PostgreSQL – A Crash Course

Joe Conway
joe.conway@crunchydata.com
mail@joeconway.com

Crunchy Data

February 03, 2018

Binary Packages

- RPMs - your distro or PGDG
- DEBs - your distro or PGDG
- Windows - third party options
- MacOS - third party options
- Solaris/*BSD - third party options
- Source?

Build Dependencies

- On Red Hat 7.x (or variant)
  - git (PostgreSQL source tree)
  - Build Dependencies (might not be all inclusive)

```
yum install git bison flex gcc ccache readline-devel \
  zlib-devel perl-devel perl-ExtUtils-Embed openssl-devel \
  pam-devel libxml2-devel libxslt-devel libuuid-devel \
  openldap-devel tcl-devel python-devel gdb
```

Clone Source

```
mkdir -p /opt/src/pgsql-git
cd /opt/src/pgsql-git/
git clone git://git.postgresql.org/git/postgresql.git
cd postgresql

# checkout the pg10 tree
git checkout REL_10_STABLE
```

http://www.postgresql.org/docs/10/static/git.html

Configure and Build

- Already have PGDG RPMs installed
- Want very similarly configured source tree
  - Capture and modify CFLAGS
    ⇒ Change -O2 -g to -O0 -g3
    ⇒ Remove -Wp,-D_FORTIFY_SOURCE=2

```
export PATH=/usr/pgsql-10/bin:$PATH
eval ./configure $(pg_config --configure | \
                   sed -e 's/-O2 -g/-O0 -g3/g' \
                       -e 's/-Wp,-D\_FORTIFY\_SOURCE=2//g')
make -j 8
make install
```

Debug Workflow

- Log in using psql
- Find backend process pid
- Attach with gdb
- Set breakpoint
- Execute SQL statement from psql
- Debug in attached gdb session

Configuration

File postgresql.conf and postgresql.auto.conf:

```
# comment
name = value
```

To activate configuration changes:

```
psql -c "SELECT pg_conf_reload()" # as postgres
pg_ctl -D $PGDATA reload          # as postgres
service postgresql reload         # typically as root
systemctl reload postgresql-10    # typically as root
kill -SIGHUP <postmaster-pid>     # as postgres or root
```

http://www.postgresql.org/docs/current/interactive/runtime-config.html

Configuration

- Persistent
  - Modify: `postgresql.conf`
  - Add: `postmaster` options (e.g. in startup script)
  - With SQL:
    - `ALTER SYSTEM SET`, `ALTER SYSTEM RESET`
    - `ALTER [DATABASE] <dbname> SET <var> = <val>;`
    - `ALTER [ROLE] <rolename> SET <var> = <val>;`
- Per Session (SQL)
  - `SET`, `RESET`, `SHOW`
  - `SELECT * FROM pg_settings;`
  - `UPDATE pg_settings SET setting = '<val>' WHERE name = '<var>';`
  - `SELECT current_setting('<var>');`
  - `SELECT set_config('<var>','<val>','<local T/F>');`

Most Commonly Tuned

```
listen_addresses = 'localhost'
max_connections = 100
shared_buffers = 128MB
work_mem = 4MB
effective_cache_size = 128MB
random_page_cost = 4.0
```

https://www.postgresql.org/docs/current/static/runtime-config.html

Logging

```
log_line_prefix = '%m %a %u %d %r %p %s %c %e: '
# %m = timestamp with milliseconds
# %a = application name
# %u = user name
# %d = database name
# %r = remote host and port
# %p = process ID
# %s = session start timestamp
# %c = session ID
# %e = SQL state
# ... and others
```

https://www.postgresql.org/docs/current/static/runtime-config.html

Host Based Authentication File

- Which hosts are allowed to connect
- How clients are authenticated
- Which PostgreSQL user names they can use
- Which databases they can access

https://www.postgresql.org/docs/10/static/auth-pg-hba-conf.html

Host Based Authentication File

- Read on server startup
- Must reload postmaster for changes to take effect
- First line matching conn type, address, database, and user is used for authentication
- If line picked and authentication fails, access denied
- If no line matches, access denied

https://www.postgresql.org/docs/10/static/auth-pg-hba-conf.html

Host Based Authentication File

Lines (rules/records) look like this

```
# CONN-TYPE  DATABASE  USER     ADDRESS     METHOD    OPTIONS
# local      <dbname>  <user>               <method>  [<opts>]
# host       <dbname>  <user>   <address>   <method>  [<opts>]
# hostssl    <dbname>  <user>   <address>   <method>  [<opts>]
# hostnossl  <dbname>  <user>   <address>   <method>  [<opts>]

# Default values on Debian-variants (PostgreSQL 10)
local         all       postgres              peer
local         all       all                   peer
host          all       all      127.0.0.1/32 md5
host          all       all      ::1/128      md5

# RHEL7 (PostgreSQL 10)
local         all       all                   peer
host          all       all      127.0.0.1/32 ident
host          all       all      ::1/128      ident
```

