Augmedix DevOps Code Challenges

This repository contains multiple scenarios using actual tools and concepts used by DevOps and Software Engineers at Augmedix. The goal is to demonstrate the ability to understand a technical need and use existing knowledge, quick-learning and familiarity with software development and operations concepts to achieve the stated outcome.

Goal

Select and develop the requested code samples for a minimum of 3 scenarios.

Scenarios

- GitHub Actions Workflows - Develop basic GitHub Actions Workflows triggered by several different events to perform basic build, test or other utility functions.
- Kubernetes Manifest Development - Develop Kubernetes manifests to deploy a simple system using several microservices
- Helm Chart Developmet - Develop a Kubernetes Helm Chart to deploy a basic microservice consisting of a web application server and queue worker.
- K6 Load Testing - Develop a simple load test using the k6 load testing framework.
- Terraform Configuration Management - Develop a Terraform module and template demonstrating the use of the module to provision some common AWS services.

Instructions

1. Review the Scenarios in this document
2. Develop code samples as described on the page for the chosen scenarios
3. Create an archive (zip/tgz) with the sample code for each scenario in it's own directory
   - github-actions
   - kubernetes
   - helm
   - k6
   - terraform
4. Transmit / Upload the archive as instructed when you received this document

Develop Some Basic GitHub Actions Workflows

GitHub Actions

Scenario

GitHub actions is an event-driven build, test and execution system powered by GitHub. Every GitHub user is given at least 2000 minute credits which can be used to execute GitHub Actions for private repositories, public repositories run for free. We want to write some basic workflows that each trigger in a different way and execute using hosted runners.

1. Write a workflow that builds a container with a basic webserver that displays a personal Top 10 list of movies, books, vacations, whatever you like. The workflow should trigger on push actions to the repo and the Top 10 webpage should be added to the container image from the repo. Suggested web server is Nginx
2. Write a workflow that is triggered manually via the GitHub Actions page on the repo. This workflow should spin up 2 different runners and run a sysbench load test for 60s.
3. Write a workflow that triggers on a schedule and checks for a newer version of the nginx container.

Acceptance Criteria

- 3 Workflows created
  - Basic webserver container build
    - Top 10 list of something in webserver
    - Triggers on push actions
  - Manual sysbench test with 2 runners
    - Sysbench for CPU runs for 60s
    - Triggered via GitHub API or GitHub UI
  - Scheduled workflow to check for newer version of webserver container
    - Checks for an updated version of webserver container used in build workflow
    - Clearly outputs whether there is an upgrade
    - BONUS if you commit, push and PR the newer tag to the repo

Develop A Basic Kubernetes YAML Manifest

Kubernetes

Scenario

Kubernetes is a common container orchestration and workload management system. We run almost all of our workloads in Kubernetes. In this scenario we want to demonstrate familiarity with the basic concepts and Kubernetes objects to enable a very simple web app running on Kubernetes using only native objects on the core APIs.

Create Kubernetes YAML Manifest For a Simple Web App

Write kubernetes YAML manifest(s) to run and deploy a set of simple services that serve and display your top 10 lists for movies, games, songs, etc. Each Top 10 list should be served by a separate service. Expose the services using a nodeport, loadbalancer or ingress controller. Ingress is provided by nginx-ingress behind an AWS ELB Load-Balancer.

Acceptance Criteria

- Deployment with a service and ingress
- Minimum 3 Top 10 lists generated
- Each Top 10 list is provided by a seperate service
- Top 10 lists should be accessible on paths like `/top10songs` , `/top10movies` , etc.
- Each list should have a link to the other lists

Suggested Approach / Guidance

- Use a basic public container image like nginx or apache
- Each service can be just a single, simple html page

Develop A Basic Helm Chart To Deploy a Service

Helm

Scenario

Helm is a package manager for Kubernetes that creates a service deployment template that can be preconfigured with sane default values. In most cases a simple `helm install <name> <chart>` command will deploy a named release, , with default values. The default values can be overriden by passing a values file or command line arguments. In this scenario we will write a basic helm chart (package) that would deploy and expose a simple service to the internet. Any simple web service will do, but an easy suggestion is `datawire/quote:0.5.0` which simply serves a quote via http on container port 8080.

