Distributed Systems and Cloud Computing
Lab 8 «Create a Simple CICD Pipeline using GitHub Repository and AWS CodeDeploy»

In this lab, groups of up to three students will use AWS CodePipeline to deploy code that is maintained in a GitHub repository to a Amazon EC2 instance. You will use AWS CodeDeploy as the deployment service.

Topics

Step 1: Create a GitHub Repository and Local Repo

You will create a repository "sample_application_linux" in GitHub. Your pipeline will get the source code from this repository when it runs. You will also create on Step 2 a local repository where you maintain and update code before pushing it to the GitHub repository.

Step 2: Add Sample Code to Your GitHub Repository

In this step, you will download code for a sample application and add it to your GitHub repository.

1. Download the following file:
   - https://s3.amazonaws.com/aws-codedeploy-us-east-1/samples/latest/SampleApp_Linux.zip
2. Unzip the files from SampleApp_Linux.zip
3. On your machine, for example, your directory and file hierarchy should look like this:

4.
```
/SampleApp_Linux
   |-- appspec.yml
   |-- index.html
   |-- LICENSE.txt
   `-- scripts
     |-- install_dependencies
     |-- start_server
     `-- stop_server
```
5. Run the following command from the SampleApp_Linux directory in your machine to stage all of your files at once:
   - `git init`
   - `git add .`
6. Run the following command to commit the files with a commit message:
   - `git commit -m "Added sample application files"`
7. Push the files from your local repo to your GitHub repository:
   - `git remote add origin http://github.com/.../sample_application_linux.git`
   - `git push -u origin master`
   - For more on the topic: https://help.github.com/articles/adding-an-existing-project-to-github-using-the-command-line/
8. The files you downloaded and added to your local repo have now been added to the master branch in your GitHub sample_application_linux repository and are ready to be included in a pipeline.

Step 3: Create an Amazon EC2 Instance and Install the AWS CodeDeploy Agent

In this step, you will create the Amazon EC2 instance to which you will deploy a sample application. As part of this process, you will install the AWS CodeDeploy agent on the instance. The AWS CodeDeploy agent is a software package that enables an instance to be used in AWS CodeDeploy deployments.

To launch an instance

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. From the console dashboard, choose **Launch Instance**.

3. On the **Step 1: Choose an Amazon Machine Image (AMI)** page, locate the row for the HVM edition of the Amazon Linux AMI, and then choose **Select**. (This AMI is labeled "Free tier eligible" and can be found at the top of the list.)

   Note: For simplicity, chose the image which includes AWS command line and Python (typically the second option)

4. On the **Step 2: Choose an Instance Type** page, choose the free tier eligible t2.micro type as the hardware configuration for your instance, and then choose **Next: Configure Instance Details**.
5. On the **Step 3: Configure Instance Details** page, do the following:
   - In **Number of instances**, enter 1.
   - In **Auto-assign Public IP**, choose **Enable**.
   - In **IAM role**, choose an IAM role for use with AWS CodeDeploy. If you do not have an IAM instance profile, chose to **Create new IAM role** (EC2 role) and follow the instructions
     - Attach the permissions: AmazonEC2FullAccess, AWSCodeDeployFullAccess and AWSCodePipelineFullAccess
6. Still on **Step 3: Configure Instance Details** page, expand **Advanced Details**, and in the **User data** field, type the following:

#!/bin/bash
yum -y update
~~yum install -y ruby~~
~~yum install -y aws-cli~~
cd /home/ec2-user
aws s3 cp s3://aws-codedeploy-us-east-1/latest/install . --region us-east-1
chmod +x ./install
./install auto

7. Leave the rest of the items on **Step 3: Configure Instance Details** page unchanged. Choose **Next: Add Storage**, leave the **Step 4: Add Storage** page unchanged, and then choose **Next: Add Tags**.
8. On the **Add Tags** page, with “Name” displayed in the **Key** box, type “MyCodePipelineDemo” in the **Value** box, and then choose **Next: Configure Security Group**.
9. On **Step 6: Configure Security Group** page, do the following:

- Next to **Assign a security group**, chose **Create a new security group**.
- In the row for **SSH**, under **Source**, choose **My IP** (remember to change it later if you use it later in a different session).
- Choose **Add Rule**, choose **HTTP**, and then under **Source**, choose **My IP** (remember to change it later if you test it later from a different address).

