Autor: Patrick Michel

Parallele und verteilte Systeme

Vorwort:
Grundlage für diese Aufgabe ist die Präsentation «Grundlagen verteilte Systeme»

Dauer: 90 Minuten

Nachfolgend setzen Sie sich mit den Grundlagen von verteilten Systemen auseinander. Dabei wird als Grundlage das Konzept von Containern stark in den Fokus gestellt welches inzwischen bekannt sein sollte.

Aufgabe 1)
Spielen Sie nun mit folgendem Docker-Compose etwas herum. Experimentieren Sie etwas und beantworten Sie dabei die nachfolgenden Fragen.
https://raw.githubusercontent.com/docker/awesome-compose/master/nextcloud-redis-mariadb/compose.yaml

a) Erklären Sie in Ihren eigenen Worten, was Sie unter einem verteilten System verstehen.

Ein verteiltes Systemn besteht aus mehreren Komponenten. Jeder Komponente übernimmt im Gesamtsystem eine bestimmte Aufgabe Z.B. Caching, Datenbank, Webserver und diese Komponenten zusammen ergeben ein grösseres Ganzes, um eine bestimmte Aufgabe zu erfüllen.

b) Erklären Sie den Unterschied zwischen einer Monolithischen Systemarchitektur und einer Microservice Architektur mit Vor- und Nachteilen.

Monolithische Architektur
Dies ist eine traditionelle Methode, bei der alle Prozesse und Funktionen in einer einzigen Anwendung zusammengeführt werden.

Vorteile
- Einfacher zu entwickeln, da alles in einem einzigen Codebase ist.
- Einfacher zu testen, da nur eine Instanz vorhanden ist.

Nachteile
- Schwierig zu skalieren, da die gesamte Anwendung skaliert werden muss.
- Änderungen können Auswirkungen auf die gesamte Anwendung haben.

Microservice-Architektur
Bei dieser Methode wird eine Anwendung in kleinere, unabhängige Dienste unterteilt, die jeweils eine spezifische Funktion erfüllen.

Vorteile
- Einfacher zu skalieren, da jeder Dienst unabhängig skaliert werden kann.
- Änderungen an einem Dienst beeinträchtigen nicht unbedingt andere Dienste.

Nachteile
- Komplexer zu entwickeln und zu koordinieren, da mehrere Dienste beteiligt sind.
- Schwieriger zu testen, da Interaktionen zwischen Diensten berücksichtigt werden müssen.

c) Welche Herausforderungen Sehen Sie für den Betrieb eines verteilten Systems?

- Netzwerklatenz: Da Komponenten nun über das Netzwerk miteinander verbunden werden, kann es dabei zu Latenz und zu Verzögerungen in der Datenübertragung kommen.
- Netzwerksicherheit: Da Komponenten über das Netzwerk kommunizieren muss die Sicherheit auf Netzwerkebene sichergestellt werden.
- Datenkonsistenz: Wenn mehrere Systeme die gleichen Daten halten, muss sichergestellt werden, dass alle den gleichen Datenstand haben und dass die Komponenten entsprechend synchronisiert werden

d) Starten Sie das oben angegebene Docker-Compose mithilfe der «Docker» CLI.

```
docker compose up -d
```

e) Handelt es sich bei diesem Docker-Compose um ein verteiltes System? Wenn ja, warum?

Ja denn es besteht aus mehreren spezialisierten «Services» welche bestimmten Aufgaben im Gesamtsystem übernehmen. Somit ist es ein System aus mehreren Komponenten, die aber aktuell noch auf dem gleichen «Node» ausgeführt werden. Dieses Problem muss später noch angegangen werden.

f) Beschäftigen Sie sich mit den einzelnen Komponenten der Komposition. Welche Aufgaben nehmen die jeweiligen Komponenten im Gesamtsystem ein?

- **nc**: Dieser Service beschreibt das Nextcloud Image welches als Apache Webserver ausgeführt wird. Nextcloud ist eine Webapplikation, welche es erlaubt, eine eigene Datenablage wie z.B. Drop Box oder Google Drive zu betreiben.
- **redis**: Ist eine In-Memory Key-Value Store Datenbank und erlaubt es dem Webserver bestimmte Informationen zu cachen und wird somit massiv beschleunigt.
- **db**: Dieser Service beschreibt die MariaDB Datenbank, welche von der Nextcloud Webapplikation verwendet wird, um Daten wie Benutzer, Konfigurationen, Freigaben usw. zu speichern. Dieser Service enthält keine Dateien, welche auf Nextcloud hochgeladen werden, diese werden direkt auf das Dateisystem geschrieben, sondern nur jeweilige Verweise.

g) Welche Komponenten in diesem System können horizontal skaliert werden?

