

Modul 346 LB2

Maurice Däppen, Mika Hannappel und Patrick Aeschlimann

December 2022/Januar 2023



Contents

1	Einleitung	3
2	EC2	4
2.1	Minecraft Server	4
2.2	Pihole	7
2.3	VPN	10
3	Datenbank	13
4	S3	20
4.1	Beschreibung	20
4.2	Installation	20
4.3	Verfügbarkeit	25
4.4	Datensicherheit	25
4.5	Kostenanalyse	26
5	Amazon Elastic Container Service	27
5.1	Beschreibung	27
5.2	Installation	27
5.3	Verfügbarkeit	34
5.4	Datensicherheit	34
5.5	Kostenanalyse	34

1 Einleitung

Mithilfe dieses Dokumentes dokumentieren wir unsere 4 Workshops der LB2 vom Modul 346 welche laut LBV 70 Prozent der Modulnote ausmacht. Von den Workshops wurden uns 2 Vorgegeben und 2 durften wir frei wählen. Vorgegeben wurde uns die beiden AWS-Services EC2 und RDS. Selbst ausgewählt haben wir die AWS-Services S3 und Amazon ECS. Beim EC2 Service haben wir uns dazu entschieden einen Minecraft Server, ein Pi-Hole und einen VPN einzurichten. Mit RDS müssen wir eine Datenbank mit 2 Tabellen erstellen, mit S3 wollen wir eine NextCloud aufsetzen und mit ECS möchten wir eine kleine Wordpress-Seite hosten.

Alle Workshops werden mit AWS durchgeführt. Dafür verfügen alle 3 Lernenden über einen Account welchen uns Zugriff auf das AWS Learner Lab gibt. Pro Person haben wir in diesem Learner Lab 100 Dollar zu Verfügung. In unserem Fall haben wir uns dazu entschieden nur einen dieser Accounts für unser fertiges Projekt zu nutzen damit wir alles zusammen haben was uns auch in den Modulunterlagen empfohlen wurde. Damit jedoch auch jeder in der Gruppe Zugriff auf diesen Account hat, haben wir ein Dummy Passwort für den Account von Maurice erstellt und die Login Daten geteilt. Mit den folgenden Logindaten kann man sich in diesen Account unter <https://awsacademy.instructure.com/login/canvas> einloggen:

E-Mail: mda133769@iet-gibb.ch

Passwort: sml12345

Viel Spass beim Lesen...

2 EC2

2.1 Minecraft Server

Key	Value
Name des Webservers	MinecraftServer
Offene Ports von aussen	22, 25565
Public Netzwerk, inkl. Gateway und Route	Ja
User	ec2-user
Public Key	-
Software	Minecraft Server
Region	us-east-1
AMI	Amazon Linux (ami-0574da719dca65348)

Wir haben unseren Minecraft Server mithilfe eines Terraform-Skripts erstellt. Dafür haben wir das untenstehende Terraform Skript verwendet. In diesem Terraform Skript erstellen wir eine Sicherheitsgruppe namens "minecraft" welche die TCP Ports 22 und 25565 von aussen für alle öffnet und von innen alle Ports für alle Protokolle öffnet. Danach erstellen wir unseren Server mit dem Namen "minecraft" und wählen das AMI und die Instance Grösse aus. Zudem adden wir die oben erstellte Sicherheitsgruppe und unser Cloud-Init Skript welches nach der Erstellung der Instance ausgeführt wird.

Listing 1: Terraform Script für Minecraft Server

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-east-1"
}

data "template_file" "user_data" {
  template = file("./cloud-init.yaml")
}
```

```

resource "aws_security_group" "minecraft" {
  name      = "minecraft"
  description = "Security group for Minecraft server"

  ingress {
    protocol    = "tcp"
    from_port   = 25565
    to_port     = 25565
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    protocol    = "tcp"
    from_port   = 22
    to_port     = 22
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    protocol    = "-1"
    from_port   = 0
    to_port     = 0
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_instance" "app_server" {
  ami          = "ami-0b5eea76982371e91"
  instance_type = "t2.small"

  vpc_security_group_ids = [aws_security_group.minecraft.id]

  user_data = data.template_file.user_data.rendered

  tags = {
    Name = "MinecraftServer"
  }
}

```

Danach haben wir den Server noch mit einem Cloud-Init Skript konfiguriert. In dem Cloud-Init Skript haben wir folgende Befehle ausgeführt:

- `sudo rpm --import https://yum.corretto.aws/corretto.key:`
Dieser Befehl importiert den GPG-Schlüssel für das Amazon Corretto-Paketrepository. Der GPG-Schlüssel wird verwendet, um die Echtheit und Integrität der heruntergeladenen Pakete zu überprüfen.
- `sudo curl -L -o /etc/yum.repos.d/corretto.repo https://yum.corretto.aws/corretto.repo:`

Dieser Befehl lädt eine Konfigurationsdatei für das Amazon Corretto-Paketrepository herunter und speichert sie im Verzeichnis `/etc/yum.repos.d/` auf dem System.

- **sudo yum install -y java-17-amazon-corretto-devel:**
Dieser Befehl installiert das Amazon Corretto 17-Entwicklungskit (JDK) auf dem System.
- **sudo adduser minecraft:**
Dieser Befehl fügt einen Minecraft-Benutzer hinzu.
- **sudo su:**
Mit diesem Befehl wird eine Superuser-Sitzung gestartet.
- **mkdir /opt/minecraft/:**
Dieser Befehl erstellt das Verzeichnis `/opt/minecraft/`.
- **mkdir /opt/minecraft/server/:**
Dieser Befehl erstellt das Verzeichnis `/opt/minecraft/server/`.
- **cd /opt/minecraft/server:**
Mit diesem Befehl wechselt man in das Verzeichnis `/opt/minecraft/server/`.
- **wget https://launcher.mojang.com/v1/objects/c8f83c5655308435b3dcf03c06d9fe8740a77469/server.jar:**
Mit diesem Befehl wird die Datei `server.jar` vom Minecraft-Server heruntergeladen und im aktuellen Verzeichnis gespeichert.
- **sudo chown -R minecraft:minecraft /opt/minecraft/:**
Mit diesem Befehl wechselt man den Eigentümer vom oben zu sehenden Verzeichniss zum User Minecraft.
- **java -Xmx1024M -Xms1024M -jar server.jar nogui:**
Mit diesem Befehl versuchen wir den Minecraft Server zu starten. Die wird jedoch nicht funktionieren da wir der EULA noch nicht zugestimmt haben. Diese wir somit nun erstellt und im selben Verzeichniss abgelegt.
- **sudo sed -i 's/eula=false/eula=true/' eula.txt:**
Mit diesem Befehl ersetzen "eula=false" zu "eula=true" im File `eula.txt`. Die bewirkt dass die EULA nun angenommen wurde.

Jetzt ist der Server bereit zu starten. Dies können mit folgendem Befehl erreichen: `"sudo java -Xmx1024M -Xms1024M -jar server.jar nogui"`. Man benötigt jetzt nur noch einen Minecraft-Client und einen Minecraft Account und man kann auf dem Server spielen. Mit der IP welche man in AWS auf der Instance nachschauen kann, kann man dem Server betreten.

2.2 Pihole

Key	Value
Name des Webservers	PiHoleServer
Offene Ports von aussen	22, 80, 443
Public Netzwerk, inkl. Gateway und Route	Ja
User	ubuntu
Public Key	pihole key
Software	Pi-Hole
Region	us-east-1
AMI	Ubuntu (ami-0574da719dca65348)

Auch den Server für das Pi-Hole haben wir mit der Hilfe eines Terraform Skriptes ausgesetzt. Dafür haben wir das untenstehende Skript verwendet. Auch dieses mal haben wir wieder eine Sicherheitsgruppe erstellt. Diese Sicherheitsgruppe öffnet die Ports 22, 80 und 443 von aussen für alle. Und auch wieder von innen werden alle Ports für alle Protokolle geöffnet. Danach erstellen wir wieder unsere Instance, wählen das Amazon Machine Image und die gröss der Instance. Zudem fügen wir unser Keypair und die Sicherheitsgruppe hinzu. Zum Schluss adden wir noch das Cloud-Init Skript welches nach dem die Instance fertig aufgesetzt ist ausgeführt wird.

Listing 2: Terraform Script für Pi-hole Server

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.16"
    }
  }
}

required_version = ">= 1.2.0"

provider "aws" {
  region = "us-east-1"
}

data "template_file" "user_data" {
  template = file("./cloud-init.yaml")
}

resource "aws_security_group" "pi_hole" {
  name = "pi_hole"
```

```

description = "Security group for Pi-Hole server"

ingress {
  protocol   = "tcp"
  from_port  = 80
  to_port    = 80
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "tcp"
  from_port  = 443
  to_port    = 443
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "tcp"
  from_port  = 22
  to_port    = 22
  cidr_blocks = ["0.0.0.0/0"]
}

egress {
  protocol   = "-1"
  from_port  = 0
  to_port    = 0
  cidr_blocks = ["0.0.0.0/0"]
}
}

resource "aws_instance" "app_server" {
  ami           = "ami-0574da719dca65348"
  instance_type = "t2.medium"
  key_name      = "pihole key"
  vpc_security_group_ids = [aws_security_group.pi_hole.id]

  user_data = data.template_file.user_data.rendered

  tags = {
    Name = "PiHoleServer"
  }
}

