

Configuring Multiple Instances

As part of homework -2, I implemented a database system for a university with multiple Postgres instances using Docker.

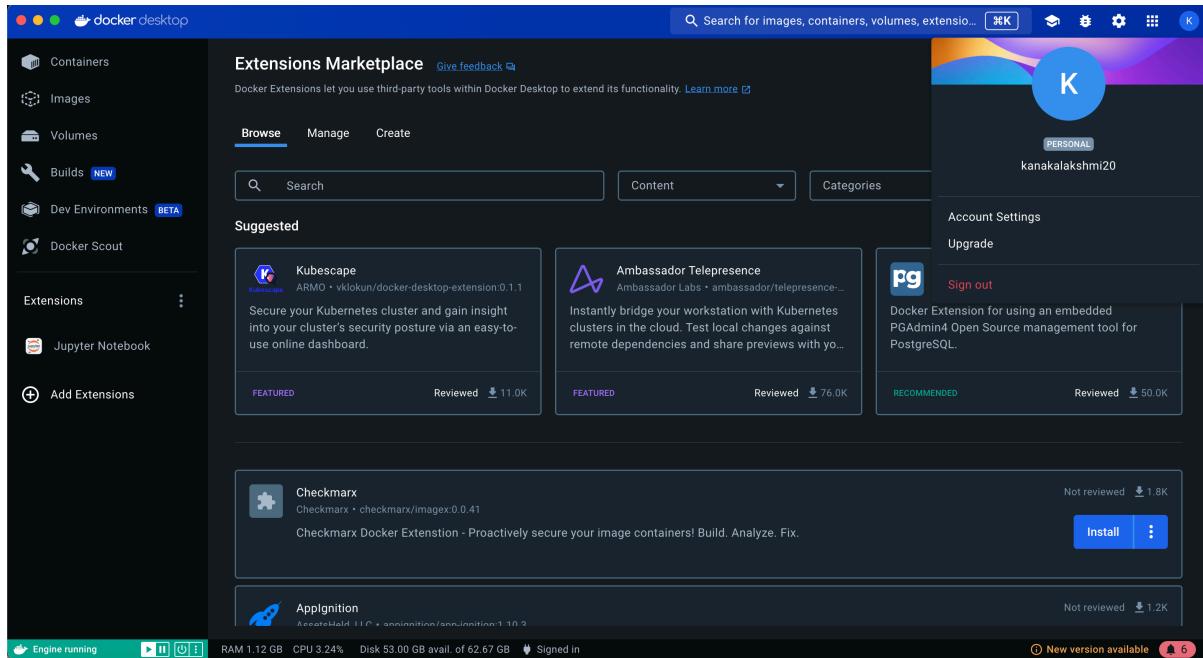


Fig-1: Docker dashboard

Explanation: This is the docker dashboard. We can download the required applications and start working on it. This is widely used when we are integrating with multiple technologies. As part our homework, it required multiple postgres instances, so I found docker is the best choice to implement it easily.

Command:

```
docker run --name primary_node -p 5432:5432 -e POSTGRES_PASSWORD=password -v /Users/lk/Documents/lakshmi_DPDB/HW-2/data_base_volume_mounts/primary:/var/lib/postgresql/data -d postgres
```

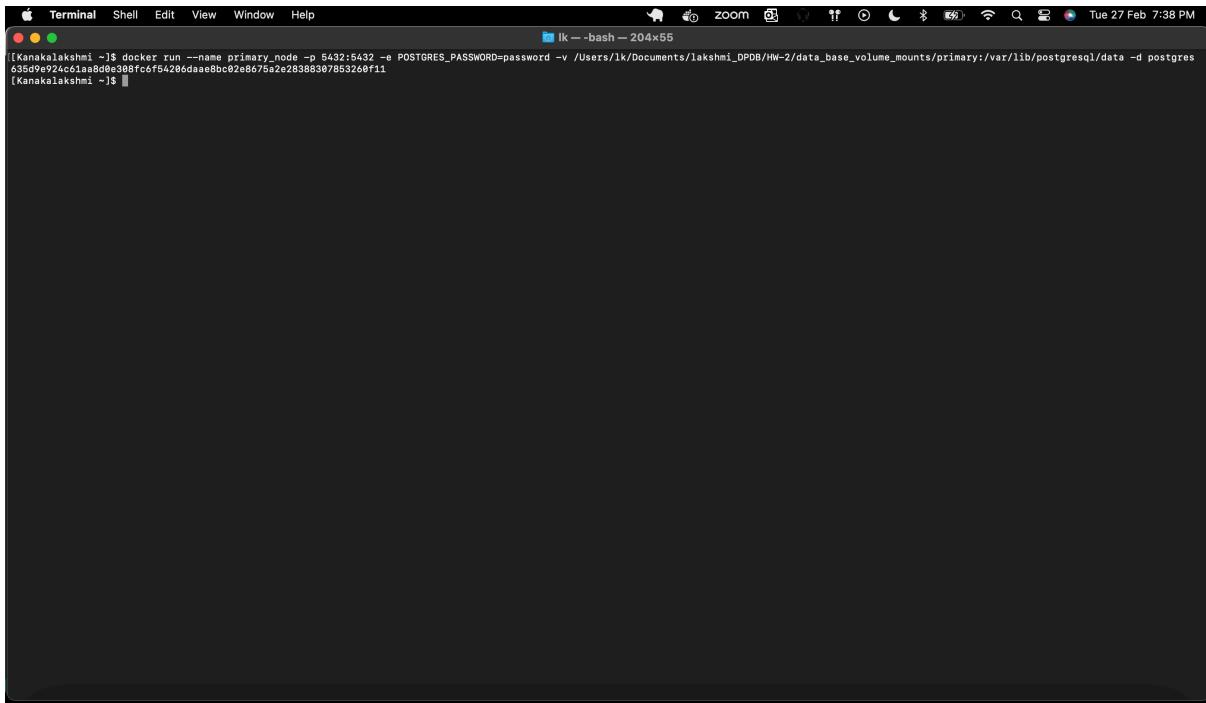
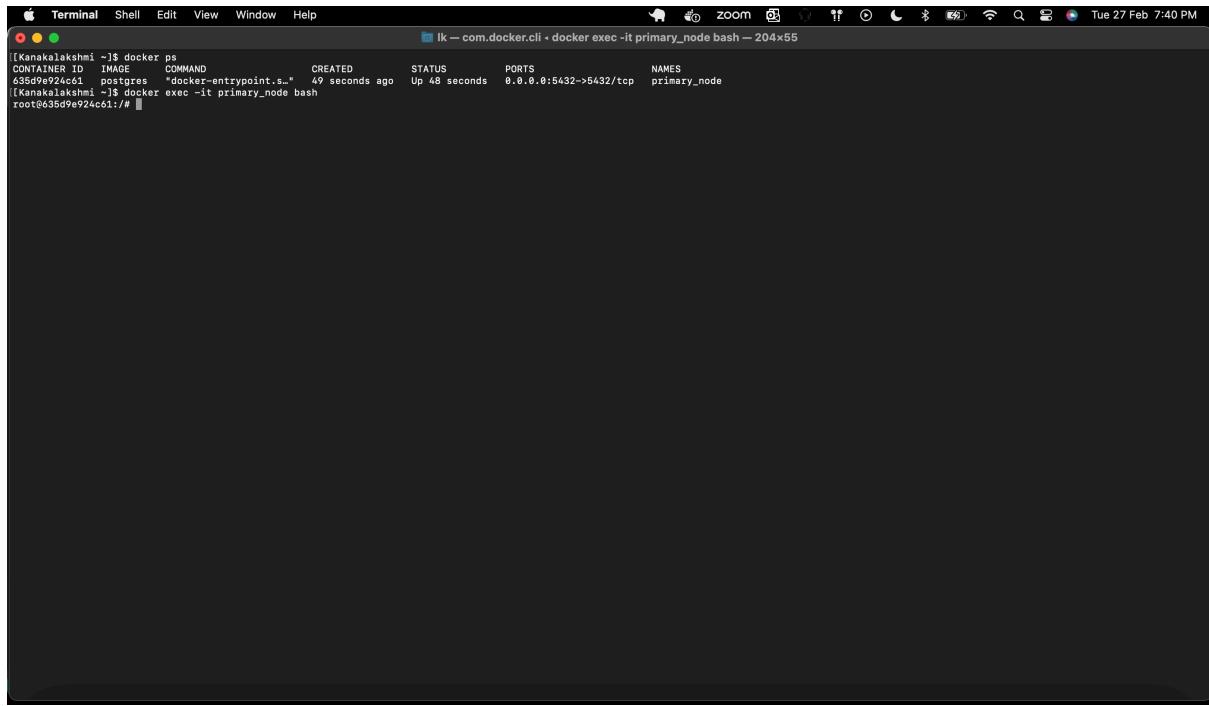


Fig-2: Pulled the docker image.

Explanation: We need to pull the postgres image as we want to work postgres. While pulling the postgres image, we need to provide the name, port_no, password. We are also creating the data_base_volume_mounts, which will be helpful to configure the replica node.

