AWS Academy Cloud Developing

Module 13: Automating Deployment Using CI/CD Pipelines

Module 13: Automating Deployment Using CI/CD Pipelines

Section 1: Introduction

Module objectives

At the end of this module, you should be able to do the following:

- Describe DevOps
- Recognize AWS code services for continuous integration and continuous delivery (CI/CD)
- Describe how AWS CloudFormation is used to deploy applications
- Describe how the AWS Serverless Application Model (AWS SAM) is used to deploy serverless applications

Module overview

Sections

1. Introduction
2. Introducing DevOps
3. Using AWS code services for CI/CD
4. Deploying applications with CloudFormation
5. Deploying serverless applications with AWS SAM

Lab

- Automating Application Deployment Using a CI/CD Pipeline

Knowledge check

Café business requirement

Sofía wants to make the release process less prone to human error. She was chatting with Mateo, an AWS SysOps engineer, and he explained that DevOps is a philosophy and a set of practices and tools that support cloud application development, including automation of the development and release process.

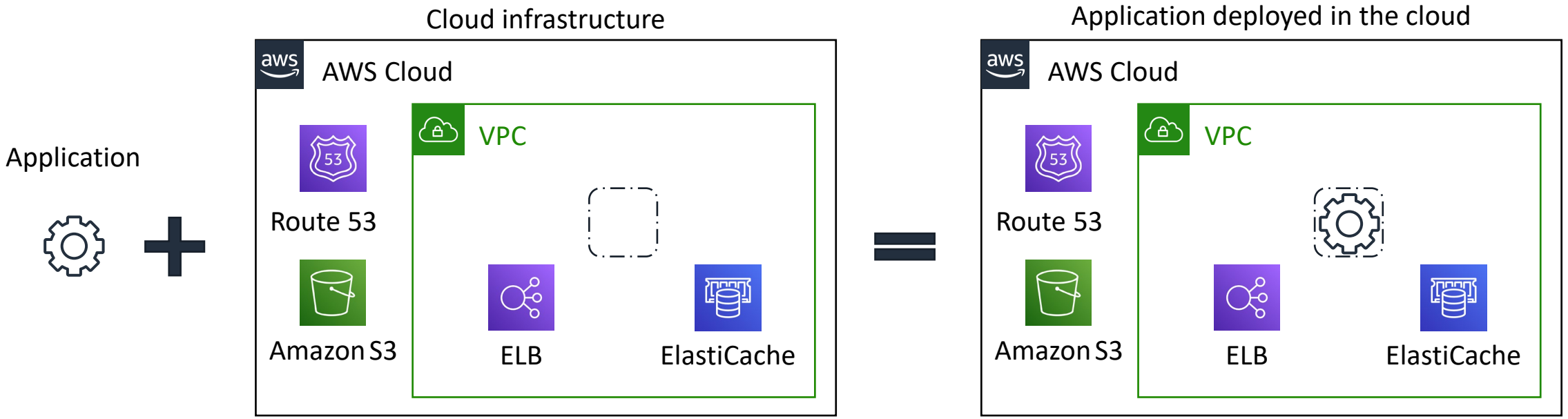
Module 13: Automating Deployment Using CI/CD Pipelines

Section 2: Introducing DevOps

aws academy

Deploying applications and infrastructure

- In the cloud, your application is not just your application. Your application is the application plus all of the associated infrastructure.
- DevOps is a combination of cultural philosophies, practices, and tools that support this method of application development.



Cultural philosophy of DevOps

- Motto: People over process over tools
- Remove barriers between development and operations
- Shared responsibility

Dev (Development)

+

Ops (Operations)

DevOps

Dev (Development)

+

Sec (Security)

