postgres auf ubuntuu

Installation und Dokumentation auf athanasiou.me

1 Inhaltsverzeichnis

Ziel dieses Dokumentes

Ziel des Tutorials / der Dokumentation ist es:

- eine Postgres Installation auf einem remote ubunto Server durchzuführen,
- auf die Postgres Installation mit einem Mac zugreifen auf dem Postico als SQL-Client installiert ist.

2 Installation postgres auf ubuntu

PostgreSQL ist ein objektrelationales Open-Source-Datenbanksystem, das die SQL-Sprache in Kombination mit vielen Funktionen erweitert. Die Ursprünge von PostgreSQL gehen auf das Jahr 1986 als Teil des POSTGRES-Projekts an der University of California in Berkeley zurück.

2.1 apt

`apt` ist ein Kommandozeilen-Programm für das „Advanced Packaging Tool“, der Paketverwaltung, die bei Debian, Ubuntu und damit auch allen offiziellen Ubuntu-Varianten zum Einsatz kommt.

`apt` kann Software - Pakete installieren, deinstallieren oder auch nur Informationen zu Paketen anzeigen.

Um PostgreSQL zu installieren, aktualisiert man zunächst das lokale apt-Verzeichnis des Servers:

```
$ sudo apt update
```

2.2 Installation

Dann installiert man mit `apt` das Postgres-Paket zusammen mit einem `-contrib`–Paket, das einige zusätzliche Dienstprogramme und Funktionen hinzufügt:

```
$ sudo apt install postgresql postgresql-contrib
```

und startet den Postgres Server:

```
$ sudo systemctl start postgresql.service
```

Postgres ist in ubuntu „weiter“ als bspw. Tomcat, so ist die Datei `postgresql.service` bereits vorhanden, sprich postgres ist `systemd` bereits „bekannt“.

Das Ganze geht auch etwas knapper und mit den selbsterklärenden Parametern start / stop und status:

```
$ sudo systemctl start postgresql

$ sudo systemctl stop postgresql

$ sudo systemctl status postgresql
```

3 Basis Konfiguration

3.1 User/ Rolle postgres

Standardmäßig nutzt postgres ein Konzept namens "Rollen", zur Authentifizierung und Autorisierung. Diese ähneln in gewisser Weise „normalen“ Usern und Gruppen im Unix-Stil. Bei der Installation ist postgres so eingerichtet, dass es eine sog. ident-Authentifizierung verwendet, was bedeutet, dass Postgres-Rollen mit einem übereinstimmenden Unix/Linux-User verknüpft werden. Wenn eine Rolle in Postgres vorhanden ist, kann sich ein Unix/Linux-Benutzername mit demselben Namen in dieser Rolle anmelden.

In Postgres gibt es eine Standardrolle „postgres“. Bei der Installation wird ein (ubuntu-) Useraccount mit dem Namen **postgres** erstellt und der „postgres“- Rolle zugeordnet.

Diesen Account kann man (muss man) nach der Installation für den Zugriff auf postgres verwenden:

```
$ sudo -u postgres psql
```

Damit startet man als ubuntu-User „postgres“ den PostgresSQL Prompt `psql`:

```
postgres=#
```

`psql` ist der Kommandozeilen Prompt von postgres. Hier kann man SQL-Kommandos eingeben, aber auch weitere Nutzer und Rollen einrichten. Wie gesagt - den Prompt kann man nur als User „postgres“ aufrufen (!). Verlassen tut man den PostgresSQL-Prompt mit `\q`.

3.2 Einrichten neuer User/Rollen in postgres

Als ubuntu-User „postgres“ kann man das Kommando `createuser` nutzen um neue User/Rollen in postgres anzulegen:

```
postgres@server: createuser –interactive
```

Wenn man lieber sudo (aus seiner Standardshell heraus) für jeden Befehl verwenden möchte:

```
$ sudo -u postgres createuser --interactive
```

In beiden Fällen fordert das Skript zu einigen Eingaben auf und führt dann die Postgres-Befehle aus, um einen Benutzer nach den gegebenen Spezifikationen zu erstellen:

```
Enter name of role to add: akis

Shall the new role be a superuser? (y/n) y
```

