Fachhochschule Nordwestschweiz
Hochschule für Architektur, Bau und Geomatik

**PCLS – Case study**

# Autoscaling Nextcloud in AWS

Viktor Weilenmann, Damjan Mlinar

Brugg, 02.01.2024

Viktor.Weilenmann@fhnw.ch
Damjan.Mlinar@fhnw.ch

# Inhaltsverzeichnis

# 1 Beschreibung Use Case

Dieses Projekt wurde im Rahmen des PCLS Moduls, während des Herbstsemesters 2023, an der Fachhochschule Nordwestschweiz umgesetzt.

Für das Abschlussprojekt haben wir uns das Ziel gesetzt, eine gehostete Version von Nextcloud bereitzustellen. Nextcloud ist eine Open-Source-Software, die es Nutzern ermöglicht, Daten wie Dateien, Kalender und Kontakte auf einem Server zu speichern. Der Zugriff auf diese Daten ist sowohl über eine Weboberfläche als auch über Client-Applikationen für Smartphones und Desktops möglich. Die Synchronisation zwischen Server und Clients gewährleistet einen konsistenten Datenbestand, der von verschiedenen Endgeräten aus abgerufen werden kann. Zusätzlich bietet Nextcloud Funktionen für Videokonferenzen sowie verschiedene Office-Applikationen über die Weboberfläche.

Die Besonderheit von Nextcloud liegt darin, dass es auf einem privaten Server oder Webspace betrieben werden kann. Dies ermöglicht es dem Nutzer, die Kontrolle über die verwalteten Daten zu behalten und potenzielle Risiken für Datenmissbrauch durch Diensteanbieter zu vermeiden. Insbesondere die kommerzielle Verwertung von Daten mit oft unklaren Datenschutzrichtlinien kann durch den Betrieb eines eigenen Nextcloud-Servers umgangen werden.

In unserem Projekt stand die Herausforderung, die optimale Konfiguration für das Hosting von Nextcloud auf Amazon Web Services (AWS) zu ermitteln. Unser Hauptziel bestand darin, dem Kunden eine detaillierte Anleitung bereitzustellen, mit der er die erforderliche Infrastruktur eigenständig und ohne umfassende AWS-Kenntnisse aufbauen kann. Das Ergebnis sollte ein reibungsloser und schneller Einsatz von Nextcloud auf der AWS-Plattform sein.

