PostgreSQL Cluster

Deployment Guide

Version 1.0.2

Contents

Copyright Notice

Document Revision History

October 8, 2018

- Initial release of documentation

October 4, 2019

- Postgres extensions

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 primary-replica mode where one of the nodes serves as a primary node and others serves as a hot-standby replica which can be promoted to become the primary node.

Requirements

Inorder to setup the PostgreSQL cluster the following are required,

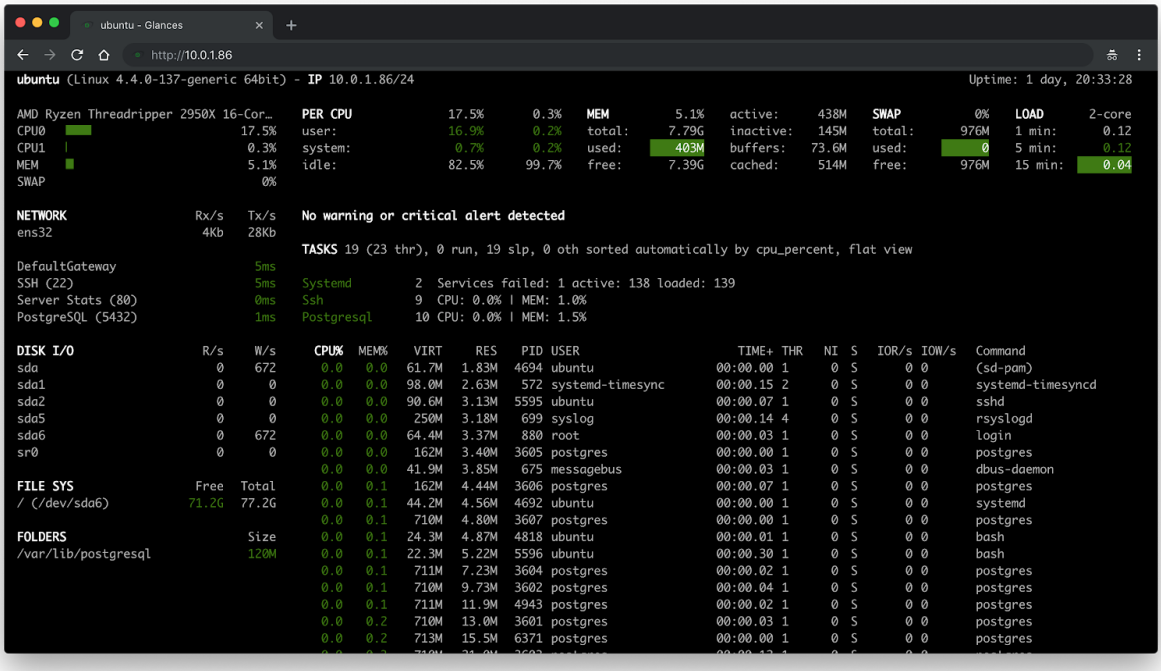
- IP address of the **primary** node, referred as **PRIMARY_IP**
- IP address of the **replica** node, referred as **REPLICA_IP**

Both the primary 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 primary server is **10.0.1.86**,



Setup

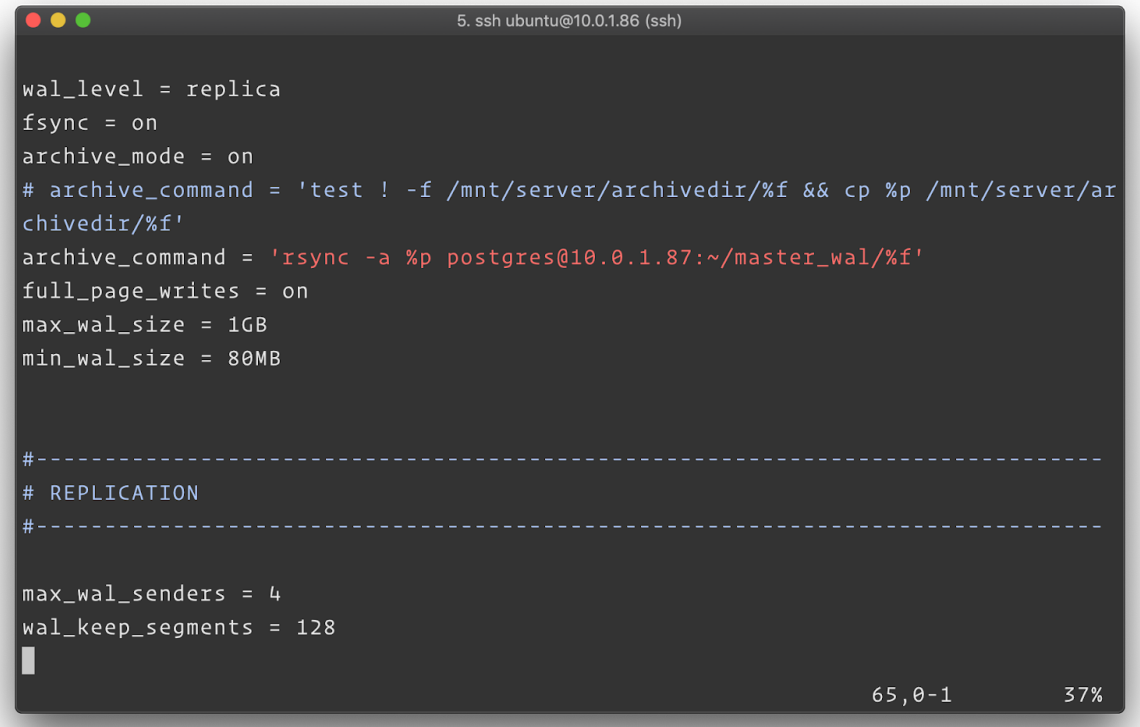
Primary Server

Open the **postgresql.conf** on the primary 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,

