images/thi_logo.pdf

Technische Hochschule Ingolstadt

Specialist area Computer Science
Bachelor's course Computer Science

Bachelor's thesis

Subject: Conception, Implementation, and Evaluation of a Highly Scalable and Highly Available Kubernetes-Based SaaS Platform on Kubernetes Control Plane (KCP)

Name and Surname: David Linhardt

Issued on: TODO: Insert Issue Date

Submitted on: TODO: Insert Submit Date

First examiner: Prof. Dr. Bernd Hafenrichter

Second examiner: Prof. Dr. Ludwig Lausser

Abstract

Contents

Glossary

1 Introduction

1.1 Problem Statement and Motivation

1.2 Objectives and Scope

1.3 Structure of the Thesis

2 Fundamentals

2.1 Kubernetes and Multi-Tenancy

Kubernetes as the Foundation for Cloud-Native Applications As the de facto standard for deploying and managing *cloud-native applications*, Kubernetes plays a pivotal role in modern cloud architecture (Nigel Poulton and Pushkar Joglekar, *The Kubernetes Book*, p. 7–8). Kubernetes works as an application orchestrator for *containerized, cloud-native microservice* apps, meaning it can deploy applications and dynamically respond to changes (Nigel Poulton and Pushkar Joglekar, *The Kubernetes Book*, p. 3). It offers a platform for declarative configuration and automation for containerized workloads, enabling organizations to run distributed applications and services at scale (Kubernetes, *Concepts / Overview*; Red Hat, *What is Kubernetes?*).

The Importance of Multi-Tenancy in Modern SaaS Platforms Multi-tenancy plays a fundamental role in modern cloud computing. By allowing multiple tenants to share the same infrastructure through virtualization, it significantly increases resource utilization, reduces operational costs, and enables essential features such as VM mobility and dynamic resource allocation (Hussain AlJahdali et al., “Multi-tenancy in Cloud Computing”, pp. 345–346). These benefits are critical for cloud providers, as they make the cloud business model economically viable and scalable. In the context of modern SaaS platforms, multi-tenancy goes even further

by enabling unified management, frictionless onboarding, and simplified operational processes that allow providers to add new tenants without introducing incremental complexity or cost (AWS, *AWS Whitepaper - SaaS Architecture Fundamentals*, pp. 9–11).
However, while multi-tenancy is indispensable for achieving efficiency, scalability, and cost-effectiveness, it simultaneously introduces complex security challenges, especially in shared environments where resource isolation is limited. In particular, the potential for cross-tenant access and side-channel attacks makes security in multi-tenant environments a primary concern (Hussain AlJahdali et al., “Multi-tenancy in Cloud Computing”, pp. 345–346). As such, understanding and addressing multi-tenancy from both operational and security perspectives is essential when designing and securing modern cloud-native platforms (AWS, *AWS Whitepaper - SaaS Architecture Fundamentals*, pp. 9–11; *Information technology - Cloud computing - Part 2: Concepts*, p. 4).

The Challenges of Multi-Tenancy and the Need for Solutions

Kubernetes Control Plane (KCP) as a Promising Approach

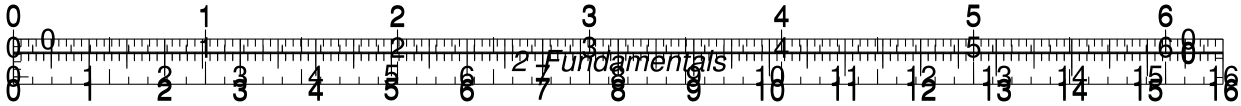
Background: The Evolution of Kubernetes Kubernetes was originally developed at Google and released as open source in 2014 (Google Cloud, *What is Kubernetes?*).

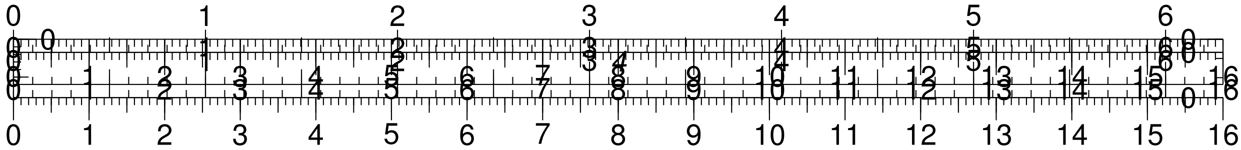
Background: Containerization as an Enabler of Kubernetes *Containerization* is a way to bundle an application’s code with all its dependencies to run on any infrastructure thus enhancing portability (AWS, *What is Containerization?*; Docker, *Use containers to Build, Share and Run your applications*). The lightweight nature and isolation can be leveraged by cloud-native software by enabling vertical and horizontal autoscaling facilitated by quick container boot times, along with self-healing mechanisms and support for distributed, resilient infrastructures (Kubernetes, *Concepts / Workloads / Autoscaling Workloads*; Kubernetes, *Kubernetes Self-Healing*; AWS, *What is Containerization?*; Cornelia Davis, *Cloud Native Patterns - Designing change-tolerant software*, pp. 58–59) Furthermore it complements the microservice architectural pattern by enabling isolated, low overhead deployments, ensuring consistent environments (A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to cloud-native architectures using microservices: an experience report”, p. 209).

