

Cloud Computing Infrastructure CCIN 2(2)



Skriptum

5. JG / Schuljahr 2023/2024
HTL-Dornbirn



¹ https://commons.wikimedia.org/wiki/File:Kubernetes_%28container_engine%29.png

Inhaltsverzeichnis

1	Kubernetes Grundlagen	3
1.1	Der Bedarf	3
1.2	Containertechnologien	6
1.2.1	Docker	7
1.2.2	Reverse Proxy	8
2	CCIN_5JG_1_03_nodeJS_docker_traefik	9
2.3	Kubernetes Ursprünge/Aufbau	12
2.3.1	Architektur	13
2.3.2	Kubernetes-Deployment	14
2.3.3	Installation microk8s	15
3	Pods: Container in Kubernetes	16
3.1	Grundlagen zu Pods	16
3.2	Pods und YAML	16
3.3	Labels	16
3.4	Namespaces	16
4	Replikationskontrollen	16
4.1	Liveness probe	16
4.2	Funktionsweise der Replikationskontrollen	16
4.3	Jobs und Cron-Jobs	16
5	Kubernetes Volumes	16
5.1	Arten von Volumes	16
5.2	Persistente Volumes and Claims	16
6	Konfiguration und Secrets	17
6.1	Konfiguration Allgemein	17
6.2	Container-Konfigurationen	17
6.3	Sensible Daten / Secrets	17
7	Use Cases	17
7.1	Startup – Reverse Proxy	17
7.2	NodeJS in Kubernetes	17
7.3	NodeJS + MongoDB in Kubernetes	17
8	Anhang	18

8.1	Hinweise zum Skriptum	18
8.2	SSH Zugriff auf Jump-Server (Schüler)	18
8.2.1	Zugriffs-Schema außerhalb der Schule (via Port 11205).....	18
8.2.2	Zugriffs-Schema innerhalb der Schule (via Port 22).....	18
8.2.3	ProxyJump via Jump-Server	19
8.2.3.1	Hinterlegung eines Public Keys beim User auf der Labormaschine	19
8.2.3.2	Zugriff via ssh:.....	21
8.2.4	Unterstützung für Situation von Zuhause und in der Schule	22

1 Kubernetes Grundlagen

1.1 Der Bedarf

In den letzten Jahren wurden große zusammenhängende Applikationen (=monolithische Anwendungen) vielfach durch aufgesplittete, kleinere Applikationsteile (Microservices, Nanoservices oder auch serverless-applications (ohne Serververwaltung) abgelöst. Diese Applikationsteile kommunizieren über programmiersprachenunabhängige Protokolle wie REST (synchron) oder auch asynchron wie AMQP

1.1 Der Bedarf - microservices

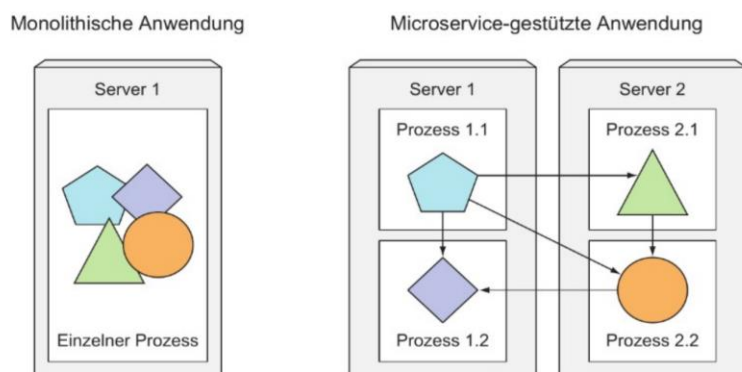
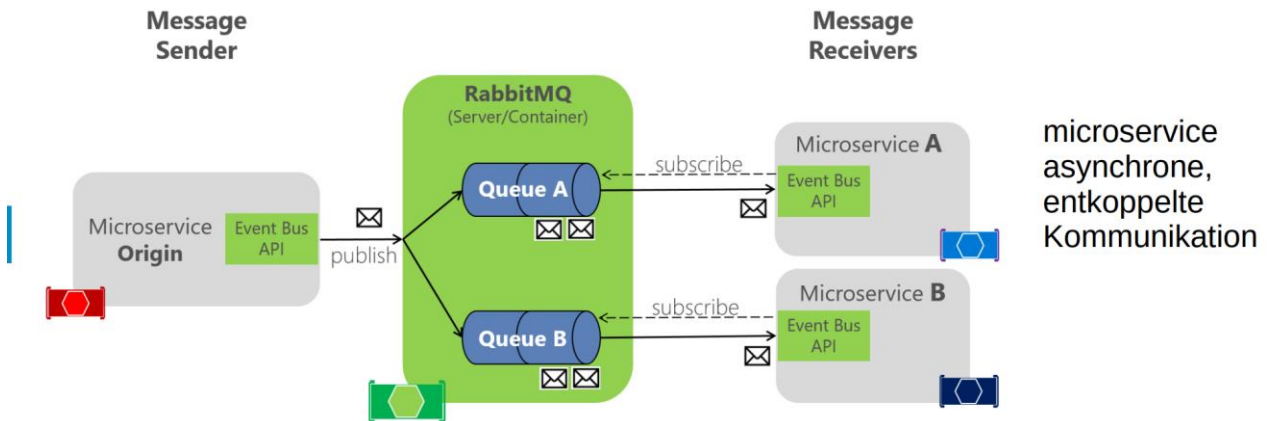
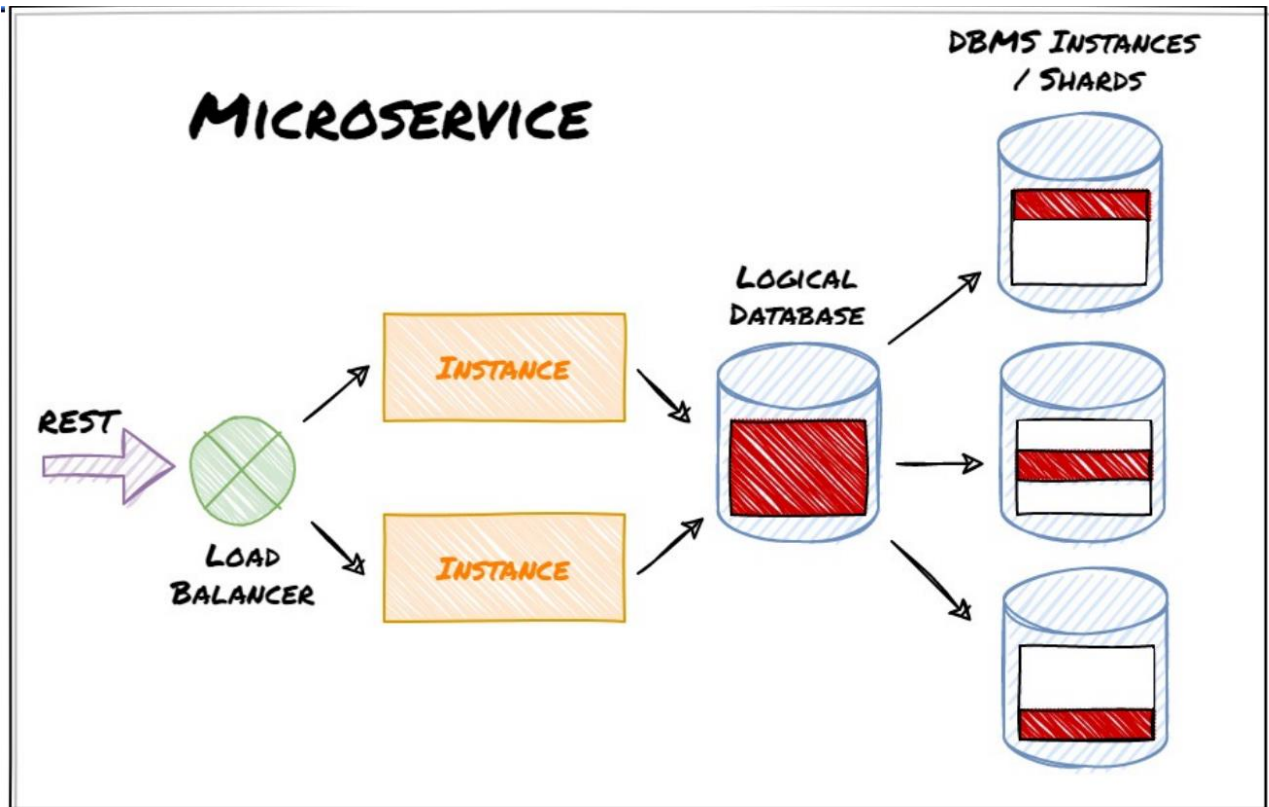


