PostgreSQL 10 or above deployment on Linux

Summary: CAST does not provide a CAST Storage Server (CSS) installer for Linux environments, however, CAST recommends installing the equivalent PostgreSQL version on a Linux server to take advantage of the superior performance over CSS installed on Windows - please follow the instructions below to do so.

Note that these instructions are for deploying PostgreSQL 10 or above.

Installation via YUM

PostgreSQL can be installed using RPMs (binary) or SRPMs (source) managed by **YUM**. This is available for the following Linux distributions (**CAST recommends using only 64-bit platforms**):

- Fedora
- CentOS
- Red Hat Enterprise Linux

Configure your YUM repository

Locate and edit your distribution **.repo** file, located:

- On **Fedora**: */etc/yum.repos.d/fedora.repo* and */etc/yum.repos.d/fedora-updates.repo*, **[fedora]** sections
- On **CentOS**: */etc/yum.repos.d/CentOS-Base.repo*, **[base]** and **[updates]** sections
- On **Red Hat**: */etc/yum/pluginconf.d/rhnplugin.conf* **[main]** section

To the section(s) identified above, you need to append a line (otherwise dependencies might resolve to the postgresql supplied by the base repository):

```
exclude=postgresql*
```

Update the RPM package

A PGDG file is available for each distribution. Browse **https://yum.postgresql.org/repopackages.php** and find your correct RPM. There is only one single repo RPM for all PostgreSQL versions for each platform. Ensure you modify the link to the rpm file in the command below: this command will download the RPM package for the latest release of **Red Hat**:

```
rpm -ivh https://download.postgresql.org/pub/repos/yum/reporpms/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm
yum update
```

Install PostgreSQL

First run the following command to view the available releases:

```
yum list postgresql*
```

Then run the install, for example to install **PostgreSQL 11**:

```
yum install postgresql11-server
```

Configure data storage location

The default location for PostgreSQL data storage is set to (where **<name>** is the release number):

```
/var/lib/pgsql/<name>/data
```

To customize this location, you need to edit the following file (where **<name>** is the release number):

```
/usr/lib/systemd/system/postgresql-<name>.service
```

Initialize the PostgreSQL database server to configure data storage location

Once installed, please run the following command to initialize the server (where **<name>** is the release number):

```
export PGSETUP_INITDB_OPTIONS="-E 'UTF-8' --no-locale"
/usr/pgsql-<name>/bin/postgresql-<name>-setup initdb
```

Configuring the server/database

Edit the pg_hba.conf file

Edit the pg_hba.conf file at **/usr/local/pgsql/data** using the vi command:

```
vi pg_hba.conf
```

Replace the sample 192.168.x.x IP address and /24 subnet below with the appropriate allowed client IP addresses/subnets:

```
# TYPE DATABASE USER CIDR-ADDRESS METHOD
local all all trust
# IPv4 local connections:
host all all 127.0.0.1/32 trust
host all all 192.168.21.57/24 md5
 # IPv6 local connections:
host all all ::1/128 md5
```

Note that adding the following line will ensure ANY client can connect:

```
host all all 0.0.0.0/0 md5
```

Check server configuration

To check if the server itself is configured correctly for CAST AIP, we need to verify and modify certain parameters in the **postgresql.conf** (the file is in the postgresql directory \db_data).

pgtune

The **postgresql.conf** settings listed below are directly related to available RAM for the PostgreSQL instance - and as such their configuration is specific to the host machine. Please use **http://pgtune.leopard.in.ua/** on the host machine to identify the correct settings and then modify **postgresql.conf** with the settings provided by **pgtune**. You should update the following settings with the values provided by **pgtune**:

- effective_cache_size
- min_wal_size
- max_wal_size
- wal_buffers

When running **pgtune**, choose the following:

- Set the **DB version** to the **installed version**
- Set the **DB Type** to **Mixed type of applications**
- Set the **Number of connections** to **300**

Some of the values suggested by **pgtune** are somewhat low. Therefore, please calculate the following **postgresql.conf** settings as follows:

