Read the following instructions before completing the given task.  
  
Step 1: Click on the Online Task Link

Step 2: Download the document

Step 3: Check the online questions

Step 4: Complete the task and take a screenshot for each and every task and upload

Step 5: Save the document

Step 6: Upload the document in Google form

* **AWS Account is mandatory to complete the tasks**.
* **Include AWS account Name in your screenshots**
* **Resume / Copied and similar tasks will be rejected directly.**

**Assignment 1**

**Associate an Elastic IP to an EC2 Instance using Terraform**

**- Create an ec2 instance using terraform workflow**

**- associate an elastic IP  
  
  
  
Upload the final output Screenshot**

**Assignment 2**

**Creating an IAM User and Assigning Permissions:  
  
Click on "Users" in the left-hand menu and choose "Add user."**

**Provide a username and select the access type (programmatic access, AWS Management Console access, or both).**

**Set permissions by adding the user to one or more IAM groups with predefined policies or by attaching custom policies directly.**

**Upload the final output Screenshot**

**Assignment 3**

**Create 2 VPC's Named "Webapp-VPC" & "Db-VPC"  
  
It should have 2 Subnets each, one with Class A IPv4 CIDR and Class B IPv4 CIDR and 255 ports in each subnet.**

**Upload the final output Screenshot**

**Assignment 4**

**Creating a Custom Amazon Machine Image (AMI)**

**- Launch a New EC2 Instance**

**- Install http on the new instance, enable the http service to start at boot.**

**- Create a New AMI from customized instance and name the AMI MicroDegreeWeb**

**- Launch a New Instance Using the Custom AM**

**- Verify that http is running.**

**Upload the final output Screenshot**

**Assignment 5**

**Creating a Basic Amazon S3 Lifecycle Policy**

**- Create an S3 Bucket and Upload an Object**

**- Create a Lifecycle Policy - Create a lifecycle policy that moves objects to Glacier Flexible Retrieval (formerly Amazon Glacier) if they haven't been accessed in the last 30 days.**

**Upload the final output Screenshot**

**Assignment 6**

**Create a deployment strategy for a containerized application. Choose a container orchestration platform such as Kubernetes or Docker Swarm, and design a deployment process that includes blue-green deployments or canary deployments. Outline the steps required to roll out new versions while minimizing downtime and ensuring service availability.**

**Upload the final output Screenshot**

**Assignment 7**

Build a custom docker image using Ubuntu as a base docker image and run the nginx application. - this docker image should be built using a Dockerfile. Once the docker image is build, start the docker image using the host network and make it accessible on Public IP

**Upload the final output Screenshot**

**Assignment 8**

**Setting Up Continuous Integration and Deployment (CI/CD):**

**Jenkins is often used for implementing CI/CD pipelines to automate the build, test, and deployment processes.Create a pipeline job using Jenkins Pipeline DSL (declarative or scripted) or a Jenkins file.**

**Define the stages of your pipeline, including building, testing, code analysis, and deployment.**

**Configure Jenkins to trigger the pipeline based on code changes, commits, or other events.**

**Upload the final output Screenshot  
  
Assignment 9**

**Creating a Docker Compose Configuration:**

**Write a Docker Compose YAML file (docker-compose.yml) to define a multi-container application setup.**

**Specify the services, their configurations, and any necessary links or dependencies between containers.**

**Use the docker-compose up command to start the containers defined in the Docker**

**Upload the final output Screenshot**

**Assignment 10**

**Working with Docker Images**

**- Pull the latest `httpd` image.**

**- Pull the latest `alpine` image.**

**- verify images pulled and create 2 containers in each server**

**Upload the final output Screenshot**  
  
 **Assignment 11**

**Exposing an Application with a Service:**

**Create a YAML file (service.yaml) that defines a Service to expose your application.**

**Specify the Service type (e.g., LoadBalancer, NodePort, ClusterIP), ports, and target selectors.**

**Apply the Service to the cluster using kubectl apply -f service.yaml.**

**Access the exposed application using the appropriate service endpoint (e.g., LoadBalancer external IP, NodePort, ClusterIP).**

**Upload the final output Screenshot  
  
  
Assignment 12**

**Create a Staging branch in GitHub and push code from the local repository to the Remote and share the full commands screen**  
  
  
 **Upload the final output Screenshot**

**Assignment 13**  
  
 **Scenario: You are deploying a web application on AWS. How would you set up an automated deployment pipeline using DevOps principles?**

**Question: What steps would you take to implement continuous integration and continuous deployment (CI/CD) for your web application on AWS?**

**Write an Answer here:**

**Assignment 14**

**Scenario: Your team is working on an application that requires scalability and high availability. How would you architect the application using AWS services and DevOps best practices?**

**Question: Explain how you would design an architecture that leverages AWS services to ensure scalability and high availability for your application, following DevOps principles.**

**Write an Answer here:**

**Assignment 15**  
  
  
**Scenario: Your application needs to process and analyze streaming data from various sources. Which AWS service can you use for real-time data streaming and analytics?**

**Write an Answer here:**