Bild 1.1 Komponenten monolithischer Anwendungen und eigenständiger Microservices im Vergleich

1.1 Der Bedarf - microservices

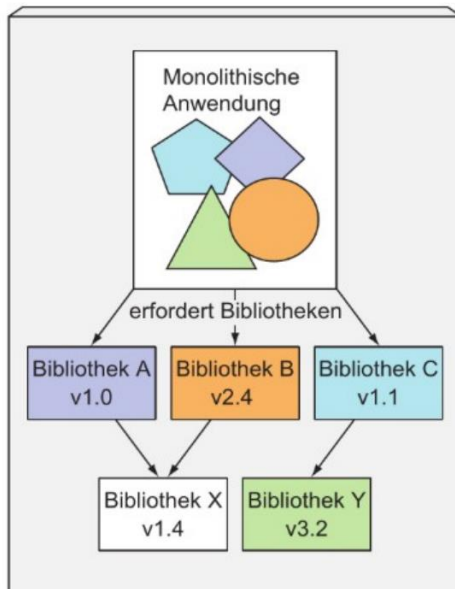


Da jeder Microservice ein eigenständiger Prozess mit einer relativ statischen externen API ist, lassen sich die einzelnen Microservices getrennt voneinander entwickeln und bereitstellen. Im Gegensatz zu monolithischen Systemen, die wir als Ganzes skalieren müssen, lassen sich Microservices einzeln skalieren.

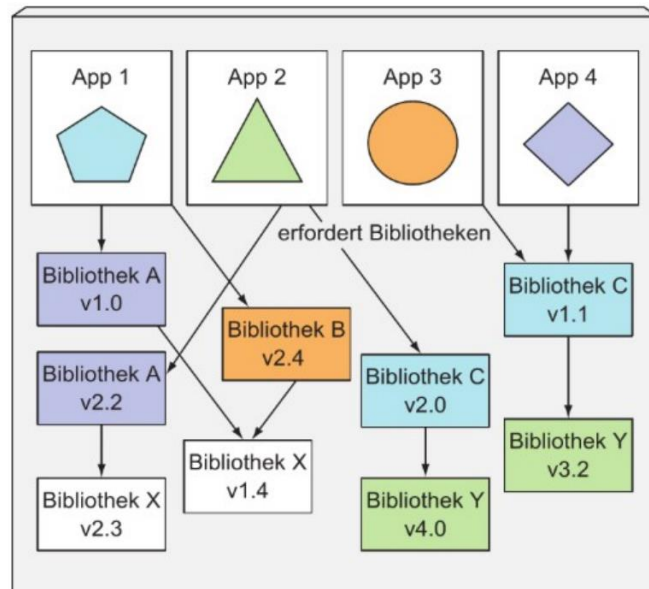


Microservices können als unabhängige Komponenten gesehen werden und auch in größeren Unternehmen von unterschiedlichen Teams entwickelt und betreut werden. Dadurch können die Services unterschiedliche Anforderungen/Abhängigkeiten aufweisen.

