

3. Installation and Setup

Introduction to PostgreSQL



PostgreSQL

AGENDA

- OS Level Installation
- Configuring PostgreSQL
- Users and Schema
- Character Sets



BINARY INSTALLERS

- Available for the operating systems where PostgreSQL is used
- Recommended installation method
- Current version is 16
- Beta releases of the next version are also available
 - Betas are not recommended for production
 - Made available for testing purposes

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

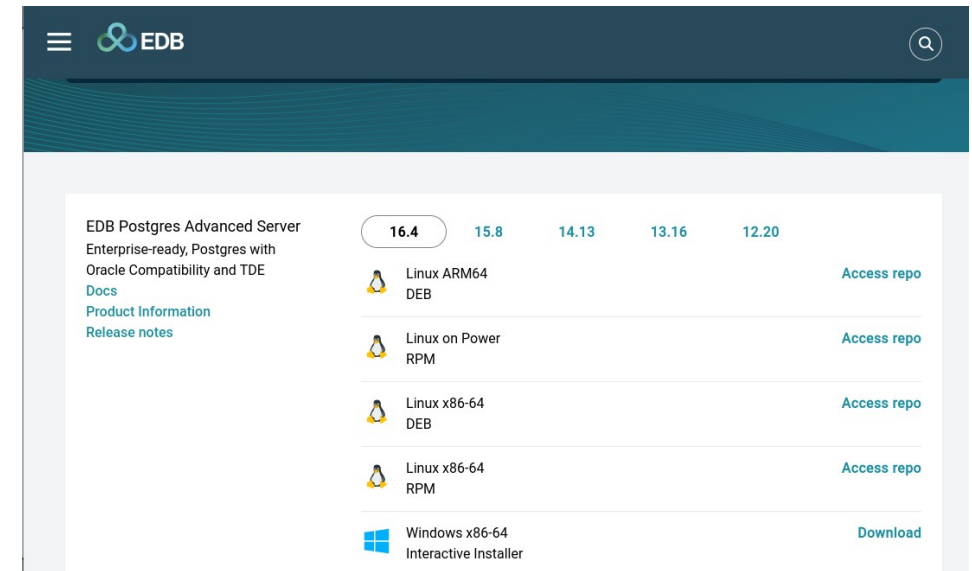
Packages and Installers

Select your operating system family:



OTHER BINARY INSTALLERS

- PostgreSQL is offered by third parties in distribution packages
 - These incorporate added features, installers, or management tools to make it easier to deploy and manage PostgreSQL.
- For example EnterpriseDB (EDB) Postgres Advanced Server which adds:
 - Oracle compatibility, advanced security, and management tools.
 - A graphical installer for easier installation
- Others are referenced in the notes




INSTALLING FROM SOURCE

- The source code is available for all previous versions
 - Available at the PostgreSQL ftp site
 - <https://www.postgresql.org/ftp/>
- In most situations, installing using binaries is the optimal solution in terms of the minimizing time and effort required for installation
- We will not cover compiling from source in this course
 - It is covered extensively in the PostgreSQL documentation

[Top](#) → [source](#) → [v11.12](#)

Directories

 [\[Parent Directory\]](#)

Files

 postgresql-11.12.tar.bz2	May 10, 2021, 8:57 p.m.	19.1 MB
 postgresql-11.12.tar.bz2.md5	May 10, 2021, 8:57 p.m.	59 bytes
 postgresql-11.12.tar.bz2.sha256	May 10, 2021, 8:57 p.m.	91 bytes
 postgresql-11.12.tar.gz	May 10, 2021, 8:57 p.m.	25.1 MB
 postgresql-11.12.tar.gz.md5	May 10, 2021, 8:57 p.m.	58 bytes
 postgresql-11.12.tar.gz.sha256	May 10, 2021, 8:57 p.m.	90 bytes

SOURCE INSTALLATION USE CASES

- Customization and Control

- Full control over the build configuration including enabling or disabling specific features, extensions, or optimizations that may not be included in standard packages.
- Allows custom compilation options, such as optimization flags, to configure the installation to specific hardware or workload requirements.

