empatica

Cloud Engineer Assignment

2021

Dear candidate,

First of all, thanks for your application in Empatica, you have passed the first step of the hiring process. Congratulations! Your profile is very interesting, and we'd like to give you the. possibility. To show us that you don't have only an interesting profile, but you are also talented and passionate about: architecting, creating, and maintaining cloud infrastructure!

This document describes your assignment. Please **pick one** of the following task, make sure you read its requirements very carefully, and start scripting! **The result should be delivered as a Git repository.**

If you have any questions or need any help to better understand the problem, please don't hesitate to let us know. Thanks!

Good luck!

CHALLENGE 1 - DEPLOY

Create in your favourite language that orchestrates the deploy of the following components

- A basic web app having the following components
    1. A backend written in go exposing a set of RESTful API
    2. A web app written in Angular connecting to the backend API
- A MySQL database running in a RDS instance

Provide a way to test that zero-downtime deploy is actually causing no downtime :-)

REQUIREMENTS

- The deploy script checks out code from a git repository
- Use Terraform (or another Infrastructure as Code tool of your choice) in order to create all required AWS resources:
    - You can create a new AWS account in order to leverage on the free tier and not to be billed for resource costs.
    - It's important that every AWS resource should be created by the chosen *Infrastructure as Code* tool.
- The API is packed in a docker image and deployed on AWS ECS
- Static assets are uploaded to a S3 bucket that is connected to a CloudFront distribution
- Your solution should allow zero-downtime deploy based on AWS ECS. Should your solution need extra containers, or AWS services, go ahead and use them!

DELIVERABLES

- Terraform (or another *Infrastructure as Code* tool of your choice) source files
- Source code of scripts for deploy
- README explaining usage and architecture

CHALLENGE 2 - SERVERLESS APP + ELASTICSEARCH

Create a simple serverless application on AWS, consisting in two basic endpoints

- A GET endpoint to list available tasks
- A POST endpoint to create a new task

Ship logs generated by the application to an AWS Elasticsearch cluster

REQUIREMENTS

- The Lambda functions implementing the GET and the POST handlers should be written in Go, but we accept Python and Node as well, if you feel more proficient with these languages.
- Implement HTTP Basic Authentication on the POST endpoint, using an API Gateway Custom Authorizer
- Use the Serverless framework and/or Terraform in order to provision all required resources:
  - You can create a new AWS account in order to leverage on the free tier and not to be billed for resource costs.
  - It's important that every AWS resource should be created by the chosen *Infrastructure as Code* tool.

DELIVERABLES

- Terraform (or another *Infrastructure as Code* tool of your choice) source files
- Source code of Lambda functions
- README explaining usage and architecture

CHALLENGE 3 - CI PIPELINE WITH GITLAB ON KUBERNETES

Provision a Kubernetes cluster on AWS.
Deploy Gitlab and all required dependencies on Kubernetes.
Setup the Gitlab Review Apps feature in order to manage dynamic environments for every branch created in an example repo

You can use a simple hello-world style webapp in order to test and show us the Review Apps in action

REQUIREMENTS

- Use Terraform (or another Infrastructure as Code tool of your choice) in order to create all required AWS resources:
  - You can create a new AWS account in order to leverage on the free tier and not to be billed for resource costs.
  - It's important that every AWS resource should be created by the chosen *Infrastructure as Code* tool.
- There are many ways to provision a Kubernetes cluster on AWS: you're free to choose the best one for you (just don't forget to share in the README the reason of your choice.

DELIVERABLES

- Terraform (or another *Infrastructure as Code* tool of your choice) source files
- README explaining usage and architecture