DEVOPS DAY-1 ASSIGNMENT - PEDAVALLI LEELA BHARGAVI.

CLOUD ACCOUNT SETUP

creation of AWS Account:

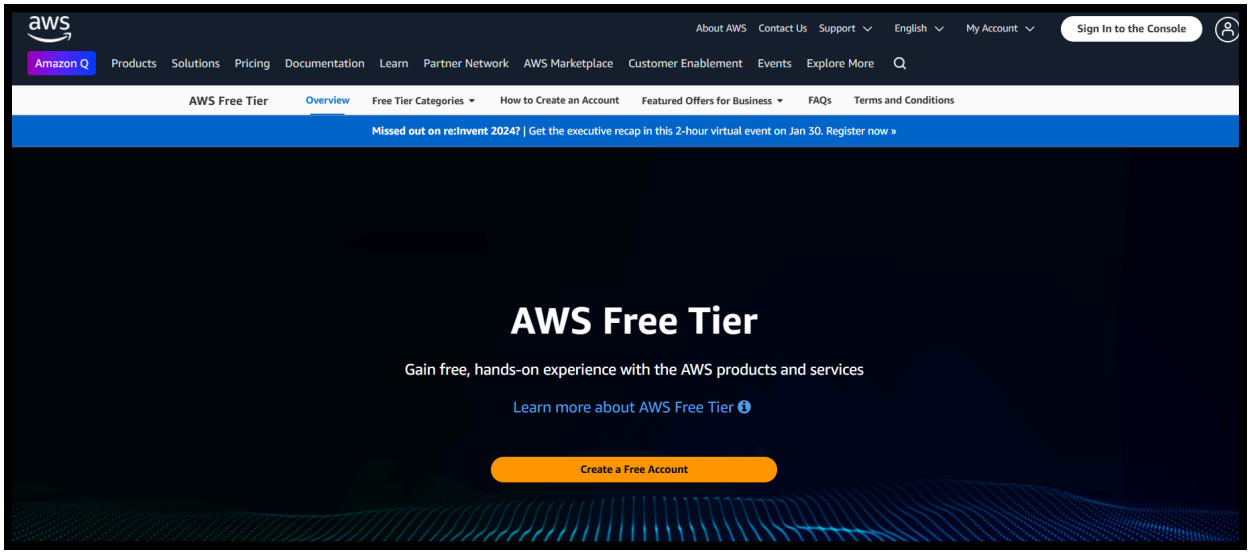
Here are some steps to create an AWS account,

step-1: Visit the AWS website:

- Open your web browser and go to the official AWS website:

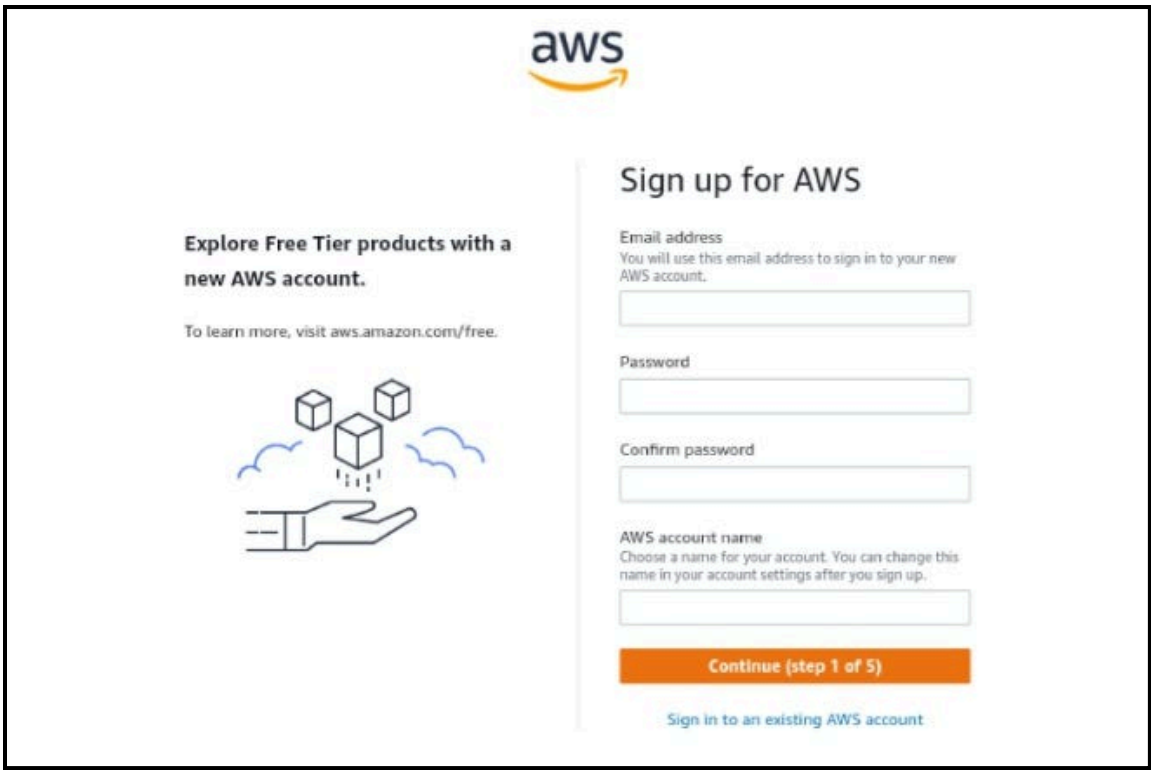
https://aws.amazon.com/

- Click on “create an AWS Account” button.