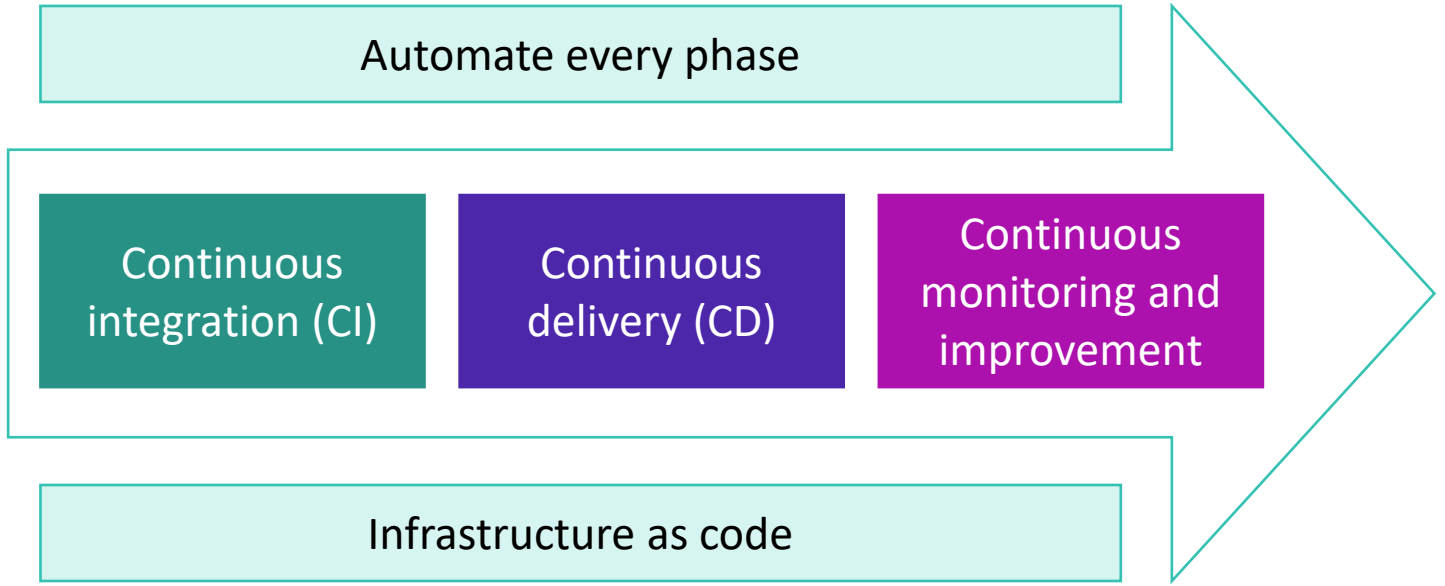
+

Ops (Operations)

DevSecOps

DevOps practices

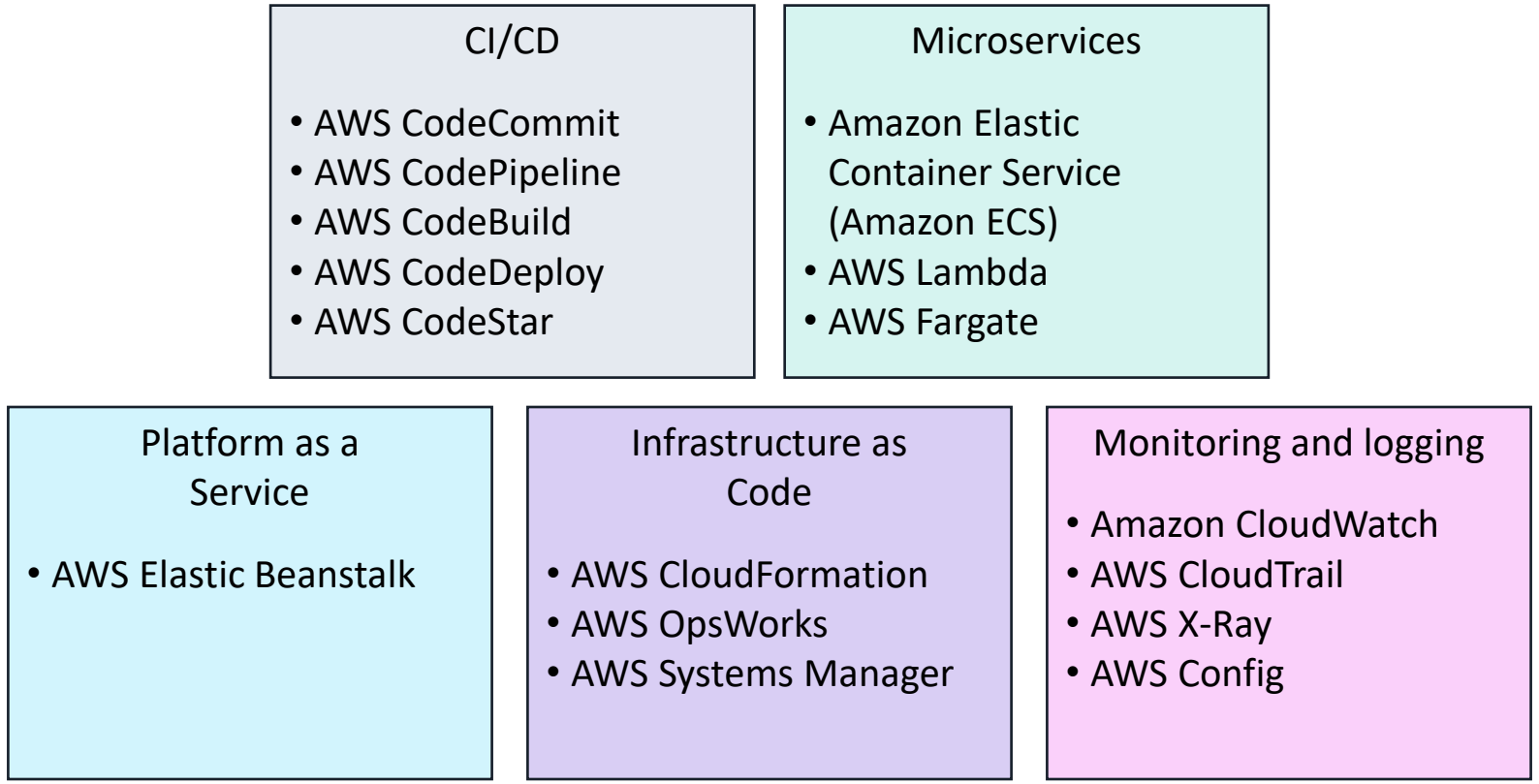
- Microservice architecture
- Continuous integration and continuous delivery (CI/CD)
- Continuous monitoring and improvement
- Automation focused
- Infrastructure as code



