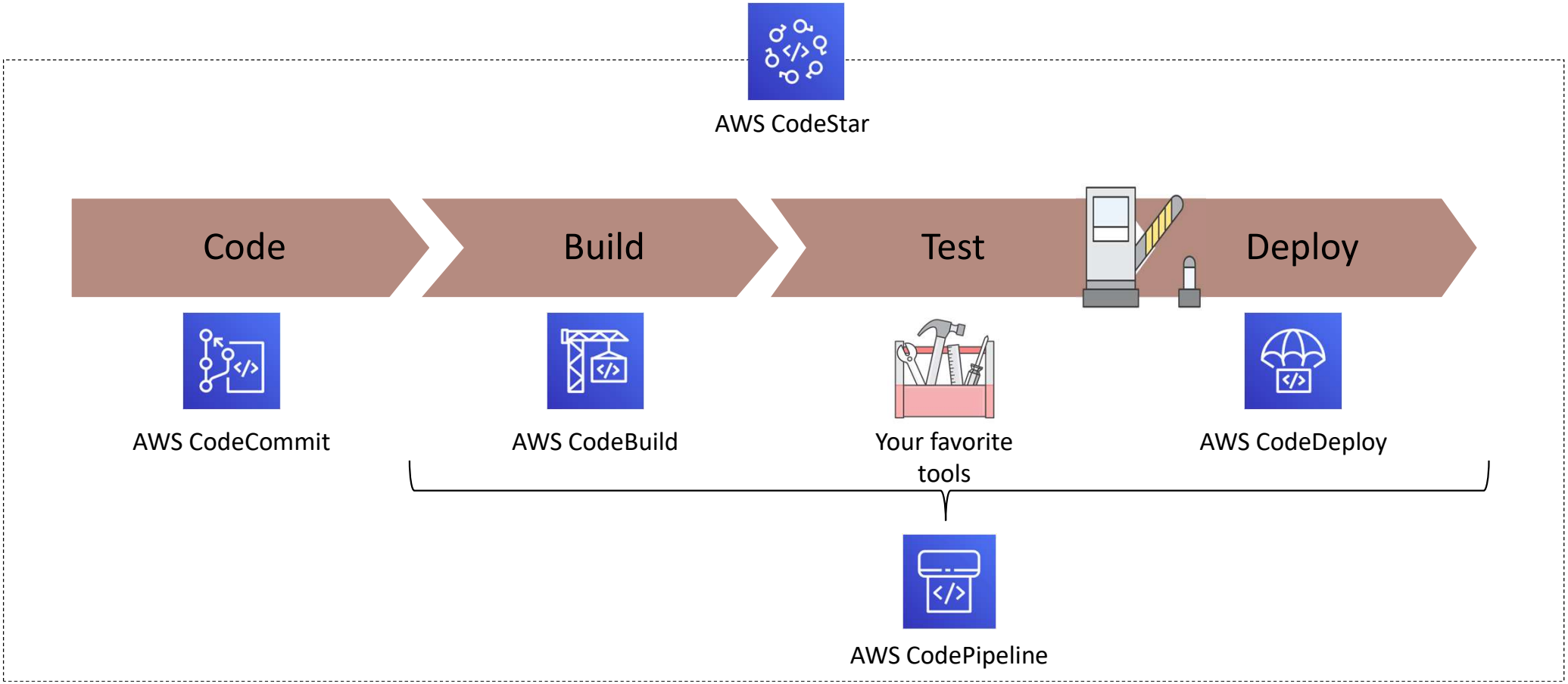


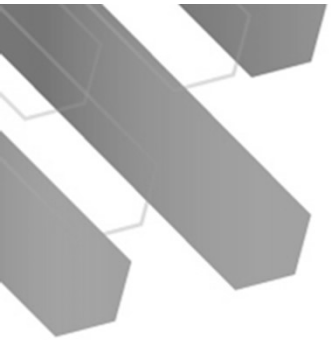
Part 2: Using AWS code services for CI/CD

Understanding CI/CD



CI/CD with AWS code services

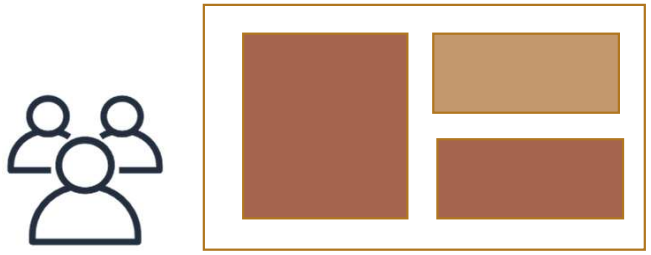




Part 3: Introducing testing and deployment strategies

A/B testing

Test new features on random subset of users to gain insight into usability, popularity, noticeability, and so forth



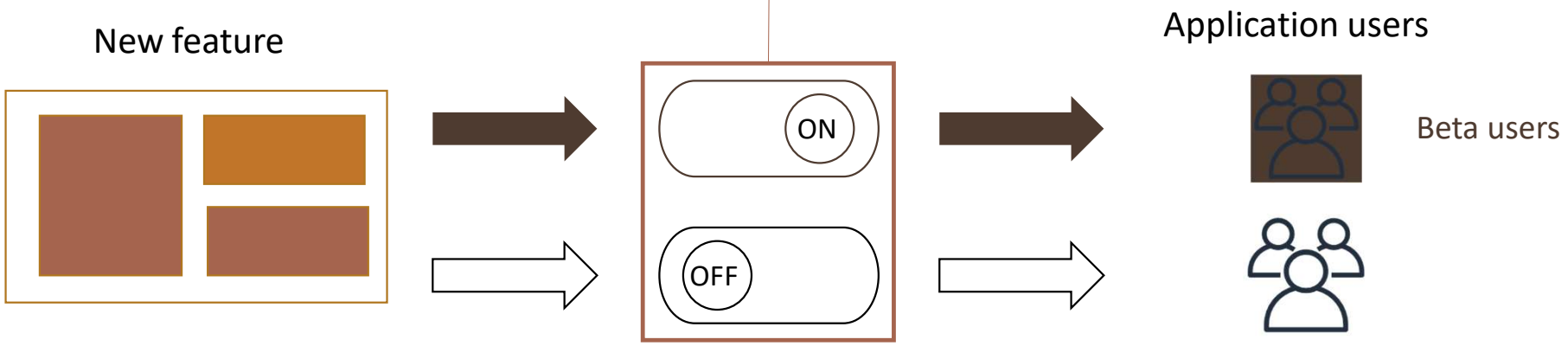
70% of application users see original version

