2024 CSUSM Software Engineering Capstone Design Conference

Serverless Orchestrator

Introduction

Viasat, a satellite communications industry leader for over 30 years, has launched an innovative project to improve application development and deployment procedures with the Serverless Orchestrator. This project seeks to simplify the event-driven orchestration and automation of serverless microservices by utilizing Kubernetes infrastructure. The Serverless Orchestrator is designed to streamline the application development process and improve scalability by allowing developers to focus on microservice creation without having to manage underlying infrastructure. This initiative is Viasat's strategic effort to stimulate agility in application development and operational efficiency, demonstrating the company's commitment to cloud-native, serverless technology. With the help and knowledge of industry mentor Gopikrishnan Selvarajan, and under the supervision of Simon Fan as both course instructor and faculty advisor, this capstone project, executed by a team of Software Engineering students at California State University, San Marcos, aims to deliver a serverless orchestrator platform that is consistent with Viasat's vision for innovative, efficient, and scalable application deployment.

Architecture

Figure 1 shows the frontend and backend architecture of Serverless Orchestrator. The frontend is found in the first layer of the diagram, the presentation layer, this is where React is used to create the website that is responsible for interacting with Developers and Consumers of Serverless Orchestrator. The JSON requests from the presentation layer are translated to executable tasks on the backend via the systems Orchestrator, after being routed by the WebAPI, in the application layer. User account actions like registration, login, browsing and uploading microservices are examples of tasks handled by the Orchestrator. Connections to Dockerhub and Github API's are also made in the application layer , to perform crucial tasks like uploading, accessing, and image creation of Developer code. The User and Microservice information is located in the Business Layer and is connected to our PostgreSQL database using a DAO_IF that links to the Data Access Layer, which houses the databases for the User and Microservice connections.

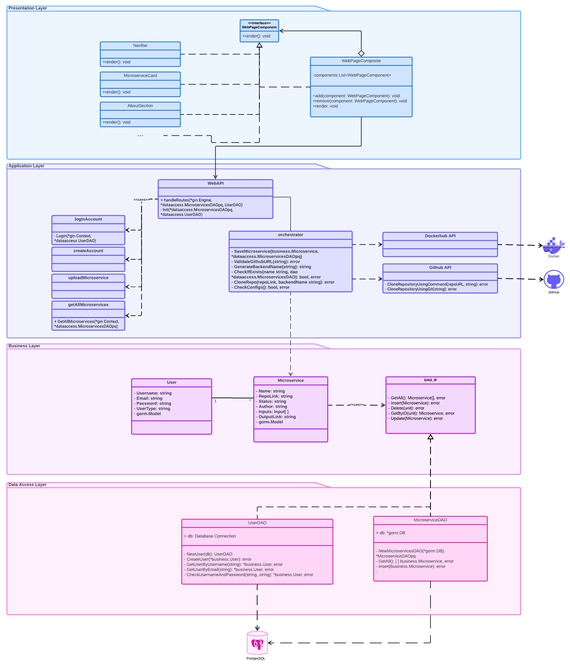


Figure 1: Architectural Design

Authors & Abstract

Student Team: Carter Rath, Ruth Jimenez, Mae Pereyra, Jaclyn Walsh

Faculty Adviser: Dr. Xiaocong (Simon) Fan

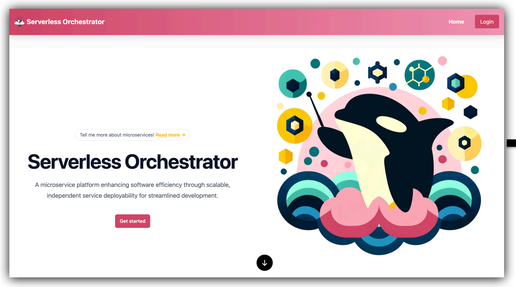
Industry Mentors: Gopikrishnan Selvarajan

Sponsored By: Viasat

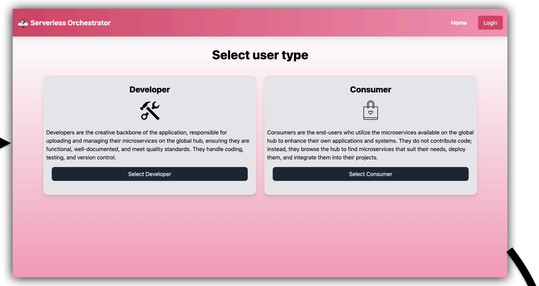
Abstract

The Serverless Orchestrator project takes a novel approach to application development and deployment, employing a serverless architecture within a Kubernetes environment. It seeks to make development easier by automating the execution and maintenance of serverless microservices in an event-driven way. The Serverless Orchestrator uses a Function as a Service (FaaS) paradigm, allowing developers to focus on designing microservices rather than the difficulties of infrastructure management. This move not only speeds up product releases and improves the developer experience, but it also greatly boosts scalability and optimizes resource utilization.

