Milestone 1: Infrastruktur-Spezifikation

Hammerschmidt, Rentenberger, Schodl, Weidinger

19. Dezember 2024

Inhaltsverzeichnis

1 Netzwerktopologie

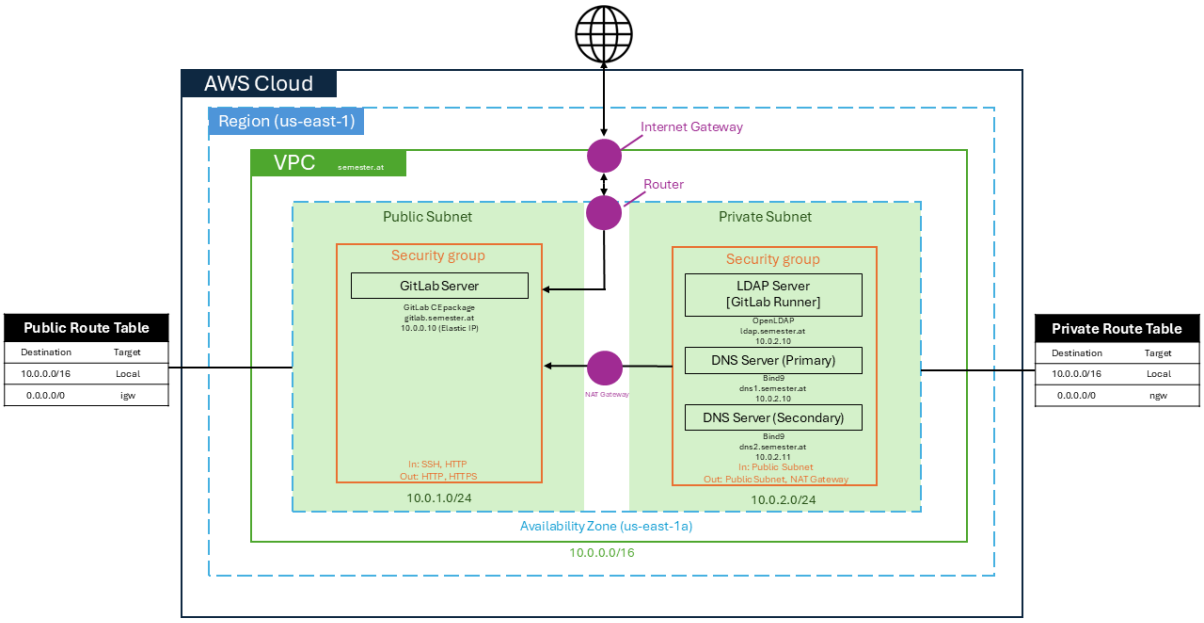

Abbildung 1: Netzwerktopologie der Infrastruktur

Dienst	**Subnetztyp**	**IP-Adresse**	**FQDN**
Primärer DNS	Privates Subnetz	10.0.2.225	ns1.semester.devops.com
Sekundärer DNS	Privates Subnetz	10.0.2.10	ns2.semester.devops.com
GitLab Server	Öffentliches Subnetz	Public	gitlab.semester.devops.com
LDAP Server	Privates Subnetz	10.0.2.10	ldap.semester.devops.com
GitLab-Runner	Privates Subnetz	10.0.2.184	gitlab.semester.devops.com

Tabelle 1: Netzwerkdienste und IP-Zuordnung

2 Geplante Security Groups und Regeln

Inbound SGRs	**DNS**	**LDAP/GitLab-R**	**GitLab-Server**
HTTP	Nein	Nein	Ja
HTTPS	Nein	Ja	Ja
SSH	Ja	Ja	Ja
DNS (UDP)	Ja	Nein	Nein
DNS (TCP)	Ja	Nein	Nein
LDAP	Nein	Ja	Ja
ALL ICMP	Ja	Nein	Ja

Tabelle 2: Eingehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste

Outbound SGRs	**DNS**	**LDAP/GitLab-R**	**GitLab-Server**
HTTP	Nein	Ja	Nein
HTTPS	Nein	Ja	Nein
SSH	Nein	Ja	Nein
DNS (UDP)	Nein	Ja	Nein
DNS (TCP)	Nein	Ja	Nein
LDAP	Nein	Ja	Ja
ALL ICMP	Nein	Ja	Nein
ALL Traffic	Ja	Nein	Ja

Tabelle 3: Ausgehende Sicherheitsgruppenregeln (SGRs) für verschiedene Dienste

3 Spezifikationen der eingesetzten Systeme

Server	OS	Packages	Version	Server Instance
Primärer DNS Server	Ubuntu Server	bind9, bind9utils	BIND 9 / Ubuntu 24.04 LTS	T3.micro
Sekundärer DNS Server	Ubuntu Server	bind9, bind9utils	BIND 9 / Ubuntu 24.04 LTS	T3.micro
LDAP Server	Ubuntu Server	slapd, ldap-utils	OpenLDAP 2.6	T3.micro
GitLab Runner	Ubuntu Server	-	Ubuntu 24.04 LTS	T3.micro
GitLab Server	Ubuntu Server	GitLab CE	GitLab CE / Ubuntu 24.04 LTS	T2.Large

Tabelle 4: Server-Spezifikationen: Betriebssystem, Pakete und Instanztypen

- **Betriebssystem**: Ubuntu 24.04 LTS LTS (64-bit)
- **DNS-Server**: BIND 9.x
- **GitLab**: GitLab CE 15.x
- **GitLab Runner**: Version kompatibel mit GitLab CE 15.x
- **LDAP**: OpenLDAP 2.6.x
- **Monitoring**: AWS CloudWatch zur Protokollierung und Überwachung.

4 Tests

DNS Resolution Testing

- **Ziel:** Sicherstellen, dass der BIND-Server Domain-Namen korrekt auflöst.
- **Methode:** Verwenden des `dig`-Befehls, um den DNS-Server nach bekannten Domains abzufragen. Überprüfen der `A`, `AAAA`, `MX` und `NS` Records:

```
dig @<DNS-server> example.com [A, AAAA, MX, NS]
```

- **Erwartetes Ergebnis:** Jede Abfrage liefert die richtigen IP-Adressen und Record-Details.