30% of application users see new version

Feature flags

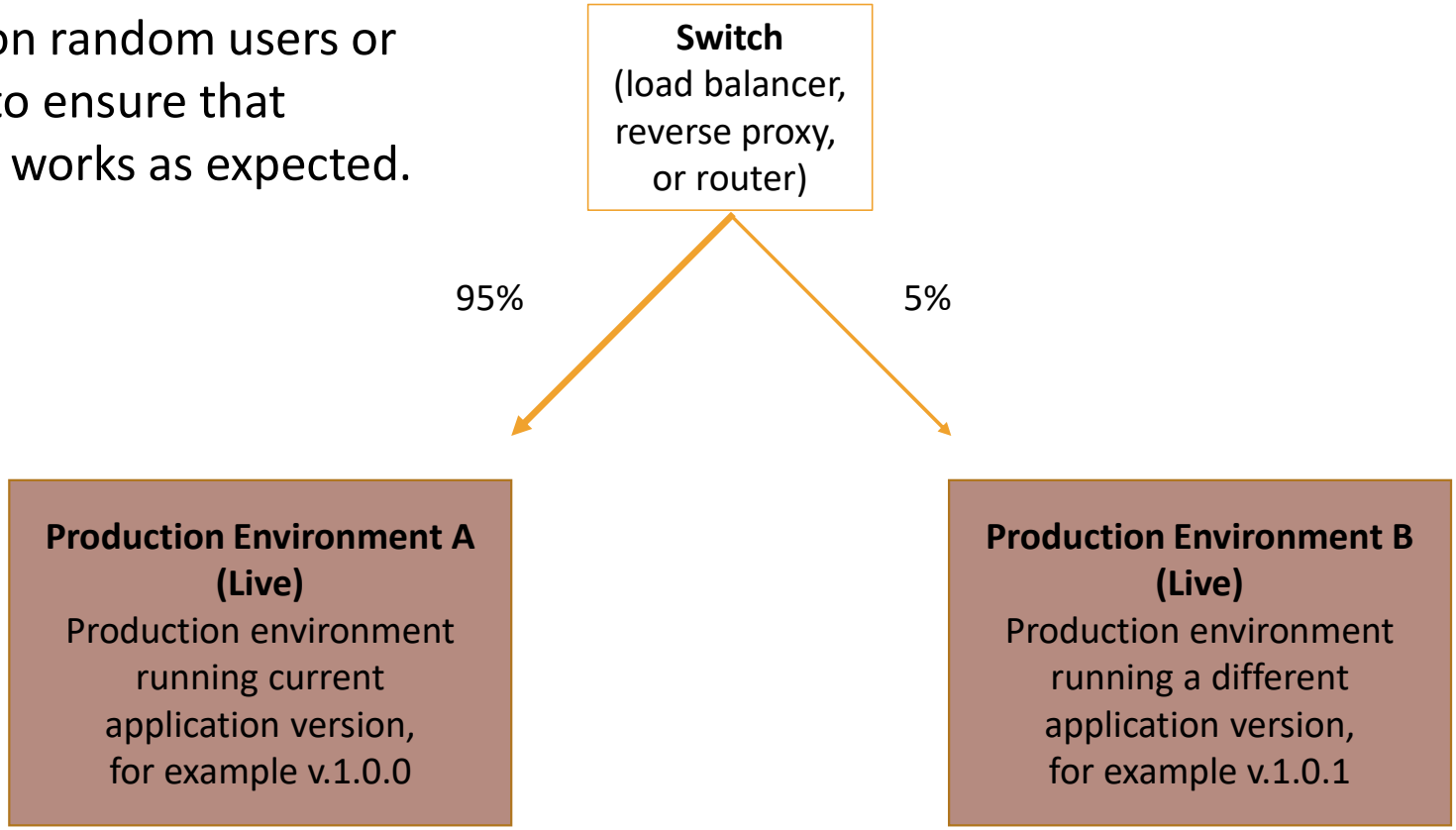
Feature flag

```
if (one.click.checkout, (key: "bob@example.com",
                         groups: ["gold", google"],
                         inBeta:true)
   )
    then
        /* show the one-click checkout feature */
    else
        /* show the old feature */
```