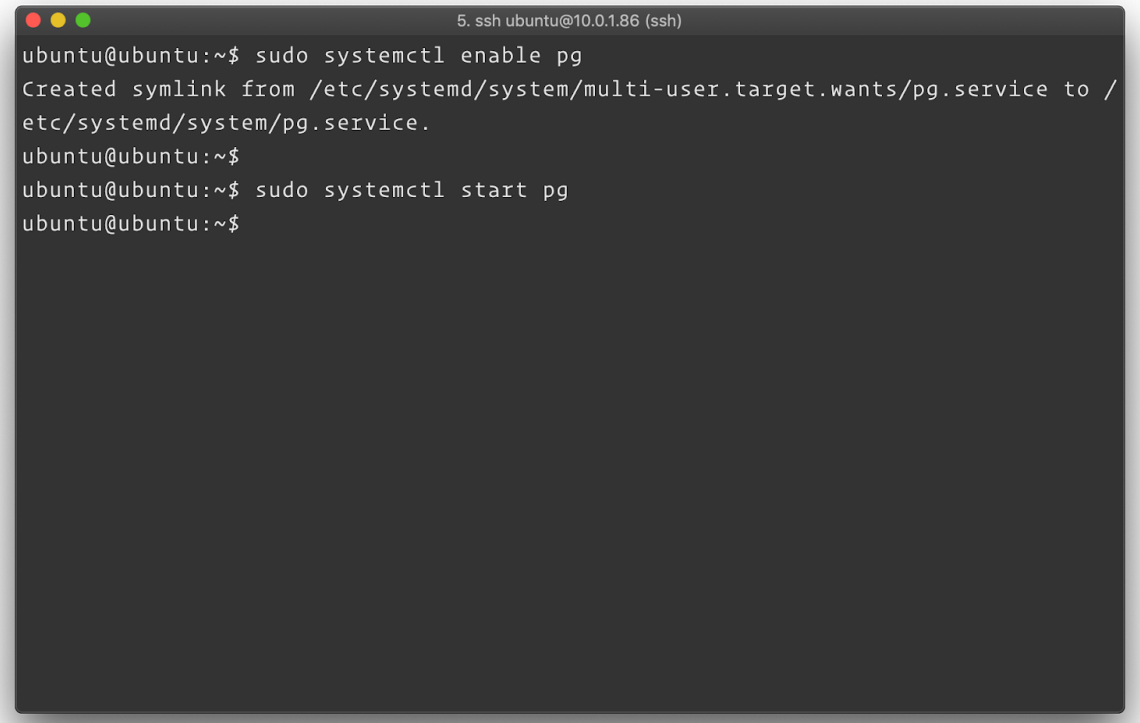
```
wal_keep_segments = 0
archive_command = 'rsync -a %p postgres@10.0.1.87:~/master_wal/%f'
```



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
```

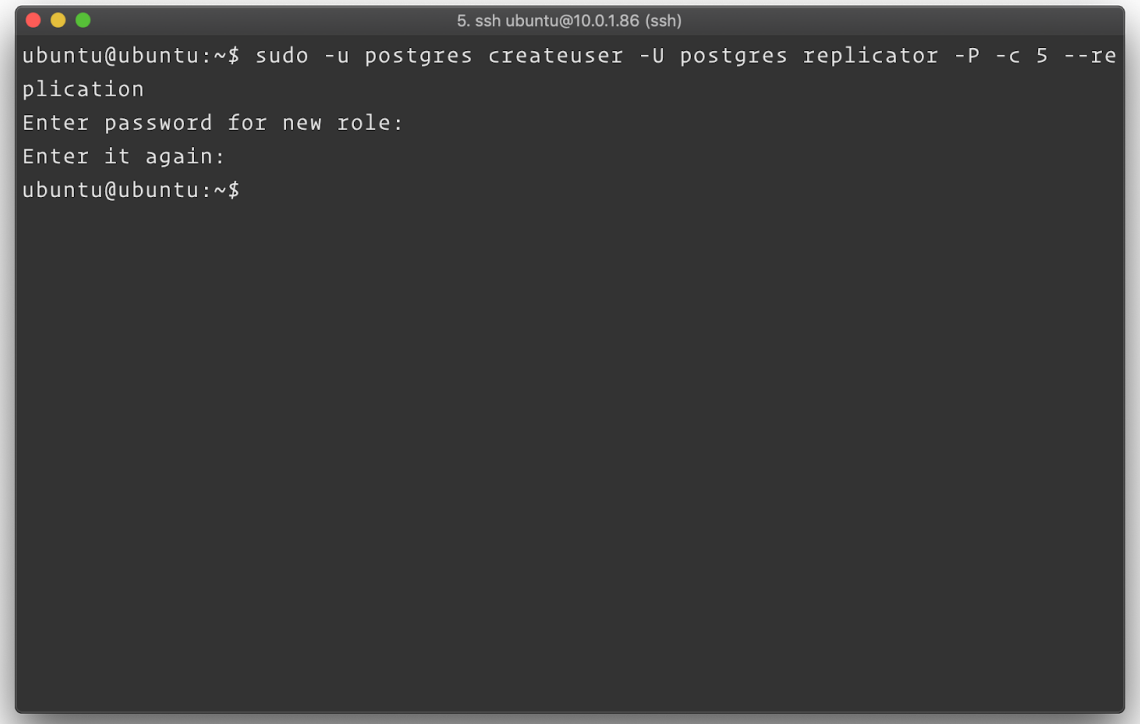


Create Replication User

Create a postgres user on **primary** 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.



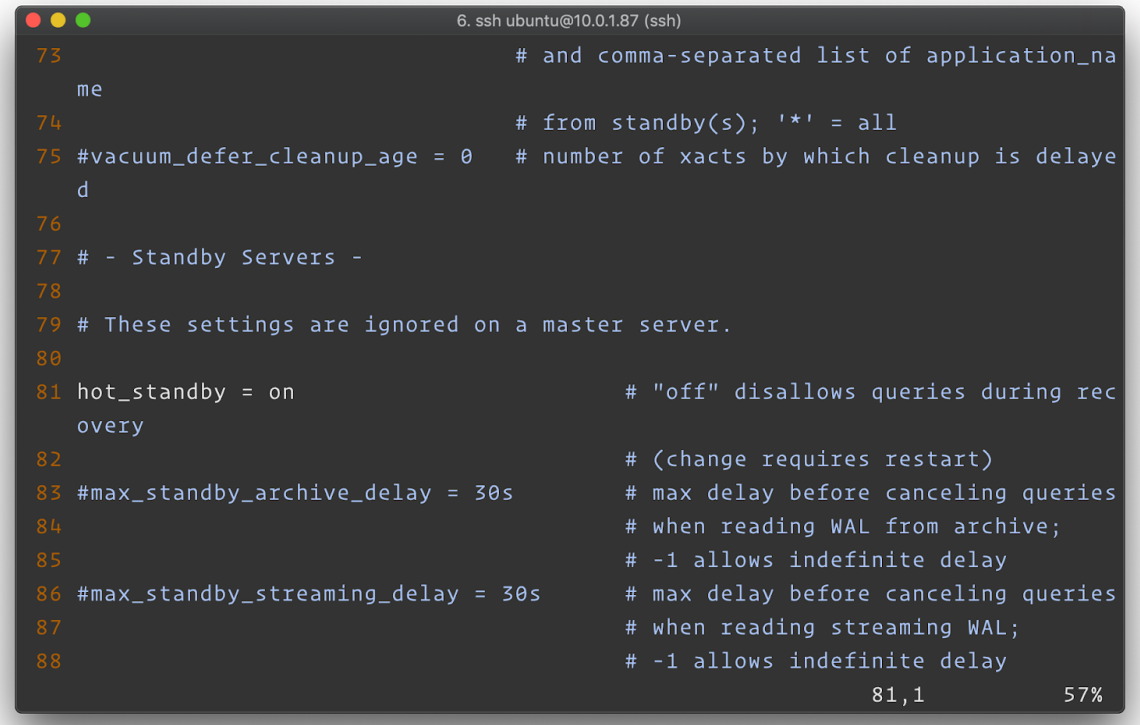
Replica Server

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
```

