

FFI/Datenbanksysteme

PostgreSQL

Prof. Dr. Patrick Cato

Technische Hochschule Ingolstadt



FFI/Datenbanksysteme

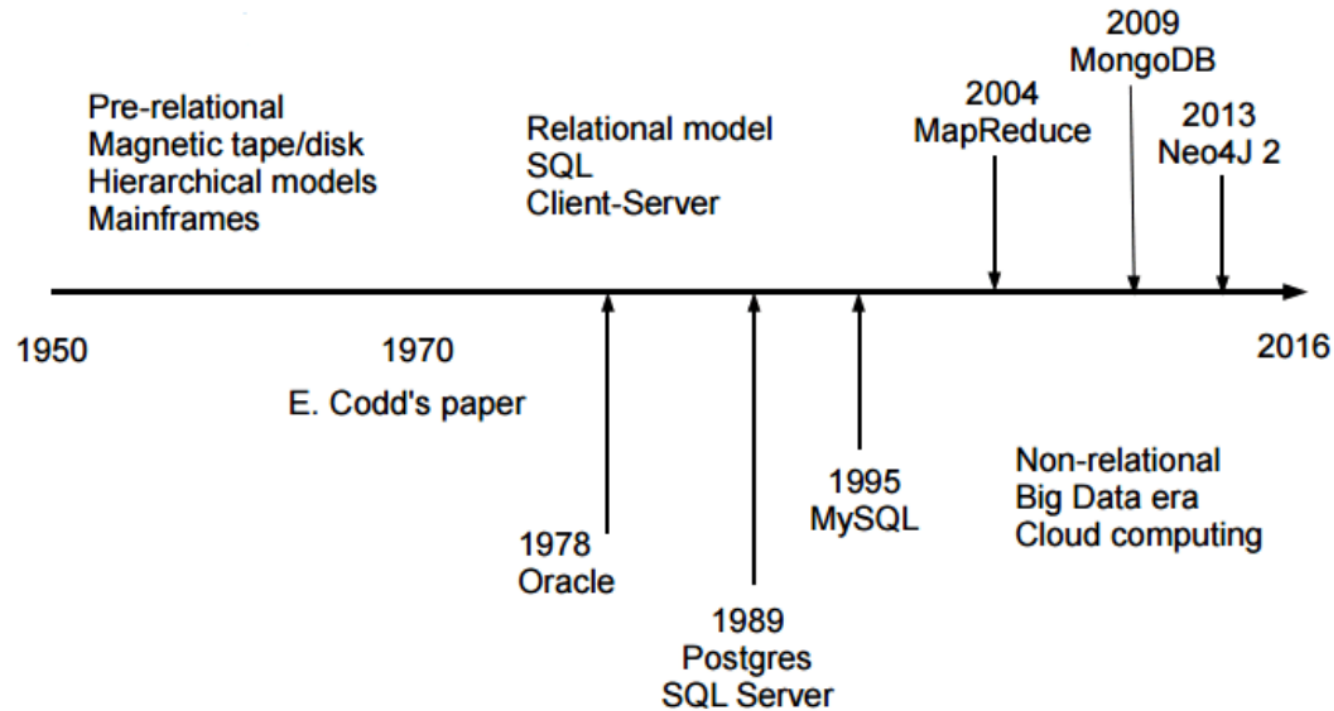
PostgreSQL

Prof. Dr. Patrick Cato

Technische Hochschule Ingolstadt 










[illegible]

Historie von PostgreSQL



— [Oracle](#), [MySQL](#), [Microsoft SQL Server](#), [PostgreSQL](#), [IBM Db2](#)

Top 5 systems in ranking, April 2023

Rank			DBMS	Database Model	Score		
Apr 2023	Mar 2023	Apr 2022			Apr 2023	Mar 2023	Apr 2022
1.	1.	1.	Oracle 	Relational , Multi-model 	1228.28	-33.01	-26.54
2.	2.	2.	MySQL 	Relational , Multi-model 	1157.78	-25.00	-46.38
3.	3.	3.	Microsoft SQL Server 	Relational , Multi-model 	918.52	-3.49	-19.94
4.	4.	4.	PostgreSQL 	Relational , Multi-model 	608.41	-5.41	-6.05
5.	5.	5.	IBM Db2	Relational , Multi-model 	145.49	+2.57	-14.97

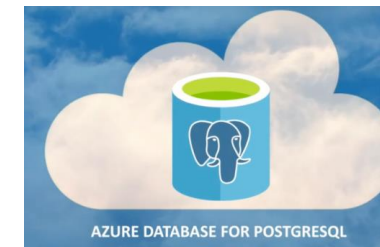
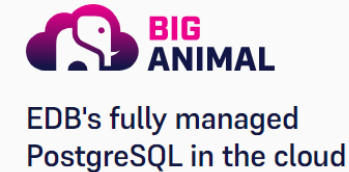
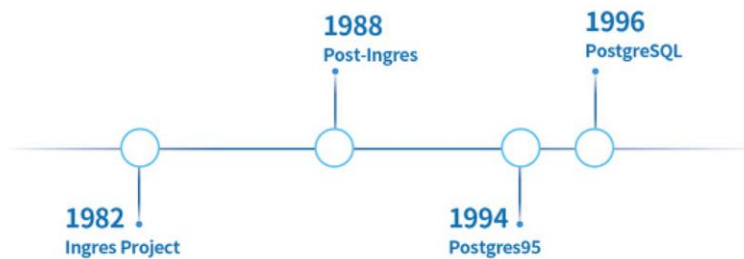
PostgreSQL – Geschichte des Systems (1/2)



<https://www.postgresql.org/>



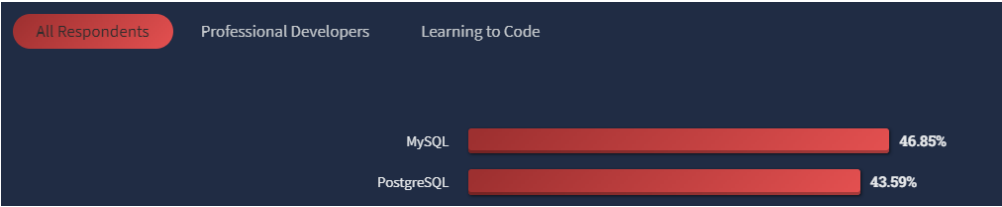
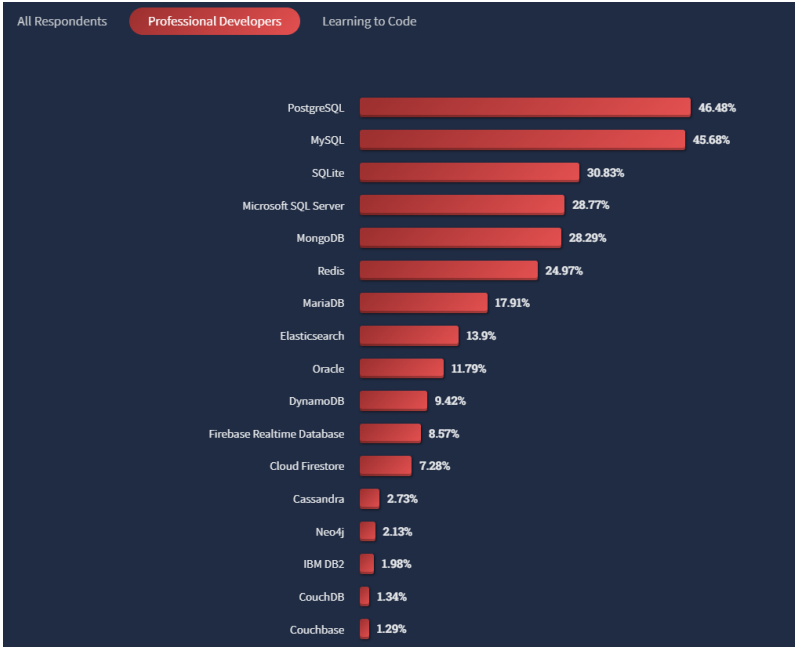
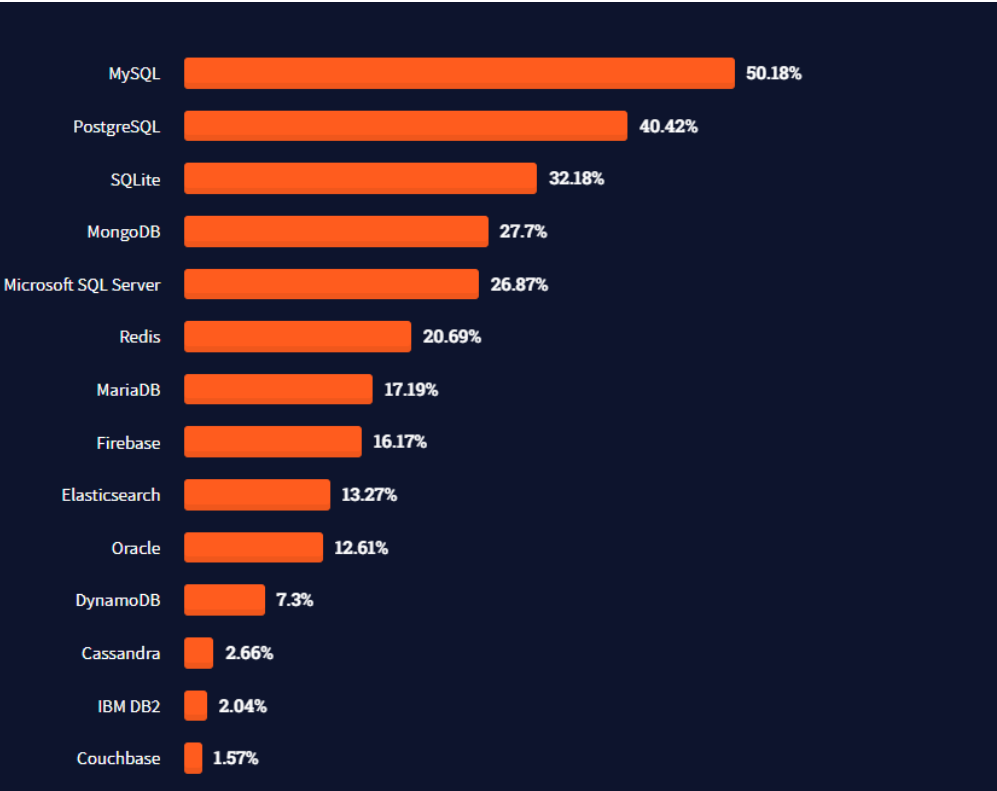
- Aus dem Projekt „Ingres“ an der University of California Berkeley entstanden. Projektleiter war Prof. Michael Stonebraker (1982)
- Sehr aktive **Open Source Community** und klare Vision
- 1988 erster **Prototyp** auf der **ACM SIGMOD Konferenz**
- PostgreSQL läuft auf **diversen Plattformen**: Microsoft Windows, UNIX, FreeBSD, Mac OS X, Solaris, HP-UX, LINUX, ...
- Heute nach DB-Engines.com Ranking das viert verbreitete relationale Datenbanksystem (nach Oracle, MySQL, Microsoft SQL Server)
- in verschiedenen Distributionen erhältlich
- Viele Systeme **implementieren nur die APIs** nach (Postgres-Kompatibilität)



- Das Post-Ingres Projekt endete 1994
- 1994 entwickelten Andrew Yu und Jolly Chen (Studenten in Berkeley) eine Version **mit SQL-Unterstützung** (statt der Datenbank-Sprache QUEL/POSTQUEL)
- Die erste Version erschien 1995 (“Postgres95”). Die **freie Weiterentwicklung** war möglich, weil POSTGRES unter einer großzügigen Open Source Lizenz herausgegeben wurde (**MIT Lizenz**)
- Der Name “PostgreSQL” wurde 1996 eingeführt
- Release History:

	Released	Release notes	Announcement	Latest minor release	GIT branch	EOL
PostgreSQL 18						
PostgreSQL 17					master	
PostgreSQL 16	2023-09-14	release notes	announcement	16.0 (2023-09-14)	REL_16_STABLE	2028-11
PostgreSQL 15	2022-10-13	release notes	announcement	15.4 (2023-08-10)	REL_15_STABLE	2027-11
PostgreSQL 14	2021-09-30	release notes	announcement	14.9 (2023-08-10)	REL_14_STABLE	2026-11
PostgreSQL 13	2020-09-24	release notes	announcement	13.12 (2023-08-10)	REL_13_STABLE	2025-11
PostgreSQL 12	2019-10-03	release notes	announcement	12.16 (2023-08-10)	REL_12_STABLE	2024-11
PostgreSQL 11	2018-10-18	release notes	announcement	11.21 (2023-08-10)	REL_11_STABLE	2023-11
PostgreSQL 10	2017-10-05	release notes	announcement	10.23 (2022-11-10)	REL_10_STABLE	2022-11
PostgreSQL 9.6	2016-09-29	release notes	announcement	9.6.24 (2021-11-11)	REL9_6_STABLE	2021-11
PostgreSQL 9.5	2016-01-07	release notes	announcement	9.5.25 (2021-02-11)	REL9_5_STABLE	2021-02

