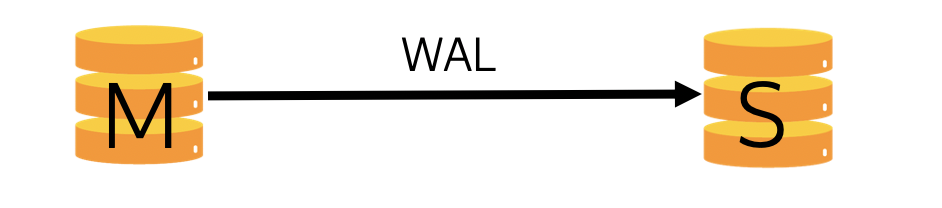
# Задание

Необходимо написать программу-агента, которая создаёт отказоустойчивый кластер PostgreSQL.

Топология кластера при этом получается примерно такой:



Здесь M - мастре репликации, в документации primary node. S - синхронная реплика, в документации standby with streaming replication and synchronous\_commit=on.

M и S находятся на различных сетевых хостах (виртуальных машинах, docker контейнерах или реальных физических серверах). Кроме хостов СУБД есть ещё хост-арбитр.

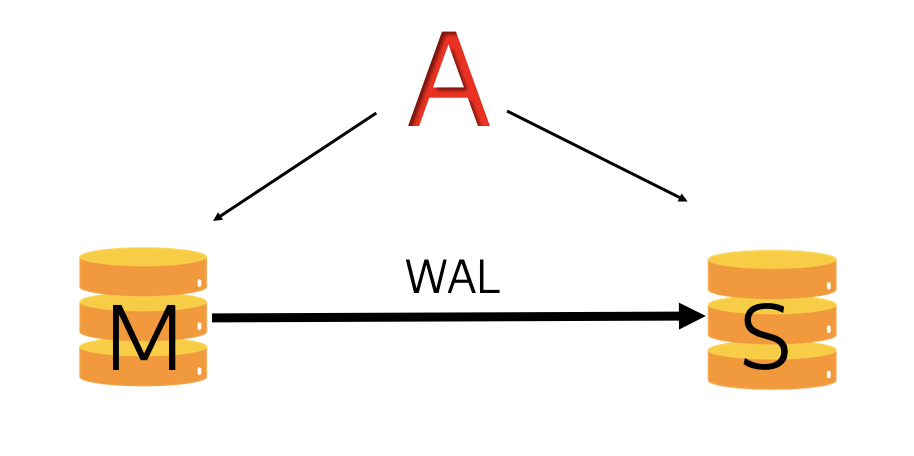
Процесс настройки потоковой репликации подробно и немного избыточно описан в <https://momjian.us/main/writings/pgsql/hot_streaming_rep.pdf>

Программа-агент запущена на каждом из хостов. Свои настройки получает из файлов конфигурации и безопасно для работоспособности системы переживает перезапуск.

Самое главное - программа-агент в случае потери сетевой связности мастера делает promote локальной СУБД до мастера. Для того чтобы проверить, что сетевая связность потеряна у M, а не у S, используется координирование с арбитром. Promote не происходит, если у S нет связи ни с M, ни с А. Promote происходит если у S нет связи с M и А подтверждает отсутствие связи с М.

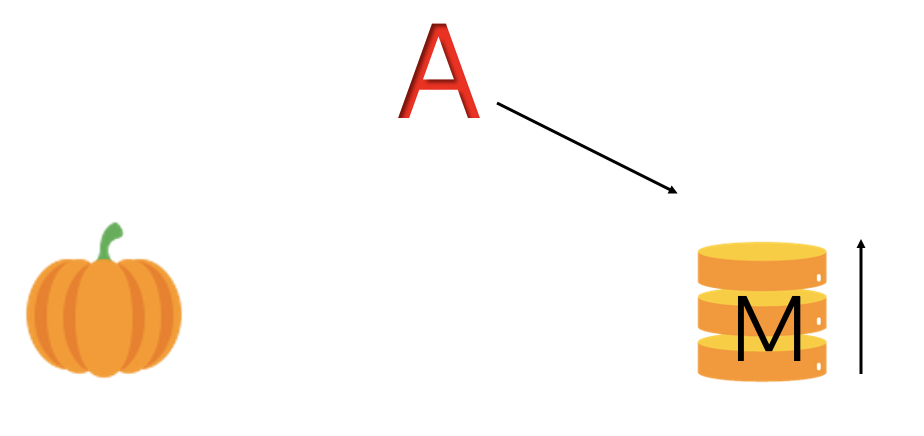
В случае потери связи с А, но признаков жизни на М promote также не происходит.