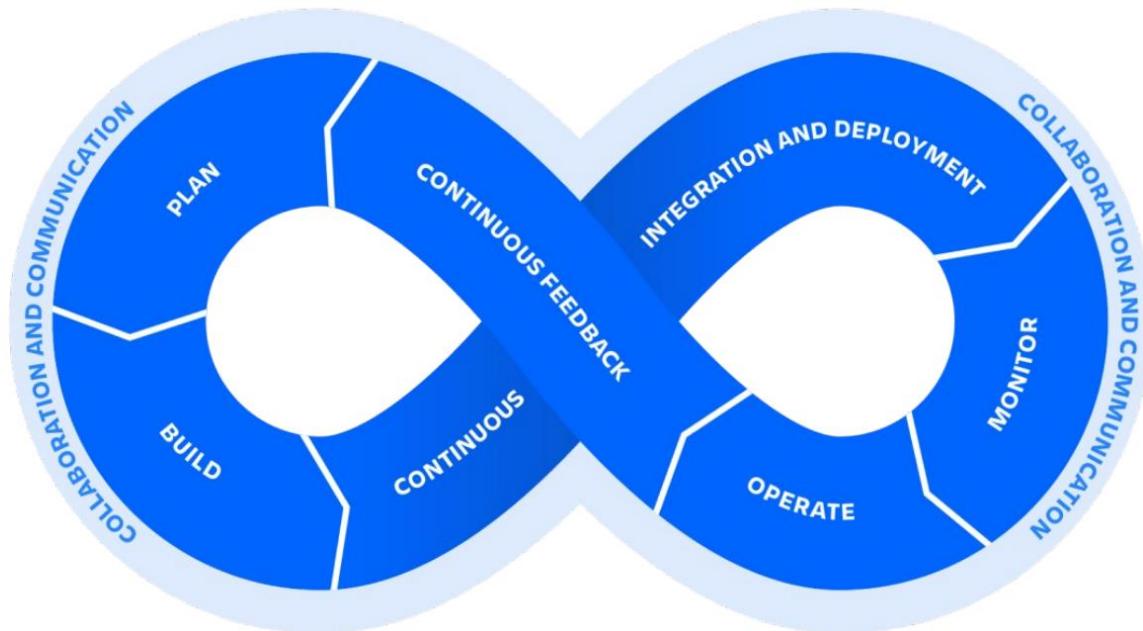
Server mit monolithischer Anwendung



Server mit mehreren Anwendungen



Unter DevOps versteht man diverse Praktiken, Tools und eine Kulturphilosophie, die die Prozesse zwischen Softwareentwicklungs- und IT-Teams automatisieren und integrieren.



Ein DevOps-Team besteht aus Entwickler- und IT-Operations-Teams, die während des gesamten Produktlebenszyklus zusammenarbeiten, um die Geschwindigkeit und Qualität des Software-Deployments zu erhöhen.

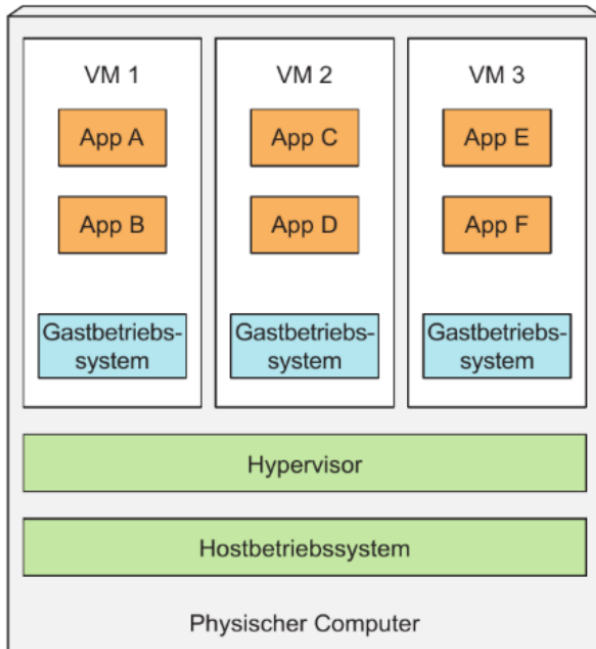
Ziel von NoOps ist es den systemadministrativen Teil gänzlich (für das Softwareentwicklungsteam) zu eliminieren. Möglichkeiten für NoOps bieten die Ansätze • Platform as a Service (PaaS mit Docker oder Kubernetes) • Function as a Service (Serverless Computing, Zustandlose Funktionen)

1.2 Containertechnologien

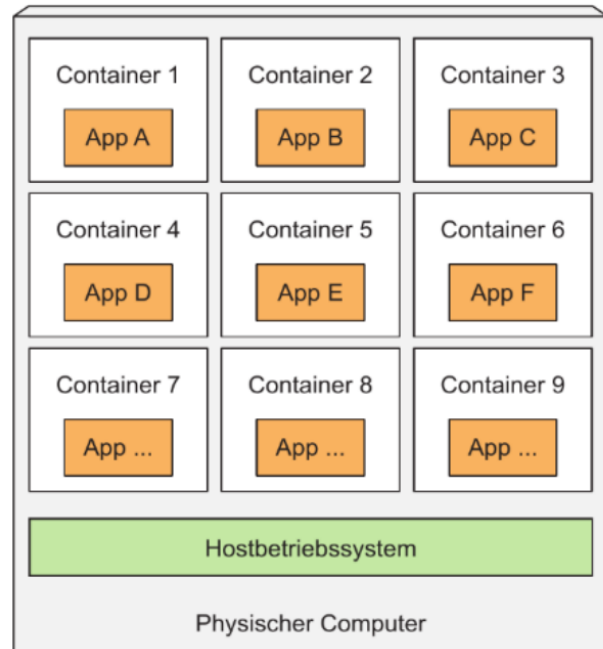
Wenn eine Anwendung nur aus einer kleinen Anzahl großer Komponenten besteht, ist es durchaus akzeptabel, jeder Komponente eine eigene virtuelle Maschine (VM) zuzuweisen und jeder eine eigene Instanz des Betriebssystems bereitzustellen, um ihre Umgebungen zu isolieren.

Linux-Containertechnologien erlauben, mehrere Dienste auf demselben Hostcomputer auszuführen. Dabei wird nicht nur jedem dieser Dienste eine eigene Umgebung bereitgestellt, sondern sie werden auch ähnlich wie bei der Verwendung von VMs voneinander isoliert, aber mit viel weniger Aufwand.

Die Anwendungen laufen in drei VMs auf demselben Rechner.



Die Anwendungen laufen in getrennten Containern.



Prinzipiell hat jedes Linux-System nur einen Standard-Namespace. Es können jedoch mehrere Namespaces eingerichtet werden. Durch Namespaces können Gruppen von Ressourcen getrennt werden.

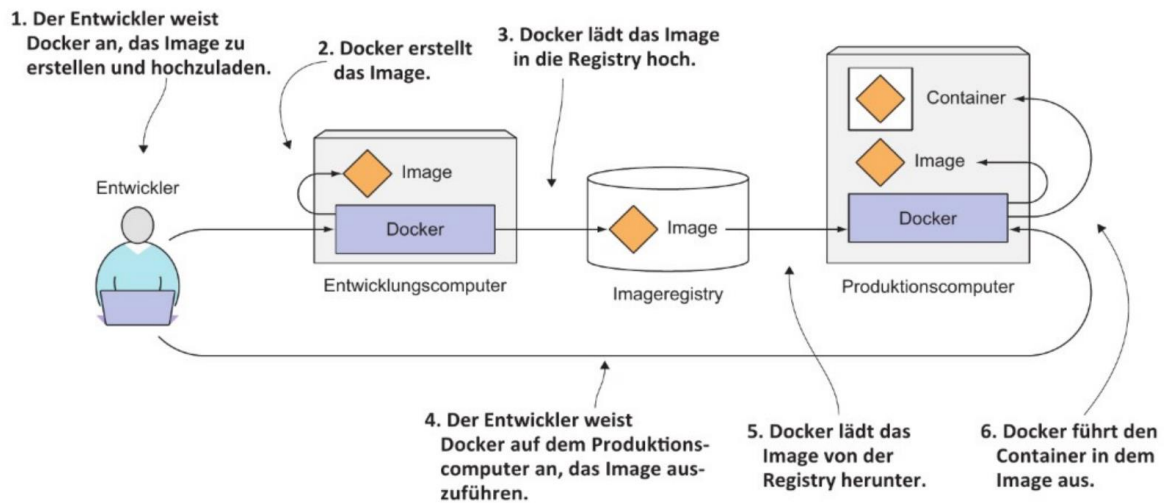
Folgende Arten von Namespaces sind möglich:

- Filesystem (Mount (mnt))
- Prozess-Id (pid)
- Netzwerk (net)
- Interprozesskommunikation (ipc)
- Hostnames (UTS) und Benutzer-ID (user)

Beim zweiten Mechanismus für die Containerisolierung geht es darum, die Menge der Systemressourcen einzuschränken, die ein Container verbrauchen kann.

1.2.1 Docker

Docker war das erste Containersystem, das nicht nur die Anwendung, sondern auch all ihre Bibliotheken und anderen Abhängigkeiten, sogar das gesamte Dateisystem des Betriebssystems, zu einem einfachen, portierbaren Paket schnüren konnte. Docker nutzt jedoch keine VMs, sondern die im vorigen Abschnitt beschriebene Linux-Containertechnologie, um (fast) den gleichen Grad an Isolation zu erreichen wie mit VMs.



Das Docker-Containerimage ist der Behälter, in den die Anwendung und deren Umgebung gepackt wird. Es enthält das Dateisystem, das der Anwendung zur Verfügung steht, sowie andere Metadaten, z. B. den Pfad zu der ausführbaren Datei, die gestartet werden soll, wenn das Image ausgeführt wird. Das Image besteht aus mehreren Schichten (pro Dockerfile-Eintrag eine Schicht). Außer der obersten Schicht sind die Schichten read-only und daher wiederverwendbar.

Ein Docker-Container ist ein laufendes Docker-Image. Ein laufender Container ist ein Prozess, der auf dem Computer mit Docker ausgeführt wird, aber vollständig vom Host und von allen anderen darauf laufenden Prozessen isoliert ist.

1.2.2 Reverse Proxy

Mit einem Reverse Proxy können Docker ServiceZugriffe an einem zentralen Punkt verwaltet werden. Dadurch kann festgelegt werden, welche Dienste in einen anderen (externen) Netzwerk verfügbar sein sollen. Eine vorgeschaltete Firewall erlaubt nur den traefik (Port-) Zugriff.

Eine docker-compose Grundlage für eine Trafik docker-compose ist die nachfolgende YAML-Konfiguration.

Docker Container:

```
marc@labor3057:~/CCIN_5JG_1_02_nodeJS_docker$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
0c68a524298c	si:1.0.0	"docker-entrypoint.s..."	6 seconds ago	Up 5 seconds	3000/tcp, 0.0.0.0:8080->8080/tcp, :::8080->8080/tcp

```
si
```

2 CCIN_5JG_1_03_nodeJS_docker_traefik

Docker compose file:

```
version: "3.3"

services:
  traefik:
    image: "traefik:v2.9"
    container_name: "traefik"
    command:
      #- "--log.level=DEBUG"
      - "--api.insecure=true"
      - "--providers.docker=true"
      # - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - "/var/run/docker.sock:/var/run/docker.sock:ro"

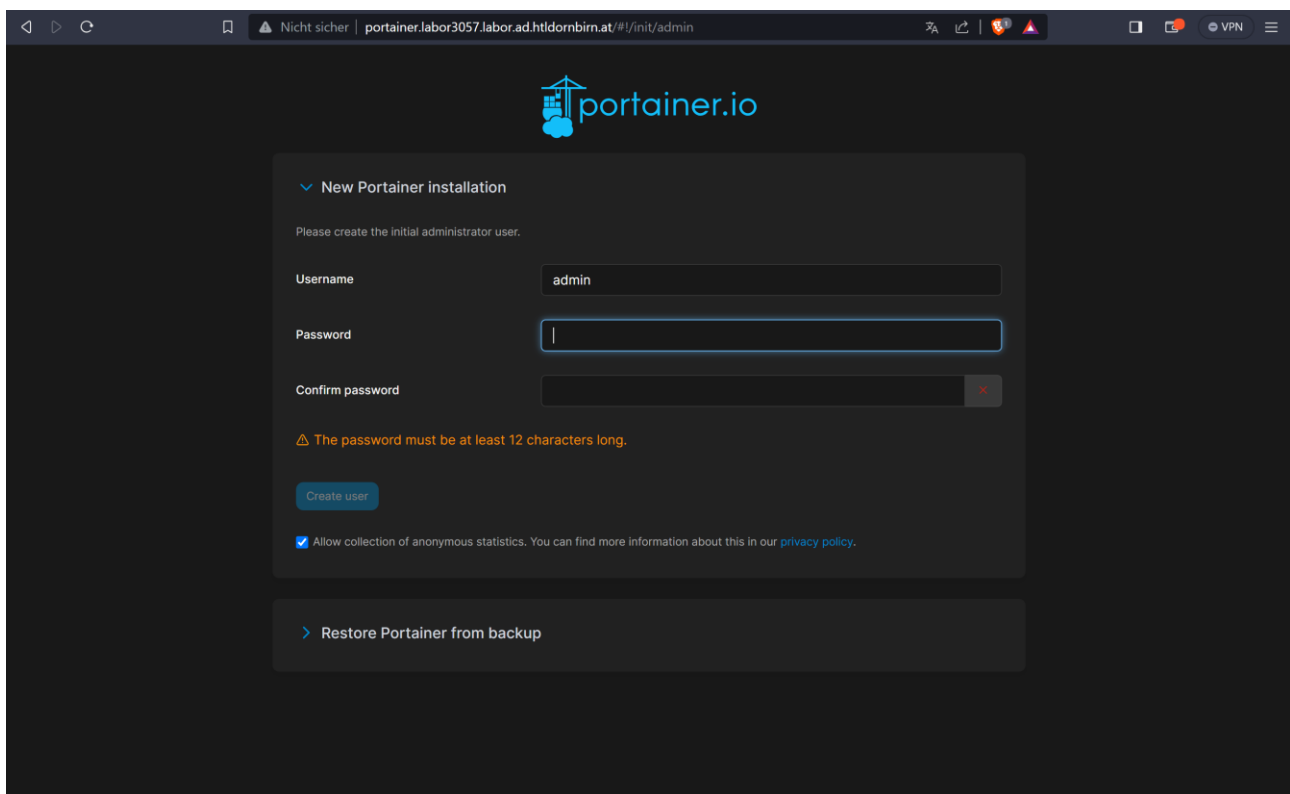
  # Container für die Info-App (NodeJS)
  info:
    image: "si:1.0.1"
    container_name: "info"
    ports:
      - "3000:3000"
    labels:
      - "traefik.http.routers.info.rule=PathPrefix(`/info`)"
      - traefik.http.routers.info.service=info
      - traefik.http.routers.info.tls=false
      - traefik.http.services.info.loadbalancer.server.port=3000

