**DOCKER, NGINX & POSTGRESQL BASIC**

** **

****

**Documented by:**

HARSHAA S S

**Contents by:**

Dockers, Nginx – HARSHAA S S

PostgreSQL – Naveen Kumar H

**Reviewers:**

LEWIS JOHNSTON, NAVEENKUMAR H

**DOCKER:**

Docker is an application that makes it simple and easy to run application processes in a container, which are like virtual machines, only more portable, more resource-friendly, and more dependent on the host operating system. In Docker, there are images and there are containers. The two are closely related, but they are different from each other partially.

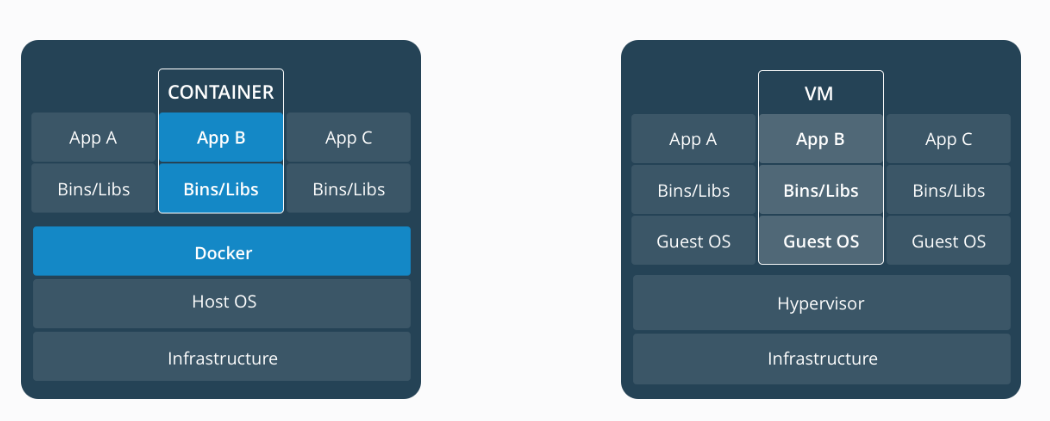
**Docker Images**

An image is a file that's essentially a snapshot of a container. Images are created with the build command, and they'll produce a container when started with run. Images are stored in a Docker registry such as registry.hub.docker.com. Because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.

**Docker Containers**

Containers are instances of images. Containers are hopefully why you're using Docker; they're lightweight and portable encapsulations of an environment in which to run applications. **docker ps** only outputs running containers. You can view all containers (running or stopped) with **docker ps -a.**

**Docker Containers versus VM: (from docker.com)**

****

**Containers:**

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), and start almost instantly.

**VMs:**

Virtual machines (VMs) are an abstraction of physical hardware turning one server into many servers. The hypervisor allows multiple VMs to run on a single machine. Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs. VMs can also be slow to boot.

**DOCKER INSTALLATION:**

To start using the Docker first we need to install the Docker repository. The following are the steps that should be followed while installing the Docker repository.

* Now let's install Docker. Add the GPG key for the official Docker repository to the system.

**sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys 58118E89F3A912897C070ADBF76221572C52609D**

* Add the Docker repository to APT sources

**sudo apt-add-repository 'deb https://apt.dockerproject.org/repo ubuntu-xenial main'**

* Update the repo and install docker engine

**sudo apt-get update && sudo apt-get install -y docker-engine**

* To check the Docker status, Run the following command.

**sudo service docker status**

**DOCKER COMMANDS:**

* To search for an online repo, we need to use the following command.

**sudo docker search docker\_image\_name**

* Select the official repo and install it by using the following command.

**sudo docker pull docker\_image\_name**

* To check the status of the docker run the command.

**sudo docker ps –a**

* To install the container with the docker run the following command.

**sudo docker run -it docker\_image\_name - to create with default name**

* To create the container with the default fields and mount points

**sudo docker run -v /pappaya/odoo\_filestore:/var/lib/odoo -v /pappaya/odoo\_config:/etc/odoo -v /pappaya/odoo\_addons:/opt/odoo/extra-addons -v /pappaya/odoo\_logs:/var/log/odoo -p 8069:8069 --name odoo\_pappaya -t odoo\_pappaya >>/dev/null 2>>/dev/null &**

* Use the following command to enter in to the container.