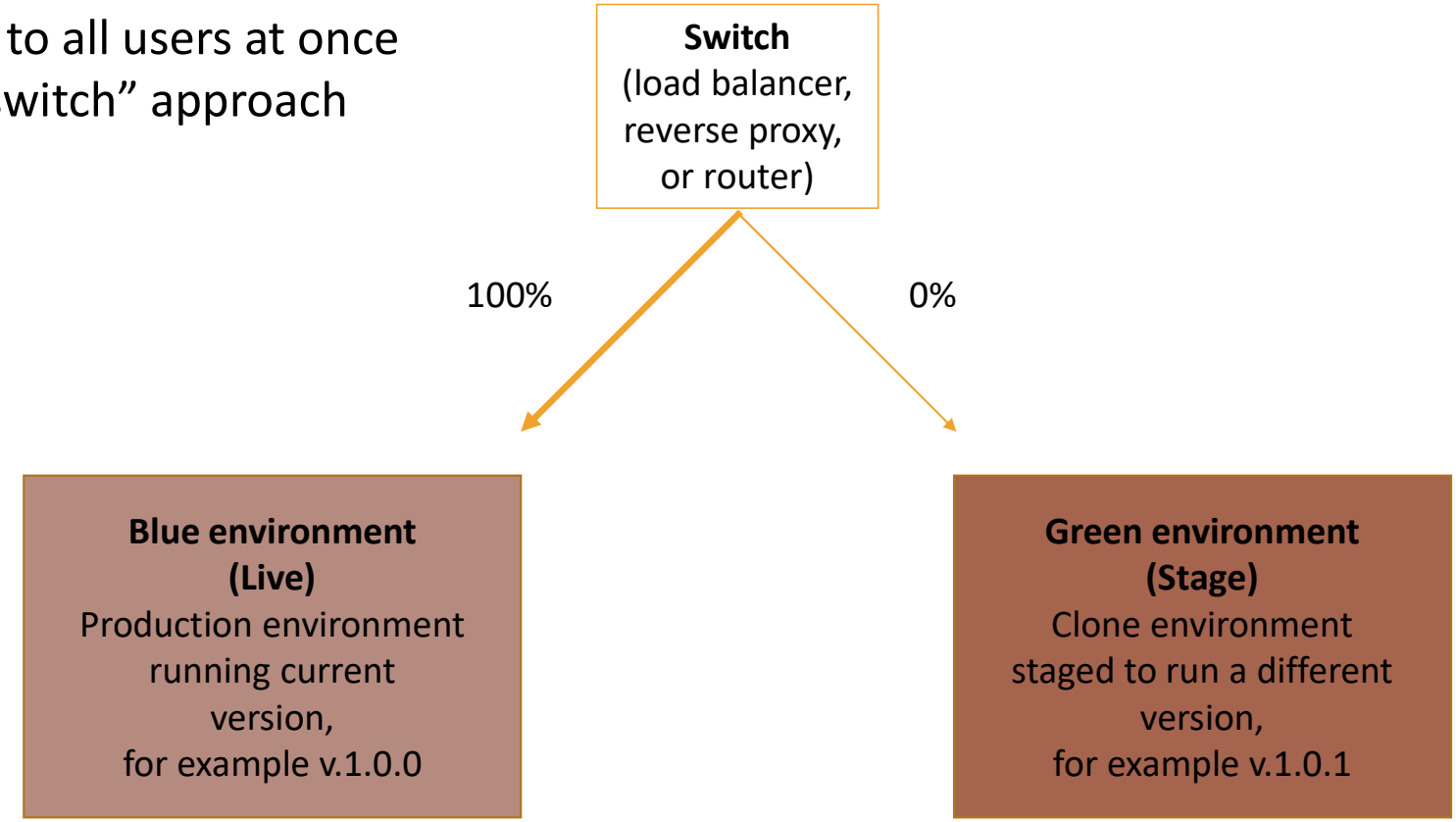
Canary testing

Test code on random users or machines to ensure that everything works as expected.