- Latest Features and Patches

- Can include the most recent features, improvements, and bug fixes before they are available in prepackaged binaries.
- Useful for installing critical fixes not yet included in packaged versions.

- Performance Optimization

- Optimal compilation for specific hardware, such as compiler flags that optimize for a specific CPU architecture.
- Allows linking with specific versions of libraries that may offer better performance or system compatibility.

SOURCE INSTALLATION USE CASES

- Custom Extensions
 - Allows inclusion of the third party or in house developed extensions that are not included in the binary package.
- Compatibility with Specific Environments
 - When deploying onto older operating systems or unique hardware, precompiled binaries may not work correctly.
 - Allows deployment onto customized or legacy operating systems.
- Security and Compliance
 - Allows for fine-grained control over the libraries and dependencies used
 - Important for compliance with specific security standards or regulatory requirements
 - Allows for auditing of the codebase and application of security patches as needed

SOURCE INSTALLATION USE CASES

- Minimal Footprint
 - Create a minimal build that with a subset of components and features reduces the footprint of the installation.
 - Useful in embedded systems or when working with limited resources.

LAB 3-1

- The lab description and documentation is in the Lab directory in the class repository



DATABASE CLUSTER

- A database cluster is a collection of databases that are managed by a single PostgreSQL instance.
 - The utility program `initdb` is used to create a new PostgreSQL database cluster.

```
ubuntu@ip-172-31-29-103:/$ /usr/lib/postgresql/16/bin/initdb -D /home/ubuntu/temp
The files belonging to this database system will be owned by user "ubuntu".
This user must also own the server process.

The database cluster will be initialized with locale "C.UTF-8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

creating directory /home/ubuntu/temp ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

initdb: warning: enabling "trust" authentication for local connections
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local
Success. You can now start the database server using:

    /usr/lib/postgresql/16/bin/pg_ctl -D /home/ubuntu/temp -l logfile start
```

DATABASE CLUSTER

- Common default directories used by installers
 - Linux : /var/lib/pgsql/data
 - Windows : C:\Program Files\PostgreSQL\16\data
- The default directory structure used in your Windows VM is shown in the screenshot

is PC > Local Disk (C:) > Program Files > PostgreSQL > 16 > data >

Name	Date modified	Type
base	9/2/2024 4:03 PM	File folder
global	9/3/2024 1:19 PM	File folder
log	9/3/2024 1:17 PM	File folder
pg_commit_ts	8/27/2024 6:25 PM	File folder
pg_dynshmem	8/27/2024 6:25 PM	File folder
pg_logical	9/3/2024 1:22 PM	File folder
pg_multixact	8/27/2024 6:25 PM	File folder
pg_notify	8/27/2024 6:25 PM	File folder
pg_replslot	8/27/2024 6:25 PM	File folder
pg_serial	8/27/2024 6:25 PM	File folder
pg_snapshots	8/27/2024 6:25 PM	File folder
pg_stat	9/3/2024 1:16 PM	File folder
pg_stat_tmp	8/27/2024 6:25 PM	File folder
pg_subtrans	8/27/2024 6:25 PM	File folder
pg_tblspc	8/27/2024 6:25 PM	File folder
pg_twophase	8/27/2024 6:25 PM	File folder
pg_wal	9/2/2024 4:06 PM	File folder
pg_xact	8/27/2024 6:25 PM	File folder
current_logfiles	9/3/2024 1:17 PM	File
pg_hba.conf	8/27/2024 6:25 PM	CONF File
pg_ident.conf	8/27/2024 6:25 PM	CONF File
PG_VERSION	8/27/2024 6:25 PM	File
postgresql.auto.conf	8/27/2024 6:25 PM	CONF File
postgresql.conf	8/27/2024 6:26 PM	CONF File
postmaster.opts	9/3/2024 1:17 PM	OPTS File
postmaster.pid	9/3/2024 1:17 PM	PID File

