

## **SEMINARARBEIT**

Im Studiengang FH Technikum Wien, Bachelorstudiengang BIF  
Lehrveranstaltung BIF-VZ-5-WS2025-INF-EN

### **Class Workshop (Docker & Portainer)**

Ausgeführt von: David Veigel

Kevin Forter

Personenkennzeichen: XXXXXXXXXXXXXXXXX

BegutachterIn:

Wien, den 16. Oktober 2025

# Inhaltsverzeichnis

<b>1 Überblick</b>	<b>1</b>
<b>2 Provisionierung der AWS-Ressourcen</b>	<b>1</b>
2.0.1 Erstellung VPC . . . . .	1
2.0.2 Erstellung Subnet . . . . .	2
2.0.3 Erstellung Sicherheitsgruppe . . . . .	2
2.0.4 EC2-Instanzen (Nodes) . . . . .	4
2.0.5 Validierung . . . . .	6
2.0.6 Docker Swarm Setup . . . . .	7
2.0.7 Portainer Installation . . . . .	9
2.0.8 Service-Deployment, Scale-Out und Scale-In . . . . .	10
2.0.9 Kurzantworten . . . . .	16
2.0.10 Anhang – Wichtige Befehle . . . . .	16
<b>3 Kubernetes Deployment Workshop mit Minikube</b>	<b>16</b>
3.0.1 Installation Minikube und Kubectl . . . . .	17
3.0.2 Start und Vorbereitung . . . . .	17
3.0.3 Deployment erstellen und skalieren . . . . .	18
3.0.4 Testzugriff auf einen Pod . . . . .	19
3.0.5 ClusterIP Service (intern) . . . . .	20
3.0.6 NodePort Service (extern) . . . . .	21
3.0.7 LoadBalancer Service (simuliert) . . . . .	22
3.0.8 Vergleich der Service-Typen . . . . .	24
3.0.9 YAML-Spezifikationen . . . . .	25
3.0.10 Zusammenfassung . . . . .	27

# 1 Überblick

Ziel: Provisionierung von AWS-Ressourcen, Aufbau eines Docker Swarm-Clusters, Installation von Portainer und Analyse von Skalierung und Ausfallszenarien.

## 2 Provisionierung der AWS-Ressourcen

### 2.0.1 Erstellung VPC

Neue VPC mit öffentlichem Subnet über den Assistenten im AWS-VPC-Dashboard erstellen. Security Group mit folgenden offenen Ports konfigurieren:

The screenshot shows the 'VPC-Einstellungen' (VPC Settings) step of the AWS VPC creation wizard. It includes fields for selecting resource type ('Nur VPC'), naming ('Name-Tag: vpc-docker-swarm'), defining a CIDR block ('IPv4-CIDR-Block: 10.0.0.0/16'), and specifying IPv6 settings. The 'Tenancy' dropdown is set to 'Standard'.

Abbildung 1: Einstellung der VPC inklusive manueller CIDR-Einstellung 10.0.0.0/16

## 2.0.2 Erstellung Subnet

**Subnetzeinstellungen**  
Geben Sie die CIDR-Blöcke und Availability Zone für das Subnetz an.

**Subnetz 1 von 1**

**Subnetz-Name**  
Ein Tag mit dem Schlüssel 'Name' und einem von Ihnen angegebenen Wert erstellen.

Der Name kann bis zu 256 Zeichen lang sein.

**Verfügbare Zone** [Info](#)  
Wählen Sie die Zone aus, in der sich Ihr Subnetz befindet, oder lassen Sie Amazon eine für Sie auswählen.

**IPv4-VPC-CIDR-Block** [Info](#)  
Wählen Sie den IPv4-CIDR-Block des VPC für das Subnetz. Der IPv4-CIDR des Subnetzes muss innerhalb dieses Blocks liegen.

**IPv4-Subnetz-CIDR-Block**  
 256 IPs  
[Neues Subnetz hinzufügen](#)

**Tags – optional**

Schlüssel	Wert – optional
<input type="text" value="Name"/>	<input type="text" value="sbn-docker-swarm"/>

[Neues Tag hinzufügen](#)  
Sie können 49 weitere Tags. hinzufügen  
[Entfernen](#)

Abbildung 2: Öffentliches Subnet mit Subnet-CIDR-Block 10.0.1.0/24

**IP-Einstellungen automatisch zuweisen** [Info](#)  
Ermöglichen Sie AWS die automatische Zuweisung einer öffentlichen IPv4- oder IPv6-Adresse an eine neue primäre Netzwerkschnittstelle für eine Instance in diesem Subnetz.

Automatische Zuweisung von öffentlichen IPv4-Adressen aktivieren [Info](#)

Kundeneigene IPv4-Adresse automatisch zuweisen erlauben [Info](#)  
Option deaktiviert, da keine kundeneigenen Pools gefunden wurden.

Abbildung 3: Automatische Vergabe der öffentlichen IPv4-Adresse aktiviert

## 2.0.3 Erstellung Sicherheitsgruppe

- 22/TCP – SSH (öffentlich)
- 80/TCP – HTTP (öffentlich)
- 2377/TCP, 7946/TCP/UDP, 4789/UDP – innerhalb der SG

Die Sicherheitsgruppe musste in zwei Schritten erstellt werden, damit sie den Vorgaben entspricht.

Schritt 1: Zuerst musste die Sicherheitsgruppe nur mit SSH und HTTP erstellt werden

**Sicherheitsgruppe erstellen** Informationen

Eine Sicherheitsgruppe fungiert als virtuelle Firewall für Ihre Instance, um den ein- und ausgehenden Datenverkehr zu steuern. Füllen Sie die nachstehenden Felder aus, um eine neue Sicherheitsgruppe zu erstellen.

**Grundlegende Details**

Name der Sicherheitsgruppe Informationen  
 Der Name kann nach der Erstellung nicht bearbeitet werden.

Beschreibung Informationen

VPC Informationen

**Regeln für eingehenden Datenverkehr** Informationen

Typ	Informationen	Protokoll	Portbereich	Quelle	Informationen	Beschreibung – optional	Informationen
SSH	▼	TCP	22	Anyw...	▼	Remote access	<input type="button" value="Löschen"/>
HTTP	▼	TCP	80	Anyw...	▼	Web traffic	<input type="button" value="Löschen"/>

Abbildung 4: Erstellung der Sicherheitsgruppe mit SSH und HTTP

Schritt 2: Erst nach dem Erstellen der Sicherheitsgruppe konnten wir für die Docker-Regeln die eigene Sicherheitsgruppe als Quelle angeben.

**Regeln für eingehenden Datenverkehr bearbeiten** Informationen

Regeln für eingehenden Datenverkehr steuern den eingehenden Datenverkehr, der die Instance erreichen darf.

ID der Sicherheitsgruppenregel	Typ	Informationen	Protokoll	Portbereich	Quelle	Informationen	Beschreibung – optional	Informationen
sgr-0d805d492a7dd8acf	HTTP	▼	TCP	80	Benut...	▼	<input type="text" value="0.0.0.0/0"/> <input type="button" value="X"/>	Web traffic <input type="button" value="Löschen"/>
sgr-091cdd67677296eb8	SSH	▼	TCP	22	Benut...	▼	<input type="text" value="0.0.0.0/0"/> <input type="button" value="X"/>	Remote access <input type="button" value="Löschen"/>
-	Benutzerdefiniertes TCP	▼	TCP	2377	Benut...	▼	<input type="text" value="sg-01c270aa28fe01d44"/> <input type="button" value="X"/>	Manager communication <input type="button" value="Löschen"/>
-	Benutzerdefiniertes TCP	▼	TCP	7946	Benut...	▼	<input type="text" value="sg-01c270aa28fe01d44"/> <input type="button" value="X"/>	Node communication <input type="button" value="Löschen"/>
-	Benutzerdefiniertes UDP	▼	UDP	7946	Benut...	▼	<input type="text" value="sg-01c270aa28fe01d44"/> <input type="button" value="X"/>	Node communication <input type="button" value="Löschen"/>
-	Benutzerdefiniertes UDP	▼	UDP	4789	Benut...	▼	<input type="text" value="sg-01c270aa28fe01d44"/> <input type="button" value="X"/>	Container overlay network <input type="button" value="Löschen"/>