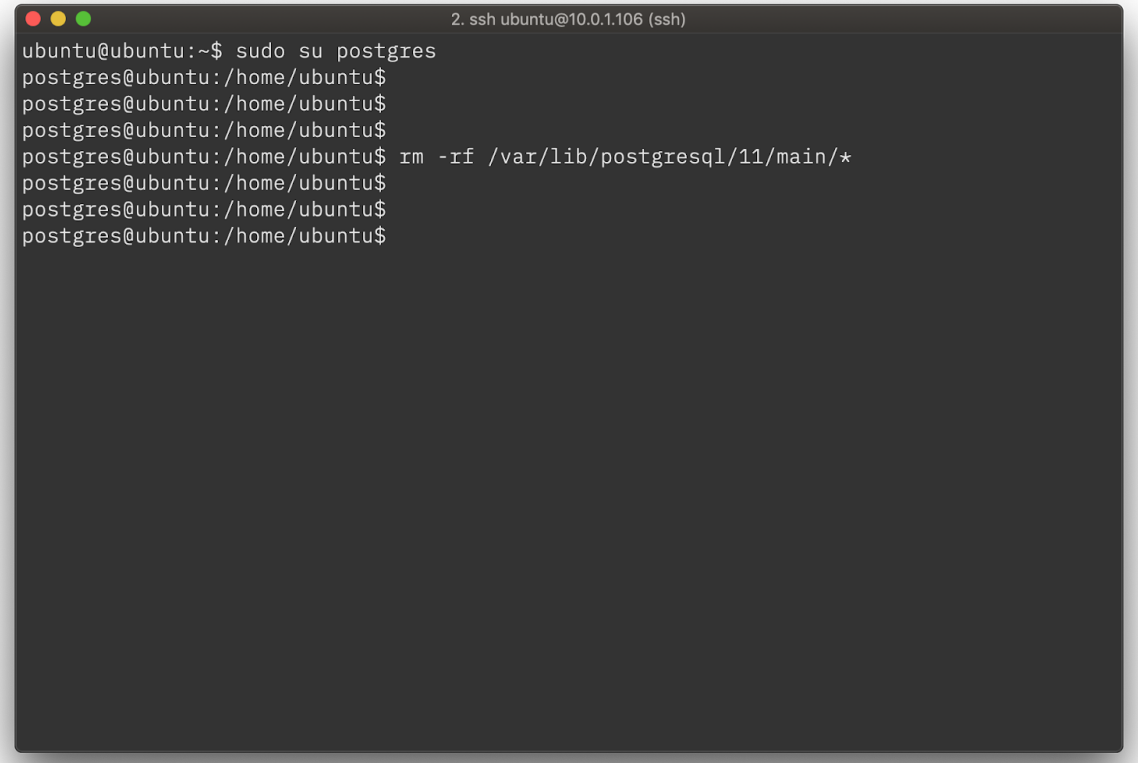


Copy data directory from Primary server

In order to sync the database base files on replica server, copy the data_directory from **primary** 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/*
```



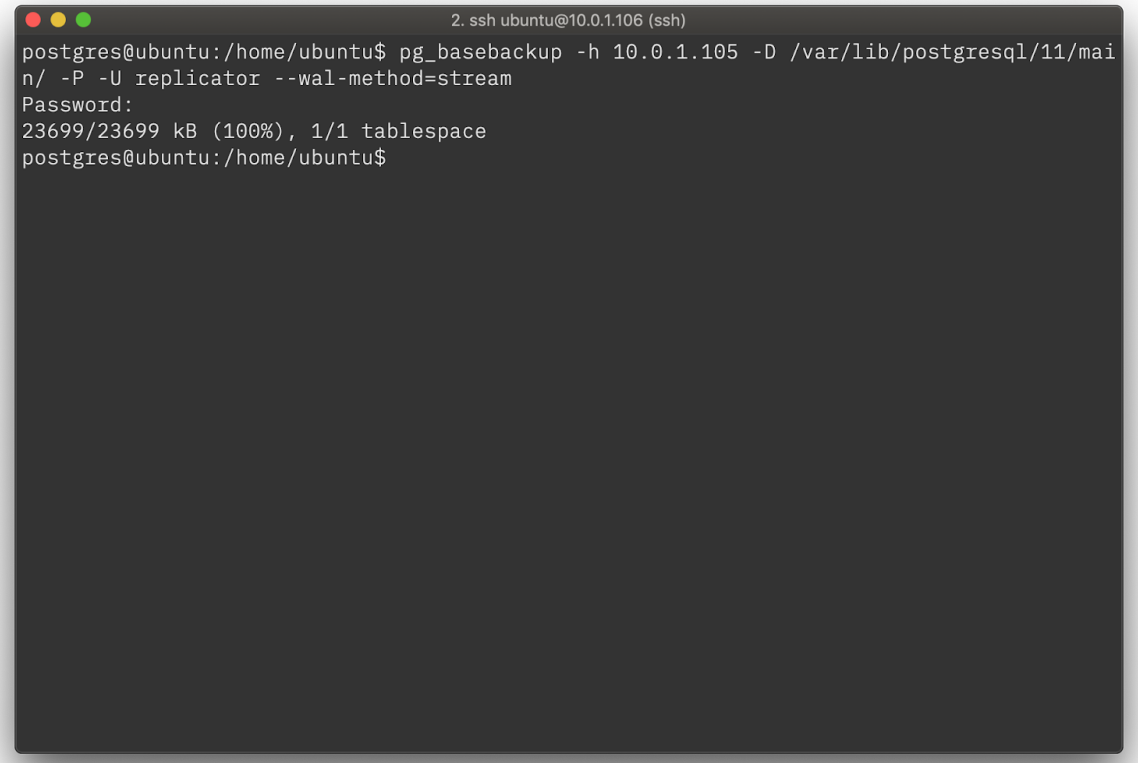
Run a remote base backup with pg_basebackup on the replica server to sync the database from primary server to replica server,

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

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

For example, if the IP address of the primary 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
```



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
```

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
```

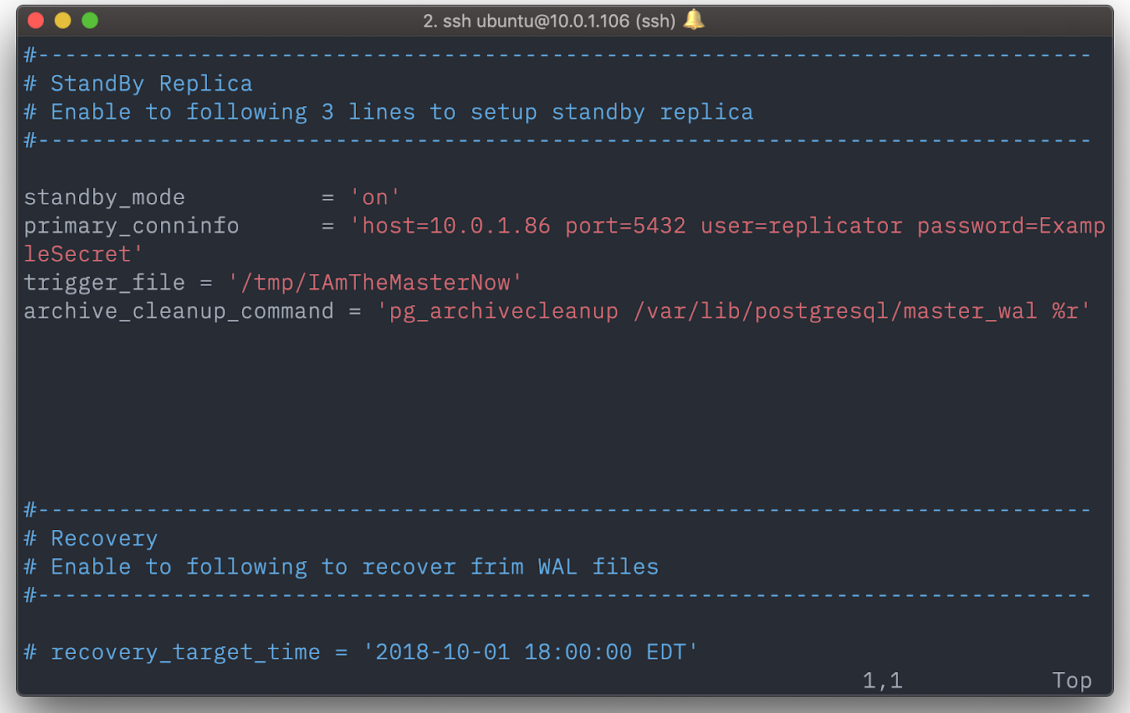
1. Set **standby_mode** to on line 6
   ```
   standby_mode = 'on'
   ```