```

Die folgenden Befehle werden im Cloud-Init Skript ausgeführt und bringen die Ubuntu Instance auf den neusten Stand und führen die ersten Schritte für die Pi-Hole Einrichtung aus.

- **sudo apt update:**
Aktualisiert die Liste der verfügbaren Pakete und ihrer Versionen, die von den Paketquellen im System verwendet werden.
- **sudo apt upgrade -y:**
Aktualisiert installierte Pakete auf ihre neuesten verfügbaren Versionen. Die Option **-y** sagt dem System, alle Abhängigkeitsfragen automatisch mit **"Ja"** zu beantworten.
- **sudo hostnamectl:**
Set-hostname AWS_PiHole ändert den Hostnamen des Systems in **"AWS_PiHole"**.

Von diesem Punkt an konnte wir leider nicht mehr mit einem Skript arbeiten da für die Pi-Hole Installation einige Installer mit "Pop-Ups" auftreten bei welchen man manuell bestätigen muss und wir leider keinen Weg fanden dies zu umgehen. Deshalb wurden die folgenden Befehle manuell auf der Instance ausgeführt.

- **sudo curl -sSL https://install.pi-hole.net | bash:**
Lädt ein Skript von der angegebenen URL herunter und führt es aus. Dieses Skript installiert Pi-hole auf dem System.
- **pihole -a -p:**
Startet den Pi-hole-Assistenten und führt ihn im **"privaten"** Modus aus. Im **"privaten"** Modus werden keine Informationen über die Konfiguration an den Pi-hole-Server gesendet.

Mit der IP, welche man in AWS auf der Instance nachschauen kann, kann man nun über einen Browser auf das PI-Hole Admin Dashboard zugreifen. Um sich einzuloggen verwendet man als Passwort **"sml12345"**.

2.3 VPN

Key	Value
Name des Webservers	OpenVpnServer
Offene Ports von aussen	22, 80, 443, 1194
Public Netzwerk, inkl. Gateway und Route	Ja
User	ubuntu
Public Key	vpn
Software	Open Vpn
Region	us-east-1
AMI	Ubuntu (ami-0574da719dca65348)

Wie auch schon für die anderen EC2 Instances haben wir auch für den VPN ein Terraform-Skript verwendet. Als erstes erstellen wir wieder eine Sicherheitsgruppe mit dem Namen "vpn" welche von aussen die Ports 22, 80, 443 und 1194 öffnet. Von Innen darf natürlich auf alles zugegriffen werden. Danach erstellen wir die eigentliche Instance. Für diese verwenden wir wie beim Pi-Hole ein Ubuntu Image mit der Grösse t2.micro. Ausserdem weisen wir noch die oben erstellte Sicherheitsgruppe zu, ein keypair welches wir bereits in aws erstellt haben und das Cloud-Init Skript welches die Maschine aktualisiert.

Listing 3: Terraform Script für VPN Server

```
terraform {
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 4.16"
    }
  }
}

required_version = ">= 1.2.0"

provider "aws" {
  region = "us-east-1"
}

data "template_file" "user_data" {
  template = file("./cloud-init.yaml")
}

resource "aws_security_group" "vpn" {
  name        = "vpn"
  description = "Security group for OpenVpn server"
```

```

ingress {
  protocol   = "tcp"
  from_port  = 80
  to_port    = 80
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "tcp"
  from_port  = 443
  to_port    = 443
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "tcp"
  from_port  = 22
  to_port    = 22
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "tcp"
  from_port  = 1194
  to_port    = 1194
  cidr_blocks = ["0.0.0.0/0"]
}

ingress {
  protocol   = "udp"
  from_port  = 1194
  to_port    = 1194
  cidr_blocks = ["0.0.0.0/0"]
}
egress {
  protocol   = "-1"
  from_port  = 0
  to_port    = 0
  cidr_blocks = ["0.0.0.0/0"]
}
}

resource "aws_instance" "app_server" {
  ami           = "ami-0574da719dca65348"
  instance_type = "t2.micro"
  key_name      = "vpn"
  vpc_security_group_ids = [aws_security_group.vpn.id]

  user_data = data.template_file.user_data.rendered
}

```

```
tags = {
  Name = "OpenVpnServer"
}
}
```

Die folgenden Befehle werden im Cloud-Init Skript, welches wir oben geaddet haben, ausgeführt und bringen die Ubuntu Instance auf den neusten stand..

- **sudo apt update:**
Aktualisiert die Liste der verfügbaren Pakete und ihrer Versionen, die von den Paketquellen im System verwendet werden.

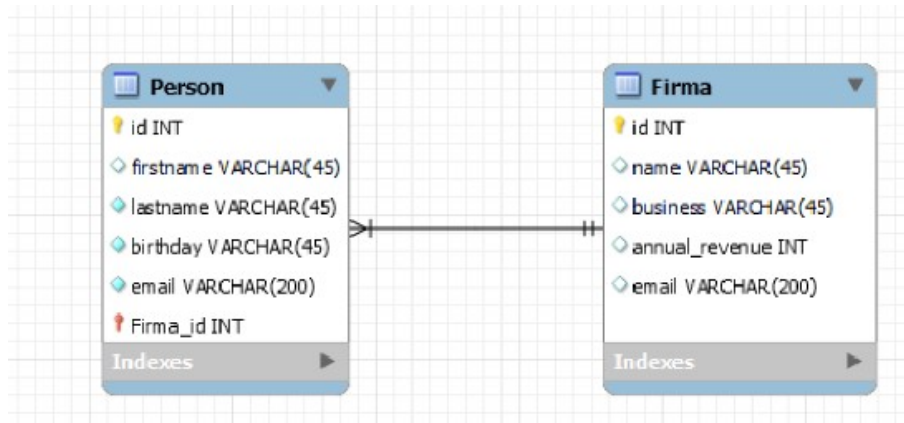
Ansonsten haben wir leider nichts ins Cloud-Init Skript schreiben können da wir die Installer Pop-Ups nicht automatisieren können. Die folgenden Befehle haben wir manuell auf der EC2 Instance ausgeführt:

- **sudo apt update:**
Aktualisiert die Liste der verfügbaren Pakete und ihrer Versionen, die von den Paketquellen im System verwendet werden.
- **sudo apt upgrade -y:**
Dieser Befehl aktualisiert die Liste der verfügbaren Pakete und führt ein Upgrade der bereits installierten Pakete durch. Der Parameter -y bestätigt automatisch alle Abfragen während des Upgrades.
- **wget https://raw.githubusercontent.com/angristan/openvpn-install/master/openvpn-install.sh**
Mit diesem Befehl wird eine Kopie der angegebenen Datei von GitHub heruntergeladen.
- **chmod +x openvpn-install.sh:**
Mit diesem Befehl wird die ausführbare Berechtigung für die heruntergeladene openvpn-install.sh Datei gesetzt.
- **./openvpn-install.sh:**
Dieser Befehl startet die openvpn-install.sh Skript, welches eine OpenVPN Server Installation automatisch durchführt.
- **ss -tupln | grep openvpn:**
Dieser Befehl zeigt alle aktiven OpenVPN-Verbindungen an. ss zeigt die aktuellen TCP und UDP-Verbindungen an, -t filtert nur TCP-Verbindungen, -u filtert nur UDP-Verbindungen, -p zeigt die Prozess-ID und -n zeigt die IP-Adressen anstelle von Hostnamen.
- **sudo apt install openvpn:**
Dieser Befehl installiert OpenVPN auf dem System.
- **openvpn --config /path/to/configuration.ovpn:**
Dieser Befehl startet den OpenVPN-Daemon mit der angegebenen Konfigurationsdatei.

