Environment Access Instructions

Dear Candidate,
You have been assigned a dedicated environment for your technical interview. Please find your access details below:

```
SSH Access: ssh root@<IP>
Password:e12UIaYQAHLs0jk


GitHub Repository: https://github.com/Reeltor/devops-interview
```

Task 1: GitHub Actions CI/CD for Node.js Application

Initial Setup

1. Fork the provided repository
2. Clone your forked repository
3. Create branches named 'dev' and 'prod'

Objectives

1. setup `github` workflow
2. setup self-hosted runner
3. Host the provided repo using `pm2`
4. Setup `NGINX reverse proxy`
5. connect with a custom domain
6. setup `ssl` using `certbot`
7. Test the hosted project on local browser.

Successful Criteria

1. Repository Management

- [ ] Successfully forked and cloned repository
- [ ] Created 'dev' and 'prod' branches

- [ ] Implemented proper branch protection rules
- [ ] Maintained clean commit history

2. GitHub Actions Workflow

- [ ] Correct workflow file structure in `.github/workflows/`
- [ ] Proper trigger configuration (on push to dev)
- [ ] Working build process
- [ ] Successful deployment process
- [ ] Error handling implemented

3. Self-hosted Runner

- [ ] Runner successfully registered and connected
- [ ] Runner showing as online in GitHub Actions
- [ ] Proper runner labels configured
- [ ] Runner executing jobs successfully

4. Application Deployment

- [ ] PM2 properly installed and configured
- [ ] Application running without errors
- [ ] PM2 ecosystem file properly configured
- [ ] Working process management
- [ ] Proper logging setup
- [ ] Automatic restart on failure

5. Nginx Configuration

- [ ] Correct Nginx configuration file
- [ ] Working reverse proxy to Node.js application
- [ ] Proper HTTP headers configured
- [ ] Error pages configured
- [ ] Logging properly setup

6. Domain & SSL

- [ ] Domain properly connected
- [ ] SSL certificate successfully installed
- [ ] Automatic HTTPS redirect
- [ ] Valid SSL configuration

- [ ] Certificate auto-renewal configured

Task 2: Deploy the similar Application in dockerized environment

Objectives

1. **Containerisation**
   - Create Dockerfile
   - Build `Node.js` application image
   - Setup proper layering and caching
   - Implement multi-stage builds
2. **Docker Compose**
   - Create `docker-compose.yml`
   - Configure services:
     - `Node.js` application
     - `Nginx` reverse proxy
   - Setup proper networking
   - Configure volume mounts

Successful Criteria

1. Dockerfile Implementation

- [ ] Proper base image selection
- [ ] Multi-stage build implemented
- [ ] Efficient layer caching
- [ ] Security best practices followed
- [ ] Environment variables properly used
- [ ] Working build process
- [ ] Optimized image size

2. Docker Compose Setup

- [ ] Valid docker-compose.yml syntax
- [ ] Proper service definitions
- [ ] Environment variables configured
- [ ] Volume mounts correctly setup
- [ ] Networks properly configured

- [ ] Container dependencies defined

3. Application Configuration

- [ ] `Node.js` application running in container
- [ ] `Nginx` reverse proxy working
- [ ] Inter-service communication working
- [ ] Proper error handling
- [ ] Logging configured correctly
- [ ] Environment variables properly passed

4. Performance & Security

- [ ] Resource limits configured
- [ ] Health checks implemented
- [ ] Security best practices followed
- [ ] No root user in containers
- [ ] Proper network isolation
- [ ] Secrets properly managed

Task 3: Infrastructure as Code with Terraform

Objectives

1. **Create Terraform Configuration**
   - Write terraform configuration to provision:
     - 2 DigitalOcean droplets (2GB RAM each)
     - 1 Load Balancer
     - Configure firewall rules
2. **Infrastructure Requirements**

```
- Region: nyc1
- Droplet Size: s-2vcpu-2gb - Ubuntu 22.04 x64
- Private networking enabled
```

3. **Success Criteria**
   - [ ] Working Terraform state management
   - [ ] Proper variable organization
   - [ ] Resource tagging implemented
   - [ ] Output values configured

- [ ] Security groups properly configured

Task 4: Configuration Management with Ansible

Duration: 45 minutes

Objectives

1. **Create Ansible Playbook to:**
   - Configure both servers from Task 3
   - Install required packages:
     - Docker
     - Node.js
     - Nginx
   - Setup monitoring (Node Exporter)
   - Configure users and SSH access
2. **Playbook Requirements**

   yaml

   Copy

   `– Use roles structure – Implement handlers – Use variables – Create custom module`
3. **Success Criteria**
   - [ ] Idempotent playbook execution
   - [ ] Proper error handling
   - [ ] Role-based access configuration
   - [ ] Working monitoring setup
   - [ ] Documented variables

Task 5: Jenkins Pipeline Implementation

Duration: 60 minutes

Objectives

1. **Setup Jenkins Pipeline**

   groovy

   Copy

`- Create Jenkinsfile - Multi-stage pipeline - Parallel execution where applicable - Integration with SonarQube`

2. **Pipeline Requirements**
   - Stages:
     - Checkout
     - Build
     - Test
     - Code Quality (SonarQube)
     - Security Scan
     - Deploy to Staging
     - Integration Tests
     - Deploy to Production
3. **Required Integrations**
   - GitHub webhook
   - Slack notifications
   - Email notifications
   - Artifact archival
4. **Success Criteria**
   - [ ] Working pipeline with all stages
   - [ ] Proper error handling and notifications
   - [ ] Archive artifacts
   - [ ] Test reports generation
   - [ ] Integration with external tools