Command:

```
docker ps  
docker exec -it primary_node bash
```



The screenshot shows a macOS Terminal window titled 'lk -- com.docker.cli + docker exec -it primary_node bash — 204x55'. The window contains the following text:

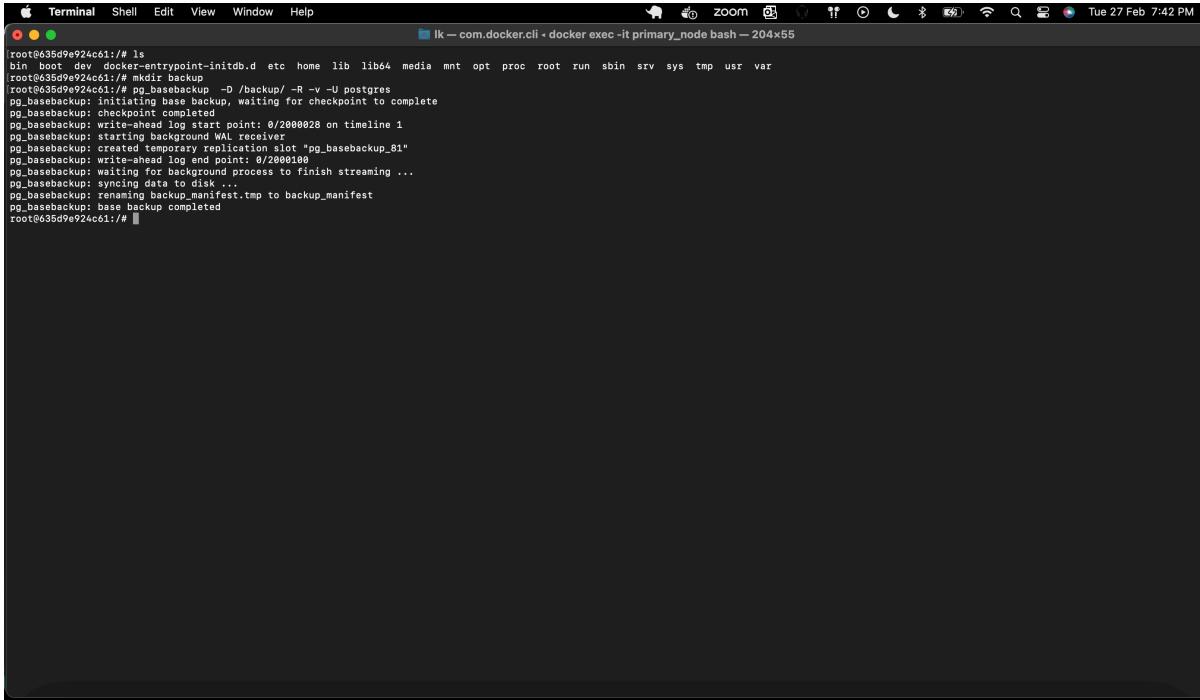
```
[Kanakalakshmi: ~]$ docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES  
635d9e924c61 postgres "docker-entrypoint.s..." 49 seconds ago Up 48 seconds 0.0.0.0:5432->5432/tcp primary_node  
[Kanakalakshmi: ~]$ docker exec -it primary_node bash  
root@635d9e924c61:/#
```

Fig-3: verify docker instances and run the docker container.

Explanation: once the postgres container is created we can see the status of it using the 'docker ps' command. We need to run the docker container using the container name.

Command:

```
ls  
mkdir backup  
pg_basebackup -D /backup/ -R -v -U postgres
```



The screenshot shows a macOS Terminal window titled "Ik — com.docker.cli · docker exec -it primary_node bash — 204x55". The window displays the following command and its output:

```
root@635d9e924c61:/# ls  
bin boot dev docker-entrypoint-initdb.d etc home lib lib64 media mnt opt proc root run sbin srv sys tmp usr var  
root@635d9e924c61:/# mkdir backup  
root@635d9e924c61:/# pg_basebackup -D /backup/ -R -v -U postgres  
pg_basebackup: initiating base backup, waiting for checkpoint to complete  
pg_basebackup: checkpoint completed  
pg_basebackup: write-ahead log start point: 0/2000028 on timeline 1  
pg_basebackup: starting background WAL receiver  
pg_basebackup: created temporary replication slot "pg_basebackup_81"  
pg_basebackup: write-ahead log end point: 0/2000100  
pg_basebackup: waiting for background process to finish streaming ...  
pg_basebackup: syncing data to disk ...  
pg_basebackup: renaming backup_manifest.tmp to backup_manifest  
pg_basebackup: base backup completed  
root@635d9e924c61:/#
```

Fig-4: Display the file list and make a backup of the files for the replica node.

Explanation: In the docker container, we can see the files related to the postgres configurations. We are taking the backup of the respective files using pg_basebackup command.

Command:

```
cp -r ./backup/ /var/lib/postgresql/data
```

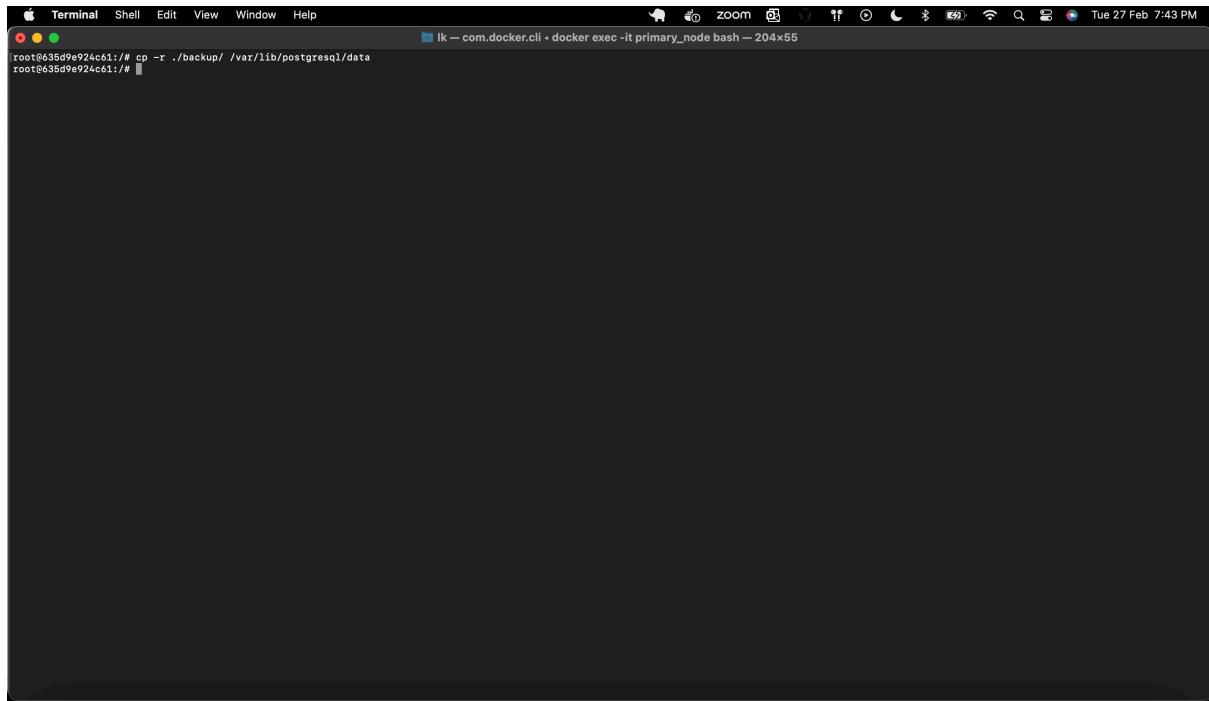


Fig-5: Copy recursively into ‘var/lib/postgresql/data’ folder.

Explanation: Copying the data recursively into the postgres/data folder.

Command:

```
psql -U postgres  
\l -- To fetch list of files
```

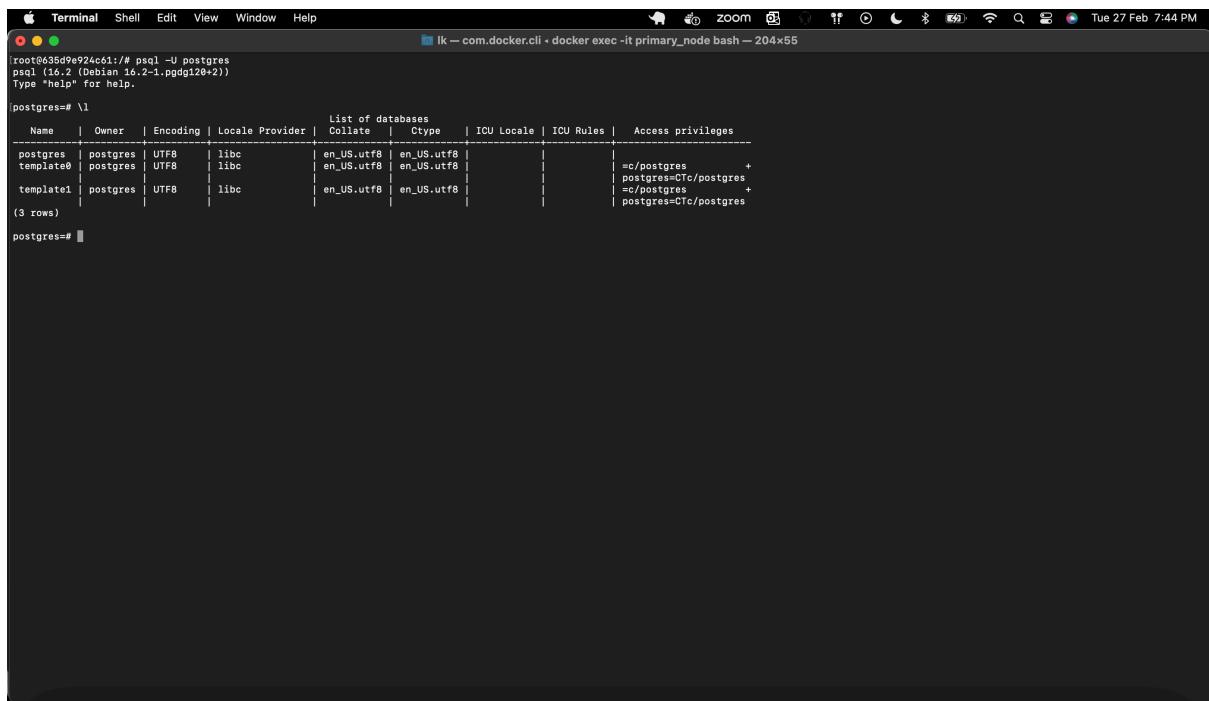


Fig-6: Connect to the postgres from docker container and display list of databases