Stackoverflow Developer Survey 2021 / 2022



<https://insights.stackoverflow.com/survey/2021#technology>
<https://survey.stackoverflow.co/2022/>



Grundarchitektur

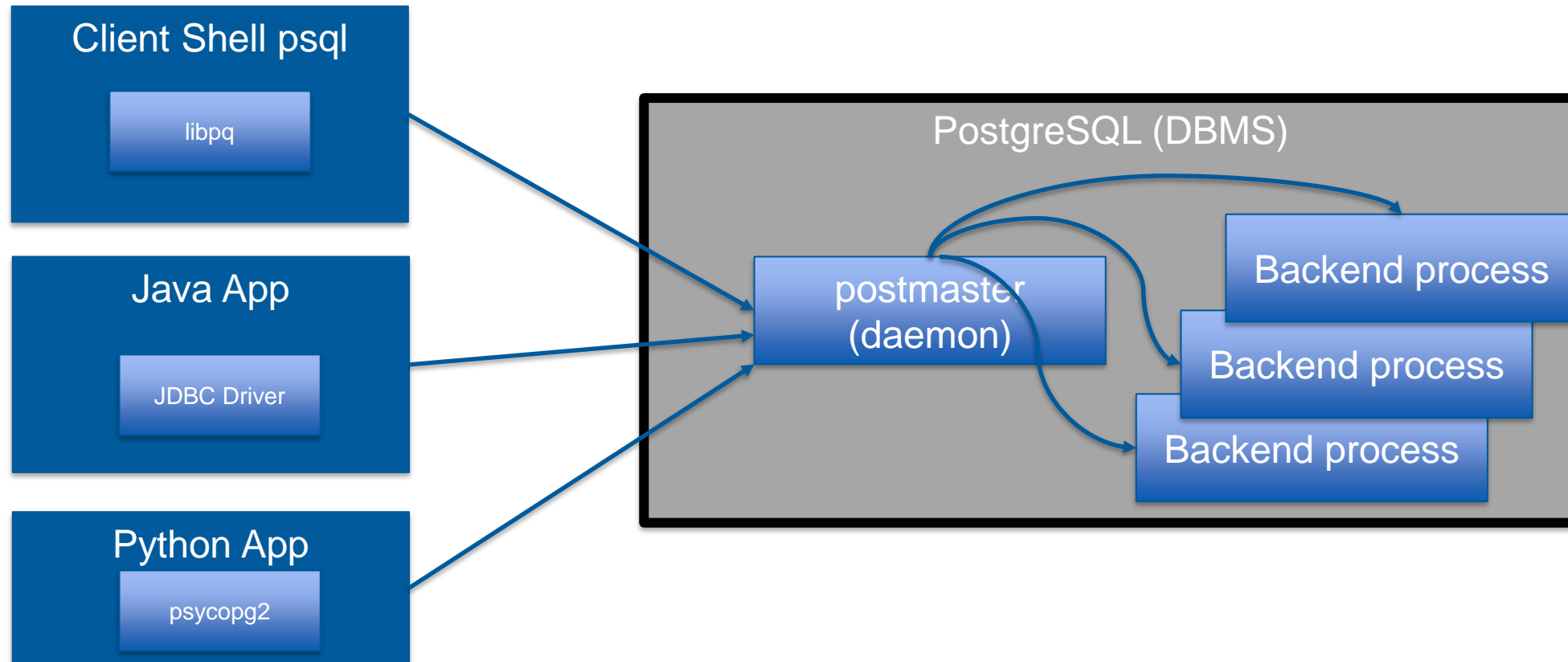
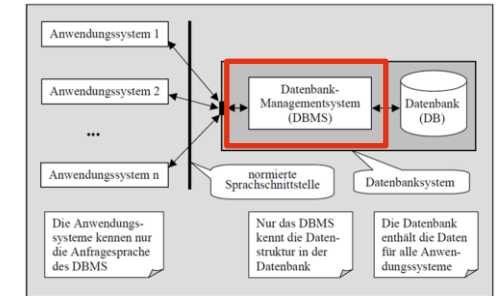
*Das Datenbankmanagementsystem ist vergleichbar mit einem Tower am Flughafen:
Es hat den Gesamtüberblick über das Rollfeld und steuert die Ressourcen*



Aufgaben des DBMS

- Organisation und Strukturierung der Daten
- Kontrolle der lesenden und schreibenden Zugriffe auf die Datenbasis
- Query Optimization und Indexierung
- Rollen und Rechte

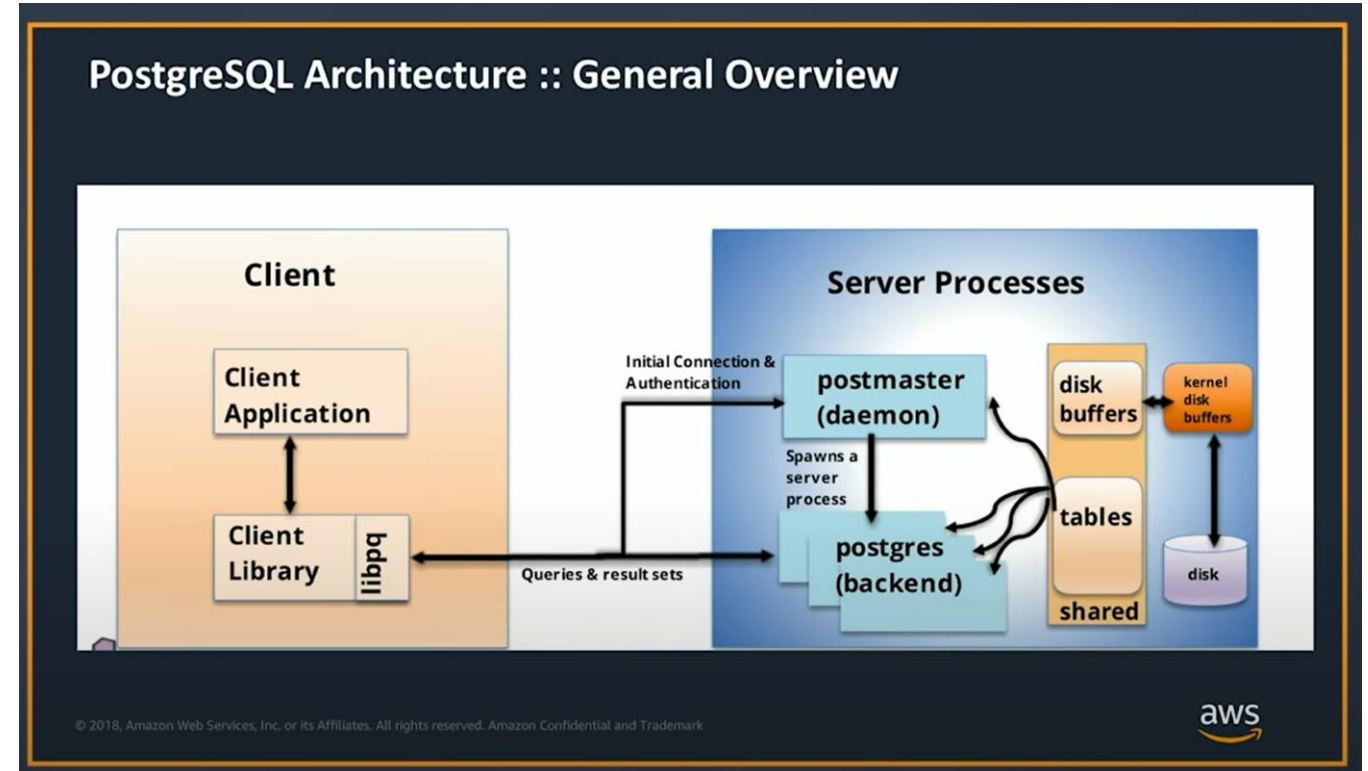




Architektur von PostgreSQL



- PostgreSQL ist ein **Client-Server-System**
- Der Server besteht aus **mehreren Prozessen** und einen “Shared Memory” Bereich
- Die Kommunikation zwischen Client und Server erfolgt über eine **Netzwerk-Verbindung**
 - Wenn ein Client und Server auf dem gleichen **Linux-Rechner laufen**, wird ein “UNIX-Domain-Socket” zur Kommunikation verwendet (Inter-Prozess-Kommunikation **über temporäre Datei**). Unter Windows wird aber auch in diesem Fall TCP/IP benutzt
- Jede Verbindung wird von einer eigenen PID (backend) verwaltet
- Die normale **Kommandozeilen-Schnittstelle** ist das Programm **psql**

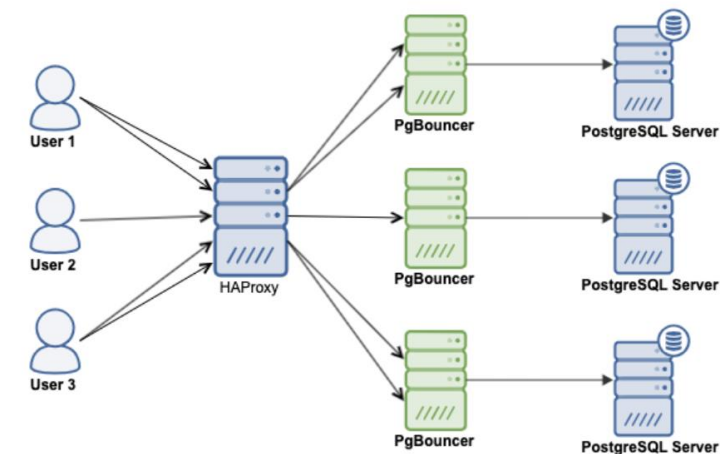
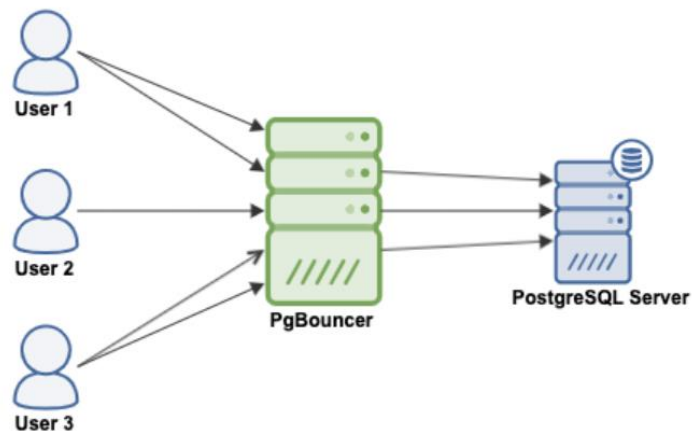


Prozess	Aufgabe
checkpointer	Schreibt bei einem Checkpoint die „Dirty Buffer“ auf die Disk.
writer	Schreibt periodisch „Dirty Buffer“ auf die Disk.
wal writer	Schreibt Sätze aus dem WAL Buffer in die WAL-Datei.
autovacuum launcher	Startet Autovacuum-Arbeiterprozesse, wenn ein VACUUM im laufenden Betrieb notwendig ist.
archiver	Wird gestartet, wenn das Cluster im Archivelog-Modus arbeitet. Kopiert die WAL-Datei ins Archiv-Verzeichnis.
stats collector	Sammelt Statistiken, unter anderem über Sessions und Tabellenbenutzung.
bgworker	Verschiedene Arbeiterprozesse, die von Hintergrundprozessen gestartet werden.

HINWEIS: Unter Windows laufen die Hintergrundprozesse bedingt durch die Architektur des Betriebssystems als Threads unter dem Hauptprozess „postgres.exe“. Die Threads können mit geeigneten Werkzeugen, zum Beispiel mit dem Windows Process Explorer, sichtbar gemacht werden.