Benefits of DevOps

- Improved collaboration
- Rapid delivery
- Scalable
- Secure
- Reliable
- Maintainable

DevOps tools



In this module, you will learn about the AWS code services that support CI/CD, as well as the AWS CloudFormation service.

Section 2 key takeaways

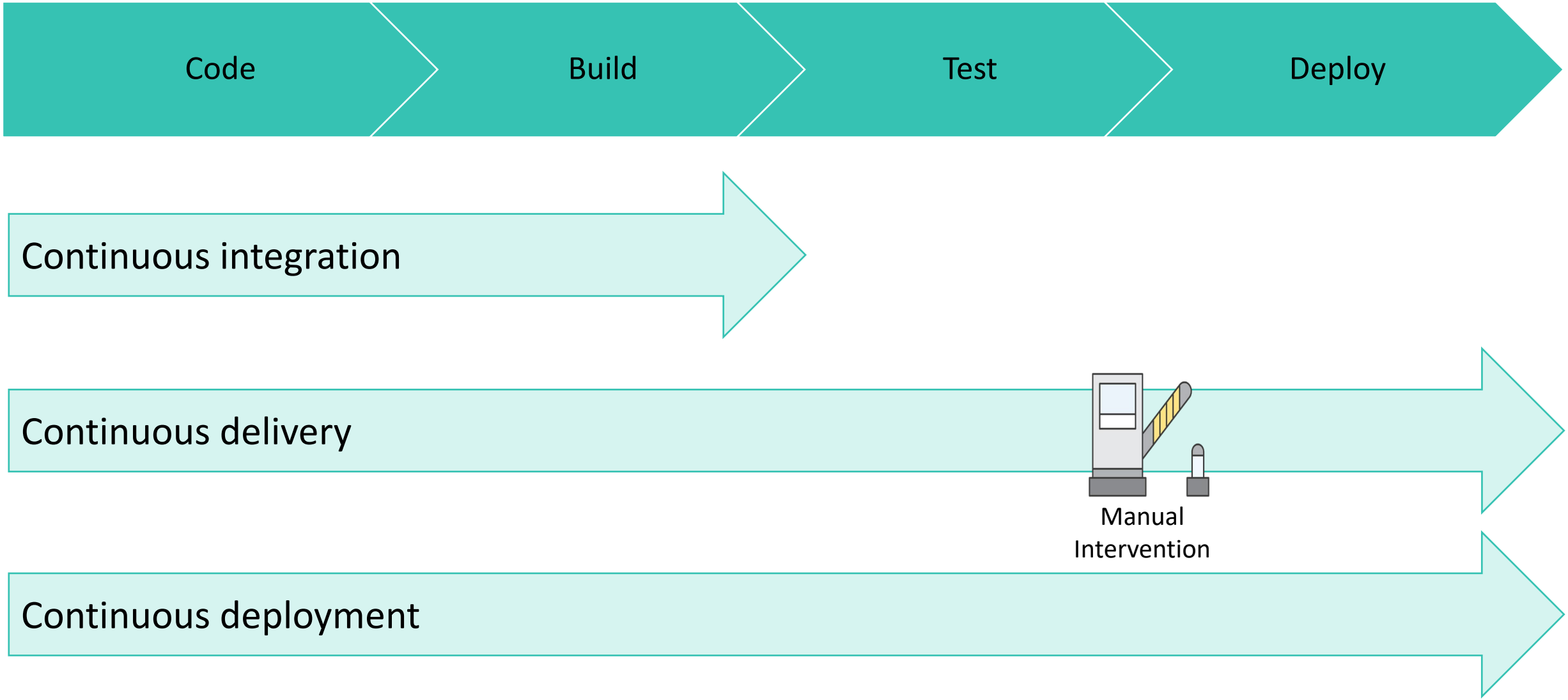
- In the cloud, your application consists of the application plus all of the associated infrastructure.
- DevOps is about removing the barrier between development and operations teams, and getting them to communicate with each other.
- DevOps practices result in code that is secure, reliable, and maintainable.

