IAM Users, Roles & Policies

Add IAM roles & policies needed to meet the assignment objectives to the CloudFormation Template.

Amazon Simple Email Service (SES) Configuration

The following steps are done **manually** and only for the subdomain in the DEMO AWS account.

* [Verify Domain in Amazon SES](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/verify-domains.html)
* [Authenticate Email with DKIM in Amazon SES](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/dkim.html)
* [Move Out of the Amazon SES Sandbox by Requesting Production Access](http://docs.aws.amazon.com/ses/latest/DeveloperGuide/request-production-access.html)

**Warning:**

* **When requesting production access for email, you must submit a genuine case or AWS will deny your request.**
* **AWS support can take 3-5 days to respond to your request so you only have one attempt at getting your request approved to meet the assignment deadline.**
* ***There is nothing I or the TAs can do in getting your AWS request approved.***
* **I do not provide a template or ticket content. Writing up the use case and your needs is one of the objectives of this assignment.**

AMI Updates

1. Update your AMI to install the CodeDeploy agent.
2. [optionally] You can remove the provisioner that installs the application in the AMI.

CI/CD for Web Application

AMI Updates

* Install CodeDeploy agent in your AMI. Install instructions can be found at <https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-linux.html>.

EC2 Instance IAM Role Updates

* Update the IAM Role attached to your EC2 instance to include necessary policies for CodeDeploy to work.

Manually Create an S3 bucket for CodeDeploy

***This setup should be done through the AWS console (in both dev and demo accounts) and not in your CloudFormation template.***

* Create a private S3 bucket in the same region as your EC2 instance. The bucket name can be anything you want.
* Enable [default encryption for S3 Buckets (Links to an external site.)](https://docs.aws.amazon.com/AmazonS3/latest/dev/bucket-encryption.html).
* Create a lifecycle rule for the bucket to delete objects that are older than 30 days. See this [article (Links to an external site.)](https://www.joe0.com/2017/05/24/amazon-s3-how-to-delete-files-older-than-x-days/) for details.

Manually Create IAM User to use for Webapp CI/CD Pipeline

From the AWS console, create a new user ghactions-app with programmatic access. All roles and policies created and used for Webapp CI/CD workflow must be attached to this user.

Create CloudFormation Template with IAM Roles & Policies for CI/CD User

* Create a CloudFormation template for the CI/CD IAM roles and policies. This template should grant the IAM user the right roles and policies on all the resources it needs access to such as the S3 bucket for CodeDeploy, etc.
* This is a separate CloudFormation stack from the application CloudFormation Stack and must be created first.

CodeDeploy AppSpec

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

CodeDeploy Application & Deployment Group Resources

* Update the CloudFormation template to create CodeDeploy resources. See "Continuous Deployment with Github, GitHub Actions, and AWS CodeDeploy" page for details.

Implement Continuous Deployment for Web Application

1. Add a GitHub actions workflow to run the application unit tests for each pull request raised. A pull request can only be merged if the workflow executes successfully.
2. Add a GitHub actions workflow to build a new artifact when the pull request is merged. The workflow should do the following:
   1. Run the unit test.
   2. Validate Packer Template
   3. Build Application Artifact(s)
   4. Build AMI (Note: New AMI image is not used in the CI/CD pipeline in this assignment)
      1. Upgrade OS packages
      2. Install dependencies (python, node.js, etc.)
      3. Install application dependencies (pip install for python)
      4. Copy application artifact from step 3.
   5. Zip the artifacts and upload the zip archive to the CodeDeploy's S3 bucket.
   6. Trigger a new CodeDeploy deployment with the latest revision of your artifact.
      1. CodeDeploy can update the application on all instances at the same time. Downtime is acceptable here.
      2. CodeDeploy deployment type can be "in-place".

Demo Steps

TAs make note of all failed GitHub Actions Workflow and CodeDeploy executions. Workflows & jobs must succeed without changes.

