

PostgreSQL Cluster Deployment Guide

Version 1.0.1

Contents

Copyright Notice	3
Document Revision History	4
OVA Download	5
OVA Deployment	6
Preparations	6
Network	7
Port Usage	7
System Requirements	8
Supported Platforms	8
Cluster Size	8
Virtual Machine Configuration	8
Deploying	9
Cluster Setup	10
Requirements	10
Server Stats	11
Setup	12
Master Server	12
Create Replication User	14
Replica Server	15
Copy data directory from Master server	16
Setup recovery.conf on Replica server	18
Verification	23
Usage	24
Create Database	24
Connect to Database	25

Copyright Notice

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without express written permission. Under the law, reproducing includes translating into another language or format.

The software is protected by United States copyright laws and international treaty provision. Therefore, you must treat the software like any other copyrighted material (e.g. a book or sound recording).

Document Revision History

October 8, 2018

- Initial release of documentation

OVA Download

The latest OVA file is available as a secure download hosted on Amazon S3.

Your professional services representative will provide you with a secure link to download the file when it becomes available.

OVA Deployment

Preparations

To set up PostgreSQL cluster, you must have:

- PostgreSQL OVA
- Supported virtual infrastructure

OVA Deployment

Network

Port Usage

Protocol	Port	Direction	Purpose
TCP	5432	Inbound/Outbound	PostgreSQL Server
HTTP	80	Outbound	Server Statistics Page
SSH	22	Inbound/Outbound	Cluster administration

OVA Deployment

System Requirements

Supported Platforms

VMware ESXI 5.5 and later are supported.

Cluster Size

The recommended size of a PostgreSQL cluster is 2 nodes.

Virtual Machine Configuration

The minimum requirements for a Backpack node are:

CPU: 3 GHz dual core or 4 virtual processors

RAM: 8GB

STORAGE: 80GB

The recommended requirements for a Backpack node are:

CPU: 3 GHz quad core or 8 virtual processors

RAM: 12 GB

STORAGE: 120GB, low-latency SATA or SSD drives

PostgreSQL Cluster Deployment

Deploying

Deploy the OVA on your platform as you would any other OVA. Refer to your platform's documentation for instructions on deploying OVA files.

Cluster Setup

Clusters are headless and they are setup in master-replica mode where one of the node serves as a master node and others serves as a hot-standby replica which can be promoted to become the master node.

Requirements

Inorder to setup the PostgreSQL cluster the following are required,

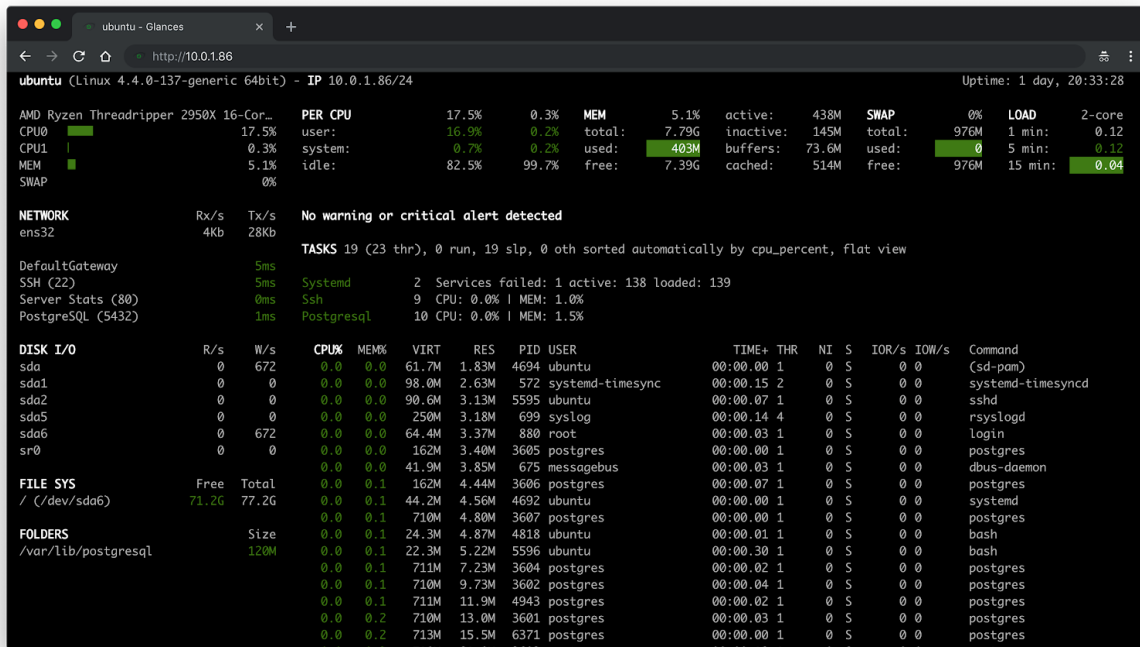
- IP address of the **master** node, referred as **MASTER_IP**
- IP address of the **replica** node, referred as **REPLICA_IP**

Both the master and replica nodes should be visible to each other.

Server Stats

Open the IP addresses in the web browser to view the server stats page,

For example if the IP address of the master server is **10.0.1.86**,



Setup

Master Server

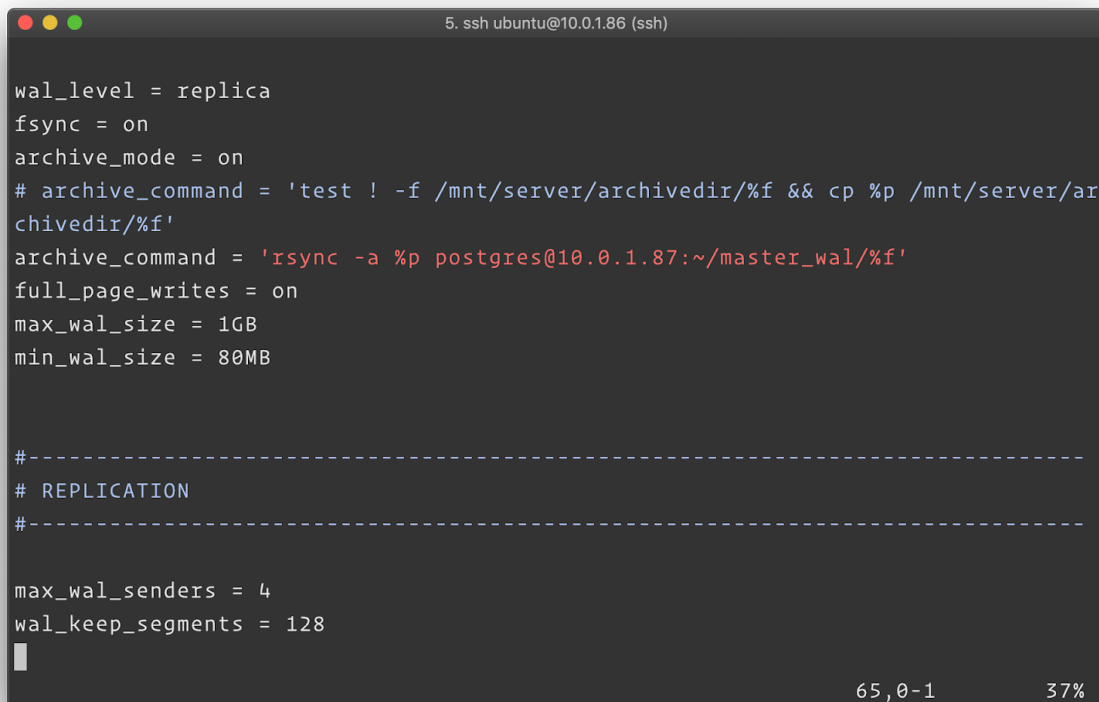
Open the **postgresql.conf** on the master node with the following command,

```
# Open the postgresql.conf
sudo vim /etc/postgresql/11/main/postgresql.conf
```

Go to **archive_command** on Line 53 and replace the **REPLICA_IP_ADDRESS** with the IP address of the replica server.

