

19. MÄRZ 2024



K3S



K3S



K3S

HIGH AVAILABILITY K3S CLUSTER

MICROSERVICES

MARIO CHRIST & ANDRIN GERMANN
O-TIN-21-T-A
TEKO Olten

Inhaltsverzeichnis

1.	Multipass.....	2
1.1	Virtueller Switch.....	2
1.2	VM aufsetzen.....	4
1.3	VM konfiguration	4
1.1.1	1.3.1 VM-Network.....	5
2.	Konfiguration Nodes.....	6
2.1	K3S	6
2.2	Kube VIP.....	7
2.3	Choco.....	9
2.4	Kubectx	10
2.5	K3S-Config Master.....	11
2.6	MetallLB.....	13
2.7	nfs common	15
2.8	Longhorn.....	16
3.	3. Zusammenstellung yaml.....	17
3.1	Docker image	17
3.2	Deployments.....	19
3.3	Services.....	24
3.4	PVC.....	26
3.5	Ingress	27
4.	Monitoring	30
4.1	Helm repo.....	30
4.2	Nginx Ingress.....	31
4.3	Prometheus	32
4.4	Grafana	34
5.	Test	39
5.1	Knotenausfall.....	39
5.2	Masterausfall.....	40
5.3	Pod Ausfall.....	41
5.4	Datenintegrität.....	42

Multipass

In dieser Anleitung wird gezeigt, wie es möglich ist ein hochverfügbares Cluster zu erstellen. In diesem Beispiel wird eine bereits bestehende Applikation in ein Cluster integriert. Es ist ein Schritt für Schritt Anleitung. Am Ende soll es jeder Person möglich sein, mit dieser Anleitung selbst ein hochverfügbares Cluster einzurichten.

Damit ein Cluster mit drei Virtuellen Maschinen entsteht, müssen diese VMs zuerst aufgesetzt werden. Für die Realisation verwenden wir Multipass. Multipass ist eine Software, die es Benutzern erlaubt, einfach und schnell eine virtuelle Ubuntu Maschine aufzusetzen. Die Maschinen können über das CLI mit einem Befehl installiert werden. Die Software kann unter <https://multipass.run/> heruntergeladen werden. Bei der Installation muss man einen Hypervisor angeben. In dieser Anleitung wird «Hyper-V» von Windows benutzt.

1.1 Virtueller Switch

Normalerweise werden die einzelnen VMs mit dem Bridge-Netzwerk verbunden. Es ist jedoch von Vorteil, einen virtuellen Switch in Hyper-V zu erstellen.

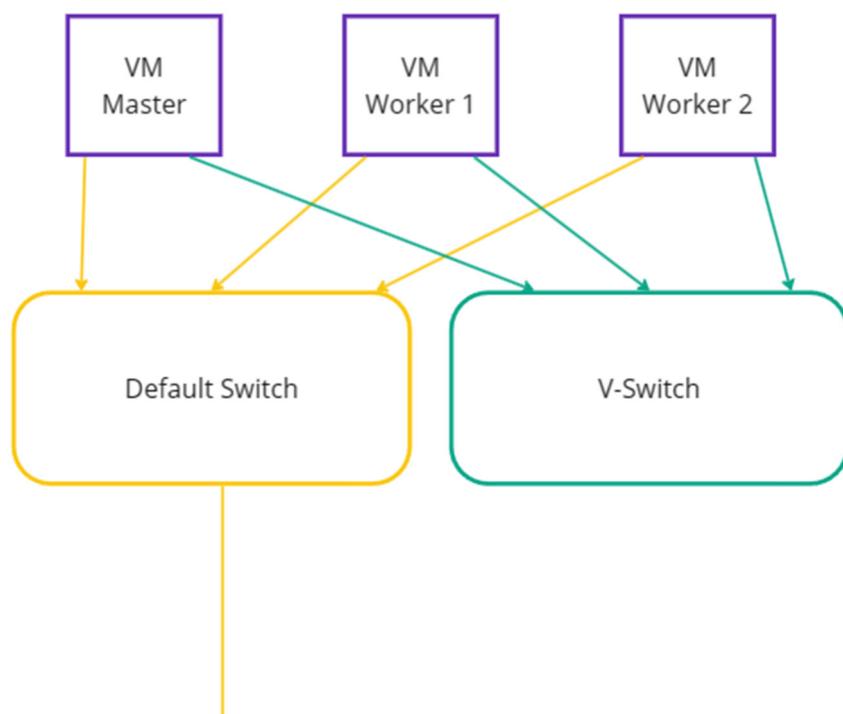


Abbildung 1 Zeichnung V-Switch

Hyper-V öffnen -> Aktionen -> Manager für virtuelle Switches -> Neuen Switch erstellen -> benennen -> internal einstellen -> fertig stellen

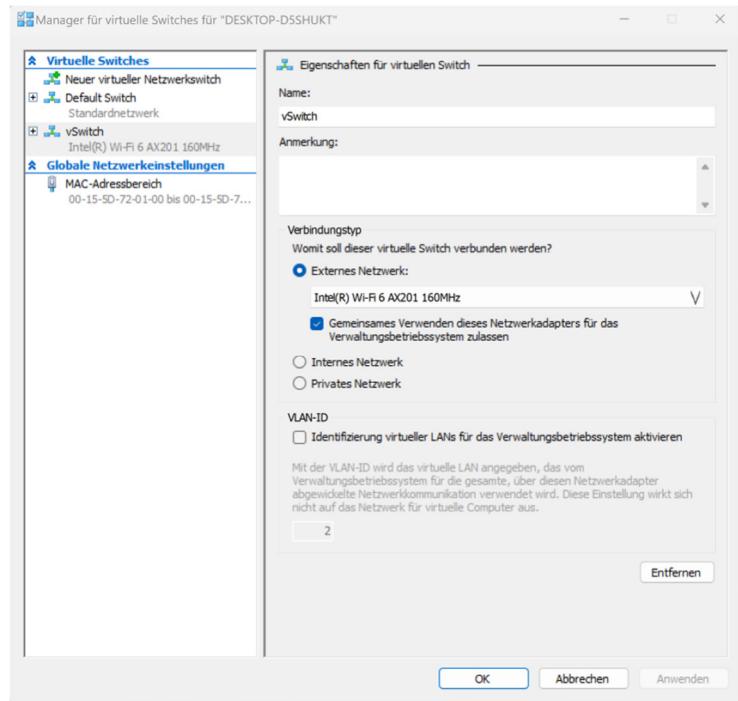


Abbildung 2 Hyper-V virtueller Switch

Mit dem folgenden Befehl kann die IP-Range des Switchs gesetzt werden. (als Admin in der PowerShell). Achtung: IP darf nicht vergeben sein!

New-NetIPAddress -InterfaceAlias 'vEthernet (<switch name>)' -IPAddress 192.168.0.105 -PrefixLength 24

```
PS C:\WINDOWS\system32> New-NetIPAddress -InterfaceAlias 'vEthernet (vSwitch)' -IPAddress 192.168.0.105 -PrefixLength 24

IPAddress      : 192.168.0.105
InterfaceIndex : 93
InterfaceAlias : vEthernet (vSwitch)
AddressFamily   : IPv4
Type           : Unicast
PrefixLength   : 24
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Tentative
ValidLifetime   :
PreferredLifetime :
SkipAsSource    : False
PolicyStore     : ActiveStore

IPAddress      : 192.168.0.105
InterfaceIndex : 93
InterfaceAlias : vEthernet (vSwitch)
AddressFamily   : IPv4
Type           : Unicast
PrefixLength   : 24
PrefixOrigin    : Manual
SuffixOrigin    : Manual
AddressState    : Invalid
ValidLifetime   :
PreferredLifetime :
SkipAsSource    : False
PolicyStore     : PersistentStore
```

Abbildung 3 V-Switch IP Range

1.2 VM aufsetzen

Da der Switch nun bereit ist, können wir die einzelnen VMs erstellen. Für ein Cluster benötigen wir mindestens einen Master Node und zwei Worker Nodes. Die Master Node ist die Steuerungseinheit. Sie koordiniert und steuert das Cluster. Die Worker Nodes enthalten die Anwendung, die im Cluster bereitgestellt werden. Der Befehl «**multipass launch**» startet den ganzen Prozess. Es ist jedoch nötig, dem Multipass gewisse Parameter mitzugeben.

```
PS C:\WINDOWS\system32> multipass launch -n k3s-master -c 2 -m 4G -d 20G --network name=vSwitch,mode=manual
Launched: k3s-master
PS C:\WINDOWS\system32> multipass launch -n k3s-worker-1 -c 2 -m 4G -d 20G --network name=vSwitch,mode=manual
Launched: k3s-worker-1
PS C:\WINDOWS\system32> multipass launch -n k3s-worker-2 -c 2 -m 4G -d 20G --network name=vSwitch,mode=manual
Launched: k3s-worker-2
PS C:\WINDOWS\system32>
```

Abbildung 4 Multipass Nodes launch

-n gibt den Namen, -c gibt die Anzahl virtueller CPUs, -m die Menge des Arbeitsspeichers, -d die Menge des Festplatten Speichers und mit dem --network die Netzwerk Konfiguration. Wir verwenden den Namen des zuvor erstellten Switches.

1.3 VM konfiguration

Für weitere Einstellungen ist es nötig, auf die VMs zuzugreifen. Mit multipass wird einem dieser Zugriff erleichtert. Befehl: **multipass shell «Name der VM»**

```
PS C:\WINDOWS\system32> multipass shell k3s-master
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-92-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

 System information as of Tue Jan 30 21:02:57 CET 2024

 System load:  0.0           Processes:          98
 Usage of /:   7.1% of 19.20GB  Users logged in:     0
 Memory usage: 6%
 Swap usage:   0%
               IPv4 address for eth0: 172.30.189.72

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@k3s-master:~$
```

Abbildung 5 Multipass Shell

1.3.1 VM-Network

Für eine reibungslose Funktion des Clusters sind fixe IPs zwingend nötig. Achte darauf, freie IPs zu verwenden. Alle drei virtuelle Maschinen brauchen eine eigene IP-Adresse. Wiederhole diesen Schritt für jede Instanz.

Master:

```
ubuntu@k3s-master:~$ sudo nano /etc/netplan/99-multipass.yaml
```

Abbildung 7 Sudo nano Netplan

```
GNU nano 6.2
network:
  ethernets:
    eth1:
      dhcp4: false
      match:
        macaddress: 52:54:00:a4:02:75
        set-name: eth1
      addresses: [192.168.10.11/24]
      version: 2
```

Abbildung 6 Netplan Master

```
ubuntu@k3s-master:~$ sudo netplan apply
```

Abbildung 8 Netplan apply

Nachdem erfolgreichen einstellen des Netplans vom Master, folgen die anderen zwei Maschinen. Mit dem Befehl «**Exit**» kann die Shell verlassen werden.

K3s-worker-1 & K3s-Worker 2:

```
GNU nano 6.2
network:
  ethernets:
    eth1:
      dhcp4: false
      match:
        macaddress: 52:54:00:a4:02:75
        set-name: eth1
      addresses: [192.168.10.11/24]
      version: 2
```

Abbildung 9 Netplan Worker

Konfiguration Nodes

Unser Grundkonstrukt ist aufgestellt. Die VMs laufen mit statischen IPs. Im nächsten Schritt werden die nötigen Applikationen für das Cluster konfiguriert.

2.1 K3S

K3s ist eine leichtgewichtige Kubernetes Distribution. Wir installieren diese auf jeder Node.

Installation auf der Master-Node:

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> multipass exec k3s-master -- bash -c "curl -sL https://get.k3s.io | INSTALL_K3S_<br/>EXEC= --node-ip=192.168.10.10 --cluster-init --disable service1" sh -<br/>[INFO] Finding release for channel stable<br/>[INFO] Using v1.28.6+k3s2 as release<br/>[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.28.6+k3s2/sha256sum-amd64.txt<br/>[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.28.6+k3s2/k3s<br/>[INFO] Verifying binary download<br/>[INFO] Installing k3s to /usr/local/bin/k3s<br/>[INFO] Skipping installation of SELinux RPM
```

Abbildung 12 Token Master

```
[INFO] Creating uninstall script /usr/local/bin/k3s-uninstall.sh<br/>[INFO] env: Creating environment file /etc/systemd/system/k3s.service.env<br/>[INFO] systemd: Creating service file /etc/systemd/system/k3s.service<br/>[INFO] systemd: Enabling k3s unit<br/>Created symlink /etc/systemd/system/multi-user.target.wants/k3s.service → /etc/systemd/system/k3s.service.<br/>[INFO] systemd: Starting k3s<br/>PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices>
```

Abbildung 10 K3S Master Installation

Nach dem erfolgreichen Installieren der Software auf der Master Node, brauchen wir für die Workers den Token des Masters. Dieser wird benötigt, damit die Workers begreifen, welches ihr Master ist.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> multipass exec k3s-master sudo cat /var/lib/rancher/k3s/server/node-token<br/>K10708F5312a9e90fdcdde4c175cf292643255f64bf4b71fcdbaba623fc41c191ec5:&:server:fe2a110fed4e87b50303d4155e7f5cd9<br/>PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices>
```