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

   For Example, if the primary 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'
   ```

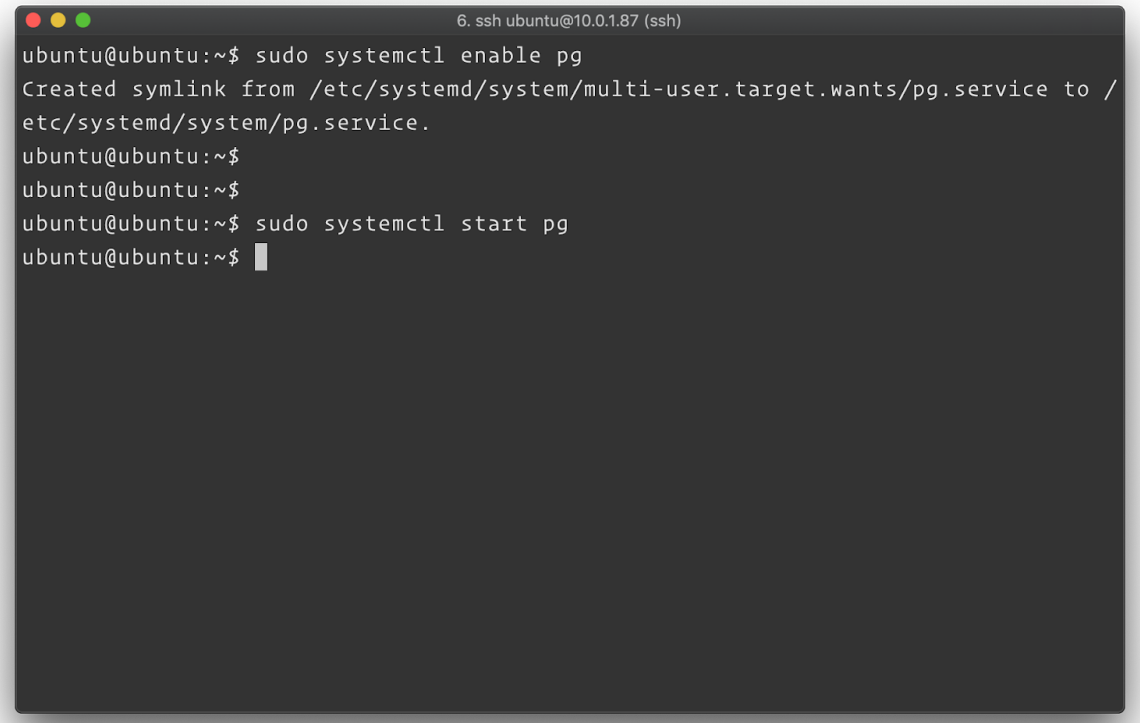


Start the postgresql service (pg.service) on replica server with the following command,

```
# Switch to ubuntu user
su ubuntu

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

# Restart the machine
sudo systemctl reboot
```



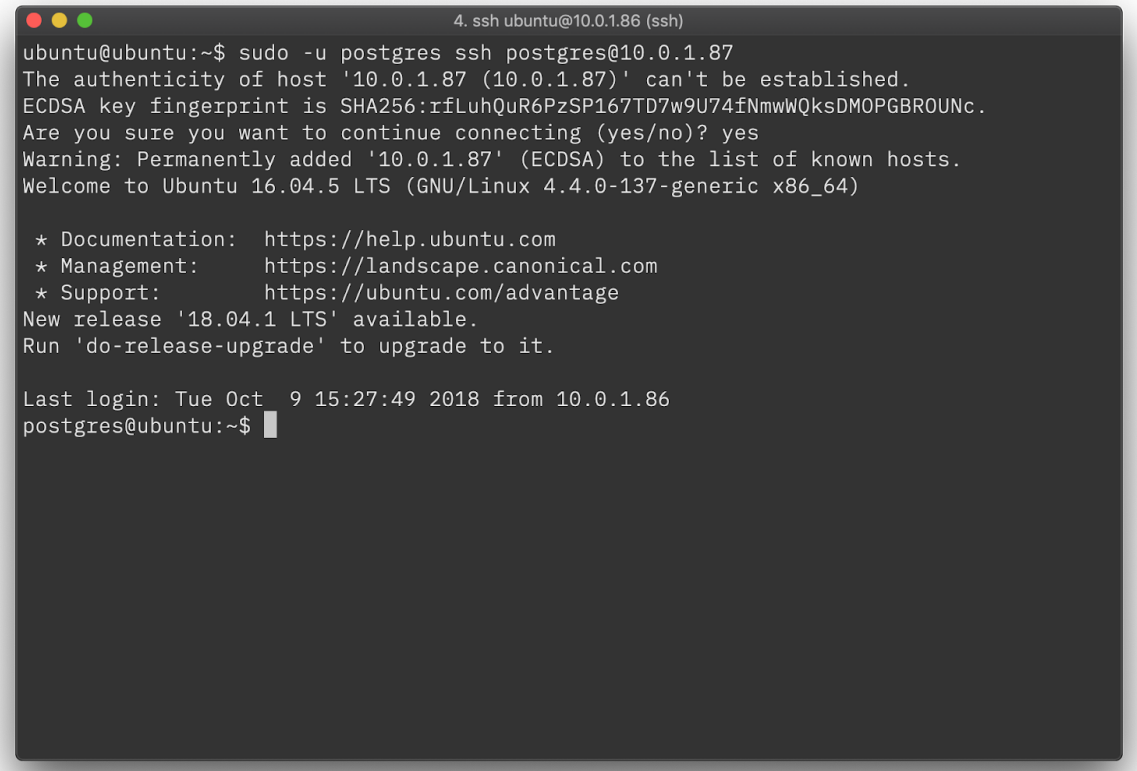
Update known_hosts

SSH into **Replica** server from primary server using postgres user by running the following command from the primary 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
```