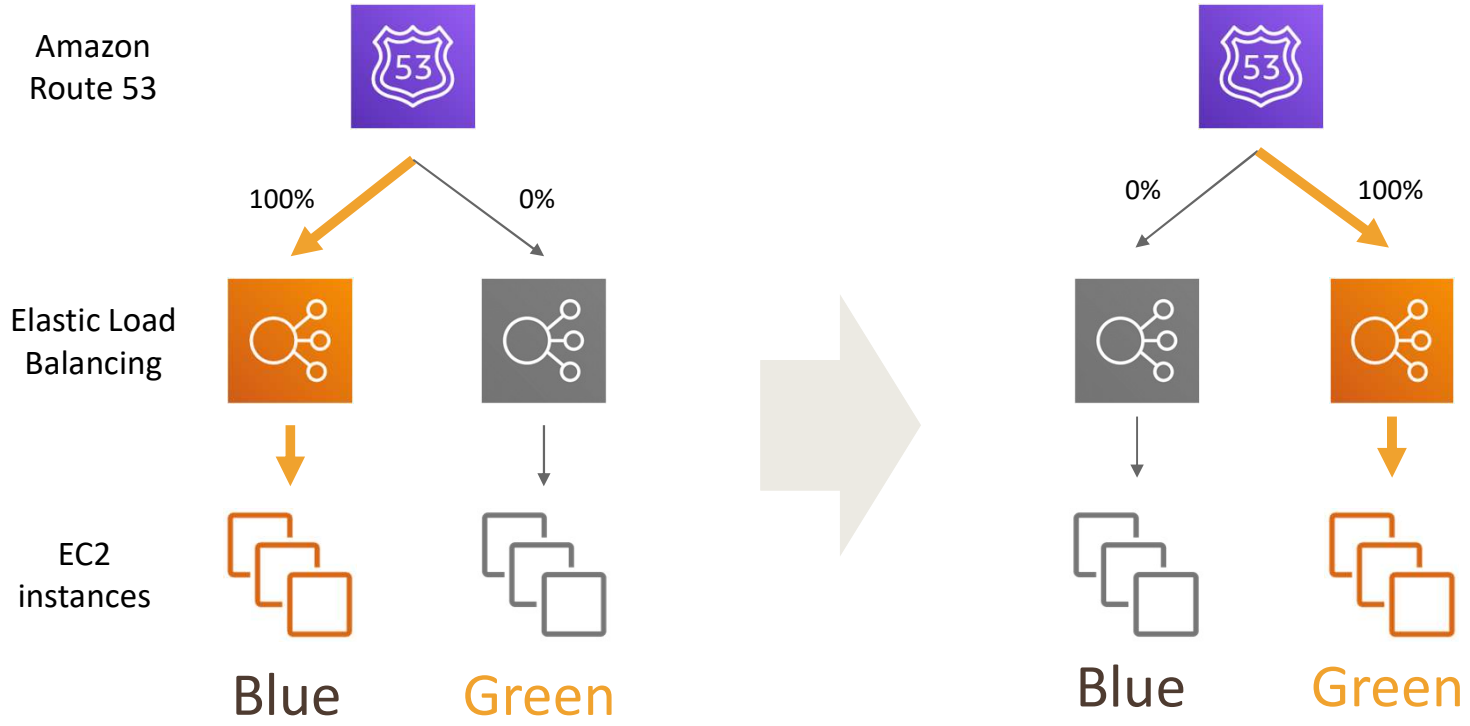


Blue/green deployment

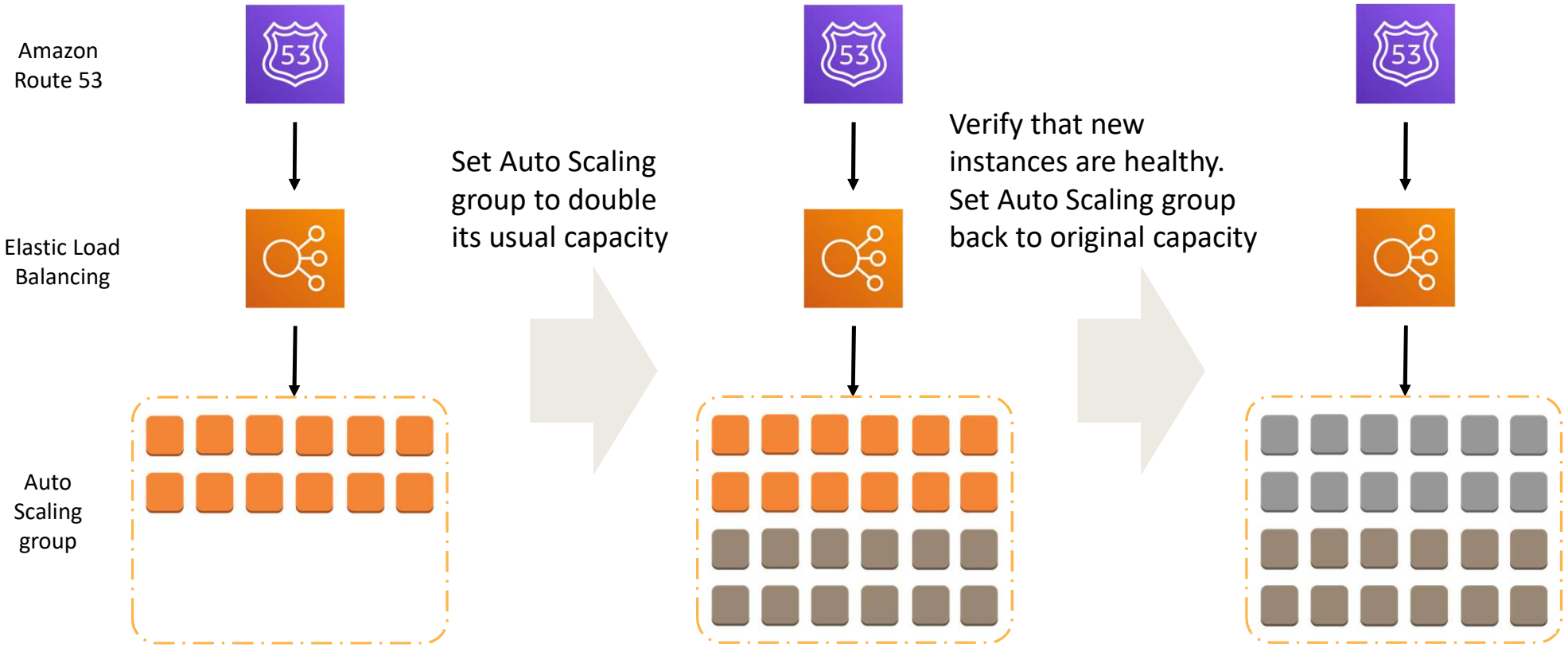
- Deploy to all users at once
- “Light switch” approach



