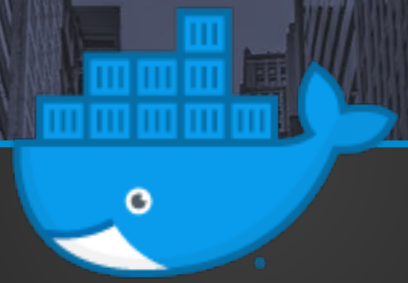
DOCKER

RV STORE



HACKATHON - OVERVIEW

- The RV store is a mock ecommerce application.
- Your task is to get the application running in Docker compose.
- There are six services plus a Mongo DB, each with their own Docker image:
  - Angular UI running in Nginx
  - Authentication service
  - 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 of 6 services. But I need your expertise to get it running in Docker. All I know is the application code and environment variables needed.

Your goals are (in order of importance):

1. Get the application fully working in Docker Compose.
2. For MongoDB, set up a volume mapping to your hard drive so that the MongoDB container can be thrown out and not lose orders.

HACKATHON – LEARNING THROUGH THE PAIN

Exercises so far have been very simple and superficial. This is by design, as I want you to get the deep dive knowledge from this hackathon.

This hackathon is designed to push you. It is intended to make you a little uncomfortable. You may not enjoy it (at least until the end when you have it working)!

The struggle is where the learning is. You will scratch your head, wonder what's going on. This is designed to mimic real life so that you can troubleshoot, then come to me (the developer) to get the proper information.

Past classes have overwhelmingly told me that this is the best part of the class because students come away with a solid foundation of Docker and have confidence that they can go implement a real application.

HACKATHON – HELPFUL 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 just write all the files at once then wonder why things aren't working. Build from simple to complex in an iterative process. The UI service is a good place to start since it just serves static information and has no dependencies on other services.

RV STORE – UI APPLICATION

- This is an Angular application running nginx to serve the files
- The application serves at port 80
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-ui
- No environment variables needed
- The UI gets it's data by making HTTP calls to the backend gateway API.
  - <backend>/products to get product information
  - <backend>/orders to get order information
  - <backend>/auth to get auth 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.

RV STORE – PRODUCT API APPLICATION

- 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
- Environment variable needed:
  - ENVIRONMENT: “containerized”
- You can test the service at http://<service>/products

RV STORE – AUTHENTICATION API APPLICATION

- This is a Golang application. It serves up a JSON Web Token (JWT) 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

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
- The application should only be accessible inside the cluster
- It communicates with the Mongo 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

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. It is not a web app but instead just a background process.
- It communicates with the Gateway service at: http://rvstore-api-gateway:9000
- Docker image: public.ecr.aws/e7e6w2e3/rvstore-order-simulator
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose

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
- 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

RV STORE – MONGODB DATABASE

- 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
- Mongo stores data internally at /data/db
- Environment variables needed:
  - MONGO_INITDB_ROOT_USERNAME: mongoadmin
  - MONGO_INITDB_ROOT_PASSWORD: secret