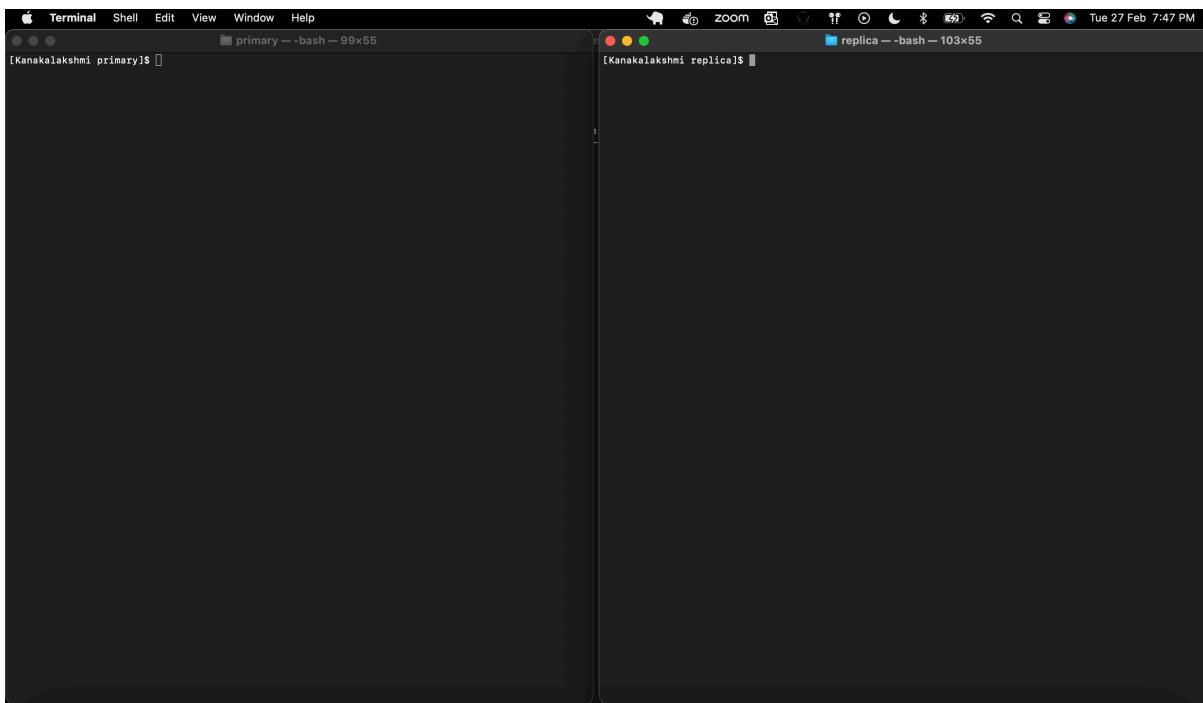


Fig-7: Open the two terminals from system, where primary and replica folders are created.

Explanation: Now open the two terminals which are created in the local system, as we initially created while pulling the postgres image. These two folders are created in data_base_volume_mounts.

Command:

\l -- To fetch list of files

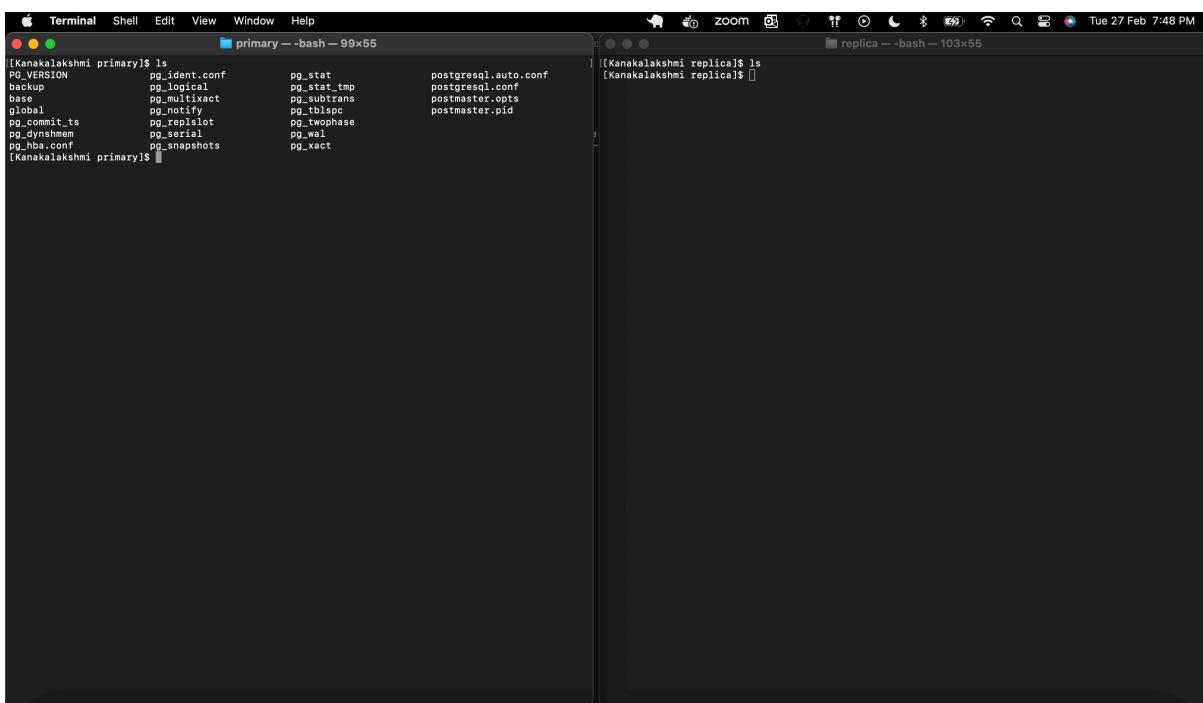


Fig-8: Display the list of files in primary folder.

Command:

```
cp -r ./backup/ ../replica/  
ls
```

The screenshot shows two terminal windows side-by-side. The left window, titled 'primary -- bash - 99x55', displays the command 'cp -r ./backup/ ../replica/'. The right window, titled 'replica -- bash - 103x55', shows the result of the 'ls' command after the copy operation, listing files such as pg_hba.conf, pg_ident.conf, pg_stat, pg_subtrans, pg_twophase, pg_wal, pg_xact, pg_dyncmmem, pg_resslot, pg_serial, and pg_wai.

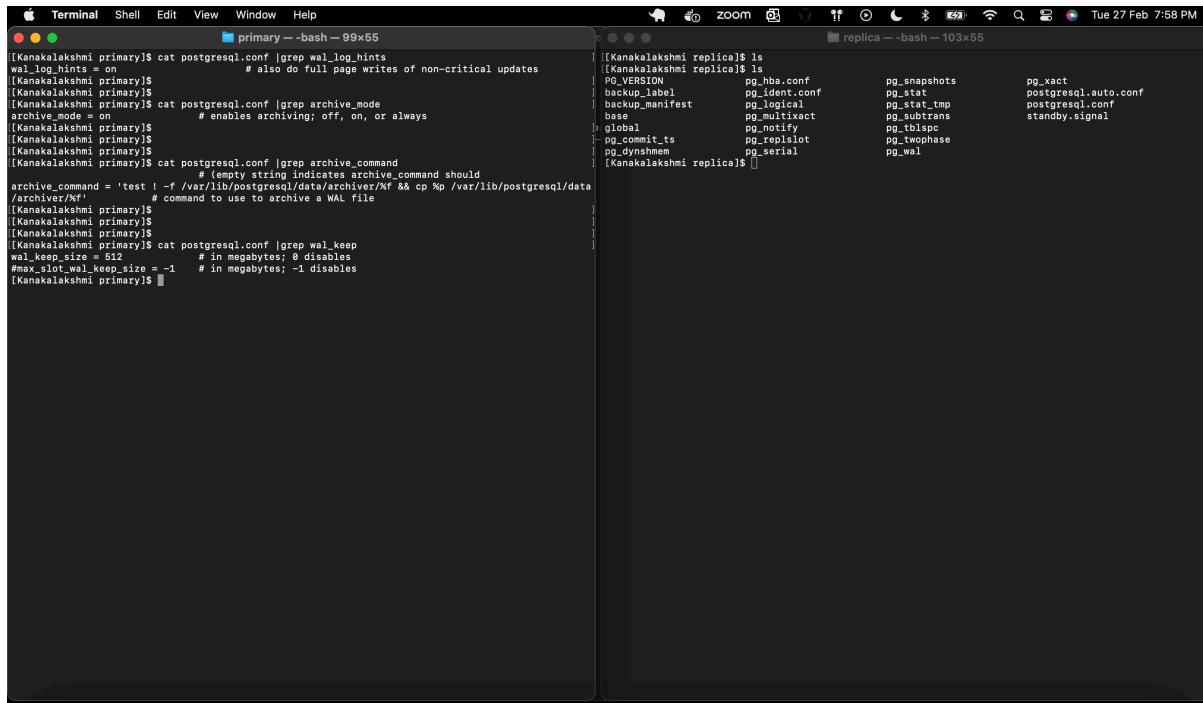
```
[Kanakalakshmi primary]$ ls  
PG_VERSION pg_ident.conf pg_stat postgresql.auto.conf  
backup pg_logical pg_stat_tmp postgresql.conf  
base pg_multixact pg_subtrans postmaster.opts  
global pg_notify pg_tblspc postmaster.pid  
pg_commit_ts pg_repslot pg_twophase  
pg_dynshmem pg_serial pg_wal  
pg_hba.conf pg_standby pg_xact  
pg_snapshots pg_twophase  
[Kanakalakshmi primary]$ cp -r ./backup/ ../replica/  
[Kanakalakshmi primary]$ ls  
[Kanakalakshmi replica]$ ls  
PG_VERSION pg_hba.conf pg_snapshots pg_xact  
backup_label pg_ident.conf pg_stat postgresql.conf  
backup_manifest pg_logical pg_stat_tmp postgresql.conf  
base pg_multixact pg_subtrans standby.signal  
global pg_notify pg_tblspc pg_twophase  
pg_commit_ts pg_resslot pg_wal  
pg_dyncmmem pg_serial pg_wai  
[Kanakalakshmi replica]$
```

Fig-9: Copy to replica folder from primary folder and display the list of files in replica folder.

