Cheatography

PostgreSQL 101 for DBAs/sysadmins Cheat Sheet

by armk via cheatography.com/215747/cs/47026/

Grab info from existing cluster

```
netstat -nap | grep postgres # To grab listening port if not default
ps uf -C postgres # List all running instances w/ process owner
ps f -U instan ce_ owner # With user found above, to see startup args
psql -c SHOW data_d ire ctory; # To get the main cluster directory aka data directory
psql -c SHOW config _file; # To get the config file path, useful if nondefault
su - db_owner pg_con tro ldata data_d ire ctory # To get cluster details
```

On redhat: all config files in `/var/l ib/ pgsql/<ve rsi on>/data` by default

On debian: `postgr es.conf` in `/etc` by default

pg_settings view structure

name	Setting name
value	Current value
context	Context of the setting (what to restart to update the value)
source/sourcefile/sourceline	Where the setting is defined
setting/unit	Value of the setting
boot_val	Value at instance startup
reset_val	Default value
pending_restart	Value has pending modifications

Contains a detailed view current running config, refers to the same settings as psql's SHOW/SET.

Instance files on disk

One directory per instance containing binaries, config and default datafile location ("the *cluster*"). All files owned by db service account with at least chmod u+rw (+x on dirs).
Datafiles are stored into one subdir per DB.
The pg_class table contains a map of DB objects to file paths.
Default cluster dir depends on package and distro: redhat is /var/lib/-pgsql/<version>
Every "user" object and datafile is located in the `pg_default` tablespace by default: hardcoded to `base` subdirectory of the cluster directory. All objects (including entire DBs) can be moved to tables-paces located in any accessible path. Once declared, a "tablespace folder" is an integral, non-optional part of the cluster.

Tools

initdb	Create file/dir structure (aka cluster) for instance
createdb, dropdb	Create/delete a DB
pg_ctl	Control instance state
pg_controldata	View config
pg_isready	Check if instance is up w/o opening a full connection
pg_resetwal	**DANGEROUS** Wipe transaction logs

By default in /usr/pgsql-*<version>*/bin (on redhat).

Performance tracking

Use extension pg_stat_statements to track execution times.
⚠☐Requires restart to install⚠☐
Note parameters **pg_stat_statements.track** and **pg_stat_stateme-nts.track_planning** (<- costly)
Query **pg_stat_statements** to view results.
General slowness issues are often caused by autovacuum and/or checkpoint configuration.

Users and permissions

Postgres does RBAC by default.
No distinction between user and group: they are both **roles**.
Colloquially, group = role with no LOGIN option.
Permissions are GRANTed/DENYed to roles.
Roles can be granted to other roles.
Roles can impersonate any role granted to them: an user can act as any of the groups he's a part of with SET ROLE.

See 21. Database Roles

GRANT/REVOKE: object permissions

Everyone has CONNECT and TEMP permissions on DBs through `public` group.
DROP and ALTER belong to the owner role only.
Change default permissions with:
ALTER DEFAULT PRIVILEGES
FOR ROLE *object_creator*
permission_stmt

See psql section to view permissions.

Access checklist

Can connect according to pg_hba
Has LOGIN permission
Has CONNECT on DB
Has USAGE on schema/namespace
Has <operation> on table/column of queryable

Create new instance

```
initdb \
 -A auth_m ethod \
 -D data_d ire ctory \
 -E encoding \
 -X tlog_d ire ctory \
  --l oca le= locale \
 [-k]
```

Create new instance (cont)

```
> # Start from service
systemctl enable postgres
systemctl start postgres
# Start standalone
/usr/pgsql-16/bin/pg_ctl -D data_directory start
# Connect as postgres w/o passwd
sudo -u postgres psql -w
```

Check instance state with "systemctl postgres-16 status" or pg_isready

pg_hba: authentication management

TYPE	local | host
DATABASE	all | replication | *db_name*
USER	`role` for specific role, `+role` for group
ADDRESS	Origin address
METHOD	Accepted auth method, see man page

Space-separated file, one rule per line with above fields.
Configuration reload required to apply.
See 20. Client Authentication

Tablespace/DB size management

PostgreSQL has **no size limit mechanism** and will only stop growing datafiles when the OS stops it (typically on full filesystem). Size quotas must be **enforced at the file level**.
Instance-level quotas are enforced by placing the cluster on its dedicated filesystem or placing quotas on the cluster directory at the OS/FS level (eg. XFS quotas).
Database or **object-level** quotas can be done by housing the DB/object in a tablespace located in dedicated filesystems/under different quota rules.

Low level/manual backup/restore flow

Physical backup: run pg_backup_start, copy data files to backup storage, call pg_backup_stop and save its output to backup storage.
PITR restore: Copy backed up files in cluster "restore" directory, edit postgresql.conf with restore_command to fetch WAL files, set recovery_* options with point in time and end-of-restore (recovery_target_action) options, create restore/recovery.signal file, start instance.