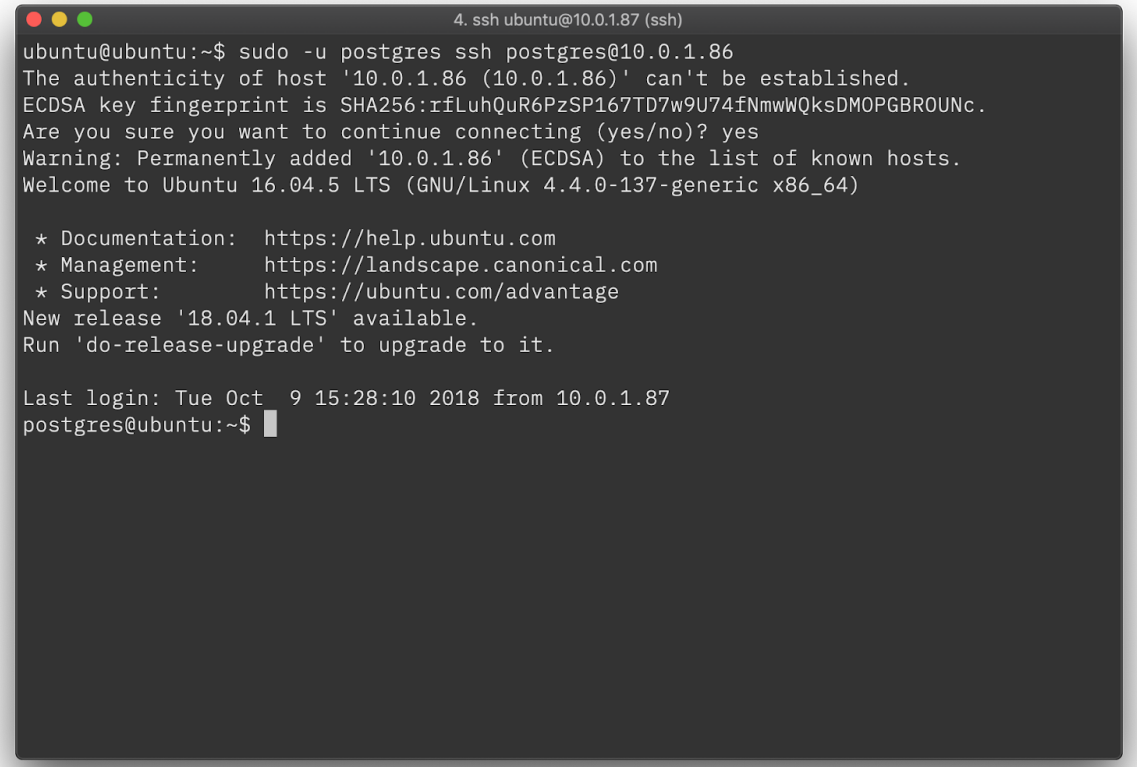


SSH into **primary** server from replica server using postgres user by running the following command from the replica server,

```
# SSH into primary server from replica server
sudo -u postgres ssh postgres@PRIMARY_IP_ADDRESS
```