Explanation: As we have the files in the primary_node, copying recursively to the replica_node to have the same configurations. In the replica_node the standby.signal file plays the vital role in communication between the primary_node and replica_node.

Command:

```
vi postgresql.conf - wal_log_hints = on, archive_mode = on  
archive_command = 'test ! -f /var/lib/postgresql/data/archiver/%f &&  
/var/lib/postgresql/data/archiver/%f', wal_keep_size = 512
```

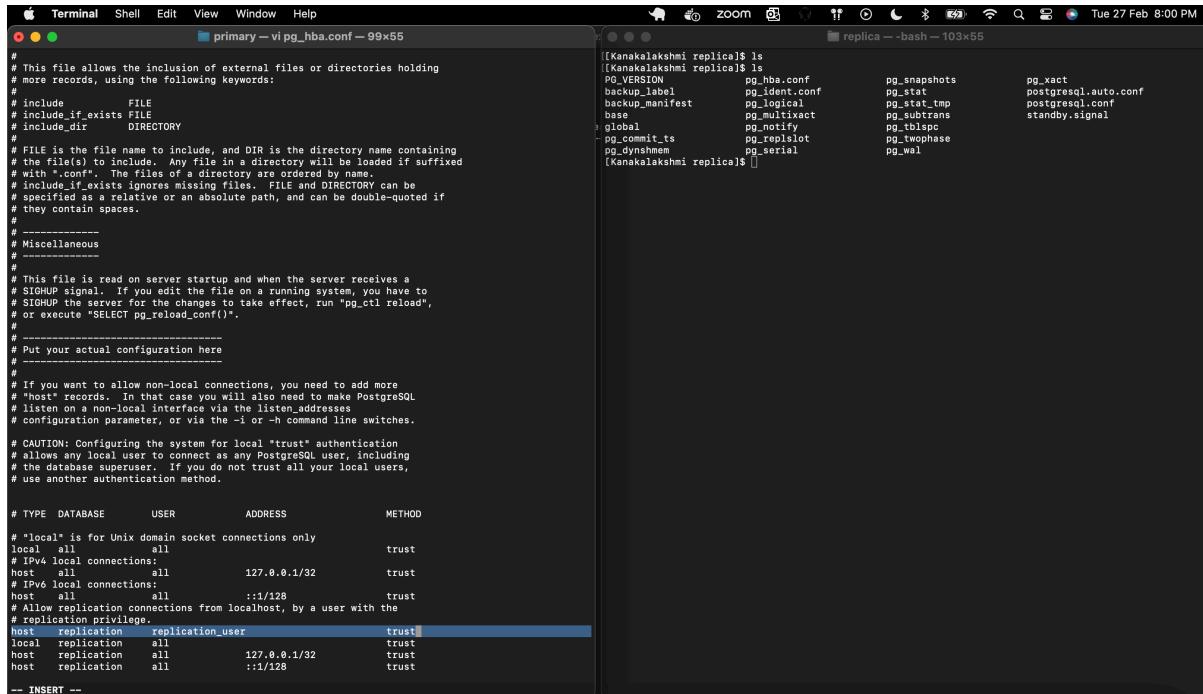


```
[Kanakalakshmi primary]$ cat postgresql.conf |grep wal_log_hints  
wal_log_hints = on          # also do full page writes of non-critical updates  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$ cat postgresql.conf |grep archive_mode  
archive_mode = on           # enables archiving; off, on, or always  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$ cat postgresql.conf |grep archive_command  
archive_command = 'test ! -f /var/lib/postgresql/data/archiver/%f && cp %p /var/lib/postgresql/data/  
/archiver/%f'                 # command to use to archive a WAL file  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$  
[Kanakalakshmi primary]$ cat postgresql.conf |grep wal_keep_size  
wal_keep_size = 512          # in megabytes; 0 disables  
#max_slot_wal_keep_size = -1 # in megabytes; -1 disables  
[Kanakalakshmi primary]$  
  
[[Kanakalakshmi replica]$ ls  
pg_hba.conf pg_snapshots pg_xact  
pg_ident.conf pg_stat pg_stat_tmp postgresql.auto.conf  
backup_manifest pg_logical pg_subtrans standby.signal  
base pg_multixact pg_twophase  
global pg_notify pg_tlsync  
pg_commit_ts pg_replslot pg_dynshmem pg_serial pg_wal  
pg_dynshmem pg_replslot pg_twophase  
pg_notify pg_tlsync pg_wal
```

Fig-10: update the configurations in postgresql.conf file in primary folder.

Command:

```
vi pg_hba.conf - host replication replication_user trust
```



```
[Kanakalakshmi primary]$ vi pg_hba.conf  
# This file allows the inclusion of external files or directories holding  
# more records, using the following keywords:  
#  
# include FILE  
# include_if_exists FILE  
# include_dir DIRECTORY  
#  
# FILE is the file name to include, and DIR is the directory name containing  
# the file(s) to include. Any file in a directory will be loaded if suffixed  
# with ".conf". The files of a directory are ordered by name.  
# include_if_exists ignores missing files. FILE and DIRECTORY can be  
# specified as a relative or an absolute path, and can be double-quoted if  
# they contain spaces.  
#  
# -----  
# Miscellaneous  
# -----  
#  
# This file is read on server startup and when the server receives a  
# SIGHUP signal. If you edit the file on a running system, you have to  
# SIGHUP the server for the changes to take effect, run "pg_ctl reload",  
# or execute "SELECT pg_reload_conf()".  
#  
# -----  
# Put your actual configuration here  
#  
# If you want to allow non-local connections, you need to add more  
# "host" records. In that case you will also need to make PostgreSQL  
# listen on the non-local interface via the listen_addresses  
# configuration parameter, or via the -l or -h command line switches.  
# CAUTION: Configuring the system for local "trust" authentication  
# allows any local user to connect as any PostgreSQL user, including  
# the database superuser. If you do not trust all your local users,  
# use another authentication method.  
  
# TYPE DATABASE USER ADDRESS METHOD  
  
# "local" is for Unix domain socket connections only  
local  all   all   trust  
# IPv4 local connections:  
host   all   all   127.0.0.1/32 trust  
# IPv6 local connections:  
host   all   all   ::1/128 trust  
# Allow replication connections from localhost, by a user with the  
# replication privilege.  
host   replication replication_user trust  
local  replication all   trust  
host   replication all   127.0.0.1/32 trust  
host   replication all   ::1/128 trust  
  
-- INSERT --
```

Fig-11: update the configuration in pg_hba.conf file in primary folder.

```
[Kanakalakshmi primary]$ cat pg_hba.conf |grep replication
# DATABASE "all" includes all databases, "replication", a
# "replication" user. Access to replication must be enabled in a separate
# "all" "sameuser", "samerole" or "replication" makes the name lose
# Allow replication connections from localhost, by a user with the
# replication privilege.
host    replication    replication_user          trust
local   replication    all                     trust
host    replication    all                     127.0.0.1/32  trust
host    replication    all                     ::/128      trust
[Kanakalakshmi primary]$ ls
[Kanakalakshmi replica]$ ls
pg_VERSION          pg_hba.conf           pg_snapshots      pg_xact
pg_ident.conf       pg_logical.conf     pg_stat            postgresql.auto.conf
backup_label        pg_manifest         pg_stat_tmp      postgresql.conf
base                pg_multixact        pg_subtrans      standby.signal
global              pg_notify           pg_replslot      pg_twophase
pg_commit_ts        pg_timezone_mon  pg_serial        pg_wal
pg_dynshmem         pg_timezone_mon
```

Fig-12: verify the updated configurations in primary folder.

Command:

```
docker inspect primary -- get the IP address of postgres docker container from terminal
```