**sudo docker-enter container\_name**

* Command to stop a container

**sudo docker stop container\_name**

* Command to start a container

**sudo docker start container\_name**

* Command to restart a container

**sudo docker restart container\_name**

* Command to delete a container (Containers needs to be turned off manually before deletion)

**sudo docker rm container\_name/container\_id**

* Command to delete a Docker image (Docker image cannot be deleted if there are running containers using the image)

**sudo docker rmi docker\_image\_name**

* Customized command for odoo\_db\_update

**sudo docker exec -t container\_name /bin/bash ""./opt/odoo/odoo.sh dbname internalP""**

**NGINX WEBSERVER:**

**Webserver:**

A Web server is a program that uses HTTP (Hypertext Transfer Protocol) to serve the files that form Web pages to users, in response to their requests, which are forwarded by their computers' HTTP clients. Dedicated computers and appliances may be referred to as Web servers as well.

**Nginx:**

Nginx is free and open source software, released under the terms of a BSD-like license. A company of the same name was founded in 2011 to provide support. Nginx can be deployed to serve dynamic HTTP content on the network using FastCGI, SCGI handlers for scripts, WSGI application servers or Phusion Passenger modules, and it can serve as a software load balancer.

Nginx uses an asynchronous event-driven approach to handling requests. Nginx's modular event-driven architecture can provide more predictable performance under high loads.

**Nginx versus Apache:**

Nginx was written with an explicit goal of outperforming the Apache web server.[44] Out of the box, serving static files, Nginx uses dramatically less memory than Apache, and can handle roughly four times more requests per second. This performance boost comes at a cost of decreased flexibility, such as the ability to override systemwide access settings on a per-file basis (Apache accomplishes this with an .htaccess file, while Nginx has no such feature built in).[46] Formerly, adding third party modules to nginx required recompiling the application from source with the modules statically linked. This was partially overcome in version 1.9.11 with the addition of dynamic module loading. However, the modules still must be compiled at the same time as nginx, and not all modules are compatible with this system; those require the older static linking process.

**NGINX INSTALLATION:**

* Command to install on Ubuntu Server

**sudo apt -y install nginx**

**NGINX COMMANDS:**

* Command to Restart, Reload, Stop, Start Nginx

**sudo service nginx start**

**sudo service nginx stop**

**sudo service nginx reload**

**sudo service nginx restart**

* Command to check the nginx service status

**sudo service nginx status**

**ps -aux | grep nginx**

* Command to check the Nginx version and modules complied

**nginx -V**

* Upgrade Nginx Server

**sudo service nginx upgrade**

* Test the nginx server for errors

**nginx -t**

**NGINX FILES:**

* All files of Nginx webserver will be at

**/etc/nginx/**

* Main config file

**/etc/nginx/nginx.conf**

* Nginx reverse proxy and other config files

**/etc/nginx/sites-enables**

**/etc/nginx/sites-available**

\*sites-enabled and sites-available has the same files inside the directory. Nginx itself creates a default file called as default for testing purpose.

* Creating software links in sites-enabled:

We’re creating software links in sites-enabled for files in sites-available because, the webserver will enable site only if the site is available in sites-enabled list. This means that there may be many sites-available whereas only enabled sites are only accessable. So, if we make changes to files in sites-available will reflect automatically in sites-enabled.

**cd /etc/nginx/sites-enabled**

**sudo ln -s /etc/nginx/sites-available/pappaya1 pappaya1**

* Website file (example)

upstream webserver {

server 127.0.0.1:8069 weight=1 fail\_timeout=300s;

}

server {

listen 80;

server\_name new.pappaya.co.uk new1.pappaya.co.uk;

# Strict Transport Security

add\_header Strict-Transport-Security max-age=2592000;

rewrite ^/.\*$ https://$host$request\_uri? permanent;

}

server {

# server port and name

listen 443;

server\_name new.pappaya.co.uk new1.pappaya.co.uk;

# Specifies the maximum accepted body size of a client request,

# as indicated by the request header Content-Length.

client\_max\_body\_size 200m;