Abbildung 5: Anpassung der zuvor erstellten Sicherheitsgruppe mit zusätzlichen Regeln für Docker

## 2.0.4 EC2-Instanzen (Nodes)

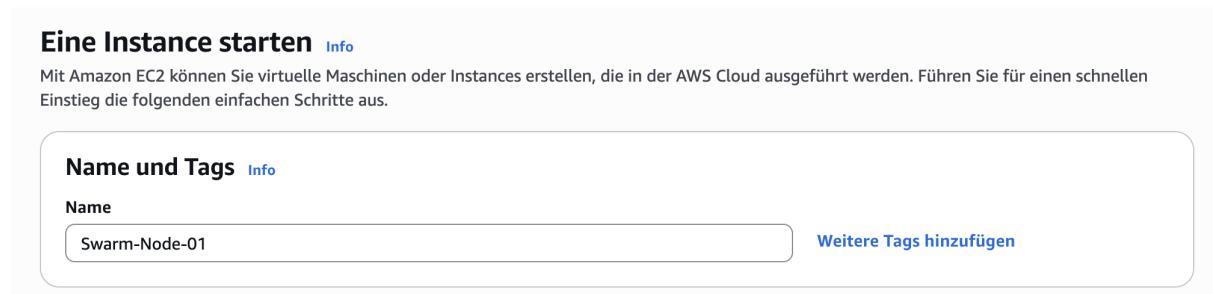


Abbildung 6: Erstellung der ersten Instanz in AWS

The screenshot shows the 'Anwendungs- und Betriebssystemabbilder (Amazon Machine Image)' (Applications and operating system images) section of the AWS Lambda console. It lists various AMI categories: Amazon Linux, macOS, Ubuntu, Windows, Red Hat, SUSE Linux, and Debian. The 'Ubuntu' category is selected. A search bar at the top right says 'Durchsuchen Sie unseren vollständigen Katalog mit Tausenden von Anwendungs- und Betriebssystemabbildern' (Search our complete catalog with thousands of application and operating system images). Below the list, a detailed view of the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI is shown, including its ID, creation date, and provider information.

Architektur	AMI-ID	Veröffentlichungsdatum	Benutzername	Verifizierter Anbieter
64-Bit (x86)	ami-0360c520857e3138f	2025-08-21	ubuntu	Verifizierter Anbieter

Abbildung 7: AMI: Ubuntu (neueste Version).

▼ **Instance-Typ** [Info](#) | [Lassen Sie sich beraten](#)

**Instance-Typ**

<b>t3.micro</b> Familie: t3 2 vCPU 1 GiB Arbeitsspeicher Aktuelle Generation: true On-Demand Ubuntu Pro Basis Preise: 0.0139 USD pro Stunde On-Demand SUSE Basis Preise: 0.0104 USD pro Stunde On-Demand Linux Basis Preise: 0.0104 USD pro Stunde On-Demand RHEL Basis Preise: 0.0392 USD pro Stunde On-Demand Windows Basis Preise: 0.0196 USD pro Stunde	<b>Zur kostenlosen Nutzung berechtigt</b> <input checked="" type="checkbox"/> Alle Generationen <a href="#">Instance-Typen vergleichen</a>
---	--

Für AMIs mit vorinstallierter Software fallen zusätzliche Kosten an

▼ **Schlüsselpaar (Anmeldung)** [Info](#)

Sie können ein Schlüsselpaar verwenden, um eine sichere Verbindung zu Ihrer Instance herzustellen. Stellen Sie sicher, dass Sie Zugriff auf das ausgewählte Schlüsselpaar haben, bevor Sie die Instance starten.

**Schlüsselpaarname - Pflichtfeld**

[Neues Schlüsselpaar erstellen](#)

Abbildung 8: t2.micro (oder vergleichbar) als Instance Type

▼ **Netzwerkeinstellungen** [Info](#)

**VPC – Pflichtfeld** | [Info](#)

vpc-0374fb79a4fa5937d (vpc-docker-swarm)  
 10.0.0.0/16

[Neues Subnetz erstellen](#)

**Subnetz** | [Info](#)

subnet-03112b387b5c36fcb sbn-docker-swarm  
 VPC: vpc-0374fb79a4fa5937d Besitzer: 767397790708  
 Availability Zone: us-east-1d (use1-az6) Zonentyp: Availability Zone  
 IP-Adressen verfügbar: 251 CIDR: 10.0.1.0/24

[Neues Subnetz erstellen](#)

**Öffentliche IP automatisch zuweisen** | [Info](#)

Aktivieren

**Firewall (Sicherheitsgruppen)** | [Info](#)

Eine Sicherheitsgruppe ist eine Reihe von Firewall-Regeln, die den Datenverkehr für Ihre Instance steuern. Fügen Sie Regeln hinzu, damit bestimmter Datenverkehr Ihre Instance erreichen kann.

Sicherheitsgruppe erstellen
 Vorhandene Sicherheitsgruppe auswählen

**Gemeinsame Sicherheitsgruppen** | [Info](#)

Sicherheitsgruppen auswählen  
 default sg-043414993d45d5310 X  
 VPC: vpc-0374fb79a4fa5937d

[Sicherheitsgruppenregeln vergleichen](#)

Sicherheitsgruppen, die Sie hier hinzufügen oder entfernen, werden zu allen Netzwerkschnittstellen hinzugefügt oder von diesen entfernt.

► **Erweiterte Netzwerkkonfiguration**

Abbildung 9: Aktivierung von **Auto-assign Public IP**.

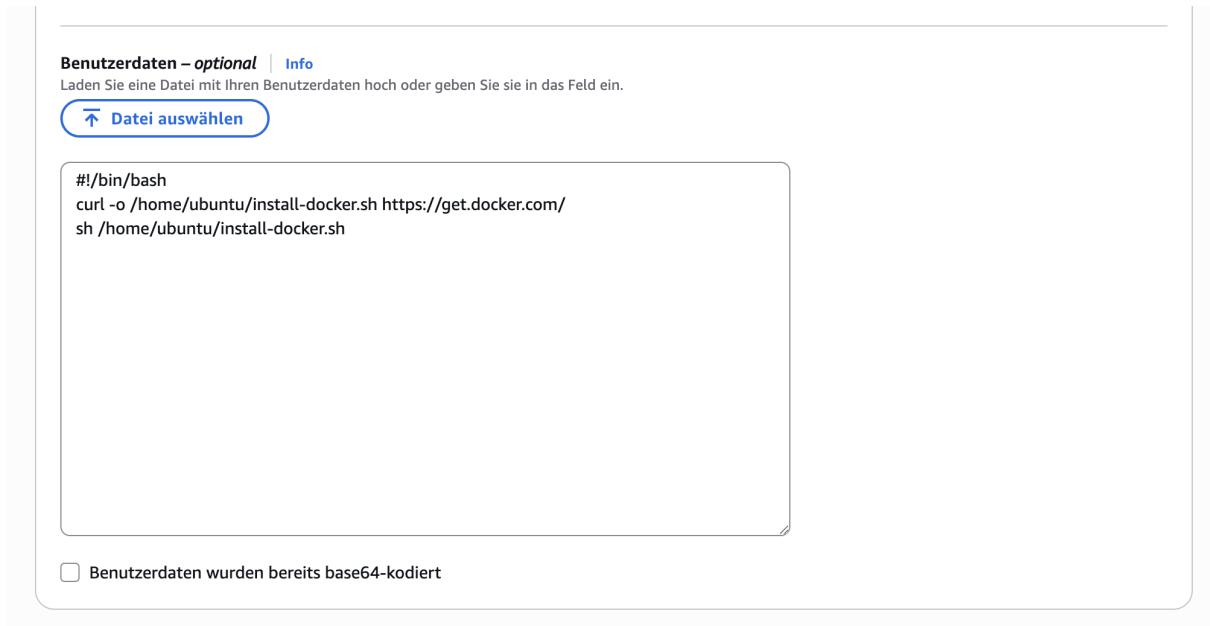


Abbildung 10: Einfügen des User-Data-Skripts

```
#!/bin/bash
curl -o /home/ubuntu/install-docker.sh https://get.docker.com/
sh /home/ubuntu/install-docker.sh
```