Connection Pooling

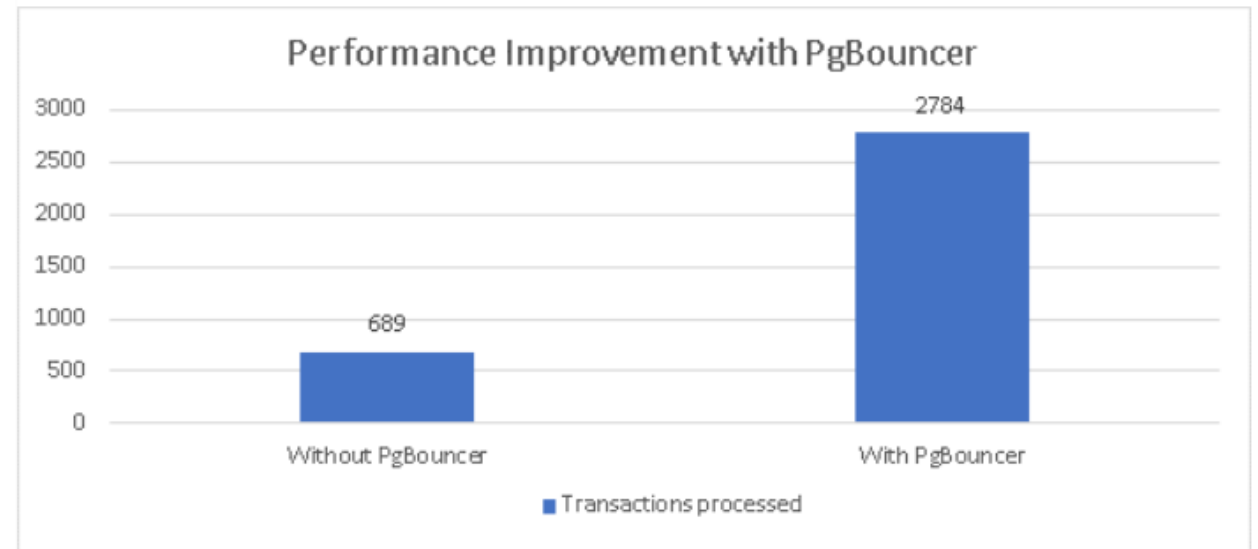
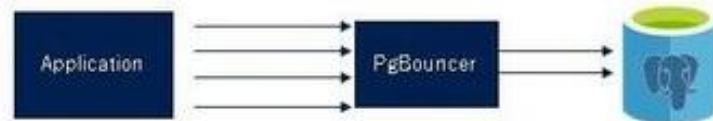
Unter einem Connection Pool versteht man einen Cache von bestehenden Datenbankverbindungen, der für Anfragen verwendet wird. Der Client greift nicht direkt auf die Datenbank, sondern den Connection Pool zu. In PostgreSQL heißt dieser leichtgewichtige Dienst „pgbouncer“.



No Connection Pooling



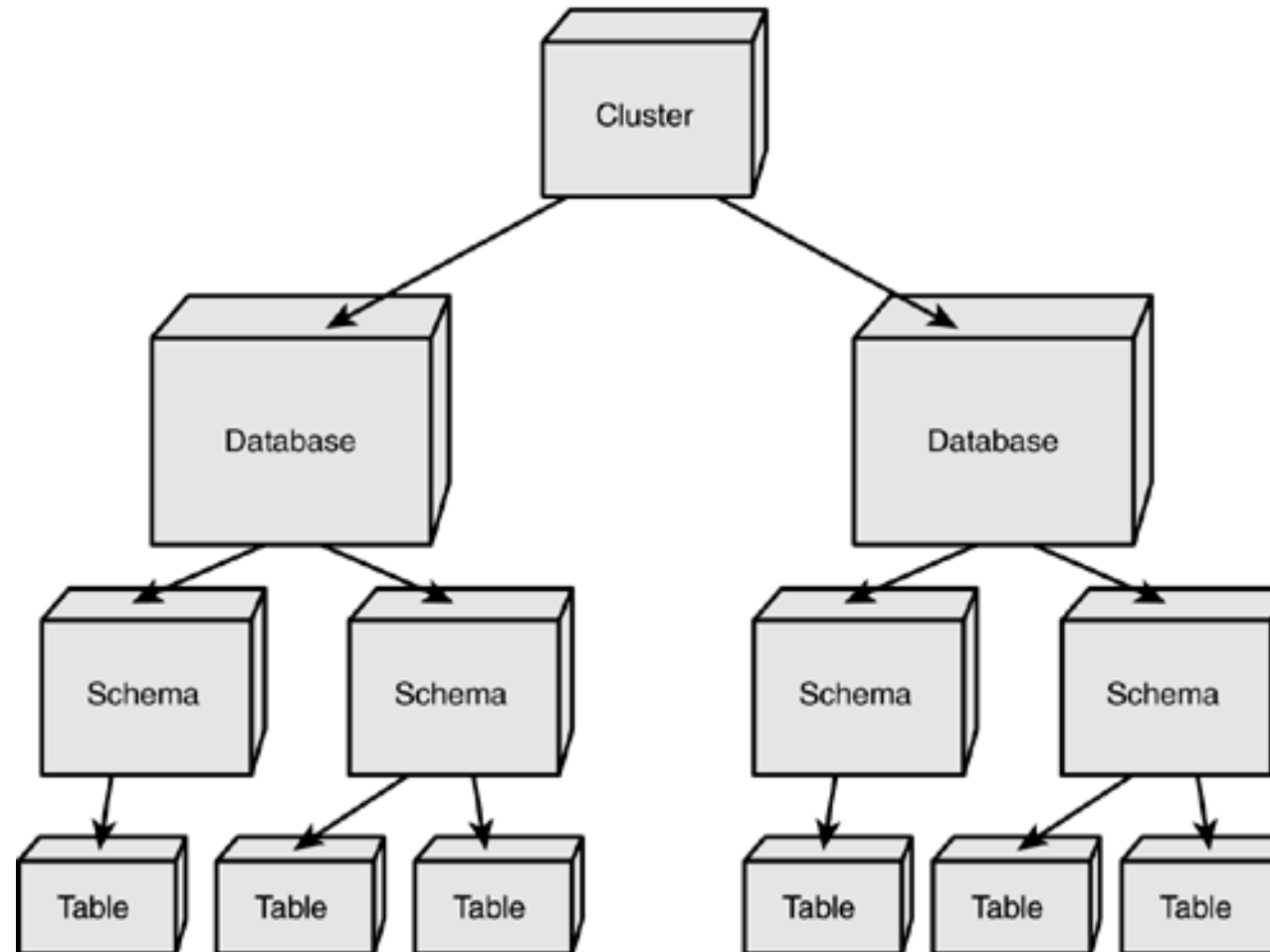
Connection Pooling using PgBouncer proxy service



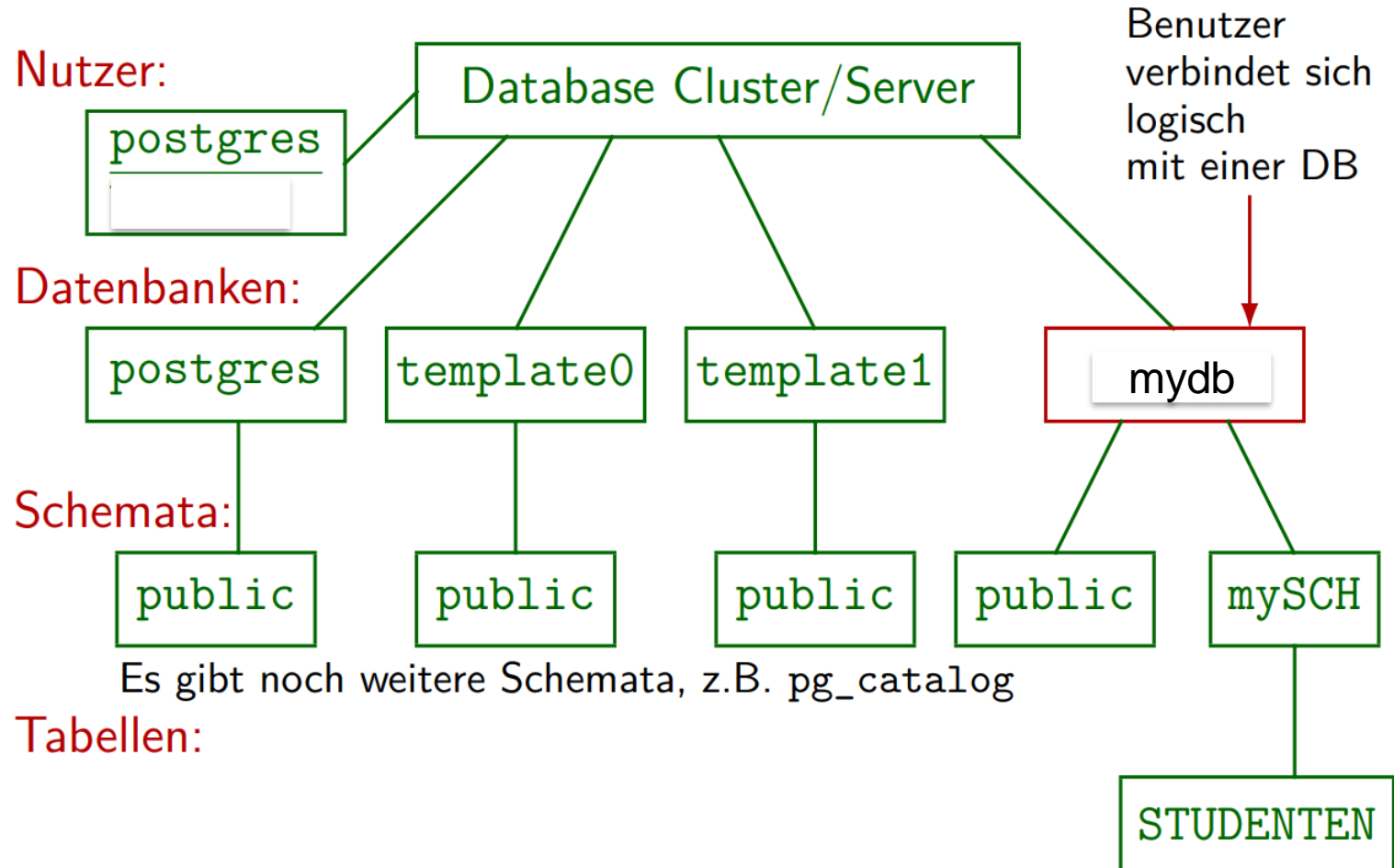
Definition

Ein “Database Cluster” in PostgreSQL besteht aus mehreren Datenbanken, die von einer Instanz des Datenbank-Servers verwaltet werden. Bei anderen Datenbanksystemen wäre ein “Cluster” eher eine Gruppe von Datenbank-Servern (mehrere Server, um Performance und Zuverlässigkeit zu steigern)

Hierarchie: Cluster, Database, Schema, Table



- Nachdem ein DB-Cluster angelegt wurde (mit initdb im Rahmen der Installation), werden darin drei Datenbanken angelegt: **postgres**, **template0** und **template1**
 - Die Datenbank postgres gibt es von Anfang an, man kann sich mit ihr verbinden, um andere Datenbanken anzulegen (oder wenn man sonst keine Datenbank-Namen kennt).
 - Nutzer-Datenbanken werden normalerweise durch **Clonen von template1 angelegt**. Es gibt zwei Template-Datenbanken, weil man template1 lokal modifizieren kann (z.B. würden dort angelegte Tabellen automatisch mitkopiert), aber template0 das Original bleiben soll. Direkt nach der Installation sind beide Template-Datenbanken gleich.



pg_catalog

- Jede Datenbank hat ein Schema „pg_catalog“ für die System-Tabellen (“Data Dictionary”)
- In pg_catalog werden folgende Metadaten verwaltet:
 - Definierte Secondary Indexe
 - Metriken
 - Definierte Constraints (PK/FK)
 - Berechtigungen auf Tabellen (pg_authid)
 - ... (>62 Tabellen)



Beispiel „pg.authid“



Data Output Explain Messages Notifications												
	oid [PK] oid	rolname name	rolsuper boolean	rolinherit boolean	rolcreatorole boolean	rolcreatedb boolean	rolcanlogin boolean	rolreplication boolean	rolbypassrls boolean	rolconnlimit integer	rolpassword text	
1	10	postgres	true	true	true	true	true	true	true	-1	SCRAM-SHA-256\$4096:xRqp4M/U	
2	3373	pg_monitor	false	true	false	false	false	false	false	-1	[null]	
3	3374	pg_read_all_settings	false	true	false	false	false	false	false	-1	[null]	
4	3375	pg_read_all_stats	false	true	false	false	false	false	false	-1	[null]	
5	3377	pg_stat_scan_tables	false	true	false	false	false	false	false	-1	[null]	
6	4200	pg_signal_backend	false	true	false	false	false	false	false	-1	[null]	
7	4569	pg_read_server_files	false	true	false	false	false	false	false	-1	[null]	
8	4570	pg_write_server_files	false	true	false	false	false	false	false	-1	[null]	
9	4571	pg_execute_server_program	false	true	false	false	false	false	false	-1	[null]	
10	6171	pg_database_owner	false	true	false	false	false	false	false	-1	[null]	
11	6181	pg_read_all_data	false	true	false	false	false	false	false	-1	[null]	
12	6182	pg_write_all_data	false	true	false	false	false	false	false	-1	[null]	

Datentypen

- SQL kennt eine **Vielzahl von Datentypen**. Diese werden unterschiedlich in den einzelnen DBMS umgesetzt, aber Strings und Zahlen (verschiedener Länge und Genauigkeit) sind immer verfügbar
- Jede Spalte kann **nur Werte eines bestimmten Datentyps** speichern
- Moderne (objektrelationale) Systeme bieten auch benutzerdefinierte Datentypen (Erweiterbarkeit). Z. B. PostgreSQL, DB2, Oracle und SQL Server unterstützen benutzerdefinierte Typen