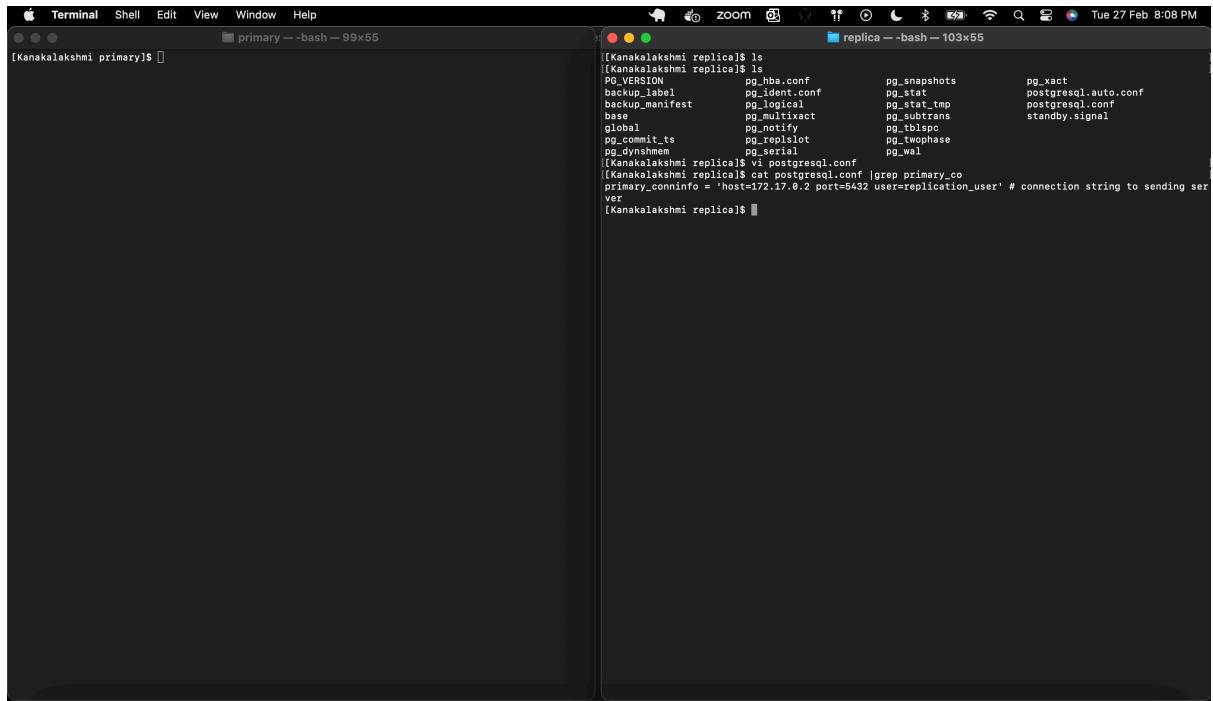
```
[Kanakalakshmi ~]$ docker inspect primary_node
[{"Id": "beccad7ad1f8161cebe5a1e41d81a32dbf71a32fea09b247e8783f7c0438ba68", "Ip": "172.17.0.2", "GlobalIPv4Address": "", "GlobalIPv6Address": "", "MacAddress": "02:42:ac:11:00:02", "Networks": {"bridge": {"IPAMConfig": null, "Links": null, "Aliases": null, "EndpointID": "beccad7ad1f8161cebe5a1e41d81a32dbf71a32fea09b247e8783f7c0438ba68", "Gateway": "172.17.0.1", "IPAddress": "172.17.0.2", "IPPrefixLen": 16, "IPRange": "172.17.0.1-172.17.0.2", "IPv6Gateway": "", "LinkLocalIPv4Address": "", "LinkLocalIPv6Address": "", "NetworkID": "6858c22d4debb91acacf3e1cf1b98e52d71e9a39451afba3723f6f9be89ba282", "EndpointID": "beccad7ad1f8161cebe5a1e41d81a32dbf71a32fea09b247e8783f7c0438ba68", "Gateway": "172.17.0.1", "IPAddress": "172.17.0.2", "IPPrefixLen": 16, "IPRange": "172.17.0.1-172.17.0.2", "IPv6Gateway": "", "LinkLocalIPv4Address": "", "LinkLocalIPv6Address": "", "DriverOpts": null, "DNSNames": null}}}
```

Fig-13: Fetch the IP of primary_node docker container from terminal.

Explanation: Fetch the IP address of the primary_node using above to update in the replica_node.

Command:

```
vi postgresql.conf - primary_conninfo = 'host=172.17.0.2 port=5432  
user=replication_user'
```

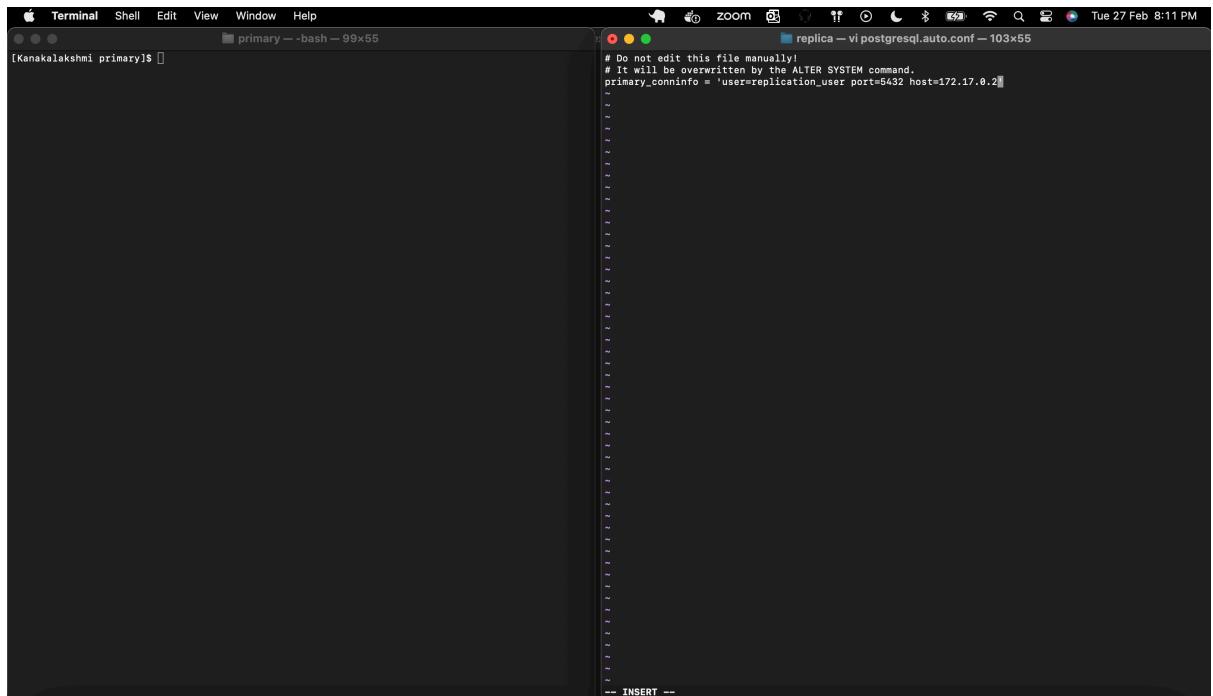


```
[Kanakalakshmi primary]$ ls  
[Kanakalakshmi replica]$ ls  
PG_VERSION pg_hba.conf pg_snapshots pg_xact  
backup_label pg_ident.conf pg_stat postgresql.auto.conf  
backup_manifest pg_logical pg_stat_tmp postgresql.conf  
base pg_notify pg_subtrans standby.signal  
dburl pg_replslot pg_twophase  
pg_commit_ts pg_serial pg_wal  
pg_dynshmem  
[Kanakalakshmi replica]$ vi postgresql.conf  
[Kanakalakshmi replica]$ cat postgresql.conf |grep primary_conninfo = 'host=172.17.0.2 port=5432 user=replication_user' # connection string to sending server  
[Kanakalakshmi replica]$
```

Fig-14: update the primary_conninfo in postgresql.conf file at replica folder.

Command:

```
vi postgresql.auto.conf  
primary_conninfo = 'user=replication_user port=5432 host=172.17.0.2'
```

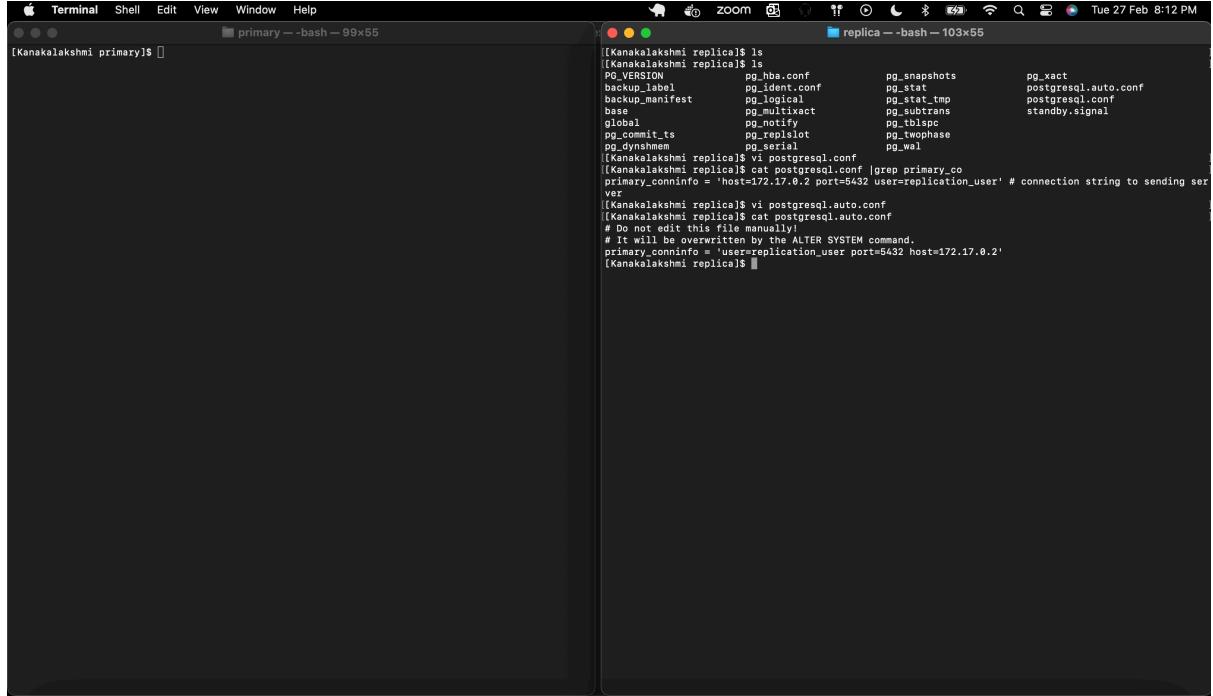


```
# Do not edit this file manually!  
# It will be overwritten by the ALTER SYSTEM command.  
primary_conninfo = "user=replication_user port=5432 host=172.17.0.2"  
-- INSERT --
```

Fig-15: Update configurations in postgresql.auto.conf file at replica folder.