* Students will pre-build and register AMI to use for the demo.
* Start the demo by building the CloudFormation stacks.
  + Create the CI/CD CloudFormation stack.
  + Create the Web App CloudFormation stack with the pre-built AMI.
* Verify the CI/CD workflow for the web app by making a simple code change.
  + Rename the **/healthz** endpoint to **/health (TA may ask for this endpoint name to be a different string)**.
  + The updated application deployed by CI/CD should now have the /health endpoint.
  + Verify **/heath** endpoint does not exist before kicking off CI/CD workflow.
* Once the CI/CD workflow for the web app has been shown to the TA, terminate the EC2 instance from the AWS console.
  + Verify new EC2 instances are launched in the auto-scaling group and the latest version of the application is running on it.
* Trigger auto-scaling by putting load on the application.
  + New instances launched by auto-scaling group should have latest revision of the application.

Submission

The assignment will be considered late if commits are made to the **main** branch after the due date.

1. All work for this assignment must be done on the feature branch in your fork and merged to main when you are dev complete.
2. The feature and main branches must be in sync.
3. Submit your code from **all** repositories in this assignment. **Read the instructions carefully to create your zip file.**
   1. Create a folder with the naming convention **firstname\_lastname\_neuid**
   2. In the **firstname\_lastname\_neuid**clone all of your GitHub (organization) repositories with the **git clone**command. It is important that you clone the repos so that your commit history and branches are preserved.
   3. Once you have cloned all of your repositories, you will create a zip of the **firstname\_lastname\_neuid\_a\_##** directory. The zip file should be **firstname\_lastname\_neuid\_a\_##.zip** where **##** is the assignment number.
   4. Now unzip the zip file in some other directory and confirm the content of the zip files.
   5. Upload the Zip to this assignment.
4. You are allowed to resubmit. If you think there may be an issue with the ZIP file, feel free to submit it again. Only the latest will be used grading.

Grading Guidelines

The following guidelines are for information only. They are subject to change at the discretion of the instructor and TA. 10% penalty will be imposed if the application crashes due to unhandled exceptions/errors.

Previous Assignment Objectives

* TAs must verify that students have resolved all open issues from the previous assignment(s).

Git Repository Content Check (10% Penalty)

* Check the repository for any IDE-specific files. IDE configuration files must not be in the repository.
  + Verify their .gitignore configuration.
* Check the repository for build artifacts such as .class, .jar, .war files and build, node\_modules directory. None of these should be checked into the repository.
* Check for dependencies. Dependencies from the Maven repository or npm should not be committed to the git repository.

Pre-requisite for Demo (5% Penalty)

* Delete all VPCs you may have created in your demo AWS account.

Application Crash (10% Penalty)

* The application should not crash or return *500 Internal Server Errors* during the demo.

Amazon SES Setup (15%)

1. Verify student’s domain has production access for SES in DEMO account.
   1. Check Verified Emails for SES and make sure all entries are removed.
2. Verify DKIM is configured for the domain.

CI/CD for Web Application (85%)

TAs make note of all failed GitHub Actions Workflow and CodeDeploy executions. Workflows & jobs must succeed on the first attempt.

Setup

* Students can configure GitHub Actions secrets before proceeding further. They are permitted to configure Secrets for the job.
* Verify that application is deployed using CodeDeploy when the EC2 instance comes up. You should be able to monitor this from the AWS console.
* Database schema must also be bootstrapped at this point. Students cannot run SQL scripts to set up DB schema. Verify RDS instance only has a private IP address and the security group only allows traffic from the web application security group.
* TAs to verify usage of EC2 user data to pass configuration information to the EC2 instances.
  1. No AWS credentials should be passed to EC2 instances.
  2. IAM roles and policies must be used to access all AWS resources.
* All IAM roles and policies must be created using Terraform template.
  1. Students cannot make any changes to created resources using the AWS console.
  2. TAs do not have to look at specific roles or policies.
  3. The check will be performed indirectly by CI/CD pipeline results.
* During the demo, note any feature of the web application that does not work as it will be an indication of IaC not being implemented correctly.

GitHub Status Check

* In the GitHub Actions log, verify that this step compiles code (if necessary) and runs the unit test only. It should not do anything else.

CI/CD

* Verify that the application endpoint works.
* Change from v1 to v2 in the API
* Merge the pull request and wait for the GitHub Actions workflow to run to completion.
  + Monitor progress in AWS console for CodeDeploy.
  + Verify that build artifact in S3 bucket.
* Once CodeDeploy has been completed successfully, make a request to any of the endpoints.
* Students will demo that web application works without requiring any manual changes to the web application or any AWS resources. Students cannot SSH to any of the EC2 instances to fix issues.