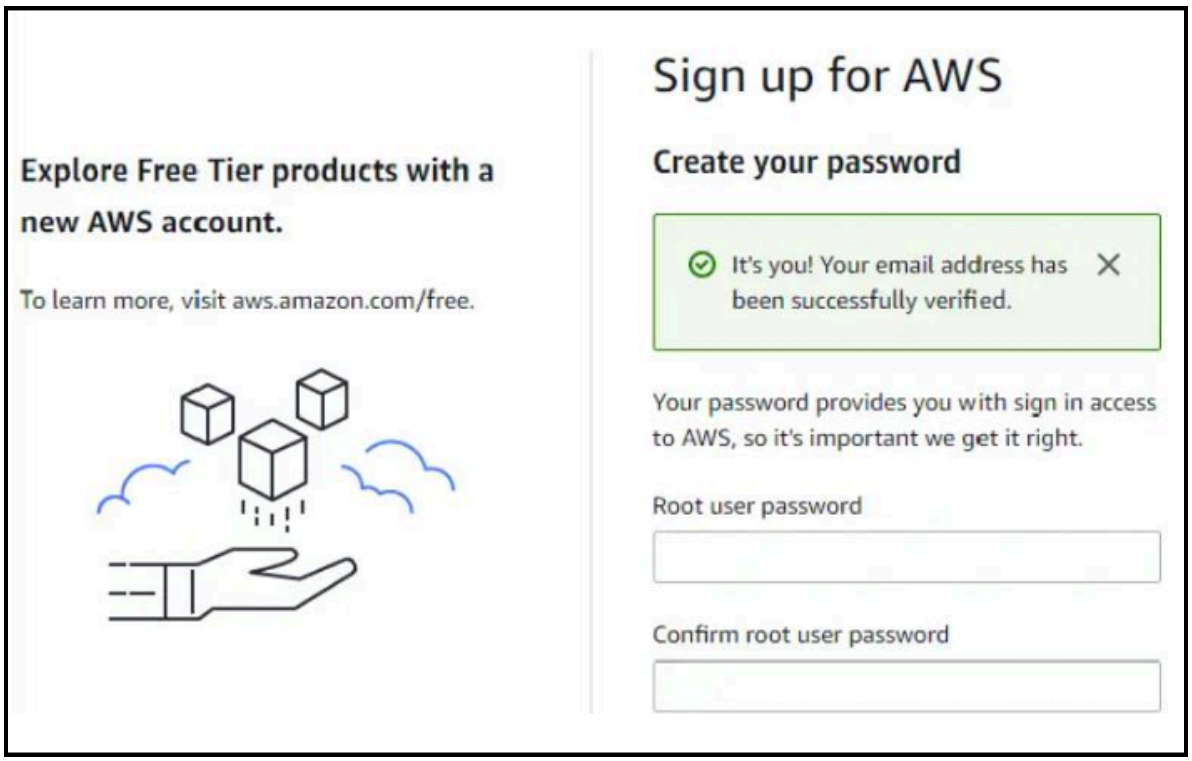
step-2:Enter Email and Account Name:

- Provide your Email Address.
- Choose an Account name.
- Click “verify Email Address”.



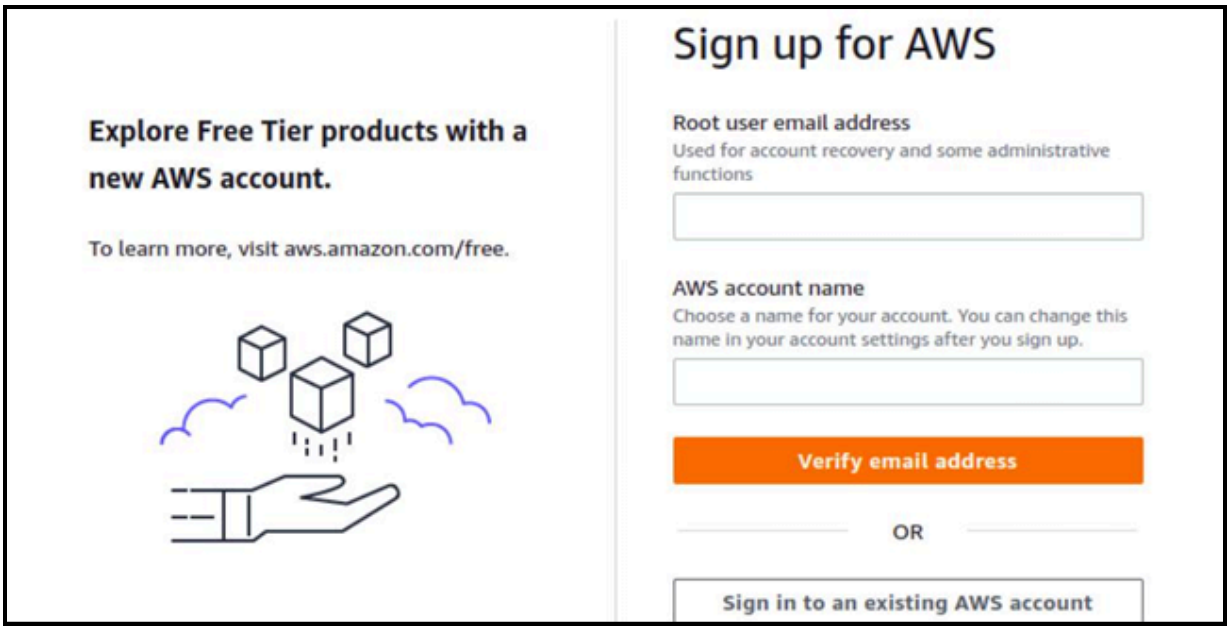
step-3:Verify Email:

- Check your Email inbox for a verification code from AWS.
- Enter the code on AWS Website and click verify.



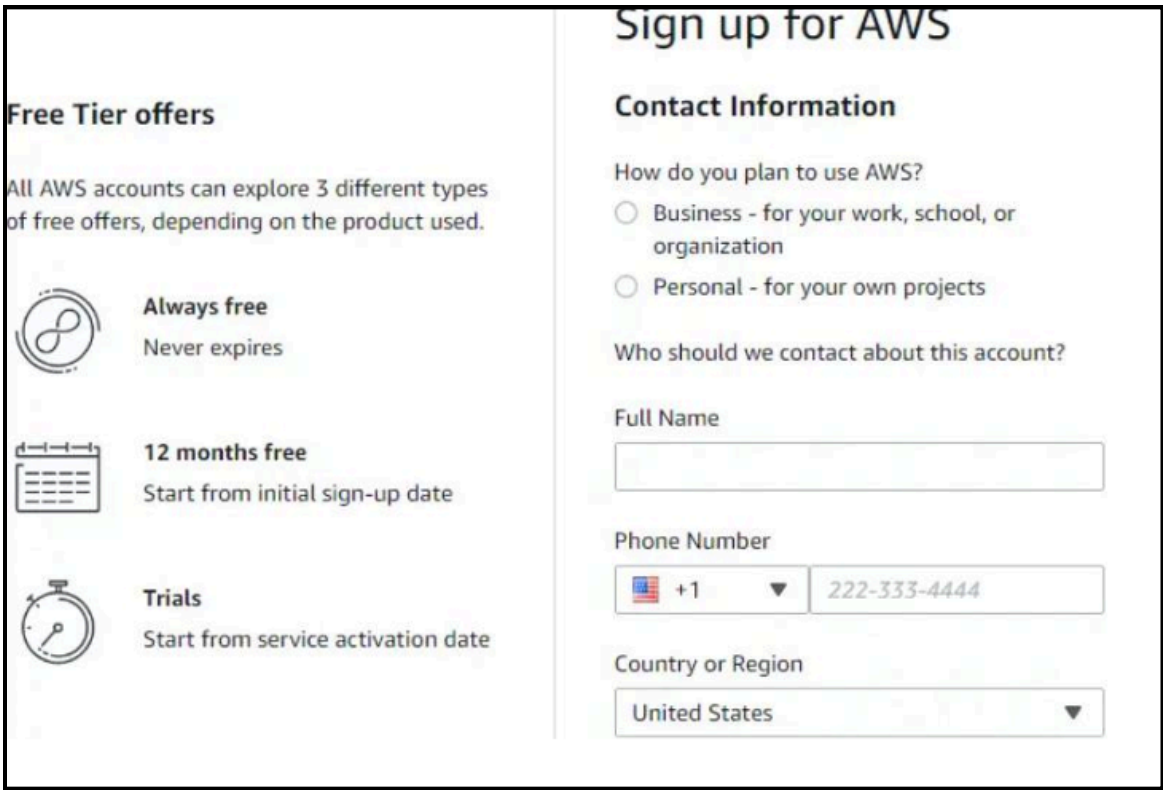
step-4: Set root user password:

