1. **a.** Initialize a new Git repository in a directory and push it to a remote repository on GitHub/GitLab.  
   **b.** Run a Docker container using an existing image (e.g., Nginx) and map a local port to a container port.
2. **a.** Create and checkout a new branch, make changes to a file, commit the changes, and merge it back into the main branch.  
   **b.** Create a multi-container application using a docker-compose.yml file to define a web server and a database.
3. **a.** Resolve a merge conflict between two branches, commit the resolution, and push the changes.  
   **b.** Deploy a simple web application using Kubernetes and expose it via a service.
4. **a.** Tag a specific commit with a version number (e.g., v1.0.0) and push the tag to the remote repository.  
   **b.** Use kubectl to scale a deployment to 3 replicas and verify the scaling operation.
5. **a.** Revert the repository to a previous commit and explain the steps taken.  
   **b.** Create a Kubernetes pod using a YAML file that runs an Nginx web server.
6. **a.** Write a Terraform configuration to provision an EC2 instance and ensure it is accessible via SSH.  
   **b.** Create a Kubernetes deployment and expose it via a LoadBalancer service type.
7. **a.** Modify the Terraform configuration to create an S3 bucket and enable versioning on it.  
   **b.** Use Jenkins to create a simple pipeline that builds a Docker image and pushes it to Docker Hub.
8. **a.** Provision a security group with inbound rules for HTTP (port 80) and SSH (port 22) using Terraform.  
   **b.** Write a Dockerfile to build a simple web application and run it inside a container.
9. **a.** Run the necessary Terraform commands to initialize, plan, and apply a configuration that provisions a resource.  
   **b.** Inspect a running Docker container using the docker inspect command and display its logs.
10. **a.** Configure a remote backend to store the Terraform state file in an S3 bucket.  
    **b.** Deploy a simple web application in Kubernetes using Helm.
11. **a.** Write an Ansible playbook that installs and starts a web server on a remote machine.  
    **b.** Create a Kubernetes namespace and deploy an application to that specific namespace.
12. **a.** Create an Ansible role that installs MySQL and ensures the service is running on a target machine.  
    **b.** Use kubectl to get the list of pods and display their status in a Kubernetes cluster.
13. **a.** Write an Ansible playbook to modify a configuration file (e.g., /etc/hostname) on a remote machine.  
    **b.** Create a Kubernetes persistent volume (PV) and persistent volume claim (PVC), then mount it to a pod.
14. **a.** Use Ansible to restart a service on a remote server and confirm it is running after the restart.  
    **b.** Update the image of a Kubernetes deployment and trigger a rolling update.
15. **a.** Use the Ansible setup module to gather facts about a remote system and display them.  
    **b.** Use Jenkins to automate the deployment of a web application to a Kubernetes cluster.
16. **a.** Write a Terraform script to provision an RDS instance on AWS.  
    **b.** Deploy a Docker container for a MySQL database and ensure it's accessible externally.
17. **a.** Write a Terraform script to provision an EC2 instance with a specific tag.  
    **b.** Use Kubernetes to create a deployment for a Node.js app and scale it to 3 replicas.
18. **a.** Create a Terraform plan and apply it to provision a security group in AWS with specific access rules.  
    **b.** Create a Docker Compose setup with an Nginx container and a MySQL container, ensuring they can communicate.
19. **a.** Use Git to clone a repository, create a branch, and push changes back to the remote repository.  
    **b.** Deploy an Nginx application on Kubernetes, exposing it via a LoadBalancer service.
20. **a.** Create a GitHub Actions workflow file to automate the process of building and testing code.  
    **b.** Set up Jenkins to run automated tests on a Docker container whenever new changes are pushed to a repository.