# ssl log files

access\_log /var/log/nginx/pappaya-access.log;

error\_log /var/log/nginx/pappaya-error.log;

# ssl certificate files

ssl on;

ssl\_certificate /etc/ssl/nginx/server.crt;

ssl\_certificate\_key /etc/ssl/nginx/server.key;

# add ssl specific settings

keepalive\_timeout 60;

# limit ciphers

ssl\_ciphers HIGH:!ADH:!MD5;

ssl\_protocols SSLv3 TLSv1;

ssl\_prefer\_server\_ciphers on;

# increase proxy buffer to handle some OpenERP web requests

proxy\_buffers 16 64k;

proxy\_buffer\_size 128k;

location / {

proxy\_pass http://webserver;

# force timeouts if the backend dies

proxy\_next\_upstream error timeout invalid\_header http\_500 http\_502 http\_503;

# set headers

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forward-For $proxy\_add\_x\_forwarded\_for;

# Let the OpenERP web service know that we're using HTTPS, otherwise

# it will generate URL using http:// and not https://

proxy\_set\_header X-Forwarded-Proto https;

# by default, do not forward anything

proxy\_redirect off;

}

# cache some static data in memory for 60mins.

# under heavy load this should relieve stress on the OpenERP web interface a bit.

location ~\* /web/static/ {

proxy\_cache\_valid 200 60m;

proxy\_buffering on;

expires 864000;

proxy\_pass http://webserver;

}

}

\*Please be aware of spaces and letter case since nginx is case sensitive and throws error if the file has un wanted space.

\*Port number can be changed to map an application running in different port.

\*server\_name is the block were the dns names should be mapped.

\*for each change in config files, nginx restart required.

**POSTGRESQL:**

**Postgres Architecture:**



**CLIENTS PROCESS:**

- Whenever we issue a query or the action made by us (client) is called the client process.   
 - It is front end.  
 - Front end may be a text application, graphical application or web server page.  
 - Through TCP/IP clients access the server   
 - Many users at a time can access the DB  
 - FORKS – This process makes multi user access possible. It don’t disturb the postgres process

**POSTMASTER:**

The work of postmaster is that it authenticates the port (5432) and allocates process for users.

**SERVER PROCESS:**

It is also called as postgres. It accepts the connection from the clients(we) like database files and manages the database action.

**CHECKPOINTS:**

When checkpoints occur, all the dirty pages must write to disk. If we increase the checkpoint\_segments then checkpoint will occur less and so I/O will be less as it need to write less to disk. IF large amount of data is inserted there is more generation of checkpoints.

Write-Ahead Logging (WAL) puts a checkpoint in the transaction log every so often. The CHECKPOINT command forces an immediate checkpoint when the command is issued, without waiting for a scheduled checkpoint.

A checkpoint is a point in the transaction log sequence at which all data files have been updated to reflect the information in the log. All data files will be flushed to disk.

If executed during recovery, the CHECKPOINT command will force a restartpoint rather than writing a new checkpoint.

Only superusers can call CHECKPOINT. The command is not intended for use during normal operation.

**PG\_LOG:**

It is not an actual postgres directory, it is the directory where RHEL stores the actual textual LOG.

**PG\_XLOG:**

Here the write ahead logs are stored. It is the log file, where all the logs are stored of committed and un committed transaction. It contains max 6 logs, and last one overwrites. If archiver is on, it moves there.

**PG\_CLOG:**

It contains the commit log files, used for recovery for instant crash

**PG\_VERSION:**

A file containing the major version number of PostgreSQL

**Base:**

Subdirectory containing per-database subdirectories

**PG\_MULTIXACT:**

Subdirectory containing multitransaction status data (used for shared row locks)

**PG\_SUBTRANS:**

Subdirectory containing subtransaction status data

**PG\_TBLSPC:**

Subdirectory containing symbolic links to tablespaces

**PG\_TWOPHASE:**

Subdirectory containing state files for prepared transactions

**POSTMASTER.OPTS:**

A file recording the command-line options the postmaster was last started with

**POSTMASTER.PID:**

A lock file recording the current postmaster PID and shared memory segment ID (not present after postmaster shutdown)

