**Northeastern University**

CS 6620 Cloud Computing

**Project #2** [200 points]

**CI/CD With AWS ECS and AWS Lambda Function**

# Guidelines

Project #2 should be electronically submitted to Canvas by midnight on the due date. A submission link is provided on Canvas.

# Assignment Overview

For this project, in part 1, you will get familiar with building CI/CD devops pipeline with AWS ECS (Amazon Elastic Container Service) and review Docker knowledge. In part 2, you will be having knowledge of uploading and storing objects to S3 and learn to connect 2 different components of AWS.

Please post any questions to Piazza and attend TA office hours to address any questions. Get started early.

**PART I: CI/CD With AWS ECS and Docker [100 points]**

Follow the referenced video to do the lab step by step. For some steps the video may introduce 2 options, pick any one you are comfortable with. You may come across some small issues, just keep watching and learn with the video to fix them. Change the author name in all the files or config from “manikcloud” to your name.10 points for each step, include screenshots and text explanation.

**Attention:** Re-produce the bug in step 6) code build and screenshot the log with bug and fixed build.

1. Docker Basics: Create EC2 Instance. Install Docker. Pull centos:centos6. Create index.html with your greeting. Create Dockerfile. Build a Docker image

Graphical user interface, diagram, application

Description automatically generated

1. [17:57] AWS ECR(Elastics Container Registry) Creation: Create AWS ECR. Login to ECR. Tag existing image as AWS ECR repo. Push the image into the ECR
2. Create ALB(Application Load Balancer): Create Target group. Create ALB
3. Create an AWS ECS(Elastics Container Service) Cluster: Create a Task with Fargate Computability. Create a Fargate Cluster. Create a Service with ALB enabled. Test ALB DNS, whether your site is running or not. Add your container with auto-config CloudWatch Logs

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

1. [38:00] AWS Code commit: Create a Repo. Set ssh connectivity in your local machine. Push your code into the newly created repo
2. [54:00] AWS Code Build: Create a Code build Project. Select an Artifact for output. Build your Docker image & push to AWS ECR
3. [1:06:55] AWS Code Pipeline (the manual one is optional and you can skip it but not harm to try, automated way starts at [1:10:05]): Create a pipeline Project. Select you Code commit repo as an input repo from step 5. Select Code build project from Step 6. Select Code Deploy for ECS. Select your cluster and Service name from step 3. Crete the Code pipeline
4. [1:19:45] Error & Resolution: In the above pipeline, first 2 steps will run successfully. Step 3 deploy will give you an error. For this, you need to Create one “imagedefinitions.json” file and push it to the code commit. The pipeline will run again and you will again get an error on step 3
5. Modify Input Artifacts Settings for Stage: Change the input.
6. Final Deployment Test and Validation: Push the new version of code in a code commit. It will automatically deploy the new task with the new version. At last, you can run the DNS ALB on your browser

**PART II: Design Character Recognition System using AWS Lambda Function [100 points]**

1. Docker Read the data from S3
2. Recognize the character present in S3 bucket, using AWS API
3. Upload the extracted text from the above step back to S3 bucket.

you need to check that the assigned role is S3 read/write permission and Amazon Rekognition API.

AWS lambda trigger: S3 event trigger.

You need to perform the following steps using AWS lambda and S3 bucket only.

The above question is completely code oriented.

Explain each subtask using screenshot, code and text as needed to fully answer these questions.

**Other notes:**

Please see TA announcement regarding project submitting guidelines. A PDF report showing the results via screenshots and brief explanations are required. In addition, submit the code file as appropriate.

# Evaluation

Your work will be evaluated on how well the code, results including screenshots and explanations (text) are in conformance to the requirements above.

# Executive Summary

Part of your completed assignment submission should be an executive summary containing an

“Assignment overview” (1 paragraph, up to about 250 words) explaining what you understand to be the purpose and scope of the assignment and a “technical impression” (1–2 paragraphs, about 200–500 words) describing your experiences while carrying out the assignment. The assignment overview shows how well you understand the assignment; the technical impression section helps to determine what parts of the assignment need clarification, improvement, etc., for the future.

# Evaluation

The grade for your executive summary is based on the effort you put into the assignment overview and technical impression. In general, if you put some effort into your writing, you will receive full credit for your executive summary (provided that it is properly formatted and submitted as a plain text file).

# Project Deliverables

The following items should be archived together, e.g., placed in a .zip file or tarball file (\*.tgz or \*.tar.gz), and electronically submitted via the link is provided on the course Moodle page.

1. All source code files plus any additional support code you developed to answer.
2. Your executive summary and the PDF report answering all questions.

**References**

<https://www.youtube.com/watch?v=d7PTjQiahOQ>