…

3.3 Datenbanken anlegen

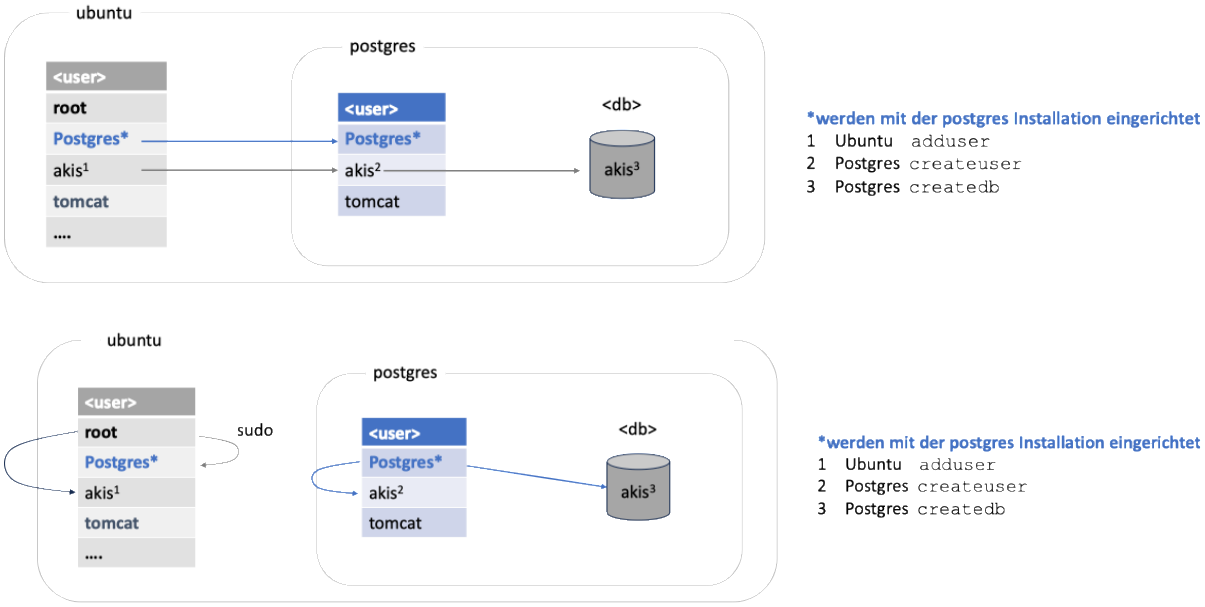
Eine weitere Annahme, die das Postgres-Authentifizierungssystem standardmäßig trifft, ist, dass jede Rolle, die für die Anmeldung verwendet wird, über eine Datenbank mit demselben Namen verfügt, auf die sie zugreifen kann. Wenn der Benutzer, der im letzten Abschnitt erstellt wurde, „akis“ heißt, versucht diese Rolle eine Verbindung zu einer Datenbank herzustellen, die standardmäßig auch "akis" heißt. Die entsprechende Datenbank wird mit dem Befehl `createdb` erstellt. Wenn man als „postgres“ angemeldet sind, heißt der Befehl:

```
postgres@server: createdb akis
```

Auch hier, wen man, `sudo` bevorzugt, heißt es

```
$ sudo -u postgres createdb akis
```

Das Ganze in zwei Schaubildern illustriert:



3.4 Zugriff auf DB gewähren

Auf dem STRATO Server wird über den Tomcat Applikationsserver auf die DB zugegriffen. Tomcat läuft als Prozess des gleichnamigen ubuntu-Users. Damit Daten via JDBC übertragen werden können, ist es nötig das der Nutzer Tomcat Zugriff auf die DB bekommt.
Das passiert in postgres mittels:

```
CREATE USER tomcat WITH LOGIN ENCRYPTED PASSWORD 'tomcat';

GRANT INSERT, UPDATE, SELECT, DELETE
ON ALL TABLES IN SCHEMA public
TO tomcat;
```

4 Client – Server Architektur

PostgreSQL basiert auf einem Client-Server-Modell: Ein Serverprozess verwaltet die Datenbankdateien sowie die Verbindungen, die von Client-Programmen zum Server aufgebaut werden und bearbeitet die Anfragen, die von diesen Client-Programmen gestellt wurden. Diese Aufgaben werden bei PostgreSQL vom Serverprogramm "postmaster" erledigt.