DATABASE CLUSTER

- The screenshot below shows the directory structure of the temp cluster created in an earlier slide

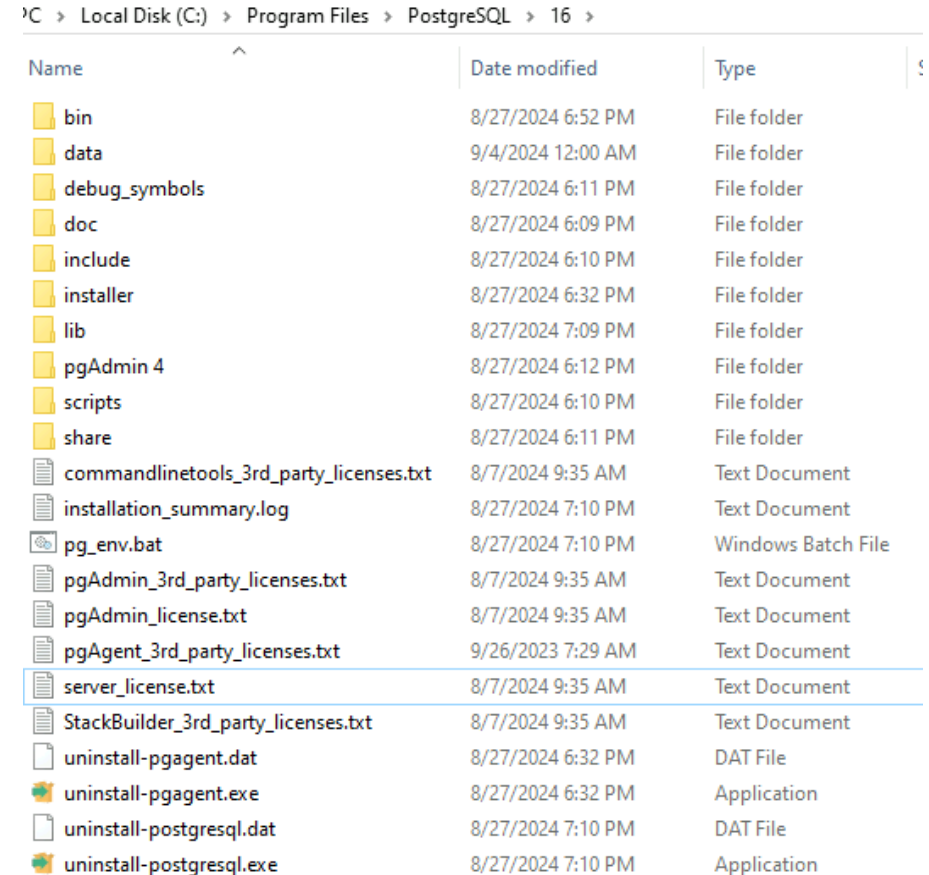
```
ubuntu@ip-172-31-29-103:~/temp$ ls -l
total 120
-rw----- 1 ubuntu ubuntu    3 Sep  3 19:37 PG_VERSION
drwx----- 5 ubuntu ubuntu 4096 Sep  3 19:37 base
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 global
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_commit_ts
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_dynshmem
-rw----- 1 ubuntu ubuntu 5711 Sep  3 19:37 pg_hba.conf
-rw----- 1 ubuntu ubuntu 2640 Sep  3 19:37 pg_ident.conf
drwx----- 4 ubuntu ubuntu 4096 Sep  3 19:37 pg_logical
drwx----- 4 ubuntu ubuntu 4096 Sep  3 19:37 pg_multixact
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_notify
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_replslot
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_serial
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_snapshots
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_stat
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_stat_tmp
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_subtrans
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_tblspc
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_twophase
drwx----- 3 ubuntu ubuntu 4096 Sep  3 19:37 pg_wal
drwx----- 2 ubuntu ubuntu 4096 Sep  3 19:37 pg_xact
-rw----- 1 ubuntu ubuntu   88 Sep  3 19:37 postgresql.auto.conf
-rw----- 1 ubuntu ubuntu 29799 Sep  3 19:37 postgresql.conf
```