3 Datenbank

Key	Value
Typ der Datenbank	MySQL
Name der Datenbank	database-01
Benutzername	admin
Passwort	Sml12345

Wir haben mit dem AWS Service RDS eine Datenbank mit 2 Tabellen erstellt. Beide dieser Tabellen enthalten je 4 Attribute und einen Primary Key. Zudem enthält eine der Tabellen noch einen Foreign Key. Um diese Struktur etwas besser darzustellen haben wir folgendes ERD erstellt:



Da man nicht einfach ein Bild von einem ERD pasten kann um eine Datenbank zu erstellen haben wir ein SQL-Skript geschrieben. Dieses SQL-Skript überprüft ob es die Datenbank schon gibt und wenn nicht erstellt es diese. Danach wählen wir die gerade erstellte Datenbank mit "USE" aus. Nun erstellen wir die beiden Tabellen welche oben im ERD zu sehen sind. Dafür erstellen wir eine Tabelle falls diese nicht schon existiert und geben darin den Name des Attributes an, den Datentypen und Constraints an. Bei Datentypen haben wir zum einen INT gewählt was eine Ganzzahl ist. Zudem haben wir uns für VARCHAR entschieden was Text darstellt. Die Zahl in der Klammer danach steht für die Länge des Textes. Bei den Constraints haben wir uns beim Primary Key für AUTOINCREMENT entschieden was soviel heisst dass bei jeder Entität die Zahl um 1 erhöht wird. NOT NULL steht dafür dass der Wert nicht Nichts sein darf, also einen Wert haben muss. Zum Schluss geben wir noch an dass die Id den Primary Key sein soll. Bei der Person Tabelle geben wir noch zusätzlich den FOREIGN KEY Constraint an und geben mit worauf sich dieser Fremdschlüssel bezieht.

Listing 4: SQL Script für die Datenbank

```
CREATE DATABASE IF NOT EXISTS mydatabase;

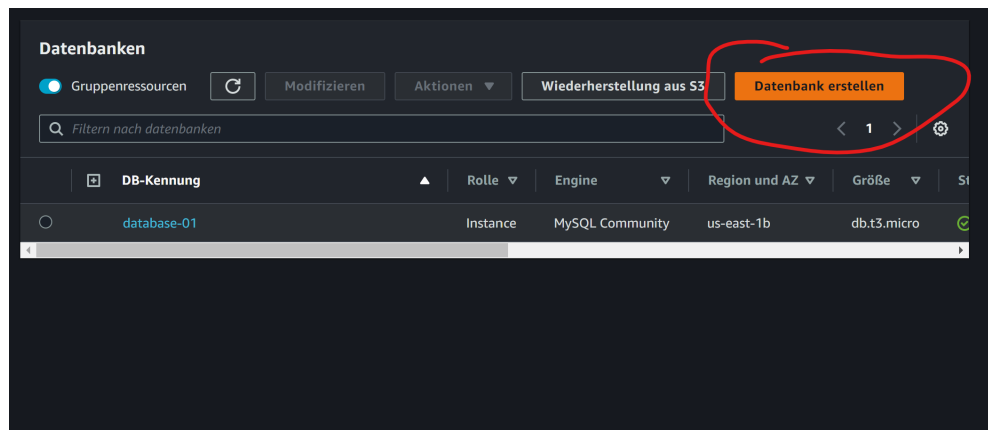
USE mydatabase;

CREATE TABLE IF NOT EXISTS Firma (
  id INT NOT NULL AUTO_INCREMENT,
  name VARCHAR(255) NOT NULL,
  business VARCHAR(255) NOT NULL,
  annual_revenue INT NOT NULL,
  email VARCHAR(255) NOT NULL,
  PRIMARY KEY (id)
);

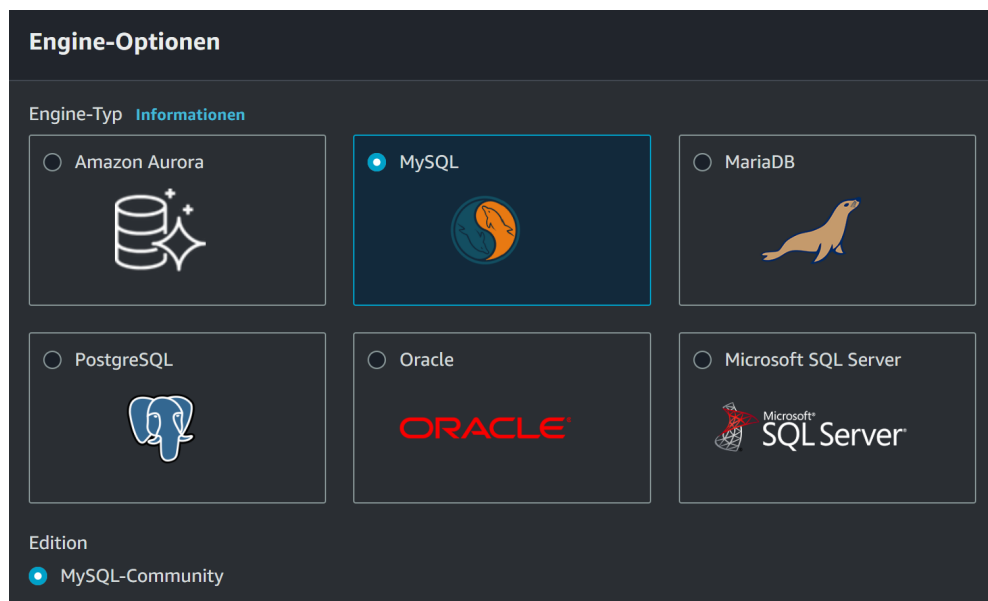
CREATE TABLE IF NOT EXISTS Person (
  id INT NOT NULL AUTO_INCREMENT,
  firstname VARCHAR(255) NOT NULL,
  lastname VARCHAR(255) NOT NULL,
  birthday DATE NOT NULL,
  email VARCHAR(255) NOT NULL,
  Firma_id INT NOT NULL,
  PRIMARY KEY (id),
  FOREIGN KEY (Firma_id) REFERENCES Firma(id)
);
```

Kommen wir zur Erstellung der RDS Instance. Dabei behandeln werden nur die Einstellungen genannt welche wir verändert haben. Falls eine Einstellungsmöglichkeit wie die Verison nicht explizit von uns genannt wird, wird davon ausgegangen dass die Standartwerte verwendet werden.

- Als erstes Suchen wir in der AWS Suchleiste oben links nach "RDS" und wählen RDS aus
- Nun wählen wir auf der Linken Seite "Datenbanken" aus
- Danach klicken wir auf "Datenbank erstellen"



- Unter "Engine-Optionen" wählen wir "MySQL"



- Unter "Vorlagen" wählen wir "Kostenloses Kontingent"

Vorlagen

Wählen Sie eine Beispielvorlage aus, die zu Ihrem Anwendungsfall passt.

☐ **Produktion**
 Verwenden Sie Standardwerte für hohe Verfügbarkeit und schnelle, kontinuierliche Leistung.

☐ **Entwicklung/Test**
 Diese Instance dient zur Verwendung für die Entwicklung außerhalb einer Produktionsumgebung.