Module 13: Automating Deployment Using CI/CD Pipelines

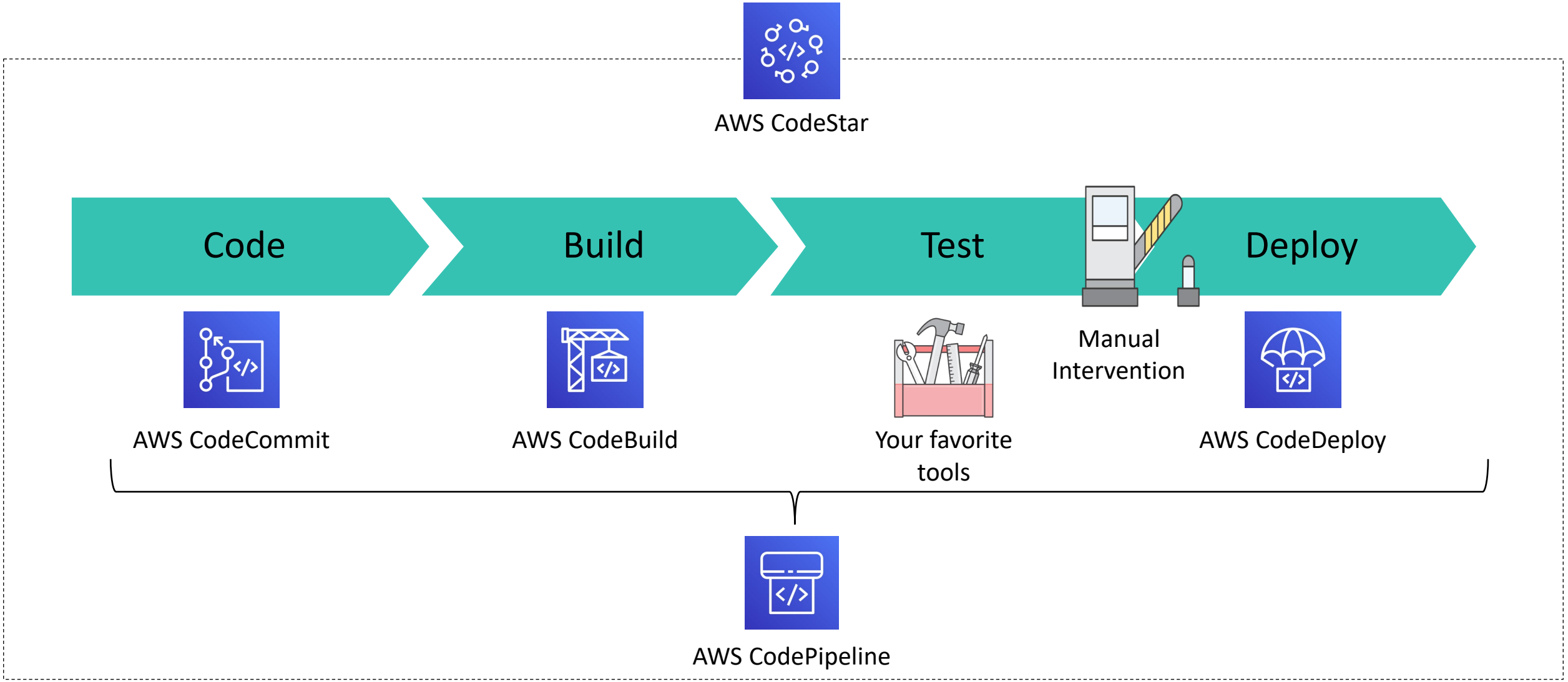
Section 3: Using AWS code services for CI/CD

aws academy

Understanding CI/CD



CI/CD with AWS code services



Section 3 key takeaways

- CI/CD spans the develop and deploy stages of the software development lifecycle.
- It is commonly thought that continuous integration stops at the build stage.
- Continuous delivery improves on continuous integration by helping teams to gain a greater level of certainty that their software will work in production.

