doctorly DevOps technical task

The focus of this assignment is to evaluate your skills as a potential doctorly DevOps employee within our cross functional teams. This is not a hardcore test and its aim is to allow you to express your opinions and knowledge.

- Results as well as explanation/documentation should be submitted on Github or Gitlab and develop the task in a similar manner as to how you would develop a task for your work
- The tasks should be completed within at most 4 hours total time
- Please choose 3 tasks from the list below that align with your strongest skills
- We request that you send us an email with your submission no later than 4 hours after receiving the task
- We do not expect you to build a full system in the couple of hours assigned to this task. Instead, we want to see a reasonable start. Get as far as you can. This can be the scaffolding and some architectural choices that will serve as the prototype for future development. Your answers can be showcased in multiple ways, code, diagrams, research paper etc.
- We will schedule a meeting after the completion of the tasks and our review of them to discuss them with you

Task #1

Create Ansible playbooks and roles to create locally or remotely a docker stack using docker-compose that runs a .net core application and a database (Postgresql or MySQL)

Task #2

1. How would you structure your Terraform project if you have multiple environments and use different cloud providers?
2. Write a Terraform script with the below requirements:
    - Creates a ubuntu aws ec2-instance
    - Install ansible
    - Execute an ansible playbook (optional: Use Ansible script from Task #1 or any other)

Task #3

If you have multiple Ubuntu prod instances, How would you monitor them? What would be your monitoring strategy?

Task #4

Set up a basic CI/CD pipeline for a sample application using any preferred CI/CD tool, with GitLab being a recommended option.
Requirements:

- You should use a sample application of your choice, such as a simple web application or a command-line utility.
- Your pipeline should have at least two stages: a build stage and a test stage.
- Your build stage should compile the application code and produce a build artifact, such as a JAR file, an executable binary, and/or a Docker build.
- You should use Docker to build and run your application in the pipeline.
- You should use your own runners or a cloud-based service to execute the pipeline.
- You should include a README file that explains how to use your pipeline and any assumptions you made during the setup.

Task #5

Create a Helm Chart for a WordPress Application

In this assignment, you will create a Helm chart for a WordPress application. The Helm chart should include all the necessary Kubernetes manifests to deploy WordPress, but should not actually deploy the application.

Requirements:

- Create a Helm chart: Create a Helm chart for WordPress. The chart should include a values.yaml file that allows for customization of the WordPress installation (e.g. database credentials, site URL, etc.).
- Configure Kubernetes manifests: The Helm chart should include all the necessary Kubernetes manifests to deploy WordPress, including deployments, services, and any required Kubernetes resources (e.g. secrets, configmaps).
- Test the Helm chart: Run a dry-run installation of the Helm chart using the *helm install --dry-run* command to ensure that all Kubernetes resources are created correctly.
- Validate the chart: Use the *helm lint* command to validate the Helm chart and ensure that it meets best practices and conventions.
- Cleanup: Once you have finished testing and validating the Helm chart, delete any resources that were created during the testing process.