Forward and Reverse DNS Lookup

- **Ziel:** Überprüfen, dass Vorwärts- und Rückwärts-Abfragen funktionieren.
- **Methode:**
  - Verwenden von `dig` für die Vorwärtsabfrage (Domain zu IP):

```
dig @<DNS-server> example.com A
```

  - Verwenden von `dig -x` für die Rückwärtsabfrage (IP zu Domain):

```
dig @<DNS-server> -x [192.0.2.1]
```

- **Erwartetes Ergebnis:** Genaues Mapping zwischen Domain-Namen und IP-Adressen.

Zone Transfer Test

- **Ziel:** Sicherstellen, dass Zonentransfers zwischen primären und sekundären DNS-Servern funktionieren.
- **Methode:** Einen Zonentransfer mit `dig AXFR` anstoßen und die Logs auf den Transfer überprüfen:

```
dig @<primary-DNS-server> example.com AXFR
```

- **Erwartetes Ergebnis:** Zonendaten werden korrekt zwischen primären und sekundären Servern repliziert.

DNS Failover Testing

- **Ziel:** Die Resilienz und Zuverlässigkeit des DNS-Dienstes unter Ausfallbedingungen bewerten.
- **Methode:**
  - Einen Ausfall des primären DNS-Servers simulieren.
  - Die Antwort des sekundären DNS-Servers überwachen:

    ```
    dig @<secondary-DNS-server> example.com A
    ```

  - Etwaige Ausfallzeiten während des Übergangs protokollieren.
- **Erwartetes Ergebnis:** Der sekundäre DNS-Server übernimmt nahtlos mit wenig bis gar keiner Unterbrechung der DNS-Auflösung.

Stress Testing

- **Ziel:** Die Leistung des Servers unter hoher Last testen.
- **Methode:** Tools wie `dnsperf` verwenden, um eine hohe Anzahl von DNS-Abfragen zu simulieren:

  ```
  dnsperf -s <primary-DNS-IP> -d queries.txt -l 30
  ```

- **Erwartetes Ergebnis:** Der Server bleibt auch unter Last genau und leistungsfähig.

subsection*LDAP-Authentifizierungstest

- **Ziel:** Funktionalität des Authentifizierungssystems überprüfen.
- **Methode:** Benutzeranmeldungen über LDAP versuchen. Tests mit gültigen und ungültigen Anmeldedaten durchführen.
- **Erwartetes Ergebnis:** Anmeldeversuche sollten für gültige Anmeldedaten akzeptiert und für ungültige Anmeldedaten abgelehnt werden.

LDAP-Suche und -Filterung

- **Ziel:** Genauigkeit der LDAP-Suche und -Filterung bestätigen.
- **Methode:** Suchen nach bestimmten Benutzergruppen oder Attributen durchführen und Filter testen.
- **Erwartetes Ergebnis:** Für jede Suche und jeden Filter werden die korrekten Daten zurückgegeben.

Benutzer- und Gruppenverwaltung

- **Ziel:** Testen der Erstellung und Änderung von Benutzern und Gruppen.
- **Methode:** Gruppen und Benutzer erstellen sowie deren Attribute ändern. Änderungen im LDAP-Verzeichnis überwachen.
- **Erwartetes Ergebnis:** Alle Änderungen werden korrekt im LDAP-Verzeichnis angezeigt.

LDAP-Integrationstest

- **Ziel:** Testen, ob die Integration der GitLab-Authentifizierung mit LDAP funktioniert.
- **Methode:** Anmeldungen bei GitLab mit LDAP-Anmeldedaten (verschiedene Rollen) durchführen.
- **Erwartetes Ergebnis:** Benutzeranmeldungen werden mit LDAP-Anmeldedaten akzeptiert.

Repository-Operationen

- **Ziel:** Testen, ob Standard-Git-Operationen innerhalb von GitLab funktionieren.
- **Methode:** Neben der Erstellung von Repositories werden Pull-, Push- und Merge-Operationen getestet. Zusätzlich werden Branches erstellt und gelöscht.
- **Erwartetes Ergebnis:** Änderungen entsprechen den erwarteten Ergebnissen der Git-Operationen.

CI/CD-Pipeline-Test

- **Ziel:** Funktionalität der CI/CD-Pipeline überprüfen.
- **Methode:** Geänderten Code pushen und die automatische Ausführung der CI/CD-Pipeline beobachten. Erfolg des Builds und der Bereitstellung prüfen.
- **Erwartetes Ergebnis:** Codeänderungen werden automatisch gebaut und fehlerfrei bereitgestellt.

Lasttest

- **Ziel:** Leistung von GitLab unter hoher Last bewerten.
- **Methode:** Mehrere Benutzer simulieren, die gleichzeitig Git-Operationen durchführen.
- **Erwartetes Ergebnis:** GitLab hält die Leistungsniveaus auch bei gleichzeitiger Nutzung aufrecht.

5 Rollen und Verantwortlichkeiten im Team

Samuel Hammerschmidt	**Lorenz Rentenberger**	**Nikolas Schodl**	**Alexander Weidinger**
LDAP Server	DNS Server (CloudWatch)	GitLab Server	GitLab Server
AWS Cloud Config	AWS Cloud Config	AWS Cloud Config	AWS Cloud Config
Tests LDAP	Tests DNS	Tests GitLab	Tests GitLab

Tabelle 5: Team-Aufgaben und Zuständigkeiten

6 Monitoring

AWS CloudWatch wird zur Protokollierung und Überwachung genutzt:

- Überwachung der CPU-, Speicher- und Netzwerknutzung.
- Automatische Alarme bei Ausfällen.

