Kubernetes Hackathon

kubernetes

Overview

- The RV store is a mock ecommerce application
- Your task is to get the application running on a Kubernetes cluster
- There are six services plus a MongoDB, each with their own Docker image:
  - Angular UI running in Nginx
  - Authentication service
  - Product service
  - Order service
  - Order simulator service
  - API Gateway service
- Solutions are provided in the repo. But try to only use them to get unstuck on a specific problem!

Objectives

- We have an application made up of 6 services. But I need your expertise to get it running on Kubernetes. All I know is the application code and environment variables needed
- Your goals are:
  - Set up the application to run in Kubernetes
  - Centralize configurations (environment variables)
  - Put any sensitive information into secrets
  - Ensure that only public services are accessible outside the cluster. These are the gateway service and the UI
  - Try adding scaling to one of your deployments, like the product API

Hints

- It is best to start out as simple as possible. Eliminate any variables that might muddy up what you're doing
- Pick a service that is the simplest and start there. Implement it, get it running, then move on
- Don't try to write all the files at once then wonder why things aren't working. Build from simple to complex in an iterative process
- The product API is a good place to start since it just serves static information and has no dependencies on other services

UI Service

- This is an Angular application running nginx to serve the files
- The application serves at port 80
- This application should be publicly accessible
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-ui
- No environment variables needed
- The UI gets its data by making HTTP calls to the backend gateway API
  - <backend>/products to get product information
  - <backend>/orders to get order information
- In the UI itself, there is a text box to enter the base URL of the backend gateway service. Note that it must include the trailing slash

Product Service

- This is a Golang application. It serves up the product information as a REST API
- The application serves at port 9001
- The application should only be accessible inside the cluster
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-product-api
- You must provide an environment variable specifying the internal directory location of the product data: PRODUCT_FILE_LOCATION. I suggest /data/products
- You must provide the products.json file to the container in a ConfigMap. Place it inside the container at the same location as the PRODUCT_FILE_LOCATION you gave above
- The products.json file can be found in the exercise files in the rvstore_hackathon directory
- You can test the service at http://<service>/products

Authentication Service

- This is a Golang application. It serves up a JWT Token in response to a login attempt. It does not take a username/password, but instead gives back a JWT any time the login endpoint is called
- The application serves at port 9003
- The application should only be accessible inside the cluster
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-auth-api
- You can test the service at http://<service>/auth/login

Order Service

- This is a Java Spring Boot application. It receives order data and stores it in the Mongo database
- The application serves at port 9002
- The application should only be accessible inside the cluster
- It communicates with the MongoDB service by name rvstore-orders-mongodb
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-order-api
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose
- You can test the service at http://<service>/orders

Order Simulator Service

- This is a Java Spring Boot application. It generates random orders and submits them to the order API periodically.
- There is no port number for this app. It is not a web app but instead just a background process
- It communicates with the Gateway service at: http://rvstore-api-gateway:9000
- Only one copy of the application should run.
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-order-simulator
- Environment variables needed:
    - SPRING_PROFILES_ACTIVE: compose

API Gateway Service

- This is a Java Spring Boot application. It routes traffic to the appropriate application based on the path. It acts as traffic cop. For example, xyz.com/products will get routed to the product API application
- Runs on port 9000
- Application should be publicly accessible as the only endpoint for the backend API
- It communicates with other services:
  - Auth service at: http://rvstore-auth-api:9003/auth
  - Product service at: http://rvstore-product-api:9001/products
  - Order service at: http://rvstore-order-api:9002/orders
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-gateway-service
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose

Database Service

- For this we’re using the public mongo image in Docker Hub
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-mongo
- Runs on port 27017
- Should be accessible only within the cluster
- MongoDB stores data internally at /data/db
- Environment variables needed:
  - MONGO_INITDB_ROOT_USERNAME: mongoadmin
  - MONGO_INITDB_ROOT_PASSWORD: secret