Connection Type

Specifies type of connection the rule matches

- `local`: Unix-domain socket
- `host`: Either plain or SSL-encrypted TCP/IP socket
- `hostssl`: SSL-encrypted TCP/IP socket
- `hostnossl`: Plain TCP/IP socket

Database

Specifies set of databases the rule matches

- `all`: Wildcard
- `sameuser`: Database name matches user name
- `samerole`: User part of role/group matching database name
- `replication`: `all` keyword does not match `replication`
- `<dbname>[,<dbname>]`: One or more specific database names
- `@<filename>`: Separate file containing names to match

User

Specifies set of users the rule matches

- `all`: Wildcard
- `<username>[,<username>]`: One or more user names
- `+<groupname>`: Any roles that are directly or indirectly members of this role
- `@<filename>`: Separate file containing names to match

Address

Specifies set of client hosts the rule matches

- `<IPAddr>/<CIDR-Mask>`: Host or Network
- `<IPAddr> <Mask>`: Host or Network
- `[.]<hostname>`: [suffix] actual FQ hostname
- `samehost`: match any of server's own IP addresses
- `samenet`: match any address in any subnet that server directly connected to

Method

Specifies authentication method to use when connection matches rule

- `trust`
- `scram-sha-256`, `md5`, `password`
- `cert`
- `peer`, `pam`, `ident`
- `gss`, `sspi`, `ldap`, `radius`
- `reject`

`https://www.postgresql.org/docs/10/static/auth-methods.html`

CRUNCHY Enterprise PostgreSQL

Options

Set of options for the authentication in the format NAME=VALUE

- Options depend authentication method
- Refer to "Client Authentication" section of docs

Brief Example

```
local all  all                  trust
host  all  all  127.0.0.1/32    trust
host  all  all  samenet         scram-sha-256
host  all  all  192.168.1.0/24  ldap ldapurl="ldap://ldap.ex.net/dc=ex,dc=net?uid?sub"
```

initdb

- Source build/do-it-yourself
  `<path-to-postgres>/bin/initdb -D $PGDATA`
- Red Hat/CentOS 7.x
  `/usr/pgsql-10/bin/postgresql10-setup initdb`
  - creates cluster and config files in `/var/lib/pgsql/10/data/`
- Debian-based
  `pg_createcluster 10 main`
  - creates cluster in `/var/lib/postgresql/10/main`
  - config files created in `/etc/postgresql/10/main`

Starting PostgreSQL

- Single user mode
  `postgres --single -D $PGDATA dbname`
- Manual
  `pg_ctl -D $PGDATA -l /path/to/logfile start`
- Red Hat/CentOS 6.x
  `service postgresql start`
- Red Hat/CentOS 7.x
  `systemctl start postgresql-10`
- Debian-based
  `pg_ctlcluster 10 main start`

Stopping PostgreSQL

- Single user mode
  `Control+D` (type EOF character)
- Manual
  `pg_ctl -D $PGDATA stop -m fast`
- Red Hat/CentOS 6.x
  `service postgresql stop`
- Red Hat/CentOS 7.x
  `systemctl stop postgresql-10`
- Debian-based
  `pg_ctlcluster 10 main stop`

Shutdown Modes

- `smart`: wait until existing sessions exit
- `fast`: gracefully terminate existing sessions (default)
- `immediate`: kill all processes

```
pg_ctl -D $PGDATA stop -m immediate
```

Terminate Particular Session

In bash terminal:

```
ps -fu postgres |grep test
postgres 30999  1837  0 16:56 ? 00:00:00 postgres: postgres test [local] idle
kill -SIGTERM 30999
```

In psql:

```
SELECT pid, state, clock_timestamp() - state_change as age, query
FROM pg_stat_activity WHERE datname = 'test';
  pid  |        state        |       age       | query
-------+---------------------+-----------------+--------
 31255 | idle in transaction | 00:00:26.020821 | begin;
(1 row)
SELECT pg_terminate_backend(31255);
```

Cancel Long Running Queries

In bash terminal:

```
ps -fu postgres |grep test
postgres 30999  1837  0 16:56 ? 00:00:00 postgres: postgres test [local] idle
kill -SIGINT 30999
```