Listing 2.1: EC2 User Data – Docker Installation

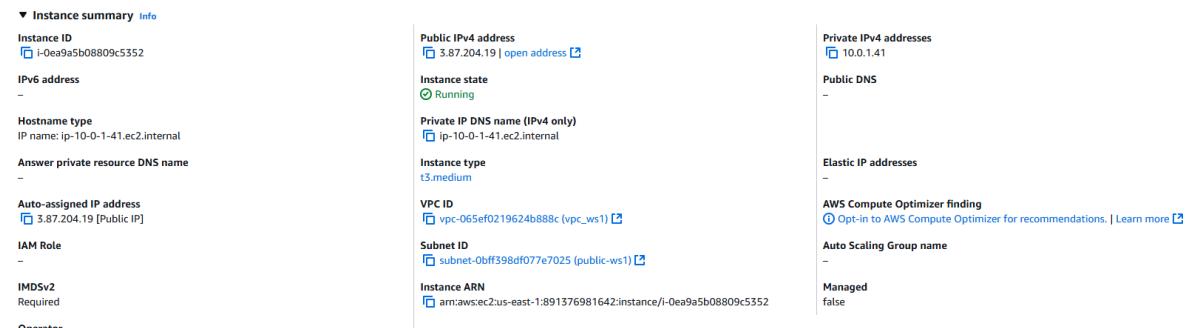


Abbildung 11: EC2 Launch – Übersicht der Einstellungen

## 2.0.5 Validierung

```
ssh -i <key.pem> ubuntu@<PUBLIC_IP>
docker --version
```

```

ubuntu@ip-10-0-1-135:~$ ssh -i "infc.pem" ubuntu@3.90.140.216
This host key is known by the following other names/addresses:
  ~./ssh/known_hosts:52: 98.85.224.143
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.90.140.216' (ED25519) to the list of known hosts.

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

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

System information as of Sun Oct 12 17:42:46 UTC 2025

System load: 0.0 Temperature: -273.1 C
Usage of /: 42.8% of 6.71GB Processes: 113
Memory usage: 28% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.1.135

Expanded Security Maintenance for Applications is not enabled.

22 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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

*** System restart required ***

ubuntu@ip-10-0-1-135:~$ Warp anything @

ubuntu@ip-10-0-1-46:~$ ssh -i "infc.pem" ubuntu@52.205.178.11
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

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

System information as of Sun Oct 12 17:43:54 UTC 2025

System load: 0.71 Temperature: -273.1 C
Usage of /: 42.8% of 6.71GB Processes: 128
Memory usage: 33% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.1.32

Expanded Security Maintenance for Applications is not enabled.

12 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

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

*** System restart required ***

ubuntu@ip-10-0-1-46:~$ Warp anything e @

ubuntu@ip-10-0-1-32:~$ ssh -i "infc.pem" ubuntu@54.166.145.123
ot", use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

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

System information as of Sun Oct 12 17:43:54 UTC 2025

System load: 0.71 Temperature: -273.1 C
Usage of /: 42.8% of 6.71GB Processes: 128
Memory usage: 33% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.1.62

Expanded Security Maintenance for Applications is not enabled.

59 updates can be applied immediately.
35 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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

*** System restart required ***

ubuntu@ip-10-0-1-32:~$ Warp anything e @

ubuntu@ip-10-0-1-62:~$ ssh -i "infc.pem" ubuntu@52.206.151.132
See "man sudo_root" for details.

Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)

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

System information as of Sun Oct 12 17:44:30 UTC 2025

System load: 0.0 Temperature: -273.1 C
Usage of /: 39.6% of 6.71GB Processes: 113
Memory usage: 29% Users logged in: 0
Swap usage: 0% IPv4 address for ens5: 10.0.1.62

Expanded Security Maintenance for Applications is not enabled.

59 updates can be applied immediately.
35 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

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

*** System restart required ***

ubuntu@ip-10-0-1-62:~$ Warp anything e.g @

```

Abbildung 12: SSH-Verbindung und Docker-Version

## 2.0.6 Docker Swarm Setup

```

docker swarm init --advertise-addr <PRIVATE_IP>
docker node ls

```

```

ubuntu@ip-10-0-1-135:~$ sudo docker swarm init
Swarm initialized: current node (iwtvfldegzatqys0akxhe386z) is now a manager.

To add a worker to this swarm, run the following command:

  docker swarm join --token SWMTKN-1-5z9kkzo27ot0npy0dha6u1z1xum6gymw889e5l7bzlxhmnn6awo-8o3x4yyj7u1v1fyjsprdchgif 10.0.1.135:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

```

Abbildung 13: docker swarm init auf der ersten Instanz

```

ubuntu@ip-10-0-1-135:~$ sudo docker node ls
ID          HOSTNAME   STATUS    AVAILABILITY  MANAGER STATUS   ENGINE VERSION
iwtvfldegzatqys0akxhe386z *  ip-10-0-1-135  Ready   Active        Leader        28.5.1

```

Abbildung 14: docker node ls auf einer Instanz

```

ubuntu@ip-10-0-1-46:~$ sudo docker swarm join --token SWMTKN-1-5z9kkzo27ot0npy0dha6u1z1xum6gymw889e5l7bzlxhm6awo-8o3x4yyj7u1v1fyjsprchgif 10.0.1.135:2377
This node joined a swarm as a worker.

ubuntu@ip-10-0-1-32:~$ sudo docker swarm join --token SWMTKN-1-5z9kkzo27ot0npy0dha6u1z1xum6gymw889e5l7bzlxhm6awo-8o3x4yyj7u1v1fyjsprchgif 10.0.1.135:2377
This node joined a swarm as a worker.

```

Abbildung 15: Hinzufügen von zwei Workern

```

ubuntu@ip-10-0-1-135:~$ sudo docker swarm join-token manager
To add a manager to this swarm, run the following command:

    docker swarm join --token SWMTKN-1-5z9kkzo27ot0npy0dha6u1z1xum6gymw889e5l7bzlxhm6awo-4ues15ehcaoqur9v0zjp2pvf5 10.0.1.135:2377

```

Abbildung 16: Generieren des Manager-Tokens

```

ubuntu@ip-10-0-1-62:~$ sudo docker swarm join --token SWMTKN-1-5z9kkzo27ot0npy0dha6u1z1xum6gymw889e5l7bzlxhm6awo-4ues15ehcaoqur9v0zjp2pvf5 10.0.1.135:2377
This node joined a swarm as a manager.

