Assignment 1 report

Abstract

This report describes the development and deployment of a cloud-based phonebook application using Kubernetes and microservices architecture. The application allows users to store and retrieve phonebook entries, with each entry containing a name, phone number, and address.

Introduction

This report covers the development and deployment of a cloud-based phonebook application. The application is designed to be scalable and deployable using Kubernetes, with a microservices architecture that separates the application logic from the database storage.

Methodology

The application was developed using Flask for the web application and MySQL for the database. Kubernetes was used for deployment, with separate deployments for the application and database. The application's source code and Kubernetes configuration files are available in the GitHub repository:
https://github.com/ouaburst/Cloud-Computing-och-Big-Data/tree/main/Assigment_1

Results or Findings

The application was successfully deployed using Kubernetes and is accessible from the outside world through a LoadBalancer service. The application and database are scalable independently of each other, with persistent storage used for the database to ensure data is not lost across restarts.

Discussion

The microservices architecture allows for independent scaling of the application and database, with persistent storage ensuring data is not lost across restarts. However, managing communication between microservices can be complex, and ensuring data consistency and handling failures can be challenging. Security measures such as authentication, authorization, and encryption should be implemented to protect user data.

Summary of Repository Analysis

The repository fulfills the following requirements:

1. **Deployable using Kubernetes**: The application is deployable using Kubernetes, as indicated by the presence of Kubernetes configuration files in the `kubernetes` folder.

2. **At least two different types of microservices**: The application consists of at least two different types of microservices: an application service (`phonebook-app`) and a database service (`phonebook-mysql`).

3. **REST API for access**: Each microservice implements a REST API for access, as seen in the `server.py` file, which contains the implementation of REST API endpoints for managing phonebook entries.

4. **Accessible from the outside**: The application is accessible from the outside via a web browser, as it runs a web server on port 80.

5. **Horizontally scalable microservices**: All microservices in the application are horizontally scalable independently from each other, as indicated by the `replicas` field in the `app-deployment.yaml` file.

6. **Microservice images on Docker Hub**: Any microservice images required to run the application have been pushed to Docker Hub, as indicated by the image field in the `app-deployment.yaml` file.

7. **Database as a separate microservice**: The application makes use of a MySQL database running as a separate microservice, as indicated by the presence of `mysql` in the service names.

8. **Persistent storage for database**: The database uses storage that is persistent across restarts of the deployment infrastructure, as indicated by the `mysql-pv.yaml` and `mysql-pvc.yaml` files.

9. **Database scalability not required**: The database does not have to be scalable, as per the requirement.

In conclusion, the repository fulfills all the listed requirements.

Conclusion

The cloud-based phonebook application successfully meets the requirements for scalability, deployability using Kubernetes, and separation of application logic and database storage. The application is accessible from the outside via a web browser, and all microservices in the application are horizontally scalable independently from each other. The REST API has been implemented for programmatic access to phonebook entries, allowing for greater flexibility and integration possibilities. The application makes use of a database running as a separate microservice, with persistent storage across restarts of the deployment infrastructure. Overall, the application is well-designed and fulfills the specified requirements, providing a solid foundation for further development and enhancements.

Recommendations

1. Implement security measures such as authentication, authorization, and encryption to protect the application and its data.
2. Consider adding additional features or functionalities to the phonebook application to enhance its usefulness and user experience.
3. Regularly update the application and its dependencies to address any security vulnerabilities and keep the software up-to-date.