**Топология с А:**



В документации Арбитра иногда называют Свидетелем (witness).

**Топология promote**



Важно помнить, что М при отсутствии реплики и связи с арбитром тоже должен быть закрыт (например через iptables).

## Инструментарий

Утилиты PostgreSQL

initdb - создаёт пустую базу данных

pg\_ctl - запускает и останавливает базу данных, инициирует promote базе

pg\_basebackup - создаёт копию базы данных

pgisready - проверяет доступность базы данных

Для настройки реплики нужно создать файл recovery.conf и запустить базу, она начнёт процесс воспроизведения у себя транзакций с М.

Документация тут <https://postgrespro.ru/docs/postgrespro/11/app-pg-ctl>

Офф доеументация на английском <https://www.postgresql.org/docs/10/app-pg-ctl.html>

Признаки жизнедеятельности базы:

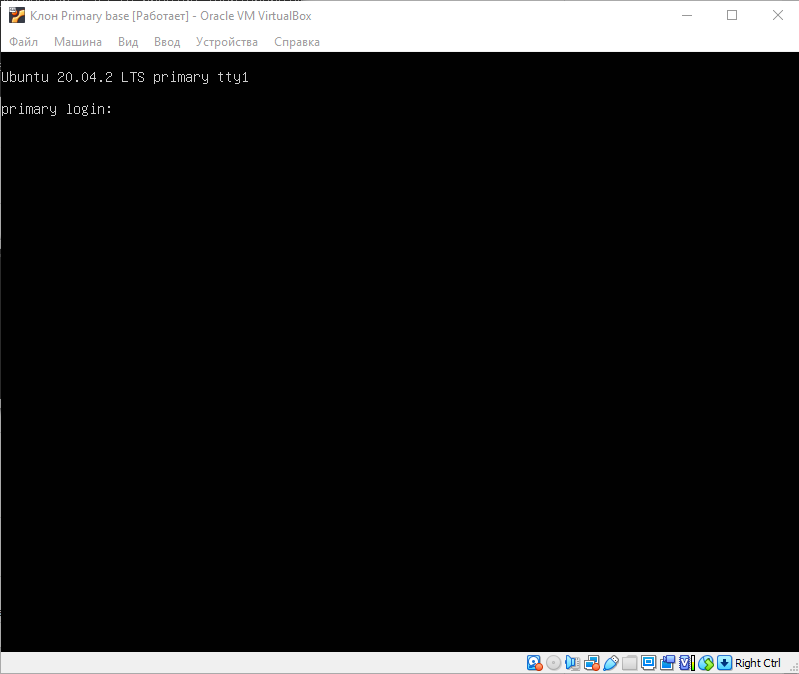
Можно подключиться к базе по порту 5432 и выполнить запрос “select 1;”