7 Create VPC

Um ein VPC zu erstellen müssen wir zu erst Sicherstellen, dass wir in der richtigen Region sind. In unserem Fall ist das üs-east-1". —SCREENSHOT—

8 Creating Subnets

8.1 Private Subnet

Zuerst erstellte wir ein privates Subnets. Wie im Bild unten angeführt, wählen wir die richtigen VPC(10.0.0.0/16) aus und konfigurieren die private CIDR-Block(10.0.2.0/24). —SCREENSHOT—

8.2 Public Subnet

Beim Erstellen vom Public Subnet, gehen wir die selbe Schritte durch, doch ändern den CIDR-Block auf 10.0.1.0/24. —SCREENSHOT—

8.2.1 Enabling Auto-assign IP for Public Subnet

Wichtig zum Erwähnen ist die Aktivierung der Option Ënable auto-assign public IPv4 address". Das sorgt dafür, dass jeder neuerstellten EC2 Instanz eine neue IP Adresse erstellt wird. —-SCREENSHOT ODER KURZ ERKLÄREN—-

9 Internet Access

9.1 Enabling Internet Access for Public Subnet

Doch eine Subnet Public zu setzen reicht nicht, Sie muss auch einen Internetzugriff bekommen. Dafür verwenden wir die 'route-table', die wiederum Zugriff auf das Internet Gateway gewährt.

9.1.1 Internet Gateway

Zuerst erstellen wir ein Gateway, mit dem Namen 'Semester-igw'. Dem Gateway geben wir zusätzlich Tags, um später die Suche von diesem zu erleichtern.

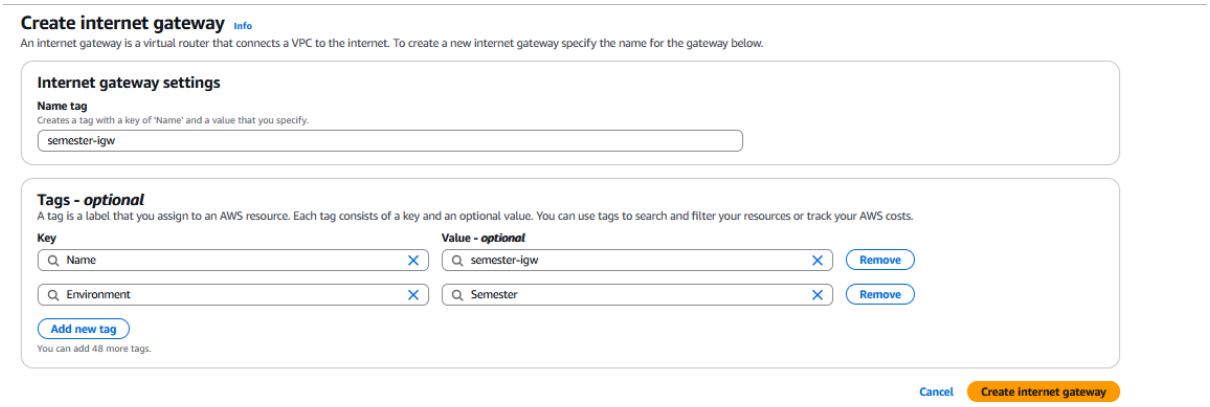

Abbildung 2: Internet Gateway erstellen

Das Gateway ist erstellt, aber ist keinem VPC zugewiesen. Beim Gateway unter 'Actions' fügen wir dies hinzu.

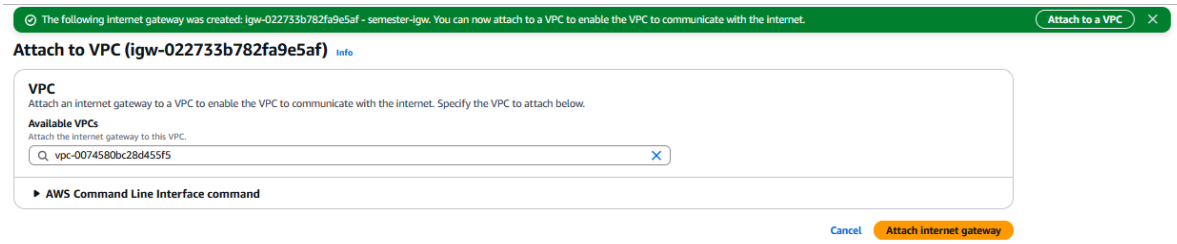

Abbildung 3: Internet Gateway einem VPC hinzufügen

9.1.2 Routing Table

Jetzt aktualisieren wir eine neue Routing Table, damit das Gateway Internetzugriff hat. In dem Tab "Route Tablesëditieren wir die Table mit dem Wert —SUCHE WERT—-". —-SCHRITT MIT "TAG"GEMACHT ODER NICHT????—-

Nun selectieren wir die Table.

—-WAS PASSIERT HIER????—-

9.2 Enabling Internet Access for Private Subnet

Das private Subnet hat keinen direkten Zugriff, sondern nutzt ein NAT Gateway vom Public Subnet.

9.2.1 NAT Gateway

—-SCREENSHOT—-

9.2.2 Routing Table

Wie beim Public Subnet, muss nun die Routing Table aktualisieren. —-SCREENSHOT—-

10 Security Groups

Für das Projekt haben wir mehrere Security Groups erstellt.

10.1 Public GitLab

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	80	0.0.0.0/0	HTTP
Inbound	TCP	443	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH(GitLab)
Inbound	TCP	2424	My IP	SSH(Admin)
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 6: Security Group: Public GitLab

10.2 Private GitLab

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	80	0.0.0.0/0	HTTP
Inbound	TCP	443	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 7: Security Group: Private GitLab

10.3 DNS

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	53	0.0.0.0/0	HTTP
Inbound	TCP	53	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 8: Security Group: DNS

10.4 LDAP

Type	Internet-Protokoll	Port	Source	Desc.
Inbound	TCP	53	0.0.0.0/0	HTTP
Inbound	TCP	53	0.0.0.0/0	HTTPS
Inbound	TCP	22	0.0.0.0/0	SSH
Outbound	ALL	ALL	0.0.0.0/0	Allow all outbound traffic

Tabelle 9: Security Group: LDAP

11 Launch Instance

Die folgenden zwei Screenshots zeigen wie wir zwei Ubuntu-Instanzen für Primary und Secondary DNS erstellen können.