Abbildung 11 Token Master

Die Installation auf den Workers funktioniert gleich. Namen der VM ändern, Node IP angeben und den Token der Master Node angeben.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> multipass exec k3s-worker-1 -- bash -c "curl -sL https://get.k3s.io | K3S_URL=https://192.168.10.10:6443 K3S_TOKEN=K10708F5312a9e90fdcdde4c175cf292643255f64bf4b71fcdbaba623fc41c191ec5:&:server:fe2a110fed4e87b50303d4155e7f5cd9 INSTALL_K3S_EXEC='--node-ip=192.168.10.11' sh -<br/>[INFO] Finding release for channel stable<br/>[INFO] Using v1.28.6+k3s2 as release<br/>[INFO] Downloading hash https://github.com/k3s-io/k3s/releases/download/v1.28.6+k3s2/sha256sum-amd64.txt<br/>[INFO] Downloading binary https://github.com/k3s-io/k3s/releases/download/v1.28.6+k3s2/k3s<br/>[INFO] Verifying binary download<br/>[INFO] Installing k3s to /usr/local/bin/k3s<br/>[INFO] Skipping installation of SELinux RPM<br/>[INFO] Creating /usr/local/bin/kubectl symlink to k3s<br/>[INFO] Creating /usr/local/bin/crtl symlink to k3s<br/>[INFO] Creating /usr/local/bin/ctr symlinks to k3s<br/>[INFO] Creating killall script /usr/local/bin/k3s-killall.sh<br/>[INFO] Creating uninstall script /usr/local/bin/k3s-agent-uninstall.sh<br/>[INFO] env: Creating environment file /etc/systemd/system/k3s-agent.service.env<br/>[INFO] systemd: Creating service file /etc/systemd/system/k3s-agent.service<br/>[INFO] systemd: Enabling k3s-agent unit<br/>Created symlink /etc/systemd/system/multi-user.target.wants/k3s-agent.service → /etc/systemd/system/k3s-agent.service.<br/>[INFO] systemd: Starting k3s-agent<br/>PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices>
```

Abbildung 13 K3S Workers Installation

2.2 Kube VIP

Eine Kube VIP ermöglicht es, eine IP auf mehreren Knoten im Cluster zur Verfügung zu stellen. Die Kube VIP ist ein grosser Vorteil im Bereich der Hochverfügbarkeit im K3S-Cluster. Es dient zur Bereitstellung und Verwaltung der virtuellen IPs.

Die folgenden Schritte werden in den Nodes selbst getätigt. Mit dem Befehl «**ip a**» wird das Network Interface ermittelt. Dieses wird im nächsten Schritt verwendet. Bei der VIP wird eine Adresse angegeben, die noch verfügbar ist. Mit dem Befehl «**sudo -i**» kann eine interaktive Shell mit dem Superuser(root) eröffnet werden.

```
root@k3s-master:~# export INTERFACE=eth1
root@k3s-master:~# echo $INTERFACE
eth1
root@k3s-master:~# export VIP=192.168.10.10
root@k3s-master:~# echo $VIP
192.168.10.10
root@k3s-master:~#
```

Abbildung 14 EXPORT Kube VIP

Installiere die RBAC-Ressourcen. Es wird für Kube-VIP benötigt, damit es die Berechtigungen hat, um im Cluster zu arbeiten. Mit dem «**apply**» wird es ausgeführt.

```
ubuntu@k3s-master:~$ sudo -i
root@k3s-master:~# curl https://kube-vip.io/manifests/rbac.yaml > /var/lib/rancher/k3s/server/manifests/kube-vip-rbac.yaml
% Total    % Received % Xferd  Average Speed   Time      Time     Current
          Dload  Upload Total Spent   Left Speed
100 1066 100 1066    0     0  4809      0 --:--:-- --:--:-- --:--:-- 4801
root@k3s-master:~#
```

Abbildung 15 RBAC-Ressourcen

```
root@k3s-master:~# kubectl apply -f /var/lib/rancher/k3s/server/manifests/kube-vip-rbac.yaml
Warning: resource serviceaccounts/kube-vip is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
serviceaccount/kube-vip configured
Warning: resource clusterroles/system:kube-vip-role is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
clusterrole.rbac.authorization.k8s.io/system:kube-vip-role configured
Warning: resource clusterrolebindings/system:kube-vip-binding is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
clusterrolebinding.rbac.authorization.k8s.io/system:kube-vip-binding configured
root@k3s-master:~#
```

Abbildung 16 RBAC-Ressourcen apply

Damit die Kube-VIP funktioniert, erstellen wir einen Pod, indem das VIP später läuft.

```
root@k3s-master:~# kubectl run -it kube-vip-init --image=ghcr.io/kube-vip/kube-vip:main --restart=Never --rm -- manifest daemonset \
> --interface $INTERFACE \
> --address $VIP \
> --inCluster \
> --taint \
> --controlplane \
> --arp \
> --leaderElection > /var/lib/rancher/k3s/server/manifests/kube-vip-daemonset.yaml
root@k3s-master:~#
```

Abbildung 17 Daemonset-Manifest

Das generierte Manifest muss noch bearbeitet werden. Am Ende des YAML befindet sich ein «**Pod deleted**». Das ist zu entfernen. Alle 0.7.0 sollen durch 0.6.4 ersetzt werden. Es gibt einen Bug mit der 0.7.0 Version.

```
root@k3s-master:~# nano /var/lib/rancher/k3s/server/manifests/kube-vip-daemonset.yaml
root@k3s-master:~#
```

Abbildung 18 Manifest öffnen

```

value: 1
  - name: address
    value: 192.168.10.10
  - name: prometheus_server
    value: :2112
  image: ghcr.io/kube-vip/kube-vip:v0.6.4
  imagePullPolicy: Always
  name: kube-vip
  resources: {}
  securityContext:
    capabilities:
      add:
        - NET_ADMIN
        - NET_RAW
  hostNetwork: true
  serviceAccountName: kube-vip
  tolerations:
    - effect: NoSchedule
      operator: Exists
    - effect: NoExecute
      operator: Exists
  updateStrategy: {}
status:
  currentNumberScheduled: 0
  desiredNumberScheduled: 0
  numberMisscheduled: 0
  numberReady: 0

```

^G Help ^O Write Out ^W Where Is ^K Cut
 ^X Exit ^R Read File ^\ Replace ^U Paste

Abbildung 19 Kube VIP yaml

Das bearbeitete File ist noch zu installieren, mit dem Befehl «**apply**».

```
root@k3s-master:~# kubectl apply -f /var/lib/rancher/k3s/server/manifests/kube-vip-daemonset.yaml
Warning: resource daemonsets/kube-vip-ds is missing the kubectl.kubernetes.io/last-applied-configuration annotation which is required by kubectl apply. kubectl apply should only be used on resources created declaratively by either kubectl create --save-config or kubectl apply. The missing annotation will be patched automatically.
daemonset.apps/kube-vip-ds configured
root@k3s-master:~#
```

Abbildung 20 Kube VIP Manifest apply

2.3 Choco

Für die weiteren Schritte ist es nötig, «Choco» zu installieren. Es ist ein Paketverwaltungsmanager. Es ermöglicht dem Benutzer Software über die CLI zu installieren oder zu aktualisieren. Auf der Website von Chocolatey [«https://docs.chocolatey.org/en-us/choco/setup»](https://docs.chocolatey.org/en-us/choco/setup) ist die Installation via PowerShell oder CMD genau beschrieben. Achte darauf, die Installation mit Admin-Rechten auszuführen!

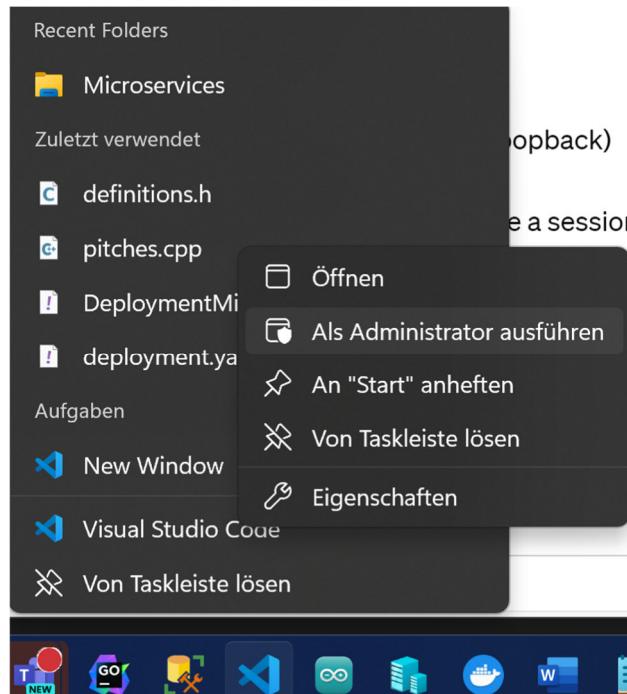


Abbildung 21 Choco Admin

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
Forcing web requests to allow TLS v1.2 (Required for requests to Chocolatey.org)
Getting latest version of the Chocolatey package for download.
Not using proxy.
Getting Chocolatey from https://community.chocolatey.org/api/v2/package/chocolatey/2.2.2.
Downloading https://community.chocolatey.org/api/v2/package/chocolatey/2.2.2 to C:\Users\mario\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip
Not using proxy.
Extracting C:\Users\mario\AppData\Local\Temp\chocolatey\chocoInstall\chocolatey.zip to C:\Users\mario\AppData\Local\Temp\chocolatey\chocoInstall
Installing Chocolatey on the local machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

chocolatey.nupkg file not installed in lib.
Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
WARNING: Not setting tab completion: Profile file does not exist at 'C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco ?? for a list of functions.
You may need to shut down and restart powershell and/or consoles
first prior to using choco.
Ensuring Chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> []
```

Abbildung 22 Choco Download

2.4 Kubectx

Installiere Kubectx via Choco. Es erleichtert das Navigieren zwischen Namespaces und anderen Kontexten.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> choco install kubens kubectx
Chocolatey v2.2.2
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot request has been detected, however, this is
being ignored due to the current Chocolatey configuration. If you
want to halt when this occurs, then either set the global feature
using:
  choco feature enable --name="exitOnRebootDetected"
or pass the option --exit-when-reboot-detected.

Installing the following packages:
kubens;kubectx
By installing, you accept licenses for the packages.
Progress: Downloading kubens 0.9.5... 100%

kubens v0.9.5 [Approved]
kubens package files install completed. Performing other installation steps.
The package kubens wants to run 'chocolateyinstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint): Y

Downloading kubens 64 bit
  from 'https://github.com/ahmetb/kubectx/releases/download/v0.9.5/kubens_v0.9.5_windows_x86_64.zip'
Progress: 100% - Completed download of C:\Users\mario\AppData\Local\Temp\chocolatey\kubens\0.9.5\kubens_v0.9.5_windows_x86_64.zip (9.59 MB).
Download of kubens_v0.9.5_windows_x86_64.zip (9.59 MB) completed.
Hashes match.
Extracting C:\Users\mario\AppData\Local\Temp\chocolatey\kubens\0.9.5\kubens_v0.9.5_windows_x86_64.zip to C:\ProgramData\chocolatey\lib\kubens...
C:\ProgramData\chocolatey\lib\kubens
ShimGen has successfully created a shim for kubens.exe
The install of kubens was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\kubens'
Progress: Downloading kubectx 0.9.5... 100%

kubectx v0.9.5 [Approved]
kubectx package files install completed. Performing other installation steps.
```

Abbildung 23 Kubectx Installieren

2.5 K3S-Config Master

Die Konfigurations-Datei des K3S muss angepasst werden. Erstelle von dem File ein Backup, damit die alten Einstellungen noch vorhanden sind.

```
ubuntu@k3s-master:~$ sudo -i  
root@k3s-master:~# cat /etc/rancher/k3s/k3s.yaml
```

Abbildung 24 K3S yaml öffnen

Abbildung 25 K3S yaml

Erstelle eine neue Config im Ordner. Die neue Config muss Cluster-Zertifikat, Kontext und Client-Zertifikat enthalten. Außerdem muss die IP auf die Master Node IP angepasst werden.

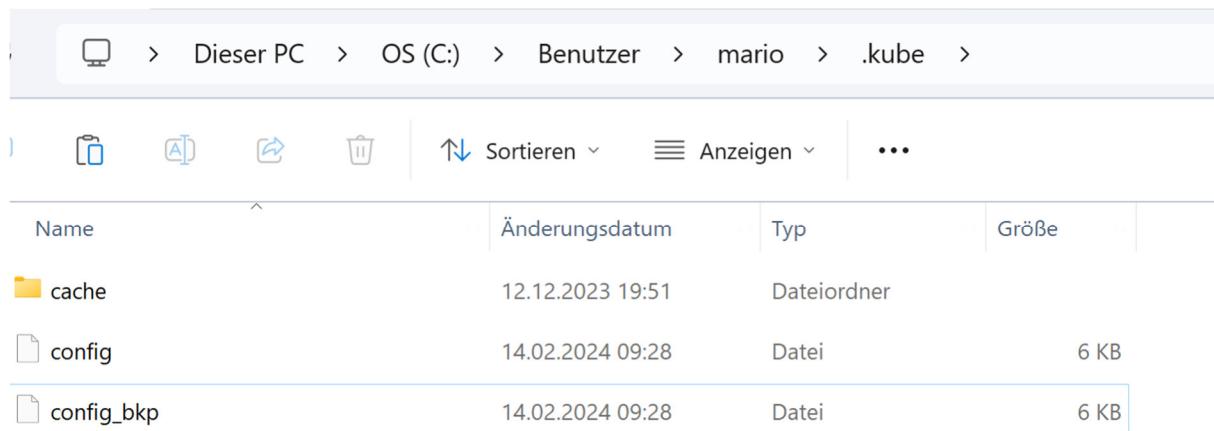


Abbildung 24 K3S config Backup

Konfiguration Nodes

Abbildung 25 K3S config angepasst

2.6 MetallB

MetallB ist ein Loadbalancer für ein Cluster. Er ermöglicht die dynamische Bereitstellung von IP-Adressen an verschiedene Dienste oder Services vom Typ Loadbalancer.

Um MetallB zu benutzen, muss zuerst das Helm-Repository installiert werden. Helm ist ein Paket Manager für Kubernetes.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> choco install kubernetes-helm
Chocolatey v2.2.2
3 validations performed. 2 success(es), 1 warning(s), and 0 error(s).

Validation Warnings:
- A pending system reboot request has been detected, however, this is
  being ignored due to the current Chocolatey configuration. If you
  want to halt when this occurs, then either set the global feature
  using:
  choco feature enable --name="exitOnRebootDetected"
  or pass the option --exit-when-reboot-detected.

Installing the following packages:
kubernetes-helm
By installing, you accept licenses for the packages.
Progress: Downloading kubernetes-helm 3.13.3... 100%

kubernetes-helm v3.13.3 [Approved]
kubernetes-helm package files install completed. Performing other installation steps.
The package kubernetes-helm wants to run 'chocolateyInstall.ps1'.
Note: If you don't run this script, the installation will fail.
Note: To confirm automatically next time, use '-y' or consider:
choco feature enable -n allowGlobalConfirmation
Do you want to run the script?([Y]es/[A]ll - yes to all/[N)o/[P]rint): Y

Downloading kubernetes-helm 64 bit
  from 'https://get.helm.sh/helm-v3.13.3-windows-amd64.zip'
Progress: 100% - Completed download of C:\Users\mario\AppData\Local\Temp\chocolatey\kubernetes-helm\3.13.3\helm-v3.13.3-windows-amd64.zip (15.59 MB).
Download of helm-v3.13.3-windows-amd64.zip (15.59 MB) completed.
Hashes match.
Extracting C:\Users\mario\AppData\Local\Temp\chocolatey\kubernetes-helm\3.13.3\helm-v3.13.3-windows-amd64.zip to C:\ProgramData\chocolatey\lib\kubernetes-helm\tools...
C:\ProgramData\chocolatey\lib\kubernetes-helm\tools
ShimGen has successfully created a shim for helm.exe
The install of kubernetes-helm was successful.
Software installed to 'C:\ProgramData\chocolatey\lib\kubernetes-helm\tools'

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).

Did you know the proceeds of Pro (and some proceeds from other
licensed editions) go into bettering the community infrastructure?
Your support ensures an active community, keeps Chocolatey tip-top,
plus it nets you some awesome features!
https://chocolatey.org/compare

○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> []
```

