Complex Systems and DevOps: Deliverable 2

Date: November 25, 2024

Course Details

Course Name: Complex Systems and DevOps **Course Code:** 62582 **Semester:** Fall 2024

Team Members

Name	Student Number
Christoffer Fink	*s205449*
Kasper Falch Skov	*s205429*
Johan Søgaard Jørgensen	*s224324*
Henrik Lynggaard Skindhøj	*s205464*
Kevin Wang Højgaard	*s195166*
Sebastian Halfdan Lauridsen	*s215769*

Link to GitHub Project

Table of Contents

4. Backend Development
    - Quarkus Framework Implementation
    - REST API Design with Siren Hypermedia
    - Database Integration
    - Business Logic Implementation
    - Security Implementation
        - JWT Authentication
        - Backend Security Measures
    - Testing Strategy
        - JUnit Implementation
        - REST-assured Testing
    - OpenAPI Documentation
5. Frontend Development
    - React Application Structure
    - TypeScript Integration
    - Vite Build Tool Implementation
    - State Management
    - Component Architecture
    - Security Features
        - Token Security Implementation
    - Package Management
6. DevOps Implementation
    - Version Control Practices
        - Git Workflow
        - GitHub Integration
    - Continuous Integration/Continuous Deployment
        - GitHub Actions Configuration
        - Build Server Setup
        - Testing Pipeline
    - Containerization
        - Docker Implementation
        - Container Registry
    - Cloud Deployment
        - Google Cloud Setup
        - Netlify Frontend Deployment
    - Monitoring and Maintenance
7. Conclusion
    - Project Outcomes
    - Future Improvements
    - Lessons Learned

---

1. Introduction

Project Scope and Objectives

REST-assured Testing

OpenAPI Documentation

5. Frontend Development

React Application Structure

TypeScript Integration

Vite Build Tool Implementation

State Management

Component Architecture

Security Features

Token Security Implementation

Package Management

6. DevOps Implementation

Version Control Practices

Git Workflow

GitHub Integration

Continuous Integration/Continuous Deployment

GitHub Actions Configuration

Build Server Setup

Testing Pipeline

Containerization Containerization is a lightweight form of virtualization that packages an application and its dependencies into a single, portable unit called a container. This approach allows our applications to run consistently across various environments, from development to production, without being affected by differences in underlying infrastructure.

Docker Implementation Docker is used to host and build the final application that Quarkus builds, both for local development and cloud deployment. This approach ensures that the production environment closely resembles the development environment, minimizing discrepancies and potential issues.

By using Docker, we can create a consistent and isolated environment for our application, which simplifies the deployment process and enhances reliability across different stages of the softwares lifecycle.

Container Registry For hosting the Docker container in the cloud, we utilize Google Cloud's Artifact Registry. This service provides a centralized location for storing and managing build artifacts and dependencies, which is crucial for maintaining a streamlined and integrated development workflow.

Given that our final deployment is hosted on Google Cloud Run, using Artifact Registry aligns perfectly with our infrastructure, ensuring seamless integration and efficient management of our container images.

Cloud Deployment Cloud deployment involves hosting applications and services on cloud infrastructure, offering scalable, flexible, and efficient resource management. By leveraging cloud platforms, we can deploy our applications across global data centers, ensuring high availability and performance. This approach reduces the need for on-premises hardware, enabling us to focus on development and innovation rather than managing infrastructures.

By deploying applications closer to users through global data centers, we can reduce latency and enhance user experience. This global reach, combined with the reliability and security of cloud services, makes cloud deployment an ideal choice for us when looking to expand our reach in the market.

Google Cloud Backend Deployment As mentioned in the Container Registry chapter, the deployment of the backend is hosted on the Google Cloud Platform (GCP). Why did we chose Google Cloud over Microsoft Azure, especially since DTU provides students with an Azure subscription. Initially, we attempted to set up our cloud infrastructure on Azure but quickly encountered authentication issues when integrating our GitHub Actions workflows.

A runner/account in Azure's IAM is required, but creating one requires access to Microsoft Entra ID, formerly known as Azure Active Directory, which students don't have. Instead of dealing with this complexity and risking losing access after further development, we decided to use Google Cloud, which offers us full control over the entire cloud environment. And it is free to use

In our cloud setup, we have the following elements which are distributed over GitHub(GH) and the Google Cloud Platform(GCP): - GCP: **SQL Instances** to host our quarkus PostgreSQL database instance. - GCP: **Artifact Registry** to host our Docker container. - GCP: **Cloud Run** to deploy our container and expose it to the web. - GCP: **Identity and Access Management (IAM)** to

create principals and service accounts with the proper access levels, ensuring we adhere to best practices for cybersecurity. - GH: **Security - Secrets and variables**: To ensure that no API keys or credentials are leaked, we use Repository Secrets. We have decided to store our secrets in GitHub to seperate our configuration from our deployment.

In summary, choosing Google Cloud Platform for our backend deployment has provided us with greater control and integration capabilities, overcoming the limitations we faced with Azure. This setup ensures a secure and scalable environment, supporting our development and deployment needs effectively. The pay-as-you-go model of GCP services eliminates the need for significant upfront investments, making it a cost-effective solution. Many of the services we utilize are free of charge under the chosen plans, allowing us to optimize our budget while leveraging advanced cloud capabilities.

Netlify Frontend Deployment

Monitoring and Maintenance

7. Conclusion

Project Outcomes

Future Improvements

Lessons Learned