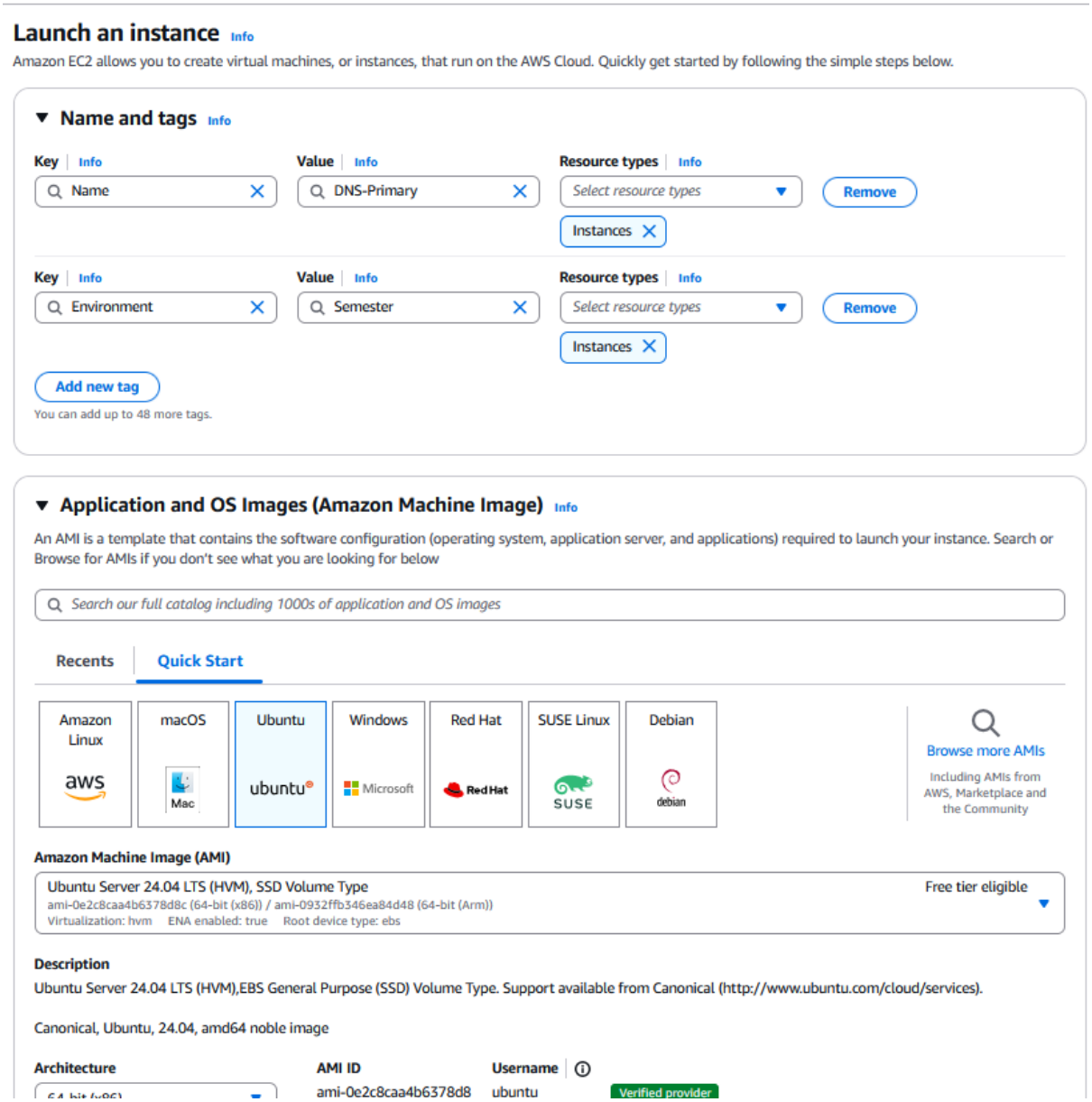

Abbildung 4: Instanzerstellung vom Primary DNS

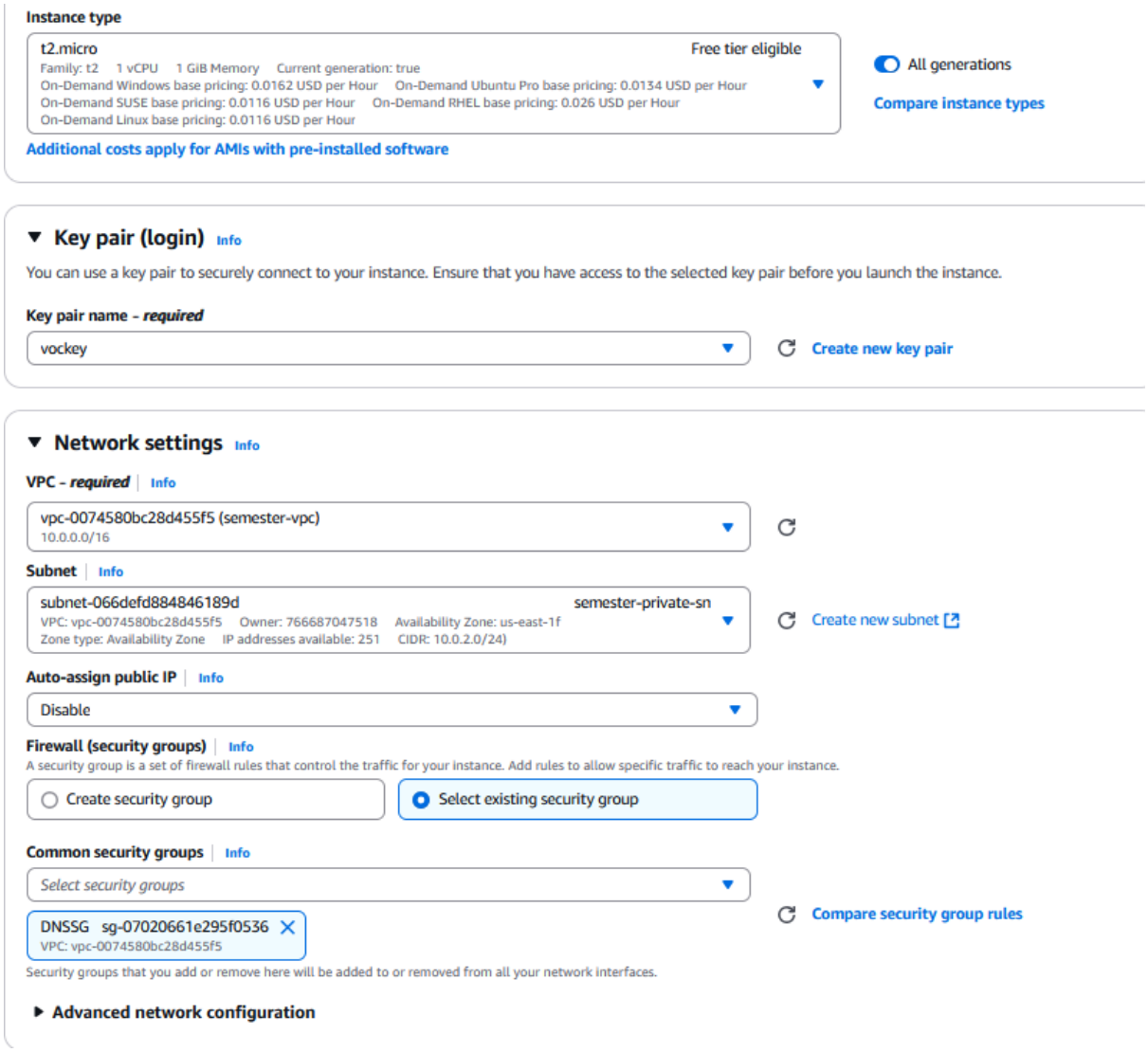

Abbildung 5: Instanzerstellung vom Primary DNS

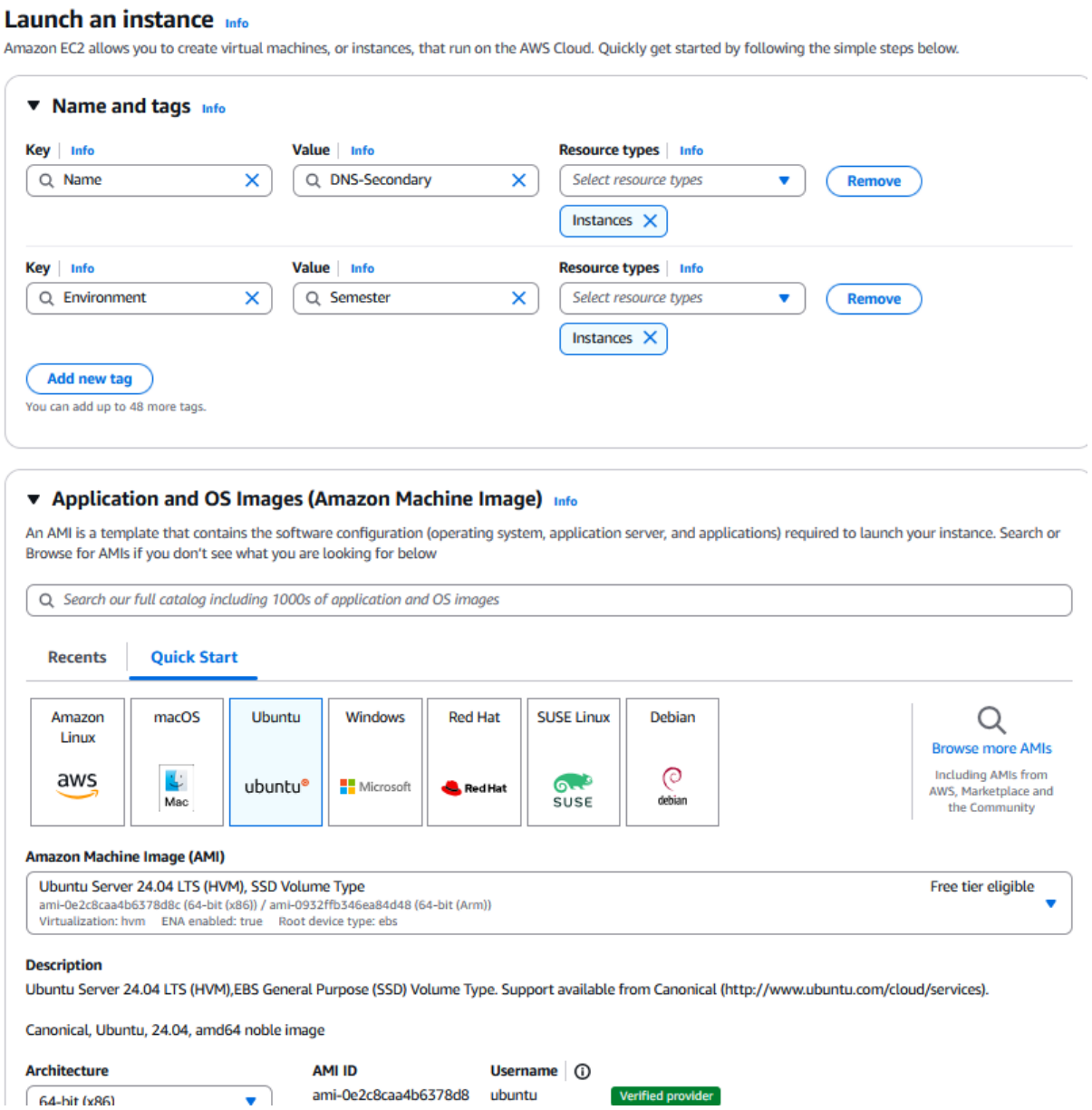

Abbildung 6: Instanzerstellung vom Secondary DNS

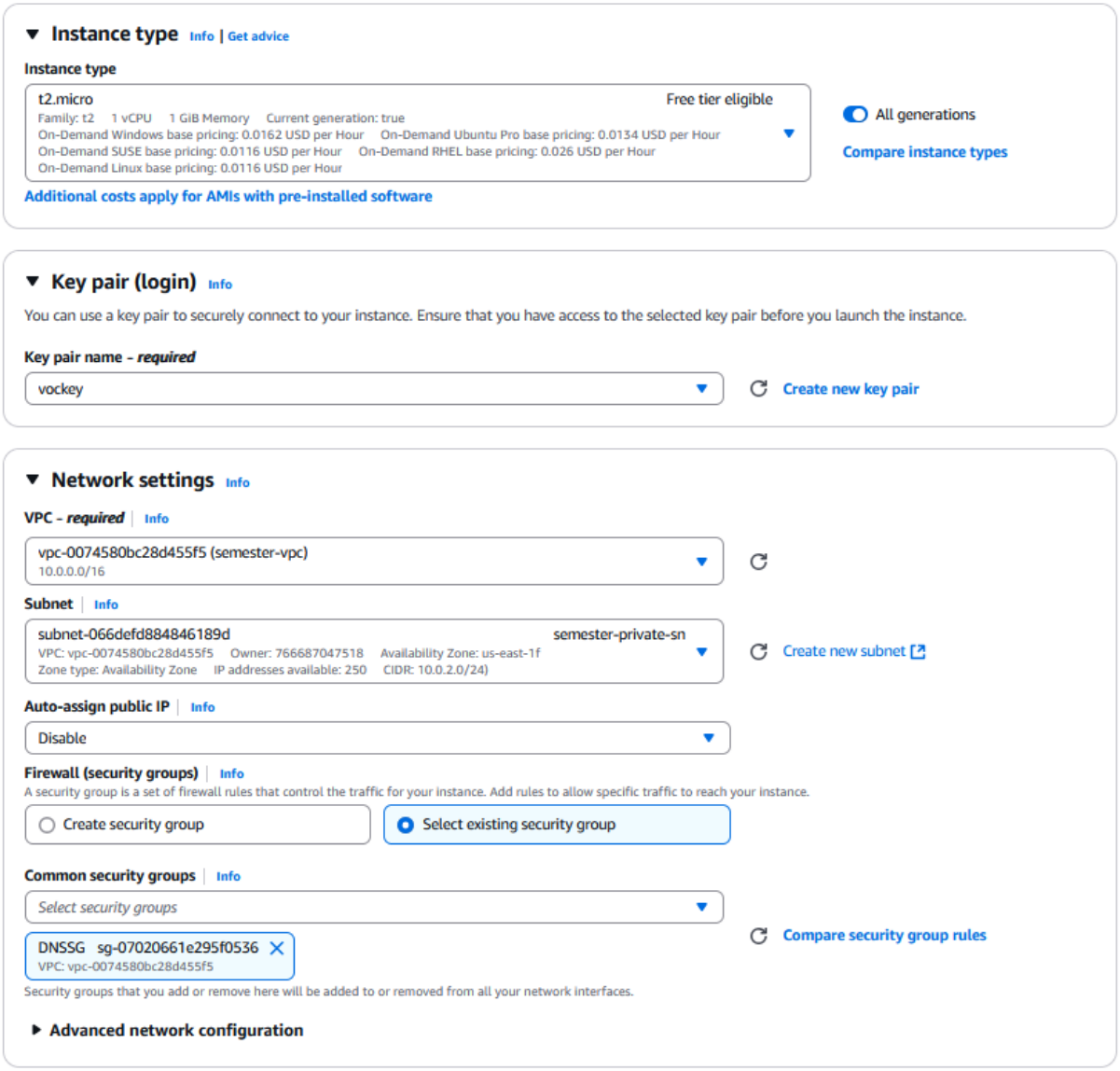

Abbildung 7: Instanzerstellung vom Secondary DNS

12 SSH Access zum Servers in Private Subnet

Weil die server in dem private Subnet keine public IPV4 Adresse haben können wir nicht auf sie direkt mit SSH zugreifen. Wir können um das doch zu erreichen einen Umweg über den GitLab server nehmen, welcher sich im public subnet befindet und eine public IPV4 Adresse hat.

13 Setting up SSH Agent Forwarding

Wir müssen zuerst den SSH Key den wir von AWS Academy bekommen haben hinzufügen.

```