Nur der Webserver, welche die Nextcloud Applikation betreibt. Diese hat nämlich keinen Zustand (Stateless) und jede weitere Instanz braucht keine spezielle Konfiguration, um dieselbe Arbeit zu erledigen.

h) Passen Sie die Komposition so an, dass mindestens 1 Teil des Systems horizontal skaliert wird und dadurch ausfallsicherer wird.

```
nc:
  image: nextcloud:apache
  restart: always
  ports:
    - 9800:80
  volumes:
    - nc_data:/var/www/html
  networks:
    - redisnet
    - dbnet
  environment:
    - REDIS_HOST=redis
    - MYSQL_HOST=db
    - MYSQL_DATABASE=nextcloud
    - MYSQL_USER=nextcloud
    - MYSQL_PASSWORD=nextcloud
nc2:
  image: nextcloud:apache
  restart: always
  ports:
    - 9900:80
  volumes:
    - nc_data:/var/www/html
  networks:
    - redisnet
    - dbnet
  environment:
    - REDIS_HOST=redis
    - MYSQL_HOST=db
    - MYSQL_DATABASE=nextcloud
    - MYSQL_USER=nextcloud
    - MYSQL_PASSWORD=nextcloud
```

i) Welche de Komponenten können nicht ohne weiteres skaliert werden und warum nicht?

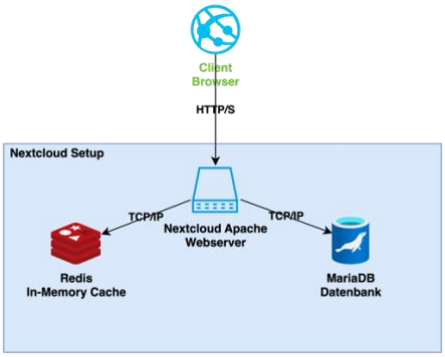
Redis oder auch die MariaDB. Der Grund ist diese speichern einen bestimmten Zustand (Persistenz). Wenn diese ausfallsicher aufgezogen werden soll, müssen diese als Cluster Setup aufgebaut werden. Das erfordert eine etwas aufwändigere Konfiguration da die einzelnen Knoten im System untereinander synchronisiert werden müssen.

j) Was bedeutet der jeweilige Ausfall einer bestimmten Komponente für das Gesamtsystem?

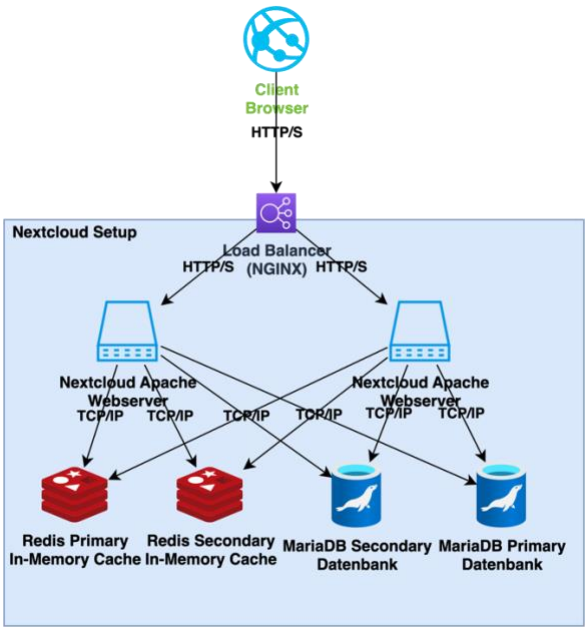
- **nc**: Solange noch eine weitere Instanz läuft, läuft auch das Gesamtsystem noch weiter. Der Benutzer muss einfach direkt auf die noch laufende Instanz verbinden.
- **redis**: Die Nextcloud sollte in der Lage sein weiter zu laufen, es kann aber zu Performance Problemen kommen.
- **db**: Wenn die Datenbank aussteigt, dann ist das Gesamtsystem lahmgelegt da die Nextcloud Webapplikation zwingend auf diese angewiesen ist.

k) Erstellen Sie eine einfache Grafik mit den Komponenten des Gesamtsystems, wie dieses Kommuniziert und welche Optimierungen vorgenommen werden könnten.
Verwenden Sie dazu ein Tool wie: https://app.diagrams.net/

Standardsetup



Redundantes Setup



l) Ist es möglich, mit Docker Compose vertikal zu skalieren? Wenn Ja, wie?