Command:

```
Cat postgresql.auto.conf
```



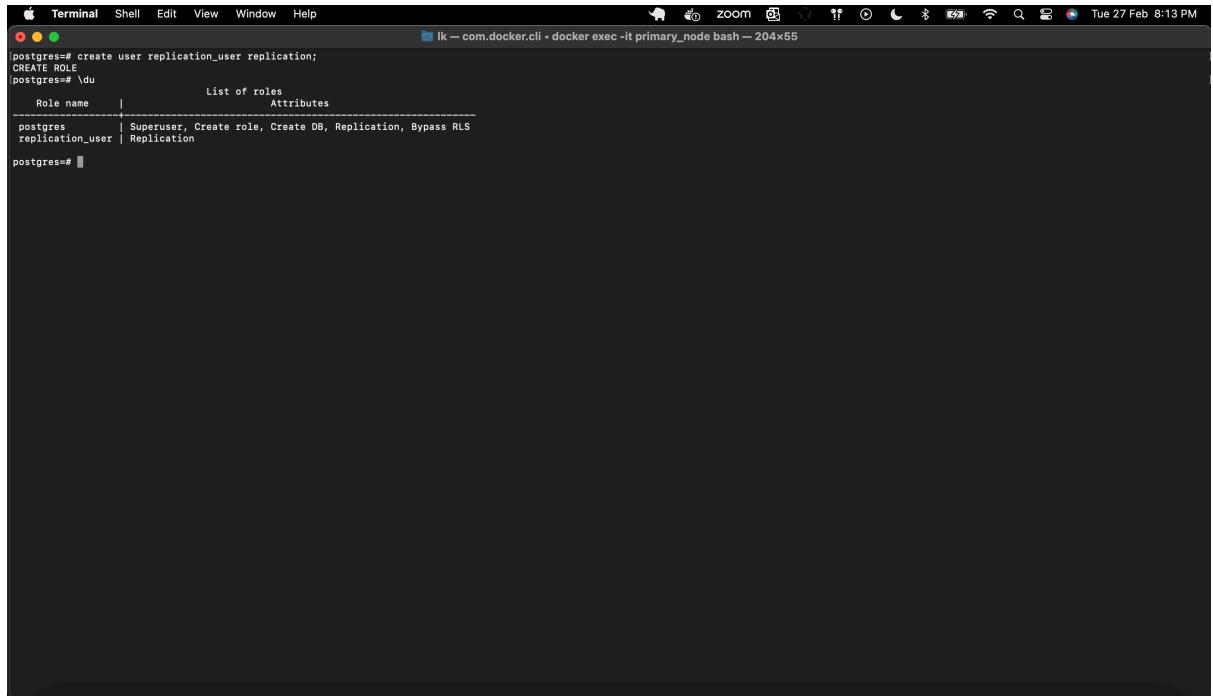
The screenshot shows two terminal windows side-by-side. The left window, titled 'primary', displays the command 'cat postgresql.auto.conf' being run, followed by its contents which include connection settings for a replication user. The right window, titled 'replica', shows the command 'ls' being run in the PostgreSQL configuration directory, listing various configuration files like pg_hba.conf, pg_ident.conf, pg_logical.conf, etc.

```
[Kanakalakshmi primary]$ cat postgresql.auto.conf
[host=172.17.0.2 port=5432 user=replication_user] # connection string to sending server
[host=172.17.0.2 port=5432 user=replication_user] # It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'host=172.17.0.2 port=5432 user=replication_user'
[Kanakalakshmi replica]$ ls
pg_hba.conf      pg_snapshots      pg_xact
pg_ident.conf   pg_stat            postgresql.conf
pg_logical.conf pg_stat_tmp       postgresql.auto.conf
pg_multixact    pg_subtrans      standby.signal
pg_notify        pg_tblspc
pg_commit_ts     pg_replslot      pg_twophase
pg_timezone.dat  pg_semanew
[Kanakalakshmi replica]$ vi postgresql.conf
[Kanakalakshmi replica]$ cat postgresql.conf | grep primary_co
primary_conninfo = 'host=172.17.0.2 port=5432 user=replication_user' # connection string to sending server
[host=172.17.0.2 port=5432 user=replication_user] # It will be overwritten by the ALTER SYSTEM command.
primary_conninfo = 'user=replication_user port=5432 host=172.17.0.2'
[Kanakalakshmi replica]$
```

Fig-16: verify postgresql.auto.conf configurations.

Command:

```
create user replication_user replication;
\du
```



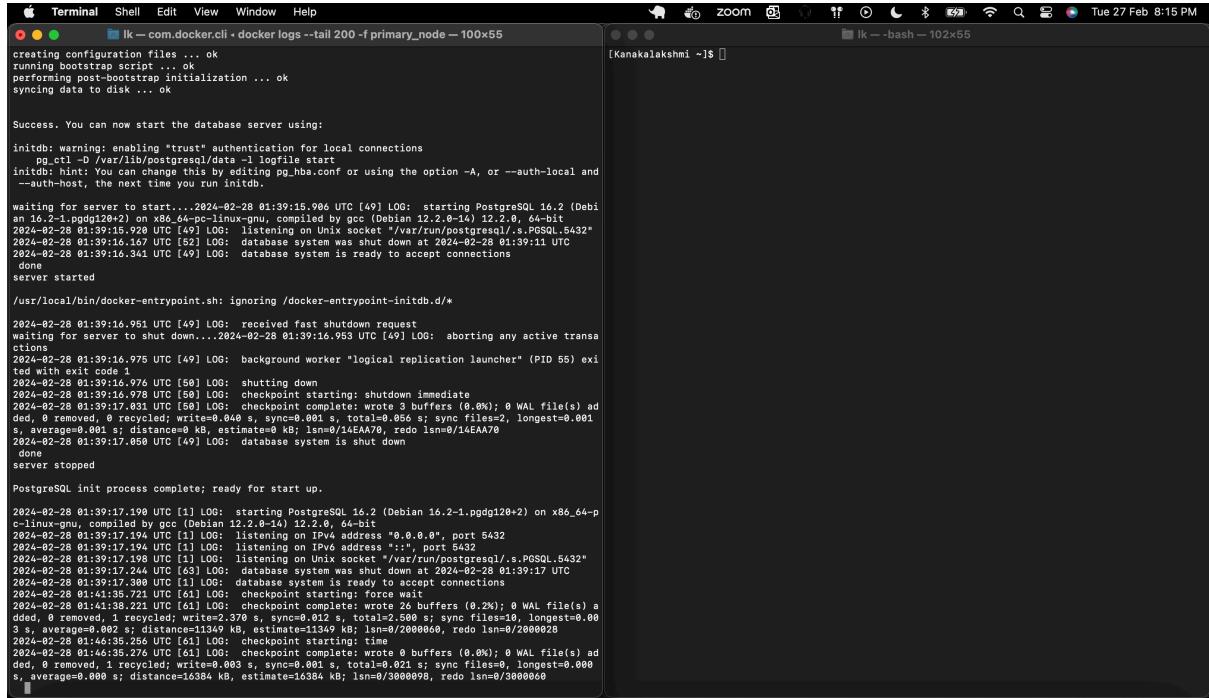
The screenshot shows a single terminal window titled 'lk'. It displays the creation of a new PostgreSQL user named 'replication_user' with the 'replication' privilege. The command '\du' is then run to list all roles, showing the newly created 'replication_user' role.

```
postgres=# create user replication_user replication;
CREATE ROLE
postgres=# \du
              List of roles
Role name | Attributes
-----+-----
postgres  | Superuser, Create role, Create DB, Replication, Bypass RLS
replication_user | Replication
postgres#
```