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

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



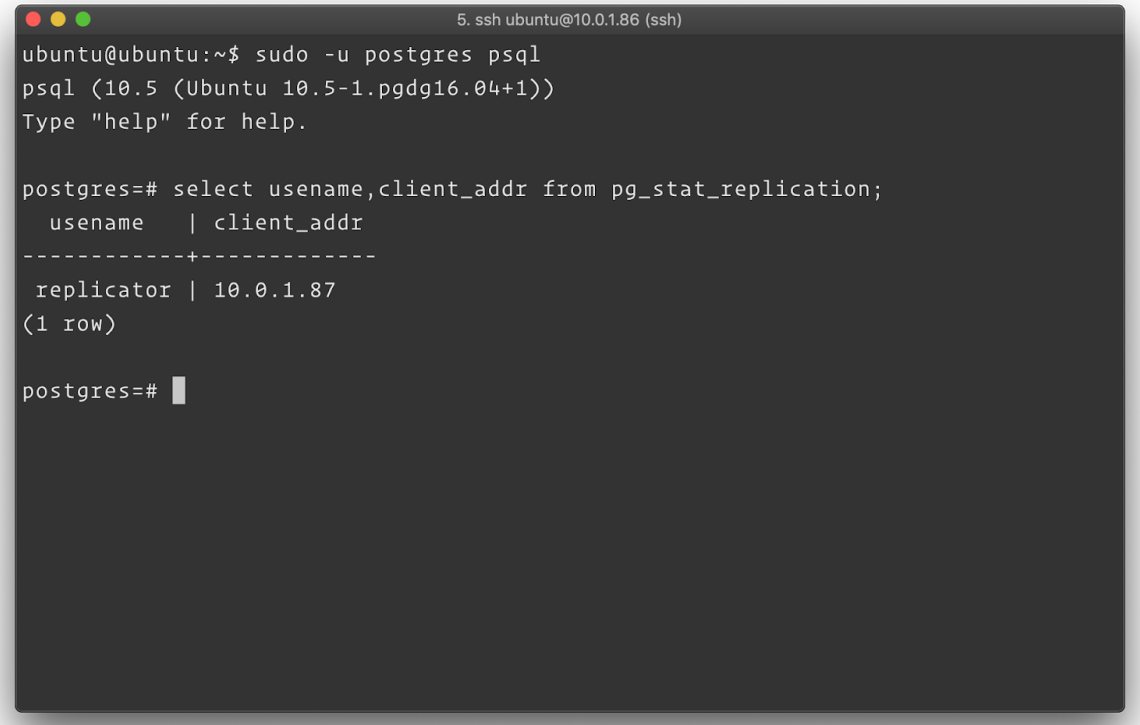
Verification

Verify Primary

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

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

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

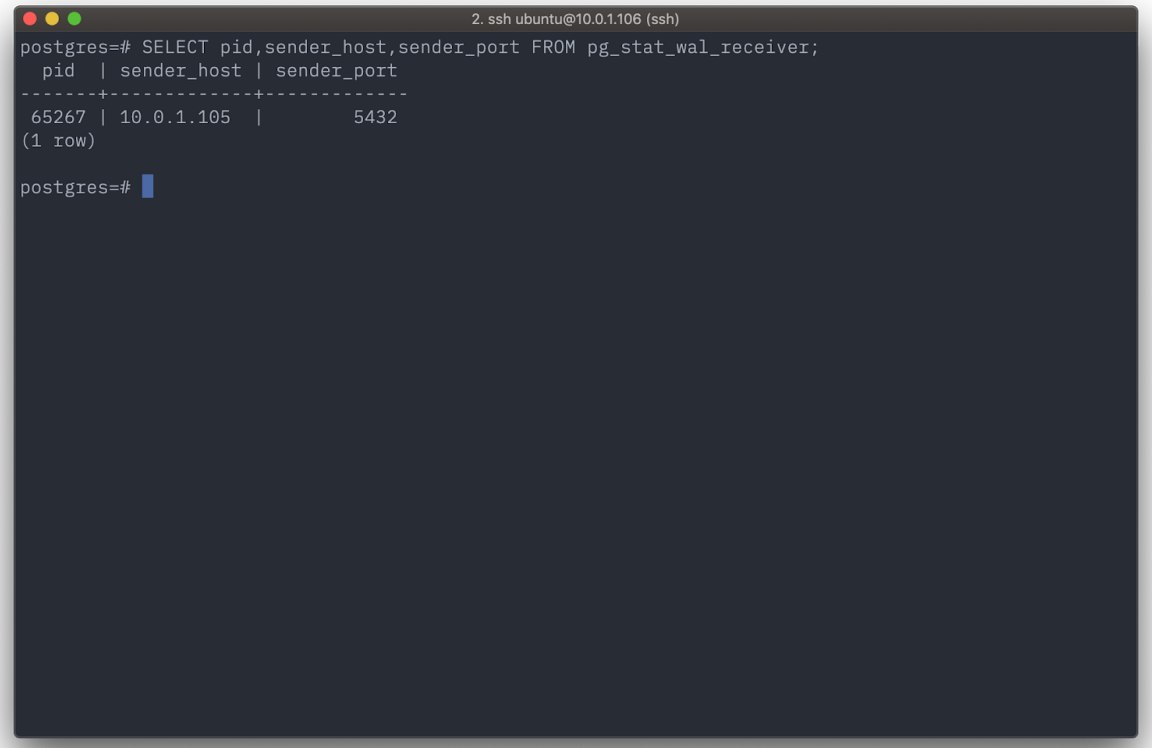


Verify Replication

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 primary is sending WAL
SELECT pid,sender_host,sender_port FROM pg_stat_wal_receiver;