DATABASE CLUSTER

- We can create multiple clusters running on a single host
- Each cluster is managed by a running instance of PostgreSQL
- Each instance
 - Listens on a different port
 - Maintains its own set of utility processes
 - Uses its own set of configuration files
- For example, to start the data cluster in the temp directory created earlier with a PostgreSQL instance listening on port 5444, this command would be used
 - `postgres -D /home/ubuntu/temp -p 5444`

INSTALLATION DIRECTORY

- The installation directory may be in different places in different installations
- In your Windows VM it looks like the screenshot
- Most of the directories are self explanatory
- The share directory contains sample configuration files
 - These are useful when we have corrupt configuration files
 - The sample files can be used to rebuild the configuration files



Name	Date modified	Type
bin	8/27/2024 6:52 PM	File folder
data	9/4/2024 12:00 AM	File folder
debug_symbols	8/27/2024 6:11 PM	File folder
doc	8/27/2024 6:09 PM	File folder
include	8/27/2024 6:10 PM	File folder
installer	8/27/2024 6:32 PM	File folder
lib	8/27/2024 7:09 PM	File folder
pgAdmin 4	8/27/2024 6:12 PM	File folder
scripts	8/27/2024 6:10 PM	File folder
share	8/27/2024 6:11 PM	File folder
commandlinetools_3rd_party_licenses.txt	8/7/2024 9:35 AM	Text Document
installation_summary.log	8/27/2024 7:10 PM	Text Document
pg_env.bat	8/27/2024 7:10 PM	Windows Batch File
pgAdmin_3rd_party_licenses.txt	8/7/2024 9:35 AM	Text Document
pgAdmin_license.txt	8/7/2024 9:35 AM	Text Document
pgAgent_3rd_party_licenses.txt	9/26/2023 7:29 AM	Text Document
server_license.txt	8/7/2024 9:35 AM	Text Document
StackBuilder_3rd_party_licenses.txt	8/7/2024 9:35 AM	Text Document
uninstall-pgagent.dat	8/27/2024 6:32 PM	DAT File
uninstall-pgagent.exe	8/27/2024 6:32 PM	Application
uninstall-postgresql.dat	8/27/2024 7:10 PM	DAT File
uninstall-postgresql.exe	8/27/2024 7:10 PM	Application

DATA DIRECTORY

- The data directory contains the working files for the database cluster
 - This usually contains configuration files
 - But these can be in a separate configuration directory
 - *base* contains all the users' data, including databases, tables, and other objects
 - *global* contains cluster-wide objects
 - *pg_wal* contains the WAL files
 - *pg_stat* and *pg_stat_tmp* are storage of permanent and temporary statistical information about the status and health of the cluster.

