Assignment Details:

***#****Deploy a****production-ready****Jenkins on ECS Fargate:****Must haves****:*

* Controller data must **persist** when the container restarts.
* Should be able to **exec** into the container for debugging.
* There must be **NO** manual configuration made through the Jenkins UI. All configurations must be made through **init groovy or JCaC**.
* EC2 plugin must be set up for running **distributed builds**.
* Relevant **monitoring**must be in place.
* A **sample job** must exist utilizing EC2 instances as agents.

Overview:

The assignment to deploy a production-ready Jenkins on ECS Fargate has been successfully completed. The Deployment meeting all the mentioned requirements. have done testing as well.

**Key Features:**

1. **Persistent Controller Data:** The Jenkins controller data is stored persistently to ensure that important configurations, plugins, and job histories are retained, I have used aws efs as persistent storage, Just replace with your own file system ID in terraform configurations ( file name: task-definition.tf under volume section.)
2. **Debugging Capabilities:** The setup allows for easy debugging by enabling ECS exec into the Jenkins container. This feature provides valuable insights and troubleshooting options. ( see service.tf file)  
   to login into the container we have to just run one aws cli command that is given below  
     
     
   aws ecs execute-command --cluster jcluster \
3. --task 08fb7e38bf2249bab9d0a0b82281e706 \
4. --container jenkins-container \
5. --command "/bin/bash" \  
   --interactive  
   Just replace cluster name, task id and container name
6. **Automated Configuration:** All configurations are managed through Jenkins Configuration as Code (JCasC). You can refer the JENKINS-CASC folder for the same.
7. **EC2 Plugin for Distributed Builds:** have setup ec2 as agent for Jenkins. Created Jenkins slave (EC2) through terraform (refer ec2-as-build-agent.tf file for agent creation) and configured this ec2 as agent ( refer JENKINS-CASC folder for the same.)
8. **Sample Job Utilizing EC2 Agents:** A sample job is configured to utilize EC2 instances as agents for executing builds. Have created a sample Jenkins file for the same ( refer Jenkinsfile under SAMPLE-JOB folder).

Step by step execution of the solution.

There are three folder for this assignment named infra, c and SAMPLE-JOB

Infra: containing all the terraform configurations related to infrastructure required for hosting Jenkins on ecs container

JENKINS-CASC: Containing all the Jenkins configuration related file.

SAMPLE-JOB: containing a sample Jenkins job that runs on an ec2 agent.

Execution:

1. First Run the docker file to create custom Jenkins image that will contain all the configurations which will be made through Jcasc. ( Dockerfile is under JENIKNS\_CASC folder)
2. update the name of image in task-definition terraform file ( task-definition in infra folder).
3. After replacing the image with custom Jenkins image, create all the infrastructure defined in infra folder.

NOTE: before running terraform apply just scan terraform files once and replace parameters like (account,region,subnet,security group and others)

1. Once the infra gets created, it will provision a Load balancer that can be used to access jenkin application.
2. Create a Jenkins job with the sample job provided. (SAMPLE-JOB/jenkinsfile)