Room1 - EC2 & S3

1. EC2 & S3

Q1 (Easy)

Scenario: A developer needs to launch a small web server for testing.
Task: Launch a t2.micro EC2 instance with Amazon Linux, install Apache, and make sure the default page is accessible publicly.
Expected Outcome / Hint: Access the EC2 public IP in a browser to see the Apache test page.
Level: Easy

Q2 (Easy)

Scenario: You are asked to store application logs centrally.
Task: Create an S3 bucket and upload at least 3 log files manually using the console.
Expected Outcome / Hint: Verify that the logs are visible in the bucket and accessible as per permissions.
Level: Easy

Q3 (Medium)

Scenario: A team wants to automate file backup from EC2 to S3.
Task: Create an IAM role for EC2 with S3 full access, attach it to an instance, and use AWS CLI to copy files to S3.
Expected Outcome / Hint: Run `aws s3 cp` command successfully without access errors.
Level: Medium

Q4 (Hard)

Scenario: Your EC2 instance stores critical files that must be synced automatically to S3 every 6 hours.
Task: Configure a cron job or systemd timer to sync `/var/data` to S3 using AWS CLI.
Expected Outcome / Hint: Files auto-sync to S3 at intervals without manual triggers.
Level: Hard

Room2 - VPC & IAM

2. VPC & IAM

Q1 (Easy)

Scenario: You need to isolate a testing environment.
Task: Create a custom VPC with one public subnet and launch an EC2 instance inside it.
Expected Outcome / Hint: Verify instance connectivity using SSH.
Level: Easy

Q2 (Easy)

Scenario: You must securely allow an intern to access AWS resources.
Task: Create an IAM user with console access, assign "ReadOnlyAccess" policy, and share login details.
Expected Outcome / Hint: Intern can view but not modify AWS resources.
Level: Easy

Q3 (Medium)

Scenario: You want EC2 in a private subnet to access the internet.
Task: Set up a NAT Gateway in a public subnet and configure route tables.
Expected Outcome / Hint: EC2 in private subnet can run `yum update` successfully.
Level: Medium

Q4 (Hard)

Scenario: Security requires that only certain IPs access EC2 and IAM users rotate passwords every 90 days.
Task: Configure VPC security groups to whitelist IPs and enforce password policy in IAM settings.
Expected Outcome / Hint: EC2 accepts connections only from specific IPs; password policy is active.
Level: Hard

Room3 - RDS & Lambda

3. RDS & Lambda

Q1 (Easy)

Scenario: A small application needs a relational database.
Task: Create an RDS MySQL instance with default settings and connect via MySQL client.
Expected Outcome / Hint: Run a simple SQL query to confirm connection.
Level: Easy

Q2 (Easy)

Scenario: You want to automate image resizing when files are uploaded to S3.
Task: Create a Lambda function triggered by S3 uploads that logs file names in CloudWatch.
Expected Outcome / Hint: File names appear in CloudWatch logs when uploading to S3.
Level: Easy

Q3 (Medium)

Scenario: You need to automatically insert user registration data into RDS from a Lambda function.
Task: Create a Lambda with VPC access, connect to RDS MySQL, and insert data via SQL query.
Expected Outcome / Hint: Check database table for new rows after Lambda execution.
Level: Medium

Q4 (Hard)

Scenario: Create a serverless backend that triggers Lambda from API Gateway and stores results in RDS.
Task: Build an API endpoint → integrate with Lambda → Lambda inserts request details into RDS.
Expected Outcome / Hint: Test API via Postman; data appears in RDS table.
Level: Hard

Room4 - Git & Jenkins

4. Git & Jenkins

Q1 (Easy)

Scenario: A team wants to maintain source code versioning.
Task: Initialize a Git repository, add files, and push to GitHub.
Expected Outcome / Hint: Repository visible on GitHub with commit history.
Level: Easy

Q2 (Easy)

Scenario: You need to automate a build every time new code is pushed.
Task: Install Jenkins and configure a freestyle job triggered by GitHub webhook.
Expected Outcome / Hint: New commits trigger automatic Jenkins builds.
Level: Easy

Q3 (Medium)

Scenario: You want Jenkins to build, test, and deploy a Node.js app.
Task: Configure Jenkins pipeline using Jenkinsfile with stages: Build → Test → Deploy.
Expected Outcome / Hint: Successful pipeline run through all three stages.
Level: Medium

Q4 (Hard)

Scenario: Build a complete CI/CD setup using Jenkins and Git for an app deployed on EC2.
Task: Use Git → Jenkins → EC2 pipeline; Jenkins deploys code automatically on EC2 via SSH.
Expected Outcome / Hint: Code updates reflect live on EC2 without manual deployment.
Level: Hard

Room5 - Docker & Kubernetes

5. Docker & Kubernetes

Q1 (Easy)

Scenario: A developer needs a consistent environment for an app.
Task: Create a Dockerfile for a simple Python app and run a container locally.
Expected Outcome / Hint: Access the app on localhost in a browser.
Level: Easy

Q2 (Easy)

Scenario: You want to share your Docker image.
Task: Push the image to Docker Hub under your account.
Expected Outcome / Hint: Image visible on your Docker Hub repository.
Level: Easy

Q3 (Medium)

Scenario: You need to run multiple containers for frontend and backend together.
Task: Create a `docker-compose.yml` with two services and verify communication between them.
Expected Outcome / Hint: Both containers run and communicate over a common network.
Level: Medium

Q4 (Hard)

Scenario: Deploy a containerized app to Kubernetes.
Task: Create Deployment and Service YAML files and expose the app externally.
Expected Outcome / Hint: Access app via NodePort or LoadBalancer service.
Level: Hard

Room6 - Terraform

6. Terraform

Q1 (Easy)

Scenario: You want to automate EC2 creation.
Task: Write Terraform code to launch a t2.micro instance in a default VPC.
Expected Outcome / Hint: Instance visible in AWS console after `terraform apply`.
Level: Easy

Q2 (Easy)

Scenario: You need to create and manage an S3 bucket via code.
Task: Write Terraform configuration for S3 with versioning enabled.
Expected Outcome / Hint: Bucket created and listed under S3 console.
Level: Easy

Q3 (Medium)

Scenario: You want to create a reusable infrastructure module.
Task: Build a Terraform module for EC2 creation with variable inputs (AMI, instance type).
Expected Outcome / Hint: Multiple EC2s can be created using the same module.
Level: Medium

Q4 (Hard)

Scenario: Build a complete Terraform-based environment (VPC, Subnets, EC2, and S3).

Task: Use multiple resources, outputs, and variables to define the infrastructure end-to-end.
Expected Outcome / Hint: All resources visible in AWS after `terraform apply` and destroy cleanly.
Level: Hard