

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 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 software's 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, allowing for scalable, flexible, and efficient resource management. By leveraging cloud platforms, organizations can deploy applications across global data centers, ensuring high availability and performance.

This approach reduces the need for on-premises hardware, enabling teams to focus on development and innovation rather than infrastructure management.

Google Cloud Setup 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.

In our cloud setup, we have the following elements: - An **SQL instance** to host our SQL database. - **Container/Artifact Registry** to host our Docker images. - **Cloud Run** to deploy our images and expose them to the web. - **Identity and Access Management (IAM)** to create principals and service accounts with the proper access levels, ensuring we adhere to best practices for cybersecurity. - **Secret Manager**: To ensure that no API keys or credentials are leaked, we use Secret Manager. Since our deployment is through GitHub, we have decided to store our secrets as repository secrets. This approach keeps them as close as possible to where they are needed, enhancing security and accessibility.

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.

Netlify Frontend Deployment

Monitoring and Maintenance