Backup & restore tools

pg_dump	Logical dumps of single DB
pg_dumpall	Logical dumps of entire instance
pg_restore	Apply logical dumps for restoration
archiver	Archive transaction logs, see archive_command in postgresql.conf
pg_backup_start()	Prepares DB for physical backup
pg_backup_stop()	Ends physical backup process and returns missing data necessary for backed up cluster to be consistent
pg_basebackup	Physical backup automation tool, does pg_backup_* calls and file copies automatically Only tool available on Windows
pg_verifybackup	Check "plain" type backup integrity
pg_receivewal	Transparent, pull-style backup and WAL archiver tool
pgBackRest	Backup/restore utility
pgBarman	Backup/restore utility (prefer pgBackRest)

No diff backups natively before version 17
See 25. Backup and Restore

psql commands

Connect from system shell	psql -h *hostname* -p *port* -U *role* -d *DB*
Connect from psql shell	\c *db user host port*
Execute sql script	\i *file*
Execute shell command	\! *command*
Edit psql options	\set [*variable*=*value*]
Execute command every 3 seconds 2 times	\watch i=2 c=3
Get help on SQL command	\h *command*

psql commands (cont)

Edit command in external editor and execute	\e
Make wide tables readable	\x on|auto
Profile script	Located in ~/.psqlrc
Re-run last command	\g (\gx to output as \x on)

Can supply connection string instead of connection arguments

List DB objects with psql

\l	**Databases**
\dt	**Tables**
\dn	**Namespaces (aka schemas)**
\d *name*	Describe queryable or index
\d	List everything
\di	Indexes
\ds	Sequences
\dp	Permissions
\du	Roles (users/groups)
\dv	Views
\dx	Extensions
\dn	Namespaces (schemas)

See psql manual
All commands can take a filter pattern as argument.
See "search path" above.

Configuration files

Main config file is **postgresql.conf** in the main cluster directory (or specified explicitly as startup argument).
Settings changed dynamically (via ALTER SYSTEM) are stored into postgresql.auto.conf which is loaded last and has priority on postgresql.conf. **Do not modify by hand.**
Settings changes (incl. ALTER SYSTEM) are often not applied immediately: see the context column of pg_settings for how/when setting changes are applied and the pending_restart column.
See PostgreSQL settings/config quick reference for more info on individual settings.

Search path

Alter for this session	SET search_path TO *my_db*;
Persist for this DB	ALTER DATABASE SET search_path to ... ; ALTER DATABASE SET search_path FROM CURRENT;
Persist for this instance	postgresql.conf

"List" psql commands (`\d`) only show what is in the search path by default.

Processes & transaction lifecycle

Backend processes transactions by loading cache pages in memory from datafiles and updating them. One per user session

WAL writer watches WAL buffers and flushes them to disk periodically

BG writer watches for individual dirty pages in shared memory and writes them to datafiles.

Checkpointer periodically uses WAL to flush all shared memory written before a checkpoint (automatic or user-requested) to disk.

Autovacuum periodically reclaims invalidated cache pages.

WAL writer, checkpointer and autovacuum work on a sleep-wake schedule, BG writer works continuously, backends work during user transactions.

External utilities

pg_activity	`top`-like monitor
dbeaver	GUI tool for DDL visualization
pgadmin	Web-based tool (slow)
pgloader	Data Migration tool from other psql instance, CSV or other DBMS
pgHero	Performance dashboard for Postgres
pgTune	Performance configuration tuning tool
pgBadger	Web-based monitoring tool (eq. OEM reports)

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

PostgreSQL 101 for DBAs/sysadmins Cheat Sheet

by armk via cheatography.com/215747/cs/47026/

Data files

no extension	Tables, indexes
.TOAST	Oversized object storage
.FSM	Free space map
.VM	Visibility map

Filesize is max 1 GB except TOAST files.
Files over 1 GB are split into .1, .2, .3, etc.

"Special" data files

pg_wal	Write-ahead log
pg_xact	Commit data
pg_commit_ts	Commit timestamps
pg_multixacts	States of multiple transactions
pg_serial	States of serializable transactions
pg_twophase	States of prepared transactions
pg_dynshmem	Dynamic shared memory
pg_logical	Logical replication
pg_notify	Listen/Notify states
pg_repslot	Replication slots
pg_snapshots	Exported snapshots

"Private" files, should not be edited in normal operation

Write-ahead log (WAL)

Postrgresql's transaction log, stores past transactions + transactions not yet written to datafiles.
Stored in the pg_wal subdirectory, split into 16 MB chunks making up 4 GB logical files.
WAL files are considered either "current" or "past".
Past WAL files are put into an archive queue (pg_wal/archive_status) and processed in sequence by the archiver (see 25.3.1. Setting Up WAL Archiving).
`pg_sta t_a rchiver` gives info about archiving processes.

25.3.1. Setting Up WAL Archiving - 28.5. WAL Configuration

Storage best practices

Recommended: ext4 first, xfs second
zfs possible but not well-known yet
ext4 recommended parameters: `noatime, data=w rit eback`
On Linux, consider scheduler config changes depending on hardware
RAID 10 preferable to raid 5 for controller load reasons

Better hardware badly configured usually outperforms well-conf-igured worse hardware.

Official useful plugins

pg_freespacemap	View free space maps
pg_prewarm	Preload caches from last run at boot
pg_stat_stat-ements	Track SQL execution statistics
auto_explain	Trace costly statements automatically
pgstattuple	Get table stats (live/dead rows, volume of data)

View available extensions with pg_available_extensions view or `\dx [+ *extension*]`
Install modules via postgres.conf: shared_preload_libraries (perma-nent) or LOAD statement (volatile).

pg_catalog: system views

pg_locks	Locks
pg_stat_database	DB-wide object statistics
pg_class	Object-ID mappings
pg_stat_*_tables	Table-level statistics
pg_stats	Column-level stats
pg_archiver_stats	Archiver status

Cast table IDs to table names with the ::regclass operator