Abbildung 26 Install Helm

Füge das MetallB-Repository der Helm-Konfiguration hinzu.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> helm repo add metallb https://metallb.github.io/metallb
"metallb" has been added to your repositories
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> []
```

Abbildung 27 Add MetallB-Repository

Das helm repo muss jetzt noch geupdatet werden.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "metallb" chart repository
Update Complete. *Happy Helming!*
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> []
```

Abbildung 28 Helm repo Update

Jetzt kann MetallLB installiert werden. Bei der Installation wird auch noch ein Namespace angegeben. In diesem Namespace wird das MetallLB platziert.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices> helm upgrade --install metallb metallb/metallb --namespace metallb-system --createNamespace
Release "metallb" does not exist. Installing it now.
NAME: metallb
LAST DEPLOYED: Wed Feb 14 13:03:43 2024
NAMESPACE: metallb-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
MetallB is now running in the cluster.

Now you can configure it via its CRs. Please refer to the metallb official docs
on how to use the CRs.
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices>
```

Abbildung 29 MetallLB Installation

Damit MetallLB auch wunschgerecht läuft, sind noch zwei YAML-Files nötig.

«**ipaddresspool.yaml**» definiert einen Pool von verfügbaren IP-Adressen, die der MetallLB benutzen darf, um sie gewissen Services zuzuteilen.

```
Microservices > K3S > ! ipaddresspool.yaml
1  apiVersion: metallb.io/v1beta1
2  kind: IPAddressPool
3  metadata:
4    name: default
5    namespace: metallb-system
6  spec:
7    addresses:
8      - 192.168.10.31-192.168.10.40
```

Abbildung 30 ipaddresspool.yaml

«**l2advertisement.yaml**» konfiguriert die Layer-2 Zuweisung auf der angegebenen Schnittstelle.

```
K3S > ! l2advertisement.yaml
1  apiVersion: metallb.io/v1beta1
2  kind: L2Advertisement
3  metadata:
4    name: default
5    namespace: metallb-system
6  spec:
7    ipAddressPools:
8      - default
9    interfaces:
10      - eth1
```

Abbildung 31 l2advertisement.yaml

Nach dem Erstellen der beiden YAML-Files müssen diese noch angewandt werden.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\k3s> kubectl apply -f ./ipaddresspool.yaml
ipaddresspool.metallb.io/default created
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\k3s> kubectl apply -f ./l2advertisement.yaml
l2advertisement.metallb.io/default created
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\k3s>
```

Abbildung 32 YAML-Files applizieren

2.7 nfs common

NFS ist ein Protokoll, welches Computern ermöglicht, Ressourcen zuzuteilen und - greifen. Es erleichtert die Skalierung und Verwaltung von Workloads in einem Cluster. Verbinde dich via «**multipass shell Name-VM**» mit den einzelnen Instanzen.

```
ubuntu@k3s-master:~$ sudo apt update
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1371 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1048 kB]
Fetched 2648 kB in 2s (1395 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@k3s-master:~$ 
```

Abbildung 33 sudo apt update für nfs

Installieren nfs-common auf jeder VM. Dafür ist es nötig diese Schritte mit jeder VM auszuführen.

```
ubuntu@k3s-master:~$ sudo apt install nfs-common
```

Abbildung 34 nfs common Installation

2.8 Longhorn

Longhorn ist eine Open-Source-Software für die Speicherung in Clustern. Longhorn arbeitet mit einer verteilten Architektur. Über mehrere Knoten im Cluster werden die Daten repliziert. Somit kann Longhorn eine Ausfallsicherheit gewährleisten. Es bietet somit eine hochverfügbare Speicherlösung für ein Cluster.

Mit dem Befehl kann Longhorn inklusive Deployment, Service, PVC und ConfigMap installiert werden. Die enthaltenen Angaben im YAML-File sind nötig, damit Longhorn korrekt installiert wird.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\k3s> kubectl apply -f https://raw.githubusercontent.com/longhorn/longhorn/v1.5.3/deploy/longhorn.yaml
namespace/longhorn-system created
serviceaccount/longhorn-service-account created
serviceaccount/longhorn-support-bundle created
configmap/longhorn-default-setting created
configmap/longhorn-storageclass created
customresourcedefinition.apiextensions.k8s.io/backingimagedatasources.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/backingimagemanagers.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/backingimages.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/backups.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/backuptargets.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/backupvolumes.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/engineimages.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/engines.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/instancemanagers.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/nodes.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/orphans.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/recurringjobs.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/replicas.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/settings.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/sharemanagers.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/snapshots.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/supportbundles.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/systembackups.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/systemrestores.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/volumes.longhorn.io created
customresourcedefinition.apiextensions.k8s.io/volumeattachments.longhorn.io created
clusterrole.rbac.authorization.k8s.io/longhorn-role created
clusterrolebinding.rbac.authorization.k8s.io/longhorn-bind created
clusterrolebinding.rbac.authorization.k8s.io/longhorn-support-bundle created
service/longhorn-backend created
service/longhorn-frontend created
service/longhorn-conversion-webhook created
service/longhorn-admission-webhook created
service/longhorn-recovery-backend created
service/longhorn-engine-manager created
service/longhorn-replica-manager created
daemonset.apps/longhorn-manager created
deployment.apps/longhorn-driver-deployer created
deployment.apps/longhorn-ui created
○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\k3s> 
```

Abbildung 35 Longhorn Installation

3. Zusammenstellung yaml

Damit die entsprechend erstellten YAML-Files greifen, können diese nach Erstellung mit folgendem Kommando angewandt werden:

Kubectl apply -f xy.yaml

Dies wurde unter den einzelnen Schritten nicht mehr manuell erwähnt. Möglich wäre auch, ganz am Schluss mit **kubectl apply -f**. alles anzuwenden, anstatt nur einzelne Teile.

3.1 Docker image

Um das Image für den Webserver und den Resultserver zu erstellen, wird zuerst ein Dockerfile benötigt, welches die Voraussetzungen installiert.

FROM: Gibt das Basis Image an, das verwendet wird

WORKDIR: Gibt das Arbeitsverzeichnis an, wo das Image die Daten verwaltet für den Buildprozess

COPY: Kopiert die requirements.txt Datei an den Root Pfad. Das wäre hier /app

RUN: Installiert via pip alle Abhängigkeiten, welche im zuvor kopierten requirements.txt kopiert wurden

COPY: Kopiert nun die im Ordner vorhandenen Dateien in das WORKDIR des Containers

CMD: Führt die Befehle **python** und anschliessend **app.py** aus

```
FROM python:3.8
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["python", "app.py"]
```

Abbildung 36 Dockerfile erstellen

Nun werden die beiden Images mit folgendem Befehl erstellt. Wichtig dabei ist, dass der Name des Images im Format **github-Account/image-name** erfolgt. Dadurch kann dies anschliessend direkt auf github hochgeladen werden.

Docker build -t trennety/web-server .

Docker build -t trennety/results-server .

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> docker build -t trennety/web-server .
[+] Building 2.5s (1/11) FINISHED
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 175B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:3.8
=> [auth] library/python:pull token for registry-1.docker.io
=> [1/5] FROM docker.io/library/python:3.8@sha256:23e62414c3310930888bb1690b7f723f52f7ab3a26ff9671e9747f60d169ee96
=> [internal] load build context
=> => transferring context: 248B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt .
=> CACHED [4/5] RUN pip install -r requirements.txt
=> CACHED [5/5] COPY .
=> exporting to image
=> => exporting layers
=> => writing image sha256:d81060d9b7d5aa5f4572732bd1966d4719416e36807105a4e6a1a16adafab644
=> => naming to docker.io/trennety/web-server

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 37 Dockerfile Image erstellen

Wenn beide Images erstellt wurden, können diese nun auf dockerhub gepusht werden. Voraussetzung ist, dass man eingeloggt ist. Sollte dem nicht so sein, kann via «**docker login**» eingeloggt werden. Anschliessend den folgenden Befehl verwenden für den push:

Docker push trennety/result-server

Docker push trennety/web-server

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> docker push trennety/result-server
Using default tag: latest
The push refers to repository [docker.io/trennety/result-server]
77a0f0fad12c: Layer already exists
a4a14b87acc4: Layer already exists
9f2986469bb3: Layer already exists
ffd6a97a92c7: Layer already exists
788727a0defa: Layer already exists
c1f788d2750d: Layer already exists
c6bad6927871: Layer already exists
e077e19b6682: Layer already exists
21e1c4948146: Layer already exists
68866beb2ed2: Layer already exists
e6e2ab10dba6: Layer already exists
0238a1790324: Layer already exists
● latest: digest: sha256:0c9456be79eb6561acce4f55f6bd8367a609f7edf232475e3ad651a2a7ad007 size: 2840
○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 38 Image auf dockerhub publizieren

Diese sind dann anschliessend auf Dockerhub ersichtlich.

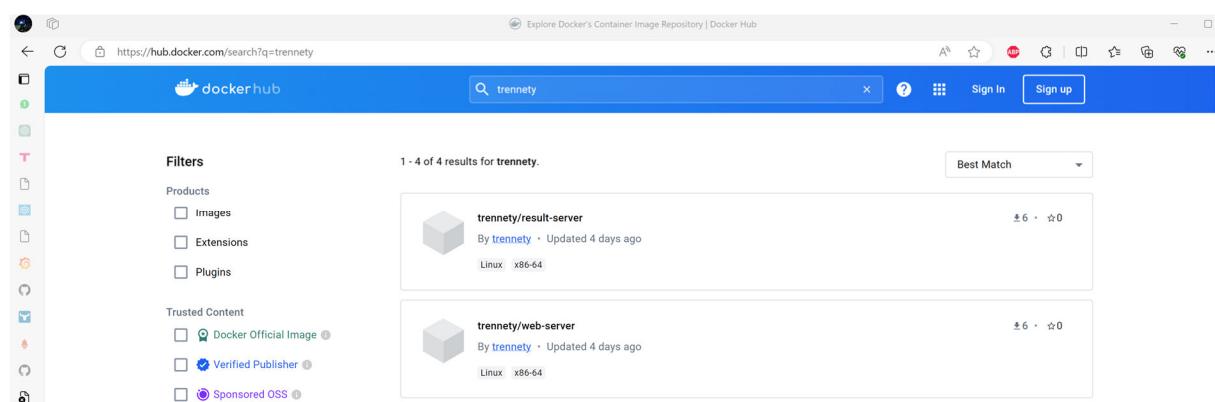


Abbildung 39 Images auf dockerhub publiziert

3.2 Deployments

In diesem Abschnitt werden alle getätigten Deployments aufgelistet. Deployments sind Konfiguration-Files, welche den Pod beschreiben und dessen Konfigurationen auflisten. Folgende 3 Deployments wurden erstellt.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> kubectl get deployments
  NAME           READY   UP-TO-DATE   AVAILABLE   AGE
mysql-deployment  1/1     1            1           3d14h
result-server-deployment  2/2     2            2           5d17h
web-server-deployment  2/2     2            2           5d17h
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 40 Deployments Übersicht

Result-server-deployment:

```
Microservices > K3S > ! result-deployment.yaml
 1  apiVersion: apps/v1
 2  kind: Deployment
 3  metadata:
 4    name: result-server-deployment
 5    namespace: exampleapp
 6    labels:
 7      app: result-server
 8  spec:
 9    replicas: 2
10    strategy:
11      type: RollingUpdate
12      rollingUpdate:
13        maxUnavailable: 1 # kann in Prozent angegeben werden
14        maxSurge: 1 # kann in Prozent angegeben werden
15    selector:
16      matchLabels:
17        app: result-server
18    template:
19      metadata:
20        labels:
21          app: result-server
22      spec:
23        containers:
24          - name: result-server
25            image: trennety/result-server
26            ports:
27              - containerPort: 80
28            env:
29              - name: MYSQL_DATABASE_HOST
30                value: mysql-service # Service name for the database
31              - name: MYSQL_USER
32                value: Patrick
33              - name: MYSQL_PASSWORD
34                value: Patrick2024
35              - name: MYSQL_DATABASE
36                value: ExampleAppDatabase
37            readinessProbe:
38              httpGet:
39                path: /
40                port: 80
41              failureThreshold: 30 # check 30 times
42              periodSeconds: 10 # every 10 seconds -> 5 Minutes
```

Abbildung 41 result-server-deployment Teil 1

```

43      livenessProbe:
44        httpGet:
45          path: /
46          port: 80
47        initialDelaySeconds: 15 # start 15 seconds after the container starts OR startup probe succeeds
48        periodSeconds: 20
49        timeoutSeconds: 5 # fail after 5 seconds without an answer
50        failureThreshold: 3 # restart container after 3 fails
51

```

Abbildung 42 result-server-deployment Teil 2

Web-server-deployment:

```

Microservices > K3S > ! web-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: web-server-deployment
5    namespace: exampleapp
6    labels:
7      app: web-server
8  spec:
9    replicas: 2
10   strategy:
11     type: RollingUpdate
12     rollingUpdate:
13       maxUnavailable: 1 # kann in Prozent angegeben werden
14       maxSurge: 1 # kann in Prozent angegeben werden
15   selector:
16     matchLabels:
17       app: web-server
18   template:
19     metadata:
20       labels:
21         app: web-server
22     spec:
23       containers:
24         - name: web-server
25           image: trennety/web-server
26           ports:
27             - containerPort: 80
28           env:
29             - name: MYSQL_DATABASE_HOST
30               value: mysql-service # Service name for the database
31             - name: MYSQL_USER
32               value: Patrick
33             - name: MYSQL_PASSWORD
34               value: Patrick2024
35             - name: MYSQL_DATABASE
36               value: ExampleAppDatabase
37   readinessProbe:
38     httpGet:
39       path: /
40       port: 80
41     failureThreshold: 30 # check 30 times
42     periodSeconds: 10 # every 10 seconds -> 5 Minutes

```

Abbildung 43 web-server-deployment Teil 1

```

43      livenessProbe:
44        httpGet:
45          path: /
46          port: 80
47        initialDelaySeconds: 15 # start 15 seconds after the container starts OR startup probe succeeds
48        periodSeconds: 20
49        timeoutSeconds: 5 # fail after 5 seconds without an answer
50        failureThreshold: 3 # restart container after 3 fails

```

Abbildung 44 web-server-deployment Teil 2

Mysql-deployment:

```

Microservices > K3S > ! mySQL-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mysql-deployment
5    namespace: exampleapp
6  spec:
7    replicas: 1
8    selector:
9      matchLabels:
10     app: mysql
11    template:
12      metadata:
13        labels:
14          app: mysql
15      spec:
16        containers:
17          - name: mysql
18            image: mysql:5.6
19            env:
20              - name: MYSQL_ROOT_PASSWORD
21                value: MarioAndrin2024
22              - name: MYSQL_USER
23                value: Patrick
24              - name: MYSQL_PASSWORD
25                value: Patrick2024
26              - name: MYSQL_DATABASE
27                value: ExampleAppDatabase
28            ports:
29              - containerPort: 3306
30            volumeMounts:
31              - name: mysql-persistent-storage
32                mountPath: /var/lib/mysql
33            volumes:
34              - name: mysql-persistent-storage
35                persistentVolumeClaim:
36                  claimName: mysql-pvc
37

```

Abbildung 45 mysql-deployment

Mysql hat kein Frontend, dafür aber result und web, welche via Portforward angeschaut werden können. Dazu z.B. im Lens das Portforward einrichten. Alternativ geht das auch mit folgendem Befehl:

kubectl port-forward Pod-Name Lokaler-Port:Pod-Port

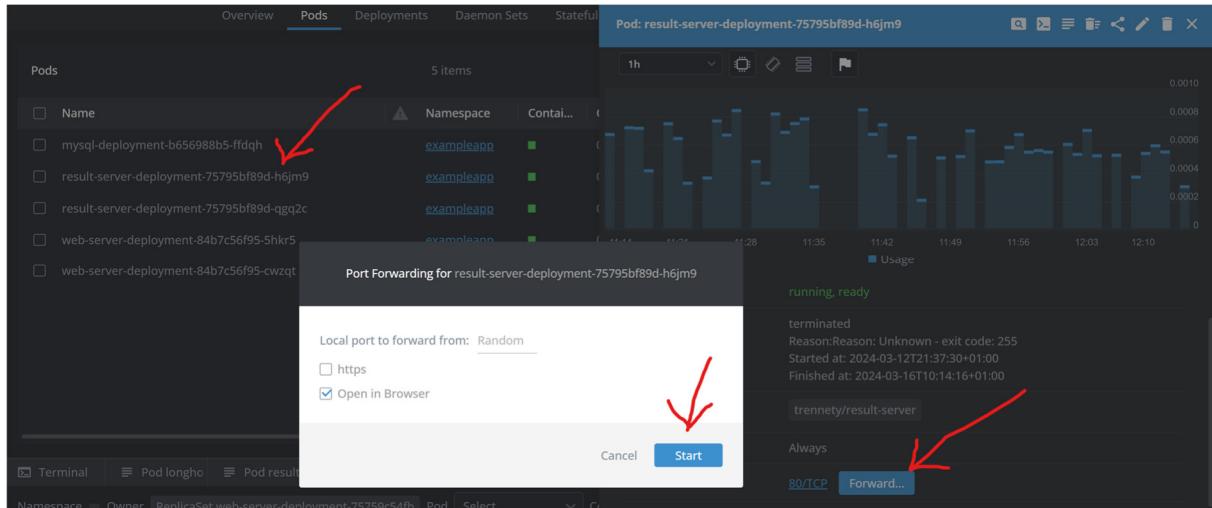


Abbildung 46 Port-forward in Lens

Der Web-Server ist nun ersichtlich, bei dem die Optionen Add, Update und Delete zur Verfügung stehen. Die Applikation kann genutzt werden.

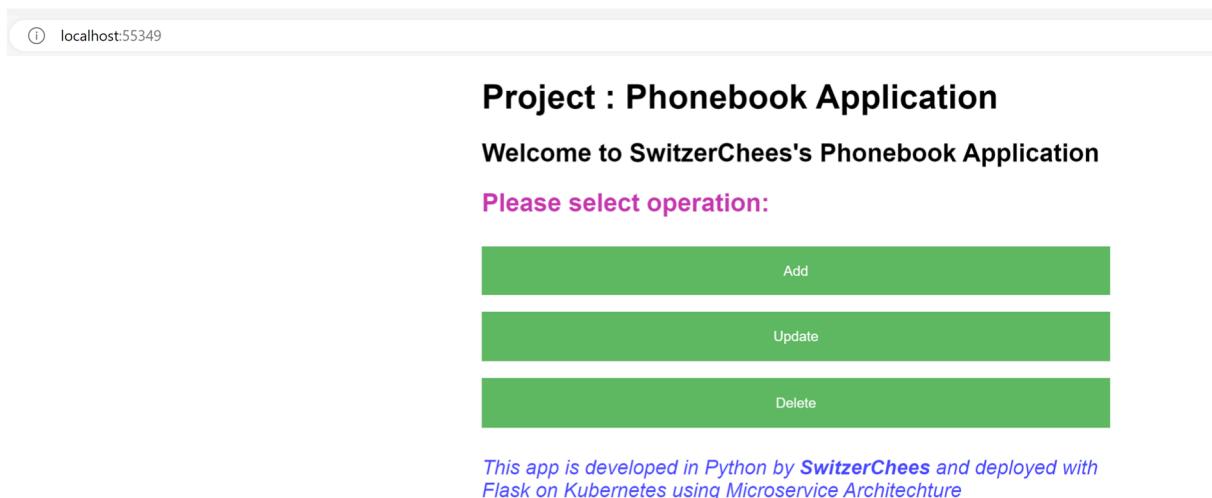


Abbildung 47 Web-Server Frontend

Der Result-Server ist nun ersichtlich, bei dem die Suchfunktion eines Eintrags zur Verfügung steht. Die Applikation kann ebenfalls genutzt werden.

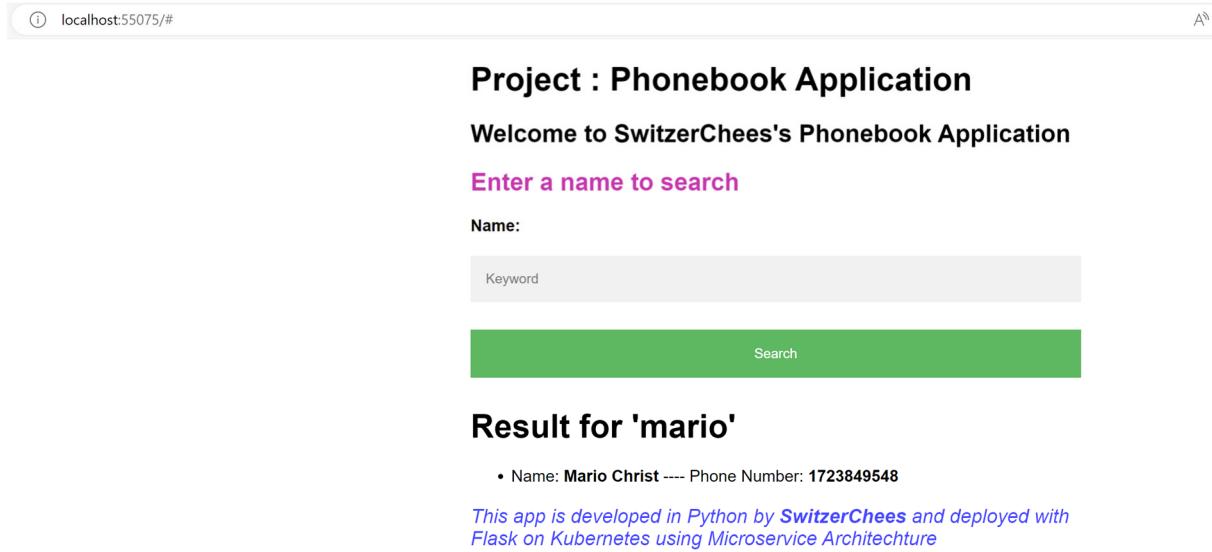


Abbildung 48 Result-Server Frontend

3.3 Services

Services in Kubernetes werden benötigt, um das Deployment bzw. den Pod verfügbar zu machen. Dies kann entweder intern, aber auch extern sein. Im folgenden Abschnitt werden sämtliche Service-Konfigurationen aufgelistet.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> kubectl get svc
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP  PORT(S)   AGE
mysql-service  ClusterIP  10.43.253.255 <none>       3306/TCP  5d17h
result-server-service  ClusterIP  10.43.213.202 <none>       80/TCP    5d17h
web-server-service  ClusterIP  10.43.73.140 <none>       80/TCP    5d17h
○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 49 Übersicht Services

Result-server-service:

```
Microservices > K3S > ! result-service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: result-server-service
5    namespace: exampleapp
6  spec:
7    selector:
8      app: result-server
9    ports:
10      - protocol: TCP
11        name: http
12        port: 80
13        targetPort: 80
14      type: ClusterIP
15 
```

Abbildung 50 result-server-service

Web-server-service:

```
Microservices > K3S > ! web-service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: web-server-service
5    namespace: exampleapp
6  spec:
7    selector:
8      app: web-server
9    ports:
10      - protocol: TCP
11        name: http
12        port: 80
13        targetPort: 80
14      type: ClusterIP 
```

Abbildung 51 web-server-service

Mysql-service:

```
Microservices > K3S > ! mySQL-service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mysql-service
5    namespace: exampleapp
6  spec:
7    selector:
8      app: mysql
9    ports:
10      - protocol: TCP
11        port: 3306
12        targetPort: 3306
13      type: ClusterIP
```

Abbildung 52 mysql-service

Im Grundsatz könnten die Services via Portforward weitergeleitet werden. Wenn das Portforwarding über das Deployment getätigkt wird, wird der lokale Pod exponiert. Beim Service würde der dafür vorgesehenen Service exponiert, was keinen Unterschied macht. Dies wird nicht explizit aufgeführt, weil die Funktionalität bereits zuvor mit den Pods demonstriert wurde.

3.4 PVC

Damit der mysql Pod einen persistenten Ablageort für die Daten hat, benötigt dieser ein sogenanntes pv (persistent volume). Dieses muss zuerst via pvc (persistent volume claim) beantragt werden.

Folgende PVC und im Zuge dessen PV wurden erstellt.

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> kubectl get pvc
  NAME      STATUS    VOLUME                                     CAPACITY  ACCESS MODES  STORAGECLASS  AGE
  mysql-pvc  Bound     pvc-0806a222-f5cf-4bef-ba8c-6c044437c86c   1Gi       RWO          longhorn     3d15h
○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 53 pvc's

```
● PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> kubectl get pv
  NAME           CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM           STORAGECLASS  REASON  AGE
  pvc-0806a222-f5cf-4bef-ba8c-6c044437c86c   1Gi       RWO          Delete        Bound   exampleapp/mysql-pvc  longhorn     3d15h
  pvc-b6f71161-66c5-4d8a-a2ae-f45ea6fd5c0b   10Gi      RWO          Delete        Bound   grafana/grafana   longhorn     3d17h
○ PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S\web_server> 
```

Abbildung 54 pv

Im Screenshot ist zusätzlich das pv von Grafana zu sehen, welches bereits zuvor installiert wurde. Dies ist aber unabhängig von unserem mysql-pvc. Die Funktionalität wird dann in den Tests nachgewiesen.

Mysql-pvc:

```
Microservices > K3S > ! mySQL-pvc.yaml
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: mysql-pvc
5  spec:
6    accessModes:
7      - ReadWriteOnce
8    resources:
9      requests:
10        storage: 1Gi
```

Abbildung 55 mysql-pvc

3.5 Ingress

Um die Applikationen von aussen hin via Namen zu erreichen, können sogenannte Ingress Konfigurationen erstellt werden. Erfolgt eine Anfrage auf die zugewiesene, vom Netz ausserhalb erreichbare IP-Adresse, so weist der Ingress Service den anfragenden Client an die korrekte interne IP-Adresse zu, basierend auf dem angefragten Namen.

Damit dies funktioniert, muss für result-server und web-server jeweils ein eigener Ingress Service konfiguriert werden.

Result-server-ingress:

```
Microservices > K3S > ! result-server-ingress.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: result-server-ingress
5    namespace: exampleapp
6    annotations:
7      nginx.ingress.kubernetes.io/from-to-www-redirect: "true" # Redirect www to non-www
8  spec:
9    ingressClassName: nginx
10   tls:
11     - hosts:
12       - result-server.meta # Replace with your domain
13       secretName: result-server.meta-tls # Replace with your domain
14   rules:
15     - host: result-server.meta # Replace with your domain
16       http:
17         paths:
18           - path: /
19             pathType: Prefix
20             backend:
21               service:
22                 name: result-server-service
23                 port:
24                   name: http
```

Abbildung 56 result-server-ingress

Web-server-ingress:

```
Microservices > K3S > ! web-server-ingress.yaml
1  apiVersion: networking.k8s.io/v1
2  kind: Ingress
3  metadata:
4    name: web-server-ingress
5    namespace: exampleapp
6    annotations:
7      nginx.ingress.kubernetes.io/from-to-www-redirect: "true" # Redirect www to non-www
8  spec:
9    ingressClassName: nginx
10   tls:
11     - hosts:
12       - web-server.meta # Replace with your domain
13       secretName: web-server.meta-tls # Replace with your domain
14     rules:
15       - host: web-server.meta # Replace with your domain
16         http:
17           paths:
18             - path: /
19               pathType: Prefix
20               backend:
21                 service:
22                   name: web-server-service
23                   port:
24                     name: http
```

Abbildung 57 web-server-ingress

Da die Namen in keinem DNS-Server eingetragen sind, müssen diese manuell im hosts File angelegt werden.

Windows: C:\Windows\System32\drivers\etc\hosts

Linux: /etc/hosts

Dort sollten dann entsprechende Einträge für die Namen eingetragen werden.

Die zu verwendende IP-Adresse wurde im IPAdressPool.yaml file konfiguriert während der MetalLB Installation.

```
GNU nano 6.2
127.0.0.1      localhost
127.0.1.1      mario-Virtual-Machine
192.168.10.31   web-server.meta
192.168.10.31   result-server.meta

# The following lines are desirable for IPv6 capable hosts
::1      ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Abbildung 58 Linux hosts file Anpassung

Sind die Namen im hosts File eingetragen, so sollten sich die Frontends für web-/ und result-server via Namen erreichen lassen. Dazu entsprechend einfach den Namen in einem Web-Browser eingeben.

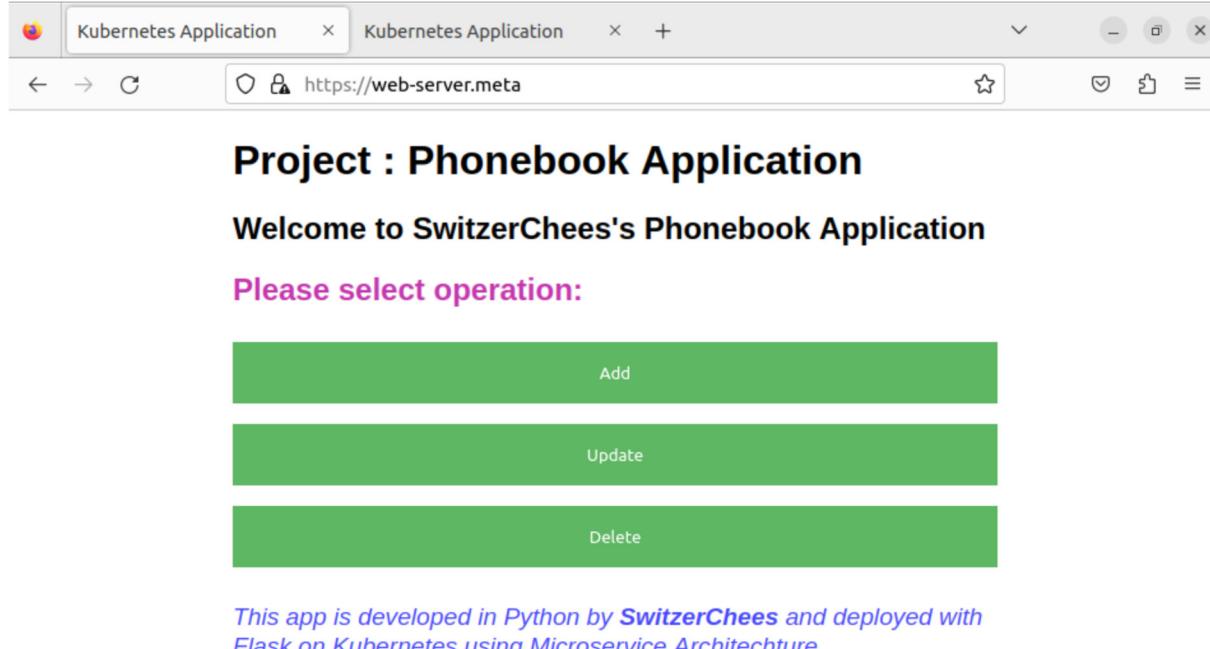


Abbildung 59 web-server.meta

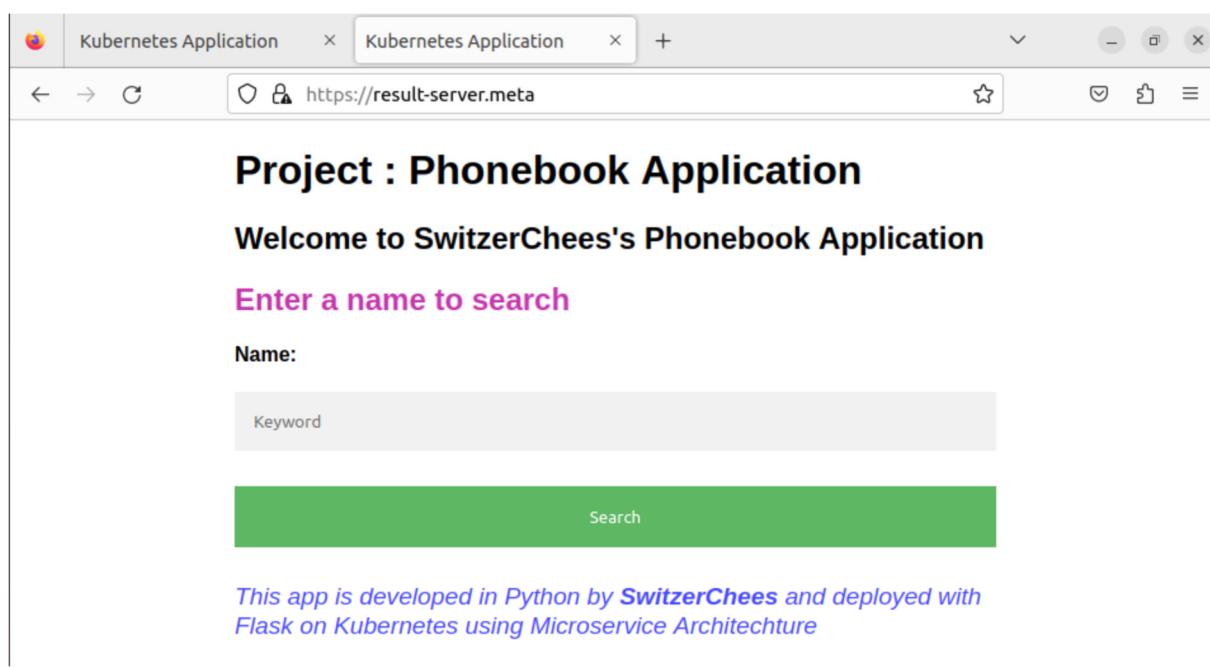


Abbildung 60 result-server.meta

Monitoring

Im folgenden Abschnitt werden alle nötigen Installationen und Konfigurationen vorgenommen, damit die einzelnen Pods überwacht werden können. Anhand der gesammelten Metriken kann im Anschluss dann dies grafisch aufbereitet und dargestellt werden.

4.1 Helm repo

Damit wir die Monitoring Tools Prometheus und Grafana vom Ersteller «bitnami» verwenden können, müssen wir diese zu unserem Repository hinzufügen. Andernfalls würde es dort gar nicht suchen. Dies wird mit folgendem Befehl gemacht:

Helm.exe repo add bitnami https://charts.bitnami.com/bitnami

```
● PS C:\Users\mario> helm.exe repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
PS C:\Users\mario> █
```

Mit folgendem Befehl muss unser Repository mit dem neu hinzugefügten Repository geupdated werden:

Helm repo update

```
● PS C:\Users\mario> helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "metallb" chart repository
...Successfully got an update from the "bitnami" chart repository
Update Complete. *Happy Helming!*
○ PS C:\Users\mario> █
```

Mit dem hinzufügen und update des Repository's von bitnami können wir später Grafana und Prometheus installieren.

4.2 Nginx Ingress

Zuvor haben wir die Ingress Services erstellt, aber es gibt noch keinen Ingress Service, der die Anfrage entgegennimmt und diese dann entsprechend verteilt. Wir haben lediglich dem web-server und dem result-server einen Ingress Namen gegeben, mit welchem der eigentliche Ingress Service die Anfragen verteilen kann.

Um nun den Ingress Service zu installieren, wird folgender Befehl verwendet:

```
helm upgrade --install nginx-ingress bitnami/nginx-ingress-controller --  
namespace nginx-ingress --create-namespace
```

Dadurch wurde ein eigener Namespace mit einem Pod erzeugt, welche diese Funktion übernimmt.

Pods											Namespace: nginx-ingress		Search	Download
Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮			
nginx-ingress-nginx-ingress-controller-677bbffcc8-shbkm	nginx-ingress	■	0.007	42.5MiB	4	ReplicaSet	k3s-worker-6	BestEffort	3d18h	Running	⋮			
nginx-ingress-nginx-ingress-controller-default-backend-544	nginx-ingress	■	0.001	5.0MiB	2	ReplicaSet	k3s-worker-6	BestEffort	3d18h	Running	⋮			

Abbildung 61 nginx ingress pods

Der Nginx-Ingress Service hat nichts mit dem Monitoring zu tun. Er befindet sich aber in diesem Abschnitt, da ebenfalls das bitnami Repository benutzt.

4.3 Prometheus

Prometheus ist eine Lösung, die in jedem Pod installiert wird und Daten wie Metriken, Logs und andere Werte sammelt. Diese Daten können von verschiedenen Tools verarbeitet und analysiert werden.

Zuerst müssen wir die folgenden Ordner mit den entsprechenden YAML-Dateien erstellen, damit der Installer später die Standardwerte auslesen kann. Ohne diese Dateien würde der Installer nicht wissen, welche Werte verwendet werden sollen.