For example if the IP address of the replica server is **10.0.1.87**, the archive_command would be,

```
wal_keep_segments = 0
archive_command = 'rsync -a %p postgres@10.0.1.87:~/master_wal/%f'
```



```
5. ssh ubuntu@10.0.1.86 (ssh)

wal_level = replica
fsync = on
archive_mode = on
# archive_command = 'test ! -f /mnt/server/archivedir/%f && cp %p /mnt/server/ar
chivedir/%f'
archive_command = 'rsync -a %p postgres@10.0.1.87:~/master_wal/%f'
full_page_writes = on
max_wal_size = 1GB
min_wal_size = 80MB

#-----
# REPLICATION
#-----

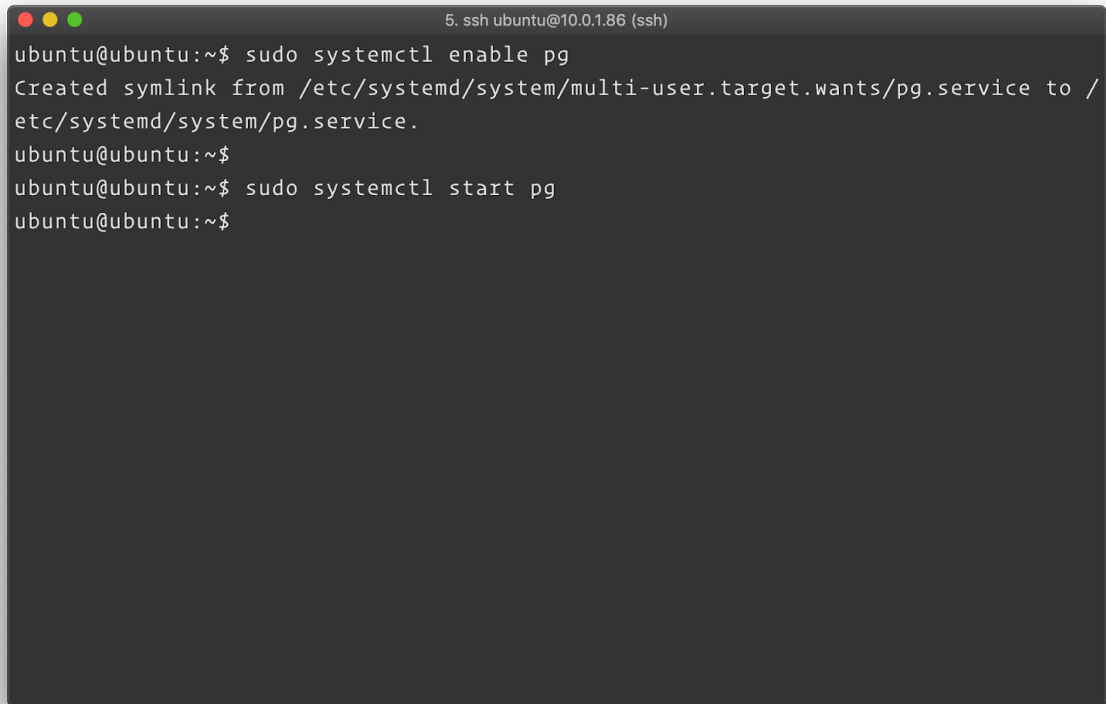
max_wal_senders = 4
wal_keep_segments = 128
█

65,0-1 37%
```

Start the postgresql service (pg.service) with the following command,

```
# Enable pg.service to start on boot
sudo systemctl enable pg
```

```
# Restart the machine
sudo systemctl reboot
```

A terminal window with a dark background and light text. The window title is "5. ssh ubuntu@10.0.1.86 (ssh)". The terminal shows the following commands and output:

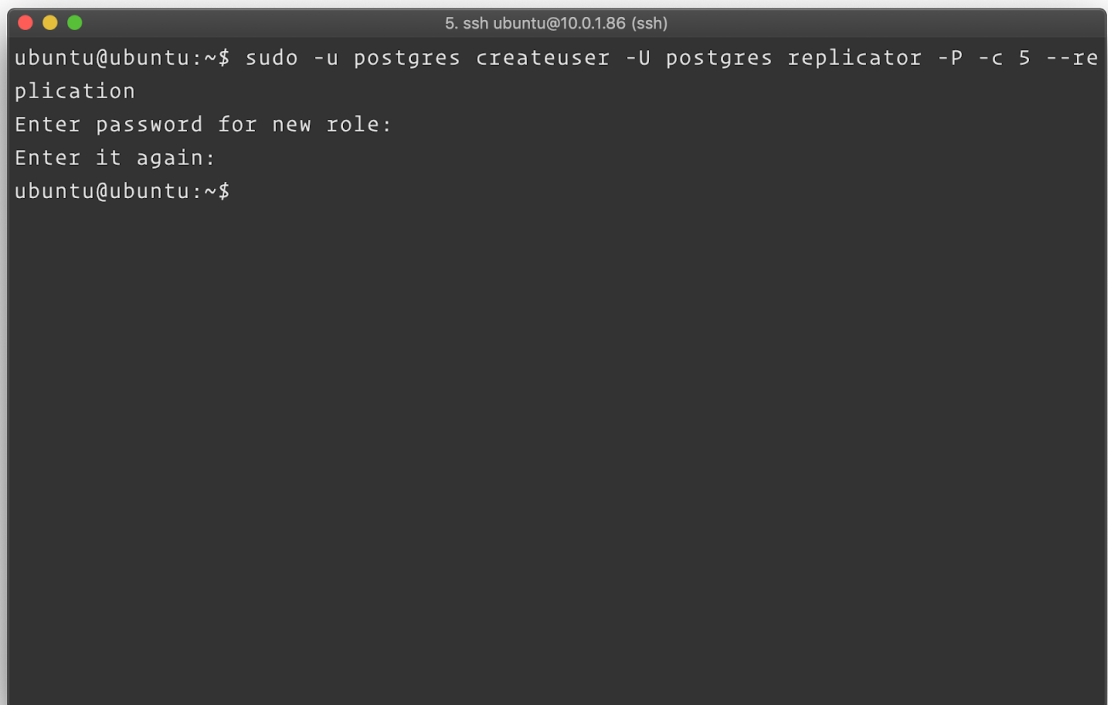
```
ubuntu@ubuntu:~$ sudo systemctl enable pg
Created symlink from /etc/systemd/system/multi-user.target.wants/pg.service to /
etc/systemd/system/pg.service.
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ sudo systemctl start pg
ubuntu@ubuntu:~$
```

Create Replication User

Create a postgres user on master server by running the following command,

```
# Creates a replication user called replicator  
sudo -u postgres createuser -U postgres replicator -P -c 5  
--replication
```

Enter a strong password and make a note of the password.