Relativ standardisiert:

- Zeichenketten (feste Länge, variable Länge)
- Zahlen (Integer, Fest- und Gleitkommazahlen)

Unterstützt, aber in jedem DBMS verschieden:

- Datums- und Zeitwerte
- Lange Zeichenketten
- Binäre Daten
- Zeichenketten mit nationalem Zeichensatz
- Benutzerdefinierte und DBMS-spezifische Datentypen

CHARACTER (n)

- Zeichenkette fester Länge mit n Zeichen
- Daten, die in einer Spalte mit diesem Datentyp gespeichert werden, werden mit Leerzeichen bis zur Länge n aufgefüllt. **Also wird immer Plattenspeicher für n Zeichen benötigt.**
- Variiert die Länge der Daten stark, sollte man VARCHAR verwenden, siehe nächste Folie
- CHARACTER (n) kann als CHAR (n) abgekürzt werden
- Wird keine Länge angegeben, wird 1 angenommen. Somit erlaubt "CHAR"(ohne Länge) das Speichern einzelner Zeichen
- Das **maximale n ist je nach System begrenzt.** $n < 255$ sollte sehr portabel sein. Postgres: darf nicht größer als 10.485.760 Zeichen sein

VARCHAR (n)

- Falls man keine Längenbeschränkung angeben möchte, kann ein Attribut auch mit VARCHAR (ohne n) deklariert werden. (PostgreSQL)
- Dieser Datentyp wurde im SQL-92-Standard hinzugefügt (im SQL-86-Standard nicht enthalten), wird jedoch in allen modernen DBMS unterstützt n < 255 sollte portabel sein, n 4000 geht meist. Der volle Name ist CHARACTER VARYING(n)
- Postgres: darf nicht größer als 10.485.760 Zeichen sein

TEXT

- Jedes moderne System erlaubt auch längere Zeichenketten bis zu ganzen Dateien (als Tabelleneintrag)
- Die Details sind systemabhängig
- TEXT in PostgreSQL erlaubt bis zu 1 GB, in Microsoft SQL Server bis zu 2 GB.
- Oracle hat den Typ CLOB (max. 128 TB). Die Nutzung dieser Werte in Anfragen ist aber meist eingeschränkt.

INT

- ganze Zahl, Wertebereich ist implementierungsabhängig
- Integer-Datentypen eignen sich besonders für eindeutige Identifikationsnummern, z.B. Primärschlüssel einer Tabelle.
Der Zugriff auf solche Datenfelder erfolgt besonders schnell, da alle Prozessoren in modernen Computern für diese Zahlenwerte optimiert sind.
- Storage: 4 Bytes
- -2 147 483 648 to +2 147 483 647
- Falls Integer zu klein gibt es noch Datentyp **BIGINT** mit 8 Bytes (ca. +- 9 Trillionen). Falls das zu wenig => Datentyp Numeric
- **SMALLINT**: 2 Bytes (-32.768 bis +32.767)

NUMERIC(precision, scale)

- Vorzeichenbehaftete Zahl mit insgesamt p Ziffern (s Ziffern hinterm Komma) Z.B. erlaubt `NUMERIC(3, 1)` die Werte -99.9 bis 99.9
- Bei Zahlen, die die angegebene Präzision und Skalierung überschreiben, wird gerundet. Z.B. wird bei `NUMERIC(3, 1)` 33.34 auf 33.3 gerundet.
- Der numerische Datentyp kann 131 072 Zahlen vor und 16 383 Zahlen nach dem Komma speichern

Name	Storage Size	Description	Resolution
timestamp [(p)] [without time zone]	8 bytes	both date and time (no time zone)	1 microsecond
timestamp [(p)] with time zone	8 bytes	both date and time, with time zone	1 microsecond
date	4 bytes	date (no time of day)	1 day Format yyyy-mm-dd
time [(p)] [without time zone]	8 bytes	time of day (no date)	1 microsecond
time [(p)] with time zone	12 bytes	time of day (no date), with time zone	1 microsecond

```
-- Check timezone in psql
SHOW time zone;
+-----+
| TimeZone |
+-----+
| UTC      |
+-----+

-- Change timezone value
SET TIME ZONE 'Europe/Rome';
```

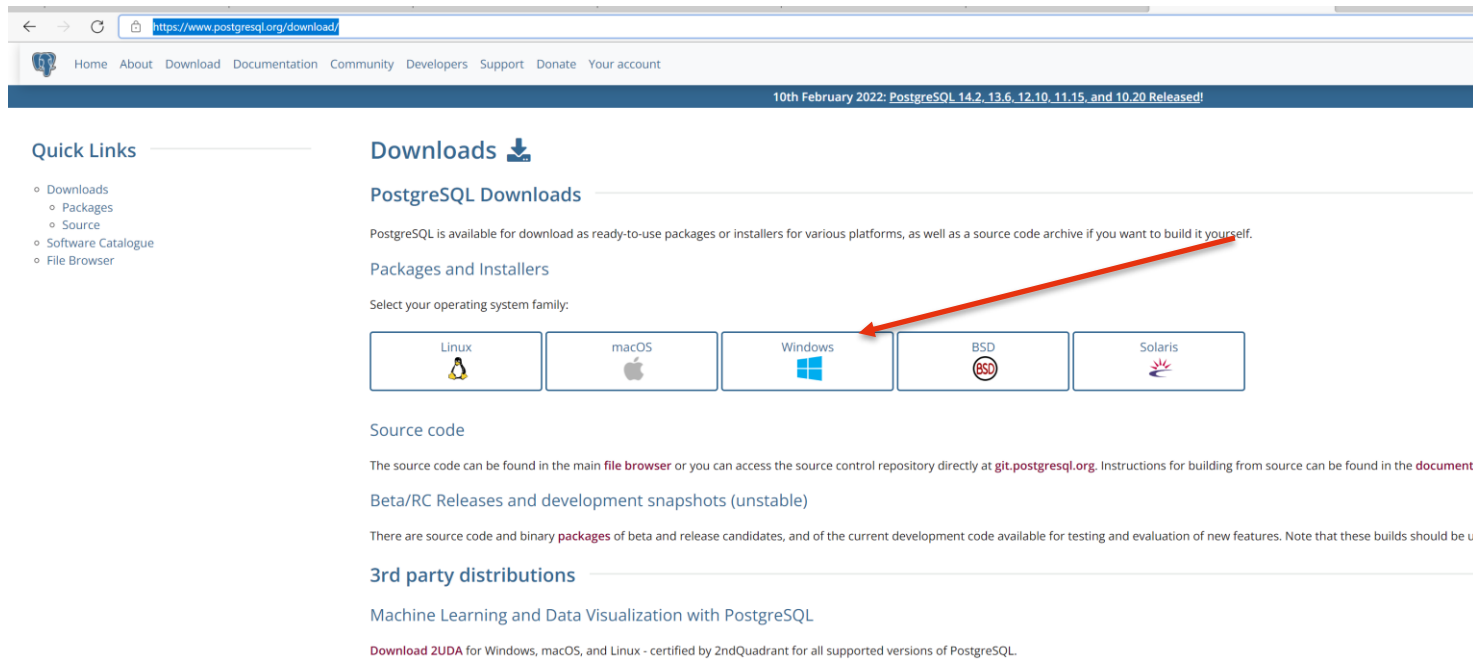
<https://www.postgresql.org/docs/current/datatype.html>

Installation

Sie finden die Installationsdateien auf

<https://www.postgresql.org/download/>

<https://www.postgresql.org/download/>






































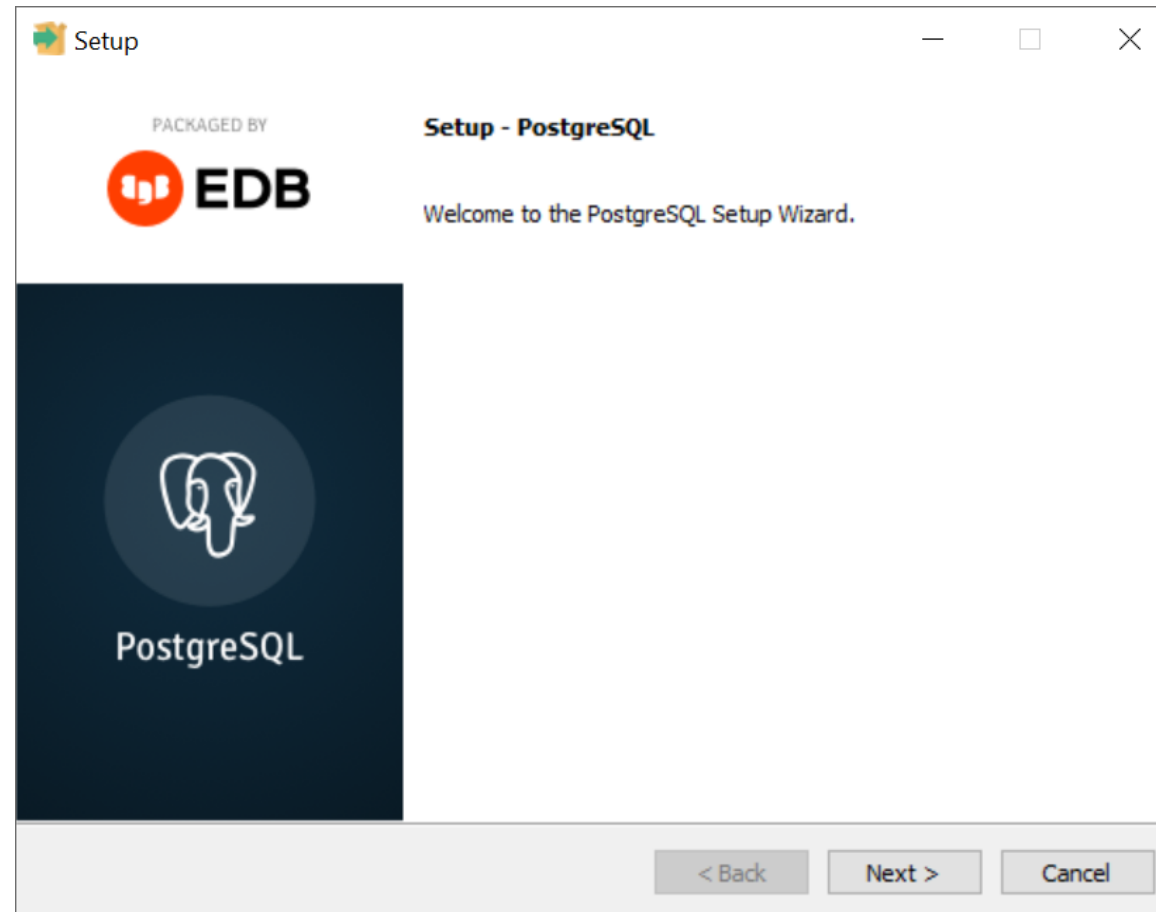
22.10.2023

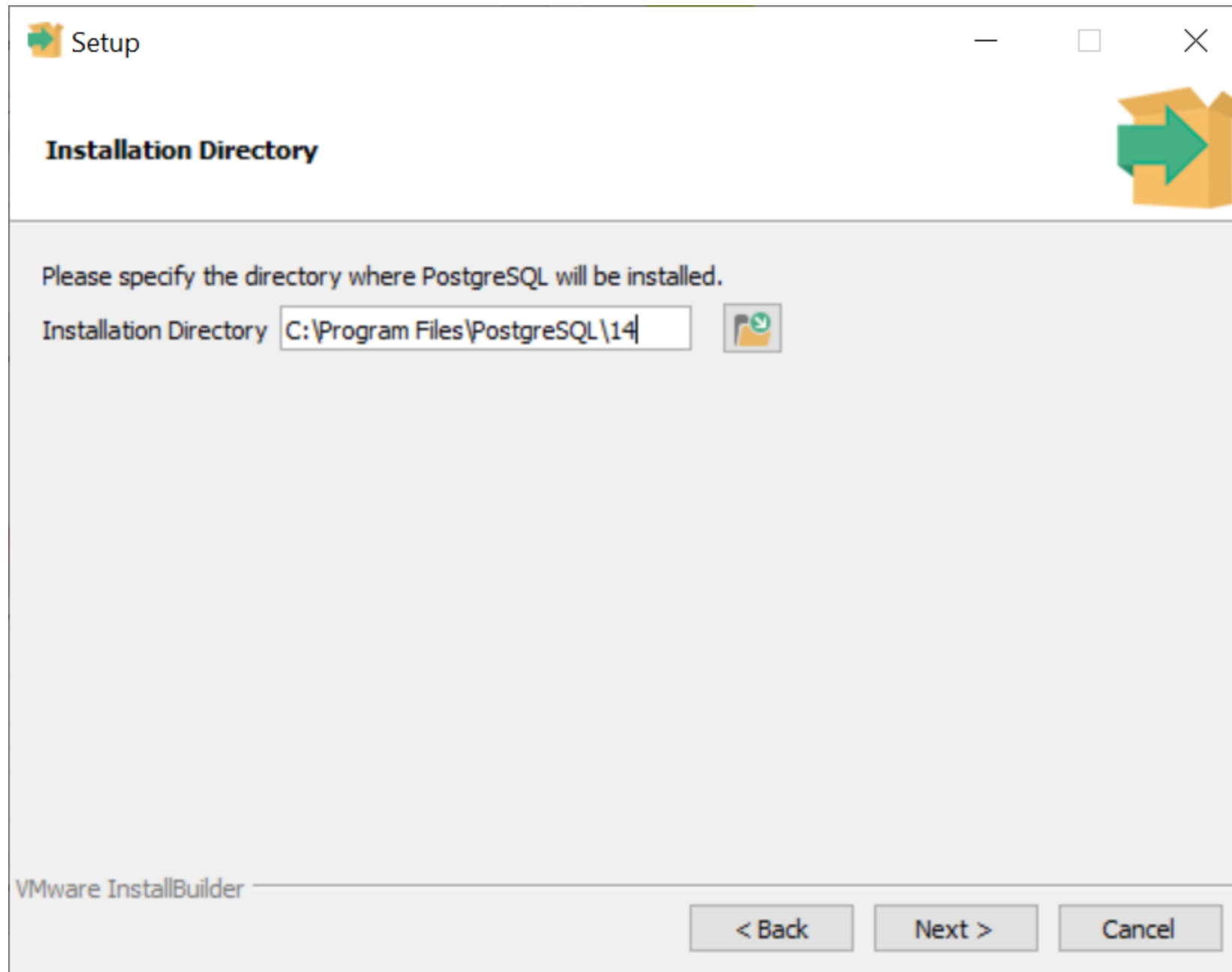
<https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>