  portainer:
    image: portainer/portainer-ce:2.19.1
    container_name: portainer
    security_opt:
      - no-new-privileges:true
```

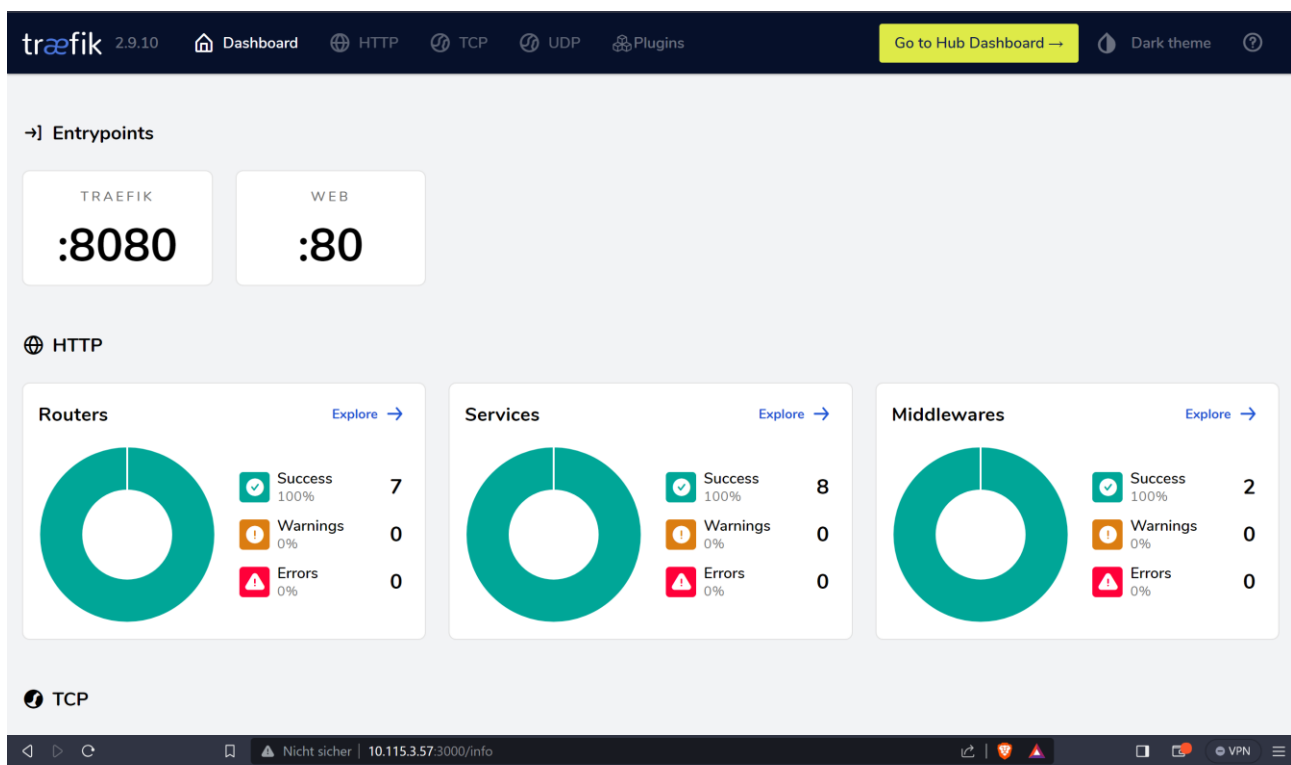
```
volumes:
  - /etc/localtime:/etc/localtime:ro
  - /var/run/docker.sock:/var/run/docker.sock:ro