Ordner: kube-prometheus

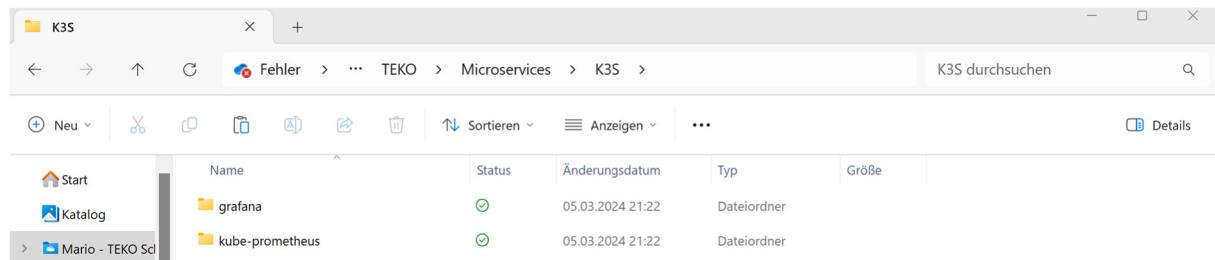


Abbildung 62 Ordner kube-prometheus

YAML-File: values.yml

```
Microservices > K3S > kube-prometheus > ! values.yml
1   operator:
2     enabled: true
3   alertmanager:
4     enabled: false
5   blackboxExporter:
6     enabled: false
7
```

Abbildung 63 kube-prometheus values.yml

Prometheus ist mit folgendem Befehl zu installieren:

```
helm upgrade --install kube-prometheus bitnami/kube-prometheus --namespace kube-prometheus --create-namespace -f kube-prometheus/values.yml
```

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> helm upgrade --install kube-prometheus bitnami/kube-prometheus --namespace kube-prometheus --create-namespace -f kube-prometheus/values.yml
Release "kube-prometheus" does not exist. Installing it now.
NAME: kube-prometheus
LAST DEPLOYED: Tue Mar 5 21:25:58 2024
NAMESPACE: kube-prometheus
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
  CHART NAME: kube-prometheus
  CHART VERSION: 8.29.1
  APP VERSION: 0.72.0
** Please be patient while the chart is being deployed **
Watch the Prometheus Operator Deployment status using the command:
  kubectl get deploy -w --namespace kube-prometheus -l app.kubernetes.io/name=kube-prometheus-operator,app.kubernetes.io/instance=kube-prometheus
Watch the Prometheus StatefulSet status using the command:
```

Abbildung 64 Installation Prometheus Teil 1

```
kubectl get deploy -w --namespace kube-prometheus -l app.kubernetes.io/name=kube-prometheus-operator,app.kubernetes.io/instance=kube-prometheus
Watch the Prometheus StatefulSet status using the command:
● kubectl get sts -w --namespace kube-prometheus -l app.kubernetes.io/name=kube-prometheus-prometheus,app.kubernetes.io/instance=kube-prometheus

Prometheus can be accessed via port "9090" on the following DNS name from within your cluster:
kube-prometheus-prometheus.kube-prometheus.svc.cluster.local

To access Prometheus from outside the cluster execute the following commands:
echo "Prometheus URL: http://127.0.0.1:9090"
kubectl port-forward --namespace kube-prometheus svc/kube-prometheus-prometheus 9090:9090

WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For production installations, please set the following values according to your workload needs:
  .prometheus.resources
  .prometheus.thanos.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> []
```

Abbildung 65 Installation Prometheus Teil 2

Ist nun Prometheus installiert, sollte sich das Frontend via Port-Forward öffnen lassen. Ebenfalls ist ersichtlich, dass wenn z.B. «machine_cpu_cores» eingegeben wird, dass die einzelnen Nodes mit den verfügbaren Cores aufgelistet werden. Die Abfrage der Metriken funktioniert einwandfrei.

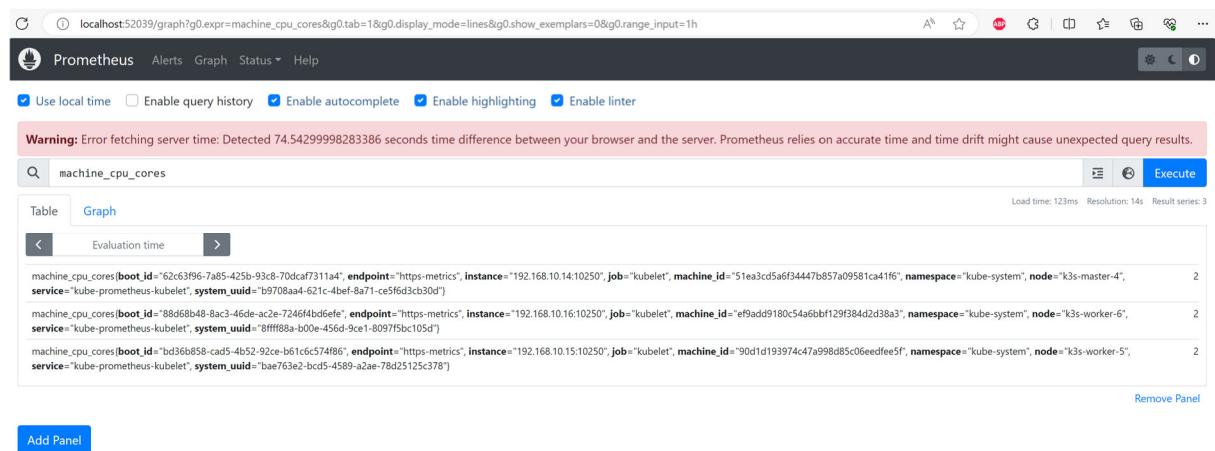
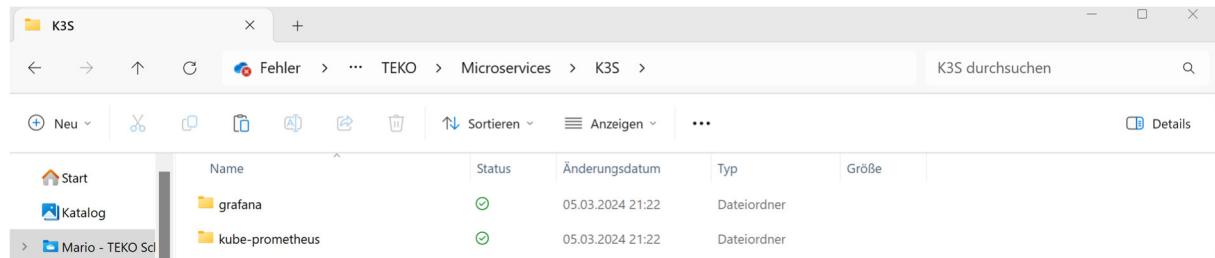


Abbildung 66 Prometheus Übersicht

4.4 Grafana

Grafana ist eine Darstellungssoftware, welche die zuvor gesammelten Metriken von Prometheus, oder anderen Metrik-sammelnden Software, aufzeigt. Auch bei Grafana ist es nötig, zuerst den Ordner und die Datei zu erstellen, damit es die Werte für die Installation benutzen kann.

Ordner: grafana



YAML-File: values.yml

Abbildung 67 Ordner Grafana

```
Microservices > K3S > grafana > ! values.yaml
1 admin:
2   ## @param admin.user Grafana admin username
3   ##
4   user: "admin"
5   ## @param admin.password Admin password. If a password is not provided a random password will be generated
6   ##
7   password: "supersecret123"
8
```

Abbildung 68 Grafana values.yaml

Grafana ist mit folgendem Befehl zu installieren:

helm upgrade --install grafana bitnami/grafana --namespace grafana --create-namespace -f grafana/values.yaml

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> helm upgrade --install grafana bitnami/grafana --namespace grafana --create-namespace -f grafana/values.yaml
Release "grafana" does not exist. Installing it now.
NAME: grafana
LAST DEPLOYED: Tue Mar 5 21:28:48 2024
NAMESPACE: grafana
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
CHART NAME: grafana
CHART VERSION: 9.11.1
APP VERSION: 10.3.1

** Please be patient while the chart is being deployed **

1. Get the application URL by running these commands:
  echo "Browse to http://127.0.0.1:8888"
  kubectl port-forward svc/grafana:8888:3000 &

2. Get the admin credentials:
  echo "User: admin"
  echo "Password: $(kubectl get secret grafana-admin --namespace grafana -o jsonpath="{.data.GF_SECURITY_ADMIN_PASSWORD}" | base64 -d)"
# Note: Do not include grafana.validateValues.database here. See https://github.com/bitnami/charts/issues/20629

WARNING: There are "resources" sections in the chart not set. Using "resourcesPreset" is not recommended for production. For production installations, please set the following values according to your workload needs:
- grafana.resources
+info https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S>
```

Abbildung 69 Installation Grafana

Einerseits können nun die Grafiken in Lens ausgelesen werden:

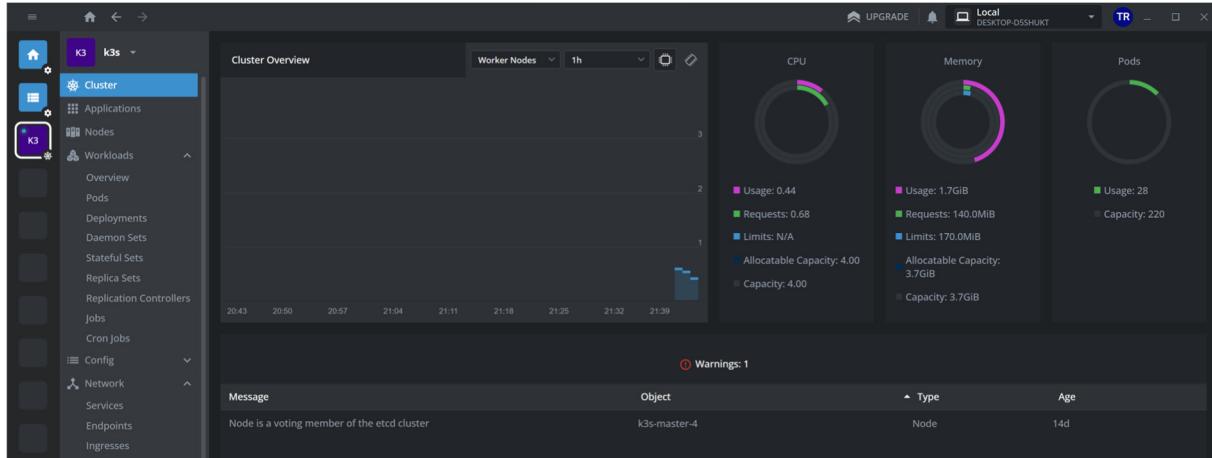


Abbildung 70 Lens Metriken auslesen

Und andererseits kann Grafana via Port Forward zur Verfügung gestellt werden.

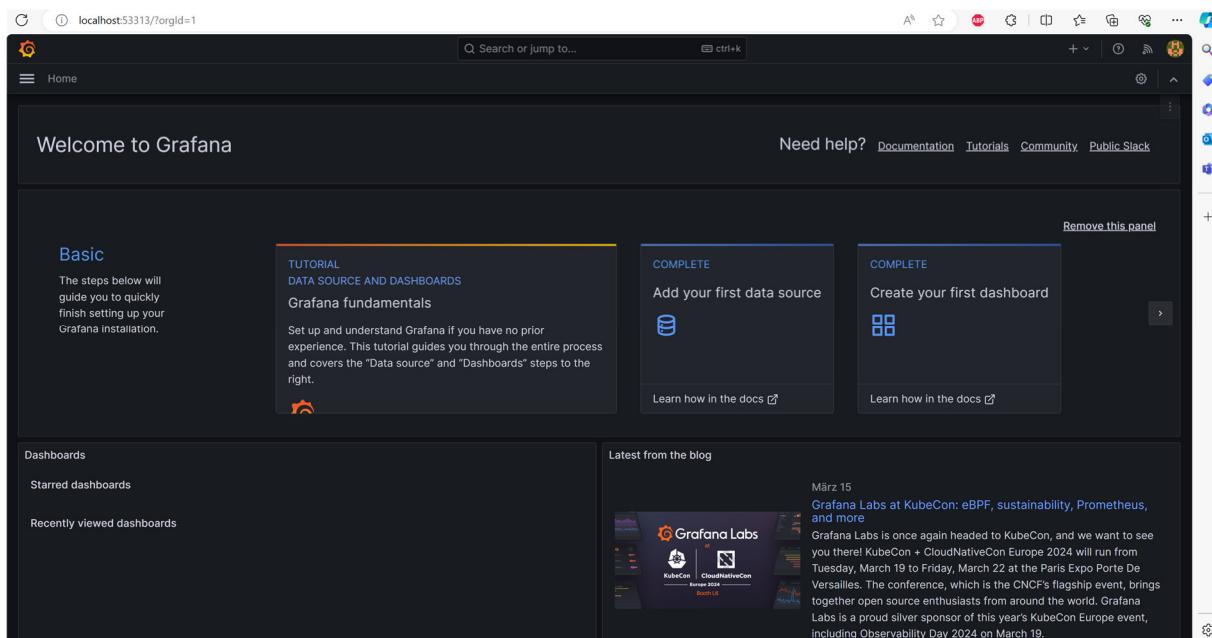


Abbildung 71 Grafana Welcome Site

Da ist standardmäßig noch nicht viel zu sehen, deshalb müssen wir erst ein Dashboard und eine Quelle, in unserem Fall Prometheus, hinzufügen. Dazu unter Home->Dashboards->New->New Dashboard ein neues Dashboard hinzufügen.

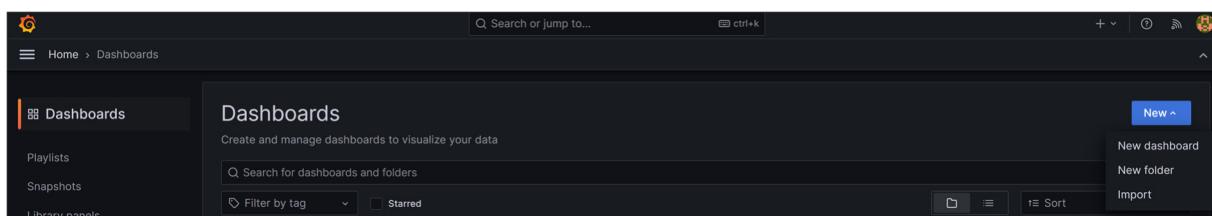


Abbildung 72 Grafana New Dashboard hinzufügen

Anschliessend wird ein neues Dashboard importiert, da es bereits zahlreiche, gute Dashboards gibt.

