Assignment #05

Due Date

Section	Due Date	Grading Deadline
Saturday & Online	09:00pm on 07/12/2019	09:00pm on 07/19/2019

Getting Help

Info

Ask all your questions on Piazza. Assign **assignment5** tag to your posts.

Assignment Weightage

Individual Assignment Weightage on Course Grade of this assignment is **15%**.

Objectives

Assignment objective is to setup continuous integration and deployment pipeline with Github, CircleCI & AWS CodeDeploy.

Sign up for CircleCI

1. Sign up for CircleCI [https://circleci.com/] using your GitHub account.
2. Free plan provides **1000 build minutes** per month. If you exceed the free tier usage, you will have to pay.
3. Enable both of your repositories for build in CircleCI.

Create IAM User for CircleCI

Note

This one time setup can be done through AWS console.

Create a new user `circleci` with programmatic access only.

Update AWS AMI to Install CodeDeploy Agent

Install [https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html] Code Deploy Agent from `us-east-1` region in the AMI you build.

CodeDeploy AppSpec

Create AWS CodeDeploy `appspec.yml` to deploy your application on EC2 instances. The `appspec.yml` file should be in root of your repository.

Create S3 bucket for CodeDeploy

Note

This setup can be done through AWS console.

Create a S3 bucket in same region as your EC2 instance. Bucket name should be `code-deploy.yourdomain.tld` where `yourdomain.tld` should be replaced with your domain name.

CloudFormation Application Stack Updates

Create IAM Roles & Policies for AMI

1. Create a **circleci-ec2-ami** IAM policy and attach it to the `circleci` user. IAM policy can be found here [https://www.packer.io/docs/builders/amazon.html#iam-task-or-instance-role].
2. Create all the roles and policies used in the Continuous Deployment lab [../../lab/cicd/].

CloudFormation EC2 Resource Updates

Make following updates to the EC2 resource in the application CloudFormation stack:

1. EC2 instance should be launched with user data [http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata].
2. EC2 instance should be tagged appropriately so that CodeDeploy can identify EC2 instances on which application is suppose to be deployed.
3. Configure your application start the application with correct profile/env using user data [http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata].
4. Database username, password, hostname should be passed to the web application using user data [http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata].

Create CodeDeploy Application

1. Application Name - **csye6225-webapp**
2. Compute Platform - **EC2/On-premises**

CREATE CODEDEPLOY DEPLOYMENT GROUP

1. Deployment group name - **csye6225-webapp-deployment**
2. Service role - **CodeDeployServiceRole**
3. Deployment type - **In-place**
4. Environment Configuration - **Amazon EC2 Instances**
5. Provide the tag group key and values.
6. Deployment settings - **CodeDeployDefault.AllAtOnce**
7. Load Balancer - **disabled**
8. Rollback - **Roll back when a deployment fails**

Everything else can be left to default values.

Web Application Updates

Update your web application to connect to RDS instance, S3 bucket using AWS IAM.

1. Any IAM roles, policies created to support this requirement must be added to your application stack's CloudFormation template.
2. Book's images must be stored in the S3 bucket when the application is running on cloud (EC2 instances).
3. Web app must connect to RDS instance when running on cloud (EC2 instances).

Implement Continuous Deployment for Building AMIs

1. Any changes pushed to your `csye6225-su2019-ami` GitHub repository should trigger a build with new private AMI being registered with AWS.
2. Any IAM roles and policies needed to support this workflow must be added to your application stack's CloudFormation template.

Implement Continuous Deployment for Web Aapplication

1. Any changes pushed to your `csye6225-su2019-webapp` GitHub repository should trigger a build with artifact deployed on the EC2 instance.
2. Any IAM roles and policies needed to support this workflow must be added to your application stack's CloudFormation template.

Documentation

AWS CLI

- CodeDeploy [https://docs.aws.amazon.com/cli/latest/reference/deploy/index.html]
- S3 API [https://docs.aws.amazon.com/cli/latest/reference/s3api/index.html]

EC2

- Instance Metadata and User Data [http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html]
- EC2 UserData [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-ec2-instance.html#cfn-ec2-instance-userdata]

Spring

- Set the active Spring profiles [https://docs.spring.io/spring-boot/docs/current/reference/html/howto-properties-and-configuration.html#howto-set-active-spring-profiles]

IAM & CloudFormation

- AWS::IAM::Role [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-role.html]