10. Choose **Review and Launch**.
11. On the **Review Instance Launch** page, choose **Launch**, and then select **Choose an existing key pair**. Select your key pair.
12. Choose **View Instances** to close the confirmation page and return to the console.
13. It can take a few minutes for the instance to be ready for you to connect to it. Check that your instance has passed its status checks. You can view this information in the **Status Checks** column.

Step 4: Create an Application in AWS CodeDeploy

In AWS CodeDeploy, an *application* is an identifier (in the form of a name) for the code you want to deploy. AWS CodeDeploy uses this name to ensure the correct combination of revision, deployment configuration, and deployment group are referenced during a deployment. You will select the name of the AWS CodeDeploy application you create in this step when you create your pipeline later in this tutorial.

To create an application in AWS CodeDeploy

1. Open the AWS CodeDeploy console
2. Select Get Started, Sample Deployment, In-place deployment
3. In the **Application name** box, type MyDemoApplication, then Next and Next.
4. In the **Deployment group name** box, type MyDemoDeploymentGroup.
5. Enter appropriate values for the remaining fields.
6. In the **Search by tags** list, select the Amazon EC2 tag type, choose “**Name”** in the **Key** box, and in the **Value** box, type MyCodePipelineDemo.
    7. **Important:** You must choose the same value for the **Name** key here that you assigned to your Amazon EC2 instance when you created it. If you

tagged your instance with something other than MyCodePipelineDemo, be sure to use it here.

8. Use the provided role
9. In the **Deployment configuration** list, select CodeDeployDefault.**One at a time** then Next and Deploy.

Task 1: Provide evidence (screenshots) that you have created your AWS CodeDeploy application.

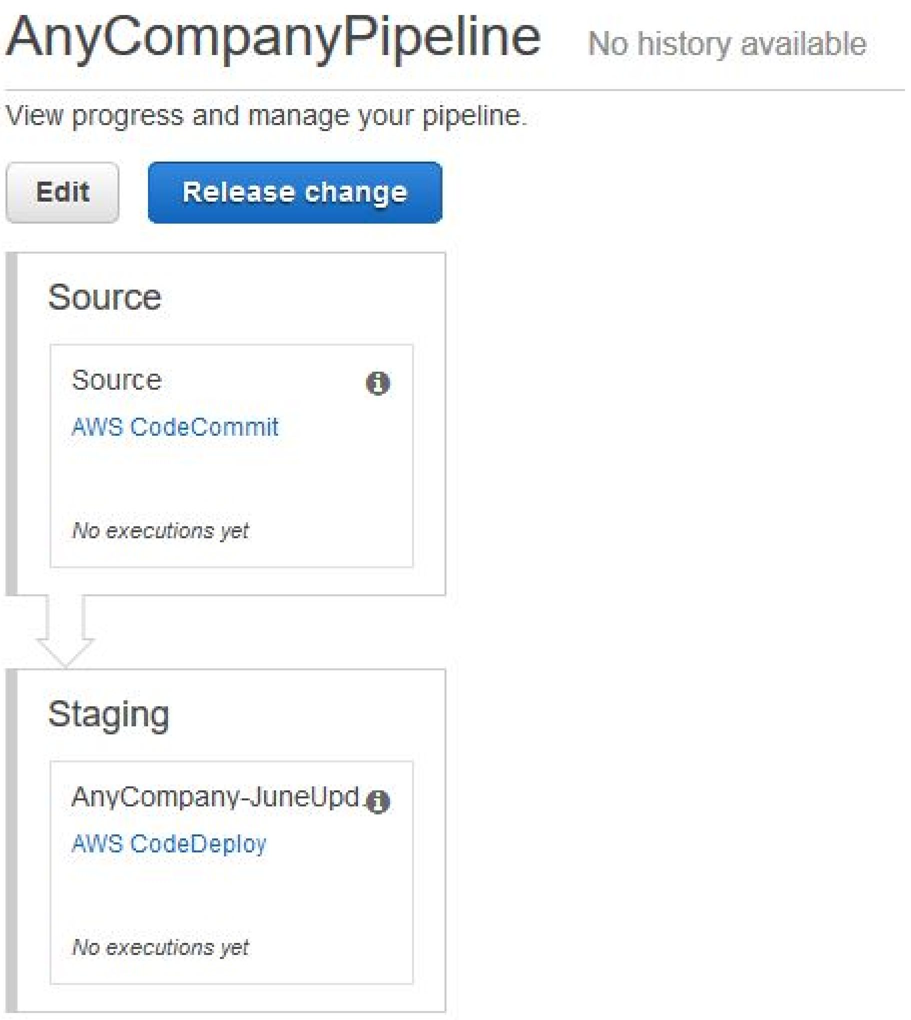
Step 5: Create Your First Pipeline in AWS CodePipeline