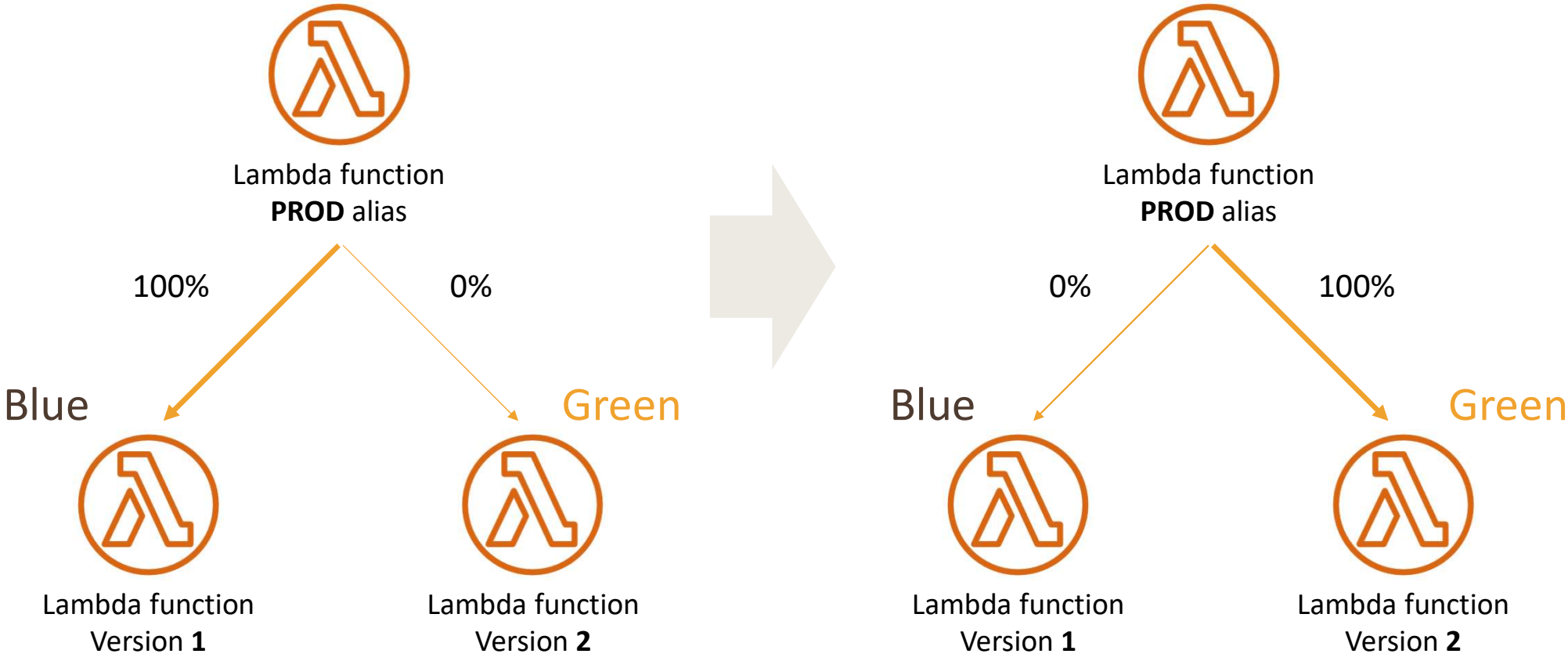
Blue/green deployment on AWS with Amazon Route 53



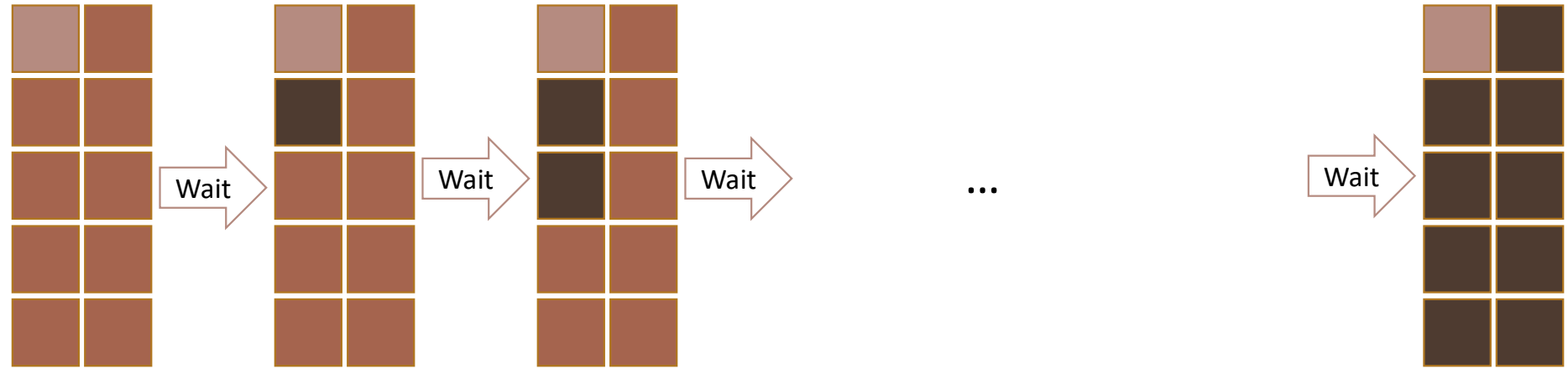
Blue/green deployment on AWS with Amazon EC2 Auto Scaling