```

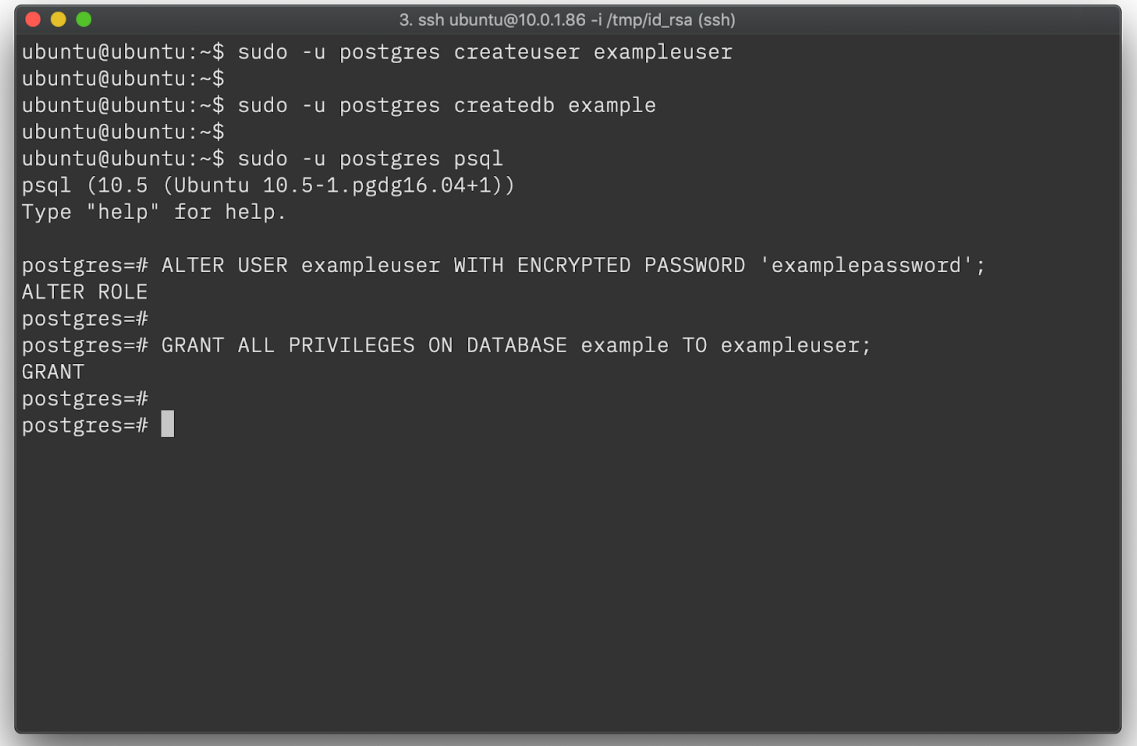


Usage

Create Database

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

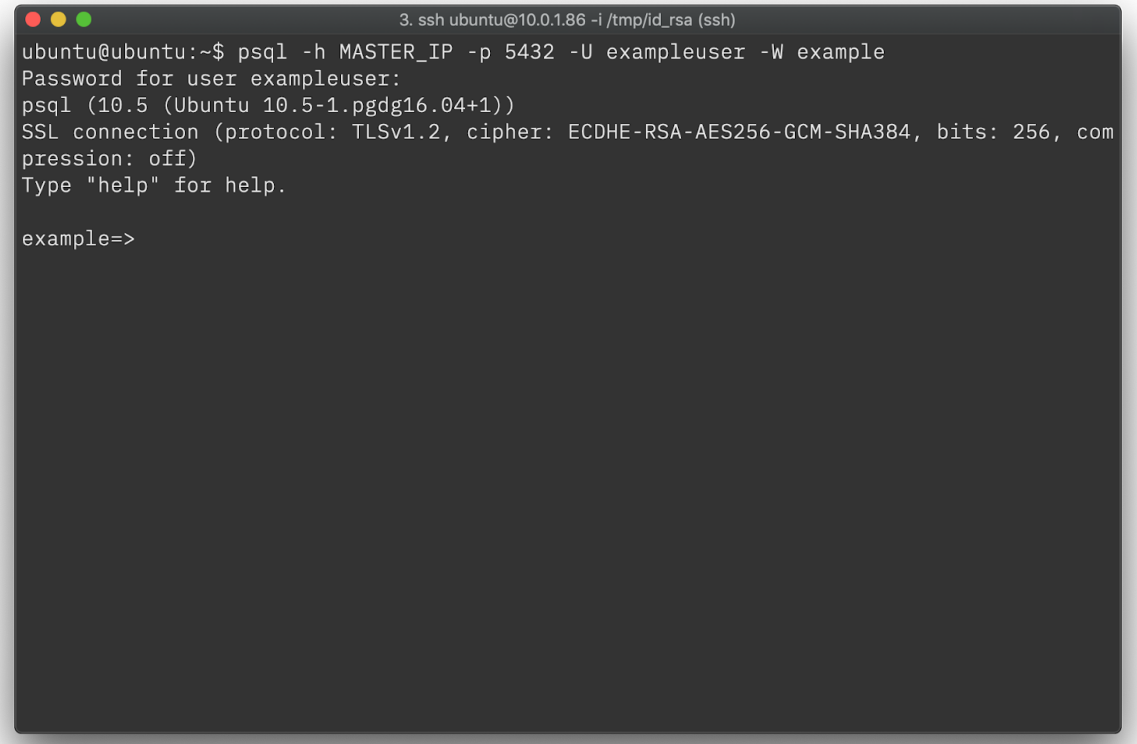
```
# Create new user
sudo -u postgres createuser exampleuser
# Create new database
sudo -u postgres createdb example
# Connect to postgresql shell
sudo -u postgres psql
# Create a strong password for exampleuser
ALTER USER exampleuser WITH ENCRYPTED PASSWORD 'examplepassword';
# Grant all privileges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