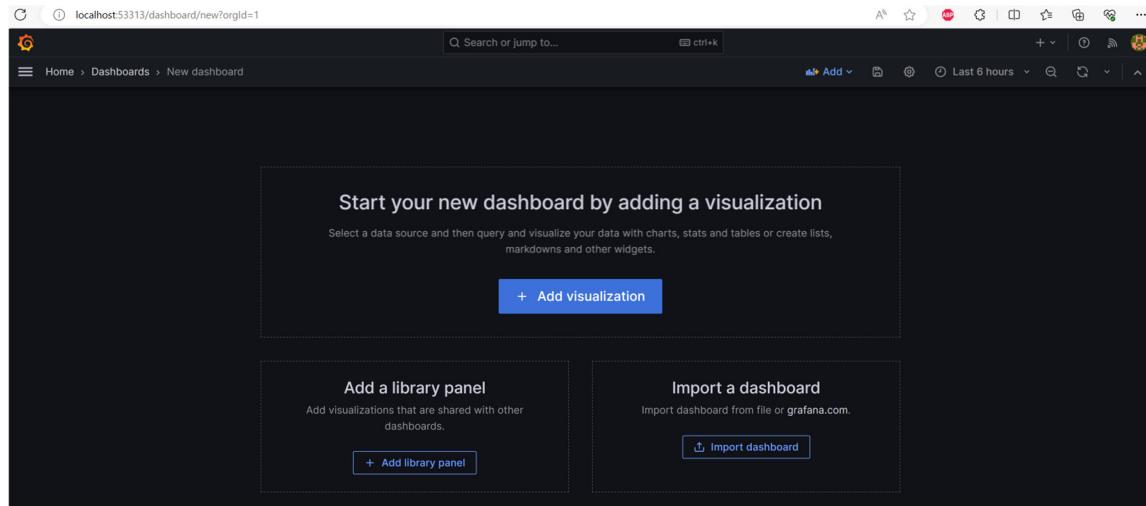


Abbildung 73 Grafana Dashboard importieren

Ein entsprechende Dashboard-ID kann unter [Dashboards | Grafana Labs](#) herausgesucht, kopiert und anschliessend bei Grafana eingefügt werden. In diesem Fall benutzen wir 1840.

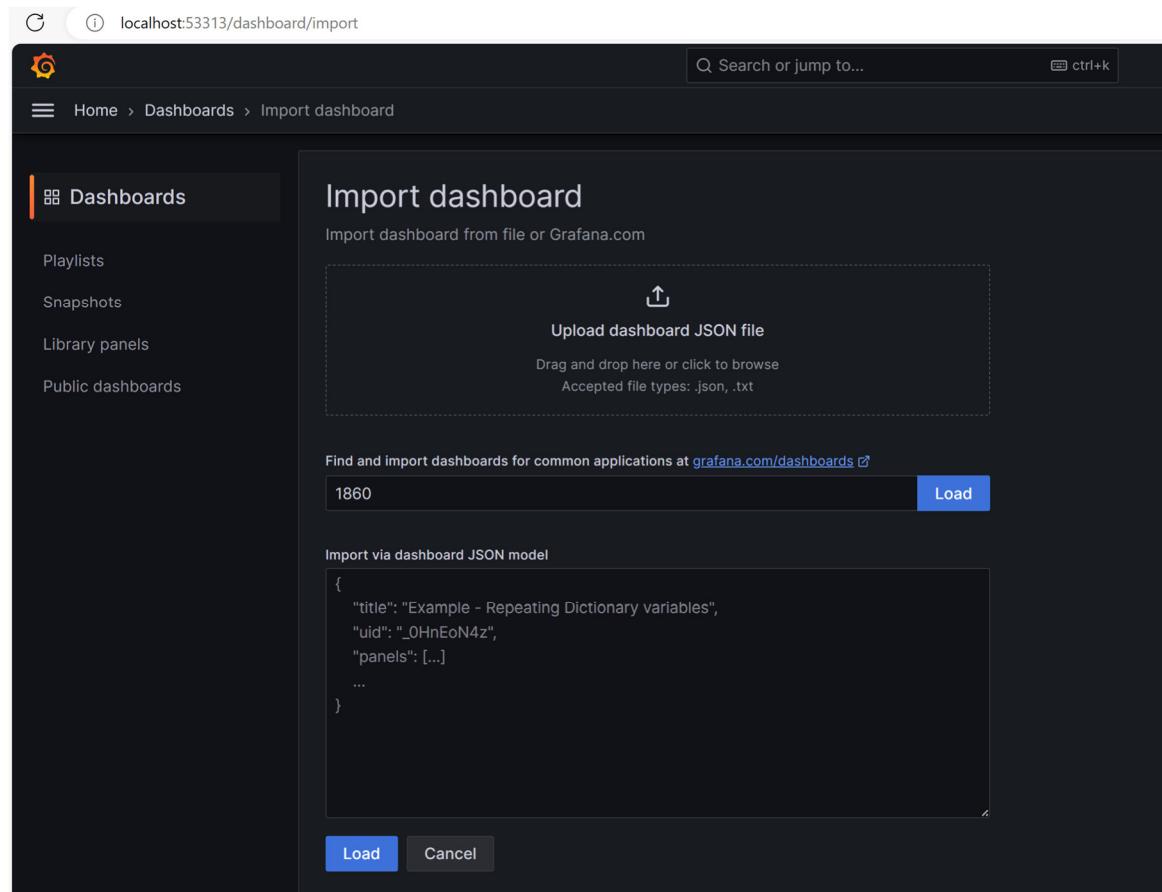


Abbildung 74 Grafana Dashboard-ID einfügen

Wenn Namen und **Quelle: Prometheus** gewählt wurden, steht das Dashboard zur Verfügung.

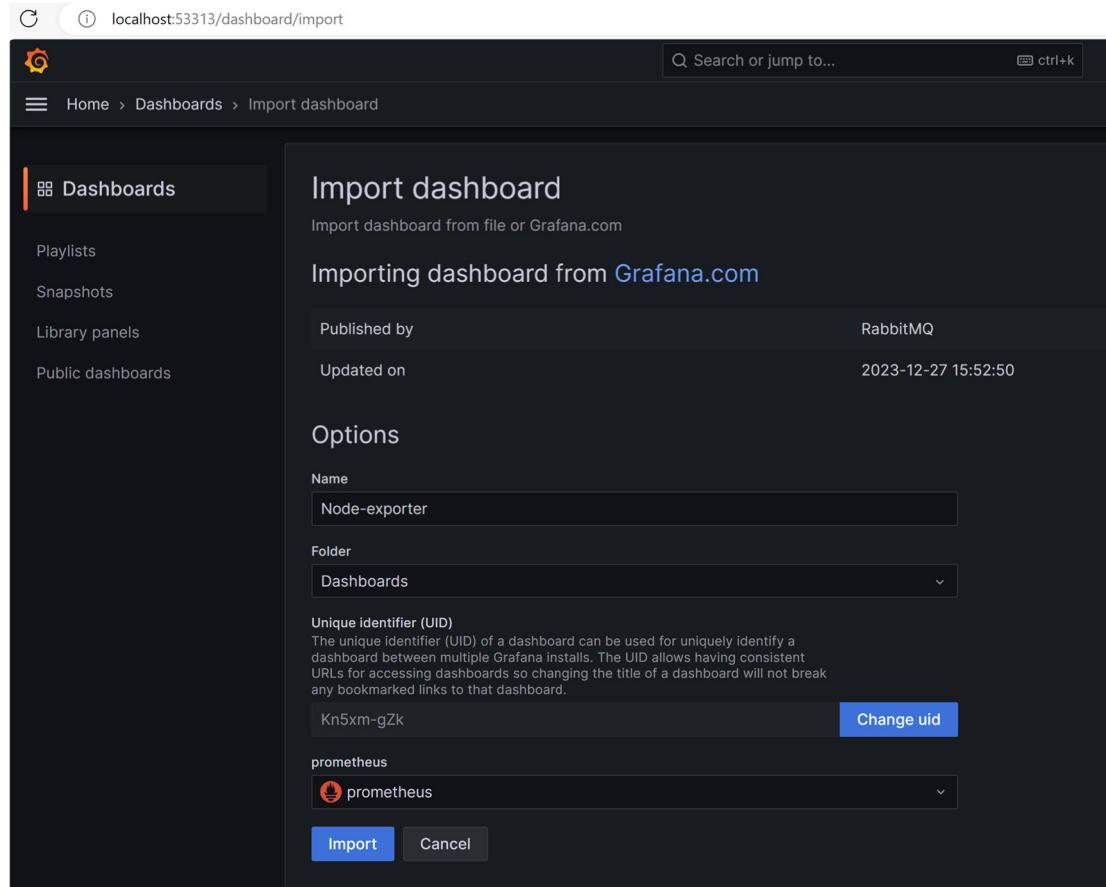


Abbildung 75 Grafana Dashboard hinzufügen Quelle

Auf dem Dashboard sind dann viele Grafiken für die visuelle Auswertung vorhanden.

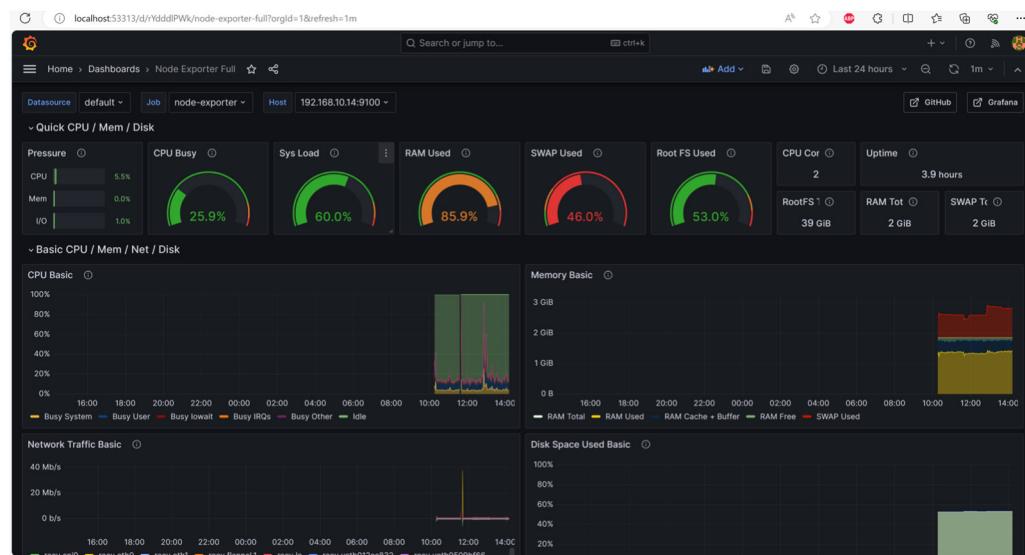


Abbildung 76 Grafana Node-Exporter Dashboard

Die Laufzeit der Geräte können wir in dem Fall ziemlich einfach gegenprüfen z.B. in Hyper-V. Dort sehen wir, dass die Uptime perfekt stimmt und die Werte somit korrekt ausgelesen werden.

	k3s-master-4	Wird ausgeführt	2%	2048 MB	04:01:15		11.0
	k3s-worker-5	Wird ausgeführt	0%	2048 MB	04:01:14		11.0
	k3s-worker-6	Wird ausgeführt	11%	2048 MB	04:01:14		11.0

Abbildung 77 Hyper-V Uptime

Auch andere Werte wie RAM und Prozessoranzahl stimmen überein. Somit funktioniert das einwandfrei.

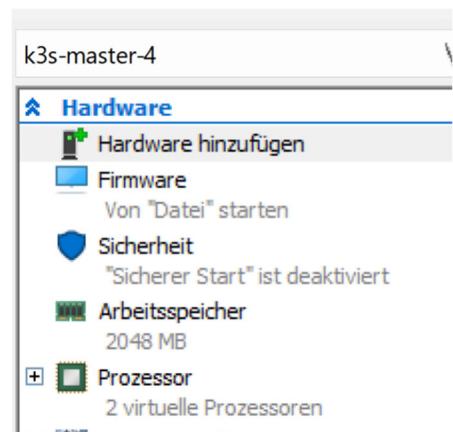


Abbildung 78 k3s-master-4 System-Ressourcen

5. Test

Die Applikation funktioniert. Die Services laufen und die Darstellung via Monitoring ist vollendet. Das Ziel ist es, ein hochverfügbares Cluster aufzubauen. Diese Hochverfügbarkeit wird nun auf die Probe gestellt. Mit den unterschiedlichen Tests werden die einzelnen Funktionen geprüft.

5.1 Knotenausfall

Wenn ein Node aus dem Cluster ausfällt, sollten die darauf laufenden Applikationen in einem anderen Node gestartet werden. Der Worker Node müsste 1 zu 1 ersetzt werden, ohne dass der Anwender einen Ausfall mitbekommt.

Resultat Knotenausfall

Festzuhalten ist, dass es 5min gedauert hat, ehe die Pods neu erstellt wurden. Dies ist aber ein erwartetes Verhalten, da beim Knotenausfall erst nach 5 Minuten die Pods neu erstellt werden durch den k3s Master. Dieses Verhalten könnte nur mit einem Replication-Controller korrigiert werden.

Danach gibt es zwei verschiedene Möglichkeiten. Wird der Node gekillt, welcher den MySQL Pod hostet, dann funktioniert es nicht. Der gelöschte Pod wartet auf den Node und hält an seinem PVC fest. Der neue MySQL Pod kann nicht mit demselben PVC gestartet werden.

Wird der Node gekillt, welche normale Pods hat, dann werden diese nach 5 Minuten korrekt hochgefahren auf einem anderen Node, wie auf dem Screenshot ersichtlich ist, erstellt.

Pods		Namespace: exampleapp										
	Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮
□	mysql-deployment-b656988b5-r9gqx	exampleapp	■	0.002	452.7MiB	0	ReplicaSet	k3s-worker-6	BestEffort	13m	Running	⋮
□	result-server-deployment-75795bf89d-2l5dw	exampleapp	■	0.002	20.2MiB	0	ReplicaSet	k3s-worker-6	BestEffort	13m	Running	⋮
□	result-server-deployment-75795bf89d-8zd95	exampleapp	■	0.000	0	0	ReplicaSet	k3s-worker-5	BestEffort	13m	Terminatir	⋮
□	result-server-deployment-75795bf89d-f49mv	exampleapp	■	0.001	19.5MiB	0	ReplicaSet	k3s-master-4	BestEffort	5m55s	Running	⋮
□	web-server-deployment-84b7c56f95-2qjc8	exampleapp	■	0.001	19.2MiB	0	ReplicaSet	k3s-worker-6	BestEffort	13m	Running	⋮
□	web-server-deployment-84b7c56f95-vkpn9	exampleapp	■	0.001	20.0MiB	0	ReplicaSet	k3s-master-4	BestEffort	5m54s	Running	⋮
□	web-server-deployment-84b7c56f95-wdrnw	exampleapp	■	0.000	0	0	ReplicaSet	k3s-worker-5	BestEffort	13m	Terminatir	⋮

Abbildung 79 Test Node Ausfall Pod neugestartet

5.2 Masterausfall

Bei dem Masterausfall ist die Angelegenheit etwas schwieriger. Der Master ist das Herzstück des Clusters. Sobald der Master ausfällt, sollte automatisch ein neuer Master als Ersatz selektiert werden.