Nicht unbedingt, ist nicht direkt dazu gedacht, obwohl es möglich ist, einzelnen Containern oder Services ein maximales Limit an CPU und RAM zuzuweisen. Dies wird in den meisten Fällen aber nicht gemacht.

Aufgabe 2)
Beschäftigen Sie sich mit den aktuell grössten Public Cloud Anbietern und deren Angeboten. Versuchen Sie herauszufinden welche der Dienste in einem verteilten System zum Einsatz kommen könnten.

a) Was versteht man unter einem Hyperscaler?

Ein Hyperscaler ist ein Unternehmen, das eine extrem große und skalierbare Cloud-Infrastruktur betreibt und bereitstellt. Diese Unternehmen bieten Cloud-Computing-Dienste, die von Unternehmen und Organisationen genutzt werden können, um ihre IT-Infrastruktur und Anwendungen zu hosten und zu betreiben.

b) Welches sind die 3 grössten Anbieter für Public Cloud Infrastruktur?

- Amazon: AWS
- Microsoft: Azure
- Google: GCloud

c) Welche Dienste werden angeboten und welche davon könnten für ein verteiltes System in Frage kommen?

Der Hauptdienst ist das zur Verfügung stellen von Computing Ressourcen wie RAM, CPU, GPU, Speicher aber auch Services die Managed Datenbanken oder Netzwerk.

d) Was verstehen Sie unter Cloud Nativen Diensten?

Cloud-native Dienste sind Anwendungen und Dienste, die speziell für die Ausführung in der Cloud entwickelt wurden. Diese Dienste sind so konzipiert, dass sie die Vorteile der Cloud vollständig nutzen können, wie beispielsweise die Skalierbarkeit, die Flexibilität, die Zuverlässigkeit und die Leistungsfähigkeit.

Cloud-native Dienste sind in der Regel containerisiert und werden in einem Cloud-orchestrierungssystem ausgeführt, das automatisch Skalierung, Lastausgleich, Fehlerbehebung und andere Aufgaben handhabt, um eine maximale Verfügbarkeit und Performance zu gewährleisten.

e) Was verstehen Sie unter Serverless?

Serverless ist ein Ansatz für die Entwicklung und Bereitstellung von Anwendungen und Diensten in der Cloud, bei dem der Entwickler sich nicht um die Verwaltung und Bereitstellung von Servern kümmern muss. Stattdessen wird die Infrastruktur automatisch von einem Cloud-Provider verwaltet, und der Entwickler zahlt nur für die tatsächlich genutzte Rechenleistung und Speicher.

Der Name "Serverless" bedeutet nicht, dass es keine Server gibt, sondern dass der Entwickler sich nicht um die Verwaltung und Wartung dieser Server kümmern muss. Stattdessen wird die Infrastruktur als Service (IaaS) bereitgestellt, und der Cloud-Provider skaliert automatisch die Anzahl der Server hoch oder runter, um die Anforderungen der Anwendung zu erfüllen.

f) Welche Architekturformen kennen Sie, um ein System aufzubauen und welche würden Sie für eine Neuentwicklung bevorzugen? Begründen Sie.

- Monolithen: Alte aber bewährte Architektur. Skalierung eher schwierig. Und Deployment statisch
- Client/Server: Für Webapplikationen am besten geeignet. Mit Auftrennung in Frontend und Backend
- Peer-to-Peer: Nur in sehr speziellen Fällen überhaupt anwendbar
- Microservice: Moderne, skalier fähige und sehr resiliente Architektur. Das Verwalten gestaltet sich etwas schwieriger hat aber ansonsten viele Vorteile betreffend Skalierbarkeit, Fehlertoleranz, Stabilität, Lastenausgleich usw.

Kein eindeutiger Gewinner es kommt jeweils auf die Anforderungen drauf an.