Yonatan Greenshpan – 204266191
Tomer Shani - 204388920
Harel Damari - 305792020

Executive Summary: K8S final project

Overview of K8S Cluster Configuration and Design Rationale

In Project bdp_3605, the Kubernetes cluster was configured as a test environment to simulate the management of containerized microservices. While the project did not utilize real-time data (due to API and fee issues), it provided to us a valuable sandbox for understanding Kubernetes and Docker capabilities and limitations. Our projects have three main components:

- **Data Microservice:** Toy data in CSV format was used within a dedicated data folder, mimicking the storage of real datasets.
- **Scripts:** Python scripts for data ingestion and analysis were containerized using Docker, which was instrumental in creating a consistent development environment.
- **Kubernetes Configurations:** two yaml files in k8s folder that help us learn about pod configurations and resource management within the cluster.

Key Insights Derived from Kubernetes Enhancement

The project's use of Kubernetes, even with toy data, highlighted the platform's robustness in handling container orchestration and its potential for real-world applications. We got some key insights during the process:

- **High Scalability**: k8s provides scalability through its ability to automatically adjust the number of running pods based on the demand. In our project it was provided by the `Deployment.yaml` file.
- **High Availability**: Kubernetes ensures high availability through its self-healing feature, which automatically replaces failed pods to maintain the desired state. In our project, this should happen through `Service.yaml` that act as internal load balancer.
- **Secret and Config Management**: Kubernetes provides efficient secret and configuration management with its Secrets and ConfigMaps. These allow us to store and manage sensitive information and configuration options separately from the pod definitions. However, we didn't intend to use this functionality in the project.

Yonatan Greenshpan – 204266191
Tomer Shani - 204388920
Harel Damari - 305792020

Challenges Encountered & Solutions Implemented

A significant portion of the project was dedicated to debugging Kubernetes configurations, which provided deeper insights into the intricacies of cluster management.

- **Configuration Debugging:** The majority of time was spent fine-tuning YAML files and resolving deployment issues, which was an educational experience in itself.
- **Design Considerations:** our main dilemma was rather to implement monolithic or microservices architecture. While in the monolithic we have one `Service.yaml` that expose our application externally and one `Deployment.yaml`, in microservices we should have one for each functionality (Data ingestion, data processing and data analysis). Due to the fact that we used toy data with excel and not real time streaming to Cassandra DB, we choose the monolithic approach in order to not overengineer the app. While this approach is easier to debug and has simpler initial setup and deployment, it has limited scalability and flexibility.

Illustrations of the benefits of K8S

It is important to note that our project utilized Excel-based toy data and did not undergo an actual migration to K8s. The anticipated improvements and graphical representations discussed are speculative projections intended to demonstrate the potential benefits of adopting K8s in a real-world application. The following two graphs are proposed to illustrate these potential improvements:

1. **Deployment Frequency vs. Downtime**: A line graph is suggested to visualize the expected increase in deployment frequency juxtaposed with a decrease in system downtime. This graph aims to highlight the efficiency of Kubernetes' automated processes, which are designed to facilitate continuous deployment without compromising system availability.
2. **Resource Utilization**: A bar graph is recommended to demonstrate the projected optimization of CPU and memory usage. This graph would underscore Kubernetes capabilities for resource management, ensuring that system resources are utilized efficiently, even under varying loads.