Fig-17: create replication_user in primary_node

Command:

```
docker logs --tail 200 -f primary_node
```



```
Ik - com.docker.cli - docker logs --tail 200 -f primary_node - 100x55
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok

Success. You can now start the database server using:

initdb: warning: enabling "trust" authentication for local connections
pg_ctl -D "/var/lib/postgresql/data" -l logfile start
initdb: hint: You can change this by editing pg_hba.conf or using the option -A, or --auth-local and
--auth-host, the next time you run initdb

waiting for server to start...2024-02-28 01:39:16.996 UTC [49] LOG:  starting PostgreSQL 16.2 (Debian 16.2-1.pgdg120+2) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2024-02-28 01:39:16.998 UTC [49] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-02-28 01:39:16.167 UTC [52] LOG:  database system was shut down at 2024-02-28 01:39:11 UTC
2024-02-28 01:39:16.341 UTC [49] LOG:  database system is ready to accept connections
done
server started

/usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
2024-02-28 01:39:16.951 UTC [49] LOG:  received fast shutdown request
waiting for server to shut down...2024-02-28 01:39:16.983 UTC [49] LOG:  aborting any active transactions
2024-02-28 01:39:16.975 UTC [49] LOG:  background worker "logical replication launcher" (PID 55) exited with exit code 1
2024-02-28 01:39:16.976 UTC [50] LOG:  shutting down
2024-02-28 01:39:16.978 UTC [50] LOG:  checkpoint starting: shutdown immediate
2024-02-28 01:39:17.031 UTC [50] LOG:  checkpoint complete: wrote 3 buffers (0.0%) ; 0 WAL file(s) added, 0 removed, 0 recycled; write=0.040 s, sync=0.001 s, total=0.056 s; sync files=2, longest=0.001 s, average=0.001 s; distance=0 kB, estimated=0 kB; lsn=0/14EAAT0, redo lsn=0/14EAAT0
2024-02-28 01:39:17.050 UTC [49] LOG:  database system is shut down
done
server stopped

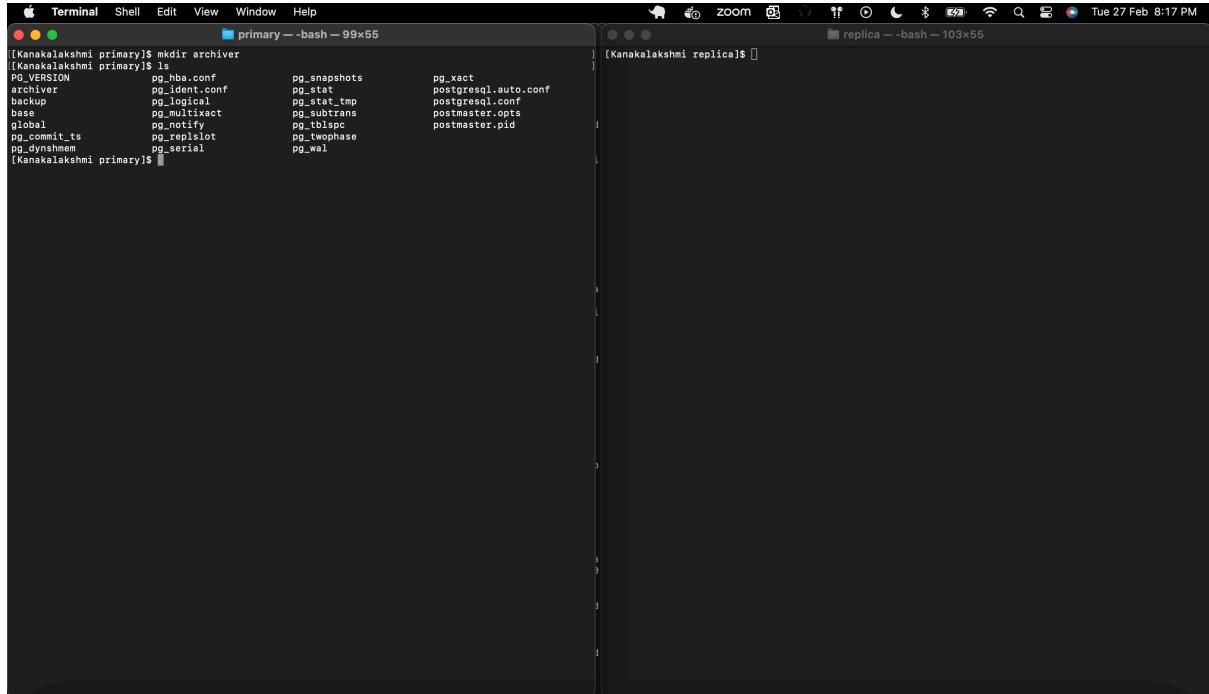
PostgreSQL init process complete; ready for start up.

2024-02-28 01:39:17.194 UTC [1] LOG:  starting PostgreSQL 16.2 (Debian 16.2-1.pgdg120+2) on x86_64-pc-linux-gnu, compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
2024-02-28 01:39:17.194 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2024-02-28 01:39:17.194 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2024-02-28 01:39:17.198 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
2024-02-28 01:39:17.244 UTC [63] LOG:  database system was shut down at 2024-02-28 01:39:17 UTC
2024-02-28 01:39:17.245 UTC [63] LOG:  database system is ready to accept connections
2024-02-28 01:41:35.721 UTC [61] LOG:  checkpoint starting: force
2024-02-28 01:41:38.221 UTC [61] LOG:  checkpoint complete: wrote 26 buffers (0.2K); 0 WAL file(s) added, 0 removed, 1 recycled; write=2.378 s, sync=0.012 s, total=2.590 s; sync files=10, longest=0.003 s, average=0.002 s; distance=11349 kB, estimated=11349 kB; lsn=0/2000068, redo lsn=0/2000028
2024-02-28 01:41:38.256 UTC [61] LOG:  checkpoint starting: time
2024-02-28 01:41:38.276 UTC [61] LOG:  checkpoint complete: wrote 0 buffers (0.0K); 0 WAL file(s) added, 0 removed, 1 recycled; write=0.003 s, sync=0.004 s, total=0.021 s; sync files=0, longest=0.000 s, average=0.000 s; distance=16384 kB, estimated=16384 kB; lsn=0/3000098, redo lsn=0/3000060
```

Fig-18: verify logs in primary_node.

Command:

```
mkdir archiver
ls
```

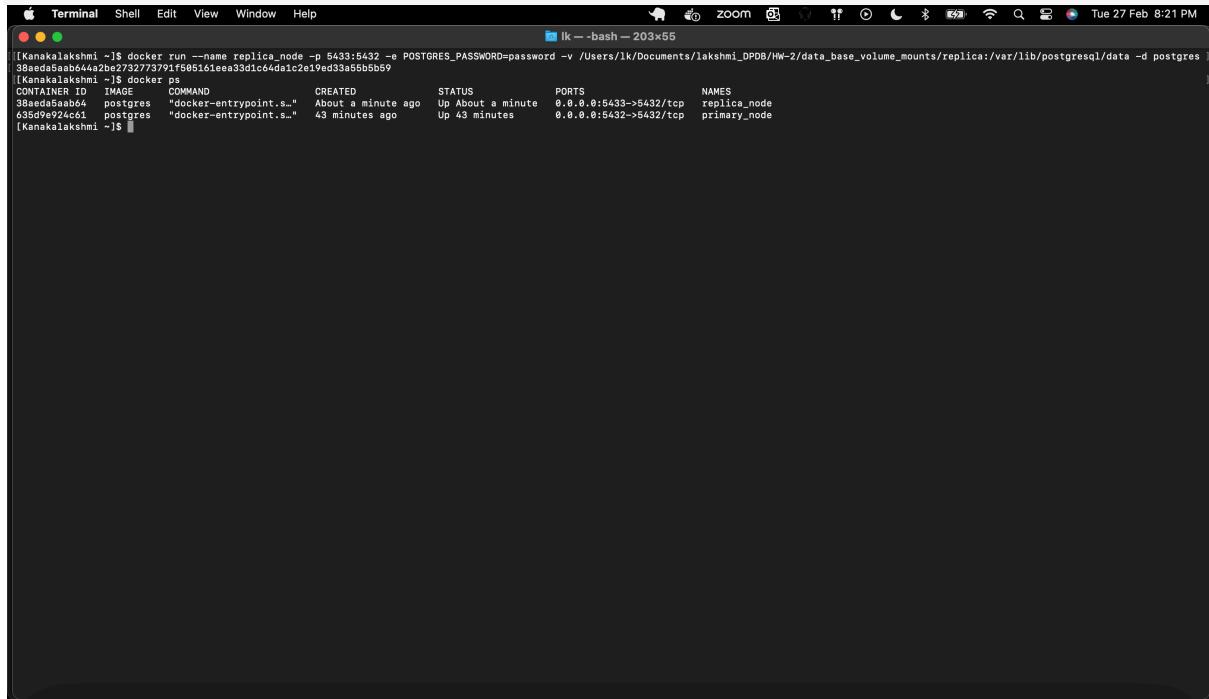


```
[Kanakalakshmi primary]$ mkdir archiver
[Kanakalakshmi primary]$ ls
pg_hba.conf      pg_snapshots    pg_xact
pg_ident.conf   pg_stat          pg_stat_tmp
pg_logical       pg_subtrans     postgresql.auto.conf
pg_multixact    pg_tblspc      postresql.conf
base             pg_twophase    postmaster.opts
global           pg_notify       postmaster.pid
pg_commit_ts    pg_replslot
pg_dynshmem      pg_serial
pg_wal
```

Fig-19: create archiver folder in system primary folder.

Command:

```
docker run --name replica_node -p 5433:5432 -e POSTGRES_PASSWORD=password -v /Users/lk/Documents/lakshmi_DPDB/HW-2/data_base_volume_mounts/replica:/var/lib/postgresql/data -d postgres
```

A screenshot of a macOS Terminal window titled "Terminal". The command "docker run" was run to create a Docker container named "replica_node". The output shows the container ID, image name ("postgres"), command ("docker-entrypoint.s..."), creation time ("About a minute ago" or "43 minutes ago"), status ("Up" or "Up 43 minutes"), ports mapping (5433 to 5432), and names ("replica_node" or "primary_node").

```
[Kanakalakshmi ~]$ docker run --name replica_node -p 5433:5432 -e POSTGRES_PASSWORD=password -v /Users/lk/Documents/lakshmi_DPDB/HW-2/data_base_volume_mounts/replica:/var/lib/postgresql/data -d postgres
38aed0aab644a2be273277371f585161eeaa33d1c64da1c2e19ed33a5bb5b59
(Kanakalakshmi ~)$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
38aed0aab644 postgres "docker-entrypoint.s..." About a minute ago Up About a minute 0.0.0.0:5433->5432/tcp replica_node
635de924c61 postgres "docker-entrypoint.s..." 43 minutes ago Up 43 minutes 0.0.0.0:5432->5432/tcp primary_node
[Kanakalakshmi ~]$
```

Fig-20: Pull the postgres docker image, used as postgres replica_node.

Explanation: Pull the postgres docker image using the configurations in the replica folder provide the details to create docker container such as name, port_no and password. Once image is pulled, we can see the status of the replica_node.