☒ **Kostenloses Kontingent**
 Mit dem kostenlosen Kontingent für RDS können Sie neue Anwendungen entwickeln, vorhandene Anwendungen testen oder praktische Erfahrung mit Amazon RDS sammeln. [Informationen](#)

- Unter "Einstellungen" geben wir unserer DB den Namen "database-01"
- Unter "Einstellungen" geben wir nun unser gewünschtes Passwort ein, nämlich "Sml12345" und bestätigen dieses

Einstellungen

DB-Instance-Kennung [Informationen](#)
 Geben Sie einen Namen für Ihre DB-Instance ein. Der Name muss für alle DB-Instances Ihres AWS-Kontos in der aktuellen AWS-Region eindeutig sein.

database-01

Bei der DB-Instance-Kennung wird zwischen Groß- und Kleinschreibung unterschieden, sie wird jedoch komplett in Kleinbuchstaben gespeichert (wie z. B. in „meinedbinstance“). Einschränkungen: Zwischen 1 und 60 alphanumerische Zeichen oder Bindestriche. Muss mit einem Buchstaben beginnen. Darf keine zwei aufeinanderfolgenden Bindestriche enthalten. Darf nicht mit einem Bindestrich enden.

▼ **Einstellungen für Anmeldeinformationen**

Hauptbenutzername [Informationen](#)
 Geben Sie eine Anmelde-ID für den Hauptbenutzer Ihrer DB-Instance ein.

admin

1 bis 16 alphanumerische Zeichen. Muss mit einem Buchstaben beginnen.

☐ **Manage master credentials in AWS Secrets Manager - new**
 Manage master user credentials in Secrets Manager. RDS can generate a password for you and manage it throughout its lifecycle.

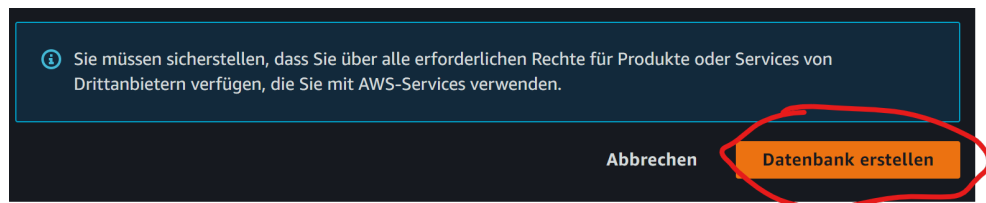
☐ **Ein Passwort automatisch erstellen**
 Amazon RDS kann ein Passwort für Sie generieren oder Sie können Ihr eigenes Passwort angeben.

Hauptpasswort [Informationen](#)

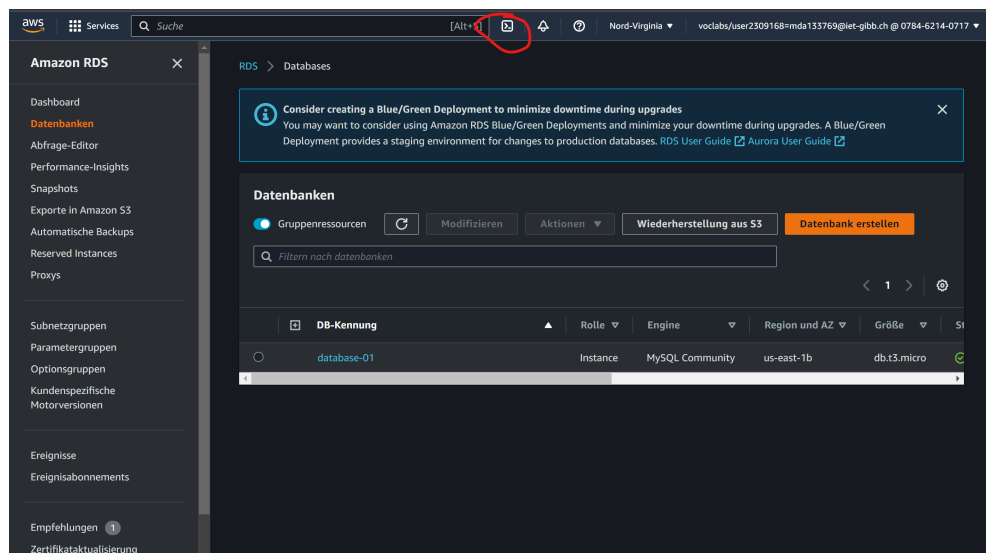
.....

Einschränkungen: Mindestens 8 druckbare ASCII-Zeichen. Folgende Zeichen dürfen nicht enthalten sein: / (Schrägstrich), ' (einzelne Anführungszeichen), " (doppelte Anführungszeichen) und @ (At-Zeichen).

- Unter "Anbindung " beim Punkt "Öffentlicher Zugriff" wählen wir "Ja" damit wir auch von ausserhalb des VPC auf unsere DB zugreifen können
- Zum Schluss der Erstellung klicken wir nun "Datenbank erstellen" und warten darauf dass die DB gestartet ist



- Nun wählen wir auf der Linken Seite "Datenbanken" aus
- Danach klicken wir auf unsere DB und kopieren unter "Endpunkt und Port" den Endpunkt
- Nun klicken wir oben rechts, links von der Glocke auf das console-Icon



- Um uns mit der DB zu verbinden geben wir folgenden Befehl ein:
"mysql -h <endpoint> -u admin -p". <endpoint> wird jedoch durch
die Adresse des Endpoint welche wir vorher kopiert haben ersetzt.
- Nun geben wir noch unser Passwort ein, was in unserem Fall "Sml12345"
ist
- Jetzt sind wir mit der DB verbunden, da wir jedoch noch keine
Tabellen haben kopieren wir das SQL-Skript von oben, fügen es
ein und bestätigen dass wir dies auch wirklich wollen und drücken
ENTER
- Um zu überprüfen ob es geklappt hat können wir folgenden Befehl
eingeben und es sollten uns unsere 2 Tabellen angezeigt werden:
"show tables;"

```

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mydatabase |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.02 sec)

MySQL [(none)]> use mydatabase;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [mydatabase]> select * from Person;
+-----+-----+-----+-----+-----+-----+
| id | firstname | lastname | birthday | email | Firma_id |
+-----+-----+-----+-----+-----+-----+
| 1 | Maurice | Däppen | 0000-00-00 | mda@ims.ch | 1 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

MySQL [mydatabase]> select * from Firma;
+-----+-----+-----+-----+-----+
| id | name | business | annual_revenue | email |
+-----+-----+-----+-----+-----+
| 1 | IMS AG | Software | 5000000 | info@ims.ch |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

MySQL [mydatabase]>

```

Zudem kann man sich z.B. mit MySQLWorkbench auf die DB verbinden und änderungen vornehmen.

4 S3

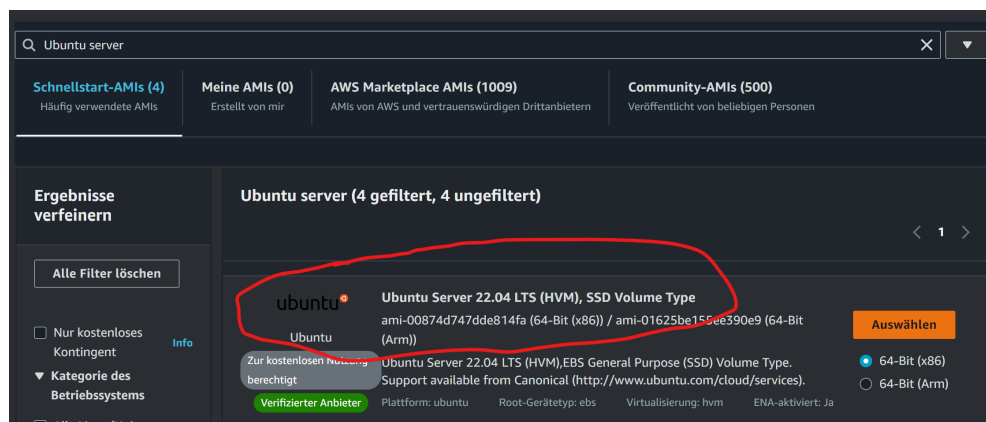
4.1 Beschreibung

Mit Amazon Simple Storage Service, kurz S3, möchten wir eine Nextcloud erstellen. Da das S3 nur ein Speicher ist benötigen wir dazu noch eine EC2 Instance welche das GUI der Nextcloud zu Verfügung stellt und über welches wir auf unseren S3 Bucket zugreifen können und auf diesem Daten verwalten können.