Local Disk (C:) > Program Files > PostgreSQL > 16 > data

Name	Date modified	Type
base	9/2/2024 4:03 PM	File folder
global	9/3/2024 1:19 PM	File folder
log	9/4/2024 12:00 AM	File folder
pg_commit_ts	8/27/2024 6:25 PM	File folder
pg_dynshmem	8/27/2024 6:25 PM	File folder
pg_logical	9/3/2024 1:22 PM	File folder
pg_multixact	8/27/2024 6:25 PM	File folder
pg_notify	8/27/2024 6:25 PM	File folder
pg_replslot	8/27/2024 6:25 PM	File folder
pg_serial	8/27/2024 6:25 PM	File folder
pg_snapshots	8/27/2024 6:25 PM	File folder
pg_stat	9/3/2024 1:17 PM	File folder
pg_stat_tmp	8/27/2024 6:25 PM	File folder
pg_subtrans	8/27/2024 6:25 PM	File folder
pg_tblspc	8/27/2024 6:25 PM	File folder
pg_twophase	8/27/2024 6:25 PM	File folder
pg_wal	9/2/2024 4:06 PM	File folder
pg_xact	8/27/2024 6:25 PM	File folder
current_logfiles	9/4/2024 12:00 AM	File
pg_hba.conf	8/27/2024 6:25 PM	CONF File
pg_ident.conf	8/27/2024 6:25 PM	CONF File
PG_VERSION	8/27/2024 6:25 PM	File
postgresql.auto.conf	8/27/2024 6:25 PM	CONF File
postgresql.conf	8/27/2024 6:26 PM	CONF File
postmaster.opts	9/3/2024 1:17 PM	OPTS File
postmaster.pid	9/3/2024 1:17 PM	PID File

SYSTEM CATALOG

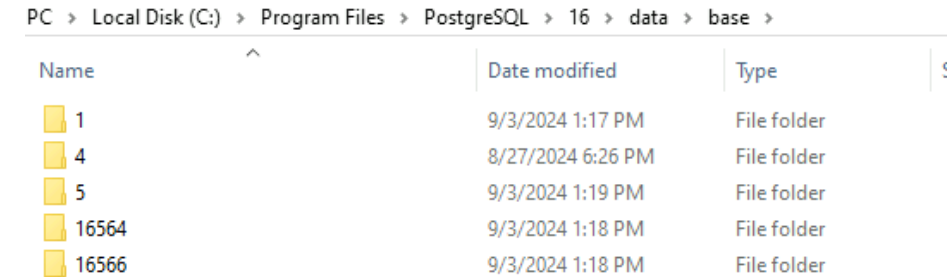
- The system catalog is in the *postgres* directory in the *pg_catalog* schema
- Depending on their use, these are either in the global or base directories
 - The distinction is not crucial unless table spaces are employed
- A list of all the catalog entries is provided in the documentation

```
postgres=# \dt pg_catalog.*
```

List of relations			
Schema	Name	Type	Owner
pg_catalog	pg_aggregate	table	postgres
pg_catalog	pg_am	table	postgres
pg_catalog	pg_amop	table	postgres
pg_catalog	pg_amproc	table	postgres
pg_catalog	pg_attrdef	table	postgres
pg_catalog	pg_attribute	table	postgres
pg_catalog	pg_auth_members	table	postgres
pg_catalog	pg_authid	table	postgres
pg_catalog	pg_cast	table	postgres
pg_catalog	pg_class	table	postgres
pg_catalog	pg_collation	table	postgres
pg_catalog	pg_constraint	table	postgres
pg_catalog	pg_conversion	table	postgres
pg_catalog	pg_database	table	postgres
pg_catalog	pg_db_role_setting	table	postgres
pg_catalog	pg_default_acl	table	postgres

BASE DIRECTORY

- The base directory holds our data.
- Every database object, like a table or index, is assigned a unique `Object Identifier` (`OID`).
 - Unique 32-bit integer assigned when a database object is created and remains constant while the object exists.
 - Used internally to reference database objects in system catalogs.
 - Used to name the physical data files on disk that correspond with a data object. Each table and index is stored in a file named after its OID.
 - If a table has a lot of data, it is split into multiple segment file named using the OID followed by a segment number (e.g., 12345.1, 12345.2, etc.).