Connect to Database

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

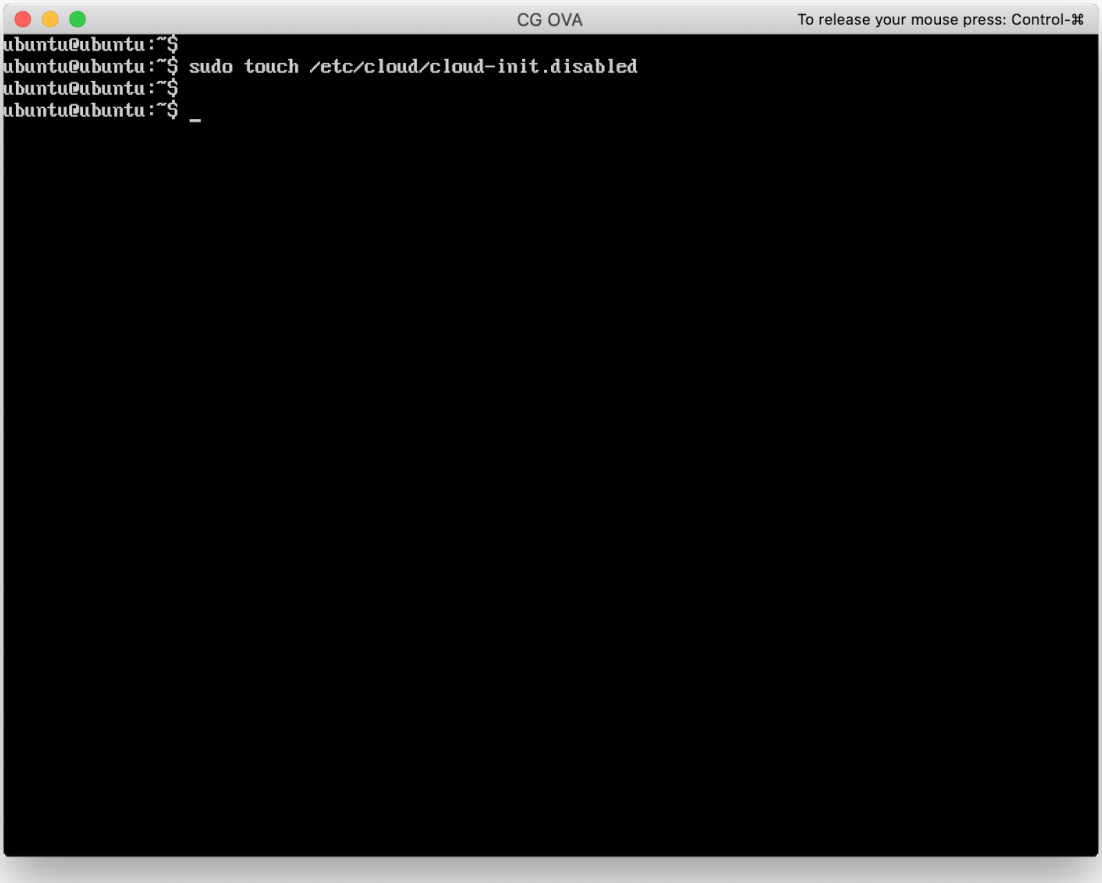
```
# Connect to example database on Primary server
psql -h PRIMARY_IP -p 5432 -U exampleuser -W example
# Password for user exampleuser:
```



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
```



Extensions

pg_trgm

Connect to the database as the superuser user (i.e postgres).

```
# Connect to postgres server as postgres user
sudo -u postgres psql
```

Enable the pg_trgm extension by running the following commands in the postgres shell.

```
# Enable pg_trgm extension
CREATE EXTENSION pg_trgm;

# Verify if the extension is successfully installed
SELECT * FROM pg_extension;
```