ports:
  - 9000:9000

labels:
  - "traefik.http.routers.portainer.rule=HostRegexp(`por-
tainer.{catchall:.*}`)"
  - traefik.http.services.portainer.loadbalancer.server.port=9000
```



The screenshot shows the Portainer.io web interface in a browser. The address bar indicates the URL is `portainer.labor3057.labor.ad.htldornbirn.at/#/init/admin`. The page title is "Nicht sicher". The main content area features the Portainer.io logo at the top. Below the logo, there is a section titled "New Portainer installation" with a sub-header "Please create the initial administrator user." This section contains three input fields: "Username" (pre-filled with "admin"), "Password" (empty), and "Confirm password" (empty). A warning message below the password fields states: "The password must be at least 12 characters long." Below the warning is a "Create user" button. At the bottom of the form, there is a checkbox labeled "Allow collection of anonymous statistics. You can find more information about this in our [privacy policy](#)." Below the form, there is a button labeled "> Restore Portainer from backup".



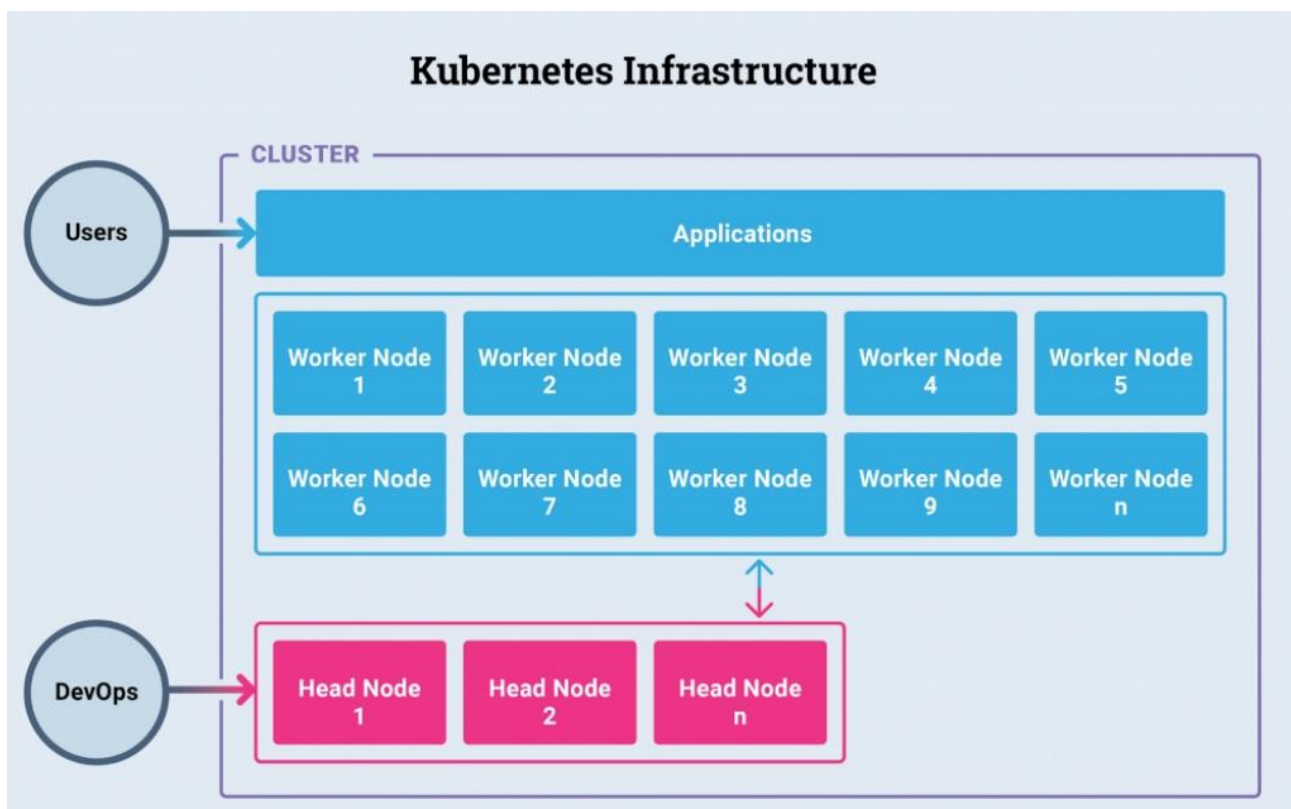
Hello World

	NAMES					
b652ea21aea1	traefik:v2.9	"/entrypoint.sh --ap..."	3 minutes ago	Up 3 minutes	0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:8080	
->8080/tcp, :::8080->8080/tcp	traefik					
84ce8179f035	si:1.0.1	"docker-entrypoint.s..."	3 minutes ago	Up 3 minutes	0.0.0.0:3000->3000/tcp, :::3000->3000/tcp	
	info					
7a61adf07918	portainer/portainer-ce:2.19.1	"/portainer"	17 minutes ago	Up 8 minutes	8000/tcp, 9443/tcp, 0.0.0.0:9000->9000/tcp, :::	
9000->9000/tcp	portainer					
a5ac8f358b42	node-web-app	"docker-entrypoint.s..."	4 weeks ago	Up 10 days	0.0.0.0:3010->3010/tcp, :::3010->3010/tcp, 8080	
/tcp	goofy_brown					
40498b76430f	mysql/mysql-server:5.7	"/entrypoint.sh mysql..."	4 weeks ago	Up 3 weeks (healthy)	0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 3306	
0/tcp	mysqlswp2					

2.3 Kubernetes Ursprünge/Aufbau

Wenn die Anzahl an Benutzern (und damit die Anzahl an Serverinstanzen) und auch die Anzahl an (Micro-) Services zunimmt, wird die Verwaltung immer aufwändiger. Um diesen Aufwand zu reduzieren, wurde von Google 2014 das OpenSource Projekt Kubernetes („K8s“) vorgestellt.

Vom Kubernetes-Ablauf können zwei Perspektiven unterschieden werden. Die Anwender_innen werden auf die Worker Nodes verteilt. Die Worker Nodes sind die „Arbeitstiere“.

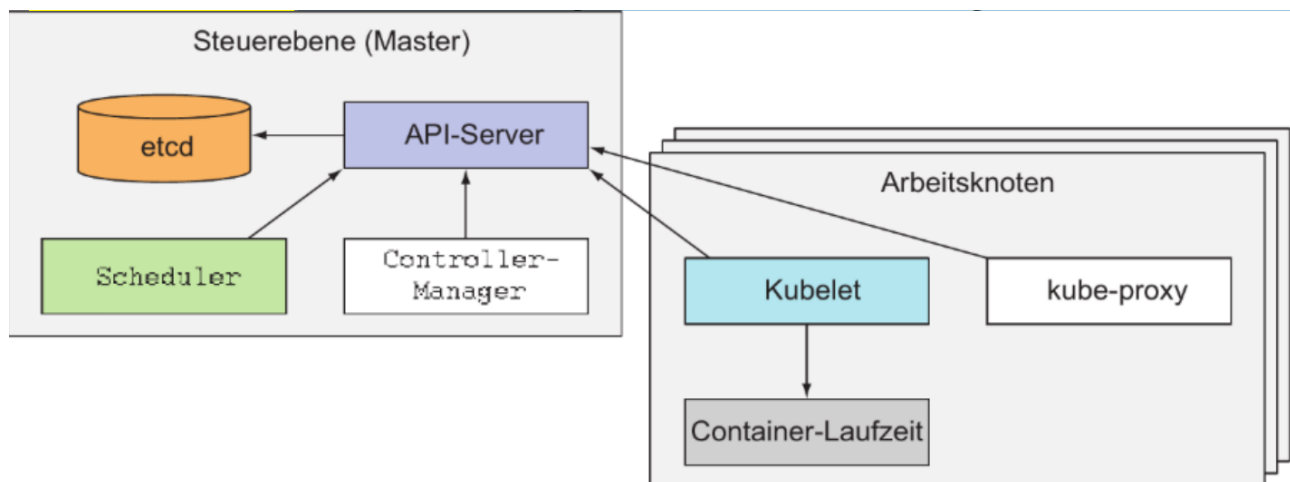


2.3.1 Architektur

Aufbau:

Der Kubernetes-Master, der die Kubernetes-Steuerebene beherbergt; sie steuert und verwaltet das gesamte Kubernetes-System

Arbeitsknoten, die die bereitgestellten Anwendungen tatsächlich ausführen



Die Steuerebene ist das, was den gesamten Cluster steuert und funktionieren lässt. Sie besteht aus mehreren Komponenten, die alle auf einem einzigen Masterknoten laufen oder auf mehrere Knoten verteilt und repliziert werden können, um für hohe Verfügbarkeit zu sorgen. Es handelt sich dabei um folgende Komponenten:

- Der **API-Server**, der dazu dient, mit dem Kubernetes-Cluster zu kommunizieren und Operationen auf ihm auszuführen

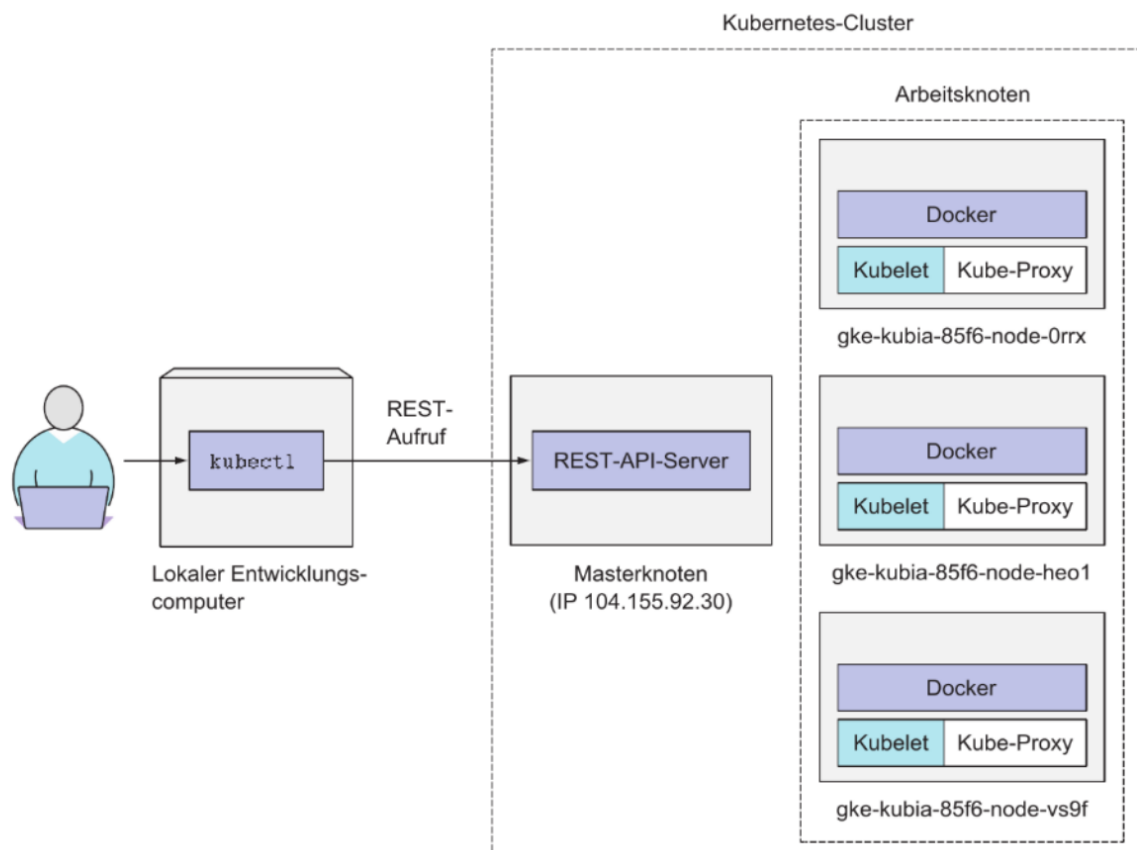
Der **Scheduler**, der für die Planung der Anwendungen zuständig ist (Zuweisung eines Arbeitsknotens zu jeder bereitstellbaren Komponente der Anwendung)

Der **Controller-Manager**, der clusterweite Funktionen ausübt, z. B. die Replikation von Komponenten, Verfolgung der Arbeitsknoten, Handhabung von Knotenausfällen usw. Der zuverlässige, verteilte Datenspeicher **etcd** (ausgesprochen „et-si-di“), in dem die gesamte Clusterkonfiguration dauerhaft in einem Key-Value-Store festgehalten ist.

Um Anwendungen in Kubernetes auszuführen, müssen diese in ein Image-Registry (Docker-Verzeichnis / Hub) geladen werden. Anschließend kann mit Hilfe von Beschreibungsdaten (yaml-Files) die Instanzierung der Anwendungen mit Hilfe des Kubernetes-Master durchgeführt werden.

2.3.2 Kubernetes-Deployment

Die Steuerung eines Kubernetes-Cluster kann erfolgen durch API-Calls mit dem Dienstprogramm `kubectl` – Dies ist der übliche Ansatzpunkt, da dadurch die Konfigurationen abgespeichert wird. Dadurch können Änderungen leicht durchgeführt werden oder es kann sehr einfach die Applikation neu deployed werden. Kubernetes Dashboard – ist für einen Überblick hilfreich. Meistens wird jedoch die Bedienung und Kontrolle auch mittels `kubectl` durchgeführt.



Das Deployment in einem Kubernetes-Cluster kann imperativ oder deklarativ erfolgen. Imperativ: Es werden einzelne Befehle mittels `kubectl` abgesetzt. Deklarativ: Dem Master/Head-Node wird mitgeteilt, was gewünscht wird.

Auftrag: Deklarativ:

```
marc@labor3057:~/CCIN_5JG_1_05_nodeJS_k8s_deklarativ$ kubectl get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/sys-info-58fcd8669c-5bmlf	1/1	Running	0	18s
pod/sys-info-58fcd8669c-p6xw9	1/1	Running	0	18s

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/kubernetes	ClusterIP	10.152.183.1	<none>	443/TCP	35d
service/sys-info	NodePort	10.152.183.36	<none>	8080:30080/TCP	5s

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
deployment.apps/sys-info	2/2	2	2	19s

NAME	DESIRED	CURRENT	READY	AGE
replicaset.apps/sys-info-58fcd8669c	2	2	2	18s

```
marc@labor3057:~/CCIN_5JG_1_05_nodeJS_k8s_deklarativ$
```

Navigation icons | Nicht sicher | 10.115.3.57:30080 | VPN

Hello

2.3.3 Installation microk8s

```
sudo snap install microk8s --edge --classic
# Start microk8s
sudo microk8s.start
#grant permissions to user
sudo usermod -a -G microk8s $USER
# Inspect the services microk8s.status
# create alias for kubectl
alias kubectl='microk8s.kubectl'
```

```
sudo microk8s.enable dns dashboard
kubectl patch --namespace kube-system service/kubernetes-dashboard -p '{"spec":
```

```
{"type": "NodePort"}}'  
kubectl get all --all-namespaces  
kubectl create token default
```

```
sudo microk8s.enable registry
```