**Comparing LCT and TE database tables:**

Login to LCT Server

Sudo su – postgres

**This commands save the list of table names in lct.txt**

psql -d lct -c "select table\_name from information\_schema.tables where table\_schema =

'public' order by table\_name;">lct.txt

C:\Users\Havish\Documents\Lightshot\Screenshot_96.png

**Login to TE Server**

Sudo su – postgres

**This commands save the list of table names in TE.txt**

psql -d lct -c "select table\_name from information\_schema.tables where table\_schema = 'public' order by table\_name;">te.txt

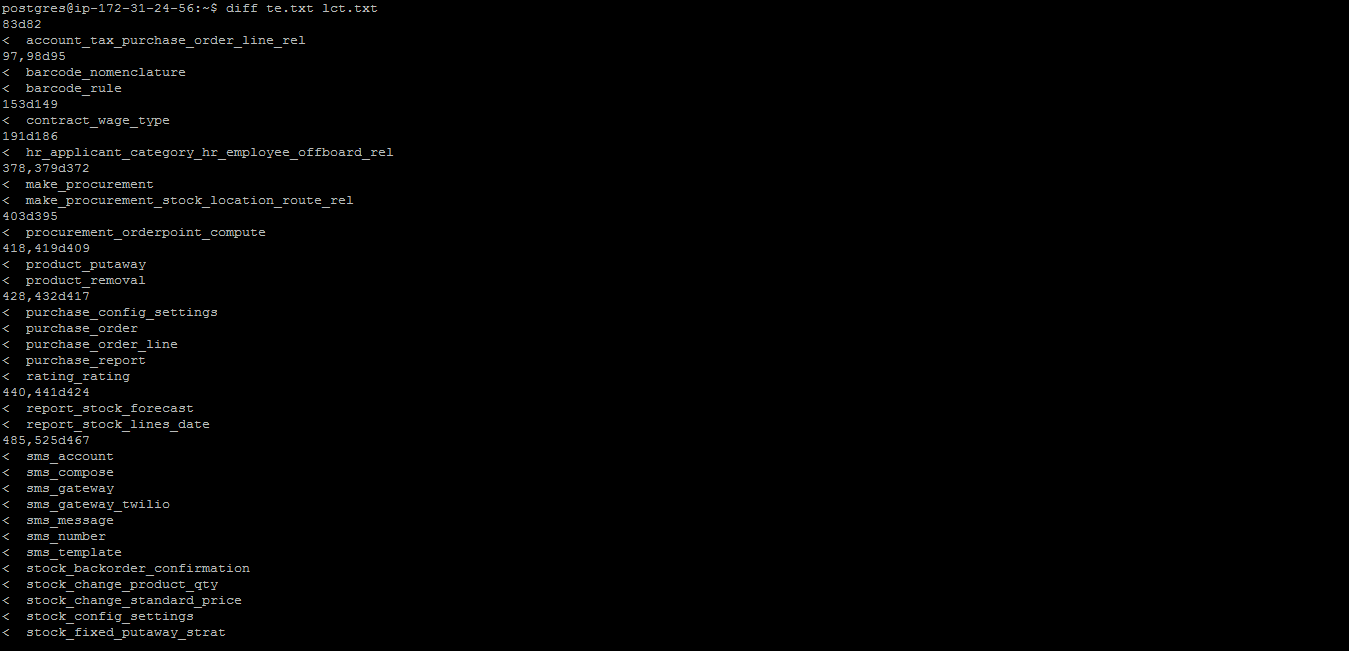
C:\Users\Havish\Documents\Lightshot\Screenshot_97.png

**Send TE.txt to lct server**

Scp /var/lib/postgresql/TE.txt ubuntu@52-209-35-196:/var/lib/postgresql/

**Compare both files using diff command:**

Diff lct.txt TE.txt



List of tables shown after the command executes.