A terminal window with a dark background and light text. The title bar at the top shows three colored circles (red, yellow, green) and the text '5. ssh ubuntu@10.0.1.86 (ssh)'. The terminal content shows the command 'sudo -u postgres createuser -U postgres replicator -P -c 5 --replication' being entered. The prompt 'Enter password for new role:' is shown, followed by 'Enter it again:'. The prompt 'ubuntu@ubuntu:~\$' is shown at the end of the line.

```
5. ssh ubuntu@10.0.1.86 (ssh)  
ubuntu@ubuntu:~$ sudo -u postgres createuser -U postgres replicator -P -c 5 --re  
plication  
Enter password for new role:  
Enter it again:  
ubuntu@ubuntu:~$
```

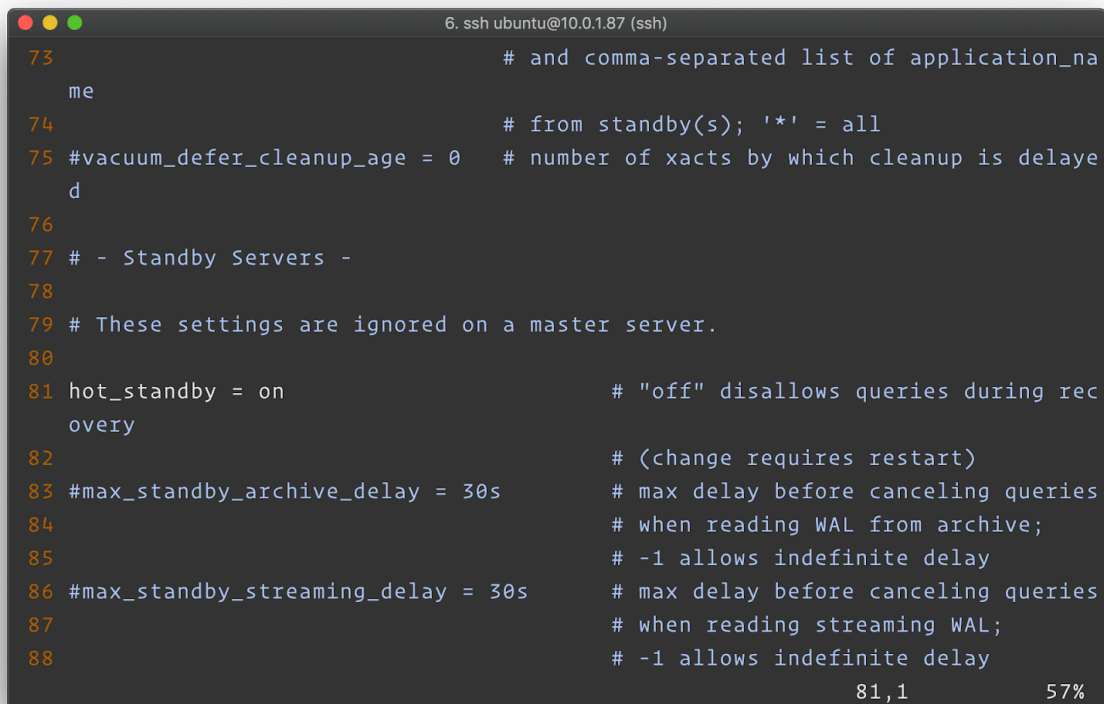
Replica Server

Open the **postgresql.conf** on the replica node with the following command,

```
# Open the postgresql.conf
sudo vim /etc/postgresql/11/main/postgresql.conf
```

Go to **hot_standby** on Line 82 and un-comment the line by removing # sign, Set hot_standby to on to enable the replica mode.

```
# Enable replica mode
hot_standby = on
```



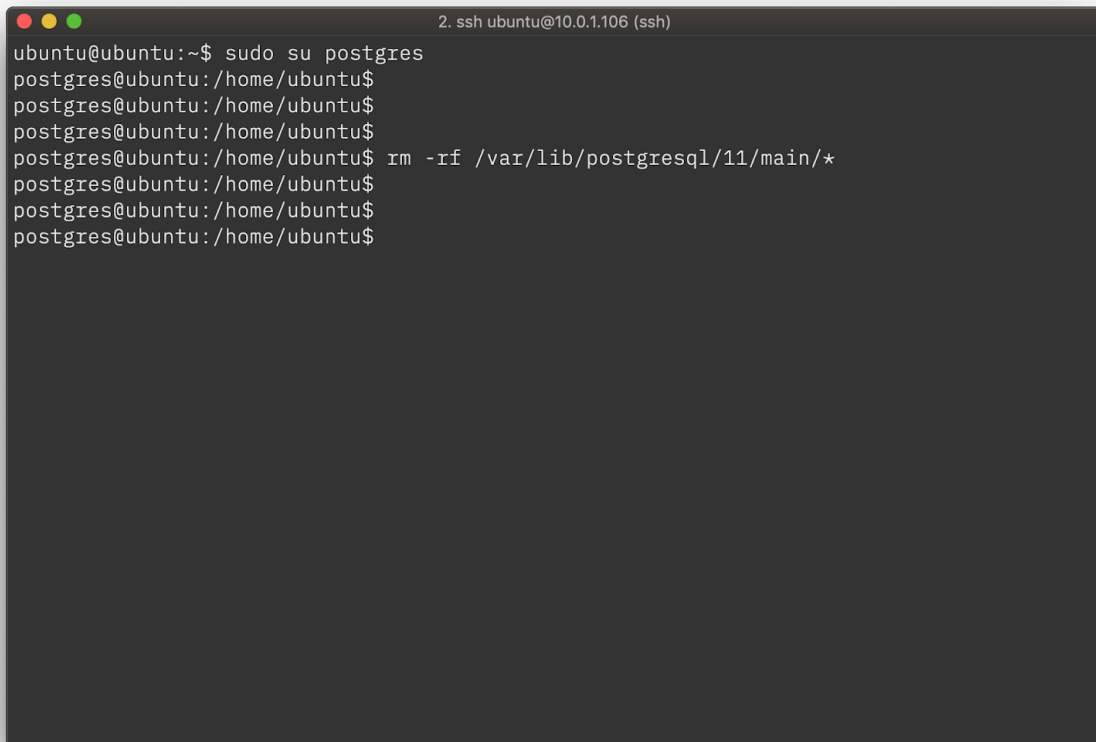
```
73                                     # and comma-separated list of application_na
me
74                                     # from standby(s); '*' = all
75 #vacuum_defer_cleanup_age = 0      # number of xacts by which cleanup is delaye
d
76
77 # - Standby Servers -
78
79 # These settings are ignored on a master server.
80
81 hot_standby = on                    # "off" disallows queries during rec
overy
82                                     # (change requires restart)
83 #max_standby_archive_delay = 30s   # max delay before canceling queries
84                                     # when reading WAL from archive;
85                                     # -1 allows indefinite delay
86 #max_standby_streaming_delay = 30s # max delay before canceling queries
87                                     # when reading streaming WAL;
88                                     # -1 allows indefinite delay
81,1                                  57%
```