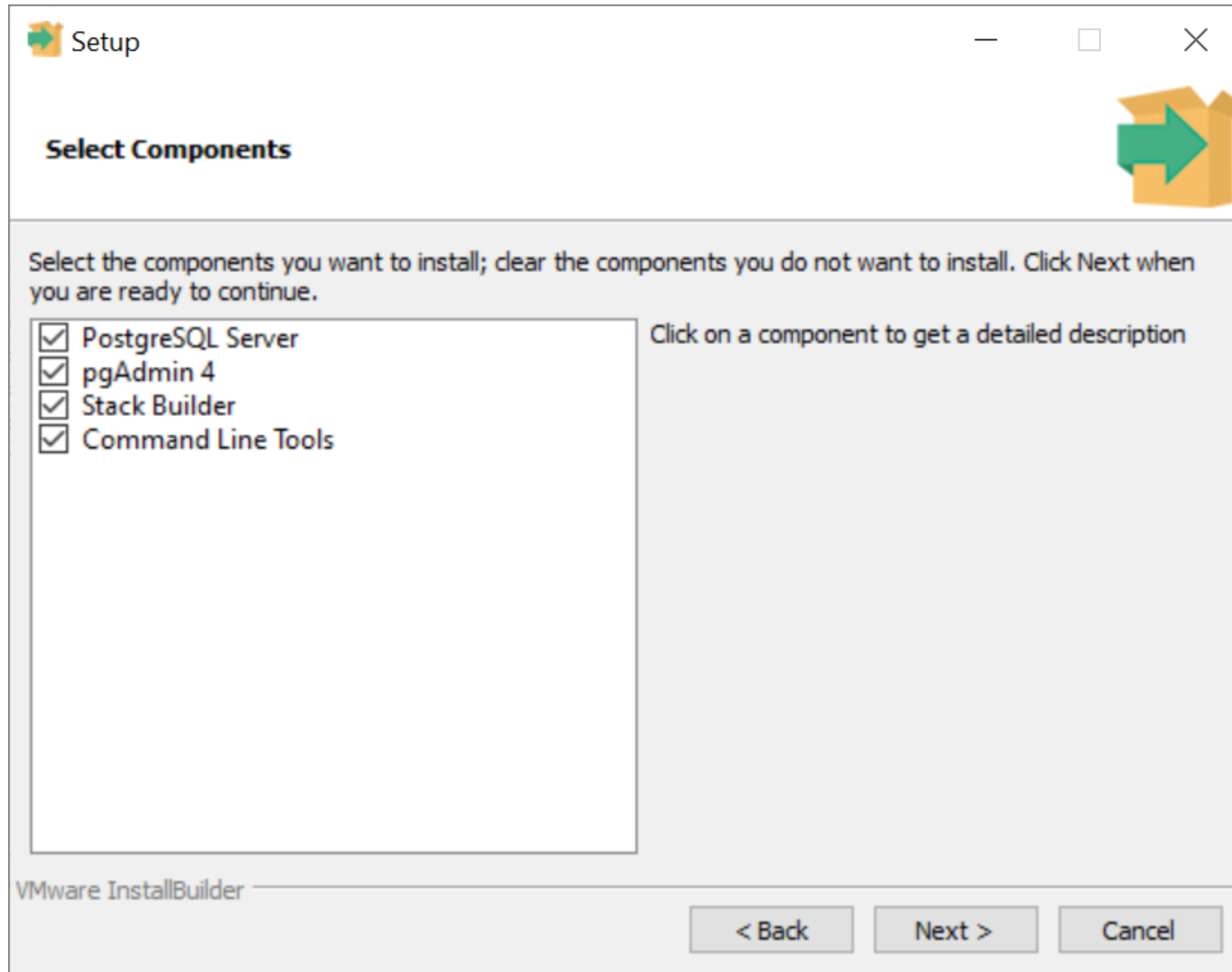
Download PostgreSQL

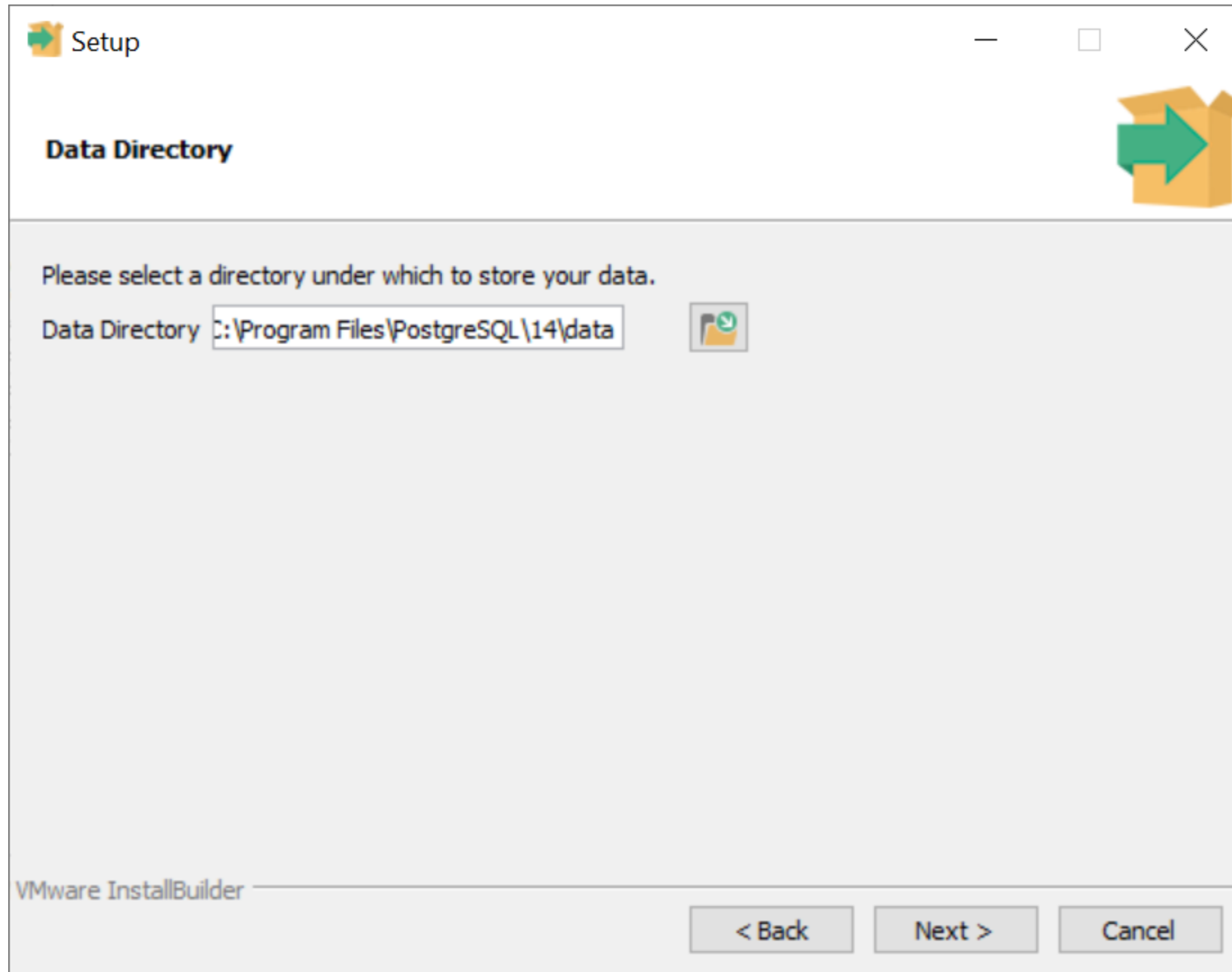
Open source PostgreSQL packages and installers from EDB

PostgreSQL Version	Linux x86-64	Linux x86-32	Mac OS X	Windows x86-64	Windows x86-32
15.0	postgresql.org	postgresql.org			Not supported
14.5	postgresql.org	postgresql.org			Not supported
13.8	postgresql.org	postgresql.org			Not supported
12.12	postgresql.org	postgresql.org			Not supported
11.17	postgresql.org	postgresql.org			Not supported
10.22					
9.6.24*					
9.5.25*					
9.4.26*					
9.3.25*					

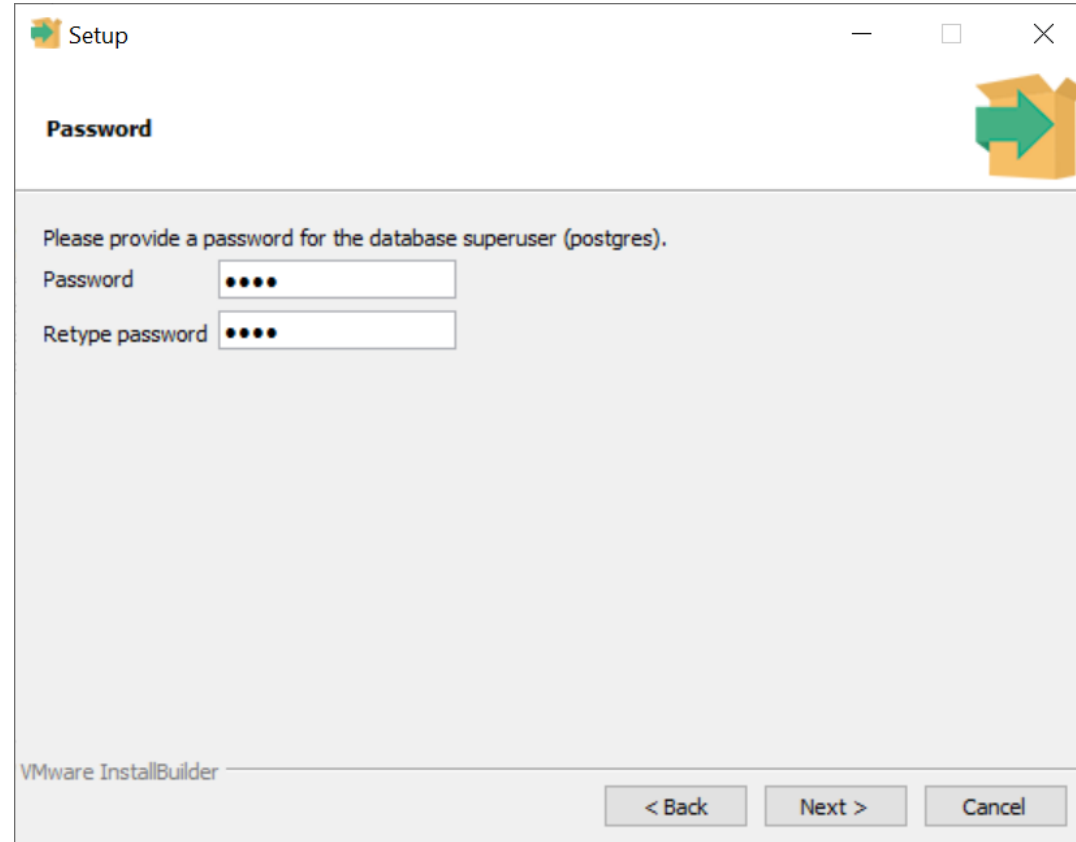








Gut merken. Sie benötigen dieses Passwort für pgadmin!