You're now ready to create and run your first pipeline.

To create an AWS CodePipeline automated release process

1. Sign in to the AWS Management Console and open the AWS CodePipeline console at https://console.aws.amazon.com/codepipeline.
2. On the introductory page, choose **Get started**.
3. If you see the **All pipelines** page, choose **Create pipeline**.
4. In **Step 1: Name**, in **Pipeline name**, type **MyFirstPipeline**, and then choose **Next step**.
5. In **Step 2: Source**, in **Source provider**, choose GitHub.
   a. Connect to GitHub
   b. Authorize aws-codesuite (if necessary)
   c. Select the created repository and master branch
6. In **Step 3: Build**, choose **No Build**, and then choose **Next step**.
   7. **Note:** In this tutorial, you are deploying code that requires no build service.
8. In **Step 4: Deploy**, in **Deployment provider**, choose **AWS CodeDeploy**. In **Application name**, type **MyDemoApplication**, or choose the **Refresh** button, and then choose the application name from the list. In **Deployment group**, type **MyDemoDeploymentGroup**, or choose it from the list, and then choose **Next step**.

9. In **Step 5: Service Role**, you will choose the IAM role to give AWS CodePipeline permission to use resources in your account. Service role creation is only required the first time you create a pipeline in AWS CodePipeline.
10. If you have not already created a service role for AWS CodePipeline:
    a. Choose **Create role**.
    b. On the IAM console page that describes the AWS-CodePipeline-Service role that will be created for you, choose **Allow**. After you create the role, AWS-CodePipeline-Service will appear in **Role name** on the **Step 5: Service Role** page.
    c. Choose **Next step**.
11. In **Step 6: Review**, review the information, and then choose **Create pipeline**.
12. The pipeline automatically starts to run. You can view progress and success and failure messages as the AWS CodePipeline sample deploys the web page to the Amazon EC2 instance in the AWS CodeDeploy deployment.


AnyCompanyPipeline No history available

View progress and manage your pipeline.

Source

Source
AWS CodeCommit
No executions yet

Staging

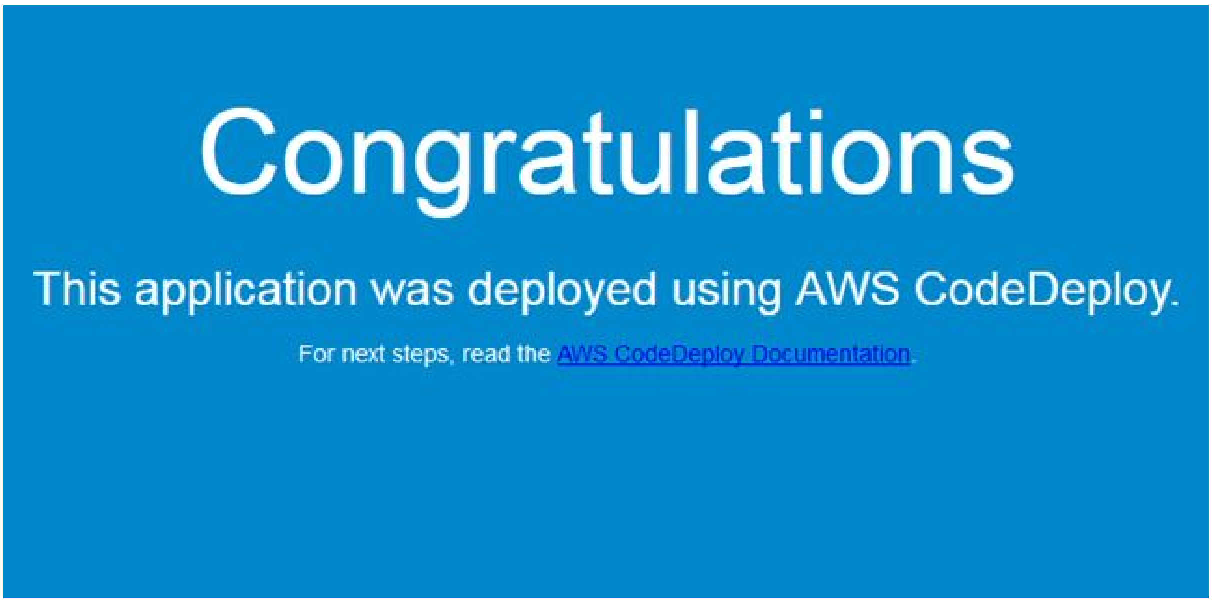
AnyCompany-JuneUpd
AWS CodeDeploy
No executions yet