```

Abbildung 17: Hinzufügen eines zweiten Managers

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION
cma7tplxrm6hdy883t9ejhya	ip-10-0-1-32	Ready	Active		28.5.1
knfdwefa0itt6pu5h5sh8p002	ip-10-0-1-46	Ready	Active		28.5.1
saejuboorjefau6apj1q4wmvz	ip-10-0-1-62	Ready	Active	Reachable	28.5.1
iwtvfldegzatqys0akxhe386z *	ip-10-0-1-135	Ready	Active	Leader	28.5.1

Abbildung 18: docker node ls mit 4 Knoten

## 2.0.7 Portainer Installation

```
curl -L https://downloads.portainer.io/ce-lts/portainer-agent-stack.yml -o
portainer-agent-stack.yml
docker stack deploy -c portainer-agent-stack.yml portainer
```

ubuntu@ip-10-0-1-62:~\$ sudo su						
ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS	ENGINE VERSION	
cma7tplxrm6hdy883t9ejhya	ip-10-0-1-32	Ready	Active		28.5.1	
knfdwefa0itt6pu5h5sh8p002	ip-10-0-1-46	Ready	Active		28.5.1	
saejuboorjefau6apj1q4wmvz *	ip-10-0-1-62	Ready	Active	Leader	28.5.1	
iwtvfldegzatqys0akxhe386z	ip-10-0-1-135	Ready	Active	Reachable	28.5.1	

```
root@ip-10-0-1-62:/home/ubuntu# curl -L https://downloads.portainer.io/ce-lts/portainer-agent-stack.yml -o portainer-agent-stack.yml
% Total    % Received % Xferd  Average Speed   Time   Time  Current
          Dload  Upload Total Spent   Left Speed
100  785  100  785    0     0  4465      0 --:--:-- --:--:-- 4485
root@ip-10-0-1-62:/home/ubuntu# docker stack deploy -c portainer-agent-stack.yml portainer
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating network portainer_agent_network
Creating service portainer_portainer
Creating service portainer_agent
root@ip-10-0-1-62:/home/ubuntu#
```

Abbildung 19: Portainer-Installation über die Kommandozeile

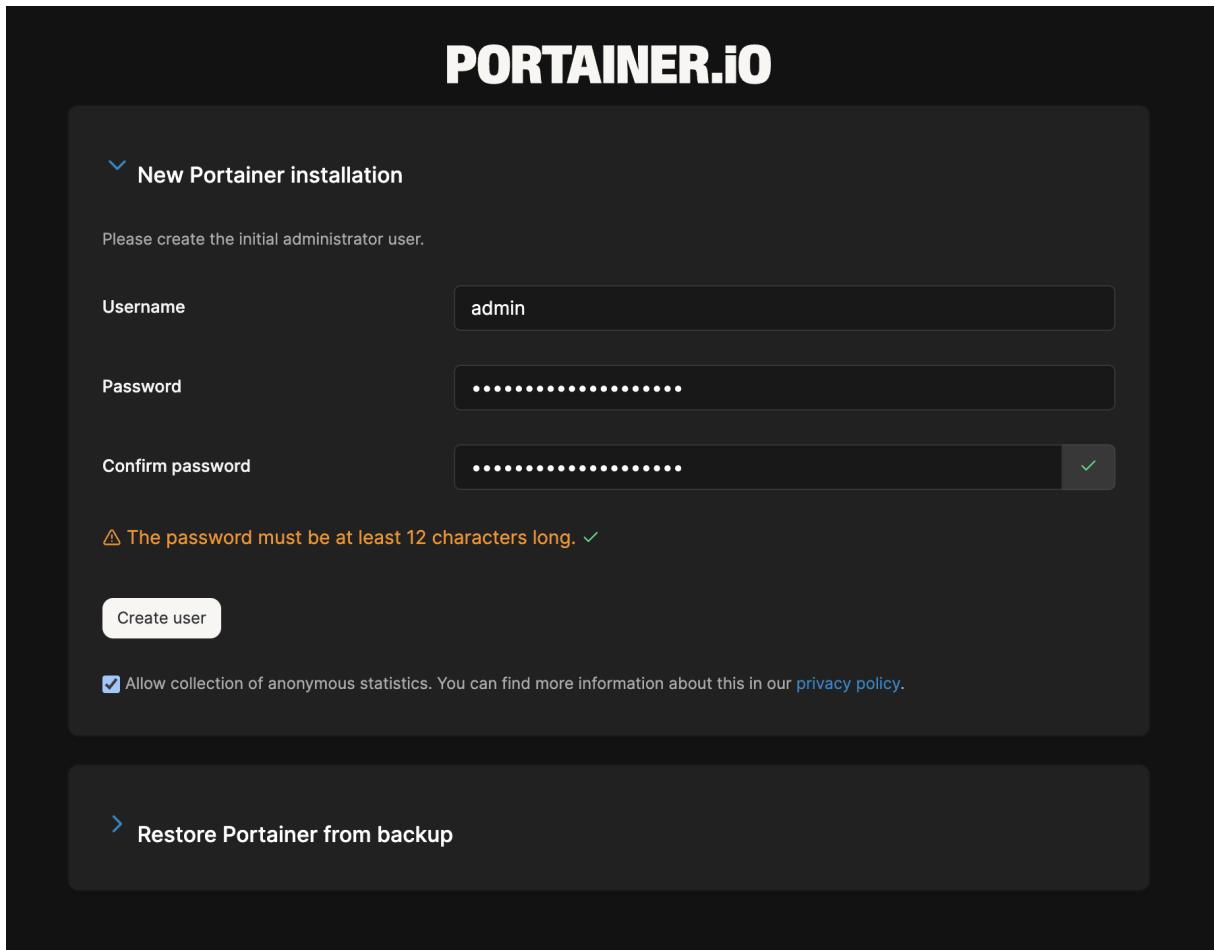


Abbildung 20: Erstellung eines neuen Logins für Portainer

Services						
	Name	Stack	Image	Scheduling Mode	Published Ports	Last Update
■	portainer_agent	portainer	portainer/agent:its	global 4 / 4	-	2025-09-26 15:16:57
	Status	Filter	Task	Actions	Slot	Node
	running	m189q7aevt0cwy0g7ydsupdb		↻	-	ip-10-0-1-109
	running	t21fphrcx0pxbjmuzq03mlaxk		↻	-	ip-10-0-1-41
	running	w8qlap3u1uu4aeks92qlvm5s0a		↻	-	ip-10-0-1-150
	running	zc49gvk2xniuaygefrngv1d		↻	-	ip-10-0-1-143
■	portainer_portainer	portainer	portainer/portainer-ce:its	replicated 1 / 1 * Scale	8000:8000 9000:9000 9443:9443	2025-09-26 15:16:57
	Status	Filter	Task	Actions	Slot	Node
	running	bjyfg4npog2asitfl18e5kno		↻	1	ip-10-0-1-41

Abbildung 21: Angezeigte Services nach der Installation von Portainer

## 2.0.8 Service-Deployment, Scale-Out und Scale-In

```
docker service create \
--name hello \
--replicas 3 \
--publish published=80,target=80 \
karthequian/helloworld:latest
```

The screenshot shows the Docker Compose interface for creating a new service. The service is named 'hello'. In the 'Image configuration' section, the registry is set to 'Docker Hub (anonymous)' and the image is 'karthequian/helloworld:latest'. Under 'Scheduling', the mode is 'Replicated' and there are 3 replicas. In the 'Ports configuration' section, a port mapping is defined from host port 80 to container port 80. The interface includes a search bar for images and a note about using an anonymous DockerHub account.

Abbildung 22: Service mit 3 Replikas und Port 80 erstellen

flamboyant_dewdney		karthequian/helloworld:latest	replicated	3 / 3	x Scale	-	2025-09-26 15:26:54	administrators
Status	Task		Actions	Slot	Node	Last Update		
running	lreb3dmwzkuz7nh1y2h0c28dq		☰ ⚡ ↻	1	ip-10-0-1-109	2025-09-26 15:27:00		
running	psm8je0apq6f89jg3rmycbs		☰ ⚡ ↻	3	ip-10-0-1-150	2025-09-26 15:27:00		
running	uhqf23mm5egrpc1lqd8t1cpd		☰ ⚡ ↻	2	ip-10-0-1-143	2025-09-26 15:27:00		