Copy data directory from Master server

In order to sync the database base files on replica server, copy the data_directory from master server to the replica server with the following commands,

```
# Switch to postgres user
sudo su postgres
```

```
# Remove any existing files in data_directory
# i.e. /var/lib/postgresql/11/main
rm -rf /var/lib/postgresql/11/main/*
```



```
2. ssh ubuntu@10.0.1.106 (ssh)
ubuntu@ubuntu:~$ sudo su postgres
postgres@ubuntu:/home/ubuntu$
postgres@ubuntu:/home/ubuntu$
postgres@ubuntu:/home/ubuntu$
postgres@ubuntu:/home/ubuntu$ rm -rf /var/lib/postgresql/11/main/*
postgres@ubuntu:/home/ubuntu$
postgres@ubuntu:/home/ubuntu$
postgres@ubuntu:/home/ubuntu$
```

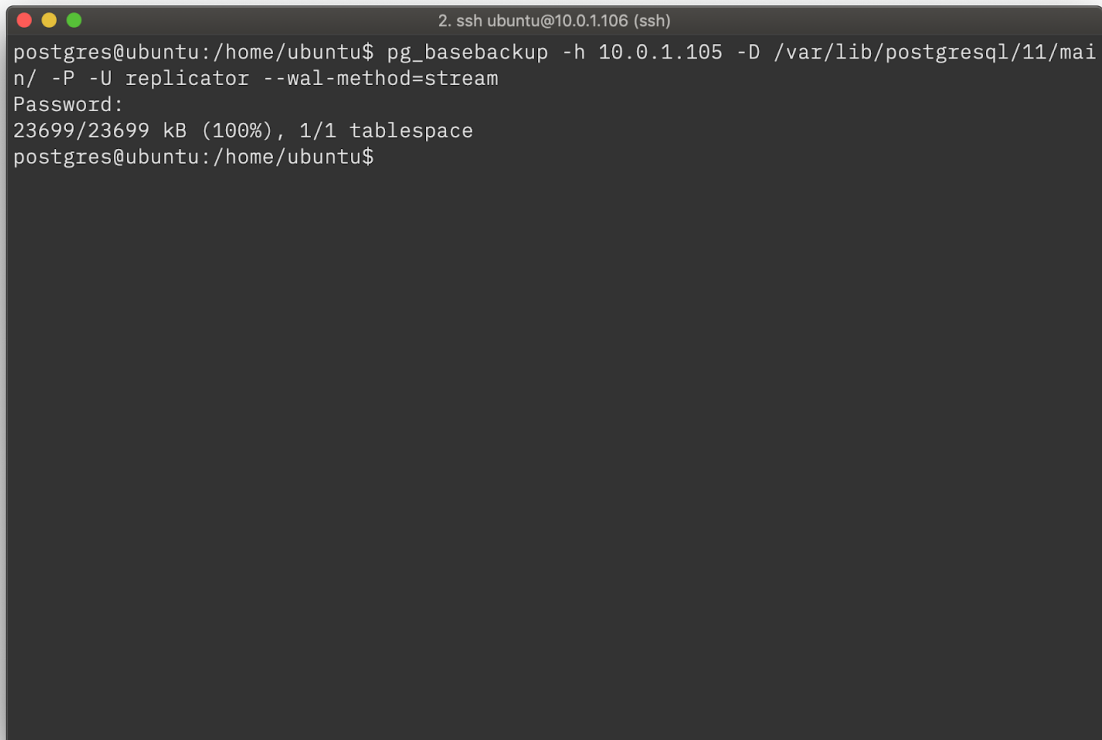

Run a remote base backup with `pg_basebackup` on the replica server to sync the database from master server to replica server,

When asked for password, use the password created in **Create Replication User** step.

```
# Run a base backup from replica server to copy from master server
pg_basebackup -h MASTER_IP -D /var/lib/postgresql/11/main/ -P -U
replicator --wal-method=stream
# Password:
```

For example, if the IP address of the master server is **10.0.1.105**, then the command would be

```
pg_basebackup -h 10.0.1.105 -D /var/lib/postgresql/11/main/ -P -U
replicator --wal-method=stream
```

A terminal window with a dark background and light text. The title bar at the top reads '2. ssh ubuntu@10.0.1.106 (ssh)'. The terminal shows the command 'pg_basebackup -h 10.0.1.105 -D /var/lib/postgresql/11/main/ -P -U replicator --wal-method=stream' being executed. It prompts for a password, which is entered. The output shows '23699/23699 kB (100%), 1/1 tablespace' and the command completes successfully.

```
postgres@ubuntu:/home/ubuntu$ pg_basebackup -h 10.0.1.105 -D /var/lib/postgresql/11/main/ -P -U replicator --wal-method=stream
Password:
23699/23699 kB (100%), 1/1 tablespace
postgres@ubuntu:/home/ubuntu$
```

Setup recovery.conf on Replica server

Copy **recovery.conf** from /var/lib/postgresql to /var/lib/postgresql/11/main with the following command,

```
# Switch to postgres user
sudo su postgres

# Go to /var/lib/postgresql/
cd /var/lib/postgresql

# Copy recovery.conf to
cp recovery.conf /var/lib/postgresql/11/main/recovery.conf
```

A terminal window titled "2. ssh ubuntu@10.0.1.106 (ssh)" showing a series of commands and their outputs. The user is 'postgres' on an 'ubuntu' machine. The commands executed are: 'cd /var/lib/postgresql/', followed by four empty prompts, and then 'cp recovery.conf /var/lib/postgresql/11/main/recovery.conf'.

```
postgres@ubuntu:~$ cd /var/lib/postgresql/
postgres@ubuntu:~$
postgres@ubuntu:~$
postgres@ubuntu:~$
postgres@ubuntu:~$
postgres@ubuntu:~$ cp recovery.conf /var/lib/postgresql/11/main/recovery.conf
postgres@ubuntu:~$
postgres@ubuntu:~$
```

Edit the **recovery.conf** located at `/var/lib/postgresql/11/main/recovery.conf`, uncomment and update the following options,

```
sudo -u postgres vim /var/lib/postgresql/11/main/recovery.conf
```

1. Set **standby_mode** to on line 6

```
standby_mode = 'on'
```

2. Update the **primary_conninfo** on line 7 with the replication user's username, password and the master server IP address

For Example, if the master server IP address is **10.0.1.86** and the **replicator** password is **ExampleSecret**, the **primary_conninfo** would look like

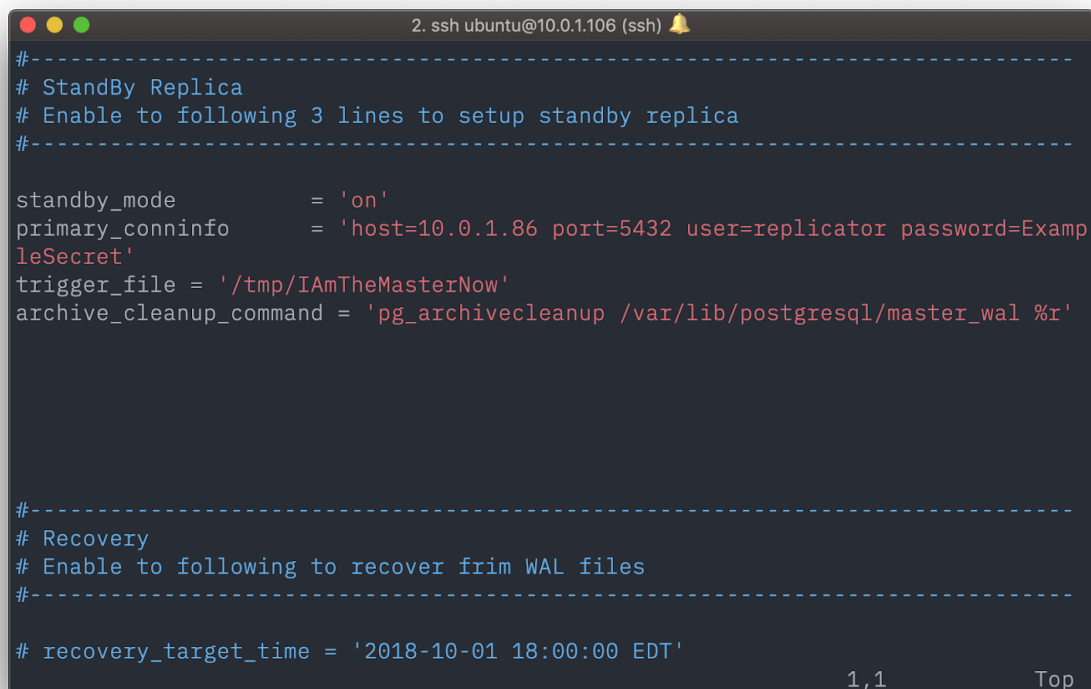
```
host=10.0.1.86 port=5432 user=replicator password=ExampleSecret
```