**This commands used to login postgres user**

Sudo su – postgres

**This commands used to login postgres command prompt**

Psql

**This command is used to switch database form one database to another database**

\c databasename

**This command used to list the database and database size**

\l+

**This command used to list of schema in database**

\dn

**This command used to list of users in database**

\du

**This query is used to list of indexes in database**

Select \* from pg\_indexes;

**This query is used to list of indexes in database**

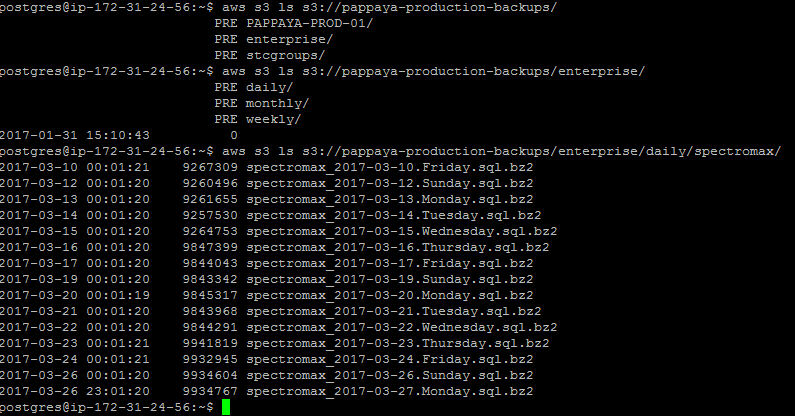
Select \* from pg\_tables;

**Daily Activities of database:**

Checking Backups in aws s3 bucket

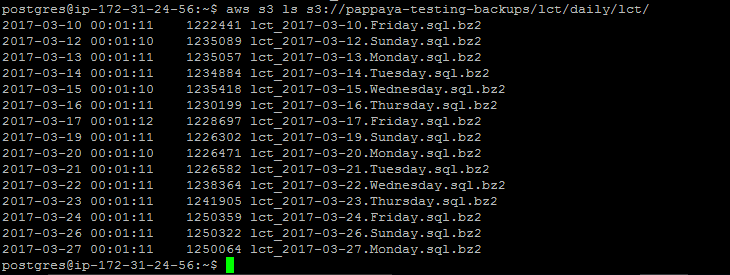
**Production backups**

Aws s3 ls s3://pappaya-production-backups/enterprise/spectromax/



**Testing backups**

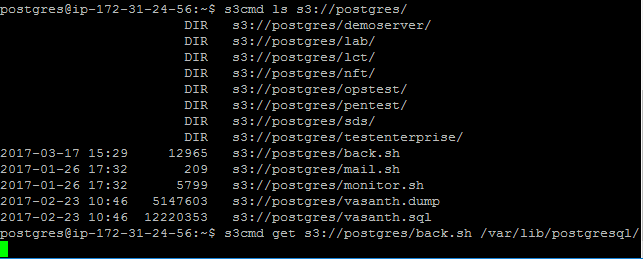
Aws s3 ls s3://pappaya-testing-backups/lct/daily/lct



**Scheduling Backup script in New instances**

🡪Download backup script from s3cmd bucket

S3cmd get backup.sh /var/lib/postgresql/



🡪In Backup script change the location of backup directory

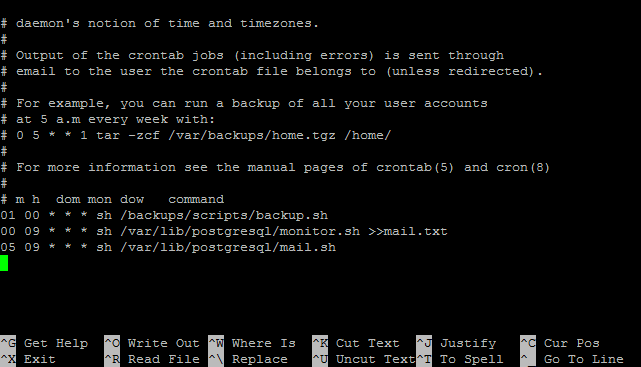
/backups/stcgroup/

🡪Schedule backups script in crontab

Crontab –e

01 \* \* \* \* sh /backups/stcgroup/back.sh

Above commands will execute daily at 12.01am and will move backups to s3 bucket



**Postgresql commands:**

1) # su - postgres

2) start the instance

./pg\_ctl start -D /opt/Postgresql/9.5/data

./pg\_ctl stop -D /opt/Postgresql/9.5/data

./pg\_ctl restart -D /opt/Postgresql/9.5/data

3) -bash-3.2$ cd bin

-bash-3.2$ ./createdb <dbname>

./initdb -D /opt/Postgres /9.1/data1 --creating cluster

data1 dir we need to create and give permission by

chown enterprisedb:enterprisedb data1

4) ./psql <dbname>

5) To check process:

ps -ef|grep post

6) List DB:

./psql -l - outside DB

\l - Inside DB

7) show Table

\dt

8) come out of DB:

\q

9) Size of DB-

select pg\_database\_size('db\_name');

10) table size:

select pg\_size\_pretty(pg\_total\_relation\_size('table\_name'));

**USERS:**

when the uer create a table, those table belong to the PUBLIC schema.

Database can be split up between multiple users using schemas.

sssdb=# create user sss1 with password 'sss';

**CREATE ROLE:**

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-----------

enterprisedb | Superuser, Create role, Create DB, Replication | {}

sss1 | | {}

sssdb=# alter user sss1 superuser;

**ALTER ROLE:**

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-----------

enterprisedb | Superuser, Create role, Create DB, Replication | {}

sss1 | Superuser | {}

sssdb=# \conninfo

You are connected to database "sssdb" as user "sss1" via socket in "/tmp" at port "5444".

sssdb=# select systimestamp from dual;

systimestamp

----------------------------------

29-OCT-12 18:17:07.907135 +05:30

(1 row)

sssdb=# select sysdate from dual;

sysdate

--------------------

29-OCT-12 18:17:14

(1 row)

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-----------

enterprisedb | Superuser, Create role, Create DB, Replication | {}

sss1 | Superuser | {}

Rename users:

sssdb=# alter user nana rename to naana;

**GROUPS:**

sssdb=# create GROUP group1;

CREATE ROLE

sssdb=# alter GROUP group1 add user naana;

ALTER ROLE

sssdb=# alter group group1 add user deena;

ALTER ROLE

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-----------

deena | | {group1}

enterprisedb | Superuser, Create role, Create DB, Replication | {}

group1 | Cannot login | {}

naana | | {group1}

sss1 | Superuser | {}

sssdb=# alter group group1 rename to group\_one;

ALTER ROLE

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-------------

deena | | {group\_one}

enterprisedb | Superuser, Create role, Create DB, Replication | {}

group\_one | Cannot login | {}

naana | | {group\_one}

sss1 | Superuser | {}

sssdb=# alter group group\_one drop user naana;

ALTER ROLE

sssdb=# \du

List of roles

Role name | Attributes | Member of

--------------+------------------------------------------------+-------------

deena | | {group\_one}

enterprisedb | Superuser, Create role, Create DB, Replication | {}

group\_one | Cannot login | {}

naana | | {}

sss1 | Superuser | {}

**SCHEMA:**

* + each user have their own schema and multiple schema
  + the schema search path can be altered so that each user find their tables first and access the other user tables also.
  + schema as a namespace for tables that also provides security.
  + By creating a schema for users and granting them rights on that schema,
  + The user will create tables on that schema instead of public.
  + schema search path is set which users objects set in postgreSQL database.
  + default for all users $user, public.
  + search path can be set on a users basis, to look thorough any random list of schemas.

**Create a schema:**

jlaxmi=# create schema joe;

CREATE SCHEMA

**To create a table in spcefic schema as user default:**

jlaxmi=# create table joe.dept(

jlaxmi(# dept\_id number(5),department varchar2(10));

CREATE TABLE

jlaxmi=# \d joe.dept;

Table "joe.dept"

Column | Type | Modifiers

------------+-----------------------+-----------

dept\_id | numeric(5,0) |

department | character varying(10) |

jlaxmi=# jlaxmi=# \du

enterprisedb | Superuser, Create role, Create DB, Replication | {}

jack | Superuser, Create DB | {}

jlaxmi=# show search\_path;

"$user",public

jlaxmi=# select \* from user\_tables;

owner | schemaname | table\_name | table\_space | status

-------+------------+------------+-------------+--------

JACK | PUBLIC | STUDENT | | VALID

JACK | PUBLIC | TEST\_NN | | VALID

JACK | PUBLIC | DEPT | | VALID

JACK | PUBLIC | STUD | | VALID

JACK | JOE | DEPT | | VALID

(5 rows)

jlaxmi=# insert into joe.dept values(1011,'Research');