Setup

Password


Please provide a password for the database superuser (postgres).




Password


Retype password

VMware InstallBuilder

< Back Next > Cancel

 Setup







Port


Please select the port number the server should listen on.

Port

VMware InstallBuilder

 Setup

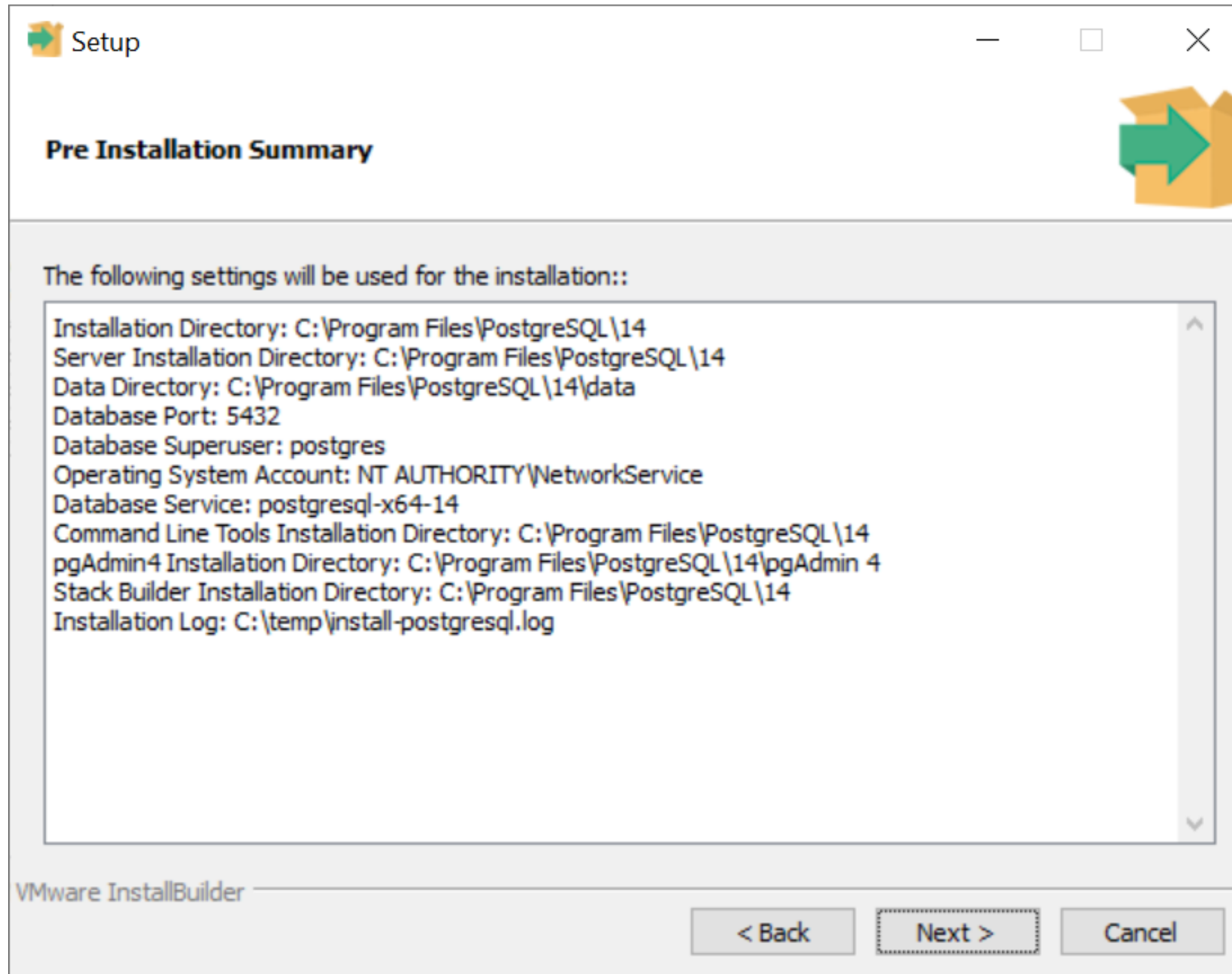


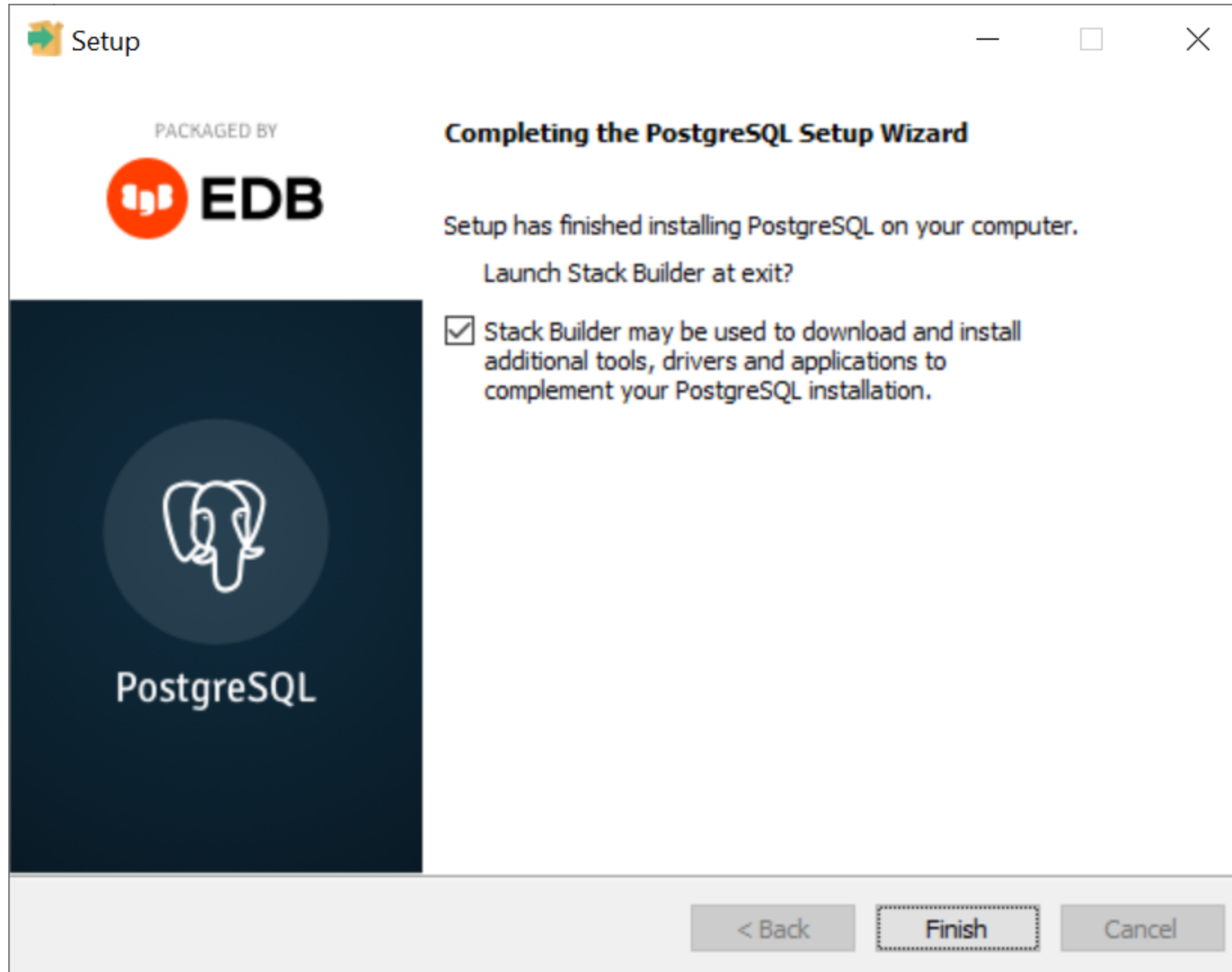
Advanced Options

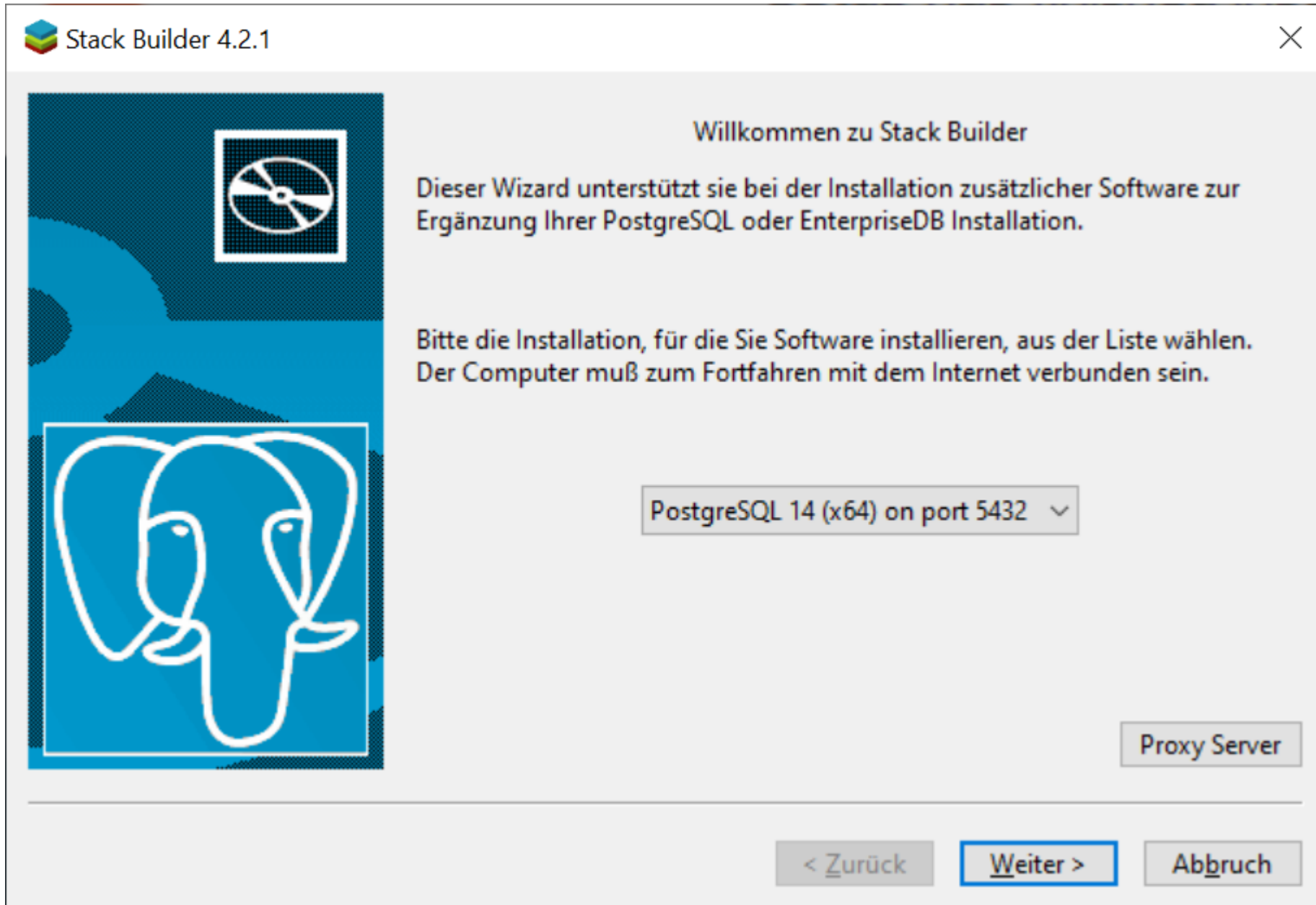
Select the locale to be used by the new database cluster.

