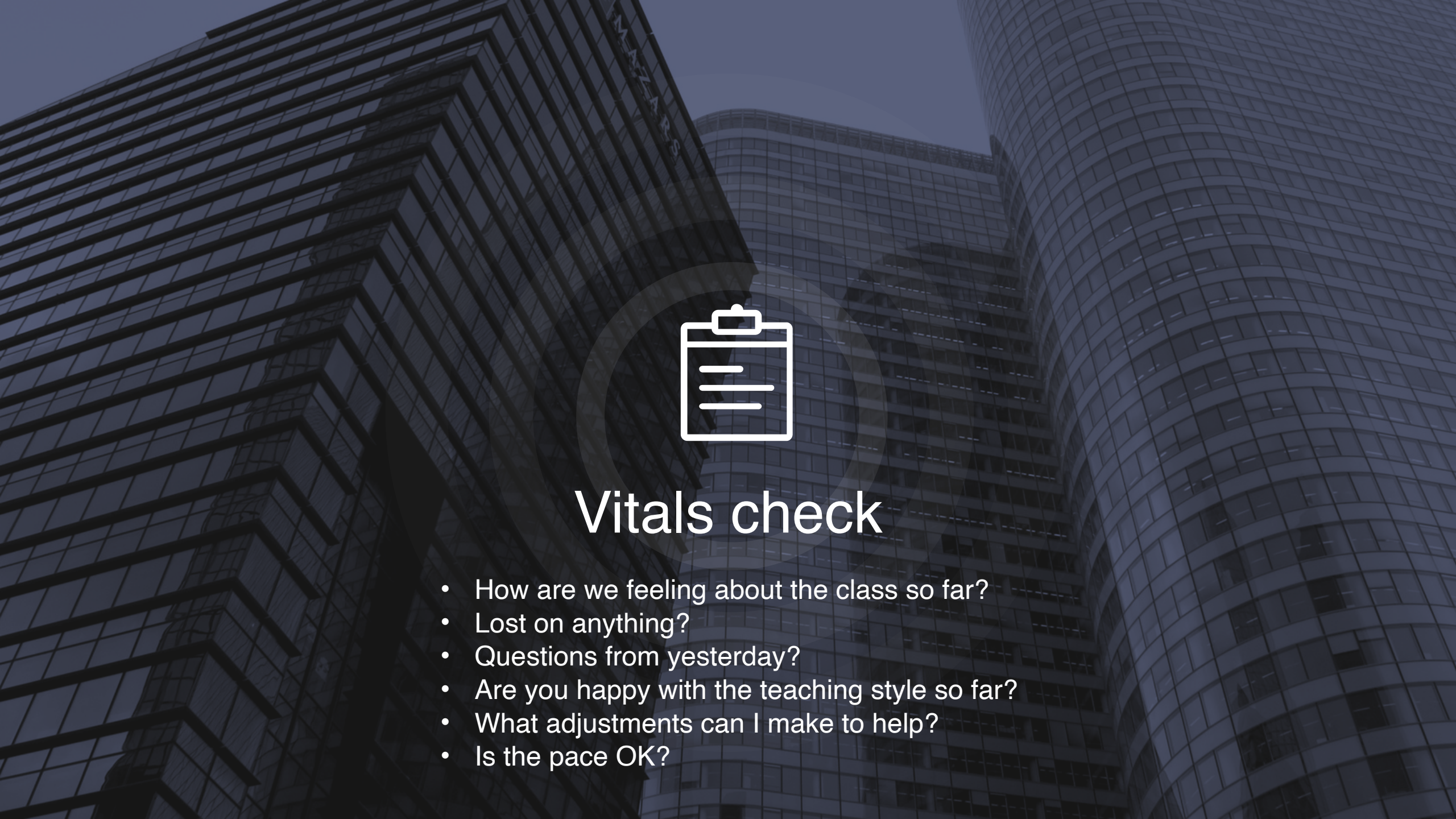
# docker and kubernetes

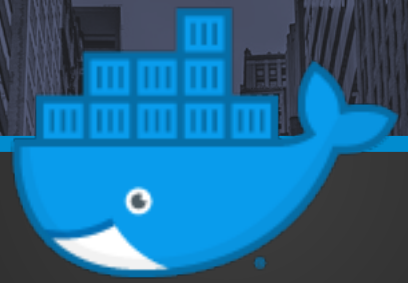


# Vitals check

- 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 plus a Mongo DB, 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://github.com/jwkidd3/k8s_fundamentals

# 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 on Kubernetes. All I know is the application code and environment variables needed.

Your goals are (in order of importance):

1. Set up the application to run in Kubernetes. For this hackathon, Minikube or Docker Kubernetes for Desktop is fine.
2. Centralize configurations (environment variables)
3. Put any sensitive information into secrets
4. Ensure that only public services are accessible outside the cluster. These are the gateway service and the UI.
5. Make the app fault-tolerant
    1. Make services redundant
    2. Set up probes
    3. Try to break it!
6. For MongoDB, set up a volume mapping to your hard drive so that the MongoDB pod can be thrown out and not lose orders.
7. Once everything is running, release version 2.0 of the UI. Once verified, you've realized that there's a problem (the styling is hideous). Try rolling back.
8. If we covered HorizontalPodAutoscaler in this class, try adding scaling to one of your deployments, like the product API.

# 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. I have deliberately left out some important information. In some cases I have given bad information. 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 Kubernetes 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 product API is a good place to start since it just serves static information and has no dependencies.

Save things like fault-tolerance for later. Don't use multiple copies of a service yet. Don't add probes. Save that for once it's working.

# 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
- 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
- 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: vergeops/k8s-rvstore-product-api

# 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: vergeops/k8s-rvstore-order-api
- Environment variables needed:
  - SPRING_PROFILES_ACTIVE: compose

# 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
- Only one copy of the application should run.
- Docker image: vergeops/k8s-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 the product service at: http://rvstore-product-api:9001/products and the order service at http://rvstore-order-api:9002/orders
- Docker image: vergeops/k8s-rvstore-api-gateway
- 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: mongo:latest
- 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