DevOps Assignment

Introduction

Our take-home assignment is to test your skills on

- Doing basic cloud-native oriented operating job
- Doing basic coding that can help automate operations

Tips

- You can take reference from others' work.
- Comment your code as much as possible.
- Screenshots can better describe your work process.
- For things that need to be deployed, also give us access to your service or cluster (if any).
- You can use any programming language to do the task, but it would be better to select from Python, Golang, Rust, or JavaScript.

Outputs

Please upload all your files, including any supporting documents (e.g., README), to a GitHub repository and invite the following users to the repository:

- https://github.com/liao1
- https://github.com/melynx

Missions

💡 You must choose at least two of the missions below.

💡 Complete as many tasks as you can within the time limit.

Mission 1: Terraform

Before launching any of your services or solutions, you'll need an infrastructure deployed. Please write a Terraform script to deploy any of the services on AWS that you would require for the tasks below.

You can make use of an AWS free tier account for some of your tests:

https://aws.amazon.com/free/

Objectives:

1. Write a Terraform script to deploy the infrastructure required by your solution for the tasks below.

2. Use a backend to maintain your Terraform state file (e.g., S3 backend).

Mission 2: Blockscout

Blockscout (https://github.com/blockscout/blockscout) is a blockchain explorer for inspecting, analyzing, and interacting with EVM chains and rollups.

While there are several explorers available to blockchain projects, most are closed systems (e.g., Etherscan, Etherchain) which are not independently verifiable. Blockscout provides a much needed open-source alternative. As the multi-chain paradigm continues to take hold in both private and public settings, transparent tools are needed to analyze and validate transactions.

Your mission is to deploy Blockscout against an existing blockchain - Ethereum Sepolia. Your deployment should be able to index the chain and present the information via the Blockscout UI.

Objectives:

1. Write a Helm chart of Blockscout that can deploy
    a. One database, either on RDS free tier or an in-cluster solution
    b. One indexer
    c. Scalable API servers with HPA
    d. Blockscout v2 WebUI https://github.com/blockscout/frontend
2. Use a CI system to package and publish your helm charts.
3. (optional) Deploy a ERC20 contract to Ethereum Sepolia testnet and verify it in your Blockscout.

Mission 3: Contract Indexer

Objectives:

1. Write a ERC20 contract indexer that indexes the activities of USDT contract
https://etherscan.io/token/0xdac17f958d2ee523a2206206994597c13d831ec7
2. The indexer should expose metrics in a format that follows OpenMetrics spec
https://github.com/OpenObservability/OpenMetrics/blob/main/specification/OpenMetrics.md
3. Deploy your indexer and write the metrics to Grafana Cloud's Prometheus
4. Create a dashboard in Grafana Cloud with panels like:
    - tx/s
    - token transfered/s
5. Create an alert rule that triggers when there's a transaction that transferred a large amount of tokens

Mission 4: Node and Stress Testing

This mission requires you to learn how to run a private blockchain, and stress test it.

Objectives:

1. Run a (Dockerized) Ethereum private testnet with geth
https://geth.ethereum.org/docs/fundamentals/private-network
2. Write your own stress tester or implement an existing stress testing solution:
    a. Normal transfer
    b. ERC20 transfer
    c. (optional) ERC721 NFT minting
3. Plot your result with the resource (CPU, memory, etc.) usage of the node as a graphical chart

> 💡 If you cannot get #1 done in time, you can use a `geth --dev` node to do the stress test

Mission 5: Optimism Stack

In order to test how good the candidate is on:

1. Quickly understanding and deploying a new blockchain
2. Writing code to automate the deployment
   We give you the following tasks:

3. Follow this doc https://stack.optimism.io/docs/build/getting-started to deploy a “OP stack devnet” in given machine.
4. Write a tool in golang to automate the deployment process above.
5. Set up a kind cluster in the given machine, deploy a op stack in it.
6. Write a tool in golang to automate the deployment process above.
7. Understand the op-stack, and we’ll have a meeting to let you describe it to us.

> 💡 You can choose to skip tasks 1 and 2 if you are confident to complete task 3 within your time limit.