Exercise 1: Cloud Computing Models (AWS Elastic Beanstalk)

1. **Main Differences Between IaaS, PaaS, and SaaS**:
    - **IaaS** provides control over infrastructure (virtual machines, networks), giving flexibility for custom environments. **PaaS** abstracts infrastructure, offering a platform for developers to build and deploy apps without worrying about the underlying systems. **SaaS** provides fully-managed software applications for users, with no control over the backend infrastructure.
2. **AWS Elastic Beanstalk** falls under **PaaS**. It simplifies application deployment by managing the infrastructure, while still giving some control over configuration.
3. **Real-World Examples**:
    - **IaaS**: Hosting a custom e-commerce site requiring specific server setups.
    - **PaaS**: Deploying a web app on **Elastic Beanstalk** to focus on code without managing servers.
    - **SaaS**: Using **AWS WorkMail** for business email services.

Exercise 2: Exploring AWS Core Services (Elastic Beanstalk)

1. **Compute Engine Equivalent in AWS (Elastic Beanstalk)**: A PaaS that automatically handles deployment, from capacity provisioning to load balancing.
    - **Use Case**: Simplified deployment of web applications.
2. **GKE Equivalent in AWS (Elastic Beanstalk)**: Elastic Beanstalk manages containerized apps but is easier to set up than GKE, which is more complex.
    - **Use Case**: Containerized apps with managed infrastructure.
3. **App Engine Equivalent in AWS**: Elastic Beanstalk serves the same purpose of deploying applications quickly without managing underlying hardware.
    - **Use Case**: Web app deployment with auto-scaling and infrastructure management.
4. **Cloud Storage Equivalent in AWS**: **S3** is used for storing unstructured data like images or files.
    - **Use Case**: Media storage for applications or backups.
5. **BigQuery Equivalent in AWS**: **Redshift** for data warehousing and large-scale queries.
    - **Use Case**: Data analysis and BI for large datasets.

Exercise 3: Creating and Managing Applications with AWS Elastic Beanstalk

1. **Steps to Create an Application on Elastic Beanstalk**:
    - Logged into AWS Console and navigated to Elastic Beanstalk.

    - Created a new environment and uploaded the application code.
    - Elastic Beanstalk automatically handled the infrastructure provisioning, scaling, and deployment.
2. **Connecting to the Environment**:
    - Elastic Beanstalk deploys the app and provides a URL for accessing it.
    - No need to manually configure SSH unless accessing the EC2 instances directly for debugging.
3. **Difference Between Stopping and Terminating**:
    - **Stopping**: Suspends the app but keeps the infrastructure (EC2 instances, load balancer, etc.) intact.
    - **Terminating**: Deletes the entire environment, removing all resources permanently.

Exercise 4: Deploying a Containerized Application on AWS Elastic Beanstalk

1. **Creating and Pushing a Docker Container**:
    - Built the Docker container locally.
    - Pushed the container image to **Amazon Elastic Container Registry (ECR)**.
2. **Deploying on AWS Elastic Beanstalk**:
    - Created a new environment in Elastic Beanstalk and selected Docker as the platform.
    - Deployed the container by linking the ECR image to the environment.
3. **Verifying Deployment**:
    - After deployment, Elastic Beanstalk provided a URL to access the running containerized app.
    - Verified accessibility by checking the app at the provided URL.