3. Update **trigger_file** on line 8 with a file path so when the file exists at the specified path. The specified file should not exist on replica server.

```
trigger_file = '/tmp/IAmTheMasterNow'
```

4. Update **archive_cleanup_command** on line 9 with the following command

```
archive_cleanup_command = 'pg_archivecleanup  
/var/lib/postgresql/master_wal %r'
```



```
2. ssh ubuntu@10.0.1.106 (ssh)
#-----
# StandBy Replica
# Enable to following 3 lines to setup standby replica
#-----

standby_mode          = 'on'
primary_conninfo      = 'host=10.0.1.86 port=5432 user=replicator password=ExampleSecret'
trigger_file          = '/tmp/IAmTheMasterNow'
archive_cleanup_command = 'pg_archivecleanup /var/lib/postgresql/master_wal %r'


#-----
# Recovery
# Enable to following to recover from WAL files
#-----

# recovery_target_time = '2018-10-01 18:00:00 EDT'

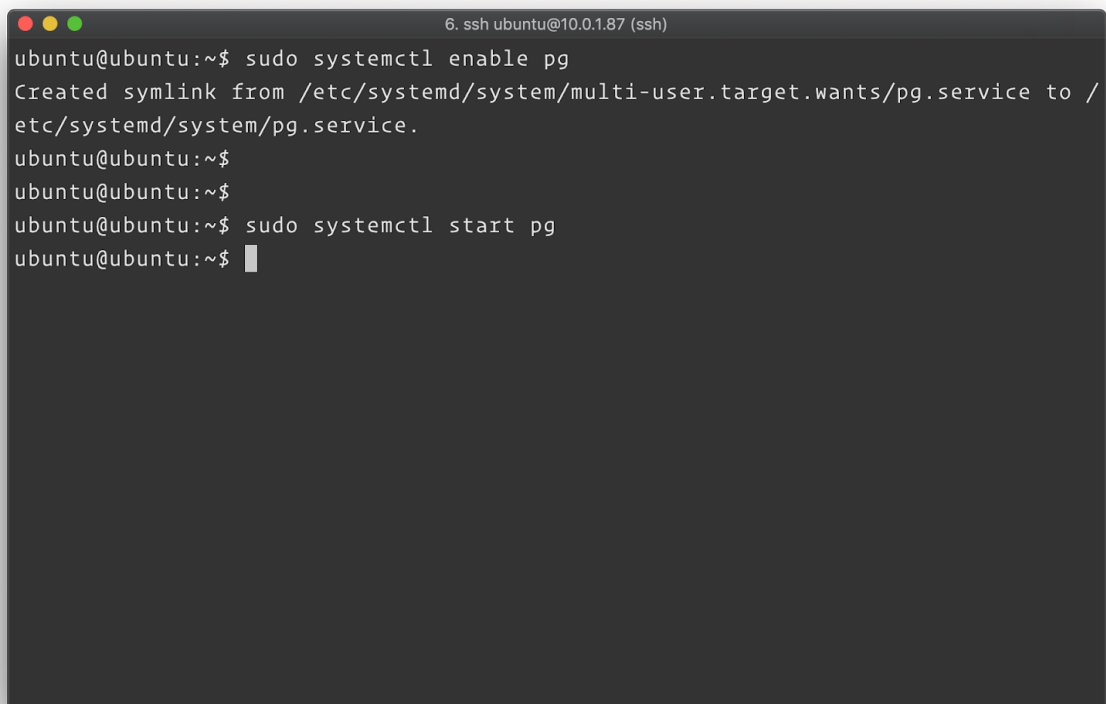
1,1 Top
```

Start the postgresql service (pg.service) on replica server with the following command,

```
# Switch to ubuntu user  
su ubuntu
```

```
# Enable pg.service to start on boot  
sudo systemctl enable pg
```

```
# Restart the machine  
sudo systemctl reboot
```



```
6. ssh ubuntu@10.0.1.87 (ssh)  
ubuntu@ubuntu:~$ sudo systemctl enable pg  
Created symlink from /etc/systemd/system/multi-user.target.wants/pg.service to /  
etc/systemd/system/pg.service.  
ubuntu@ubuntu:~$  
ubuntu@ubuntu:~$  
ubuntu@ubuntu:~$ sudo systemctl start pg  
ubuntu@ubuntu:~$
```

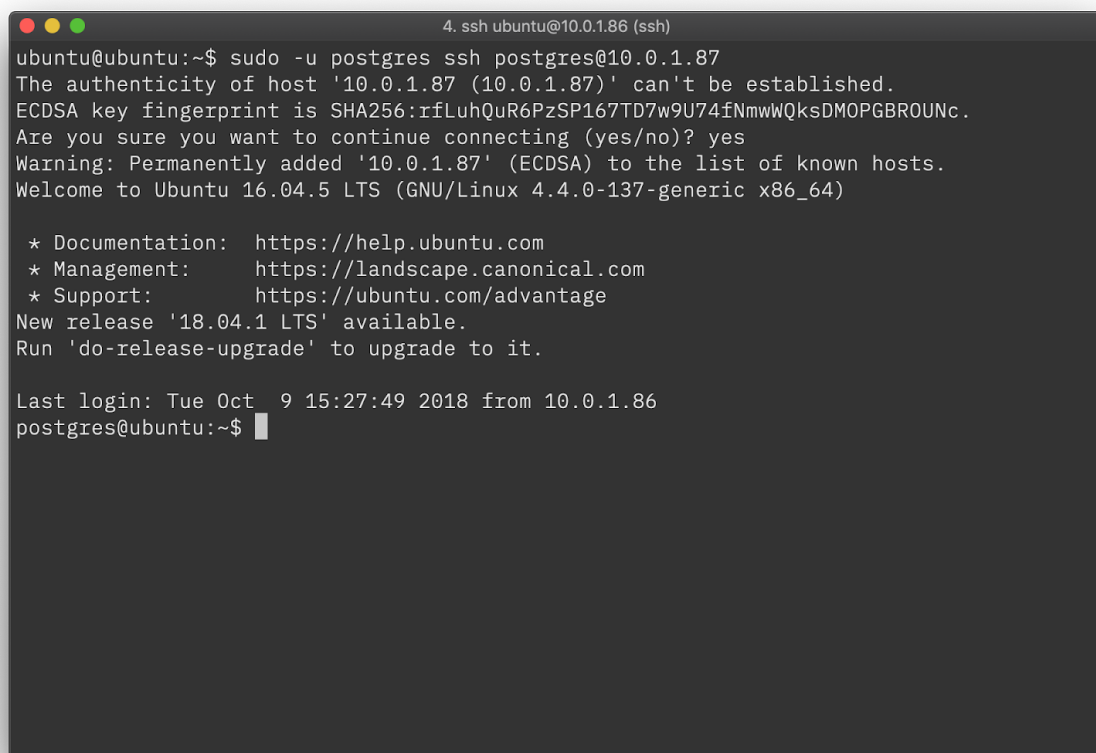
Update known_hosts

SSH into **Replica** server from master server using postgres user by running the following command from the master server,

```
# SSH into replica server
sudo -u postgres ssh postgres@REPLICA_IP_ADDRESS
```