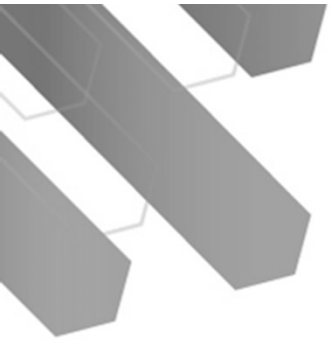


Blue/green deployment of serverless applications



Rolling deployment





Part 4: Deploying applications with AWS Elastic Beanstalk

AWS Elastic Beanstalk

AWS Elastic Beanstalk

Designed to let developers deploy code

Handles the details for you

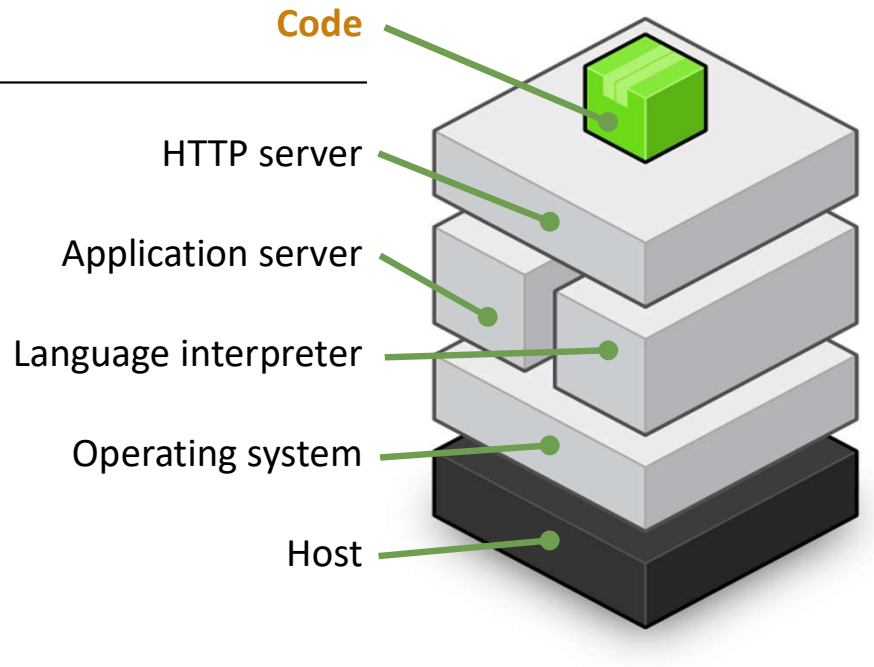
Compatible with

- PHP
- Java
- Python
- Ruby
- Node.js
- .NET
- Go
- Docker applications

Elastic Beanstalk: Resource provisioning

Developer

AWS Elastic Beanstalk



AWS Elastic Beanstalk components

Environment

- Tiers
  - Web server
  - Worker
- Type
  - Single EC2 instance
  - Auto Scaling
- One application per environment—can have multiple environments

Application Versions

- Application code, stored in Amazon S3
- Deploy different versions to different environments
- Multiple versions available to support rollback
- Support for rolling updates

Configuration

- Configure the individual services used by AWS Elastic Beanstalk
- Install additional yum packages and supply your own daemon configuration files

Elastic Beanstalk environments

My application

V1.2

Test environment

PHP 5.5

AMI ID
ami-abcd2222

V1.1

Staging environment

PHP 5.4

AMI ID
ami-abcd1111

Production environment

PHP 5.4

AMI ID
ami-abcd1111

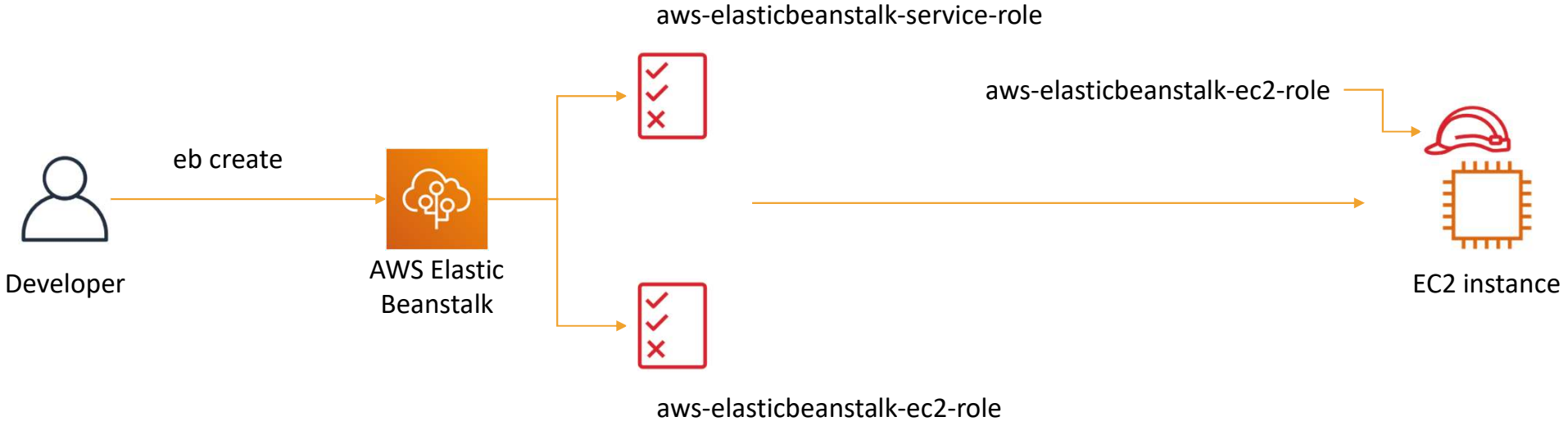
Elastic Beanstalk CLI

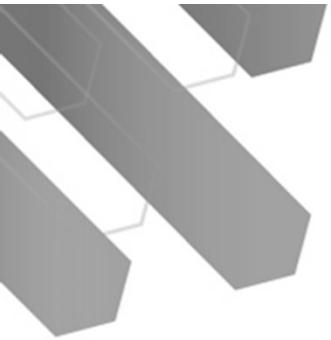
- Use the Elastic Beanstalk CLI to rapidly deploy and test your in-development application
  - `eb init`
  - `eb create`
  - `eb deploy`
- Use `eb deploy` to re-deploy revisions to your application

Elastic Beanstalk permission model

AWS Elastic Beanstalk requires two IAM roles:

- Service role: Permission for AWS Elastic Beanstalk to create and manage resources in your account
- Instance profile: AWS permissions for the EC2 instance itself