Abbildung 23: Die 3 Replikas erhalten jeweils einen eigenen Slot

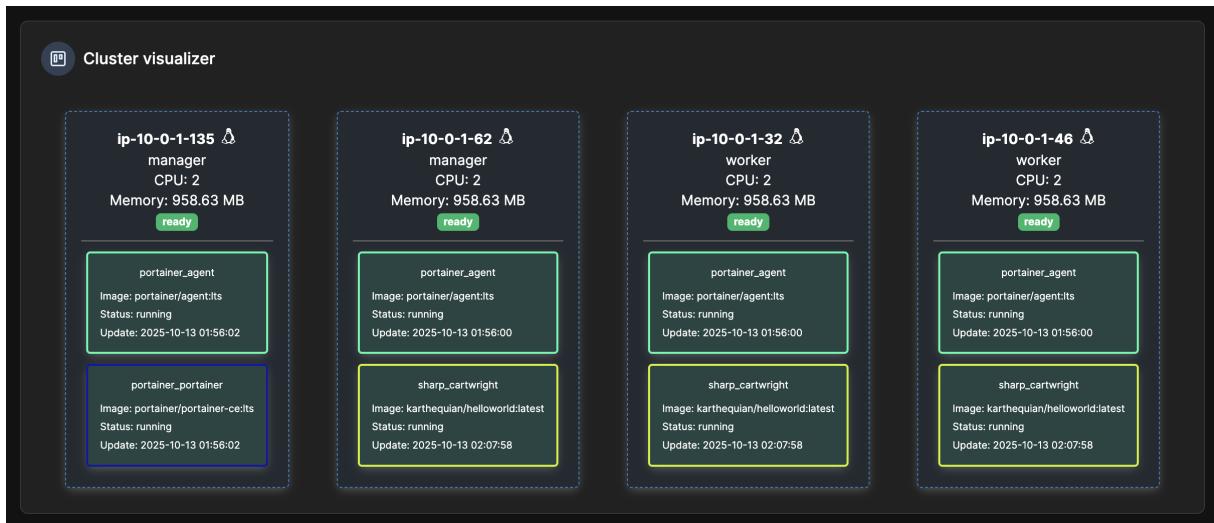


Abbildung 24: Die 3 Replikas werden auf die 4 verschiedenen Nodes in AWS verteilt

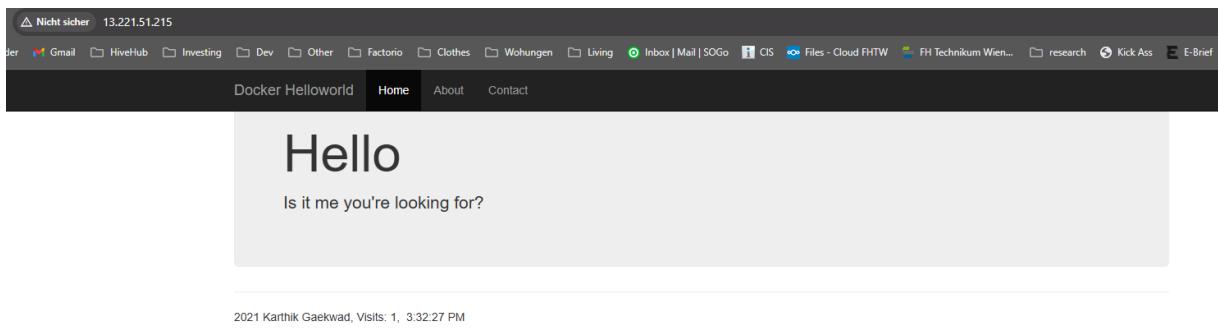


Abbildung 25: Hello-World-Seite nach Deployment des Service

```
docker service scale hello=10
```

Services						
	Name ↴	Stack ↴	Image ↴	Scheduling Mode ↴	Published Ports ↴	Last Update ↴
heuristic_tesla	-	-	karthequian/helloworld:latest	replicated 10 / 10 * Scale	80:80	2025-09-26 15:34:12
	Status ↴	Task ↴		Actions	Slot ↴	Node ↴
	running	46qtcrrcczfe6c55vysxvhvb2d		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	2	ip-10-0-1-150
	running	4ely57111v4u9y7gms1a7agre9		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	6	ip-10-0-1-150
	running	6vv1pdwhy8eatlog9xss1ohf2		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	1	ip-10-0-1-143
	running	iu0a8tboxe8z0rs4uf93smhej		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	3	ip-10-0-1-109
	running	1fc5uv5jg58kr8pwbdxe559cn1		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	8	ip-10-0-1-143
	running	maickcotz6getvhinp2x1687b		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	10	ip-10-0-1-109
	running	ntwvxmu9wz07b637qh7cp8kp		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	5	ip-10-0-1-143
	running	sqytta7853c05d05sfrrk6r3e6		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	4	ip-10-0-1-41
	running	xvkxxux82eybhemy1v6dg60		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	7	ip-10-0-1-41
	running	zp16ecjuxofaw6yfr-fc0utqnup		⋮ ⌂ ⌃ ⌄ ⌅ ⌆ ⌇ ⌈ ⌉ ⌊ ⌋	9	ip-10-0-1-150

Abbildung 26: Die 10 Replikas werden nach dem Hochskalieren auf verschiedene Slots verteilt



Abbildung 27: Die 10 Replikas werden auf die 4 verschiedenen Nodes verteilt

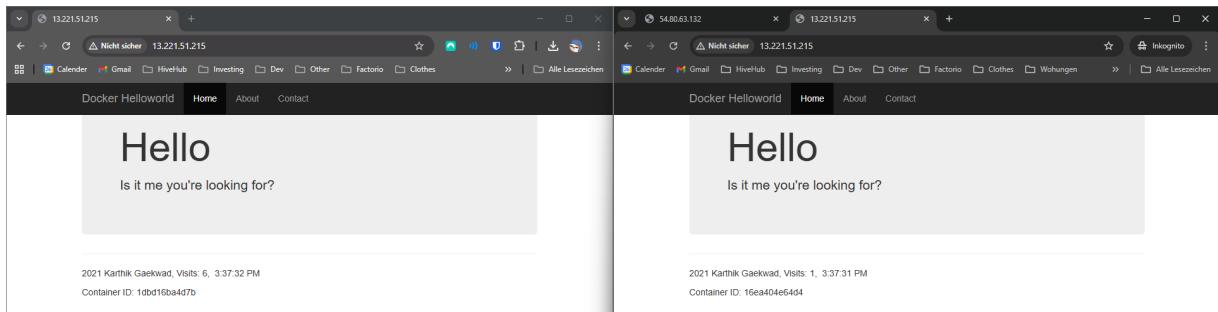


Abbildung 28: Hello-World-Seite wird nun von verschiedenen Containern bedient

Name	Role	CPU	Memory	Engine	IP Address	Status	Availability
ip-10-0-1-41	manager	2	4 GB	28.4.0	10.0.1.41	ready	active
ip-10-0-1-143	worker	2	4 GB	28.4.0	10.0.1.143	down	active
ip-10-0-1-150	worker	2	4 GB	28.4.0	10.0.1.150	ready	active
ip-10-0-1-109	manager	2	4 GB	28.4.0	10.0.1.109	ready	active

Abbildung 29: Node wird als „down“ angezeigt, nachdem eine EC2 heruntergefahren wurde

Name	Stack	Image	Scheduling Mode	Published Ports	Last Update	Ownership
heuristic.tesla	-	karthequian/helloworld:latest	replicated	7 / 10 * Scale	80:80	2025-09-26 15:34:12
						administrators
Status	Task		Actions	Slot	Node	Last Update
running	46qtcrrcczf6c55vysxvhb2d		⋮ ⚡ ↻	2	ip-10-0-1-150	2025-09-26 15:32:12
running	4ely57111v4dy/gm1n7agre9		⋮ ⚡ ↻	6	ip-10-0-1-150	2025-09-26 15:34:13
ready	5mv73rd5twww027qtjnehkgle		⋮ ⚡	8	ip-10-0-1-41	2025-09-26 15:38:52
running	6vv1pdwhy8eatloq9xss1ohf2		⋮ ⚡	1	ip-10-0-1-143	2025-09-26 15:38:52
ready	91cuvvu4u8qg7cqz9mpx1o1yys		⋮ ⚡	5	ip-10-0-1-150	2025-09-26 15:38:52
running	iu0a81boxe820es4uf93smhej		⋮ ⚡ ↻	3	ip-10-0-1-109	2025-09-26 15:32:12
running	1fo5ov5g58kr8pwbxde599cnl		⋮ ⚡	8	ip-10-0-1-143	2025-09-26 15:38:52
running	maickco1z6getvhinp2x1687b		⋮ ⚡ ↻	10	ip-10-0-1-109	2025-09-26 15:34:13
running	mtwvvvmuhz07b637qh7cp8kp		⋮ ⚡	5	ip-10-0-1-143	2025-09-26 15:38:52
ready	pdqsd18z3vifst74dspfeobjy		⋮ ⚡	1	ip-10-0-1-109	2025-09-26 15:38:52