Create a helm chart that deploys the simple service to a kubernetes cluster given the acceptance criteria below.

Acceptance Criteria

- Scalable deployment with a service and ingress
- values.yaml file the following defaults that can be overridden:
  - hostname: helm-example.fndev.net
  - tlsSecret: wildcard-fndev-net
  - replicaCount: 3

- chart compatible with helm v3 / api v2
- chart passes helm's built-in linting

Write a basic load test using k6

K6 Load Testing

Scenario

We have a service, osrm, that provides the estimated driving distance between 2 locations given 2 sets of GPS coordinates within the United States.

The USA can be considered a rectangle bound by the following Longitude and Latitude values: - West -124.73 - North 49.39 - East -66.95 - South 24.54

```
(-124.73,49.39)
               |----------------------------|
               |                            |
               |  USA LONG/LAT Rectangle    |
               |                            |
               ------------------------------ (-66.95,24.54)
```

The service we're testing, OSRM, takes a request of the format:
`https://<hostname>:<port>/route/v1/driving/<longitude_origin>,<latitude_origin>;<longitude_destination>,<latitude_destination>?overview`

The service returns JSON with details about the route:

```
$ curl 'https://osrm.meena.fndev.net/route/v1/driving/-93.2733259,44.9756602;-93.1062874,44.954394?overview=false'

{"code":"Ok","routes":[{"legs":[{"steps":[],
"distance":15187.8,"duration":942.6,"summary":"","weight":955.4}],"distance":15187.8,"duration":942.6,
"weight_name":"routability","weight":955.4}],
"waypoints":[{"hint":"lfoCgP___3-EvAAARwAAAEkAAAAAAAAAAAAAAEcAAABJAAAAAAAAAAAAAAD4YAIAkCkAAALEcPpKR64CEsNw-ixGrgIAABEPTjn1mA=="
,"name":"South 8th Street","location":[-93.273086,44.975946]},
{"hint":"cWtFgz9sRYMAAAAAAAAAAAE0AAABYAQAAAAAAAAAAAAAmAAAArAAAAAAAAAABA7M8BkCkAAE9Pc_r78q0CkU9z-hrzrQIBABEPTjn1mA==",
"name":"","location":[-93.106353,44.954363]}]}
```

Using k6, write a test that meets acceptance criteria below.

Acceptance Criteria

- Generate requests using a pair of random locations in the USA to retrieve routing information.
- Simulate 10 virtual users for 30s
- Simulate 50 virtual users for 60s
- Simulate 20 virtual users for 30s
- Use https://osrm.meena.fndev.net as the host for the tests (it's live)

Develop A Basic Terraform Module and Template

[Terraform]

Scenario

Terraform is a configuration management tool that declaritively manages infrastructure and services. It uses an idempotent approach and can detect and correct drift with clear indication of the changes necessary to restore the desired state. In this example we will create a terraform module and template to provision some basic resources in AWS for a microservice.

Create a Terraform module and template to provision the following resources for a new microservice named `customer-suggestion`. All resources created should use the microservice name. Use variables with defaults for required fields like security groups, s3 buckets, etc.

- S3 bucket
- RDS MySQL Instance
- IAM Role
- IAM Policy granting full permissions to the role for the provisioned resources; S3 bucket and RDS
- SSM parameters stored on the path `/services/<service name>/` for S3 bucket, RDS credentials and URL

Acceptance Criteria

- Compatible with latest Terraform v1.2+
- Uses variables where appropriate for commonly updated values
- Terraform module created that provisions the resources with defaults that can be overriden
- Terraform template created that uses the module to provision resources for a new service `customer-feedback`
    - S3 Bucket,
    - RDS instance
    - IAM role
    - IAM policies for role
    - SSM Parameter keys with appropriate values
        - S3 Bucket Name
        - RDS Endpoint URL
        - RDS Root User Name
        - RDS Root User Password
        - IAM Role Name
- Terraform template includes outputs
    - RDS URL
    - IAM Role Name
    - S3 Bucket Name