Module 13: Automating Deployment Using CI/CD Pipelines

Section 4: Deploying applications with CloudFormation

aws academy

Infrastructure as code

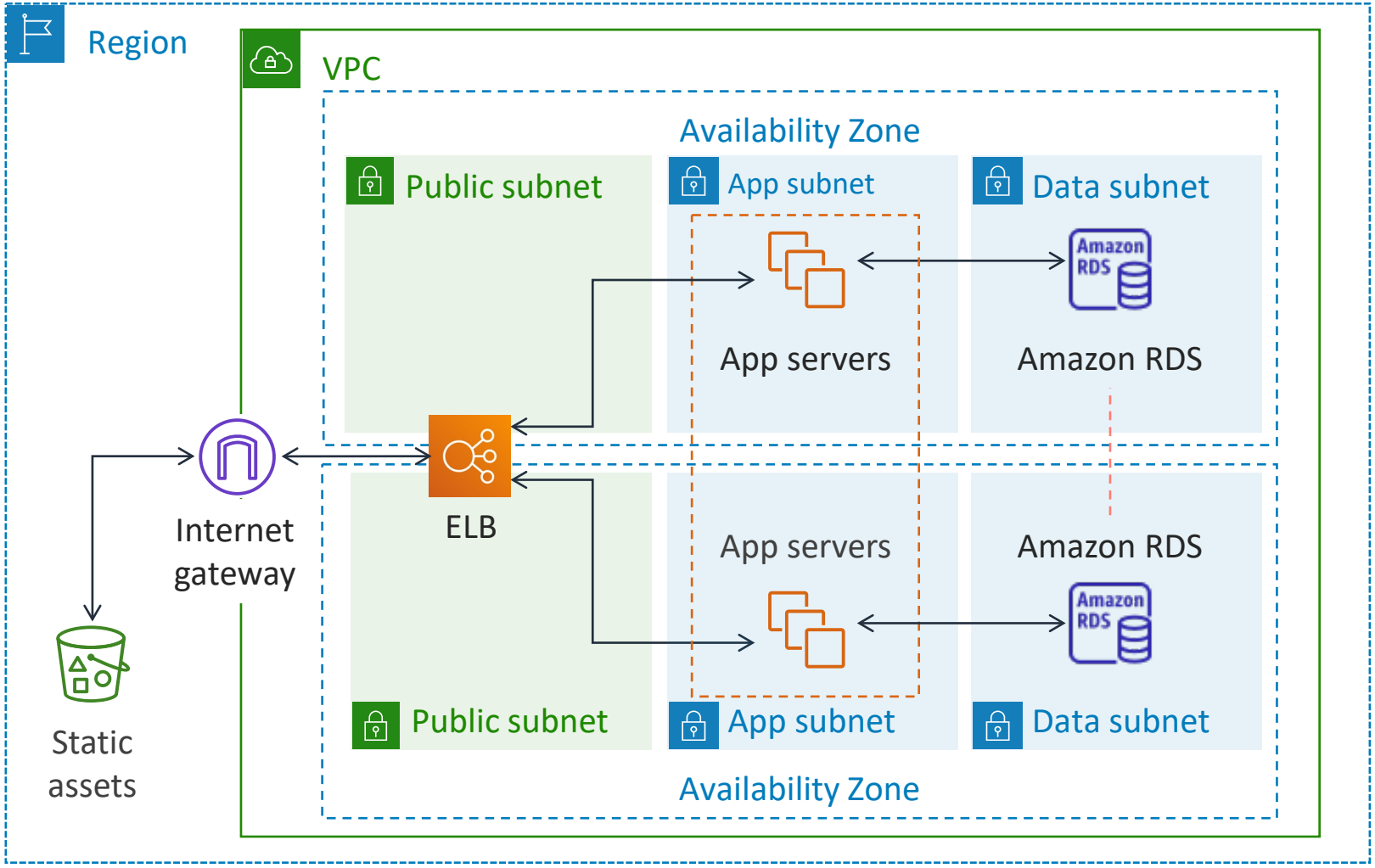
- Infrastructure as code is a method to automate the process of creating, updating, and deleting AWS infrastructure.
- Stand up identical dev/test environments on demand.
- Use the same code to create your production environment that you used to create your other environments.

