DOCKER AND KUBERNETES

VITALS CHECK

- How are we feeling about the class so far?
- Lost on anything?
- Questions from yesterday?
- Are you happy with the teaching style so far?
- What adjustments can I make to help?
- Is the pace OK?

RV STORE KUBERNETES HACKATHON

RV STORE

HACKATHON - OVERVIEW

- The RV store is a mock ecommerce application.
- Your task is to get the application running on a Kubernetes cluster.
- There are five services, each with their own Docker image:
  - Angular UI running in Nginx
  - Product service
  - Order service
  - Order simulator
  - Gateway edge service
- Solutions are provided in the Github repo. But try to only use them to get unstuck on a specific problem!
- Github repo is at https://www.github.com/VergeOps/k8s-rvstore

HACKATHON - OBJECTIVES

- Your humble instructor is playing the role of developer. I've written an application made up on the services. But I need your Docker expertise to get it running on Docker. All I know is the application code and environment variables needed.
- Your goals are:

1. Create a Docker image for each service
2. Set up the application in Docker Compose so that it can be run locally
3. Set up the application to run in Kubernetes. For this hackathon, Minikube is fine.

RV STORE – UI APPLICATION

- 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: vergeops/k8s-rvstore-ui
- No environment variables needed
- Bonus points for building the writing a Dockerfile for the application yourself before just using the image in Docker Hub. The application artifacts are in dist/ui

RV STORE – PRODUCT API APPLICATION

- This is a Java Spring Boot application. It serves up the product information as a REST API.
- The application serves at port 9001
- Service name rvstore-product-api
- The application should only be accessible inside the cluster
- Docker image: vergeops/k8s-rvstore-product-api
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose
  - Bonus points for using a configmap
- Bonus points for building the writing a Dockerfile for the application yourself before just using the image in Docker Hub. The application artifact is in target.

RV STORE – ORDER API APPLICATION

- This is a Java Spring Boot application. It receives order data and stores it in the Mongo database
- The application serves at port 9002
- Service name rvstore-order-api
- The application should only be accessible inside the cluster
- Docker image: vergeops/k8s-rvstore-order-api
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose
  - Bonus points for using a configmap
- Bonus points for building the writing a Dockerfile for the application yourself before just using the image in Docker Hub. The application artifact is in target.

RV STORE – ORDER SIMULATOR APPLICATION

- 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.
- Only one copy of the application should run.
- Docker image: vergeops/k8s-rvstore-order-api
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose
  - Bonus points for using a configmap
- Bonus points for building the writing a Dockerfile for the application yourself before just using the image in Docker Hub. The application artifact is in target.

RV STORE – API GATEWAY APPLICATION

- 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 inside the pods
- Runs on port 30090 to the outside
- Service name rvstore-api-gateway
- Application should be publicly accessible as the only endpoint for the backend API
- Docker image: vergeops/k8s-rvstore-api-gateway
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose
  - Bonus points for using a configmap
- Bonus points for building the writing a Dockerfile for the application yourself before just using the image in Docker Hub. The application artifact is in target.

RV STORE – MONGODB DATABASE

- For this we're using the public mongo image in Docker Hub.
- Runs on port 9000
- Docker image: mongo
- Runs on port 27017
- Should be accessible only within the cluster
- If using AWS, data should be stored on a EBS volume to survive pod or node failure. You'll need a StorageClass, PersistentVolumeClaim, and PersistentVolume. Mongo stores data at /data/db
- Environment variables needed:
  - MONGO_INITDB_ROOT_USERNAME: mongoadmin
  - MONGO_INITDB_ROOT_PASSWORD: secret
  - Bonus points for using a configmap