4.1 Lokale Client Programme

Um mit dem "postmaster" zu kommunizieren, können die unterschiedlichsten Client-Programme benutzt werden. Mit einem Client-Programm werden Verbindungen zum Datenbankserver aufgebaut und Aktionen in der Datenbank ausgeführt.

- In der PostgreSQL-Distribution direkt enthalten ist `psql`, ein einfacher aber schneller Datenbankmonitor – dem sind wir schon begegnet.
- `pgAccess`, ein grafischer Client, wird in den meisten Linux-Distributionen mitgeliefert
- bei interaktiven Webseiten übernimmt der Webserver die Rolle des Clients.

Das Wesen von `psql` als Client zeigt sich wenn man im `psql` das Kommando /`conninfo` eingibt:

```
postgres-# \conninfo
You are connected to database "postgres" as user "postgres" via socket in
"/var/run/postgresql" at port "5432".
postgres-#
```

Der Client psql ist auf dem Host mit dem Postmaster mittels eines lokalen Unix-Domain-Sockets am Port 5432 verbunden. Dort ist er als User “postgres“ angemeldet und hat Zugriff auf die Datenbank „postgres“.

4.2 Remote Zugriff

Beim Client-Server-Modell müssen Client und Server nicht auf derselben Maschine installiert sein und sind es tatsächlich in den wenigsten Fällen. Sie kommunizieren über eine TCP/IP Verbindung. Der PostgreSQL Server kann viele parallele Verbindungen verwalten. Jedesmal, wenn ein Client sich mit dem Server verbindet, startet der "postmaster" einen neuen Prozess, der dann die Client-Server-Kommunikation übernimmt.

4.3 Server-seitige Konfiguration remote Zugriff

Einen remote Zugriff gibt es nicht ganz ohne etwas Arbeit. Man muss dem postgres System in einer Konfigurationsdatei schon sehr genau mitteilen, welcher remote Client (genauer aus welchem Netzwerk oder von welcher IP Adresse), mit welcher User-Kennnung, was genau tun darf - sprich wie sich ein Client authentifiziert, und was er darf.

Die Clientauthentifizierung wird durch eine Konfigurationsdatei gesteuert, die den Namen `pg_hba.conf` trägt und im Datenverzeichnis des Datenbankclusters gespeichert wird. (HBA steht für hostbasierte Authentifizierung).

Das allgemeine Format der Datei `pg_hba.conf` ist eine Folge von Datensätzen, einer pro Zeile. Jeder Authentifizierungsdatensatz spezifiziert einen Verbindungstyp (Connection type), einen Client-IP-Adressbereich (falls für den Verbindungstyp relevant), einen Datenbanknamen, einen Benutzernamen und die Authentifizierungsmethode, die für Verbindungen verwendet werden soll, die diesen Parametern entsprechen.

Der erste Datensatz mit einem übereinstimmenden Verbindungstyp, einer Clientadresse, einer angeforderten Datenbank und einem Benutzernamen wird für die Authentifizierung verwendet.

Es gibt kein "Fall-Through" oder "Backup": Wenn ein Datensatz ausgewählt wird und die Authentifizierung fehlschlägt, werden nachfolgende Datensätze nicht berücksichtigt. Wenn kein Datensatz übereinstimmt, wird der Zugriff verweigert.

4.4 pg_hba.conf

Ein Datensatz in `pg_hba.conf` kann mehrere Formen haben:

```
local            database  user  auth-method [auth-options]
host             database  user  address     auth-method  [auth-options]
hostssl          database  user  address     auth-method  [auth-options]
hostnossl        database  user  address     auth-method  [auth-options]
hostgssenc       database  user  address     auth-method  [auth-options]
hostnogssenc     database  user  address     auth-method  [auth-options]
host             database  user  IP-address  IP-mask      auth-method  [auth-options]
hostssl          database  user  IP-address  IP-mask      auth-method  [auth-options]
hostnossl        database  user  IP-address  IP-mask      auth-method  [auth-options]
hostgssenc       database  user  IP-address  IP-mask      auth-method  [auth-options]
hostnogssenc     database  user  IP-address  IP-mask      auth-method  [auth-options]
include          file
include_if_exists   file
include_dir      directory
```

Unter https://www.postgresql.org/docs/current/auth-pg-hba-conf.html ist die Syntax in allen Details dokumentiert. In diesem Dokument werden jetzt nur die wichtigsten Parameter und die Ausprägung auf `athanassiou.me` beschrieben.