Congratulations! You just created a simple pipeline in AWS CodePipeline. The pipeline has two stages: a source stage named **Source**, which detects changes in the sample application stored in the GitHub repository and pulls those changes into the pipeline, and a **Staging** stage that deploys those changes to the Amazon EC2 instance using AWS CodeDeploy. Next, you will verify the results.

To verify that your pipeline ran successfully

1. View the initial progress of the pipeline. The status of each stage will change from **No executions yet** to **In Progress**, and then to either **Succeeded** or **Failed**. The pipeline should complete the first run within a few minutes.
2. After the pipeline status displays **Succeeded**, in the status area for the **Staging** stage, choose **Details**.
3. In the AWS CodeDeploy console

   a. Deployments
   b. Select the first Instance ID
4. On the **Description** tab, in **Public DNS**, copy the address, and then paste it into the address bar of your web browser.
5. This is the sample application you downloaded and pushed to your GitHub repository.


Congratulations

This application was deployed using AWS CodeDeploy.

For next steps, read the AWS CodeDeploy Documentation.

Task 2: Provide evidence (up to 3 screenshots) that you have created your AWS CodePipeline and it is working properly.

Step 6: Modify Code in Your AWS CodeCommit Repository

In this step, you will make changes to the HTML file that is part of the sample AWS CodeDeploy application you deployed to your Amazon EC2 instance. When your pipeline runs again later in this tutorial, the changes you make will be visible at the http://*PublicDNS* URLs.

1. Use a text editor to modify the index.html file:
2. Revise the contents of the index.html file to change the background color and some of the text on the web page, and then save the file.
3. Commit and push your changes to your GitHub repository by running the following commands, one at a time:
4. git commit -am "Updated sample application files"

5. `git push`

Your pipeline is configured to run whenever code changes are made to your GitHub repository.

To verify your pipeline ran successfully

1. View the initial progress of the pipeline. The status of each stage will change from **No executions yet** to **In Progress**, and then to either **Succeeded** or **Failed**. The pipeline should complete within a few minutes.
2. After the action status displays **Succeeded**, in the status area for the **Staging** stage, choose **Details**.
3. In the **Deployment Details** section, in **Instance ID**, choose the instance ID of the instance.
4. On the **Description** tab, in **Public DNS**, copy the address, and then paste it into the address bar of your web browser.
5. The updated web page will be displayed

Step 8: Clean Up Resources

After you complete this tutorial, you should delete the pipeline and the resources it uses, so that you will not be charged for the continued use of those resources. First, delete the pipeline, then the AWS CodeDeploy application and its associated Amazon EC2 instance.

To clean up the resources used in this tutorial

1. To clean up your AWS CodePipeline resources
2. To clean up your AWS CodeDeploy resources

Recomended reference:

https://docs.aws.amazon.com/cli/latest/reference/autoscaling/index.html

Final instructions

1. Only one student (leader) will upload the full report on Moodle, which should show the full name of each student in the group.
2. The remaining students in the group will upload a “report” on Moodle consisting of a blank page containing only his/her leader’s name.