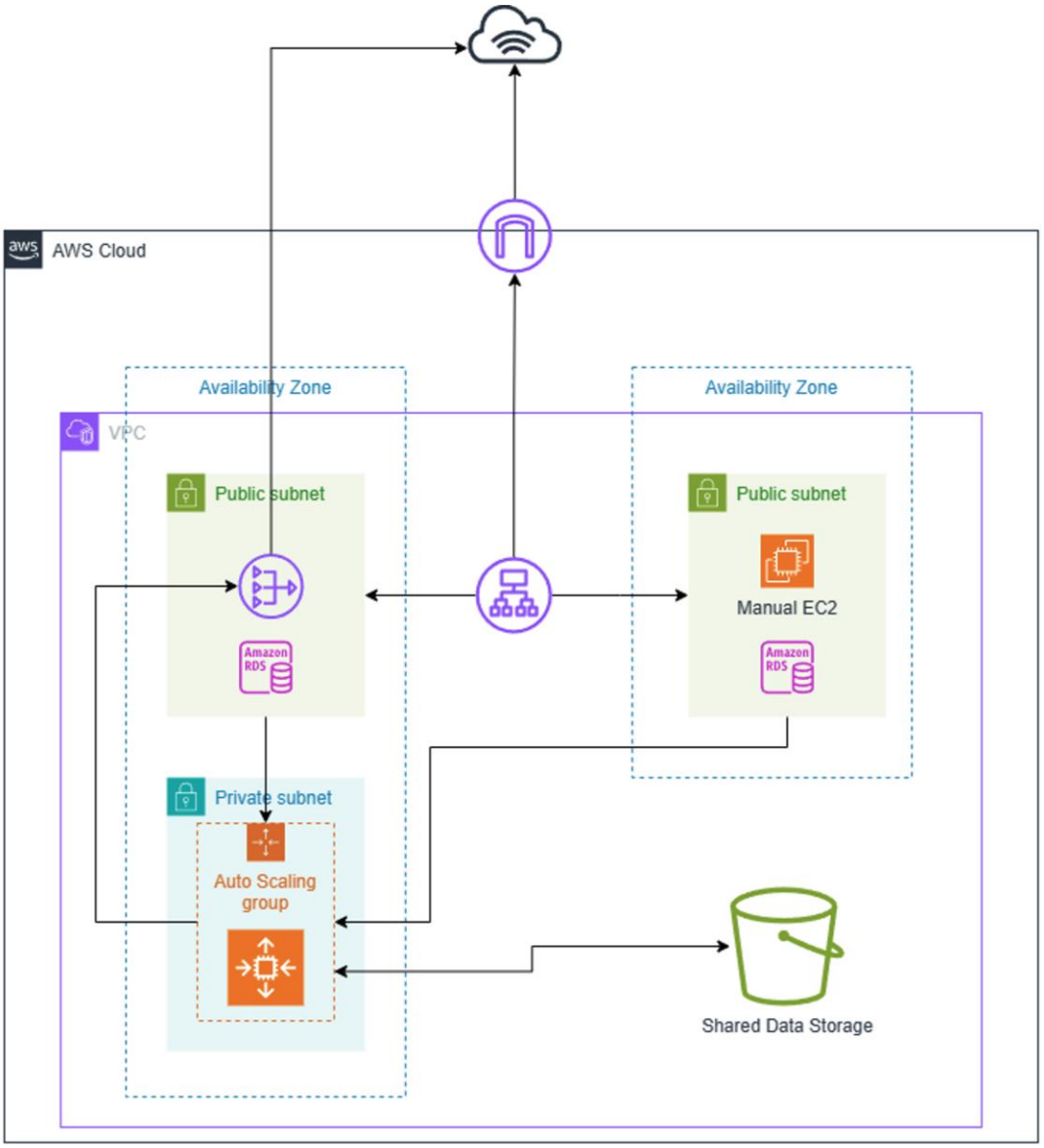
# 2 Gewählte Architektur

In unserer Entscheidung für die Architektur haben wir Terraform als zentrales Tool ausgewählt. Als Gruppe verfügen wir bisher über keine Erfahrung in der Cloud-Technologie, und unser Ziel ist es, durch die Anwendung von Terraform einen fundierten Einstieg in das Scripting und die Automatisierung von Cloud-Ressourcen zu erlangen.

Zusätzlich haben wir uns für Amazon Web Services (AWS) als unsere Cloud-Plattform entschieden. Obwohl wir auch hier über wenig Erfahrung verfügen, möchten wir die Gelegenheit nutzen, AWS zu erkunden und durch praktische Anwendung mehr über Cloud-Technologien zu lernen. Unser Hauptziel besteht darin, Erfahrungen zu sammeln und in einem experimentellen Umfeld erste Schritte mit AWS zu unternehmen.

Die Wahl von Terraform als Infrastruktur-as-Code (IaC) Tool und AWS als Cloud-Provider bietet uns eine flexible und konsistente Möglichkeit, unsere AWS-Ressourcen zu verwalten. Terraform ermöglicht es uns, die gesamte Infrastruktur, einschliesslich EC2-Instanzen, Datenbanken und Netzwerkkomponenten, in einem einzigen Konfigurationscode zu definieren. Dies fördert die Wiederverwendbarkeit, Konsistenz und Skalierbarkeit unserer Infrastrukturdefinitionen.

- S3 Bucket
- EC-2
- LoadBalancer
- Auto Scaling Group
- RDS DB
- VPC




TODO Description

# 3 Vorgehensweise für Umsetzung

Wir hatten beide noch nie mit AWS gearbeitet, die einzigen Inputs bekamen wir aus dem PCLS Modul während dem Semester. Wir stellten uns die Frage, was wir von diesem Projekt mitnehmen wollten. Beiden war es wichtig erste Erfahrungen mit dem Infrastructure-as-Code Tool Terraform zu machen. Um einen guten ersten Eindruck mit AWS zu bekommen wollten wir auch unbedingt den Infrastructure-as-a-Service (IaaS) Ansatz verwenden. Von den Services von AWS her wollten wir uns auf drei Dinge fokusieren: Wie verwendet man Virtual Private Cloud mit den Features, wie verbinden sich EC2 Instanzen mit RDS Datenbanken und wie konfiguriert man Autoscaling.

Aus der fachlichen Perspective bietet das IaC Tool Terraform eine

# 4 Beschreibung der Implementation

## 4.1 RDS DB

Für die Datenbankintegration haben wir eine PostgreSQL-Datenbank mithilfe von Terraform erstellt. Die nachfolgende Terraform-Konfiguration zeigt die relevanten Parameter und Einstellungen für die Instanzierung der PostgreSQL-Datenbank:

```
#create database
resource "aws_db_instance" "postgres" {
  allocated_storage    = 20
  storage_type         = "gp2"
  engine               = "postgres"
  engine_version       = "15"
  instance_class       = "db.t3.small"
  db_name              = "postgres"
  username             = "nextcloud"
  password             = "nextcloud"
  skip_final_snapshot  = true
  publicly_accessible = true
  vpc_security_group_ids = [aws_security_group.ec2-securitygroups.id]
  db_subnet_group_name = aws_db_subnet_group.postgres_subnet_group.name
  depends_on = [aws_internet_gateway.gw]
}
```

