Tech challenge Senior SRE - July ’24

Introduction

Thank you for expressing your interest in the Senior Site Reliability Engineer position at Prima.
Please make yourself comfortable and take a moment to review the test requirements carefully.
You will have one week to complete the test, and we expect you to spend no more than 4 hours on it. We understand that you may have other commitments, so we want to provide you with enough time to tackle the challenge. If you decide not to proceed with the test, that is completely fine, but please inform us of your decision.
If you have any questions about anything, please feel free to ask. It's better to be safe than sorry! We are thrilled to receive your results. Enjoy the process and good luck!

Important Information

Please ensure that you store your completed solution in a private Git repository on Github. Once you have finished the solution, please invite at least two collaborators from this list:

- https://github.com/depontimatteo
- https://github.com/francesco995
- https://github.com/EugenioLaghi
- https://github.com/jasonlinhamprima
- https://github.com/gianluca-mascolo

and let the recruiter know you completed the task.
We will fork your private Github repository and review your tech challenge.
Kindly note the following guidelines:

- Ensure that the solution is not stored in a public repository to prevent future candidates from accessing the solutions online.
- Your solution should be able to build and run on Ubuntu Linux.
- It is important that the solution you provide is entirely your own work. While it is acceptable to use small code fragments from online resources, make sure to cite these sources appropriately.
- Additionally, you may utilise third-party libraries or modules where necessary.

During the technical interview, be prepared to explain and discuss your solution. It will be evaluated not only based on its ability to produce correct output but also on its documentation, maintainability, test coverage and ease of building and using.

Reliability-Driven Tech Challenge: Building and Deploying a Python Application with Docker, Infrastructure as Code, and Kubernetes

Objective:

In this tech challenge, your objective is to showcase your ability to design, develop, and

deploy a reliable Python application using Docker, Infrastructure as Code (IaC), and Kubernetes with a strong emphasis on ensuring system reliability and robustness. You will be assessed on your proficiency in various aspects of infrastructure management, including containerisation, IaC implementation, and Kubernetes orchestration. This challenge aims to evaluate your problem-solving skills, attention to reliability, and understanding of contemporary DevOps practices.

Solution Overview:

Task 1: Develop the Python API Server

Begin by developing a API Server using Python as the chosen programming language, with two essential endpoints:

- `GET /users`: Retrieve a list of all users.
- `POST /user`: Enable the creation of a new user, including the upload of an image as avatar. If you aren't familiar with Python Web Development, you can start from the template prima-tech-challenge provided in this repo. Otherwise, you can write your own from scratch.

To enhance the reliability and maintainability of your application, please adhere to the following guidelines:

Structure your application code in a well-organised manner, following coding standards and separation of concerns.
Implement data persistence mechanisms with DynamoDB to securely store user information. This will not only extend the functionality of your application but will also provide an opportunity to demonstrate your commitment to reliability.
Implement object storage mechanism with S3 to securely store your avatar
Apply basic error handling techniques to handle common scenarios, such as invalid requests or database errors.

The expectations for the Python API Server are:

Both routes are working, the GET /users is returning a structure like this one:

```
[
	{
		"avatar_url":
"https://prima-tech-challenge.s3.us-east-1.amazonaws.com/test-avatar.png",
		"email": "test-user@prima.it",
		"name": "Test User"
	}
]
```

The image upload to S3 is working
The DynamoDB database contains a list of items with this structure

```
	{
		"name": {
			"S": "Test User"
		},
		"avatar_url": {
```

```
                "S":
"https://prima-tech-challenge.s3.us-east-1.amazonaws.com/test-avatar.png"
            },
            "email": {
                "S": "test-user@prima.it"
            }
        }
```

Task 2: Dockerise Your Application

Next, containerise your Python API server using Docker, ensuring that it can be seamlessly deployed in containerised environments.

Task 3: Infrastructure as Code (IaC)

In this step, employ IaC principles and tools like Terraform or Pulumi to orchestrate the creation of the DynamoDB Table, S3 bucket and a role to access it. As mentioned before, the application needs a set of AWS resources to work correctly, you can use both Localstack CE or an Official AWS Account (consider the Free Tier option).

Task 4: Kubernetes Deployment with Helm

Now, create a custom Helm Chart that includes templates for deploying your Python API server within the Kubernetes cluster. Define Helm Chart templates to encapsulate deployment, configuration, and scaling aspects of your application within the Kubernetes cluster. Focus on the reliability of the microservice (please consider autoscaling, healthchecks, and so on).

You can test your Kubernetes Deployment with a local K8S cluster (e.g. minikube). If you're using Localstack CE for your AWS resources, configure bind addresses to 0.0.0.0 otherwise your AWS mocked services won't be reachable from your deployment.

Also, please provision all the essential resources, including AWS IAM roles and policies for K8S ServiceAccounts, required for authenticating and operating with an EKS cluster. You can skip the testing part (the EKS control plane costs even if you are using an AWS Free Tier, and isn’t supported by Localstack CE), but the AWS resources should be coded with IaC.

Please provide clear and concise documentation explaining the solution architecture, including how to use the scripts and any prerequisites.