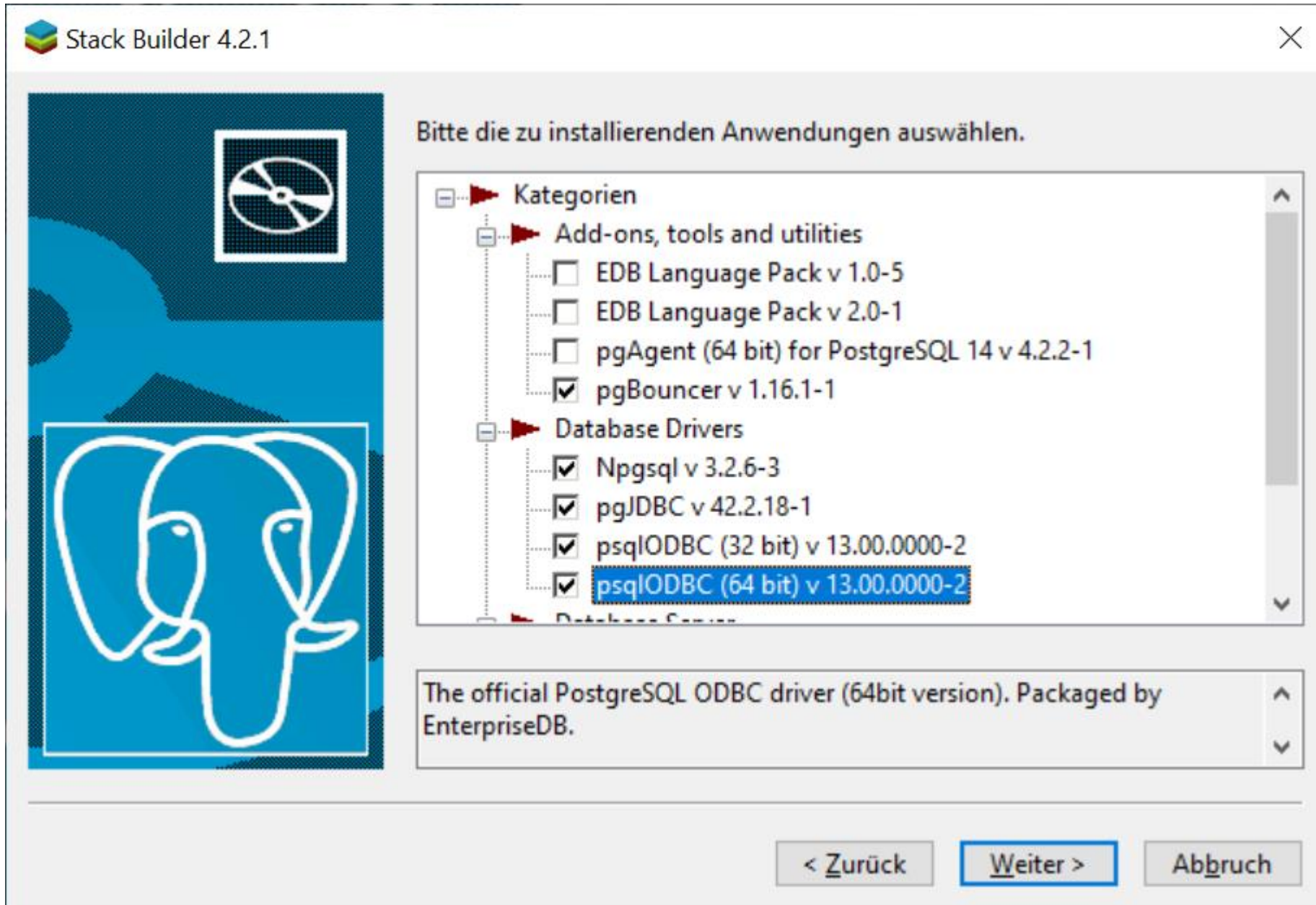
Locale

VMware InstallBuilder







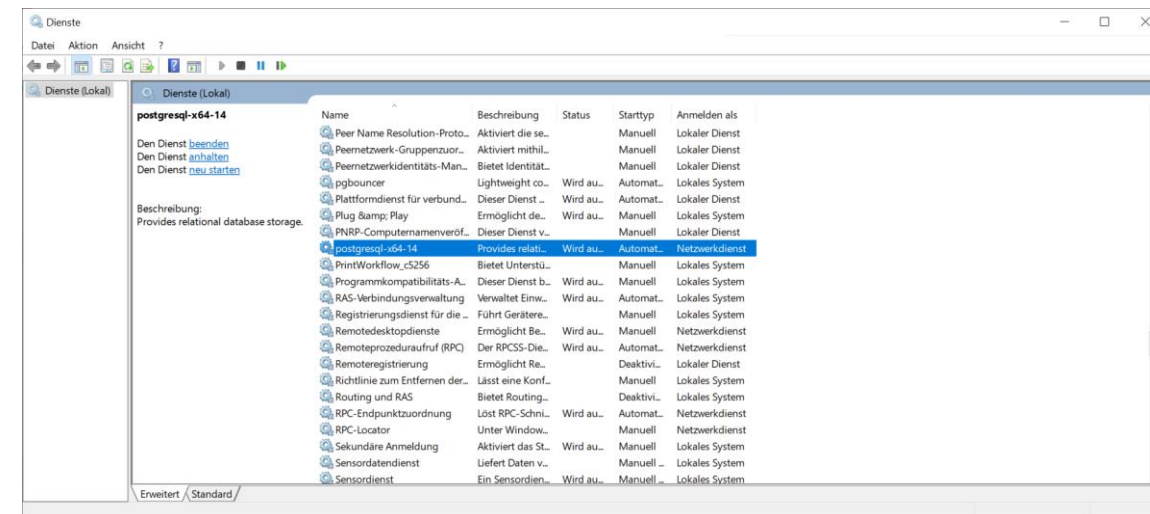
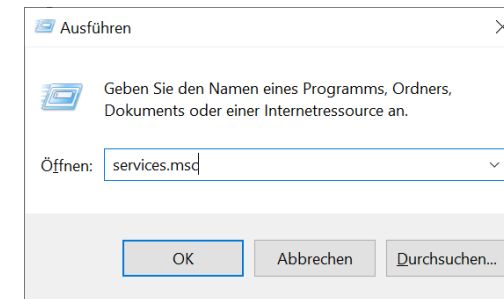


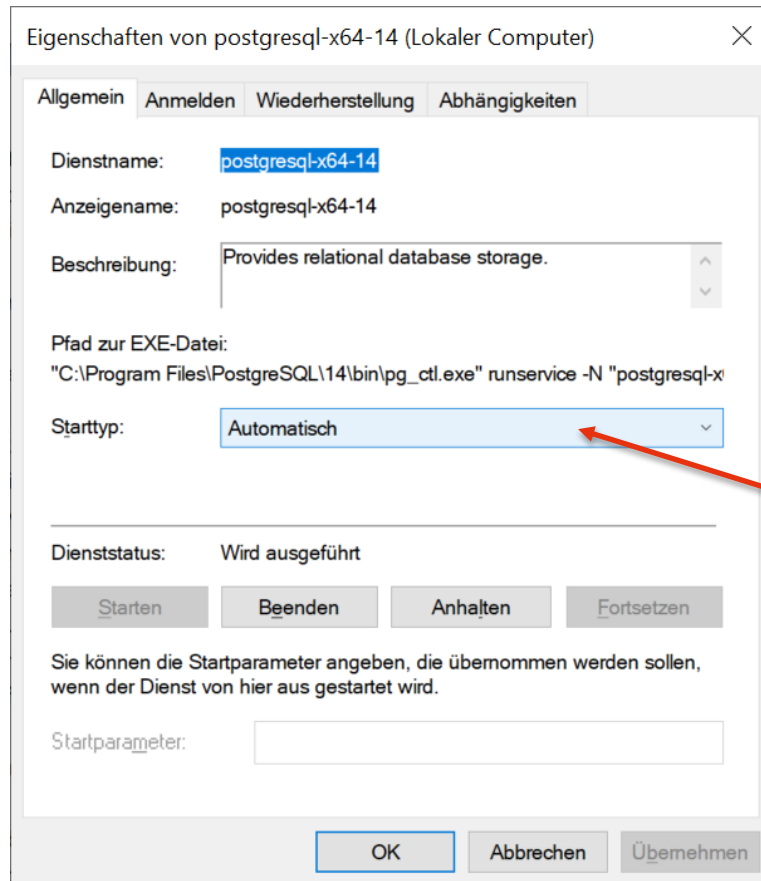
Service „postgresql“



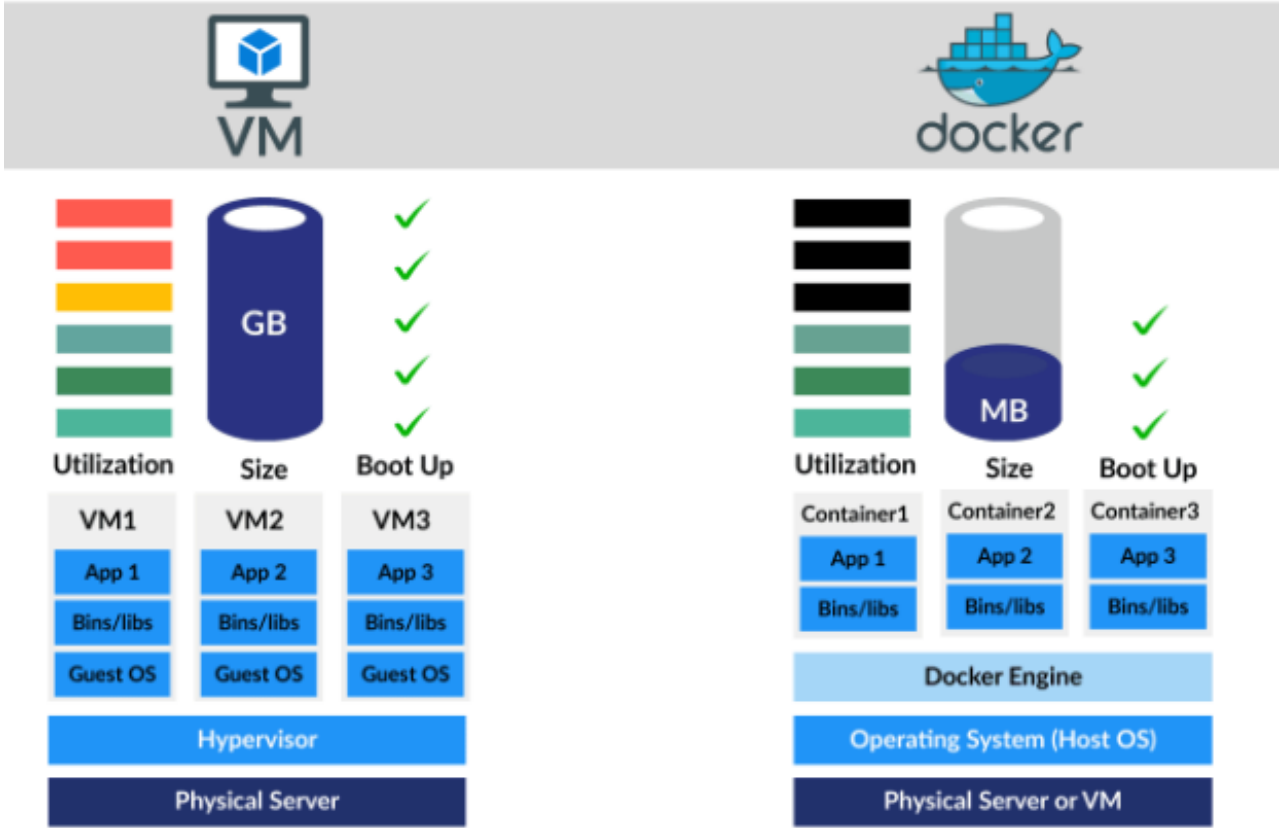
Die Datenbank startet automatisch beim Booten von Windows. Dies kann sehr ressourcenintensiv sein. Daher empfehle ich den Autostart zu deaktivieren und die Datenbank bei Bedarf manuell zu starten

1. Windowstaste und R gleichzeitig drücken
2. Services.msc eingeben
3. Posrgres und Pg admin Start auf manuell setzen (Eigenschaften)





Auf manuell setzen



Docker auf Windows:

<https://docs.docker.com/desktop/install/windows-install/>

Docker auf Linux (Ubuntu):

<https://docs.docker.com/engine/install/ubuntu/>

Starten einer postgres in Docker

docker pull postgres

```
$ docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
26c5c85e47da: Pulling fs layer
1c30a4c3f519: Pulling fs layer
d5c0f1ae682d: Pulling fs layer
1b1b2890ec0f: Pulling fs layer
391087799df7: Pulling fs layer
b413b4057e31: Pulling fs layer
4fa4edfeab8b: Pulling fs layer
b0a4d596bc61: Pulling fs layer
f6d73cd87199: Pulling fs layer
62b0bb33c69b: Pulling fs layer
bb0ddb7e7f1a: Pulling fs layer
583ec94d38ee: Pulling fs layer
efdf2a922e82: Pulling fs layer
b413b4057e31: Waiting
```

Ohne Datenpersistenz:

```
docker run --name mypg -e POSTGRES_PASSWORD=mysecretpassword -p 54320:5432 -d postgres
```

Mit Datenpersistenz:

```
docker volume create postgresdata
```

```
docker run --name mypg -e POSTGRES_PASSWORD=mysecretpassword -p 54320:5432 -v postgresdata:/var/lib/postgresql/data -d postgres
```

```
$ docker run --name mypg -e POSTGRES_PASSWORD=mysecretpassword -p 54320:5432 -v postgresdata:/var/lib/postgresql/data -d postgres
6f05688ad2e0229d2ffdd4206371c3566bb886fad56c3b0abc9d9c2ce26bcc14
```

Wenn Docker Container angelegt kann man starten stoppen mit: `docker start mypg` / `docker stop mypg`

exit

```
PS C:\Users\cato> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                    NAMES
6f05688ad2e0   postgres  "docker-entrypoint.s..." 44 hours ago  Up 3 minutes  0.0.0.0:54320->5432/tcp   mypg
PS C:\Users\cato> docker exec -it 6f bash
root@6f05688ad2e0:/#
```



```
sudo apt update
```

```
sudo apt-get install postgresql-15
```

- Wenn man **psql** aufruft, und der Server läuft nicht, bekommt man die Fehlermeldung 'could not connect to server'
- Man kann auch schauen, ob postgres Prozesse laufen
 - Unter Linux, z.B. **ps -ef | grep postgres**
 - Unter Windows mit dem Taskmanager (Ctrl+Alt+Delete)
- Man kann server-basierte Datenbanken so installieren, dass der Server beim Start des Betriebssystems automatisch mit gestartet wird, oder dass man ihn manuell startet muss (bei Bedarf). In Ubuntu z. B. **sudo systemctl enable postgresql**
- Bei vielen Datenbanken ist es sehr wichtig, sie vor dem Shutdown des Betriebssystems ordnungsgemäß herunterzufahren (Beim nächsten Hochfahren wird ein Recoveryverfahren gestartet, der entsprechende Ressourcen bindet)
- Mit **pg_ctl stop** kann der PostgreSQL Server geordnet heruntergefahren werden

```
sudo -u postgres pg_ctl -D /data/pg -l /data/logfile -w start
```

Mit -D wird das Hauptverzeichnis von Postgres angegeben, mit -l eine Logdatei für Fehlermeldungen (sonst Terminal, von dem pg_ctl aufgerufen).