AWS CloudFormation

AWS CloudFormation

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

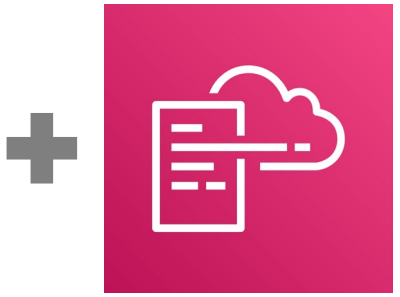
Automated provisioning of AWS resources



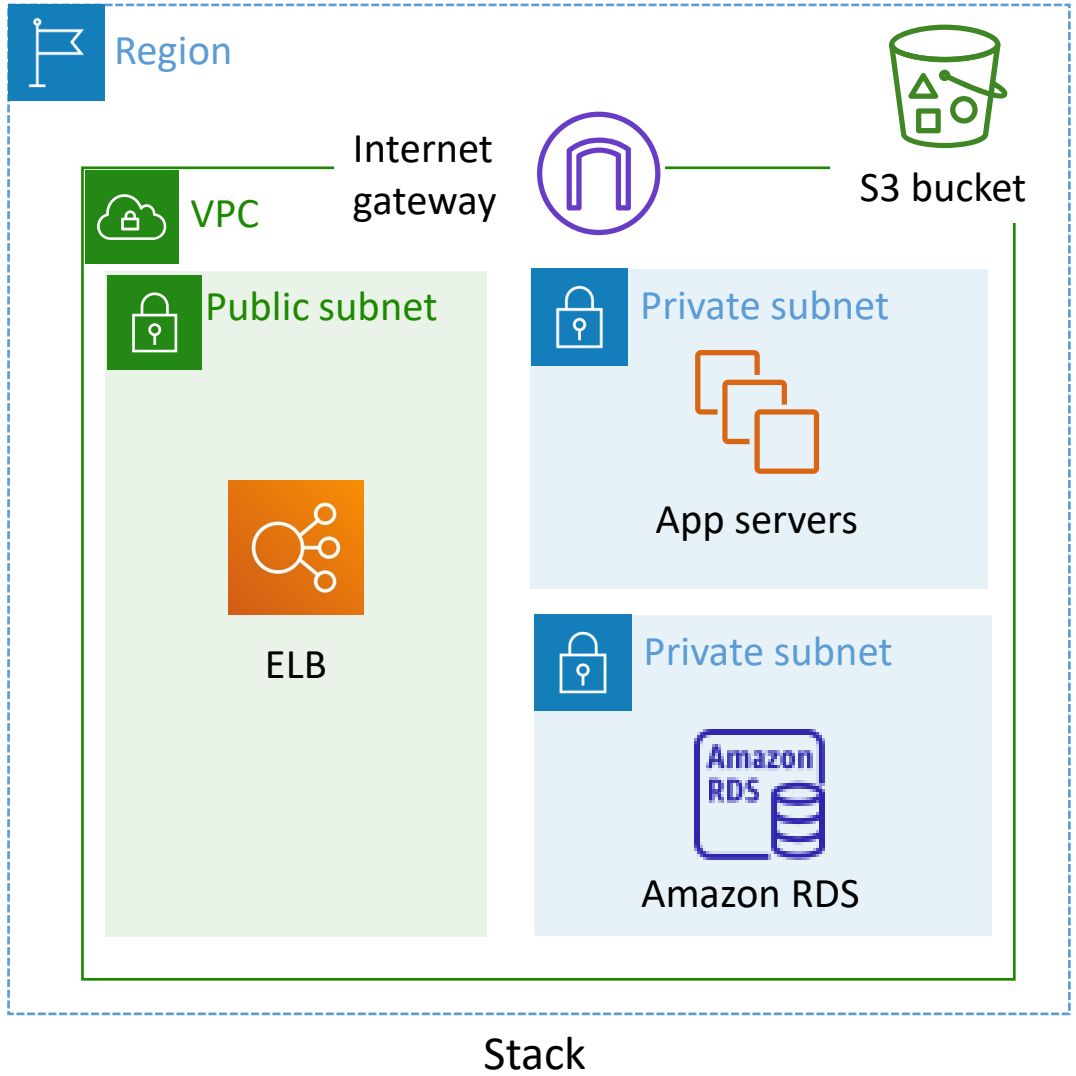
How CloudFormation works

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

Template file



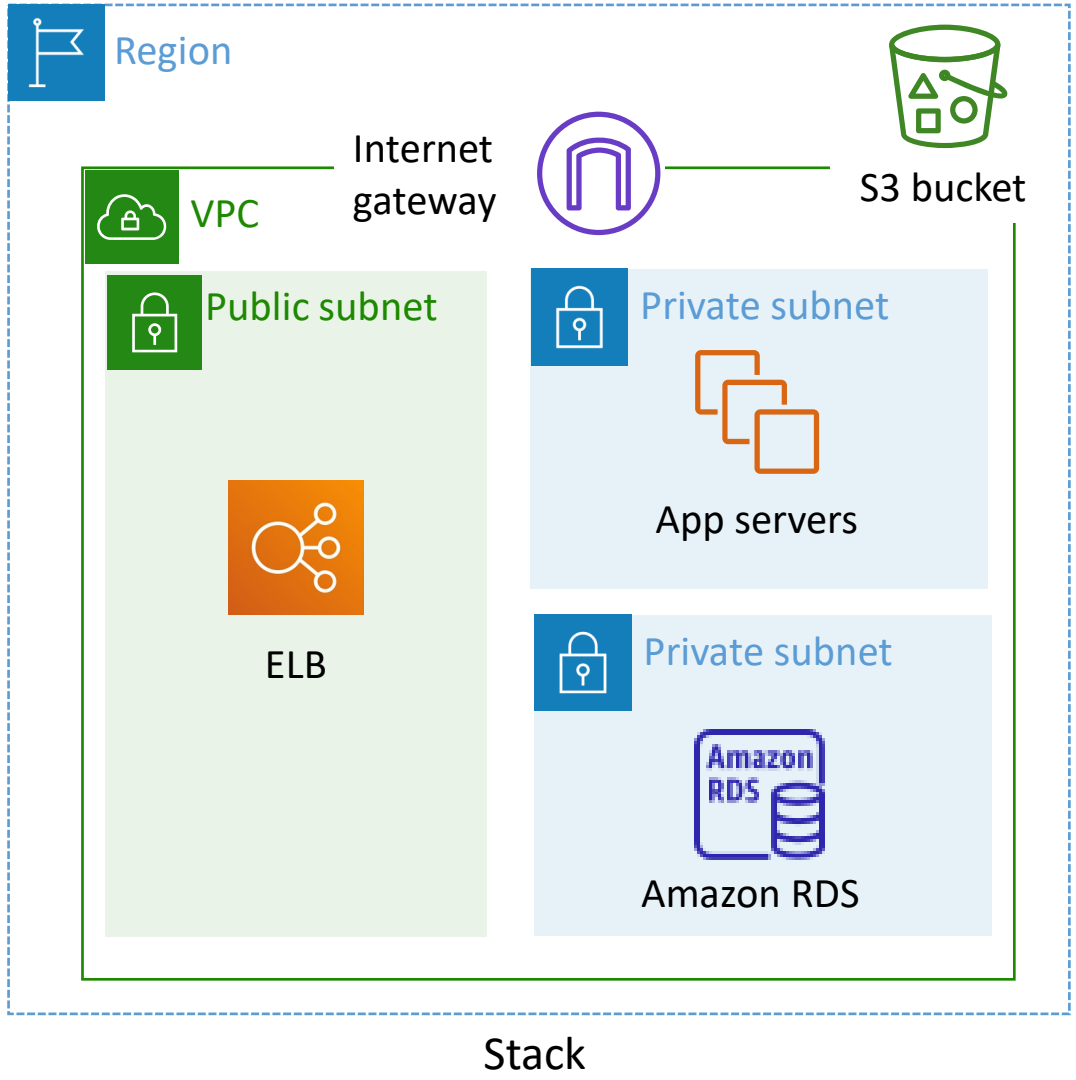
CloudFormation



Stack

CloudFormation stacks

- A CloudFormation *stack* is a unit of deployment.
- Stacks consist of resources that are generated by a template.
- By using templates, you can create and update the collection of resources in stacks.
- The number of stack you deploy may be limited by quotas.
- If you delete a stack, all of the resources in the stack are deleted.



CloudFormation templates

A CloudFormation template:

- Is a text file
- Is formatted in JSON or YAML
- Is a self-documenting environment
- Designates resources to provision
- Supports the DevOps practice of infrastructure as code

```
"AWSTemplateFormatVersion" : "20…",
"Description" : "My…",
"Resources" : {
     "MyEC2Instance" : {
     "Type" : "AWS::EC2::In…",
     "Properties" : {
          "ImageId" : "ami…",
          "InstanceType" : "t2.mi…"
```

Template file

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

CloudFormation template example: Description

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

  "Description": "AWS CloudFormation Sample Template   ElasticBeanstalk_Simple:
  Configure and launch an Elastic Beanstalk application that connects to an
  Amazon RDS database instance. Monitoring is set up on the database."
  "Parameters": {
    "DBUser": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Test database admin account name"