4.2 Installation

Um nun das Ganze zu erstellen, loggen wir uns als erstes im Learner Lab ein und starten dieses. Nun besuchen wir die aws console.

- Als erstes suchen wir nun in der Suchleiste nach "EC2" und wählen diesen Dienst aus.
- Klicke rechts oben auf den "Instance Starten"-Button.
- Gib unter "Name und Tags" den Namen "next-cloud-01"
- Unter "Anwendungs- und Betriebssystemabbilder" drückst du auf "Weitere AMI's durchsuchen"
- Suche in der Suchleiste nach "Ubuntu Server" und wähle diesen aus:



- Unter "Instance-Typ" wählst du "t3.micro" aus.
- Beim Schlüsselpaar wählst du dir ein Schlüsselpaar deiner Wahl aus oder erstellst eines.
- Unter den "Netzwerkeinstellungen" wählst du folgende Konfiguration aus:

▼ **Netzwerkeinstellungen** Info

Bearbeiten

Netzwerk Info

vpc-0c3795fe7d60da7de

Subnetz Info

No preference (Default subnet in any availability zone)

Öffentliche IP automatisch zuweisen Info

Aktivieren

Firewall (Sicherheitsgruppen) Info

☒ Sicherheitsgruppe erstellen

☐ Vorhandene Sicherheitsgruppe auswählen

Wir erstellen eine neue Sicherheitsgruppe namens 'launch-wizard-10' mit den folgenden Regeln:

☒ Datenverkehr von SSH zulassen
Hilft Ihnen, eine Verbindung zu Ihrer Instance herzustellen

Überall

0.0.0.0/0

☒ HTTPS-Datenverkehr aus dem Internet zulassen
Um einen Endpunkt einzurichten, zum Beispiel beim Erstellen eines Webservers

☒ HTTP-Datenverkehr aus dem Internet zulassen
Um einen Endpunkt einzurichten, zum Beispiel beim Erstellen eines Webservers

⚠

Regeln mit der Quelle 0.0.0.0/0 erlauben es allen IP-Adressen, auf Ihre Instance zuzugreifen. Wir empfehlen Ihnen, die Regeln Ihrer Sicherheitsgruppe so einzurichten, dass sie den Zugriff nur von bekannten IP-Adressen aus zulassen.

×

- Alle anderen Einstellungen kannst du so lassen wie sie sind. Klicke nun auf "Instance erstellen"

- Kehre nun auf die Übersicht deiner Instances zurück. Nun solltest du deine neue Instance sehen.
- Klicke auf der Linken Seite auf "Elastic Ip's" und danach auf "Elastic IP Adresse zuweisen"
- Scrolle nach unten und drücke "Zuweisen".
- Wähle nun deine Elastic IP aus und drücke "Elastic IP Adresse zuordnen", wähle deine Instance aus und drücke "zuordnen"
- Kehre nun auf die Übersicht deiner Instances zurück. Nun solltest du deine neue Instance sehen.
- Wenn deine EC2 Instance läuft klicke auf ihre Id und danach auf "Verbinden".
- Wähle "EC2 Instance Verbindung" aus und klicke "Verbinden":

Mit Instance verbinden [Info](#)

Stellen Sie mithilfe einer dieser Optionen eine Verbindung zur Instance i-08778660dd703c142 (next-cloud-01) her

EC2-Instance-Verbindung | Session Manager | SSH-Client | Serielle EC2-Konsole

Instance-ID
i-08778660dd703c142 (next-cloud-01)

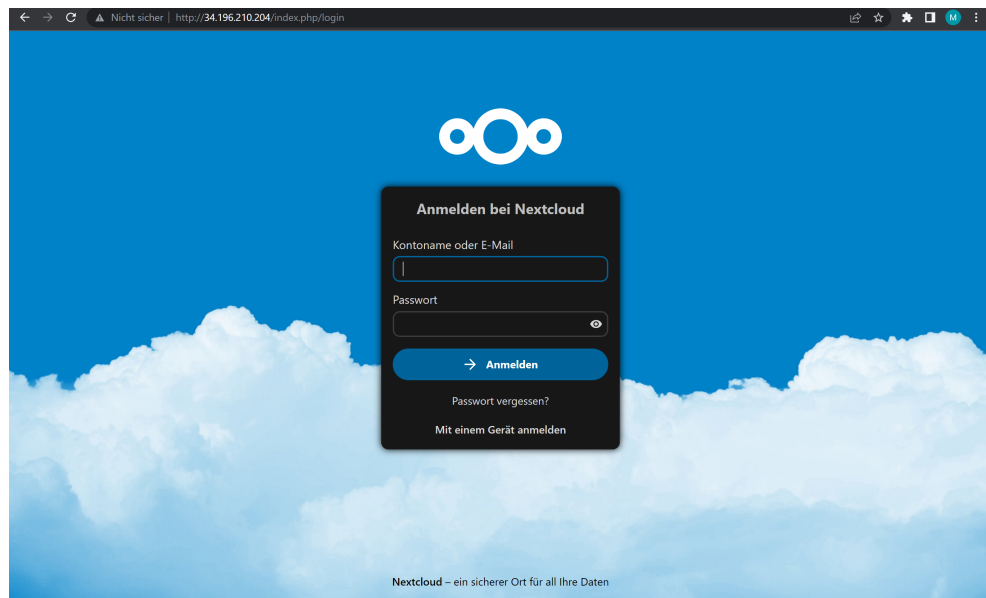
Öffentliche IP-Adresse
34.196.210.204

Benutzername
Enter the user name defined in the AMI used to launch the instance. If you didn't define a custom user name, use the default user name, ubuntu.

Note: In most cases, the default user name, ubuntu, is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Abbrechen **Verbinden**

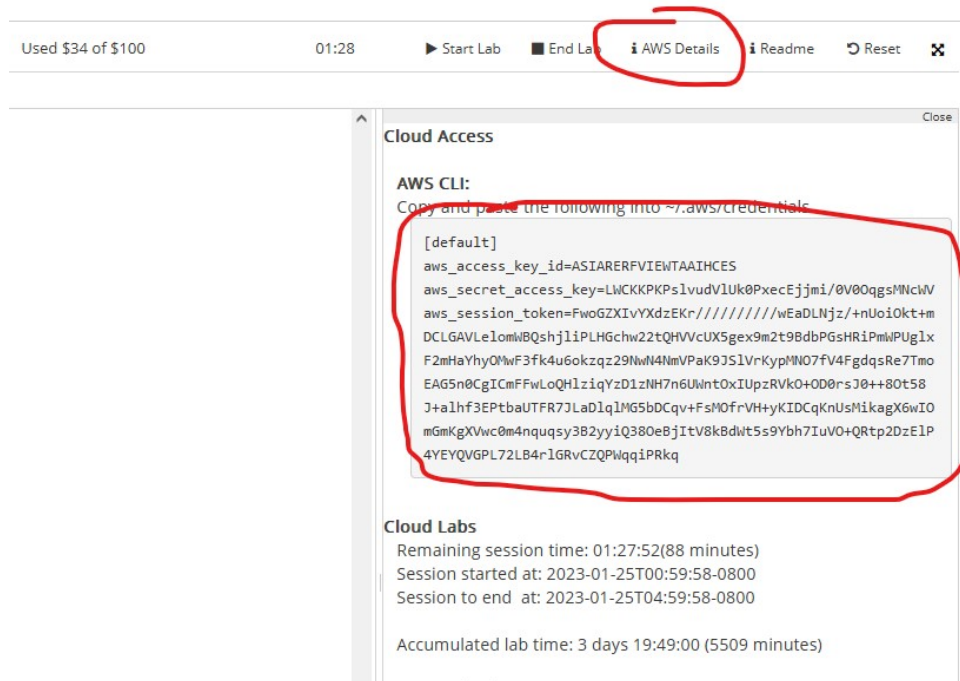
- Nun solltest du dich in einem Terminal im Browser befinden. Installiere die Nextcloud Software mit folgendem Command "sudo snap install nextcloud"
- Gebe nun folgenden Command ein: "sudo nextcloud.manual-install <adminusername> <adminpassword>". adminusername und adminpassword kannst du selbst wählen, jedoch merke dir was du gewählt hast.
- Als nächstes nutzen wir folgenden Command um Nextcloud zu sagen von welcher IP wir zugreifen möchten: "sudo nextcloud.occ config:system:set trusted_domains 1 --value=<dns-domain>". Ersetze "dns-domain" durch deine Elastic IP.
- Zum Schluss nutzen wir noch diesen Command: "sudo snap set nextcloud nextcloud.cron-interval=10m".
- Nun kannst du mit http über deine Elastic IP im Brower auf deine Nextcloud verbinden. Dies sollte etwa so aussehen:



- Nun kannst du dich mit den Daten anmelden welche du vorher im Command ersetzt hast, in unserem Fall als Name "admin" und als Passwort "sml12345".
- Drücke nun oben Rechts auf deinen Account und wähle "Apps". Danach "Deaktivierte Apps". Aktiviere nun "External storage support".
- Wechsle zurück in deine AWS Console und suche in der Suchleiste nach "S3" und wähle diesen Dienst aus.
- Klicke nun auf "Bucket erstellen" und gib diesem einen Einzigartigen Namen.
- Die andere Einstellungen kannst du lassen wie sie sind. Scrolle nach unten und drücke "Bucket erstellen."
- In der Nextcloud kannst du nun nochmals auf dein Profil klicken und danach "Administratoreinstellungen" und danach "Externer Speicher"
- Nehme dort folgende Konfiguration vor. Wobei du "your-bucket-name" durch den vorher gewählten Bucketname ersetzt.

Ordnername	Externer Speicher	Authentifizierung	Konfiguration	Verfügbar für	
AmazonS3	Amazon S3	Zugangsschl...	<input type="text" value="your-bucket-name"/> <input type="text" value="Host-Name"/> <input type="text" value="Port"/> <input type="text" value="Region"/>	<input type="checkbox"/> SSL aktivieren <input type="checkbox"/> Pfad-Stil aktivieren <input type="checkbox"/> Legacy-Authentifizierung (V2) <input type="text" value="Zugangsschlüssel"/> <input type="text" value="Geheimer Schlüssel"/>	<input type="button" value="Alle Benutzer. Benutz"/> ...

- Den Zugangsschlüssel und geheimen Zugangsschlüssel findest du im Learner Lab:



4.3 Verfügbarkeit

In Bezug auf die Verfügbarkeit wurde die Nextcloud-Instanz auf einer EC2-Instanz in der Zone US-EAST1 bereitgestellt. Es wurde nur ein AWS-Account verwendet. Obwohl die Verwendung mehrerer Zonen und Accounts die Verfügbarkeit weiter erhöhen kann, gewährleistet die Verwendung einer Zone und eines Accounts immer noch eine hohe Verfügbarkeit der Nextcloud-Instanz.

4.4 Datensicherheit

In Bezug auf die Datensicherheit werden die Daten der Nextcloud auf einem S3-Storage gespeichert. Dieser bietet eine hohe Redundanz und Verfügbarkeit der Daten, um sicherzustellen, dass die Daten auch im Falle eines Ausfalls des Storage-Systems weiterhin verfügbar sind. Leider haben wir keine Wiederherstellung eingerichtet. Diese Möglichkeit besteht jedoch durch die Verwendung von S3, da es automatisch Backups und Wiederherstellungsmöglichkeiten für Daten bereitstellt.

4.5 Kostenanalyse

Wir haben für die Kostenanalyse den AWS Pricing Calculator verwendet. Wir gehen davon aus dass wir eine EC2 von der Grösse t3.micro, eine Elastic Ip und einen S3 Bucket verwenden. Der S3 Bucket hat eine Grösse 20 GB. Andere Services werden für die Nextcloud nicht verwendet.

Übersicht über die Schätzung

Informationen

Vorabkosten

123,52 USD

Monatliche Kosten

2,41 USD

Gesamtkosten 12 months

152,44 USD

Einschließlich Vorabkosten

Vertrieb kontaktieren

Melden Sie sich bei der Konsole an

My Estimate

Duplikat

Löschen

Wechseln zu

Gruppe erstellen

Support hinzufügen

Service hinzufügen

Find resources

<1>

<input type="checkbox"/>	Servicename	Vorabkosten	Monatliche Kosten	Beschreib...	Region	Config Su...
<input type="checkbox"/>	Amazon Simple Storage S... ✎	0,00 USD	0,49 USD	-	USA Ost (Nord-Virginia)	S3-Standardspei...
<input type="checkbox"/>	Amazon EC2 ✎	123,52 USD	1,92 USD	-	USA Ost (Nord-Virginia)	Tenancy (Gemei...
<input type="checkbox"/>	Amazon Elastic IP ✎	0,00 USD	0,00 USD	-	USA Ost (Nord-Virginia)	Anzahl der EC2-I...

Vorabkosten entstehen keine für dieses kleine Projekt. Die Elastic Ip ist kostenlos da wir nur eine für die EC2 Instance verwenden. Die Nextcloud Software welche wir nutzen ist auch kostenlos. Für die EC2 Instance bezahlen wir 7.59 Dollar pro Monat. Das S3 Bucket ist erstaunlich Günstig mit einem Preis von 49 Mill im Monat. Zusammen mach das 8.08 Dollar im Monat und 96.98 Dollar im Jahr.

5 Amazon Elastic Container Service

5.1 Beschreibung

Mit Amazon Elastic Container Service, kurz ECS, wollen wir einen Wordpress Blog mithilfe von EC2 Instances hosten. Da jedoch sind fast alle in öffentlichen Foren wie StackOverflow, in Blogposts oder in YT-Tutorials der Meinung dass man das Ganze am Besten mit Fargate anstatt mit EC2 umsetzt. Deshalb benutzen wir in unserem Projekt für unseren Wordpress-Blog die AWS Dienste ECS, ECR, und Fargate.

5.2 Installation

Um nun das Ganze zu erstellen, loggen wir uns als erstes im Learner Lab ein und starten dieses. Nun besuchen wir die aws console.

- Als erstes suchen wir nun in der Suchleiste nach "ECR" und wählen diesen Dienst aus.
- Auf der Rechten Seite klicken wir nun auf "Repository erstellen"

- Nun nehmen wir folgende Konfiguration vor und drücken auf "Repository erstellen":

Repository erstellen

Allgemeine Einstellungen


Sichtbarekeitseinstellungen [Info](#)
Wählen Sie die Sichtbarekeitseinstellung für das Repository.

☒ **Privat**
Der Zugriff wird durch IAM und Repository-Richtlinienberechtigungen verwaltet.

☐ **Öffentlich**
Öffentlich sichtbar und für Image-Pulls zugänglich.


Repository-Name
Geben Sie einen präzisen Namen an. Ein Entwickler sollte in der Lage sein, die Repository-Inhalte anhand des Namens zu identifizieren.