Die Option -w bedeutet, dass pg_ctl wartet, bis der Server gestartet/gestoppt ist.

```
sudo -u postgres pg_ctl -D /data/pg -m fast -w stop
```

pgAdmin

PostgreSQL Tools

pgAdmin is the most popular and feature rich Open Source administration and development platform for PostgreSQL, the most advanced Open Source database in the world.

pgAdmin may be used on Linux, Unix, macOS and Windows to manage PostgreSQL and EDB Advanced Server 10 and above.



Verbindungsaufbau mit Server

- Wenn man nicht explizit einen Server angibt, verbindet sich psql über einen Unix-Domain Socket oder TCP/IP zu einem Server auf dem lokalen Rechner

Auf UNIX/Linux-Rechnern ist der Default "Unix-Domain Socket". Das kann scheitern, obwohl ein Zugriff über TCP/IP zu localhost möglich wäre. Man sollte probierhalber die Verbindungsdaten explizit angeben, wie unten gezeigt. Der TCP/IP Port ist normalerweise 5432 (aber konfigurierbar)

- Verbindung zu Port 5432 auf dem Rechner localhost, Datenbank thi:

```
psql -h localhost -p 5432 thi
```

```
psql -d postgres -U student
```

- Eine bekannte graphische Schnittstelle ist pgAdmin
- Es ist in erster Linie zur Administration gedacht, erlaubt aber auch, SQL-Anfragen zu stellen
- Enthält einen Browser für alle Datenbank-Objekte in einer Baum-Ansicht (Expand/Collapse)
- Das Programm pg_ctl (postgres control) dient u.a. zum Starten und Stoppen des Servers
- Die normale Kommandozeilen-Schnittstelle ist das Programm psql

Dies ist nur ein Client, der SQL Befehle entgegennimmt und an den Server schickt, dann das Ergebnis abholt und ausgibt

psql

- Man kann SQL-Befehle über mehrere Zeilen verteilt eingeben und muss sie dann mit einem Semikolon “;” abschließen
- Außerdem gibt es Befehle, die von psql selbst ausgeführt werden. Sie beginnen mit einem Rückwärts-Schrägstrich “\” und werden direkt ausgeführt, sobald Enter gedrückt wird

\h	Hilfe zu Anweisungen
\c	Kontext Wechsel auf Datenbank
\d	Liste der Relationen im aktuellen Kontext mit Constraint
\dt *	Alle Tabellen abfragen, auch mit Katalog
\d schemaname.*	Alle Tabellen mit Constraint für ein Schema auflisten
\dn	Liste der Schemas im aktuellen Kontext
\q	Beenden
\l+	Liste aller Datenbanken mit Speicher-Größe
\du	Liste der Rollen mit Rechten

CREATE database

Datenbank anlegen

DROP database

Datenbank löschen

\r

Reset Buffer

- Es ist wichtig, den Unterschied zwischen *psql*-Befehlen und SQL-Befehlen zu verstehen: *psql*-Befehle beginnen mit \ und erstrecken sich bis zum Ende der Zeile. Sie werden sofort ausgeführt (von *psql*, also dem Client)
- Alles andere hält *psql* für einen SQL-Befehl. Es sammelt ggf. mehrere Zeilen in einem Puffer, bis zu einem Semikolon “;”
- Die Eingabe kann auch mit speziellen *psql*-Befehlen beendet werden, wie z.B. \r (reset), was die Eingabe abbricht und den Puffer leert
- Beim “;” wird der SQL-Befehl an den Server geschickt, das Ergebnis geholt, und angezeigt.

Rollen- und Rechtekonzept

- Unter Linux laufen die Datenbankprozesse in der Regel unter dem Benutzer „postgres“. Dieser wird bei der Installation angelegt.
- DB user postgres und Linux user postgres hängen nicht zusammen
- In Produktionsbetrieb wird in der Regel nicht mit dem „Superuser“ postgres gearbeitet, sondern weitere Nutzer und Datenbanken mit eingeschränkten Rechten angelegt (auch um unbeabsichtigtes Löschen zu verhindern)
- Der Datenbankadministrator führt folgendes aus:
 - `CREATE ROLE <username> LOGIN CREATEDB \password testuser;`
- Siehe Dokumentation:
 - <https://www.postgresql.org/docs/14/sql-createrole.html>
 - <https://www.postgresql.org/docs/14/sql-grant.html>

OPTION	Beschreibung
CREATEDB	Erlaubt das anlegen von DBs
LOGIN	Erlaubt Login auf DB-Server
SUPERUSER	Alle Rechte
CREATE ROLE	Neue Rollen dürfen definiert werden

```
root@thi-server1:~# nano /etc/postgresql/14/main/pg_hba.conf
```

```
# DO NOT DISABLE!
# If you change this first entry you will need to make sure that the
# database superuser can access the database using some other method.
# Noninteractive access to all databases is required during automatic
# maintenance (custom daily cronjobs, replication, and similar tasks).
#
# 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                                peer
# IPv4 local connections:
host     all             all             127.0.0.1/32            scram-sha-256
# IPv6 local connections:
host     all             all             ::1/128                 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local    replication     all                                peer
host     replication     all             127.0.0.1/32            scram-sha-256
host     replication     all             ::1/128                 scram-sha-256
```

```
#
# 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 md5
# IPv4 local connections:
host all all 127.0.0.1/32 scram-sha-256
# IPv6 local connections:
host all all ::1/128 scram-sha-256
# Allow replication connections from localhost, by a user with the
# replication privilege.
local replication all peer
host replication all 127.0.0.1/32 scram-sha-256
host replication all ::1/128 scram-sha-256
```

```
root@thi-server1:~# nano /etc/postgresql/14/main/pg_hba.conf
```

But if you intend to force password authentication over Unix sockets instead of the peer method, try changing the following `pg_hba.conf` * line:

from

```
# TYPE DATABASE USER ADDRESS METHOD
local all all peer
```

to

```
# TYPE DATABASE USER ADDRESS METHOD
local all all md5
```

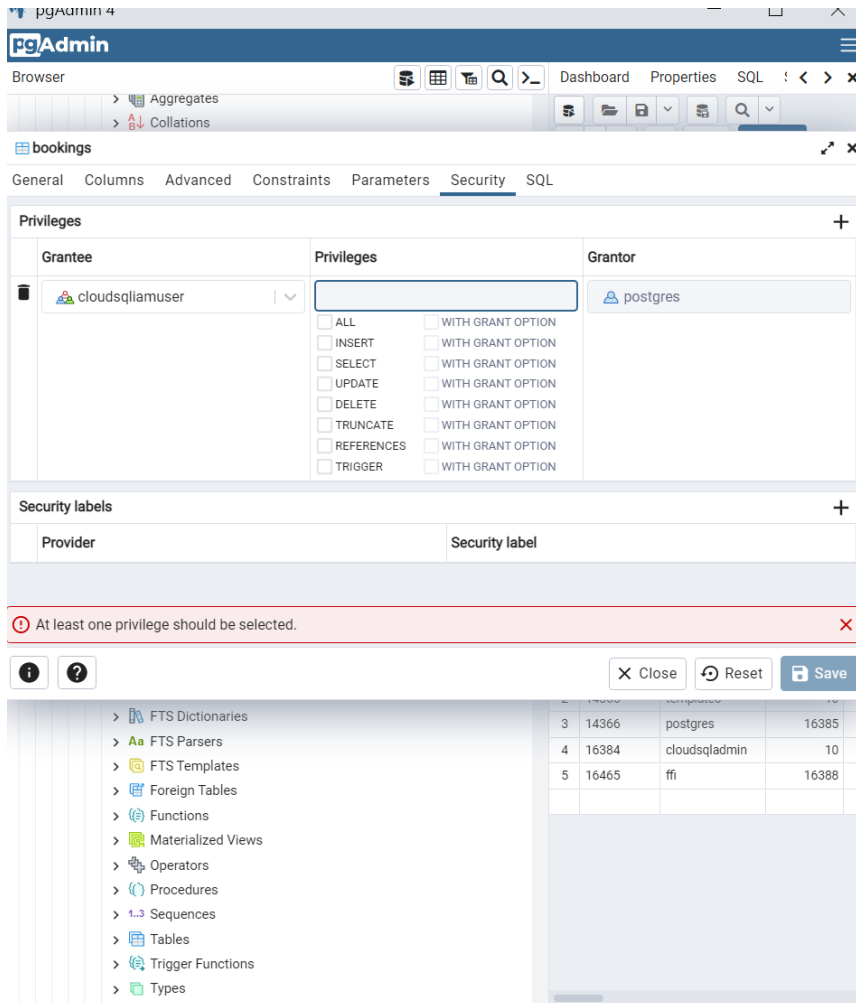
- `peer` means it will trust the identity (authenticity) of UNIX user. So not asking for a password.
- `md5` means it will always ask for a password, and validate it after hashing with `MD5`.
- `trust` means it will never ask for a password, and always trust any connection.

In PostgreSQL gibt es keine strikte Trennung zwischen Rollen, Benutzern und Gruppen. Stattdessen verwendet PostgreSQL das Konzept der "Rollen" (Roles), um Berechtigungen und Zugriff auf die Datenbank zu verwalten. Rollen können als Benutzer, Gruppen oder sogar als beides fungieren, abhängig von den ihnen zugewiesenen Berechtigungen und der Art, wie sie verwendet werden.

- Wenn nichts spezifiziert wird, ist der User Owner, der die Datenbank anlegt
- Owner kann optional spezifiziert werden

```
CREATE DATABASE datenbankname;  
CREATE DATABASE datenbankname OWNER role;
```

Weitere Einschränkungen sind auf Schema und Tabellen-Level möglich



```
GRANT { { SELECT | INSERT | UPDATE | DELETE | TRUNCATE | REFERENCES | TRIGGER }  
[, ...] | ALL [ PRIVILEGES ] }  
ON { [ TABLE ] table_name [, ...]  
      | ALL TABLES IN SCHEMA schema_name [, ...] }  
TO { [ GROUP ] role_name | PUBLIC } [, ...] [ WITH GRANT OPTION ]
```

“with grant option” heißt dass der Berechtigte seine Rechte wieder weitergeben kann.

- PostgreSQL hat seine eigene Liste von Nutzern (Datenbank-Nutzer haben zunächst nichts mit Betriebssystem-Nutzern zu tun)
- DB-Nutzer können sich eventuell auf dem Server-Rechner gar nicht einloggen (kein Betriebssystem-Account), aber schon mit dem DB-Server verbinden
- Ob ein Password oder eine andere Authentifizierung verlangt ist, steht in der Datei `pg_hba.conf`. “hba”: “host based authentication”. Steht im “data” Verzeichnis
- Falls da für lokale Logins “trust” eingetragen ist, wird kein Password verlangt (egal für welchen Nutzer)
- Interne Userverwaltung:

```
SELECT * FROM pg_catalog.pg_authid;
```

CSV Import

```
CREATE TABLE personen (
  id INTEGER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  first_name VARCHAR(50),
  last_name VARCHAR(50),
  dob DATE,
  email VARCHAR(255)
);
```

```
COPY personen(first_name, last_name, dob, email)
FROM 'C:\personen.csv'
DELIMITER ','
csv header encoding 'UTF8';
```