      },
```

CloudFormation template example: Parameters

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

  "Description": "AWS CloudFormation Sample Template Config: This template
  demonstrates the use of AWS Config resources. **WARNING** You will be billed
  for the AWS resources used if you create a stack from this template.",

  "Parameters": {
    "Ec2VolumeAutoEnableIO": {
      "Type": "String",
      "Default": "false",
      "AllowedValues": ["false", "true"]
    },
```

CloudFormation template example: Resources

```
"AWSTemplateFormatVersion": "2010-09-09",
"Resources": {
  "Ec2Volume": {
    "Type": "AWS::EC2::Volume",
    "Properties": {
      "AutoEnableIO": {"Ref": "Ec2VolumeAutoEnableIO"},
      "Size": "5",
      "AvailabilityZone": {"Fn::Select": [0, {"Fn::GetAZs": ""}]},
      "Tags": [{
        "Key": {"Ref": "Ec2VolumeTagKey"},
        "Value": "Ec2VolumeTagValue"
```

Section 4 key takeaways

- CloudFormation is a fully managed service that automates AWS resource provisioning.
- CloudFormation uses JSON-formatted or YAML-formatted template files to provide resource provisioning instructions.
- A stack is a unit of deployment that you can create and update by using templates. When a stack is deleted, resources in the stack are also deleted.

Module 13: Automating Deployment Using CI/CD Pipelines

Section 5: Deploying applications with AWS SAM

AWS Serverless Application Model (AWS SAM)

- Open-source framework for building serverless applications
- Use to build templates that define your serverless applications
- Deploy your template with CloudFormation
- Integrates with CI/CD tools
- Provides an invocation environment locally