Abbildung 30: Services auf dem betroffenen Node wechselten erst nach „ready“, dann nach „shutdown“, neue Replikas wurden auf andere Nodes verteilt

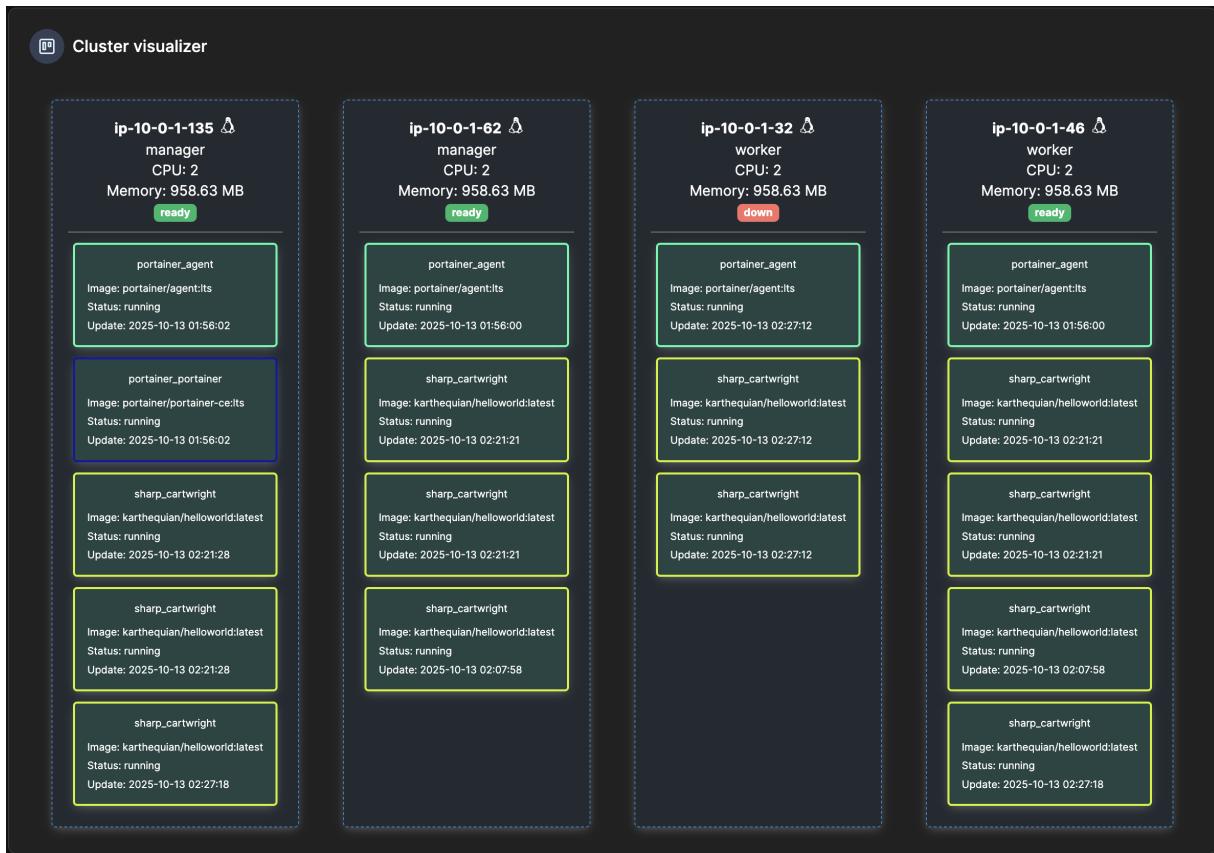


Abbildung 31: Node down mit zwei Services; auf den restlichen drei Nodes laufen insgesamt 10 Services

Services

Name	Stack	Image	Scheduling Mode	Published Ports	Last Update	Ownership
heuristic_testa	-	karthequian/helloworld:latest	replicated	15 / 15 * Scale	2025-09-26 15:55:48	administrators
Status Filter Task			Actions	Slot	Node	Last Update
running	0g2mnx41xse4r2bsn3jfhfs7z		↻ ↻ ↺ ↻ ↻	15	ip-10-0-1-143	2025-09-26 15:55:50
running	46qtcrrcczf6c55vysxvhvb2d		↻ ↻ ↺ ↻ ↻	2	ip-10-0-1-150	2025-09-26 15:32:12
shutdown	6vv1pdwhy8etllog9xs5lohf2		↻ ↻ ↺ ↻ ↻	1	ip-10-0-1-143	2025-09-26 15:47:18
running	91cvvv4u8lg7cqz9np10ivs		↻ ↻ ↺ ↻ ↻	5	ip-10-0-1-150	2025-09-26 15:38:57
running	bms9633qt0gsl6w28tk2le12a		↻ ↻ ↺ ↻ ↻	7	ip-10-0-1-150	2025-09-26 15:55:49
running	e2h04bes99g6kpkqd3t4kjhg		↻ ↻ ↺ ↻ ↻	12	ip-10-0-1-41	2025-09-26 15:55:49
running	f646mcuicht93ojconvmlld7d		↻ ↻ ↺ ↻ ↻	6	ip-10-0-1-41	2025-09-26 15:55:49
running	iu0o81boxe8z0rs4uf93smhej		↻ ↻ ↺ ↻ ↻	3	ip-10-0-1-109	2025-09-26 15:32:12
shutdown	1fo5ov5g58kr8pbwdx559cn1		↻ ↻	8	ip-10-0-1-143	2025-09-26 15:47:18
shutdown	mtwwvxmu9wz07b637qh7cp8kp		↻ ↻	5	ip-10-0-1-143	2025-09-26 15:47:18

Abbildung 32: Nach dem Hochfahren wurden die übrigen Services wieder sauber auf alle 4 Nodes verteilt

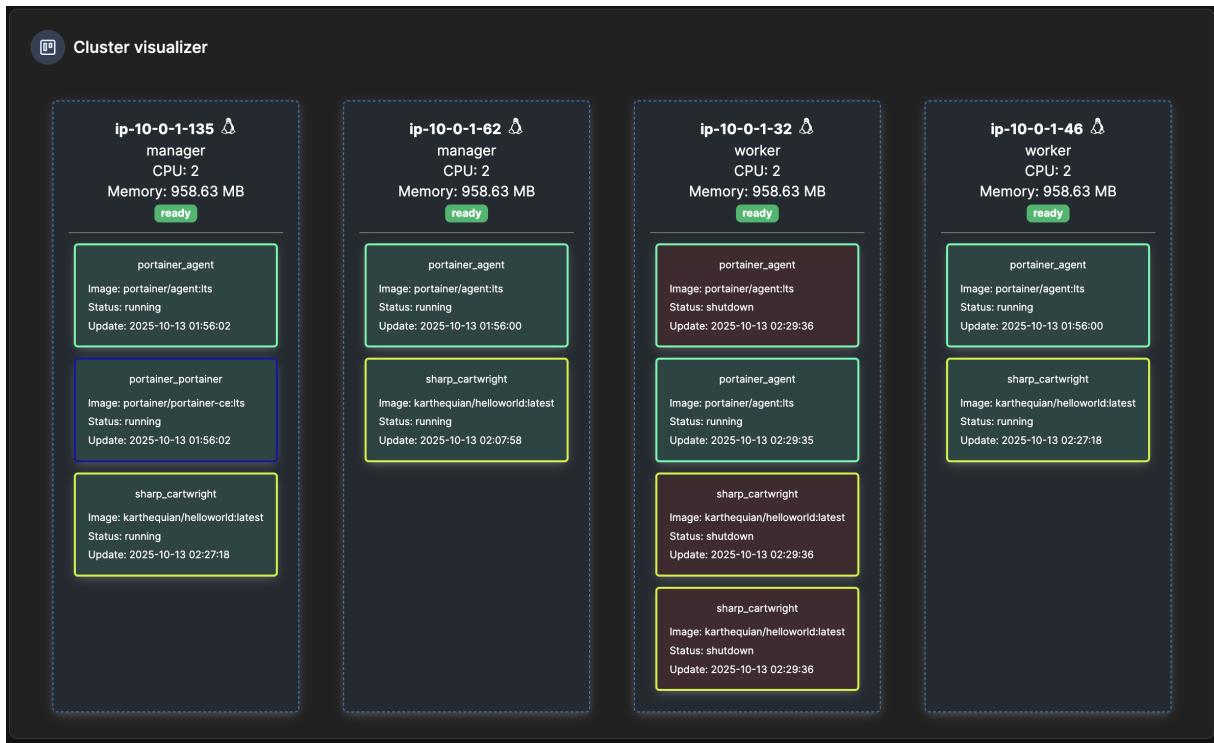


Abbildung 33: Beim Scale-Down wurden die verbleibenden 3 Services erneut gleichmäßig verteilt

## 2.0.9 Kurzantworten

- **Skalierung des Clusters:**

Skalieren Sie Ihren Cluster von 3 auf 10 Maschinen.