Wir haben Subnetze in unserer Architektur integriert, um Netzwerkkomponenten effektiv zu organisieren und zu isolieren. Hier sind die relevanten Teile unserer Terraform-Konfiguration:

```
# subnet linking to database
resource "aws_db_subnet_group" "postgres_subnet_group" {
  name       = "postgresubgroup"
  subnet_ids = [aws_subnet.pub-sub-1.id,aws_subnet.pub-sub-2.id]
}
```

Diese Ressource definiert eine Gruppe von Subnetzen, die für die PostgreSQL-Datenbankinstanz verwendet werden. Die Subnetze pub-sub-1 und pub-sub-2 sind hier eingebunden, um eine optimale Verteilung und Verfügbarkeit zu gewährleisten. Diese Subnetze sind jeweils in unterschiedlichen Availability Zones lokalisiert, nämlich "eu-north-1a" für pub-sub-1 und "eu-north-1b" für pub-sub-2.

### 4.2 S3 Bucket (Data Storage)

Die Einrichtung des Buckets für den gemeinsamen Serverdatenspeicher von Nextcloud gestaltet sich dank Terraform äusserst einfach:

```
#create shared S3 bucket
resource "aws_s3_bucket" "s3bucket" {
  bucket = "nextcloud-aio-bucket-pcls"
}
```

Für die Authentifizierung und den Zugriff auf diesen Bucket verwendet Nextcloud Access Keys und Secrets. Aufgrund dieser Konfiguration benötigen wir keine weiteren spezifischen Einstellungen für den Bucket selbst. Die Integration von Terraform in diesen Prozess vereinfacht nicht nur die Bereitstellung, sondern gewährleistet auch die Konsistenz und Wiederholbarkeit dieser Konfiguration in verschiedenen Umgebungen.

## 4.3 EC2

Unsere Implementierung erfolgte in zwei Teilen: Zuerst erfolgte die manuelle Instanzierung, gefolgt vom Einsatz des Autoscaling-Skripts.

```
# Create first EC2 instance
resource "aws_instance" "server" {
  ami           = "ami-0014ce3e52359afbd"
  instance_type = "t3.micro"
  availability_zone = "eu-north-1a"
  key_name = "pcls-sshkey"
  network_interface {
    device_index = 0
    network_interface_id = aws_network_interface.pcls-networkinterface.id
  }

  user_data = templatefile("setup.sh.tpl", {
access_key = var.access_key
secret_key = var.secret_key
postgres_host = aws_db_instance.postgres.address
postgres_db = aws_db_instance.postgres.db_name
postgres_user = aws_db_instance.postgres.username
postgres_password = aws_db_instance.postgres.password
s3_bucket = aws_s3_bucket.s3bucket.bucket
})
}
```

```
1   #!/bin/bash
2   sudo apt update -y
3   sudo apt upgrade -y
4   sudo apt install docker.io -y
5   sudo docker run -d -p 8080:80 \
6   --name NextCloudContainer  \
7   -e POSTGRES_HOST=${postgres_host} \
8   -e POSTGRES_DB=${postgres_db} \
9   -e POSTGRES_USER=${postgres_user} \
10  -e POSTGRES_PASSWORD=${postgres_password} \
11  -e OBJECTSTORE_S3_HOST=s3.eu-north-1.amazonaws.com \
12  -e OBJECTSTORE_S3_BUCKET=${s3_bucket} \
13  -e OBJECTSTORE_S3_KEY=${access_key} \
14  -e OBJECTSTORE_S3_SECRET=${secret_key} \
15  -e OBJECTSTORE_S3_REGION=eu-north-1 \
16  nextcloud
```

Zunächst haben wir uns darauf konzentriert, eine einzelne Nextcloud-Instanz zum Laufen zu bringen. Nach einigen Versuchen gelang dies erfolgreich. Innerhalb der EC2-Instanz verwenden wir ein Docker-Image von Nextcloud, das mit bestimmten Umgebungsvariablen konfiguriert ist, um den initialen Setup zu vereinfachen.

Dieser Initiierungsschritt erfordert jedoch erhebliche manuelle Arbeit. Es ist notwendig, die Instanz zuerst zu starten, das Nextcloud-Setup durchzuführen, die Konfigurationsdatei aus dem Container zu extrahieren und sie auf einem separaten Bucket oder FileShare-Dienst hochzuladen. Dies ermöglicht es, die Konfigurationsdatei schnell per Curl bei der Neuinstanzierung der Autoscaling-Gruppe zu verwenden.