AWS SAM

AWS SAM template

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
   GetHtmlFunction:
      Type: AWS::Serverless::Function
      Properties:
         CodeUri: s3://sam-demo-bucket/todo_list.zip
         Handler: index.gethtml
         Runtime: nodejs4.3
         Policies: AmazonDynamoDBReadOnlyAccess
         Events:
            GetHtml:
            Type: Api
            Properties:
               Path: /{proxy+}
               Method: ANY
ToDoListTable:
        Type: AWS::Serverless::SimpleTable
```

Tells CloudFormation that this is an AWS SAM template that it needs to transform

Creates a Lambda function…

…with the code at the referenced .zip location, using the specified runtime and handler…

…and using the referenced IAM policy

Creates an Amazon API Gateway endpoint and takes care of all necessary mappings and permissions

Creates an Amazon DynamoDB table

Section 5 key takeaways

- AWS SAM provides the capability to deploy your stack to each AWS account.
- As an extension of CloudFormation, AWS SAM provides the reliable deployment capabilities of CloudFormation.
- AWS SAM has deep integration with CI/CD tools such as:
  - AWS CodeBuild
  - AWS CodeDeploy
  - AWS CodePipeline
  - AWS CodeStar

Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline

Lab: Scenario

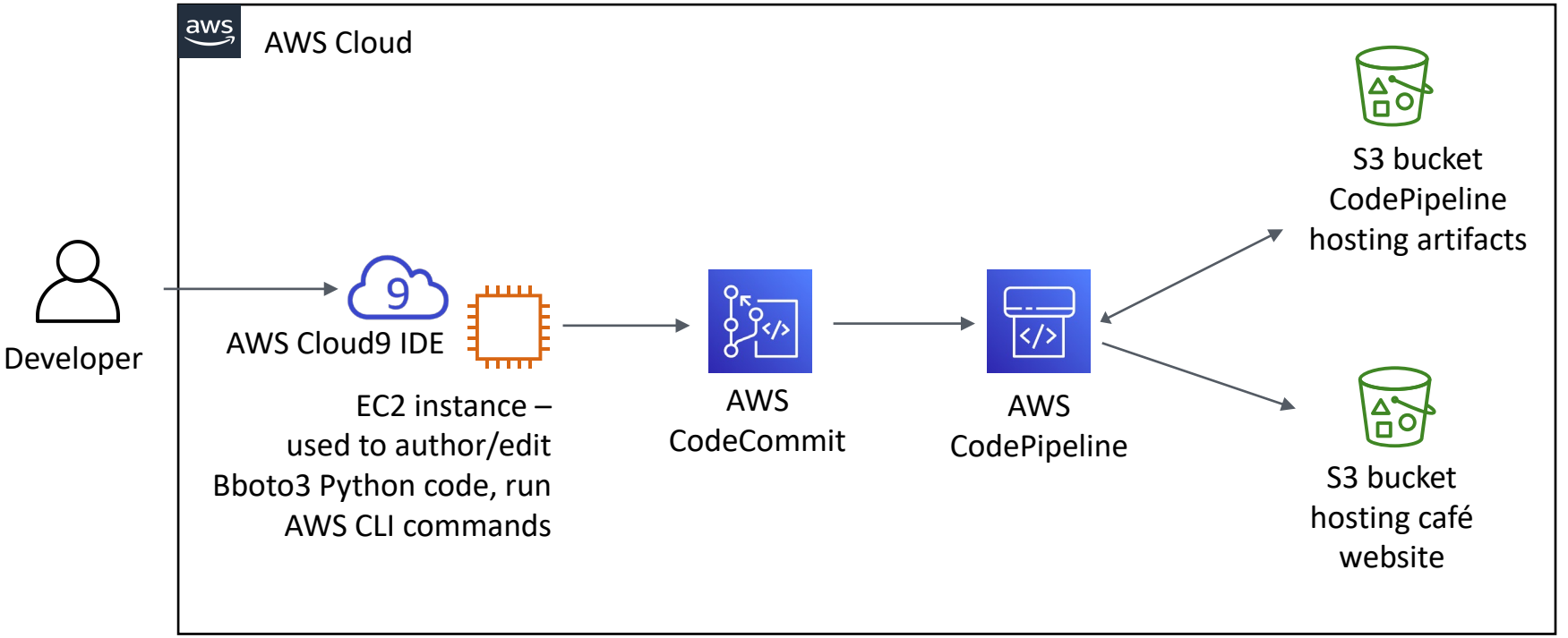
Now that the café website is in production, Sofía needs to centralize the website code and add version control. She also wants to automate the process of updating the website.

Sophia plans to use CodeCommit to store the website code. Mateo, the AWS consultant, suggested using CodePipeline to support automating updates to the website.

Lab: Tasks

1. Preparing the development environment
2. Creating a CodeCommit repository
3. Creating a pipeline to automate website updates
4. Cloning a repository in AWS Cloud9
5. Exploring the Git integration with the AWS Cloud9 IDE
6. Pushing the café website code to CodeCommit

Lab: Final product



~ 60 minutes

Begin Lab 13.1: Automating Application Deployment Using a CI/CD Pipeline

Lab debrief: Key takeaways

Module 13: Automating Deployment Using CI/CD Pipelines

Module wrap-up

Module summary

In summary, in this module, you learned how to do the following:

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

Complete the knowledge check

Sample exam question

A developer has reached the account quota for the number of CloudFormation stacks in a Region. How could they increase the quota?

A) Use the AWS Command Line Interface (AWS CLI).

B) Send an email to quotas@amazon.com with the subject "CloudFormation."

C) Use the Support Center in the AWS Management Console.

D) All service quotas are fixed and cannot be increased.

Additional resources

- Blog posts:
  - Complete CI/CD with AWS CodeCommit, AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline
- Whitepapers:
  - Introduction to DevOps on AWS
  - Infrastructure as Code
  - Overview of Deployment Options on AWS
  - Practicing Continuous Integration and Continuous Delivery on AWS
- Tutorial:
  - Build a Serverless Web Application

Thank you

aws academy