3 Pods: Container in Kubernetes

3.1 Grundlagen zu Pods

3.2 Pods und YAML

3.3 Labels

3.4 Namespaces

4 Replikationskontroller

4.1 Liveness probe

4.2 Funktionsweise der Replikationskontroller

4.3 Jobs und Cron-Jobs

5 Kubernetes Volumes

5.1 Arten von Volumes

5.2 Persistente Volumes and Claims

6 Konfiguration und Secrets

6.1 *Konfiguration Allgemein*

6.2 *Container-Konfigurationen*

6.3 *Sensible Daten / Secrets*

7 Use Cases

7.1 *Startup – Reverse Proxy*

7.2 *NodeJS in Kubernetes*

7.3 *NodeJS + MongoDB in Kubernetes*

8 Anhang

8.1 Hinweise zum Skriptum

- Das Skriptum ist wichtiger Teil der Mitarbeit-Dokumentation und dient auch als Lerngrundlage
- Bitte das Skriptum laufend mit dem aktuellen **Theoriestoff komplettieren**.
- Darauf acht geben, dass **ALLE bisherigen Übungen** entsprechend dokumentiert sind
 - Eine Übungsdokumentation enthält zumindest Screenshots, Beschreibungen und Code-Ausschnitte
- Anmerkungen / Feedback aus letzten Beurteilungen berücksichtigen
- Inhaltsverzeichnis aktualisieren
- Form/Layout prüfen
- **Abgabe immer als PDF**

8.2 SSH Zugriff auf Jump-Server (Schüler)

Über den „Jump“-Server `itlabor3.htldornbirn.vol.at` kann sowohl innerhalb der Schule als auch außerhalb der Schule der Login auf die eigene Virtuelle Maschine erfolgen.

(Beispiel: Schüler mit Maturajahr 2024)

Voraussetzungen:

Die private Keys der jeweiligen Klasse sind im `$USER\.ssh` Verzeichnis hinterlegt:

`id_awi_2024`

`id_bwi_2024`

`id_cwi_2024`

8.2.1 Zugriffs-Schema außerhalb der Schule (via Port 11205)

```
ssh -i ~/.ssh/id_awi_2024 -p '11205' jump-awi-2024@itlabor3.htldornbirn.vol.at'
```

```
ssh -i ~/.ssh/id_awi_2024 -p '11205' 'jump-awi-2024@itlabor3.htldornbirn.vol.at'
Enter passphrase for key 'id_awi_2024':
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-58-generic x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
System information as of Mon May 24 17:53:43 UTC 2021
...
```

8.2.2 Zugriffs-Schema innerhalb der Schule (via Port 22)

```
ssh -i ~/.ssh/id_awi_2024 'jump-awi-2024@10.115.1.205'
```

8.2.3 ProxyJump via Jump-Server

Damit man bequem gleich auf den Zielserver springen kann, wird in `~/.ssh/config` folgendes eingetragen:

```
Host jump
  HostName itlabor3.htldornbirn.vol.at
  User jump-awi-2024
  Port 11205
  IdentityFile ~/.ssh/id_jump

Host Labor1205
  HostName 10.115.1.205
  User klaus
  ProxyJump jump
```

Damit kann man auf die eigene Labormaschine zugreifen, in dem man in der GitBash einfach

```
ssh Labor1205
```

eingibt. Man wird nach der `id_jump` „Passphrase“ und dem eigenen Passwort auf dem Server gefragt.

8.2.3.1 Hinterlegung eines Public Keys beim User auf der Labormaschine

Wenn man für den User ein eignes Key-Pair erzeugt und auf den Zielrechner stellt, kann der Zugriff nur durch Eingabe der Passphrase für den `id_jump` Key erfolgen:

```
cd ~/.ssh
ssh-keygen -t rsa -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/klaus/.ssh/id_rsa): id_klaus
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in id_klaus.
Your public key has been saved in id_klaus.pub.
The key fingerprint is:
SHA256:tEwor3Q+nYdyANuISsw3towjck106LIH8lbN0MELMKA klaus@DESKTOP-8OCTPRI
The key's randomart image is:
```

```
+---[RSA 4096]-----+
| ..o. .. |
| .. o.... |
| E = o +... |
| o + X + =. |
| +. +O * S o |
| .. = +oO + o |
| +o.o+ B = . |
| o.. o + . |
| |
+----[SHA256]-----+
```

klaus@DESKTOP-8OCTPRI

Der Public Key wird auf den Zielsystem übertragen: (mehrfache Eingabe der id_jump Passphrase!)

```
klaus@DESKTOP-8OCTPRI MINGW64 ~/.ssh
$ ssh-copy-id -i id_klaus.pub klaus@Labor1205
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "id_klaus.pub"
Enter passphrase for key '/c/Users/klaus/.ssh/id_jump':
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
Enter passphrase for key '/c/Users/klaus/.ssh/id_jump':
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompt
ed now it is to install the new keys
Enter passphrase for key '/c/Users/klaus/.ssh/id_jump':
klaus@10.115.1.205's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'klaus@Labor1205'"
and check to make sure that only the key(s) you wanted were added.
```

Dieser Private Key wird im `~/.ssh/config` eingetragen:

```
Host jump
```

```
HostName itlabor3.htldornbirn.vol.at
User jump-awi-2024
Port 11205
IdentityFile ~/.ssh/id_jump

Host Labor1205
  HostName 10.115.1.205
  User klaus
  IdentityFile ~/.ssh/id_klaus
  ProxyJump jump
```

~

8.2.3.2 Zugriff via ssh:

```
$ ssh klaus@Labor1205
Enter passphrase for key '/c/Users/klaus/.ssh/id_jump':
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-58-generic x86_64)

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

8.2.4 Unterstützung für Situation von Zuhause und in der Schule

Im `~/ssh/config` werden zwei Einträge für die Labormaschine gemacht. Jene mit „x“ am Ende ist die eXterne, jene ohne die für den Zugriff innerhalb der Schule auf Port 22 – die verwendet dann auch keinen Jump-Host.

```
Host jump
  HostName itlabor3.htldornbirn.vol.at
  User jump-awi-2024
  Port 11205
  IdentityFile ~/.ssh/id_jump
```

```
Host Labor1205x
  HostName 10.115.1.205
  User klaus
  IdentityFile ~/.ssh/id_klaus
  ProxyJump jump
```

```
Host Labor1205
  HostName 10.115.1.205
  User klaus
  IdentityFile ~/.ssh/id_klaus
```