PC > Local Disk (C:) > Program Files > PostgreSQL > 16 > data > base >		
Name	Date modified	Type
1	9/3/2024 1:17 PM	File folder
4	8/27/2024 6:26 PM	File folder
5	9/3/2024 1:19 PM	File folder
16564	9/3/2024 1:18 PM	File folder
16566	9/3/2024 1:18 PM	File folder

CONFIGURATION FILES

- PostgreSQL.conf file contains the parameters to configure the server
 - Initdb installs a default copy of postgresql.conf and is usually located in data directory.
 - A different configuration file can be specified when starting the server
 - Some parameter changes require a restart of the server
- The main categories of configuration parameters are
- Connection and Authentication
 - Configures how clients connect to the database and how authentication is handled.
 - *listen_addresses* (IP addresses on which the server listens)
 - *port* (server port)
 - *max_connections* (maximum number of concurrent connections)
 - *authentication_timeout* (timeout for authentication)
 - *ssl* (SSL/TLS support)

CONFIGURATION FILES

- Resource Usage:
 - Controls memory, disk, and CPU usage for optimizing resource allocation.
 - *shared_buffers* (memory allocated for shared data)
 - *work_mem* (memory for sort operations and hash tables)
 - *maintenance_work_mem* (memory for maintenance tasks like vacuuming)
 - *effective_cache_size* (estimate of available memory for disk caching)
- Logging
 - Configures logging options to monitor server activity, errors, and performance.
 - *log_destination* (log output destinations like stderr, csvlog, etc.)
 - *logging_collector* (enables collecting logs into files)
 - *log_min_duration_statement* (logs queries running longer than specified time)
 - *log_line_prefix* (prefix format for log lines)

CONFIGURATION FILES

- Security and Authentication:
 - Manages security settings, including user access, encryption, and authentication.
 - *password_encryption* (method for encrypting passwords)
 - *ssl_cert_file* (path to SSL certificate file)
 - *ssl_key_file* (path to SSL key file)
- Client Connection Defaults:
 - Sets default behavior for client connections.
 - *statement_timeout* (timeout for queries)
 - *idle_in_transaction_session_timeout* (timeout for idle transactions)
 - *search_path* (default schema search path)
- Query Tuning:
 - Provides options for optimizing query execution.
 - *enable_seqscan* (enables sequential scans)
 - *enable_indexscan* (enables index scans)
 - *effective_cache_size* (expected available cache memory)

CONFIGURATION FILES

- Locale and Formatting:
 - Manages settings for locale-specific formatting of data.
 - *lc_messages* (locale for messages)
 - *lc_monetary* (locale for monetary formatting)
 - *datestyle* (format for date and time output)
- Replication
 - Configures replication settings for high availability and failover.
 - *wal_level* (amount of WAL information written)
 - *max_wal_senders* (number of concurrent replication connections)
 - *synchronous_standby_names* (list of standby servers for synchronous replication)
- Archiving and WAL (Write-Ahead Logging):
 - Manages WAL settings, including archiving and retention.
 - *archive_mode* (enables archiving of WAL files)
 - *archive_command* (command to archive WAL files)
 - *wal_keep_size* (amount of WAL files to retain)

CONFIGURATION FILES

- Autovacuum:
 - Configures the autovacuum process for automatic maintenance
 - *autovacuum* (enables autovacuum)
 - *autovacuum_naptime* (frequency of autovacuum runs)
 - *autovacuum_max_workers* (maximum number of autovacuum workers)
- Background Writer:
 - Manages the background writer process that helps maintain data consistency.
 - *bgwriter_delay* (interval between background writer checks)
 - *bgwriter_lru_maxpages* (maximum pages written by the background writer)
- Checkpoint Settings:
 - Configures checkpoint behavior to balance write performance and data safety.
 - *checkpoint_timeout* (maximum time between checkpoints)
 - *checkpoint_completion_target* (target duration for checkpoints)

CONFIGURATION FILES

- File Locations:
 - Specifies paths for important files and directories.
 - *data_directory* (location of the data directory)
 - *hba_file* (path to the *pg_hba.conf* file)
 - *ident_file* (path to the *pg_ident.conf* file)
- Miscellaneous:
 - Includes various other settings that do not fit neatly into the above categories
 - *timezone* (default time zone for the server)
 - *unix_socket_directories* (locations for Unix domain sockets)