- Create a strong password for your root user account,This is the main account for your AWS.
- Click on continue.



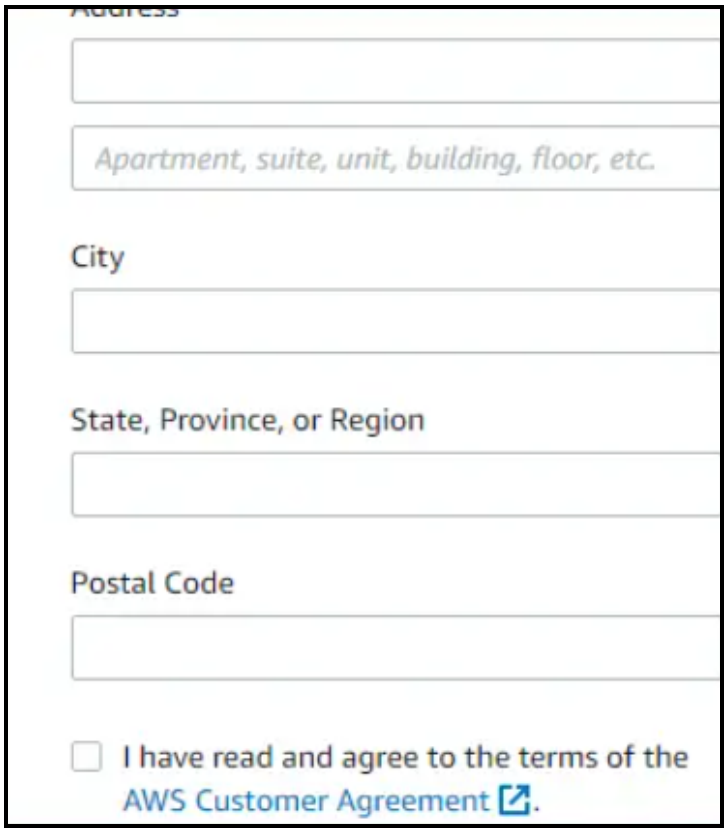
step-5:.Provide Account Information:

- Select “Business” or “Personal” account type.
- Enter your Business or Personal information.(name,mobile number,address)



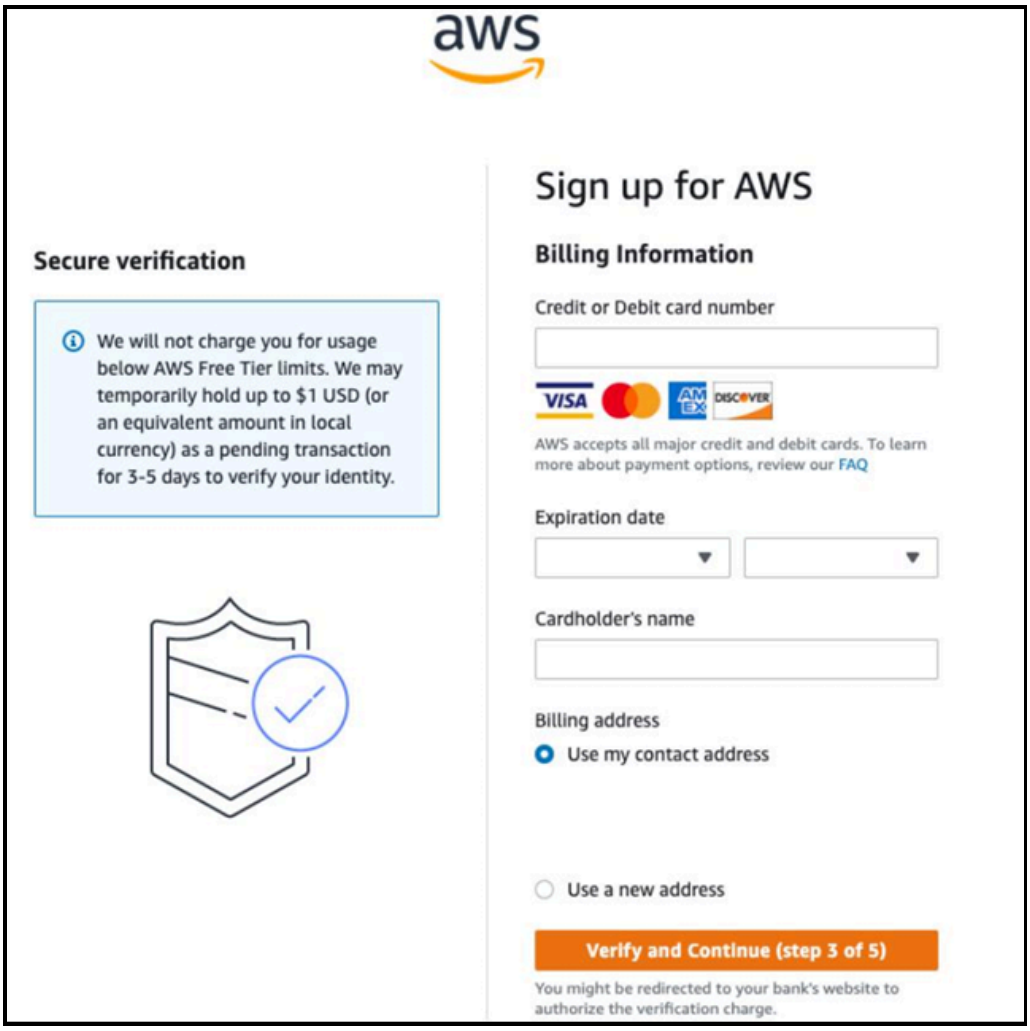
step-6:Accept the AWS customer Agreement:

- Review and accept the terms of AWS customer agreement.