ssh-add. ~/.ssh/labsuser.pem
```

Wir müssen nun SSH Agent Forwarding durch Änderung des files ~ */.ssh/config* und durch den folgenden Command ermöglichen:

```
Host example.com
ForwardAgent yes
```

Nun müssen wir example.com mit der public IPV4 Adresse von GitLab ersetzen.

14 SSH Zugriff

Wir können jetzt SSH verwenden um den GitLab Server zu erreichen.

```
ssh ubuntu@93.71.270.50
```

Public IP vom EC2 GitLab Server und vom GitLab Server zu einem Private Subnet Server springen.

```
ssh ubuntu@10.0.2.225(Private IP des DNS-Primary EC2 Servers)
```

15 Aktualisieren des Betriebssystems

Sobald wir auf dem primary DNS Server sind müssen wir zuerst sicherstellen, dass das Betriebssystem auf dem neusesten Stand ist. Um das zu erreichen führen wir die folgenden Commands aus.

```
sudo apt update && sudo apt upgrade && sudo apt full-upgrade
sudo reboot
```

16 Setup der DNS Server

Nun konfigurieren wir die beiden DNS Server

17 Installation von BIND9

Nachdem wir sichergegangen sind, dass das Betriebssystem auf dem neuesten Stand ist installieren wir BIND9. Eine Software Package mit welchem namens resolution machen können. Das erreichen wir mit dem folgenden command:

```
sudo apt install bind9 bind9utils bind9-doc
```

Wir setzen nun BIND in den IPV4 Modus. Das erreichen wir indem wir das file */etc/default/named ändern* und -4 am Ende vom OPTIONS Parameter hinzufügen.

```
OPTIONS=‘‘-u bind -4’’
```

nun starten wir BIND9 neu

```
sudo systemctl restart bind9
```

18 Konfiguration des Primary DNS Servers

Zunächst konfigurieren wir den primary DNS Server, welcher auf der EC2 Instanz läuft.