For example if the replica server IP address is **10.0.1.87** then the command would look like

```
sudo -u postgres ssh postgres@10.0.1.87
```

A terminal window titled '4. ssh ubuntu@10.0.1.86 (ssh)' showing the execution of the command 'sudo -u postgres ssh postgres@10.0.1.87'. The output shows the SSH connection process, including a warning about the host's authenticity and a confirmation to add it to the known_hosts list. The terminal then displays the Ubuntu 16.04.5 LTS login banner and the prompt 'postgres@ubuntu:~\$'.

```
4. ssh ubuntu@10.0.1.86 (ssh)
ubuntu@ubuntu:~$ sudo -u postgres ssh postgres@10.0.1.87
The authenticity of host '10.0.1.87 (10.0.1.87)' can't be established.
ECDSA key fingerprint is SHA256:rfLuhQuR6PzSP167TD7w9U74fNmwwQksDMOPGBROUNc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.1.87' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-137-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

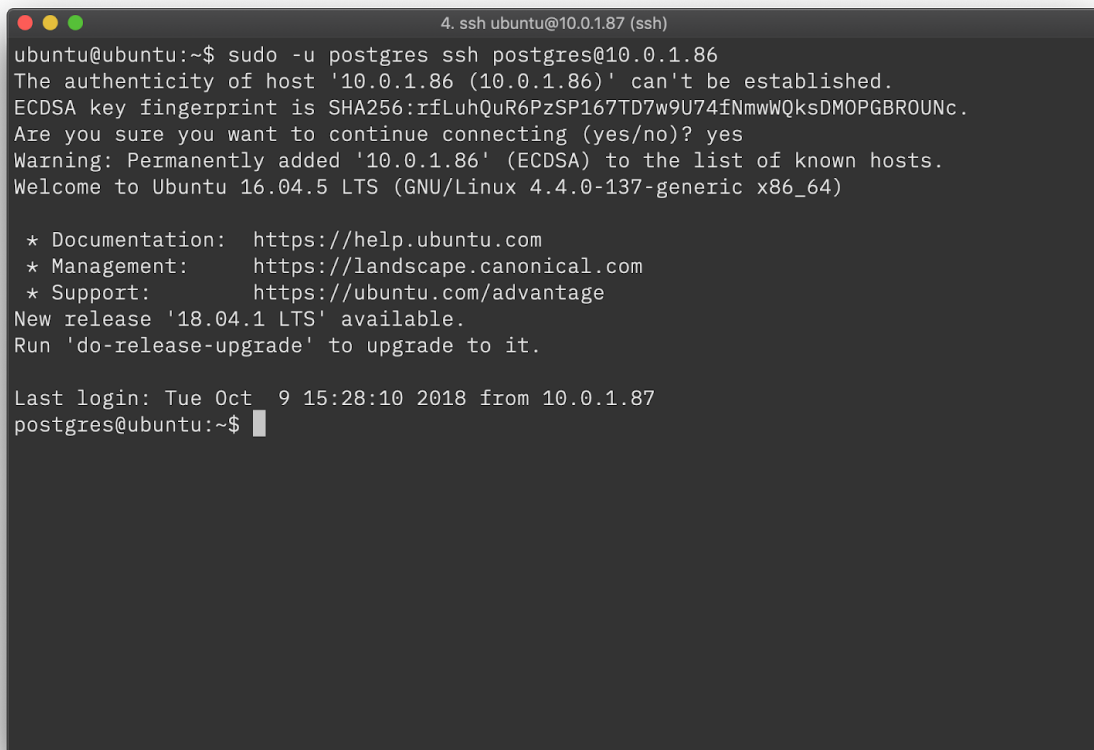
Last login: Tue Oct  9 15:27:49 2018 from 10.0.1.86
postgres@ubuntu:~$
```

SSH into **Master** server from replica server using postgres user by running the following command from the replica server,

```
# SSH into master server from replica server
sudo -u postgres ssh postgres@MASTER_IP_ADDRESS
```

For example if the master server IP address is **10.0.1.86** then the command would look like

```
sudo -u postgres ssh postgres@10.0.1.86
```

A terminal window titled "4. ssh ubuntu@10.0.1.87 (ssh)" showing the execution of the command "sudo -u postgres ssh postgres@10.0.1.86". The terminal output includes a warning about the host's authenticity, a confirmation to continue, and a message about a new Ubuntu release. The prompt then changes to "postgres@ubuntu:~\$".

```
4. ssh ubuntu@10.0.1.87 (ssh)
ubuntu@ubuntu:~$ sudo -u postgres ssh postgres@10.0.1.86
The authenticity of host '10.0.1.86 (10.0.1.86)' can't be established.
ECDSA key fingerprint is SHA256:rfLuhQuR6PzSP167TD7w9U74fNmWWQksDMOPGBROUNc.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.1.86' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-137-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage
New release '18.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Oct  9 15:28:10 2018 from 10.0.1.87
postgres@ubuntu:~$
```