078462140717.dkr.ecr.us-east-1.amazonaws.com/

 Ein Repository mit diesem Namen ist bereits vorhanden
14 von maximal 256 Zeichen (mindestens 2). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

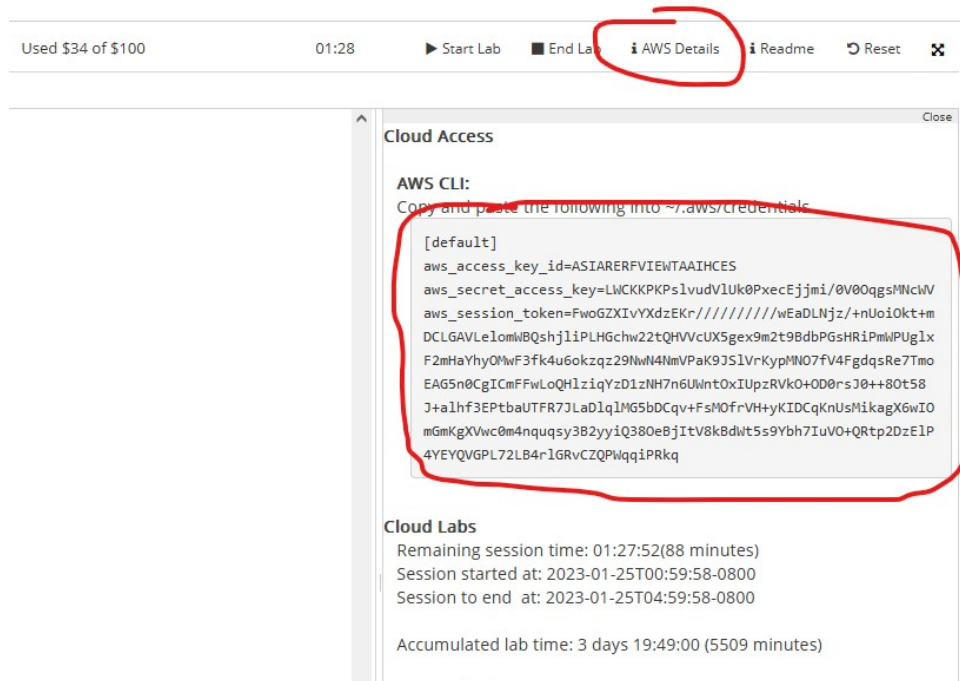
Unveränderlichkeit von Tags [Info](#)
Aktivieren Sie die Unveränderlichkeit von Tags, um zu verhindern, dass Image-Tags durch nachfolgende Image-Pushes mit demselben Tag überschrieben werden. Deaktivieren Sie die Unveränderlichkeit von Tags, damit Image-Tags überschrieben werden können.

☐ **Deaktiviert**

 Sobald ein Repository erstellt wurde, kann die Sichtbarekeitseinstellung des Repositorys nicht mehr geändert werden.

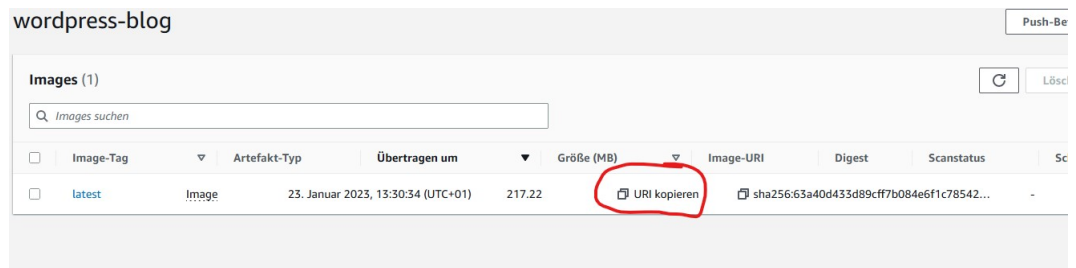
- Stelle sicher dass du den AWS CLI installiert hast, falls nicht
downloade diesen unter "<https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html>".

- Stelle nun sicher dass du den die credentials vom Learner Lab ins "credentials"-File in deinem Userhome im ".aws"-Ornder kopierst. Die Credentials findest du im Learner Lab, dort wo du auch dieses startest unter "AWS Details":

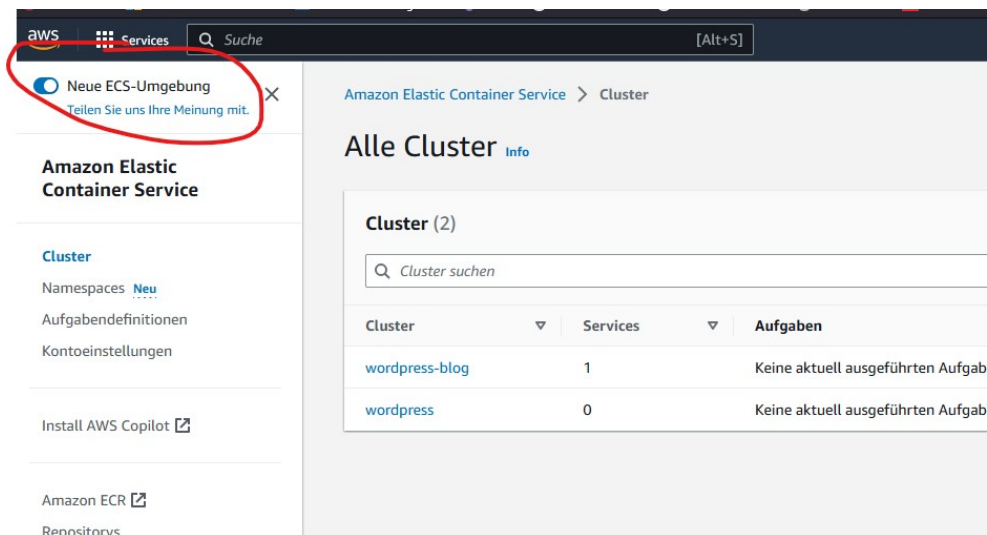


- Öffne eine Lokale Konsole und gebe folgenden befehl ein "aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin ACCOUNTID.dkr.ecr.us-east-1.amazonaws.com" Für ACCOUNTID gibst du deine Account Id ein welche du unter anderm in deinem Profil nachscheun kannst.
- Als nächstes nutzen wir folgenden Command um das offizielle Dockerimage von Dockerhub zu ziehen: "docker pull wordpress"
- Nun möchten wir dieses Image zu ECR pushen. Dies können wir mit diesem Command "docker tag wordpress ACCOUNTID.dkr.ecr.us-east-1.amazonaws.com/sample-wordpress:latest" gefolgt von diesem "docker push ACCOUNTID.dkr.ecr.us-east-1.amazonaws.com/sample-wordpress:latest" erreichen. Auch hier wieder ACCOUNTID durch deine eigene Account-Id ersetzten.

- Nun kannst du in deinem ECR-Repository, welches wir erstellt haben das Image sehen. Kopiere davon die URL da wir diese später benötigen:



- Suche nun in der Suchleiste oben links nach "ECS" und wähle diesen Dienst aus.
- Falls es oben in der Mitte keinen Button mit "Erste Schritte" hat, dann toggle den Toggle-Button oben links welcher mit "Neue ECS-Umgebung" beschriftet ist:



- Klicke nun auf "Erste Schritte" und wähle "custom" aus und klicke auf "konfigurieren".

- Gebe nun folgende Konfiguration an wobei du IMAGE URL durch die vorher kopierte Image URL ersetzt und drücke unten rechts auf aktualisieren:

Container bearbeiten

▼ Standard

Containername*

wordpress-blog ⓘ

Bild*

{IMAGE_URL} ⓘ

Authentifizierung des privaten Repositorys*

☐

 ⓘ

Speicherlimits (MiB)

Weiches ... ▼

128 ⓘ

[Hartes Limit hinzufügen](#)

Definieren Sie harte und/oder weiche Speicherlimits in MiB für Ihren Container. Hard- und Soft-Limits entsprechen den Parametern „memory“ bzw. „memoryReservation“ in Aufgabendefinitionen. ECS empfiehlt 300 bis 500 MiB als Ausgangspunkt für Webanwendungen.