Background: The Role of Microservices in Cloud-Native Architectures *Microservices* play a pivotal role in cloud-native architectures by promoting agility, scalability, and maintainability of applications. By decomposing applications into independent, granular services, microservices facilitate development, testing, and deployment using diverse technology stacks, enhancing interoperability across platforms (M. Waseem, P. Liang, and M. Shahin, “A systematic mapping study on microservices architecture in devops”, p. 1; Xabier Larrucea et al., “Microservices”, p. 1) and help prevent failures in one component from propagating across the system, by isolating functionality into distinct, self-contained services (Cornelia Davis, *Cloud Native Patterns - Designing change-tolerant software*, p. 62). This architectural style aligns well with cloud environments, as it allows services to evolve independently, effectively addressing challenges associated with scaling and maintenance without being tied to a singular technological framework (A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to cloud-native architectures using microservices: an experience report”, pp. 202–203). Furthermore, the integration of microservices with platforms like Kubernetes enhances deployment automation and orchestration, thus providing substantial elasticity to accommodate fluctuating workloads (S. Haugeland et al., “Migrating monoliths to microservices-based customizable multi-tenant cloud-native apps”, p. 170). Additionally, migrating legacy applications to microservices can foster modernization and efficiency, thus positioning organizations favorably in competitive landscapes (A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Migrating to cloud-native architectures using microservices: an experience report”, p. 214). Overall, the synergy between microservices and cloud-native architectures stems from their inherent capability to optimize resource utilization and streamline continuous integration and deployment processes.

Background: Kubernetes Resource Isolation Mechanisms

Relevance to SaaS and this Thesis





2.2 Kubernetes Control Plane (KCP)

2.3 SaaS Architecture and Automation

3 State of the Art and Related Work

3.1 Zero-Downtime Deployment Strategies

3.2 Kubernetes Scaling Methods

3.3 Multi-Tenancy Concepts in the Cloud

4 Conceptual Design

4.1 System Requirements

4.2 Architecture Design with KCP for SaaS

4.3 Automated Deployment Strategies

5 Prototypical Implementation

5.1 Infrastructure with KCP

5.2 Tenant Provisioning (Automation, Multi-Tenancy)

5.3 Scaling Mechanisms (Horizontal Pod Autoscaler)

5.4 Monitoring and Logging (Prometheus, Grafana)

6 Evaluation

6.1 Performance Measurements (Downtime, Latency, Scaling)

6.2 Scaling Scenarios & Optimizations

6.3 Discussion of Results

6.4 Related Work

7 Conclusion and Outlook

7.1 Summary

7.2 Personal Conclusion

7.3 Future Outlook

References

Hussain AlJahdali et al. “Multi-tenancy in Cloud Computing”. In: *2014 IEEE 8th International Symposium on Service Oriented System Engineering*. 2014, pp. 344–351. DOI: 10.1109/SOSE.2014.50.

AWS. *AWS Whitepaper - SaaS Architecture Fundamentals*. AWS, 2022. URL: https://docs.aws.amazon.com/whitepapers/latest/saas-architecture-fundamentals/re-defining-multi-tenancy.html (visited on 05/01/2025).

– *What is Containerization?* URL: https://aws.amazon.com/what-is/containerization/ (visited on 05/01/2025).

A. Balalaie, A. Heydarnoori, and P. Jamshidi. "Migrating to cloud-native architectures using microservices: an experience report". In: (2016), pp. 201–215. DOI: 10.1007/978-3-319-33313-7_15.

Cornelia Davis. *Cloud Native Patterns - Designing change-tolerant software*. Shelter Island, NY: Manning, 2019. ISBN: 9781617294297.

Docker. *Use containers to Build, Share and Run your applications*. URL: https://www.docker.com/resources/what-container/ (visited on 05/01/2025).

Google Cloud. *What is Kubernetes?* URL: https://cloud.google.com/learn/what-is-kubernetes (visited on 05/01/2025).

S. Haugeland et al. "Migrating monoliths to microservices-based customizable multi-tenant cloud-native apps". In: (2021), pp. 170–177. DOI: 10.1109/seaa53835.2021.00030.

Information technology - Cloud computing - Part 2: Concepts. Standard. 2023.

Kubernetes. *Concepts / Overview*. 2024. URL: https://kubernetes.io/docs/concepts/overview/ (visited on 05/01/2025).

– *Concepts / Workloads / Autoscaling Workloads*. URL: https://kubernetes.io/docs/concepts/workloads/autoscaling/ (visited on 05/01/2025).

– *Kubernetes Self-Healing*. URL: https://kubernetes.io/docs/concepts/architecture/self-healing/ (visited on 05/01/2025).

Xabier Larrucea et al. "Microservices". In: *IEEE Software* 35.3 (2018), pp. 96–100. DOI: 10.1109/MS.2018.2141030.

Nigel Poulton and Pushkar Joglekar. *The Kubernetes Book*. 2021 Edition. No ISBN provided. Independently published, 2021, p. 243.

Red Hat. *What is Kubernetes?* 2024. URL: https://www.redhat.com/en/topics/containers/what-is-kubernetes (visited on 05/01/2025).

M. Waseem, P. Liang, and M. Shahin. "A systematic mapping study on microservices architecture in devops". In: *Journal of Systems and Software* 170 (2020), p. 110798. DOI: 10.1016/j.jss.2020.110798.

List of Figures

Appendix