19 Erstellung der Access Contol List

Zuerst müssen wir eine ACL(Access Contol List) für den primary DNS Server erstellen. Dadurch können wir kontrollieren wer Zugriff auf den name server hat. Die ACL kann in named.conf.options geändert werden:

```
sudo nano /etc/bind/named.conf.options
```

Über dem Options-Block erstellen wir eine neue Access Control List namens trusted. In dieser benennen wir alle Clients denen wir Zugriff auf den Name Server geben wollen.

```
acl "trusted" {
    10.0.2.184
    10.0.2.225
    10.0.1.246
    10.0.2.10
};
```

20 Konfiguration der Allgemeine Optionen

Nun editieren wir den options block in dem named.conf.options file um generelle Einstellungen festzulegen.

```
options {
  directory "/var/cache/bind";
  recursion yes; # enables recursive queries
  allow-recursion { trusted; }; # allows recursive queries from "trusted" clients
  listen-on { 10.0.0.0/16; }; # VPC CIDR - listen on private network only
  allow-transfer { none; }; # disable zone transfers by default
  forwarders {
   8.8.8.8;
   8.8.4.4;
  };
  ...
};
```

Wenn der DNS Server eine Anfrage bekommt welche er nicht selbst beantworten kann schickt er diese an den unter forwarders definierten Name Server.

20.1 Konfiguration des “Local” Files

In der Local File geben wir die Forward- und Reverse-Zonen an, die den Bereich für die Verwaltung und Definition von DNS-Einträgen festlegen.

```
sudo nano /etc/bind/named.conf.local
```

20.2 Adding the forward zone

```
zone "semesterDevOps.com" {
type primary;
file "/etc/bind/zones/db.semesterDevOps.com";
allow-transfer { 10.0.2.225; };
};
```

type primary → definiert diesen Server als den primary name Server für die Zone. *File* → Pfad zum zone file *allow-transger* → IP Adressen des sekundären Servers welchem es erlaubt ist von diesem Server zu transferieren.

20.3 Hinzufügen der Reverse Zone für das Public Subnet

```
zone "1.0.10.in-addr.arpa" {
   type primary;
   file "/etc/bind/zones/db.10.0.1"; # 10.0.1.0/24 Subnet
   allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary
};
```

Um reverse lookups zu erlauben müssen wir die reverse Zone so konfigurieren, dass wir in der Range unseres Public Subnets sind.

20.4 Adding the reverse file for the Private Subnet

```
zone "2.0.10.in-addr.arpa" {
   type primary;
   file "/etc/bind/zones/db.10.0.2"; # 10.0.2.0/24 Subnet
   allow-transfer { 10.0.2.225; }; # ns2 private IP address - secondary
   };
```

Hier machen wir das selbe wie oben für das Private Subnet.

20.5 Kreieren des Forward Zone Files

Die Forward-Nonendatei ist der Ort, an dem wir DNS-Einträge für Forward-DNS-Abfragen definieren. Das bedeutet, wenn der DNS eine Namesabfrage erhält, wird er in der Forward-Zonendatei nachschauen, um die entsprechende private IP-Adresse von host1 zu ermitteln. Zuerst müssen wir das Directory kreieren. Hier werden wir auch die forward zone files einfügen:

```
sudo mkdir /etc/bind/zones
```

Jetzt kreieren wir das zone file:

```
sudo nano /etc/bind/zones/db.semesterDevOps.com

;
; BIND data file for local loopback interface
;
$TTL 604800
```

```
@ IN SOA ns1.semesterDevOps.com. admin.semesterDevOps.com. (
    3 ; Serial
    604800 ; Refresh
    86400 ; Retry
    2419200 ; Expire
    604800 ) ; Negative Cache TTL
;

; name servers - NS records
  IN NS  ns1.semesterDevOps.com.
  IN  NS  ns2.semesterDevOps.com.

; name servers - A records
ns1.semesterDevOps.com.  IN A  10.0.2.225
ns2.semesterDevOps.com.  IN A  10.0.2.10

; 10.0.0.0/16 - A records
gitlab.semesterDevOps.com. IN  A  10.0.1.246
glrunner.semesterDevOps.com.  IN  A  10.0.2.184
```

20.6 Kreieren des Reverese Zone Files für das Public Subnet

Reverse-Zonendateien sind der Ort, an dem wir DNS PTR-Einträge für Reverse-DNS-Abfragen definieren. Das bedeutet, wenn der DNS eine Abfrage nach einer IP-Adresse erhält, z. B. 10.0.1.77, wird er in der Reverse-Zonendatei(en) nachschauen, um den entsprechenden FQDN zu ermitteln, z. B. gitlab.semesterDevOps.com in diesem Fall. Lassen Sie uns die Zonendatei erstellen:

```
sudo nano /etc/bind/zones/db.10.0.1

;
; BIND reverse data file for 10.0.1.0/24
;
$TTL 604800
@ IN SOA ns1.semesterDevOps.com. admin.semesterDevOps.com. (
  2024011201 ; Serial (YYYYMMDDnn)
  604800 ; Refresh
  86400 ; Retry
  2419200 ; Expire
  604800
) ; Negative Cache TTL
;
; Name servers - NS records
IN NS ns1.semesterDevOps.com.
IN NS ns2.semesterDevOps.com.
; PTR Records
77 IN PTR gitlab.semesterDevOps.com. ; 10.0.1.246
```

- semesterDevOps.com. oder @ : Wurzel der Zone. Dies gibt an, dass die Zonendatei für die Domain semesterDevOps.com bestimmt ist. @ ist nur ein Platzhalter, der den Inhalt der $ORIGIN-Variable ersetzt.
- IN SOA: Der Teil „IN“ steht für Internet. SOA ist der Hinweis darauf, dass dies ein Start of Authority-Eintrag ist.
- ns1.semesterDevOps.com.: Definiert den primären Nameserver für diese Domain.
- admin.semesterDevOps.com.: E-Mail-Adresse des Administrators für diese Zone. Das „@“ wird in der E-Mail-Adresse durch einen Punkt ersetzt.
- Serial: Seriennummer für die Zonendatei. Jedes Mal, wenn die Zonendatei bearbeitet wird, muss diese Nummer inkrementiert werden, damit die Zonendatei korrekt propagiert wird. Sekundäre Server prüfen, ob die Seriennummer der Zone auf dem primären Server größer ist als die, die sie auf ihrem System haben. Ist dies der Fall, fordert der sekundäre Server die neue Zonendatei an; wenn nicht, wird weiterhin die ursprüngliche Datei bereitgestellt.
- Refresh: Aktualisierungsintervall für die Zone. Dies ist der Zeitraum, den der sekundäre Server wartet, bevor er den primären Server auf Änderungen der Zonendatei abfragt.
- Retry: Wiederholungsintervall für diese Zone. Wenn der sekundäre Server nach Ablauf des Aktualisierungsintervalls keine Verbindung zum primären Server herstellen kann, wartet er diesen Zeitraum und versucht dann erneut, den primären Server abzufragen.
- Expire: Ablaufzeitraum. Wenn ein sekundärer Nameserver den primären Server für diesen Zeitraum nicht kontaktieren konnte, gibt er keine Antworten mehr als autoritative Quelle für diese Zone zurück.
- Negative Cache TTL: Zeitraum, für den der Nameserver einen Namensfehler zwischenspeichert, wenn der angeforderte Name in dieser Datei nicht gefunden werden kann.