INSERT 0 1

jlaxmi=# insert into joe.dept values(1012,'Sales');

INSERT 0 1

jlaxmi=# select \* from joe.dept;

dept\_id | department

---------+------------

1011 | Research

1012 | Sales

**As a user can create multiple schema:**

jlaxmi=# create schema steve;

CREATE SCHEMA

jlaxmi=# create table steve.dept (dept\_id number(5), department varchar2(10));

CREATE TABLE

jlaxmi=# insert into steve.dept values(1002,'HR');

INSERT 0 1

jlaxmi=# insert into steve.dept values(1013,'Manager');

INSERT 0 1

jlaxmi=# commit;

COMMIT

jlaxmi=# select \* from steve.dept;

dept\_id | department

---------+------------

1002 | HR

1013 | Manager

(2 rows)

jlaxmi=# select \* from user\_tables;

owner | schemaname | table\_name | table\_space | status

-------+------------+------------+-------------+--------

JACK | PUBLIC | STUDENT | | VALID

JACK | PUBLIC | TEST\_NN | | VALID

JACK | PUBLIC | DEPT | | VALID

JACK | PUBLIC | STUD | | VALID

JACK | JOE | DEPT | | VALID

JACK | STEVE | DEPT | | VALID

(6 rows)

**We can permanently assign search path for a user:**

jlaxmi=# alter user jack set search\_path to joe,steve;

ALTER ROLE

jlaxmi=# show search\_path;

search\_path

-------------

joe

(1 row)

**Backup and Restore:**

* As everything that contains valuable data, PostgreSQL DBs should be backed up regularly.
* DB backups allow DBs to be restored if a disk drive fails, a table is accidentally dropped, or a DB file is accidentally deleted.
* The idea behind the SQL-dump method is to generate a text file with SQL commands that, when fed back to the server, will recreate the DB in the same state as it was at the time of the dump.
* Dump Individual DBs with pg\_dump

PostgreSQL provides the utility program pg\_dump for dumping individual DBs:

* pg\_dump dbname > outfile
* pg\_dump writes its results to the standard output.
* pg\_dump is a regular PostgreSQL client application.
* This means that you can do this backup procedure from any remote host that has access to the DB.
* pg\_dump does not operate with special permissions. You must have read access to all tables that you want to back up.
* Large objects (blobs) are not dumped by default,

Restoring the dump

* The text files created by pg\_dump are intended to be read in by the psql program:

psql dbname < infile

The DB dbname will not be created by this command, you must create it yourself before.   
psql and pg\_dump support options for controlling the DB server location and the user names.

Example:

Dump the workshop DB in the file "workshop.dump":

$> pg\_dump workshop >workshop.dump

Look at the file "workshop.dump":

$> more workshop.dump

Create a new DB for restoring:

$> createdb dump\_test

Restore the workshop DB into the dump\_test DB:

$> psql dump\_test < workshop.dump

Connect to the dump\_test DB:

$> psql dump\_test

The ability of pg\_dump and psql to write to or read from pipes makes it possible to dump a DB directly from one server to another, for example:

pg\_dump -h host1 dbname | psql -h host2 dbname

Tricks for large DBs:

Compressed dumps:

pg\_dump dbname | gzip > filename.gz

Reload with

createdb dbname

gunzip -c filename.gz | psql dbname

split allows you to split the output into pieces that are acceptable in size to the underlying file system. For example, to make chunks of 1 megabyte:

pg\_dump dbname | split -b 1m - filename

Reload with

createdb dbname

cat filename\* | psql dbname

**File system level backup**

An alternative backup strategy is to directly copy the files that PostgreSQL uses to store the data in the database cluster with whatever method you prefer for doing file system backups, for example:

tar -cf backup.tar /home/fred/databases/postgresql/data

* The database server must be shut down before.
* It will not work to restore only certain individual tables or databases from their respective files or directories, because the information contained in these files must be combined with the commit log files pg\_clog/\*, which contain the commit status of all transactions.
* The file system backup will likely be larger than an SQL dump, because a pg\_dump does not need to dump the contents of indexes for example, just the commands to recreate them.

