Frankfurt University of Applied Sciences

Master's Thesis

Containerized multi-level deployment for a distributed adaptive microservice application

Author:
Tim Wißmann

Supervisors:
Prof. Dr. Peter Thoma
Prof. Dr. Eicke Godehardt

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the course

Allgemeine Informatik Master

March 17th, 2023

Declaration of Authorship

I, Tim Wißmann, declare that this thesis titled, 'Containerized multi-level deployment for a distributed adaptive microservice application' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ________________________________

Date: ________________________________

“Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.”

Dave Barry

FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Abstract

Faculty 2 - Computer Science and Engineering
Allgemeine Informatik Master

Master of Science

Containerized multi-level deployment for a distributed adaptive microservice application

by Tim Wißmann

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere

Chapter 1

Introduction

1.1 Scope

1.2 Intended audience

1.3 Limitations

1.4 Outline

Chapter 2

Background and related work

2.1 Baseline architecture

2.2 Problem statement

2.3 Related Work

Chapter 3

System design

3.1 Target architecture

3.2 Applied technologies

Various applications for realizing the architecture have been compared. In the following sections the different options are that were taken into account are presented.

3.2.1 Cluster management

For managing the cluster several technologies were compared. - Hyper-V Replication controller - docker swarm - Kubernetes - open Shift

3.2.2 Container environment

As container environment `containerd` is used. Containerd is.. In comparison to other container environments this is ...

- docker - containerd -> Only tech with support for Windows - equal tech stack everywhere - LXC container

Chapter 4

Implementation

4.1 Cluster Setup

The following section describes the setup of different machines in the cluster, so called nodes. While the master node refers to the Kubernetes Control-Plane node which is responsible for distribution of the workers, the worker nodes are the actual machines that are executing the applications. During development the cluster was set up on virtual machines completely, due to the lack of physical hardware.

4.1.1 Creating the master node

For setting up the master node on Linux a system based on Debian Bullseye 11.5 has been used. After installing and setting up the operating system, the swap mechanism needs to be permanently turned off. This is done by editing the file system table (fstab) in file `/etc/fstab` respectively by commenting out the swap partitions and masking the systemd swap units.

After installing the pre-requisite packages, a containerd config file needs to be created. For this, the command from Listing 4.1 is applied.

```
sudo sysctl net.bridge.bridge-nf-call-iptables=1
echo 1 > /proc/sys/net/ipv4/ip_forward
sudo containerd config default | sudo tee /etc/containerd/config.toml &>/dev/null
```

Listing 4.1: Bash command for setting up containerd config

Afterwards the systemd control group (cgroup) is added to the runtime options of containerd and the its service is restarted. After setting up the prerequisites, the cluster can be initialized by running the command line tool as shown in Listing 4.2 with the appropriate configuration as parameter.

```
        sudo kubeadm init --config config.yaml
```

Listing 4.2: Bash command for setting up the cluster

Installing a Container Network Interface After successfully running the initialization, the cluster overlay network `flanel` needs to be setup. This is required for working with Windows worker nodes. To setup `flanel` the respective pod description can be directly downloaded and applied on the cluster from the vendor. This is what can was achieved in Listing 4.3.

```
kubectl apply -f \
https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/\
kube-flannel.yml
```

Listing 4.3: Bash command for setting up the network interface

Successful setup of the cluster can be checked with can be locally tested with `kubectl`.

4.1.2 Creating the worker node

Chapter 5

Discussion

5.1 Analysis

5.2 Dunno whata write yet...

Chapter 6

Conclusion and future work

6.1 Future work

- Linux port and cluster based on linux

6.2 Conclusion

Appendix A

Appendix Title Here

Write your Appendix content here.

cgroup control group

Bibliography