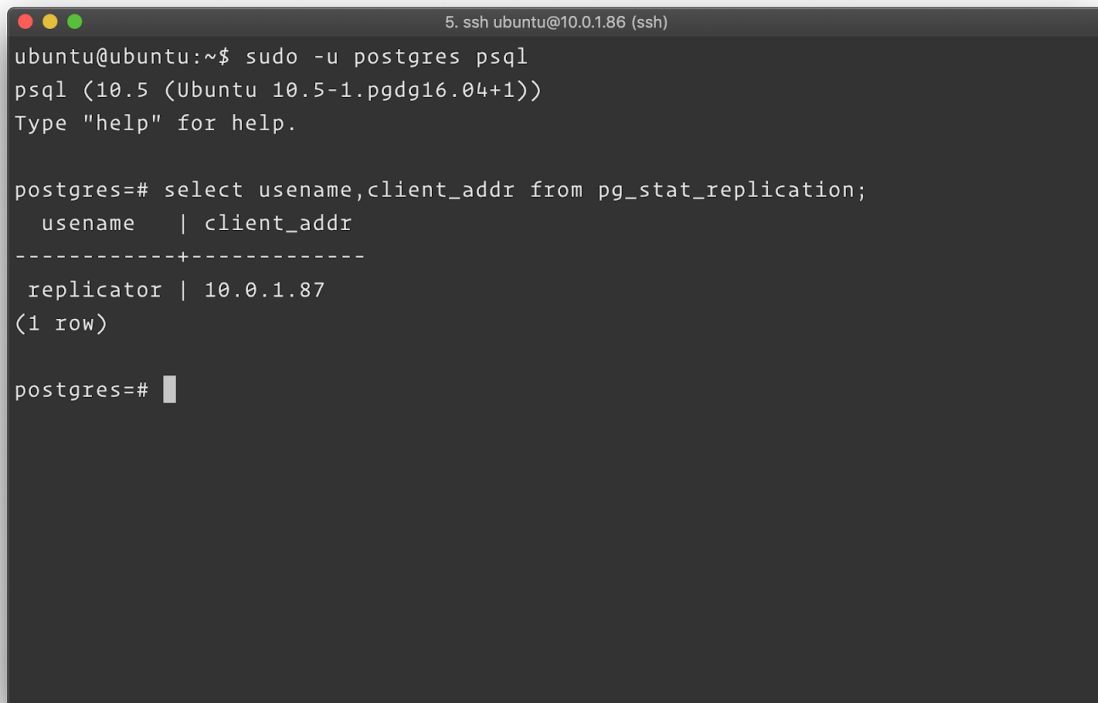
Verification

Verify Master

Verify the replication setup by querying the **master server** for replication activity should show the IP address of the replica server.

```
# Connect to the postgresql shell on master server
sudo -u postgres psql
```

```
# Query for pg_stat_replication
select username,client_addr from pg_stat_replication;
```



A terminal window titled "5. ssh ubuntu@10.0.1.86 (ssh)" showing the execution of a PostgreSQL query. The user runs `sudo -u postgres psql` to enter the PostgreSQL shell. Then, they run `select username,client_addr from pg_stat_replication;`. The output shows a single row with the username `replicator` and the client address `10.0.1.87`.

```
ubuntu@ubuntu:~$ sudo -u postgres psql
psql (10.5 (Ubuntu 10.5-1.pgdg16.04+1))
Type "help" for help.

postgres=# select username,client_addr from pg_stat_replication;
 username | client_addr 
-----+-----
 replicator | 10.0.1.87
(1 row)

postgres=#
```

Verify Replication

Run the following command on **replica server** to verify the postgres cluster is running without any issues

```
# Connect to postgres on replica server
sudo -u postgres psql
```

```
# Query pg_stat_wal_receiver to check if the master is sending WAL
SELECT pid,sender_host,sender_port FROM pg_stat_wal_receiver;
```



A terminal window titled "2. ssh ubuntu@10.0.1.106 (ssh)" shows the execution of a SQL query. The prompt is "postgres=#". The query is "SELECT pid,sender_host,sender_port FROM pg_stat_wal_receiver;". The output is a table with three columns: "pid", "sender_host", and "sender_port". The first row shows "65267", "10.0.1.105", and "5432". Below the table, it says "(1 row)". The prompt is now "postgres=#" with a cursor.

```
postgres=# SELECT pid,sender_host,sender_port FROM pg_stat_wal_receiver;
 pid | sender_host | sender_port 
-----+-----+-----
 65267 | 10.0.1.105  |         5432
(1 row)

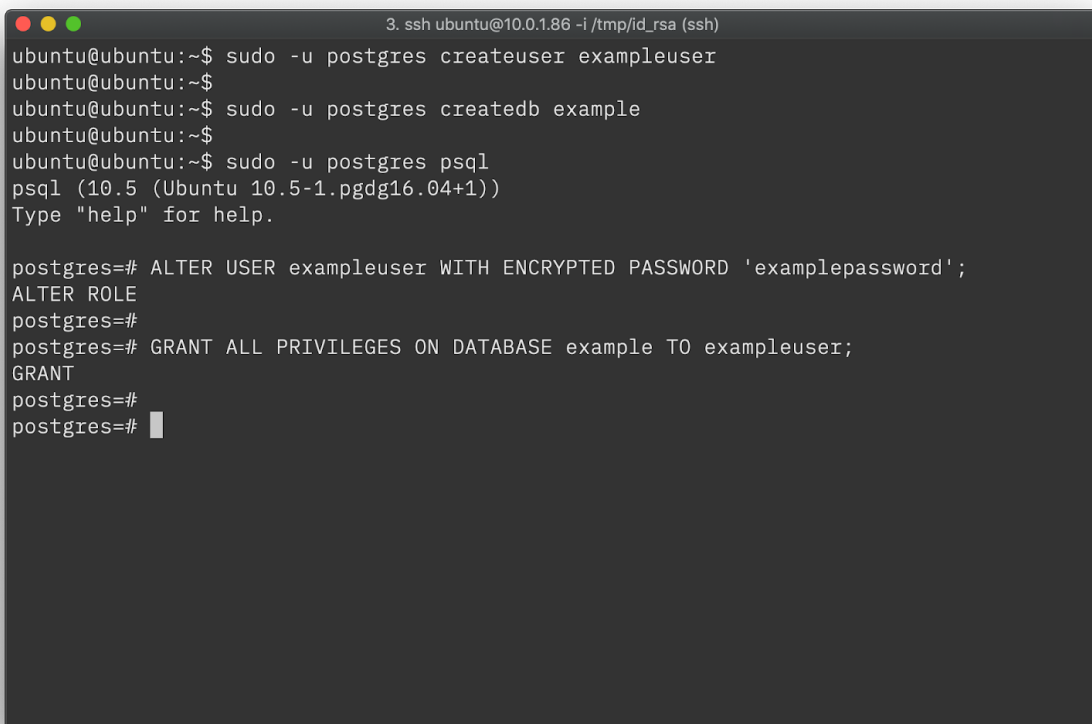
postgres=#
```


Usage

Create Database

In order to create the database on the master server for applications to use run the following commands,

```
# Create new user
sudo -u postgres createuser exampleuser
# Create new database
sudo -u postgres createdb example
# Connect to postgresql shell
sudo -u postgres psql
# Create a strong password for exampleuser
ALTER USER exampleuser WITH ENCRYPTED PASSWORD 'examplepassword';
# Grant all privileges on the example database to exampleuser
GRANT ALL PRIVILEGES ON DATABASE example TO exampleuser;
```

A terminal window titled '3. ssh ubuntu@10.0.1.86 -i /tmp/id_rsa (ssh)' showing the execution of the commands listed in the previous block. The user 'ubuntu' runs 'sudo -u postgres createuser exampleuser', 'sudo -u postgres createdb example', and 'sudo -u postgres psql'. The 'psql' shell shows the execution of 'ALTER USER exampleuser WITH ENCRYPTED PASSWORD 'examplepassword';', 'ALTER ROLE', and 'GRANT ALL PRIVILEGES ON DATABASE example TO exampleuser;'. The terminal ends with a blank line after the last command.