Resultat Masterausfall

Nachdem der Master Node ausgefallen ist, wird kein neuer Master selektiert. Dies ist das Standardverhalten, sofern dies nicht weiter konfiguriert ist. Zusätzlich würde es mehrere Master benötigen, um ein solches Szenario abzudecken.

```
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> kubectl get nodes
NAME      STATUS    ROLES     AGE   VERSION
k3s-master-4   Ready    control-plane,etcd,master   24d   v1.28.6+k3s2
k3s-worker-5   Ready    <none>    17d   v1.28.6+k3s2
k3s-worker-6   Ready    <none>    24d   v1.28.6+k3s2
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> kubectl get nodes
E0316 16:45:57.312470  30648 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
E0316 16:46:18.377199  30648 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
E0316 16:46:39.410489  30648 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
E0316 16:47:00.449828  30648 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
E0316 16:47:21.493735  30648 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
Unable to connect to the server: dial tcp 192.168.10.30:6443: connectex: A connection attempt failed because the connected party did not properly respond after a period of time
, or established connection failed because connected host has failed to respond.
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> kubectl cluster-info
E0316 16:49:43.678634  24172 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
E0316 16:50:04.796188  24172 memcache.go:265] couldn't get current server API group list: Get "https://192.168.10.30:6443/api?timeout=32s": dial tcp 192.168.10.30:6443: connec
tex: A connection attempt failed because the connected party did not properly respond after a period of time, or established connection failed because connected host has failed
to respond.
PS C:\Users\mario\OneDrive - TEKO Schweizerische Fachschule AG\TEKO\Microservices\K3S> 
```

Abbildung 80 Test Master Ausfall

5.3 Pod Ausfall

Falls ein Pod auf einer Node ausfällt, sollte fast zeitgleich ein neuer Pod starten, vorzugsweise auf einer anderen Node.

Resultat Pod Ausfall

Einer der web-server Pods wurde gekillt. Kurz danach wird ein neuer Pod gestartet, welcher problemlos läuft.

Pods											Namespace: exampleapp		
	Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮	
□	mysql-deployment-b656988b5-kglsh	exampleapp	■	0.001	452.3MiB	0	ReplicaSet	k3s-worker-6	BestEffort	3m19s	Running	⋮	
□	result-server-deployment-75795bf89d-dhw9	exampleapp	■	0.000	19.3MiB	0	ReplicaSet	k3s-worker-6	BestEffort	12m	Running	⋮	
□	result-server-deployment-75795bf89d-jt2h5	result-server-deployment-75795bf89d-jt2h5		0.000	20.8MiB	0	ReplicaSet	k3s-master-4	BestEffort	12m	Running	⋮	
□	web-server-deployment-84b7c56f95-dfmzb	exampleapp	■	0.000	19.6MiB	0	ReplicaSet	k3s-worker-6	BestEffort	12m	Terminatin	⋮	
□	web-server-deployment-84b7c56f95-p642g	exampleapp	■	0.001	26.9MiB	0	ReplicaSet	k3s-worker-5	BestEffort	12m	Running	⋮	
□	web-server-deployment-84b7c56f95-xj5j7	exampleapp		0.000	0	0	ReplicaSet		BestEffort	78s	Pending	⋮	

Abbildung 81 Test Pod Ausfall Pod gekillt

Pods											Namespace: exampleapp		
	Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮	
□	mysql-deployment-b656988b5-kglsh	exampleapp	■	0.001	452.3MiB	0	ReplicaSet	k3s-worker-6	BestEffort	4m27s	Running	⋮	
□	result-server-deployment-75795bf89d-dhw9	exampleapp	■	0.001	19.1MiB	0	ReplicaSet	k3s-worker-6	BestEffort	13m	Running	⋮	
□	result-server-deployment-75795bf89d-jt2h5	exampleapp	■	0.001	22.6MiB	0	ReplicaSet	k3s-master-4	BestEffort	13m	Running	⋮	
□	web-server-deployment-84b7c56f95-p642g	exampleapp	■	0.001	26.9MiB	0	ReplicaSet	k3s-worker-5	BestEffort	13m	Running	⋮	
□	web-server-deployment-84b7c56f95-xj5j7	exampleapp	■	0.001	21.9MiB	0	ReplicaSet	k3s-worker-6	BestEffort	2m26s	Running	⋮	

Abbildung 82 Test Pod Ausfall Pod gestartet

5.4 Datenintegrität

Der MySQL Pod wird via PVC ausfallsicher bereitgestellt. Der Pod wird bei einem Ausfall mit dem gleichen Volume auf einem anderen Knoten übertragen. Die MySQL Datenbank sollte weiterhin die Daten enthalten.

Resultat Datenintegrität

Nach dem Löschen des mysql Pods wird dieser neu hochgefahren. Anschliessend holt er sich via PVC dasselbe PV. Die Daten des Result-Servers sind beständig und geben das gleiche Resultat für Andrin zurück.

Project : Phonebook Application

Welcome to SwitzerChees's Phonebook Application

Enter a name to search

Name:

Keyword

Search

Result for 'andrin'

- Name: **Andrin Germann** ---- Phone Number: **9834788737**

*This app is developed in Python by **SwitzerChees** and deployed with Flask on Kubernetes using Microservice Architechture*

Abbildung 83 Test Datenintegrität Phonebook

Nun wird der Pod gekillt.

Pods												Namespace: exampleapp		▼	🔍	⬇
	Name	⚠	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮			
□	mysql-deployment-b656988b5-92wz9	⚠	exampleapp	■	0.001	457.4MiB	0	ReplicaSet	k3s-worker-6	BestEffort	17m	Terminatir	⋮			
□	mysql-deployment-b656988b5-lsg9q	⚠	exampleapp	■	0.000	0	0	ReplicaSet	k3s-worker-6	BestEffort	78s	Pending	⋮			
□	result-server-deployment-75795bf89d-dhws9	⚠	exampleapp	■	0.001	19.1MiB	0	ReplicaSet	k3s-worker-6	BestEffort	8m12s	Running	⋮			
□	result-server-deployment-75795bf89d-jt2h5	⚠	exampleapp	■	0.001	20.7MiB	0	ReplicaSet	k3s-master-4	BestEffort	8m11s	Running	⋮			
□	web-server-deployment-84b7c56f95-dfmzb	⚠	exampleapp	■	0.001	22.4MiB	0	ReplicaSet	k3s-worker-6	BestEffort	8m12s	Running	⋮			
□	web-server-deployment-84b7c56f95-p642g	⚠	exampleapp	■	0.001	26.9MiB	0	ReplicaSet	k3s-worker-5	BestEffort	8m11s	Running	⋮			

Abbildung 84 Test Datenintegrität Pod gekillt

Der Pod wurde neu erstellt und läuft ohne Probleme.

Pods											Namespace: exampleapp			Search	Download
	Name	Namespace	Contai...	CPU	Memory	Restarts	Controlled By	Node	QoS	Age	Status	⋮			
□	mysql-deployment-b656988b5-lsg9q	exampleapp	■	0.000	0	0	ReplicaSet	k3s-worker-6	BestEffort	100s	Running	⋮			
□	result-server-deployment-75795bf89d-dhw9	exampleapp	■	0.001	19.1MiB	0	ReplicaSet	k3s-worker-6	BestEffort	8m34s	Running	⋮			
□	result-server-deployment-75795bf89d-jt2h5	exampleapp	■	0.001	19.5MiB	0	ReplicaSet	k3s-master-4	BestEffort	8m33s	Running	⋮			
□	web-server-deployment-84b7c56f95-dfmzb	exampleapp	■	0.001	22.4MiB	0	ReplicaSet	k3s-worker-6	BestEffort	8m34s	Running	⋮			
□	web-server-deployment-84b7c56f95-p642g	exampleapp	■	0.000	26.9MiB	0	ReplicaSet	k3s-worker-5	BestEffort	8m33s	Running	⋮			

Abbildung 85 Test Datenintegrität Pod gestartet

Im Anschluss kann nach Germann gesucht werden und der Eintrag ist weiterhin vorhanden.

Project : Phonebook Application

Welcome to SwitzerChees's Phonebook Application

Enter a name to search

Name:

Keyword

Search

Result for 'germann'

- Name: Andrin Germann ---- Phone Number: 9834788737

This app is developed in Python by [SwitzerChees](#) and deployed with Flask on Kubernetes using Microservice Architechture

Abbildung 86 Test Datenintegrität Resultat Suche

Abbildung 1 Zeichnung V-Switch.....	2
Abbildung 2 Hyper-V virtueller Switch.....	3
Abbildung 3 V-Switch IP Range	3
Abbildung 4 Multipass Nodes launch.....	4
Abbildung 5 Multipass Shell.....	4
Abbildung 7 Sudo nano Netplan.....	5
Abbildung 6 Netplan Master	5
Abbildung 8 Netplan apply	5
Abbildung 9 Netplan Worker.....	5
Abbildung 10 K3S Master Installation	6
Abbildung 11 Token Master.....	6
Abbildung 12 Token Master.....	6
Abbildung 13 K3S Workers Installation.....	6
Abbildung 14 EXPORT Kube VIP.....	7
Abbildung 15 RBAC-Ressourcen.....	7
Abbildung 16 RBAC-Ressourcen apply	7
Abbildung 17 Daemonset-Manifest.....	7
Abbildung 18 Manifest öffnen	8
Abbildung 19 Kube VIP yaml	8
Abbildung 20 Kube VIP Manifest apply	8
Abbildung 21 Choco Admin.....	9
Abbildung 22 Choco Download.....	9
Abbildung 23 Kubectx Installieren.....	10
Abbildung 24 K3S config Backup.....	11
Abbildung 25 K3S config angepasst.....	12
Abbildung 26 Install Helm	13
Abbildung 27 Add MetallB-Repository	13
Abbildung 28 Helm repo Update	13
Abbildung 29 MetallB Installation	14
Abbildung 30 ipaddresspool.yaml	14
Abbildung 31 l2advertisement.yaml	14
Abbildung 32 YAML-Files applizieren	14
Abbildung 33 sudo apt update für nfs.....	15
Abbildung 34 nfs common Installation.....	15
Abbildung 35 Longhorn Installation.....	16
Abbildung 36 Dockerfile erstellen.....	17
Abbildung 37 Dockerfile Image erstellen.....	18
Abbildung 38 Image auf dockerhub publizieren	18
Abbildung 39 Images auf dockerhub publiziert	18

Abbildung 40 Deployments Übersicht.....	19
Abbildung 41 result-server-deployment Teil 1.....	19
Abbildung 42 result-server-deployment Teil 2.....	20
Abbildung 43 web-server-deployment Teil 1	20
Abbildung 44 web-server-deployment Teil 2.....	21
Abbildung 45 mysql-deployment	21
Abbildung 46 Port-forward in Lens.....	22
Abbildung 47 Web-Server Frontend.....	22
Abbildung 48 Result-Server Frontend.....	23
Abbildung 49 Übersicht Services.....	24
Abbildung 50 result-server-service	24
Abbildung 51 web-server-service.....	24
Abbildung 52 mysql-service.....	25
Abbildung 53 pvc's.....	26
Abbildung 54 pv	26
Abbildung 55 mysql-pvc.....	26
Abbildung 56 result-server-ingress.....	27
Abbildung 57 web-server-ingress.....	28
Abbildung 58 Linux hosts file Anpassung.....	28
Abbildung 59 web-server.meta	29
Abbildung 60 result-server.meta.....	29
Abbildung 61 nginx ingress pods	31
Abbildung 62 Ordner kube-prometheus.....	32
Abbildung 63 kube-prometheus values.yml	32
Abbildung 64 Installation Prometheus Teil 1.....	32
Abbildung 65 Installation Prometheus Teil 2.....	33
Abbildung 66 Prometheus Übersicht.....	33
Abbildung 67 Ordner Grafana	34
Abbildung 68 Grafana yvalues.yml	34
Abbildung 69 Installation Grafana	34
Abbildung 70 Lens Metriken auslesen.....	35
Abbildung 71 Grafana Welcome Site.....	35
Abbildung 72 Grafana New Dashboard hinzufügen.....	35
Abbildung 73 Grafana Dashboard importieren.....	36
Abbildung 74 Grafana Dashboard-ID einfügen	36
Abbildung 75 Grafana Dashboard hinzufügen Quelle.....	37
Abbildung 76 Grafana Node-Exporter Dashboard.....	37
Abbildung 77 Hyper-V Uptime	38
Abbildung 78 k3s-master-4 System-Ressourcen.....	38

Abbildung 79 Test Node Ausfall Pod neugestartet	39
Abbildung 80 Test Master Ausfall.....	40
Abbildung 81 Test Pod Ausfall Pod gekillt.....	41
Abbildung 82Test Pod Ausfall Pod gestartet.....	41
Abbildung 83 Test Datenintegrität Phonebook	42
Abbildung 84 Test Datenintegrität Pod gekillt	42
Abbildung 85 Test Datenintegrität Pod gestartet	43
Abbildung 86 Test Datenintegrität Resultat Suche	43