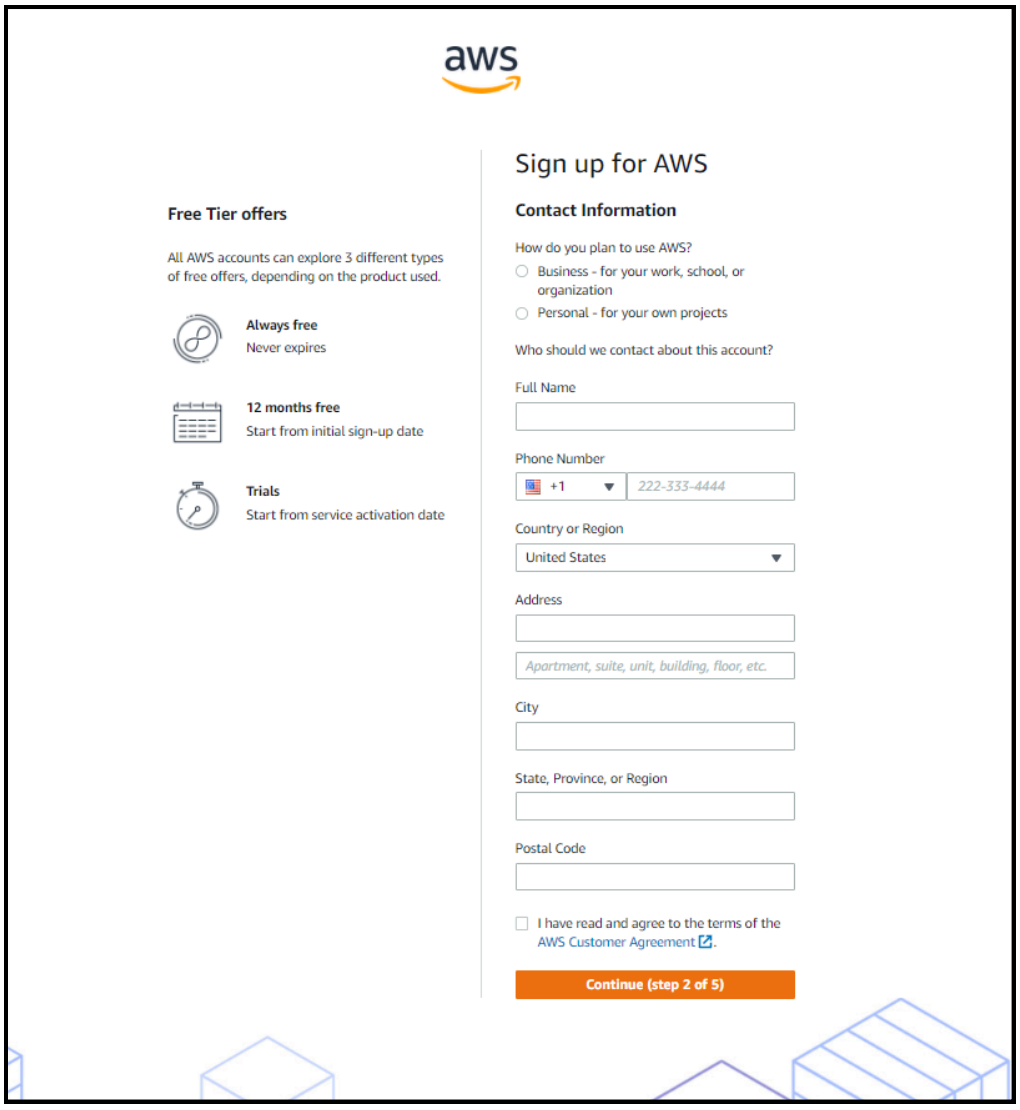
step-7:Add a Payment Method:

- Enter your credit or debit card information.
- Click verify and continue.



step-8:Verify your phone number:

- Choose how you want to receive a verification code(SMS or voice call).
- Enter the code you receive.