- AWS::IAM::Policy [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-iam-policy.html]
- AWS::S3::Bucket [https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-s3-bucket.html]
- Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances [http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_switch-role-ec2.html]

AWS CodeDeploy

- AWS CodeDeploy AppSpec File Reference [http://docs.aws.amazon.com/codedeploy/latest/userguide/app-spec-ref.html]
- AWS CodeDeploy [http://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html]
- CodeDeploy [https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html]
- Overview of CodeDeploy Deployment Types [https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html#welcome-deployment-overview]
- Troubleshooting CodeDeploy [https://docs.aws.amazon.com/codedeploy/latest/userguide/troubleshooting.html]

CircleCI

- CircleCI [https://circleci.com/]
- CircleCI Documentation [https://circleci.com/docs]

- Using Environment Variables [https://circleci.com/docs/2.0/env-vars/]
- Pre-Built CircleCI Docker Images [https://circleci.com/docs/2.0/circleci-images/]
- Concepts [https://circleci.com/docs/2.0/concepts/#section=getting-started]

Submission

Danger

Assignment will be considered late if commits are made to `master` and `feature` branch after due date.

1. All work for this assignment must be done on **assignment5** feature branch and merged to master when you are dev complete.
2. All team member's feature and master branches must be in-sync.

Grading Guidelines

Warning

Following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

Demo Guidelines

Sync your repos after demo

All team members must sync their master branch due to the changes made for demo.

Warning

> Following guidelines are for information only. They are subject to change at the discretion of the instructor and TA.

1. Delete all existing custom AMIs from the account prior to the demo.
2. Delete all existing web application zip files from the S3 bucket used to store build artifacts. Do NOT delete the bucket itself. TAs must verify the bucket is empty.
3. Delete all existing CloudFormation stacks. TAs will verify that nothing is setup in the AWS account before the demo begins.
4. Verify that CircleCI user in IAM has no roles or policies attached. This user should have no permission at this point.
5. Download the code from the GitHub repository as `.zip` from `master` branch. You must use code from this download for rest of the demo. TAs must verify this step.
6. Create IAM and networking stack. At this point TAs will verify that networking has been setup correctly and all IAM roles and policies needed for CircleCI users are created.
7. Build a new AMI using AMI CI/CD pipeline. **CircleCI build must be triggered using API. Do NOT commit to GitHub repo to trigger the build.**
   a. Verify that AMI is built on top of `CentOS 7` base image.
8. Using the newly built AMI, create the application stack with `1` EC2 instance.
   a. Resources created by the application stack must be in the network that was setup in networking stack.
   b. TAs to verify that no resource exist in the default VPC after the application stack is created.
   c. TAs must also verify that application CloudFormation stack does not have anything hardcode for networking stack.
9. Once the application stack is ready and EC2 instances are up, trigger a new CircleCI build for the web application using API. Do NOT commit to GitHub repo to trigger the build.

    a. Verify that build artifact in S3 bucket.
    b. Only one build artifact must exist in S3 bucket at this point.
10. Database schema must also be bootstrapped at this point.
11. CodeDeploy will pull latest build artifact and deploy it.
    a. TAs to monitor progress via AWS console.
    b. Once CodeDeploy has successfully deployed the web application on all instances, TAs will start testing the web application itself.
12. TAs to verify usage of EC2 user data to pass configuration information to the EC2 instances. No AWS credentials should be passed to EC2 instances. IAM roles and policies must be used to access all AWS resources.
13. Demo that web application works without requiring any manual changes to web application or any AWS resources. Students cannot SSH to any of the EC2 instances to fix issues. Demo web application on EC2 instances. Make following API call to verify application
    a. Make an API call to create account.
    b. Make an API call using the newly created account to create a new book.
    c. Make an API call to attach an image to the book.
    d. Make API call to GET the book created.
    e. Check that book's images is stored in the S3 bucket. Deleting the book should delete the image and the book.
14. Now, students will make a simple code change to the API endpoint for demo. The `/book` endpoint will be renamed to `/book<student-first-name>` endpoint. Commit this change directly to the `master` branch.
15. A build should be automatically triggered by CircleCI now.
    a. Check that a new build artifact exists in the S3 bucket.
    b. Monitor CodeDeploy status.
    c. Test changes with API calls.
        i. Make an API call to create account.

    ii. Make an API call using the newly created account to create a new book.
    iii. Make an API call to attach an image to the book.
    iv. Make API call to GET the book created.

16. Revert the API endpoint changes made for demo and commit it back to your `master` branch.