- shared_buffers - **value should be 25% of available RAM with a max of 8 GB**
- maintenance_work_mem= **512 MB**

Other settings

The following settings should also be modified to the values listed below:

Parameters	Required value	Comments
listen_addresses	'*'	Instead of localhost by default. Enables connectivity from other machines.
Port	2282	
max_connections	300	
fsync	off	
synchronous_commit	off	
full_page_writes	off	
commit_delay	10	
checkpoint_completion_target	0.9	
cursor_tuple_fraction	1.0	
log_checkpoints	on	
log_destination	'stderr'	
logging_collector	on	
log_line_prefix	'%t [%p]: [%l-1] '	Don't forget the space before final quote mark.
log_temp_files	1024kB	
log_autovacuum_min_duration	1000ms	
autovacuum_vacuum_cost_limit	200	
bytea_output	'escape'	
datestyle	'iso, mdy'	Ensure that this option is active (it may already be active).
lc_messages	'C'	
lc_monetary	'C'	
lc_numeric	'C'	
lc_time	'C'	
max_locks_per_transaction	4096	Higher value than 64 by default.
standard_conforming_strings	on	

Start the PostgreSQL database server

Where **<name>** is the release number:

```
systemctl start postgresql-<name>.service
systemctl enable postgresql-<name>.service
systemctl status postgresql-<name>.service
```

Create users

By default **PostgreSQL** will create a system account user named **postgres** with the role **postgres**. The equivalent CAST Storage Service provided by CAST includes two default users as follows:

Username	Password	Permissions	Notes
operator	CastAIP	SUPERUSER	-
guest	WelcomeToAIP	-	Note that in the CAST AIP **8.3.11**, the "guest" user is no longer used.

If you would like to create these users, use the following commands:

```
psql
*create user operator with SUPERUSER password 'CastAIP';
*create user guest with password 'WelcomeToAIP';
*grant postgres to operator;
```

CAST AIP 8.3.12 - custom users

If you are using **CAST AIP 8.3.12**, you are also free to create your own users and then use them with CAST AIP and related software - you need to have a minimum of one user with the SUPERUSER permission and then grant the "postgres" role to this user. For example:

```
[postgres@cssx data]$ psql
*create user my_user with SUPERUSER password 'my_password';
*grant postgres to my_user;
```

When using **CAST AIP 8.3.30**, it is possible to create custom users that do NOT require the SUPERUSER permission if you prefer not to grant this. For example, this script creates a role "my_role" with the password "my_password" that can login. The role has not been granted the SUPERUSER permission and instead only the minimum permissions required to operate CAST AIP are granted:

```
[postgres@cssx data]$ psql
*create role my_role LOGIN PASSWORD 'my_password' NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
*grant create, connect, temporary on database postgres to my_role;
```

postgres database

All CAST AIP products will expect the CAST AIP schemas to be members of the **postgres database** - i.e. all connections to a PostgreSQL instance will connect direct to the **postgres database**. This database is created by default during the installation of the PostgreSQL instance and should be retained. While you can create an alternative custom database, it is not possible to connect to a custom database using any CAST AIP products.

How to log all statement's plan

Change the following parameters in the configuration file (..\db_data\postgresql.conf):

```
auto_explain.log_min_duration = '0s'           # setting this to zero logs all plan
auto_explain.log_nested_statements = on        # log statements executed inside a function
auto_explain.log_verbose = on                  # it's equivalent to the VERBOSE option of EXPLAIN
auto_explain.log_buffers = on                  # it's equivalent to the BUFFERS option of EXPLAIN
auto_explain.log_analyze = on                  # causes EXPLAIN ANALYZE output; when this parameter is on, per-
plan-node timing occurs for all statements executed, whether or not they run long enough to actually get
logged. This can have an extremely negative impact on performance. Turning off auto_explain.log_timing
ameliorates the performance cost, at the price of obtaining less information.
shared_preload_libraries = 'auto_explain'      # change requires restart
```

Set this parameter to **off** when only actual row counts, and not exact times, are needed:

```
auto_explain.log_timing = off                  #  this parameter has no effect unlessauto_explain.log_analyze
is enabled
```