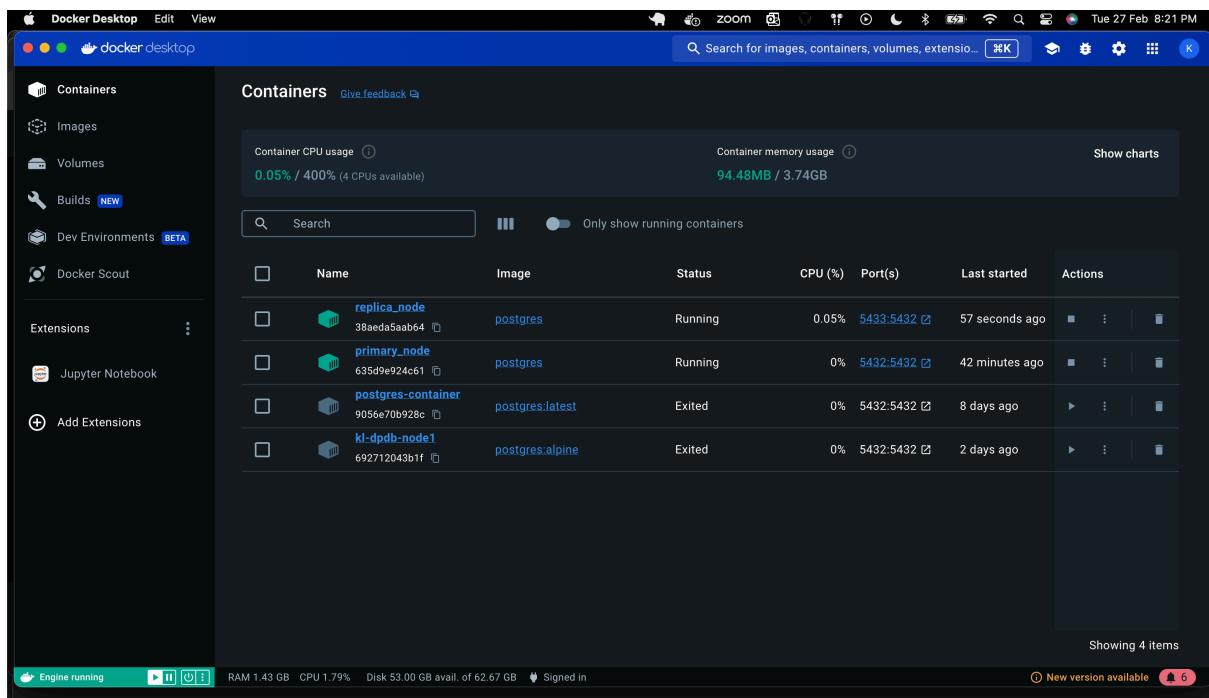


Fig-21: verify two postgres instances in docker dashboard.

Command:

```
docker logs --tail 200 -f replica_node
```

```
Ik - com.docker.cli - docker logs --tail 200 -f replica_node - 203x55
2024-02-28 02:21:33.526 UTC [45] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:33.526 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:21:38.532 UTC [46] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:38.536 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:21:43.538 UTC [47] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:43.538 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:21:48.589 UTC [48] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:48.593 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:21:53.556 UTC [49] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:53.556 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:21:58.498 UTC [60] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:21:58.704 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:03.594 UTC [61] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:03.594 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:08.555 UTC [62] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:08.558 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:13.597 UTC [63] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:13.597 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:18.753 UTC [64] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:18.753 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:23.638 UTC [65] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:23.638 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:28.704 UTC [66] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:28.707 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:33.632 UTC [67] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:33.632 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:38.753 UTC [68] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:38.753 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:43.634 UTC [69] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:43.634 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:48.644 UTC [70] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:48.644 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:53.643 UTC [61] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:53.643 UTC [29] LOG: waiting for WAL to become available at 0/3000018
2024-02-28 02:22:58.753 UTC [62] FATAL: could not connect to the primary server: connection to server at "172.17.0.2", port 5432 failed: FATAL: no pg_hba.conf entry for replication connection from host
"172.17.0.3", user "replication_user", no encryption
2024-02-28 02:22:58.753 UTC [29] LOG: waiting for WAL to become available at 0/3000018
```

Fig-22: check the logs for replica node.

Command:

```
docker inspect replica_node
```

```
Ik - bash - 203x55
{
  "Id": "Karnakalakshmi",
  "Created": "2024-02-28T08:55:20.000Z",
  "Path": "/etc/docker/entrypoint.sh",
  "Args": [],
  "State": {
    "Status": "running",
    "RunningSince": "2024-02-28T08:55:20.000Z",
    "StartedAt": "2024-02-28T08:55:20.000Z",
    "Health": "healthy"
  },
  "Image": "4165170edc921c9ee5d48a9152661e022433ceb24176dfd7883c8dc63921c60",
  "Labels": {},
  "LabelsRaw": {},
  "Mounts": [
    {
      "Type": "bind",
      "Source": "/var/run/docker/netns/4165170edc921c60",
      "Destination": "/etc/docker/entrypoint.sh"
    }
  ],
  "Config": {
    "WorkingDir": "",
    "Entrypoint": [
      "docker-entrypoint.sh"
    ],
    "OnBuild": null,
    "Labels": {},
    "StopSignal": "SIGINT"
  },
  "NetworkSettings": {
    "Bridge": "bridge",
    "SandboxID": "4165170edc921c9ee5d48a9152661e022433ceb24176dfd7883c8dc63921c60",
    "GlobalIPv6Address": null,
    "GlobalIPv6PrefixLen": 0,
    "LinkLocalIPv6Address": null,
    "LinkLocalIPv6PrefixLen": 0,
    "MacAddress": null,
    "EndpointID": "45fd542f9d5bac82c347971e365122bf0ba67a1a361e80f4a4f24e1043a7fa1",
    "Gateway": "172.17.0.1",
    "GlobalIPv4Address": null,
    "GlobalIPv4PrefixLen": 0,
    "IPAddress": "172.17.0.3",
    "IPPrefixLen": 16,
    "IPv6Gateway": null,
    "IPv6PrefixLen": 0,
    "IPv6Gateway": null,
    "MacAddress": "02:42:ac:11:00:03",
    "Networks": [
      {
        "bridge": "bridge",
        "GlobalIPv6Address": null,
        "GlobalIPv6PrefixLen": 0,
        "LinkLocalIPv6Address": null,
        "LinkLocalIPv6PrefixLen": 0,
        "Aliases": null,
        "MacAddress": "02:42:ac:11:00:03",
        "NetworkID": "485fc022d6eb019acac3e1cf1b08652d71e9a38451af843723f6f9be80ba202",
        "EndpointID": "45fd542f9d5bac82c347971e365122bf0ba67a1a361e80f4a4f24e1043a7fa1",
        "Gateway": "172.17.0.1",
        "GlobalIPv4Address": "172.17.0.3",
        "IPPrefixLen": 16,
        "IPv6Gateway": null,
        "GlobalIPv6PrefixLen": 0,
        "DriverOpts": null,
        "DNSNames": null
      }
    ]
  }
}
[Karnakalakshmi - ]$
```

Fig-23: Get the IP of the replica node.

Explanation: Get the IP address of the replica_node to update respective details in the primary_node.

Command:

```
vi pg_hba.conf
host replication replication_user 172.17.0.3/32 trust
cat pg_hba.conf
```

Fig-24: update replica_node IP in primary_node pg_hba.conf file.

Command:

```
docker restart primary_node
docker logs --tail 200 -f primary_node
```

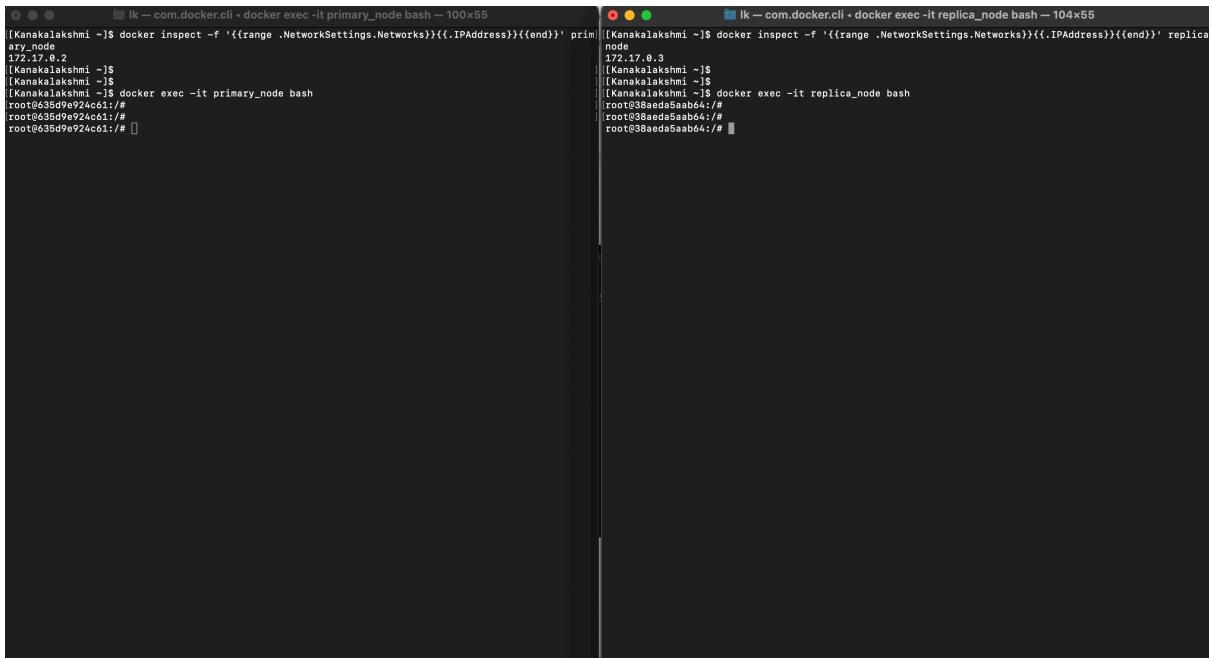
Fig-25: Restart the primary_node and verify the logs.

Now two postgres docker containers are ready with all configurations, we can start using them.

Command:

```
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' primary_node
docker exec -it primary_node bash

docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' replica_node
docker exec -it replica_node bash
```



The image shows two terminal windows side-by-side. The left window is titled 'Ik - com.docker.cli - docker exec -it primary_node bash - 100x55' and the right window is titled 'Ik - com.docker.cli - docker exec -it replica_node bash - 104x55'. Both windows show the command 'docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' being run on their respective nodes. The output for the primary node shows an IP address of 172.17.0.2, and the output for the replica node shows an IP address of 172.17.0.3. Below these outputs, both windows show the command 'docker exec -it [node_name] bash' being run, and the root prompt '#>' is visible in both.

Fig-26: check the IP and execute primary and replica nodes.

Explanation: These are the two postgres docker containers. One is the **primary_node with 172.17.0.2** and the other is **replica_node with 172.17.0.3**. Executing the run command on both the nodes.

Now we are connected to two different postgres instances.