Part 5: Deploying applications with AWS CloudFormation

Infrastructure as code

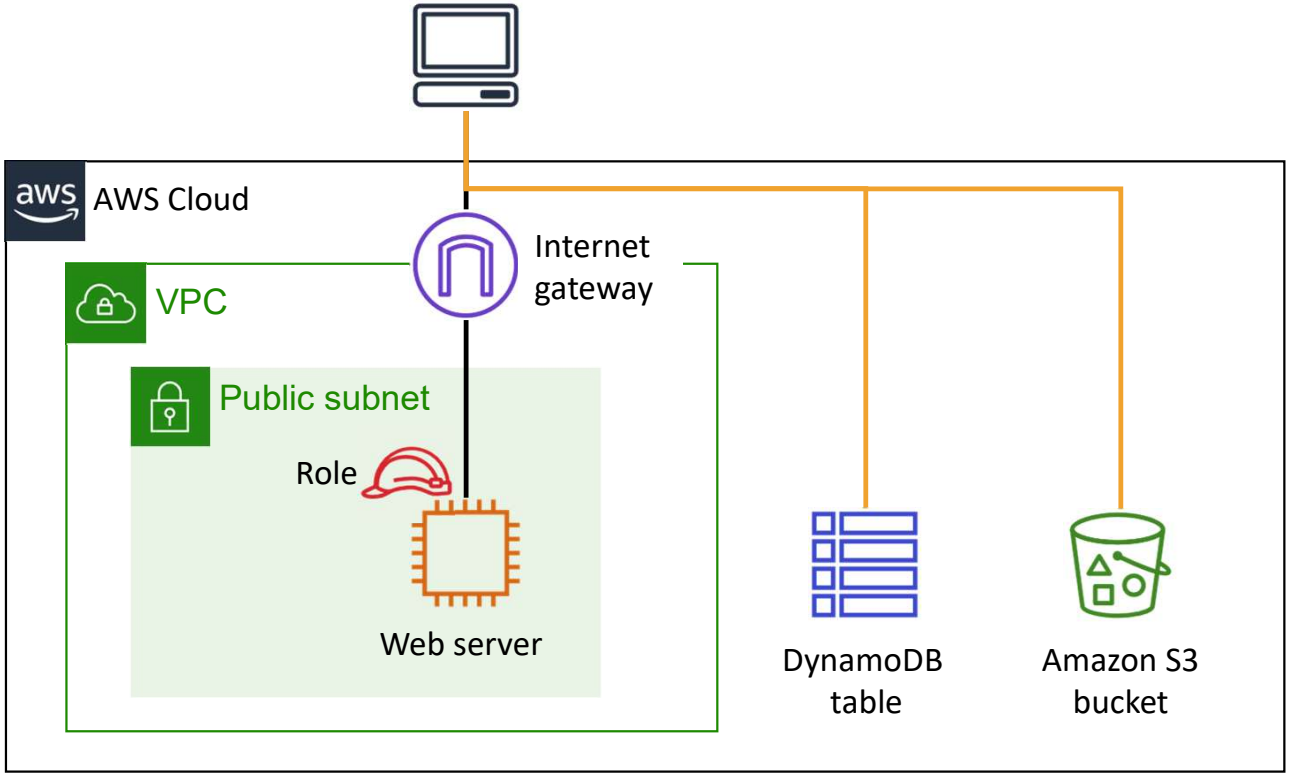
- Method for automating the process of creating, updating, and deleting AWS infrastructure
- Stand up identical dev/test environments on demand
- Use the same code to create your production environment that you used to create your other environments

AWS CloudFormation

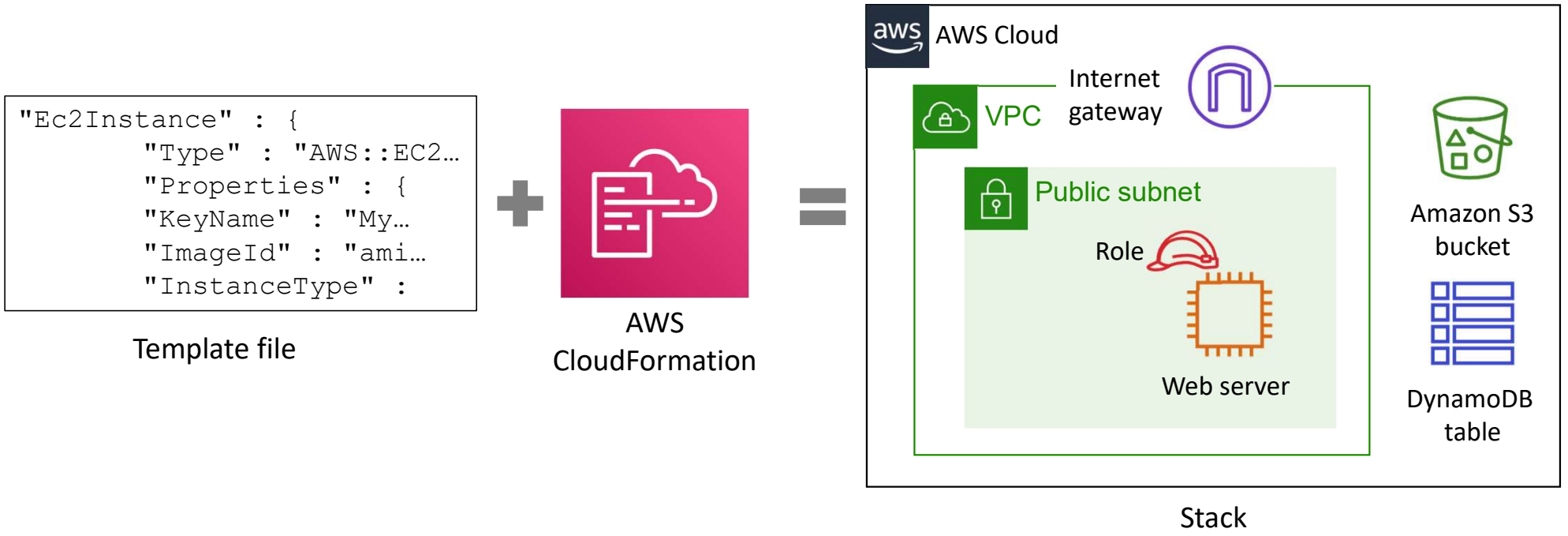
AWS CloudFormation

- Fully managed service
- Creates, updates, and deletes resources in stacks
- Automates AWS resource provisioning
- Simplifies task of repeatedly and predictably creating groups of related resources that power your applications