*Beobachtung:* Beim Öffnen der Hello-World-Seite wird jeweils ein anderer Container angezeigt.

- **Wiederinbetriebnahme eines Workers:**

Wenn Sie den Worker wieder online schalten, werden die Services automatisch auf alle vier Nodes verteilt.

- **Erneute Skalierung:**

Bei einer erneuten Skalierung werden die drei Services erneut gleichmäßig auf die vier Nodes verteilt.

## 2.0.10 Anhang – Wichtige Befehle

```
docker service ls
docker service ps hello
docker node ls
docker service scale hello=10
```

# 3 Kubernetes Deployment Workshop mit Minikube

Ziel: In diesem Workshop wurde ein Kubernetes-Cluster mit Minikube erstellt, ein nginx-Deployment angelegt und verschiedene Service-Typen (ClusterIP, NodePort, LoadBalancer) getestet.

## 3.0.1 Installation Minikube und Kubectl

```
curl -LO https://github.com/kubernetes/minikube/releases/latest/download/
      minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-
      linux-amd64
```

Listing 3.1: Installation von Minikube

```
ubuntu@ip-10-0-1-120:~$ curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
% Total    % Received % Xferd  Average Speed   Time     Time     Current
          Dload  Upload   Total   Spent    Left  Speed
0       0     0     0       0      0      0 --:--:-- --:--:-- --:--:--   0
0       0     0     0       0      0      0 --:--:-- --:--:-- --:--:--   0
100  133M  100  133M     0      0  172M      0 --:--:-- --:--:-- --:--:-- 106M

ubuntu@ip-10-0-1-120:~$ sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

Abbildung 34: Download und Installation von Minikube

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/
stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

Listing 3.2: Installation von Kubectl

```
ubuntu@ip-10-0-1-120:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
% Total    % Received % Xferd  Average Speed   Time     Time     Current
          Dload  Upload   Total   Spent    Left  Speed
100  138  100  138     0      0  2053      0 --:--:-- --:--:-- --:--:-- 2059
100 57.7M  100 57.7M     0      0  220M      0 --:--:-- --:--:-- --:--:-- 220M

ubuntu@ip-10-0-1-120:~$ sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

ubuntu@ip-10-0-1-120:~$ kubectl version --client
Client Version: v1.34.1
Kustomize Version: v5.7.1
```

Abbildung 35: Download und Installation von Kubectl

## 3.0.2 Start und Vorbereitung

```
minikube start  
kubectl get nodes
```

Listing 3.3: Start von Minikube und Überprüfung der Umgebung

```
ubuntu@ip-10-0-1-120:~$ minikube start --driver=docker --memory=2048mb  
😄 minikube v1.37.0 on Ubuntu 24.04  
⭐ Using the docker driver based on user configuration  
📌 Using Docker driver with root privileges  
👍 Starting "minikube" primary control-plane node in "minikube" cluster  
🚜 Pulling base image v0.0.48 ...  
🌐 Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...  
🔗 Configuring bridge CNI (Container Networking Interface) ...  
🌐 Verifying Kubernetes components...  
▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5  
⭐ Enabled addons: default-storageclass, storage-provisioner  
🌟 Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
```

Abbildung 36: Start von Minikube mit einer einzelnen Node (2GB Ram da sonst alles gebraucht wird)

```
ubuntu@ip-10-0-1-120:~$ kubectl get nodes  
NAME      STATUS   ROLES      AGE      VERSION  
minikube  Ready    control-plane  88s     v1.34.0  
ubuntu@ip-10-0-1-120:~$ █
```

Abbildung 37: Alle laufenden Notes

### 3.0.3 Deployment erstellen und skalieren

```
kubectl create deployment nginx --image=nginx:stable  
kubectl get deployments  
kubectl get pods
```

Listing 3.4: nginx-Deployment mit einem Pod erstellen

```
ubuntu@ip-10-0-1-120:~$ kubectl create deployment nginx --image=nginx:stable  
deployment.apps/nginx created  
ubuntu@ip-10-0-1-120:~$ █
```

Abbildung 38: Erstellung des nginx-Deployments

```

ubuntu@ip-10-0-1-120:~$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
nginx    1/1        1           1          14s
ubuntu@ip-10-0-1-120:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
nginx-76bb4bdf9f-lq8zz 1/1     Running   0          21s
ubuntu@ip-10-0-1-120:~$

```

Abbildung 39: Anzeigen des erstellten Deployments

```

kubectl scale deployment nginx --replicas=4
kubectl get pods

```

Listing 3.5: Deployment auf vier Replikas skalieren

```

ubuntu@ip-10-0-1-120:~$ kubectl scale deployment nginx --replicas=4
deployment.apps/nginx scaled
ubuntu@ip-10-0-1-120:~$

```

Abbildung 40: Vier Replikas des nginx-Deployments sind aktiv

```

ubuntu@ip-10-0-1-120:~$ kubectl get pods
NAME                  READY   STATUS    RESTARTS   AGE
nginx-76bb4bdf9f-j8rjq 1/1     Running   0          12s
nginx-76bb4bdf9f-lq8zz 1/1     Running   0          2m2s
nginx-76bb4bdf9f-mxgsf 1/1     Running   0          12s
nginx-76bb4bdf9f-pkkfn 1/1     Running   0          12s
ubuntu@ip-10-0-1-120:~$

```

Abbildung 41: Anzeigen der erstellten Pods

### 3.0.4 Testzugriff auf einen Pod

```

kubectl expose deployment nginx --type=NodePort --port=8080

```

Listing 3.6: Port-Forwarding für lokalen Zugriff

```

ubuntu@ip-10-0-1-120:~$ kubectl expose deployment nginx --type=NodePort --port=8080
service/nginx exposed
ubuntu@ip-10-0-1-120:~$

```

Abbildung 42: Erstelltes Deployment auf Port 8080 exposen

```

ubuntu@ip-10-0-1-120:~$ minikube service nginx

```

NAMESPACE	NAME	TARGET PORT	URL
default	nginx	8080	http://192.168.49.2:32381

```

💡 Opening service default/nginx in default browser...
👉 http://192.168.49.2:32381
ubuntu@ip-10-0-1-120:~$ █

```

Abbildung 43: Browser öffnen mit Zugriff auf Pod

### 3.0.5 ClusterIP Service (intern)

```

kubectl expose deployment nginx \
--name=nginx-clusterip \
--type=ClusterIP \
--port=8080 \
--target-port=80
kubectl get svc nginx-clusterip

```

Listing 3.7: Erstellung eines internen ClusterIP-Service

```

ubuntu@ip-10-0-1-120:~$ kubectl expose deployment nginx \
--name=nginx-clusterip \
--type=ClusterIP \
--port=8080 \
--target-port=80
service/nginx-clusterip exposed
ubuntu@ip-10-0-1-120:~$ █

```

Abbildung 44: Interner ClusterIP-Service für das nginx-Deployment