`local`	Verbindungtyp für lokale Unix-Domain-Sockets verwenden (psql Client). Ohne einen Eintrag dieses Typs sind lokale Unix-Domain-Socket-Verbindungen nicht zulässig.
`host`	Verbindungen über TCP/IP.
`database`	Gibt an, den Datenbanknamen an, auf den zugegriffen werden kann: • Der Wert `all` gibt an, dass er allen Datenbanken nutzen kann. • Der Wert `sameuser` gibt an, das angeforderte Datenbank den gleichen Namen wie der angeforderte Benutzer haben soll • Der Wert `samerole` gibt an, dass der angeforderte Benutzer Mitglied der Rolle mit dem gleichen Namen wie die angeforderte Datenbank sein muss.

<table>
<tr><td>user</td><td colspan="2">Datenbank Username. Der Wert <code>all</code> lässt alle Benutzer übereinstimmt. Andernfalls handelt es sich entweder um den Namen eines bestimmten Datenbankbenutzers, oder um einen regulären Ausdruck (beginnt mit /).</td></tr>
<tr><td>adress</td><td colspan="2">Gibt die Adresse(n) des Clientcomputers an. Dieses Feld kann entweder einen Hostnamen, einen IP-Adressbereich oder ein spezielles Schlüsselwort enthalten. Die IP-address/mask-length Notation kann benutzt werden.</td></tr>
<tr><td rowspan="5">auth-method</td><td colspan="2">Gibt die Authentifizierungsmethode an, die verwendet werden soll. Hier gibt es eine Reihe von Optionen, die wichtigsten sind:</td></tr>
<tr><td><code>trust</code></td><td>ohne Einschränkung</td></tr>
<tr><td><code>peer</code></td><td>Ruft den Usernamen des Clients vom Betriebssystem ab, und prüft, ob er mit dem angeforderten Datenbankbenutzernamen übereinstimmt. Dies ist nur für lokale Verbindungen verfügbar.</td></tr>
<tr><td><code>cert</code></td><td>SSL Zertifikatsbasiert</td></tr>
<tr><td><code>password</code></td><td>Erfordert, dass der Client ein unverschlüsseltes Kennwort für die Authentifizierung angibt.</td></tr>
<tr><td></td><td><code>md5</code></td><td>Führt SCRAM-SHA-256- oder MD5-Authentifizierung durch, um das Kennwort des Benutzers zu überprüfen.</td></tr>
</table>

Auf `athanassiou.me` sehen die Einträge in das `pg_hba.conf` recht simpel aus – da der Zugriff nur von einem Rechner erfolgen soll.

`/etc/postgresql/16/main/ pg_hba.conf:`

```
# Database administrative login by Unix domain socket
local   all             postgres                                peer

# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust

# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
host    all             akis            127.0.0.1/0             md5

# IPv6 local connections:
host    all             all             ::1/128                 scram-sha-256
```

Das heißt für IPv4 T:

1. Eine Verbindung vom gleichen Rechner wird ohne weiteres zugelassen
2. Eine IP-Verbindung aus jedem Netzwerk wird mit dem Nutzer „akis“ und dem entsprechenden verschlüsselten Password (postgres Passwort) zugelassen.

4.5 Client-seitige Konfiguration: Postico

Jetzt noch der Client, Auf dem Mac gibt es Postico als Client Software, das sowohl als lokaler als auch remote Client arbeiten kann.

Das Einrichten erfolgt ganz simpel über die Postico-GUI, hier muss der User „akis“ und sein postgres Passwort eingegeben werden - *that’s it.*