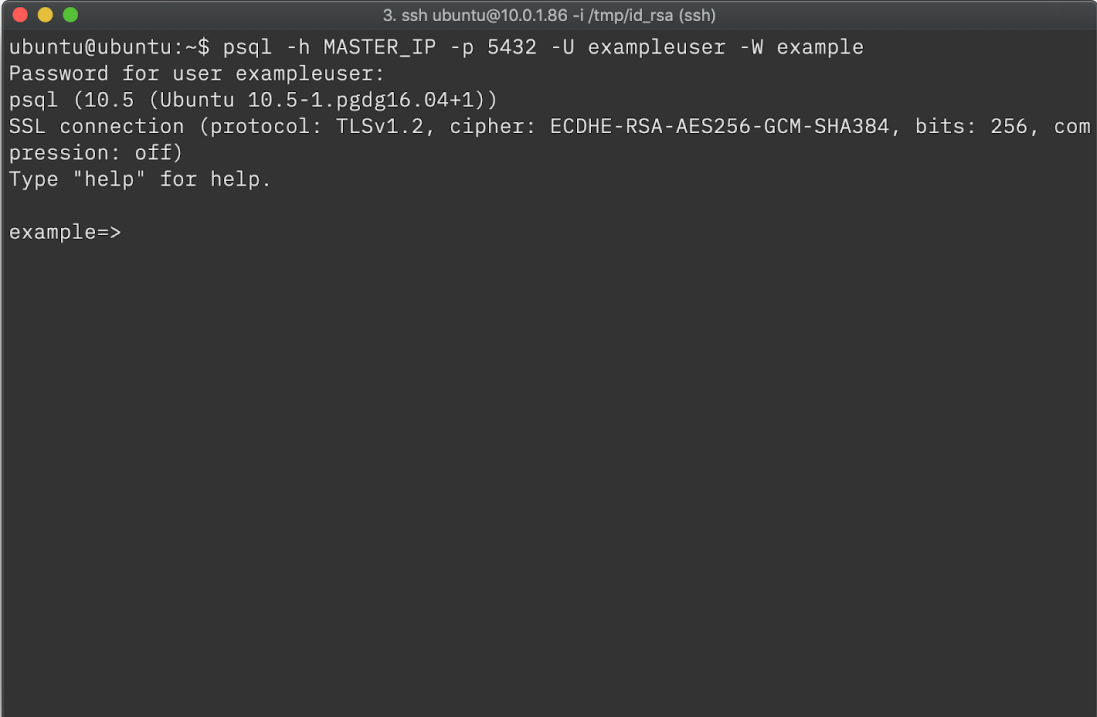
```
3. ssh ubuntu@10.0.1.86 -i /tmp/id_rsa (ssh)
ubuntu@ubuntu:~$ sudo -u postgres createuser exampleuser
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ sudo -u postgres createdb example
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ sudo -u postgres psql
psql (10.5 (Ubuntu 10.5-1.pgdg16.04+1))
Type "help" for help.

postgres=# ALTER USER exampleuser WITH ENCRYPTED PASSWORD 'examplepassword';
ALTER ROLE
postgres=#
postgres=# GRANT ALL PRIVILEGES ON DATABASE example TO exampleuser;
GRANT
postgres=#
postgres=#
```

Connect to Database

Connect to the newly created database by running the following command,

```
# Connect to example database on Master Server
psql -h MASTER_IP -p 5432 -U exampleuser -W example
# Password for user exampleuser:
```

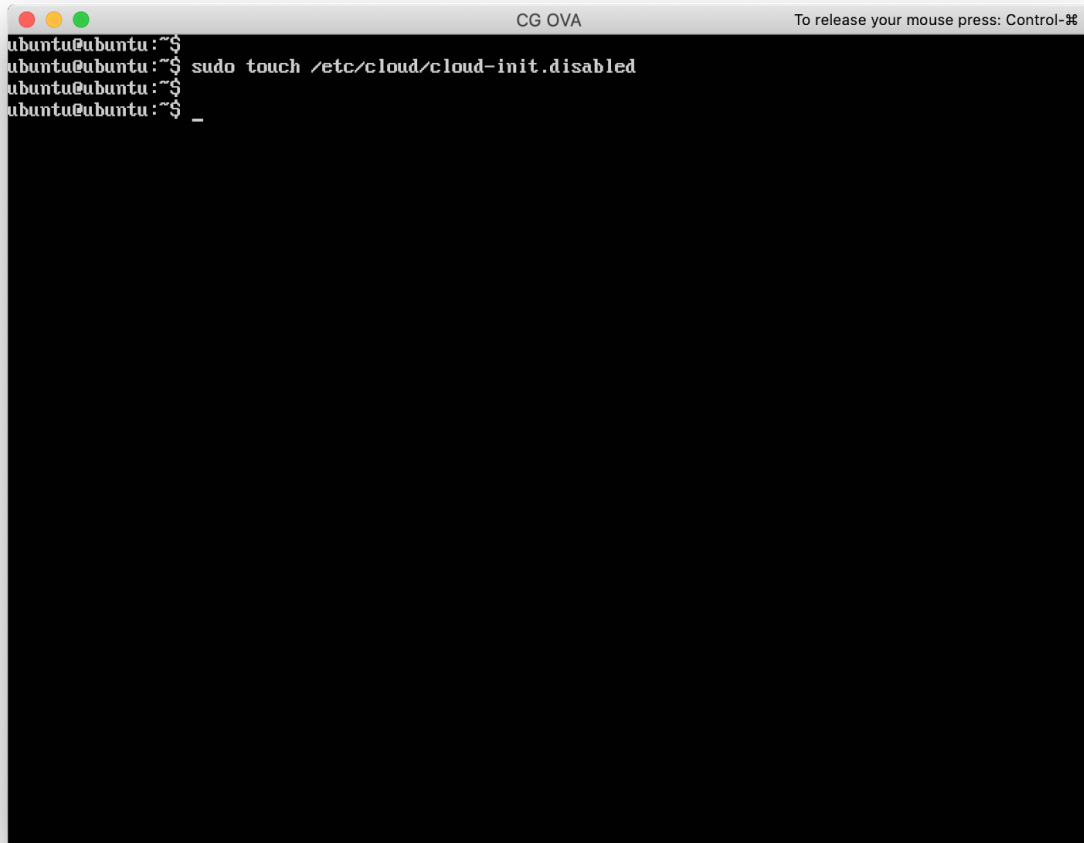
A terminal window titled '3. ssh ubuntu@10.0.1.86 -i /tmp/id_rsa (ssh)' showing the execution of a psql command. The user 'ubuntu@ubuntu' runs 'psql -h MASTER_IP -p 5432 -U exampleuser -W example'. The terminal prompts for the password for user 'exampleuser'. The output shows 'psql (10.5 (Ubuntu 10.5-1.pgdg16.04+1))', 'SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)', and 'Type "help" for help.' The prompt 'example=>' is visible at the bottom.

```
3. ssh ubuntu@10.0.1.86 -i /tmp/id_rsa (ssh)
ubuntu@ubuntu:~$ psql -h MASTER_IP -p 5432 -U exampleuser -W example
Password for user exampleuser:
psql (10.5 (Ubuntu 10.5-1.pgdg16.04+1))
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits: 256, compression: off)
Type "help" for help.
example=>
```

Disable Cloud-Init

Cloud-init is the service that initializes cloud images on EC2. However, it is not required when running the server on-premise. Disable cloud-init by running the following command,

```
sudo touch /etc/cloud/cloud-init.disabled
```

A terminal window titled "CG OVA" with a subtitle "To release your mouse press: Control-⌘". The terminal shows a series of prompts and commands: "ubuntu@ubuntu:~\$", "ubuntu@ubuntu:~\$ sudo touch /etc/cloud/cloud-init.disabled", "ubuntu@ubuntu:~\$", and "ubuntu@ubuntu:~\$ _". The terminal background is black, and the text is white.

```
ubuntu@ubuntu:~$  
ubuntu@ubuntu:~$ sudo touch /etc/cloud/cloud-init.disabled  
ubuntu@ubuntu:~$  
ubuntu@ubuntu:~$ _
```