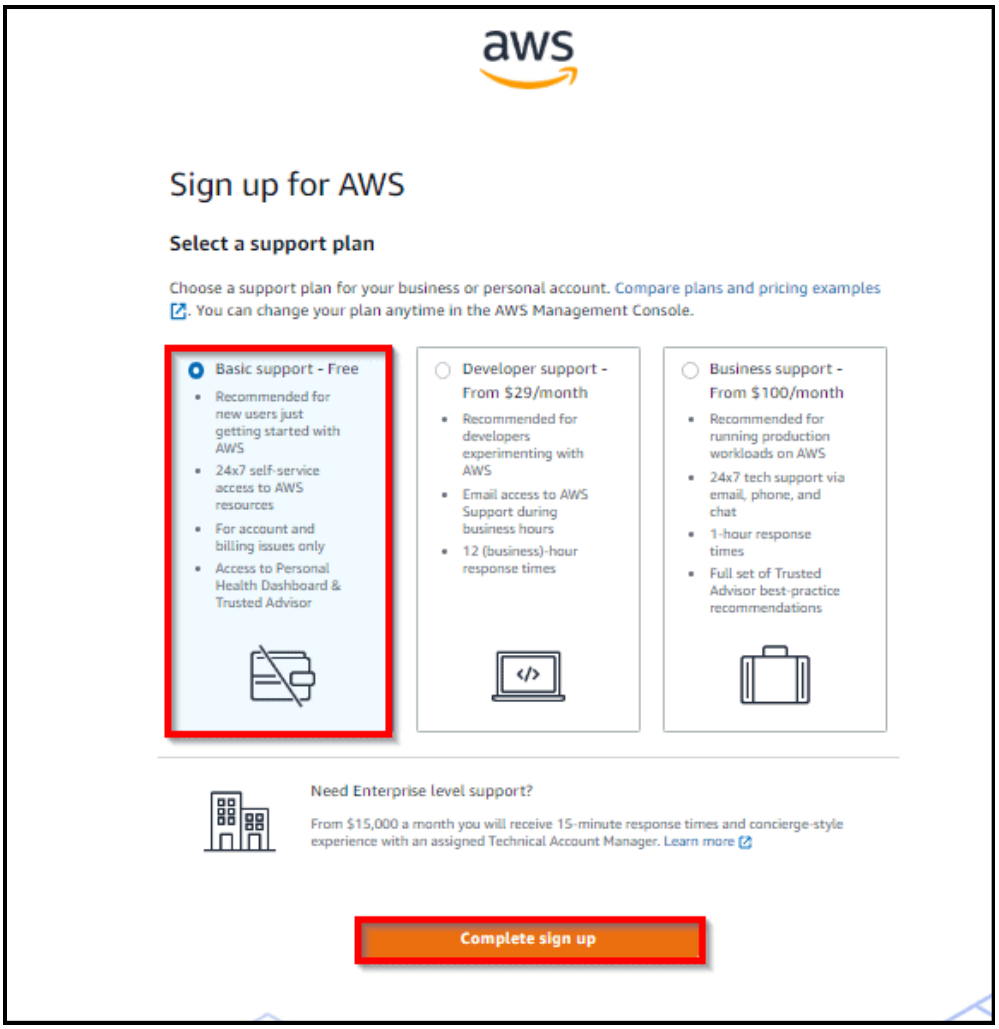


step-9:.Support Plan:

- Basic support plan is “free”.

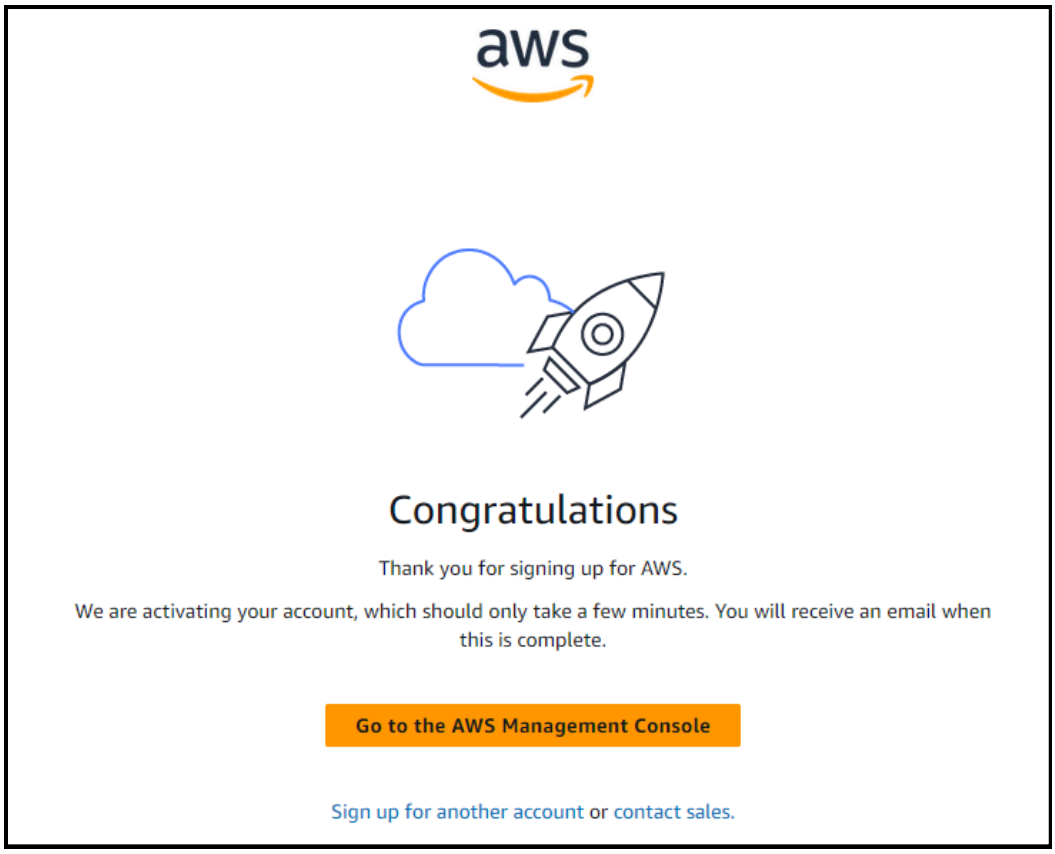
step-10: complete sign up:

- Click on complete “sign up”.



step-11:Account Activation:

- Account will be activated in few minutes. Receives an Activation mail once the Activation is complete.

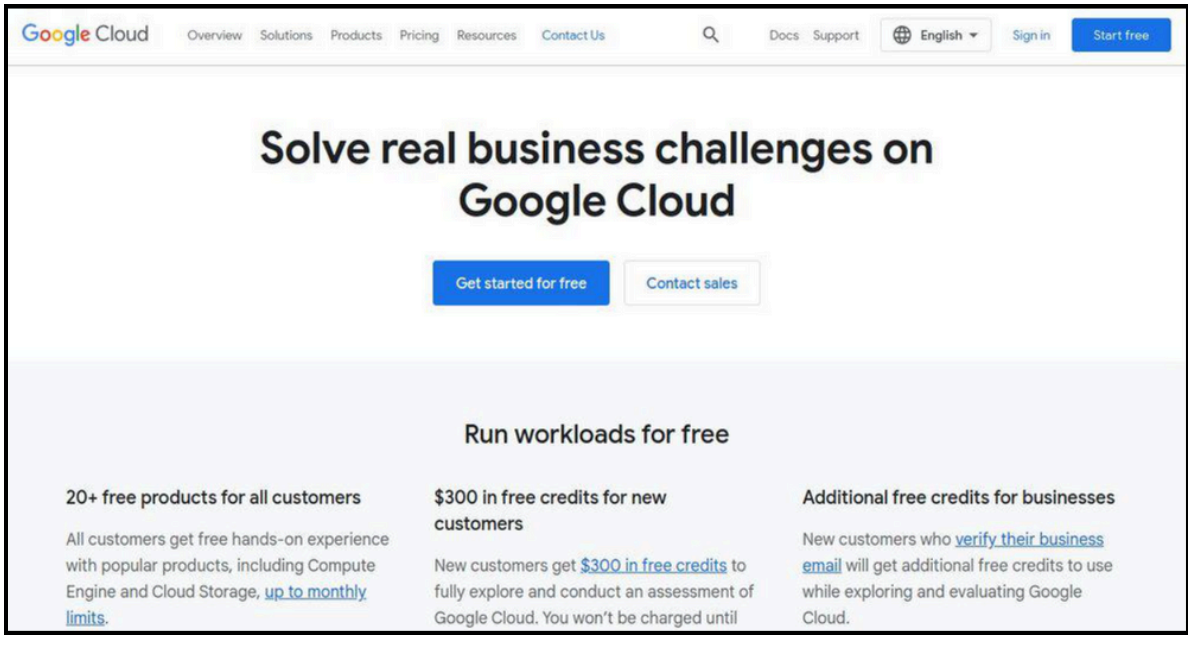


creation of GCP Account:

Here are some steps to create an GCP account,

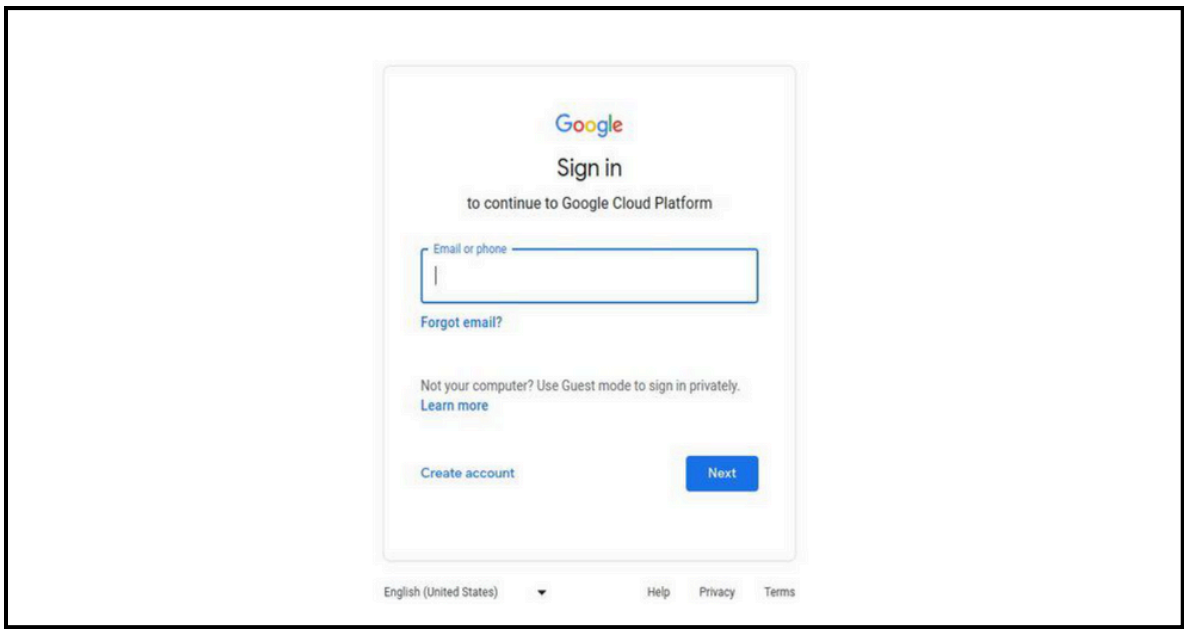
step-1:Go to the GCP account creation page:

- Open your web browser and navigate to the Google Cloud Console: https://console.cloud.google.com/



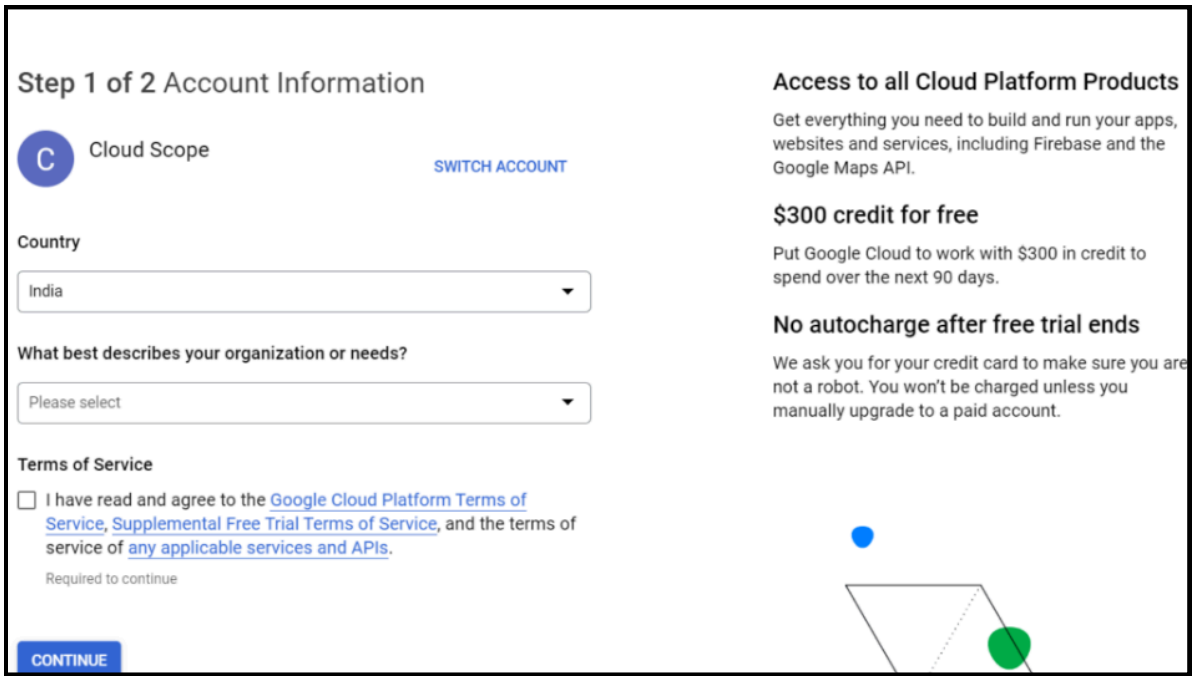
step-2:click on the start free button on the top corner.

step-3:sign in to your Gmail account:



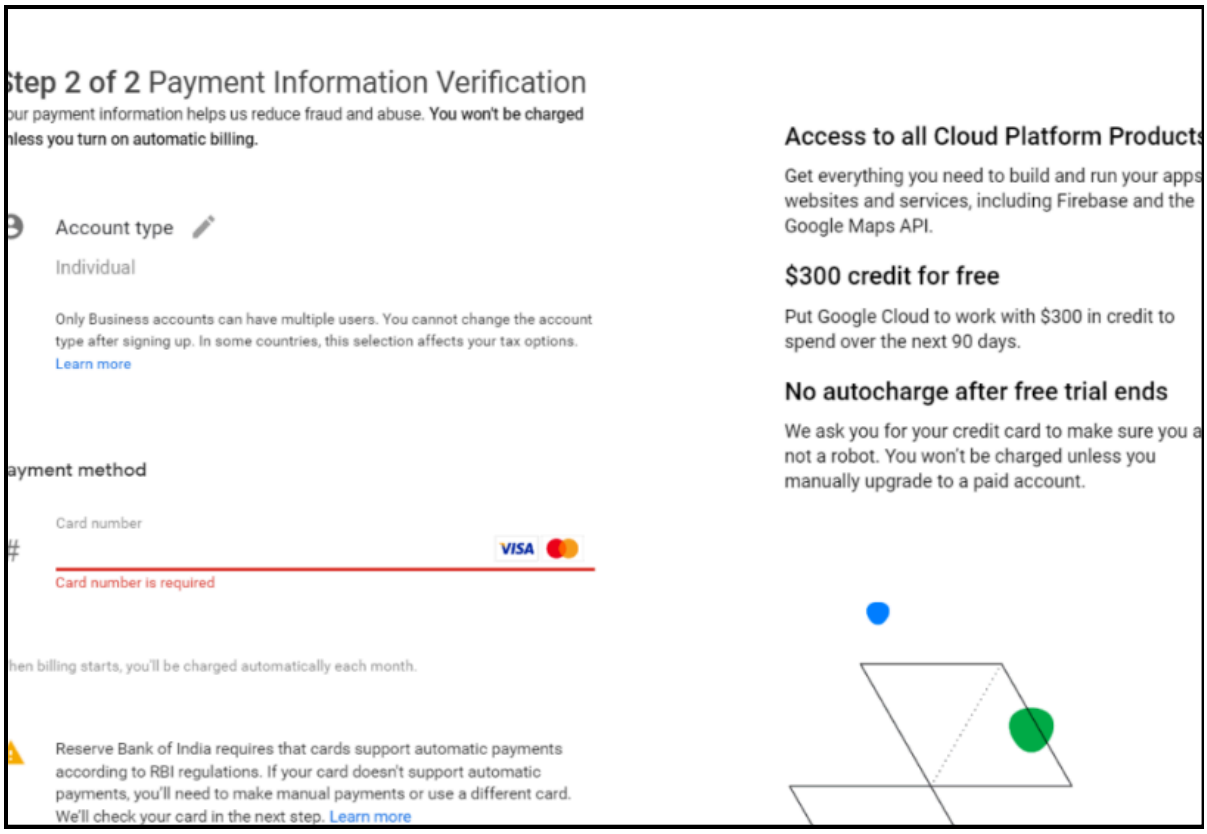
step-4:Accept the GCP customer Agreement:

- Review and accept the terms of GCP agreement.



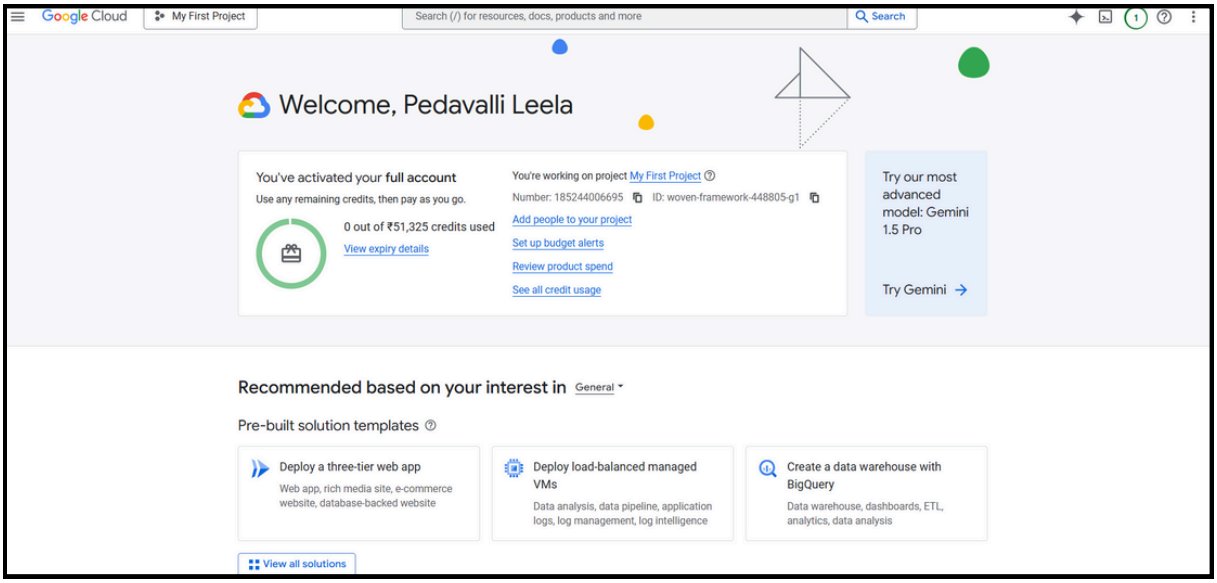
step-5:Enable Billing:

- Enter your debit card (only VISA and MASTER cards are applicable) or credit card information.



step-6:Account completion:

- After completion of payment you will be redirected to Gcp console page.



▢ Introduction To DevOps:

Write a brief summary (250-300 words) explaining the following:

- **DevOps culture and principles: What does it mean to have a DevOps culture in an organization?**

DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). It aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

A DevOps culture emphasizes collaboration, communication, and shared responsibility between development and operations teams.

Here's a breakdown of what it means to have a DevOps culture:

1. **Collaboration and Communication:**
- Development and operations teams work closely together, sharing knowledge, and communicating openly throughout the entire process.
- GCP fosters this collaboration through several integrated services, such as Cloud Source Repositories, Cloud Build, and Google Kubernetes Engine (GKE).
2. **Automation:**
- Automating tasks like testing, deployment, and infrastructure provisioning reduces manual effort, minimizes human error, and increases efficiency.
- This allows for faster and more frequent releases of software.
3. **Continuous Improvement:**
- Regularly analyzing and improving the development and delivery process based on feedback and data.
- This iterative approach ensures that the organization is constantly learning and adapting to improve its software delivery capabilities.
4. **Continuous Integration/Continuous Delivery (CI/CD):**
- Regularly integrating code changes into a shared repository and automating the build, test, and deployment process.
5. **Shared Responsibility:**
- Development and operations teams share responsibility for the entire software lifecycle, from development to production support.
6. **Feedback:**
- Continuous feedback is a core principle with rapid feedback form monitoring, testing ,and deployments, teams can quickly address issues, which leads to improve software quality and user satisfaction.