```

kubectl run test-client --rm -it --image=curlimages/curl --restart=Never --
sh
curl -I http://nginx-clusterip:8080

```

Listing 3.8: Test im Cluster über temporären Curl-Pod

```

ubuntu@ip-10-0-1-120:~$ kubectl run test-client --rm -it --image=curlimages/curl --restart=Never -- sh
All commands and output from this session will be recorded in container logs, including credentials and sensitive information passed through the command prompt.
If you don't see a command prompt, try pressing enter.
~ $ curl -I http://nginx-clusterip:8080
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Mon, 13 Oct 2025 23:58:16 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Wed, 23 Apr 2025 11:48:54 GMT
Connection: keep-alive
ETag: "6808d3a6-267"
Accept-Ranges: bytes
~ $ █

```

# STAY [FOCUSED]

Abbildung 45: Erfolgreicher Curl-Test des internen ClusterIP-Service

### 3.0.6 NodePort Service (extern)

```

kubectl expose deployment nginx \
--name=nginx-nodeport \
--type=NodePort \
--port=8080 \
--target-port=80
kubectl get svc nginx-nodeport

```

Listing 3.9: Erstellung eines NodePort-Service

```

ubuntu@ip-10-0-1-120:~$ kubectl expose deployment nginx \
--name=nginx-nodeport \
--type=NodePort \
--port=8080 \
--target-port=80
service/nginx-nodeport exposed
ubuntu@ip-10-0-1-120:~$ █

```

Abbildung 46: NodePort-Service mit zugewiesenen externen Port (z. B. 31245)

```
minikube service nginx-nodeport --url
```

Listing 3.10: Abrufen der externen URL über Minikube

```

ubuntu@ip-10-0-1-120:~$ minikube service nginx-nodeport --url
http://192.168.49.2:30095
ubuntu@ip-10-0-1-120:~$ █

```

Abbildung 47: Von Minikube generierte URL für NodePort-Zugriff

Zugriff im Browser oder mit curl:

```
curl -I $(minikube service nginx-nodeport --url)
```

```
ubuntu@ip-10-0-1-120:~$ curl -I $(minikube service nginx-nodeport --url)
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Tue, 14 Oct 2025 00:00:10 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Wed, 23 Apr 2025 11:48:54 GMT
Connection: keep-alive
ETag: "6808d3a6-267"
Accept-Ranges: bytes
```

```
ubuntu@ip-10-0-1-120:~$ █
```

Abbildung 48: Erfolgreicher Curl-Test des internen Nodeport-Service

### 3.0.7 LoadBalancer Service (simuliert)

```
kubectl expose deployment nginx \
--name=nginx-lb \
--type=LoadBalancer \
--port=8080 \
--target-port=80
```

Listing 3.11: Erstellung eines LoadBalancer-Service

```
ubuntu@ip-10-0-1-120:~$ kubectl expose deployment nginx \
--name=nginx-lb \
--type=LoadBalancer \
--port=8080 \
--target-port=80
service/nginx-lb exposed
ubuntu@ip-10-0-1-120:~$ █
```

Abbildung 49: LoadBalancer-Service mit zugewiesenenem externen Port (z. B. 31245)

Da Minikube keinen echten Cloud-Load-Balancer bereitstellt, wird die Funktion über einen lokalen Tunnel simuliert.

```
minikube tunnel
```

Listing 3.12: Start des Minikube-Tunnels (neues Terminal)

```
ubuntu@ip-10-0-1-120:~$ minikube tunnel
Status:
  machine: minikube
  pid: 5636
  route: 10.96.0.0/12 -> 192.168.49.2
  minikube: Running
  services: [nginx-lb]
errors:
  minikube: no errors
  router: no errors
  loadbalancer emulator: no errors
```

Abbildung 50: Simulierter LoadBalancer über Minikube-Tunnel

Prüfen der externen IP:

```
kubectl get svc nginx-lb
```

```
ubuntu@ip-10-0-1-120:~$ kubectl get svc nginx-lb
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx-lb   LoadBalancer   10.102.48.195   10.102.48.195   8080:31366/TCP   4m14s
ubuntu@ip-10-0-1-120:~$
```

Abbildung 51: LoadBalancer mit zugewiesener externer IP-Adresse

```
curl -I http://<EXTERNAL-IP>:8080
```

Listing 3.13: Zugriff über die externe IP

```

ubuntu@ip-10-0-1-120:~$ kubectl get svc nginx-lb
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
nginx-lb   LoadBalancer   10.102.48.195   10.102.48.195   8080:31366/TCP   4m14s

ubuntu@ip-10-0-1-120:~$ curl -I http://10.102.48.195:8080
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Tue, 14 Oct 2025 00:05:36 GMT
Content-Type: text/html
Content-Length: 615
Last-Modified: Wed, 23 Apr 2025 11:48:54 GMT
Connection: keep-alive
ETag: "6808d3a6-267"
Accept-Ranges: bytes

ubuntu@ip-10-0-1-120:~$ █

```

Abbildung 52: Erfolgreicher Zugriff über den LoadBalancer-Service

### 3.0.8 Vergleich der Service-Typen

- **ClusterIP (Default):** Erzeugt eine virtuelle, *interne* Cluster-IP. Service ist nur *innerhalb* des Clusters erreichbar. Geeignet für Service-zu-Service-Kommunikation (z. B. Backend).
- **NodePort:** Öffnet zusätzlich auf jedem Node einen Port (Standardbereich 30000–32767). Der Service ist *außerhalb* des Clusters via <NodeIP>:<NodePort> erreichbar. Einfach, aber eingeschränkte Flexibilität und keine integrierte L7-Funktionen.
- **LoadBalancer:** Fragt (in Cloud-Umgebungen) einen externen L4-Load-Balancer an und erhält eine *EXTERNAL-IP*. In Minikube wird dies über `minikube service` oder `minikube tunnel` simuliert. Ideal für produktiven Ingress-Traffic pro Service (L4).

Typ	Erreichbarkeit	Einsatzgebiet / Beschreibung
ClusterIP	Nur innerhalb des Clusters	Standardtyp; ermöglicht Kommunikation zwischen Pods oder internen Diensten.
NodePort	Von außen über Node-IP und festen Port erreichbar	Zugriff über <NodeIP>:<NodePort> (z. B. für Tests).
LoadBalancer	Öffentliche IP / Cloud-Load-Balancer	Für produktive Setups; verteilt Traffic über alle Replikas.

Tabelle 1: Kubernetes Service-Typen

### 3.0.9 YAML-Spezifikationen

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:stable
          ports:
            - containerPort: 80
```

Listing 3.14: Deployment YAML – nginx

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
  labels:
    app: nginx
spec:
```

```

type: LoadBalancer
selector:
  app: nginx
ports:
  - name: http
    port: 8080
    targetPort: 80

```

Listing 3.15: Service YAML – LoadBalancer

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:stable
          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-clusterip
  labels:
    app: nginx
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
    - name: http
      port: 8080
      targetPort: 80

```

```

---
apiVersion: v1
kind: Service
metadata:
  name: nginx-nodeport
  labels:
    app: nginx
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
    - name: http
      port: 8080
      targetPort: 80
      nodePort: 30080
---
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
  labels:
    app: nginx
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
    - name: http
      port: 8080
      targetPort: 80

```

Listing 3.16: Kombinierte YAML-Datei (Deployment + alle Service)

### 3.0.10 Zusammenfassung

- **Cluster:** Ein lokales Kubernetes-Cluster mit Minikube wurde gestartet.
- **Deployment:** Ein nginx-Deployment mit vier Replikas wurde erstellt.
- **Service-Typen:** ClusterIP, NodePort und LoadBalancer wurden erfolgreich getestet.
- **Zugriff:** Über Port-Forwarding, NodePort und Minikube-Tunnel konnte der Webserver erreicht werden.

- **Fazit:** Der Workshop verdeutlicht, wie Deployments und Services in Kubernetes strukturiert sind und welche Service-Typen für interne bzw. externe Zugriffe geeignet sind.