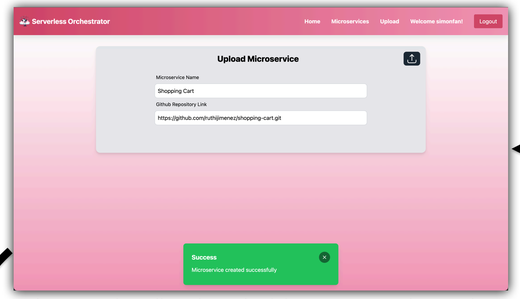
Results



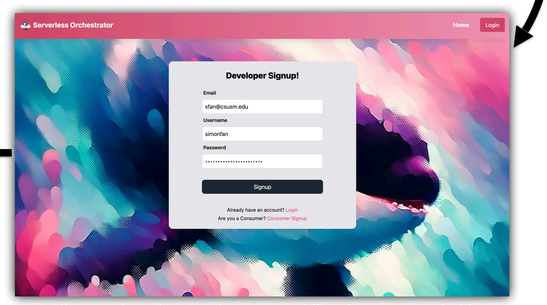
Our application begins here (**Figure 2: Home Page**), where users can learn about the application's objective, terminology, and the basic structure of a microservice.



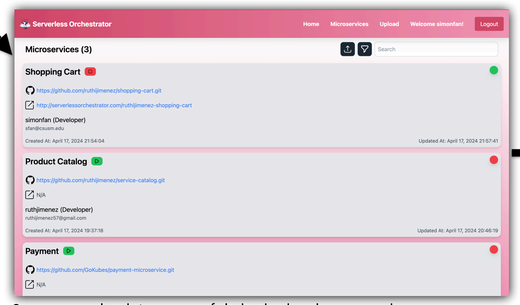
After clicking "Get Started", the user can decide if they want to be a developer who uploads microservices or a consumer who uses the uploaded microservices (**Figure 3: Get Started**).



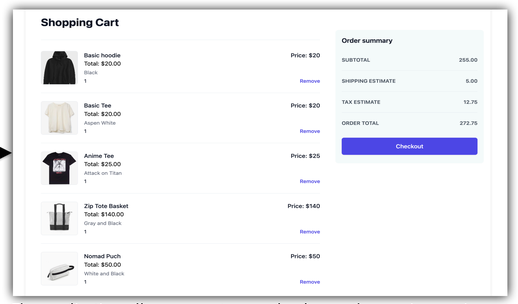
Developers are then allowed to upload their microservice by providing a name and link to its GitHub repository (**Figure 5: Upload Microservice**).



Following the selection, the user can fill in the necessary fields to create their own account and log in (**Figure 4: Signup**).



Once an upload is successful, both developers and consumers can browse the uploaded microservices (**Figure 6: Browse Microservices**).



The application allows users to seamlessly run these microservices at any time and view the results (**Figure 7: Microservice in Action!**).

Core Functionality

The core functionality of the system is the automated orchestration of Dockerized microservice lifecycle, from source code retrieval to image deployment. Upon initiation through the web interface, it validates user credentials, manages code repositories, automates Docker image builds with Kubernetes, and facilitating seamless microservice integration and deployment.

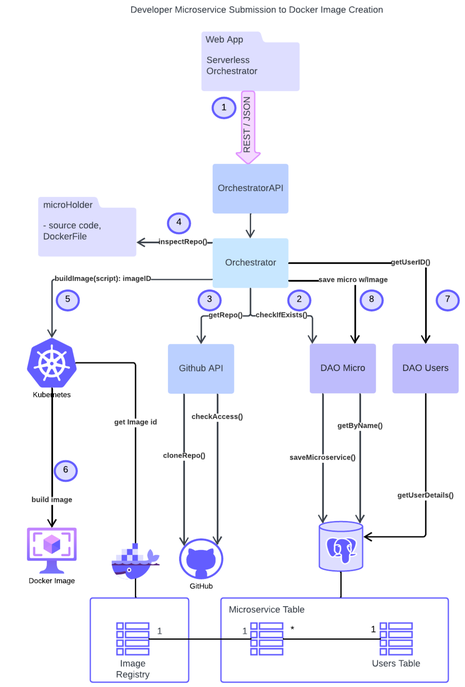


Figure 6: Context Diagram

Third-Party Software

tailwindcss · React · GO · aws · GitHub · TS · GORM · docker

References

- Amazon Web Services: https://docs.aws.amazon.com/
- Docker: https://docs.docker.com/
- GitHub APIs: https://docs.github.com/en
- Go: https://go.dev/doc/
- PostgreSQL: https://www.postgresql.org/docs/16/index.html
- React: https://react.dev/
- TailwindCSS: https://tailwindui.com/documentation
- Viasat, Inc: https://www.viasat.com/about/
- Vite: https://v4.vitejs.dev/guide/