20.7 Kreieren des Reverse Zone Files für das Private Subnet

Nun das selbe für das private subnet

```
sudo nano /etc/bind/zones/db.10.0.2
```

```
;
; BIND reverse data file for 10.0.2.0/24
;
$TTL 604800
@ IN SOA semesterDevOps.com. admin.semesterDevOps.com. (
  1 ; Serial
  604800 ; Refresh
  86400 ; Retry
  2419200 ; Expire
```

```
    604800
) ; Negative Cache TTL
;
; Name servers - NS records
IN NS ns1.semesterDevOps.com.
IN NS ns2.semesterDevOps.com.
; PTR Records
252 IN PTR ns1.semesterDevOps.com. ; 10.0.2.225
217 IN PTR ns2.semesterDevOps.com. ; 10.0.2.10
148 IN PTR glrunner.semesterDevOps.com. ; 10.0.2.184
```

21 Checking the BIND Configuration Syntax

Wir können die Syntax aller named.conf* Files mit dem folgenden Command überprüfen:

```
sudo named-checkconf
```

um die Zone Files zu überprüfen helfen uns diese beiden:

```
sudo named-checkzone 1.0.10.in-addr.arpa /etc/bind/zones/db.10.0.1
sudo named-checkzone 2.0.10.in-addr.arpa /etc/bind/zones/db.10.0.2
```

sobald die Syntax über prüft wurde starten wir BIND9 restarten damit die änderungen angenommen werden:

```
sudo systemctl restart bind9
```

22 Configuring the Secondary DNS Server

23 GitLab

23.1 Wechsel zum öffentlichen GitLab-Server

Wir aktualisieren alle Pakete und starten das System neu, falls dies noch nicht geschehen ist.

23.2 Konfiguration des SSH-Ports

GitLab erlaubt es Benutzern, über SSH auf Repositories zuzugreifen. Dies führt jedoch zu Konflikten mit dem standardmäßigen SSH-Zugang des Systems. Daher ändern wir den SSH-Port des Systems auf `2424`, sodass der Port `22` für GitLab verwendet werden kann.

Diese Konfiguration wurde gewählt, da das Klonen eines Repositories eine häufigere Aufgabe ist als der SSH-Zugriff auf den Server. Durch die Verwendung des Standard-Ports für das Repository-Klonen entfällt die Notwendigkeit, einen benutzerdefinierten Port anzugeben. Für den selteneren SSH-Zugriff muss der alternative Port explizit angegeben werden.

Die Änderung wird in der Datei `/etc/ssh/sshd_config` vorgenommen:

```
Port 2424
```

Laut Dokumentation genügt ein Neustart des SSH-Dienstes, um die Änderungen zu übernehmen. In unserem Fall war jedoch ein Neustart des Servers erforderlich.

23.3 Anpassen der Sicherheitsgruppe auf AWS

Damit eine Verbindung über den Port `2424` möglich ist, muss dieser Port in der Sicherheitsgruppe `PublicGitLabSecurityGroup` freigegeben werden. Beachten Sie, dass bei jedem SSH-Zugriff auf die GitLab-Instanz der alternative Port angegeben werden muss:

```
ssh ubuntu@54.221.15.220 -p 2424
```

23.4 Installation von GitLab CE mit Docker Compose

Im Anschluss wird Docker gemäß der offiziellen Dokumentation installiert. Nach erfolgreichem Test mit dem `hello-world`-Image folgen wir der GitLab-Dokumentation.

23.4.1 Erstellen von Verzeichnissen zur Datenpersistenz

Gemäß der Dokumentation erstellen wir ein Verzeichnis für Konfigurations-, Log- und Daten-Dateien:

```
sudo mkdir -p /srv/gitlab
sudo chown -R ubuntu:ubuntu /srv/gitlab
export GITLAB_HOME=/srv/gitlab
```

23.4.2 Erstellen des Containers mit Docker Compose

Wir verwenden Docker Compose, um GitLab zu installieren. Dazu erstellen wir eine Datei `docker-compose.yml` mit folgendem Inhalt:

```
services:
  gitlab:
    image: gitlab/gitlab-ce:latest
    container_name: gitlab
    restart: always
    hostname: 'gitlab.ilovedevops.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://54.89.164.241'
    ports:
      - '80:80'
      - '443:443'
      - '22:22'
    volumes:
      - '$GITLAB_HOME/config:/etc/gitlab'
      - '$GITLAB_HOME/logs:/var/log/gitlab'
      - '$GITLAB_HOME/data:/var/opt/gitlab'
    shm_size: '256m'
```

Die `external_url` muss mit der öffentlichen IPv4-Adresse der GitLab-EC2-Instanz aktualisiert werden. Da die IP-Adresse nach jedem Neustart wechselt, ist eine Anpassung der Datei `docker-compose.yml` nach jedem Neustart notwendig.

Zum Starten des Containers führen wir folgenden Befehl aus:

```
docker compose up -d
```

GitLab ist nach kurzer Zeit unter `http://54.89.164.241` erreichbar. Der Standard-Benutzername lautet `root`, das Passwort kann mit folgendem Befehl abgerufen werden:

```
docker compose exec -it gitlab cat /etc/gitlab/initial_root_password
```

Abbildungsverzeichnis

Tabellenverzeichnis