In psql:

```
SELECT pid, state, clock_timestamp() - state_change as age, query
FROM pg_stat_activity WHERE datname = 'test';
  pid  |        state        |       age       | query
-------+---------------------+-----------------+--------
 31255 | idle in transaction | 00:00:26.020821 | begin;
(1 row)
SELECT pg_cancel_backend(31255);
```

psql - Hints

```
# Connect to named database
psql [-h <host> -p <port> -U <user>] <dbname>

# Execute the given command string
psql <dbname> -c "some sql"
psql <dbname> --command="some sql"
echo "some sql" | psql <dbname>

# Repeatedly execute command
watch -n 1 "psql <dbname> -c 'some sql'"
```

https://www.postgresql.org/docs/10/static/app-psql.html

psql - Hints

```
# List all available databases, then exit
psql -l
psql --list

# Echo the actual queries generated by \d and other backslash commands
psql -E
psql --echo-hidden
```

psql - Hints

```
<Up>        -- last line in history
<Ctrl>-r    -- search history
<tab>       -- autocomplete
```

psql - Hints

```
\?                          -- backslash cmd help
\h [<some SQL command>]     -- SQL cmd help
\dt [<schema>].[<tbl>][*]   -- table list
\d  <tbl>[*]                -- table definition
\df <func>[*]               -- function definition
\x [on|off|auto]            -- expanded output
\e [FILE] [LINE]            -- edit the query buffer or file
\watch [SEC]                -- execute query buffer every SEC seconds
```

General Notes on Syntax

- Identifier:
  - Unquoted: `name` (lower cased)
  - Quoted: `"Name"` (preserved)
  - Embedded Quotes: `"List ""A"""`
  - Unicode identifier: `U&"\0441\043B\043E\043D"`
- String Literal:
  - Simple: `'text'`
  - Dollar: see next slide
  - Embedded Quotes: `'McDonald''s restaurant'`
  - Unicode literal: `U&'\0441\043B\043E\043D'`
- Comments:
  - Single line: `-- comment`
  - Multi-line: `/* comment */`

Dollar Quoting

- $<tag>$
- <tag> is zero or more characters
- Start and End tag must match
- Particularly useful for function bodies
- Works for all character strings
- Nest by choosing different <tag> at each level

```
CREATE OR REPLACE FUNCTION dummy()
RETURNS text AS $_$
BEGIN
  RETURN $$Say 'hello'$$;
END;
$_$ LANGUAGE plpgsql;
```

http://www.postgresql.org/docs/10/static/sql-syntax-lexical.html#SQL-SYNTAX-DOLLAR-QUOTING

Data Types - Character Strings

- `text`: character string with variable length without upper limit
- `varchar(X)`: character string, at most X characters long
- `char(X)`: character string, exactly X characters long (padded with spaces)

Data Types - Numbers

- `smallint`: integer (2 bytes) (alias: int2)
- `integer`: integer (4 bytes) (alias: int4, int)
- `bigint`: integer (8 bytes) (alias: int8)
- `serial`: not a real datatype – int with a default value expression that automatically takes the next value from a sequence
- `bigserial`: the same using bigint as base type
- `real`: floating-point number (4 bytes) (alias: float4)
- `double precision`: floating-point number (8 bytes) (alias: float8)
- `numeric(M,N)`: fixed-point number, max. M digits total, thereof N digits after the decimal point

Data Types - Date/Time

- `date`: date (without time)
- `time [without time zone]`: time without time zone
- `time with time zone`: time with time zone
- `timestamp [without time zone]`: date and time without time zone
- `timestamp with time zone`: date and time with time zone
- `interval`: interval length

Data Types - Enumeration Types

- Real enumeration type
- Seamless operator and function support

```
CREATE TYPE color AS enum('red', 'green', 'blue');
CREATE TABLE clothes (type text, color color);
INSERT INTO clothes VALUES('shirt', 'red');
```

Data Types - Geometric

- Don't use native, use PostGIS instead
- Out of scope for this talk

```
CREATE EXTENSION postgis;
```

Data Types - JSON

- Checks well-formedness
- Compliant JSON requires UTF-8 server encoding
- Stored an exact copy of the input text as a string
  - Preserves semantically-insignificant white space between tokens
  - Preserves order of keys within JSON objects
  - If contains same key more than once, all key/value pairs are kept

Data Types - JSONB

- Checks well-formedness
- Compliant JSON requires UTF-8 server encoding
- Full indexing
- Stored in a decomposed binary format
  - Does not preserve white space
  - Does not preserve the order of object keys
  - Does not keep duplicate object keys (only the last value is kept)

`https://www.postgresql.org/docs/10/static/datatype-json.html`

Data Types - XML Type

- Checks well-formedness
- Requires additional support from the underlying operating system (libxml2, libxslt)
- Needs some care regarding encoding issues (UTF-8 vs. others)
- No comparison or indexable operators available
- No DTD validation

Data Types - Additional

- `bytea`: binary data
- `boolean`: `true` or `false`
- `array`: multidimensional arrays from elements of same scalar data type
- `composite`: tuples from elements of any data type
- `range`: represent a range of values of some element type
- `text search`: designed to support full text search
- `other`: many other types for network addresses, uuid, ...

`https://www.postgresql.org/docs/10/static/datatype.html`

Thank You

- Questions?