DYNAMIC CONFIGURATIONS

- The *postgresql.auto.conf* file is used to store configuration settings that are modified dynamically using the SQL command ALTER SYSTEM.
 - This file is automatically updated by ALTER SYSTEM SET commands
 - Parameters that require a restart should not be altered this way
 - Overrides the corresponding settings in the main postgresql.conf file
 - Changes made via ALTER SYSTEM are persistent across server restarts
 - After making changes with ALTER SYSTEM, reload the configuration using SELECT pg_reload_conf(); for the changes to take effect.
 - Syntax : ALTER SYSTEM SET configuration_parameter = 'value'
 - The altered settings can be cleared using RESET
 - Syntax to reset : ALTER SYSTEM RESET configuration_parameter;
 - Syntax to reset all : ALTER SYSTEM RESET ALL;

LAB 3-2

- The lab description and documentation is in the Lab directory in the class repository



USERS

- There are two basic types of users
 - Operating users – defined by the operating system, like Administrator in our Windows VM
 - DB users, defined in the PostgreSQL cluster
- DB users are referred to as roles
- A list of all the roles defined in the VM are shown to the right
 - Roles starting with pg_ are system roles
 - \du shows the non-system roles
- *postgres* is the superuser role defined when PostgreSQL is installed

```
postgres=# SELECT rolname from pg_roles;
          rolname
-----
pg_database_owner
pg_read_all_data
pg_write_all_data
pg_monitor
pg_read_all_settings
pg_read_all_stats
pg_stat_scan_tables
pg_read_server_files
pg_write_server_files
pg_execute_server_program
pg_signal_backend
pg_checkpoint
pg_use_reserved_connections
pg_create_subscription
postgres
student
(16 rows)
```

USERS vs GROUPS

- The PostgreSQL user management is modeled after the UNIX system of users and groups.
 - This distinction was enforced early but has been dropped.
 - Both users and groups are roles
 - Users are roles that have the ability to log into PostgreSQL
 - A role can be a single account, a group of accounts, or both
 - *It is considered to be a best practice to have a role not function as both, even though it can be done*
- A user group is often used to create a package of permissions
 - Individual users can then be assigned to these groups to inherit the permissions
 - For example, to give a team members permission to work on a specific database

CREATING ROLES

- In the example below, a user role bob is created and dropped

```
postgres=# CREATE ROLE bob;
CREATE ROLE
postgres=# \du

                                List of roles
Role name |                               Attributes
-----+-----
bob       | Cannot login
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
student   | Superuser, Create role, Create DB

postgres=# DROP ROLE bob;
DROP ROLE
postgres=# \du

                                List of roles
Role name |                               Attributes
-----+-----
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
student   | Superuser, Create role, Create DB
```

CREATING ROLES

- If 'bob' is created with *CREATE ROLE bob LOGIN PASSWORD 'bob';*
 - Then bob can login and locate the the database 'test'
 - Depending on permission, bob may not be able to do anything in the database

```
C:\Users\Administrator>psql -U bob -d test
Password for user bob:
psql (16.4)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

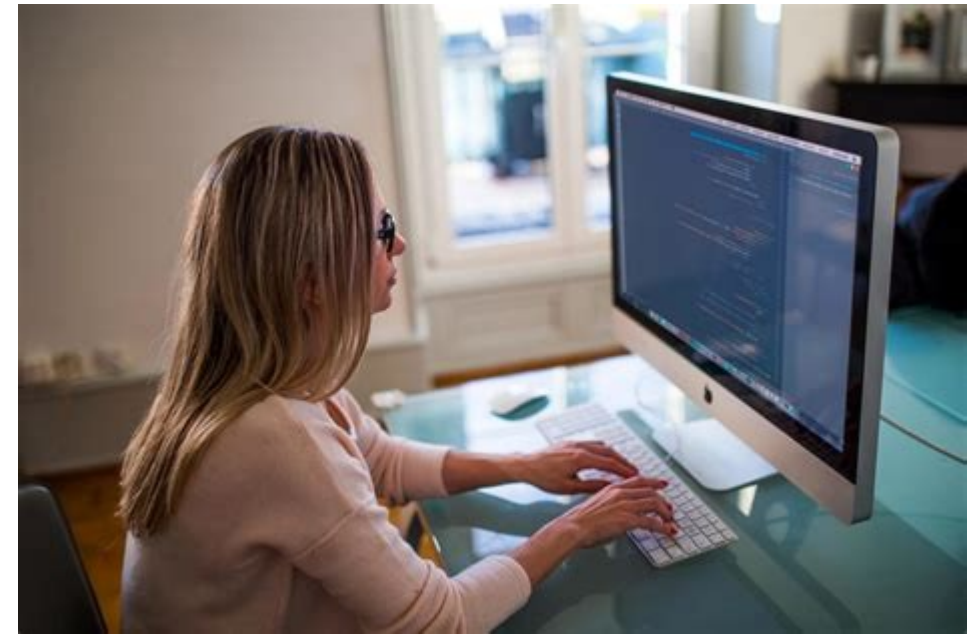
test=>
```