pg\_dump -t MyTable mydb > db.sql

-bash-3.2$ ./pg\_dump -U sss1 -c -f /u02/spl\_bkp/sample\_bakcup sssdb

Password:

[root@asmhost u02]# cd spl\_bkp/

[root@asmhost spl\_bkp]# ls

sample\_bakcup

[root@asmhost spl\_bkp]# cat sample\_bakcup

--

-- EnterpriseDB database dump

--

SET statement\_timeout = 0;

SET client\_encoding = 'UTF8';

SET standard\_conforming\_strings = on;

SET check\_function\_bodies = false;

SET client\_min\_messages = warning;

DROP EXTENSION edb\_dblink\_oci;

DROP EXTENSION edb\_dblink\_libpq;

DROP EXTENSION plpgsql;

DROP SCHEMA public;

--

-- Name: public; Type: SCHEMA; Schema: -; Owner: enterprisedb

--

CREATE SCHEMA public;

ALTER SCHEMA public OWNER TO enterprisedb;

--

-- Name: SCHEMA public; Type: COMMENT; Schema: -; Owner: enterprisedb

--

COMMENT ON SCHEMA public IS 'Standard public schema';

--

-- Name: plpgsql; Type: EXTENSION; Schema: -; Owner:

--

CREATE EXTENSION IF NOT EXISTS plpgsql WITH SCHEMA pg\_catalog;

--

-- Name: EXTENSION plpgsql; Type: COMMENT; Schema: -; Owner:

--

COMMENT ON EXTENSION plpgsql IS 'PL/pgSQL procedural language';

--

-- Name: edb\_dblink\_libpq; Type: EXTENSION; Schema: -; Owner:

--

CREATE EXTENSION IF NOT EXISTS edb\_dblink\_libpq WITH SCHEMA pg\_catalog;

--

-- Name: EXTENSION edb\_dblink\_libpq; Type: COMMENT; Schema: -; Owner:

--

COMMENT ON EXTENSION edb\_dblink\_libpq IS 'EnterpriseDB Foreign Data Wrapper for PostgreSQL';

--

-- Name: edb\_dblink\_oci; Type: EXTENSION; Schema: -; Owner:

--

CREATE EXTENSION IF NOT EXISTS edb\_dblink\_oci WITH SCHEMA pg\_catalog;

--

-- Name: EXTENSION edb\_dblink\_oci; Type: COMMENT; Schema: -; Owner:

--

COMMENT ON EXTENSION edb\_dblink\_oci IS 'EnterpriseDB Foreign Data Wrapper for Oracle';

--

-- Name: public; Type: ACL; Schema: -; Owner: enterprisedb

--

REVOKE ALL ON SCHEMA public FROM PUBLIC;

REVOKE ALL ON SCHEMA public FROM enterprisedb;

GRANT ALL ON SCHEMA public TO enterprisedb;

GRANT ALL ON SCHEMA public TO PUBLIC;

EnterpriseDB database dump complete

sssdb=# create database sssnew;

CREATE DATABASE

sssdb=# .\l

List of databases

Name | Owner | Encoding | Collate | Ctype | Access

privileges

-----------+--------------+----------+-------------+-------------+--------------

-----------------

edb | enterprisedb | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

sssdb | enterprisedb | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

sssnew | sss1 | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |

Backup the database:

-bash-3.2$ ./pg\_dump -U sss1 -Fc -f /u02/spl\_bkp/level2\_bkp sssdb

Restore the database:

-bash-3.2$ ./pg\_restore -l /u02/spl\_bkp/level2\_bkp

**ODOO Framework**

**Installing Odoo is very simple since we use dockers (Refer Docker part for details on installing). The file will located in the path /etc/odoo/ or can be found at the location /pappaya/odoo\_config/**

**Note: There might be extra dependencies to be installed to the container to make the server up and running.**

**Opener-server.conf:**

**This file is the main configuration of the framework. Take a backup of the file always before making any changes to the file.**

1. **DB\_Filter:**

**DB\_Filter - Disabled**

**A picture containing thing, device, gauge

Description generated with high confidence**

**DB\_Filter – Enabled**



**References:**

**To be filled.**