Port-Zuweisungen

Container-Port

80

Protokoll

tcp ⓘ

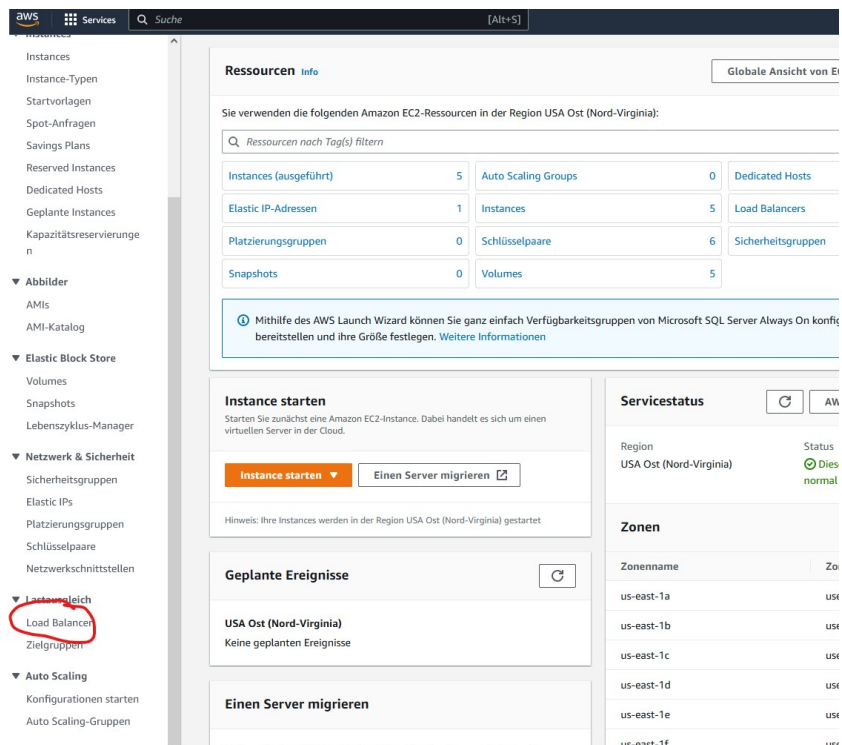
[Port-Zuweisung hinzufügen](#)

- Unter "Aufgabendefinition" klickst du auf bearbeiten und änderst den Namen der Aufgabendefinition zu "wordpress-blog" und setzt die "Aufgabenausführungsrolle" zu "LabRole" und speichere und drücke "weiter".
- Klicke nun auf weiter.
- Wähle unter "Load Balancer" Application Load Balancer aus und klicke "weiter" an.
- Jetzt kannst du dir noch einen Cluster-Namen aussuchen und weiter klicken

- Jetzt wird dir die Gesamte konfiguration angezeigt. Scroll nach unten und erstelle den Cluster. Dies solle nun etwa so aussehen:



- Nun wollen wir noch wissen wie wir überhaupt daraus zugreifen können. Dafür suchst du in der Suchleiste oben rechts nach EC2 und wählst diesen Dienst aus.
- Auf der Linken Seite wählst du nun "Load Balancer" aus:



- Nun klickst du auf den Namen deines Load Balancers:

Load balancers (1)
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter by property or value

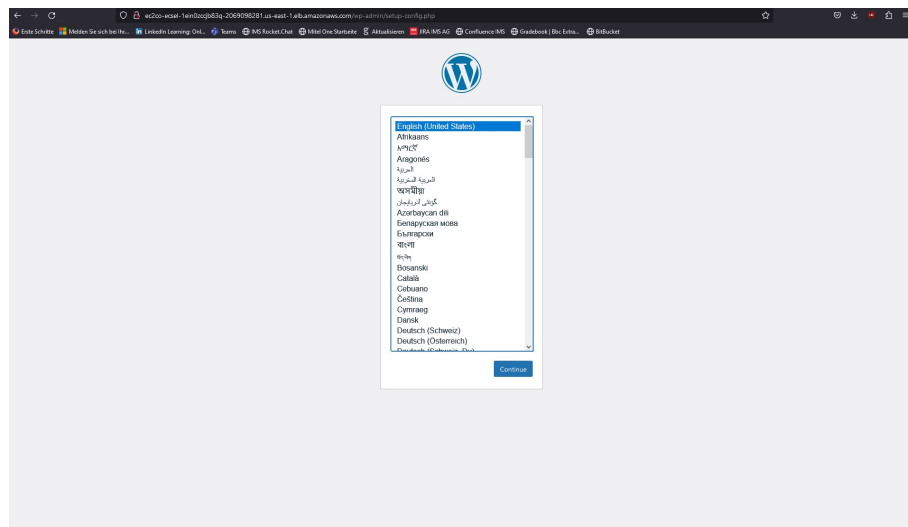
<input type="checkbox"/>	Name	DNS name	State	VPC ID	Availability Zones	Type
<input type="checkbox"/>	EC2Co-EcsEL-1E1NOZCCJB83Q	EC2Co-EcsEL-1E1NOZCCJB83Q...	Active	vpc-09bb65d8828338581	2 Availability Zones	application

- Der DNS Name ist nun die URL mit welcher du auf deine Wordpresseite zugreifen kannst:

Details
arn:aws:elasticloadbalancing:us-east-1:078462140717:loadbalancer/app/EC2Co-EcsEL-1E1NOZCCJB83Q/a102d99d50bcb653

Load balancer type Application	DNS name EC2Co-EcsEL-1E1NOZCCJB83Q-2069098281.us-east-1.elb.amazonaws.com (A Record)	Status Active	VPC vpc-09bb65d8828338581
IP address type IPv4	Scheme Internet-facing	Availability Zones subnet-0f26065aedfcd2d82 us-east-1b (use1-az1) subnet-00183d16196acf385 us-east-1a (use1-az6)	Hosted zone Z355XDOTRQ7X7K
Date created January 23, 2023, 14:26 (UTC+01:00)			

- Wenn wir nun auf die Seite zugreifen sollte es folgendermassen aussehen:



5.3 Verfügbarkeit

Der Cluster und die dazugehörigen Komponenten sind nur in der Region us-east-1 verfügbar. Dies schliesst jedoch nicht aus dass man ihn von der ganzen Welt aus anschauen kann, die Latenzen sind einfach höher. Zudem kann man nur Änderungen an den Komponenten mit dem in der Einleitung genannten Account vornehmen.

5.4 Datensicherheit

Bezüglich der Datensicherheit fallen keine grosse Risiken auf. Den der Cluster auf welchem der Blog läuft wurde mit einem Wordpress-Image aufgesetzt welches man jederzeit auf Dockerhub wieder bekommen kann. Der Blog selbst würde auf einem RDS System laufen welches selbst eine Backup-Funktion bietet was einen Datenverlust fast zu 100 Prozent ausschliesst.

5.5 Kostenanalyse

Wir haben unsere Kostenanalyse mit dem AWS Pricing Calculator für die Region us-east-1 also North-Virginia erstellt. Dabei haben wir die Services ECS, Fargate und RDS eingebunden. Du wirst dich warscheinlich fragen "RDS?". Ja, wenn wir wirklich einen Blog hosten möchten brauchen wir noch eine DB. Da diese jedoch nicht zu unserem Service gehört haben wir diese ausgelassen denn wir haben RDS schon verwenden und haben auch nicht genügend Zeit für alles da die Nextcloud schon extrem viel Zeit in anspruch genommen hat. Für die DB haben wir angenommen dass wir eine MySQL DB mit 10 GB storage und nochmal 10 GB für ein Backup nutzen damit wir unsere Daten sicher können. Bei ECS und Fargate haben wir mit Durchschnittlich 10 Connections pro Tag gerechnet.

Somit sind wir auf folgende Kosten gekommen. Pro Monat würden wir für ECS 38 Mill, für Fargate 39 Mill und für RDS 28.01 Dollar bezahlen. Zusammen mach das 28.78 Dollar pro Monat. Im Jahr sind das 345.42 Dollar.

Estimate summaryInfo

Upfront cost

0.00 USD

Monthly cost

28.78 USD

Total 12 months cost

345.42 USD

Includes upfront cost

Getting Started with AWS

Contact Sales

Sign in to the Console

My Estimate

Find resources

Duplicate

Delete

Move to

Create group

Add support

Add service

<input type="checkbox"/>	Service Name	<input type="checkbox"/>	Upfront cost	<input type="checkbox"/>	Monthly cost	<input type="checkbox"/>	Description	<input type="checkbox"/>	Region	<input type="checkbox"/>	Config Sa...
<input type="checkbox"/>	Amazon Elastic Container Registry	<input checked="" type="checkbox"/>	0.00 USD		0.38 USD		-		US East (N. Virginia)		DT Inbound: Int...
<input type="checkbox"/>	AWS Fargate	<input checked="" type="checkbox"/>	0.00 USD		0.39 USD		-		US East (N. Virginia)		Operating syste...
<input type="checkbox"/>	Amazon RDS for MySQL	<input checked="" type="checkbox"/>	0.00 USD		28.01 USD		-		US East (N. Virginia)		Storage for each...