SUPERUSER

- The superuser role has unrestricted access to the entire database cluster
- Has the following capabilities
 - Full control over all databases and objects
 - Bypass all permission checks
 - Alter system configuration
 - Execute unrestricted file system access commands
 - And more
- Generally a superuser role is created for the cluster admin so that the postgres default account isn't used for administration tasks

PRIVILEGES

- In order to perform specific actions on a database
 - Roles have to be granted privileges
 - These can be fine grained or “all”
- More general privileges can be granted
 - Like CREATEDB to create databases
 - And CREATEROLE to create other users
- Privileges are either:
 - Cluster based: granted by superuser
 - Object based: allow operations on a specific database object, granted by the owner of that object.
 - *GRANT CONNECTION ON DATABASE test TO bob*



SCHEMA

- Schema is a name space that contains database objects
 - One database can have multiple schemas.
 - Schema partition the database objects between different applications.
 - Allows organizing database objects into logical groups.
- Allows a single database can be used by different users in different contexts without interference.
- The Schema names are used as prefixes to the database objects
 - There is a default schema for each database called the public schema

SCHEMA OPERATIONS

- DROP
 - Schema can be dropped along with any object it contains if the CASCADE option is used
- ALTER can be used to
 - Rename a schema
 - Change a schema's owner
 - To move objects between schema
- GRANT/REVOKE
 - Used to set and remove privileges for a schema
- Search path
 - When looking for a name, the schema are searched in a search order
 - The SET command can be used to alter the search order

LAB 3-3

- The lab description and documentation is in the Lab directory in the class repository



CHARACTER SETS

- Character sets, or encoding, define the set of symbols and the way those symbols are represented internally in the database.
- There are a several standard encodings in use.
- UTF-8:
 - Most used character set, a variable width unicode encoding
 - Default encoding in PostgreSQL
- LATIN1 (ISO 8859-1):
 - A single-byte encoding for Western European languages.
 - Suitable for older systems but limited to a small set of languages.

CHARACTER SETS

- Setting the character set for a database
 - `CREATE DATABASE mydb WITH ENCODING 'UTF8';`
- Client and server encoding
 - The client and server may also use different character encoding
 - Conversions are handled automatically

CHARACTER SETS

- Setting the character set for a database
 - CREATE DATABASE mydb WITH ENCODING 'UTF8';
- Client and server encoding
 - The client and server may also use different character encoding
 - Conversions are handled automatically
- The two encoding from the Windows and Ubuntu host are shown on the right

```
student=# SHOW client_encoding;  
client_encoding  
-----  
WIN1252  
(1 row)
```

```
student=> SHOW client_encoding;  
client_encoding  
-----  
UTF8  
(1 row)
```

LAB 3-3

- The lab description and documentation is in the Lab directory in the class repository



End Module



PostgreSQL