Während des Nextcloud-Setups werden alle Datenbankeinträge sowie die Konfiguration für den S3-Bucket und die RDS-Datenbank erstellt. Bedauerlicherweise konnten wir während unserer Recherche keine Methode finden, um diesen Schritt zu automatisieren. Daher erfordert unser Skript zunächst die Erstellung der einzelnen EC2-Instanz, das Abschliessen des Nextcloud-Setups und anschliessend das Kopieren der Konfigurationsdatei auf einen Bucket. Das Autoscaling-Skript kann dann diese Konfigurationsdatei für jede neue Instanz einbinden.

Ein Ausschnitt aus unserem Bash-Skript für eine neue Instanz ist wie folgt:

```
#!/bin/bash
  sudo apt update -y
  sudo apt upgrade -y
  sudo apt install docker.io -y
  sudo docker run -d -p 80:80 \
       --name NextCloudContainer  \
       nextcloud
  sleep 20
  sudo docker exec NextCloudContainer bash -c "apt update && apt install curl -y"
  sudo docker exec NextCloudContainer bash -c "curl 'https://onlyforconfig.s3.eu-north-
  sudo docker exec NextCloudContainer  rm /var/www/html/config/CAN_INSTALL
  sudo docker exec NextCloudContainer  chown -R 33:33 /var/
  sudo docker exec NextCloudContainer  touch /var/www/html/data/.ocdata
```

Dieser Abschnitt des Skripts installiert erforderliche Abhängigkeiten, lädt die Konfigurationsdatei von einem S3-Bucket herunter und führt anschliessend Anpassungen an den Dateien in der Nextcloud-Instanz durch, um den automatischen Einrichtungsprozess abzuschliessen. Dieser Schritt hat den grössten Zeitbedarf erfordert. Die Dokumentation von Nextcloud selbst war wenig hilfreich. Erst nach umfangreichen Recherchen in Foren und auf Reddit stiessen wir auf eine Methode, wie wir die Automatisierung für eine neue Instanz realisieren können.

Es ist notwendig, die Datei "CAN_INSTALL" zu entfernen, um anzuzeigen, dass das initiale Setup abgeschlossen wurde. Anschliessend wird dem Benutzer, auf dem Nextcloud läuft, die Berechtigung für das Verzeichnis "var" zugewiesen. Abschliessend benötigt Nextcloud noch eine vorhandene ".ocdata"-Datei.

Mit diesen Schritten ist das Skript zur Erstellung neuer Instanzen anhand einer Konfigurationsdatei funktionsfähig.

### 4.3.1 Autoscaling & Load Balancer

TODO erklärung autoscaling, stickyness etc

Damjan

## 4.4 VPC

# 5 Erkenntnisse und Fazit

Die Durchführung dieser Case Study gewährte uns einen faszinierenden Einblick in die Welt von AWS sowie die Anwendung von Terraform als mächtigem Werkzeug zur Ressourcenverwaltung. Terraform erwies sich als äusserst benutzerfreundlich und ermöglichte uns, AWS-Ressourcen auf eine klare und unkomplizierte Weise zu erstellen.

Unsere Hauptherausforderung bestand darin, die Funktionalität von Nextcloud nahtlos in unsere AWS-Umgebung zu integrieren. Trotz der vergleichsweisen unkomplizierten Handhabung von Terraform war es notwendig, beträchtliche Zeit in die Feinabstimmung der Nextcloud-Instanz zu investieren. Insbesondere die Erstellung neuer Instanzen mit einer vorhandenen Datenbank und bereits vorhandenen Daten gestaltete sich als anspruchsvolle Aufgabe.

Die Automatisierung des Setups gestaltete sich aufgrund der spezifischen Anforderungen von Nextcloud als herausfordernd. Das Fehlen einer umfassenden Methode zur Automatisierung bestimmter Schritte erforderte eine manuelle Intervention, um sicherzustellen, dass die neuen Instanzen korrekt konfiguriert und mit den erforderlichen Datenbank- und S3-Konfigurationen versehen wurden.

Trotz dieser Herausforderungen haben wir wertvolle Einsichten in die Prinzipien der Cloud-Architektur und deren Umsetzung gewonnen. Die Kombination aus AWS und Terraform bietet eine solide Grundlage für die Implementierung skalierbarer und kosteneffizienter Cloud-Infrastrukturen. Wir schätzen die Möglichkeit, praktische Erfahrungen zu sammeln und sind zuversichtlich, dass die gewonnenen Erkenntnisse einen Mehrwert für zukünftige Projekte darstellen werden.

Insgesamt war diese Case Study nicht nur lehrreich, sondern auch ein bedeutender Schritt in unserer Reise, Cloud-Technologien erfolgreich zu nutzen und in zukünftigen Projekten effektiv einzusetzen

KOMMENTAR Route 53, Bucket probleme beim Sandbox Account