▢ The DevOps lifecycle:

Outline the key phases of the DevOps lifecycle and their importance.

The DevOps lifecycle is a continuous process that involves several stages aimed operations to improve collaboration and efficiency. Here's a brief overview of the key stages in the DevOps lifecycle:

1. **planning:**
- Define project requirements and objectives.
- Create a road map for the project.
2. **Development:**
- Writing and Reviewing the code.
- Testing the code.
3. **Testing:**
- performs Automated testing to ensure code quality.
- Identify errors in the development process.
4. **Continuous Integration:**
- Runs Automated testing to ensure code quality.
5. **Continuous Deployment:**
- Continuous deployment to push changes to product quickly.
6. **Monitoring:**
- Monitor application performance.
- System health and optimization.
7. **Feedback:**
- Gathering feedback from users.
- use feedback for improvement in project.

▢ Benefits of DevOps in software development:

Discuss how adopting DevOps can improve software delivery and collaboration.

1. **Efficiency:**
- Automating repetitive tasks such as testing, deployment and minimize the risk of human resources.
- Efficiency leads to Cost savings and time saving.
2. **Faster Updates:**

- Continuous delivery practices enable quicker updates to applications.

3. **Higher quality software:**
- continuous testing improves software quality.

4. **Reliability:**
- The stable operation of software applications and services.

5. **Improved Collaboration:**
- Improves a collaboration between development and operation teams.
- Encouraging shared responsibility.

6. **Cost savings:**
- Efficient processes, fast recovery contribute to cost savings.

▢ Explore DevOps Tools:

Choose one tool from each category(CI/CD, Containerization, infrastructure as a code)and provide:

A brief overview (150-200 words) of the tool, its primary purpose, and how it fits into the DevOps lifecycle.

Examples of how this tool can be used on AWS, Azure, and GCP. Include links to the official documentation or relevant resources.

1.CI/CD: Jenkins

Description: Jenkins is an automation server that helps coders continuously integrate and deliver their code. It allows fast software releases with better quality of software due to the automation of the build, test, and deploy processes.

DevOps Lifecycle:

Jenkins is a critical component in the CI/CD pipeline. It automates the following stages:

Continuous Integration (CI):

Builds and tests code changes automatically as soon as they are committed to the version control system.

This makes it easier to catch integration problems early.

Continuous Delivery (CD):

Deploy validated code changes automatically to various environments (staging, production, etc.)

Examples

AWS:

Integrate with AWS Code Pipeline to build and manage CI/CD pipelines
Use AWS EC2 instances to run your Jenkins servers
Use AWS services like S3 to host your artifacts and Code Deploy to do deployments

Azure:

Enable use through Azure DevOps Services with CI/CD pipeline
Run Jenkins on Azure Virtual Machines or AKS.
c) Use of Azure Blob Storage to store artifacts, and deployment using Azure Pipelines.

GCP:

a) Google Cloud Build can be used for implementing the CI/CD pipeline.
b) Jenkins can be run on Google Compute Engine or GKE.
c) Use of Google Cloud Storage to store artifacts and Google Cloud Run to deploy the application.

2. Containerization: Docker:

Docker is an open-source platform for developing, shipping, and running a containerized application. It packages applications and their dependencies together inside containers so that the applications run uniformly regardless of their environment.

DevOps Lifecycle:

Docker aids in the following: Development-snowballing and testing software in isolated environments;
Deployment-deploying its applications easily on any environment-on-premises, cloud, or hybrid;
Scalability-scale down or up with demand.

Examples

AWS

Run Docker containers on Amazon ECS (Elastic Container Service) or Amazon EKS (Elastic Kubernetes Service).
Use AWS Far gate for container orchestration in a serverless approach.

Azure:

Run Docker containers on Azure Container Instances or Azure Kubernetes Service (AKS)

GCP:

Run Docker containers on Google Kubernetes Engine (GKE) or Google Cloud

Run.

3. Infrastructure as Code (IaC): Terraform

Terraform is a completely open source, infrastructure-as-code tool enabling one to describe and create both physical and virtualised cloud and network infrastructures through declarative configuration files.
DevOps Lifecycle: Terraform enables infrastructure provisioning. With the help of terraform, we can automate creation and management of the infrastructure resources.
Version control: Track infrastructure changes as one would code, allowing better collaboration and easier rollbacks.
Consistency: Ensure that infrastructure deployments are consistent across various environments.
Examples:
AWS:
Provision different services of AWS, including EC2, S3, RDS, etc.

Azure:

Set up a multi-tier application with VMs in different availability zones.

GCP:

Deploy a Kubernetes cluster on Google Kubernetes Engine (GKE).

▢ Tools Overview:

CI/CD Tools: Jenkins, GitHub Actions .

Containerization Tools: Docker, Kubernetes.

Infrastructure as Code Tools: Terraform, AWS CloudFormation, Azure Resource Manager (ARM) templates.

- **CI/CD tools**
  • **Jenkins**: Jenkins is an automation server that provides totally free and opensource continuous integration and continuous delivery pipelines, auto-automates build, testing, and the deployment of the codes.
  • GitHub Actions: It is a CI/CD platform natively built in GitHub, allowing

users to automate any software workflow, including builds, tests, and deployments, right inside their repositories.

- **Containerization tools**
  •**Docker**: Docker is the platform through which users can create, execute, and manage applications using containers. Its Docker utility simplifies the process of packaging and deployment of an application because it isolates an application along with all of its dependencies into a container.
  •Kubernetes is an open-source system that manages containerized applications across multiple clusters of hosts and automates their deployment, scaling, and management.
- **Infrastructure as Code Tools**
  • **Terraform:** An open-source infrastructure as a code tool based on declarative language used in defining and providing infrastructure resources to include servers, network, and also storage.
  •**AWS CloudFormation :** The service provisioned by the AWS in generating and provision AWS resources based on the description within JSON or YAML.
  • **Azure Resource Manager (ARM) templates:** A service offered by Azure that enables users to model and provision Azure resources using templates written in JSON.