Automated provisioning of AWS resources

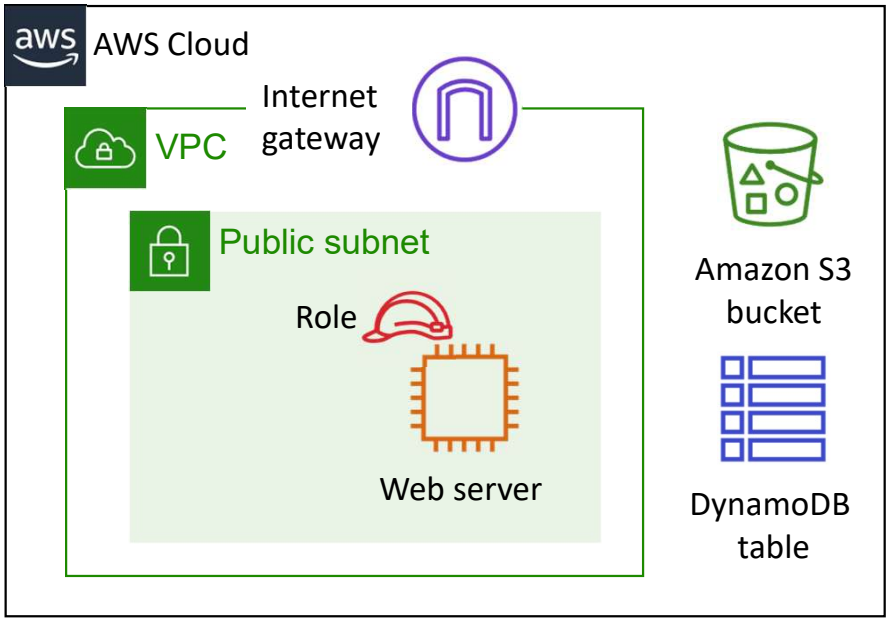


How AWS CloudFormation works



AWS CloudFormation stacks

- Resources generated
- Unit of deployment
- Create stack
- Update stack
- Delete stack



Stack

AWS CloudFormation templates

- Resources to provision
- Text file
- JSON or YAML format
- Self-documenting environment

```
"Ec2Instance" : {
        "Type" : "AWS::EC2…
        "Properties" : {
        "KeyName" : "My…
        "ImageId" : "ami…
        "InstanceType"
```

Template file

Each template is an example of infrastructure as code, which means you control your infrastructure through software.

CloudFormation template structure

```
{
  "AWSTemplateFormatVersion" : "version date",
  "Description" : "JSON string",
  "Metadata" : {template metadata},
  "Parameters" : {set of parameters},
  "Mappings" : {set of mappings},
  "Conditions" : {set of conditions},
  "Transform" : {set of transforms},
  "Resources" : {set of resources},
  "Outputs" : {set of outputs}
}
```

2010-09-09 is the latest version

Required

Question #1

What happens, by default, when one of the resources in a CloudFormation stack cannot be created?

A. Previously-created resources are kept but the stack creation terminates.

B. Previously-created resources are deleted and the stack creation terminates.

C. The stack creation continues, and the final results indicate which steps failed.

D. CloudFormation templates are parsed in advance so stack creation is guaranteed to succeed.

Question #1

What happens, by default, when one of the resources in a CloudFormation stack cannot be created?

A. Previously-created resources are kept but the stack creation terminates.

B. Previously-created resources are deleted and the stack creation terminates.

C. The stack creation continues, and the final results indicate which steps failed.

D. CloudFormation templates are parsed in advance so stack creation is guaranteed to succeed.

Question #2

You have an environment that consists of a public subnet using Amazon VPC and 3 instances that are running in this subnet. These three instances can successfully communicate with other hosts on the Internet. You launch a fourth instance in the same subnet, using the same AMI and security group configuration you used for the others, but find that this instance cannot be accessed from the Internet.

What should you do to enable internet access?

A. Deploy a NAT instance into the public subnet.

B. Modify the routing table for the public subnet

C. Configure a publically routable IP Address In the host OS of the fourth instance.

D. Assign an Elastic IP address to the fourth instance.

Question #2

You have an environment that consists of a public subnet using Amazon VPC and 3 instances that are running in this subnet. These three instances can successfully communicate with other hosts on the Internet. You launch a fourth instance in the same subnet, using the same AMI and security group configuration you used for the others, but find that this instance cannot be accessed from the Internet.

What should you do to enable internet access?

A. Deploy a NAT instance into the public subnet.

B. Modify the routing table for the public subnet

C. Configure a publically routable IP Address In the host OS of the fourth instance.

D. Assign an Elastic IP address to the fourth instance.

Question #3

Which of the following are valid arguments for an SNS Publish request? (Choose three.)

A. TopicArn

B. Subject

C. Destination

D. Format

E. Message

F. Language

Question #3

Which of the following are valid arguments for an SNS Publish request? (Choose three.)

A. TopicArn

B. Subject

C. Destination

D. Format

E. Message

F. Language

Module review

- Introduced DevOps
- Introduced deployment and testing strategies
- Discussed how to deploy applications with:
  - AWS Elastic Beanstalk
  - AWS CloudFormation