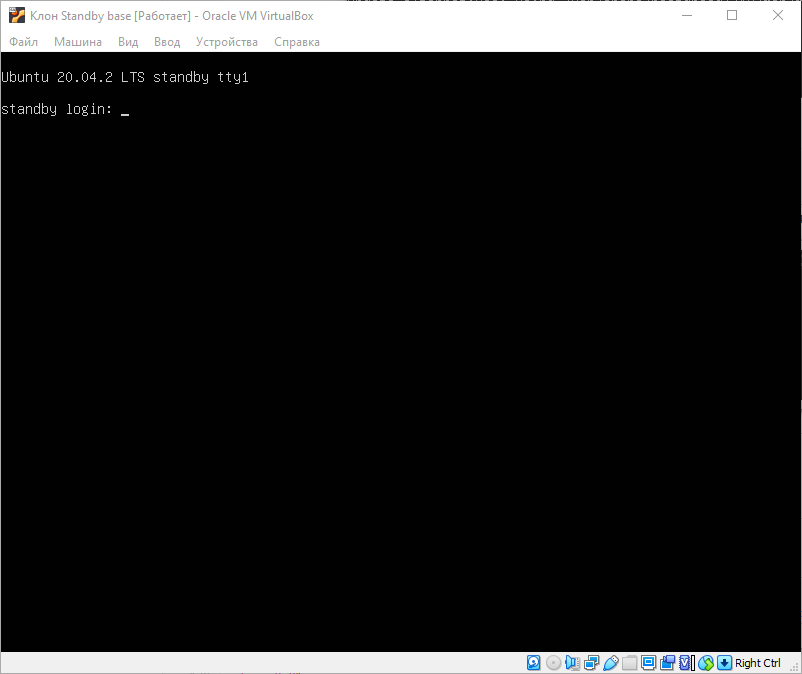
## Артефакты

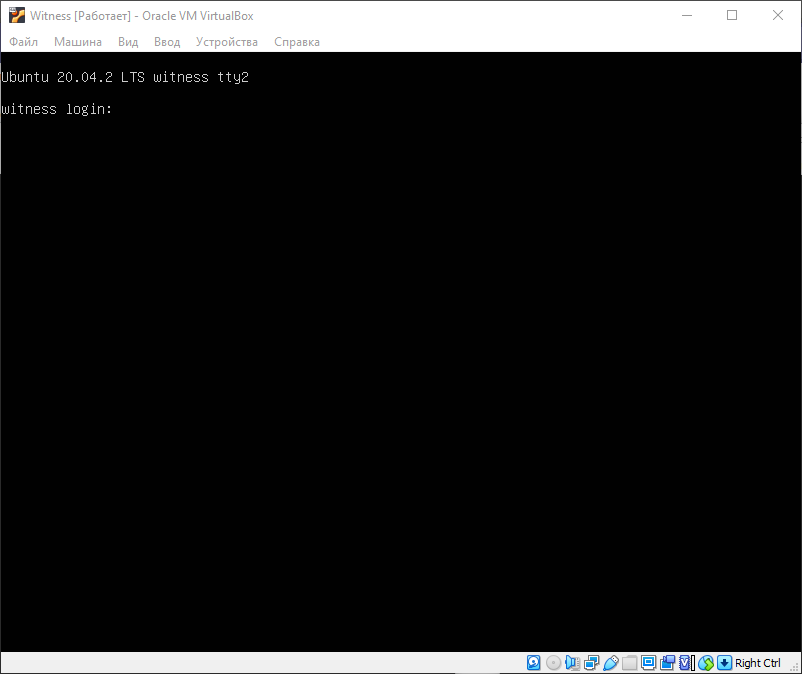
1. Программа-агент на github, gitlab, bb или pastebin
2. Собранный с её помощью стенд в котором можно оторвать сеть у M и проверить что в бывший S теперь можно выполнить пишущий запрос (например “create table x(i int);”)

# Исходные данные

Имеем 3 машины







# Первоначальная настройка

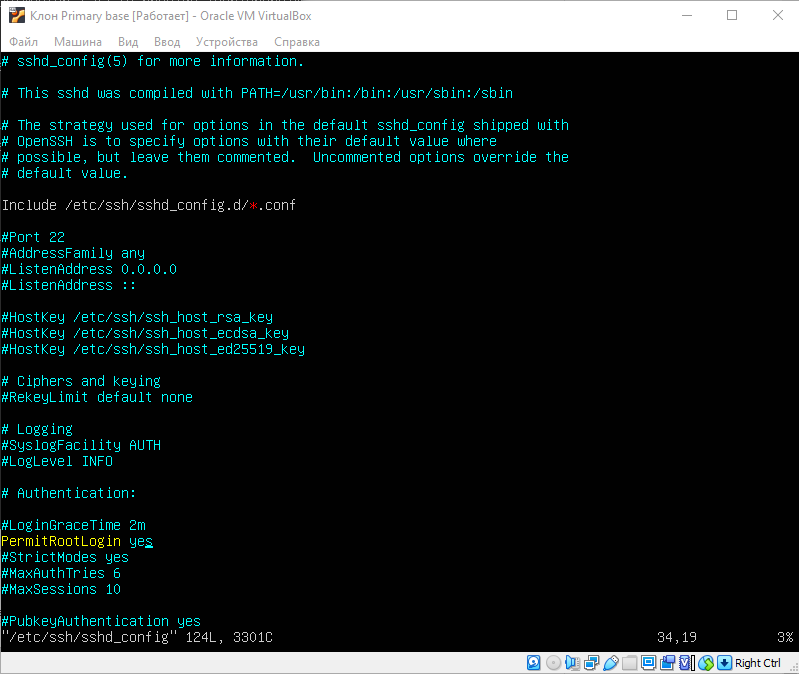
Предварительно на всех машинах зададим пароль для пользователя root

sudo passwd root

Разрешим root доступ по ssh

sudo vim /etc/ssh/sshd\_config

Отредактируем следующим образом строку



И перезапустим сервисы

systemctl restart ssh

systemctl restart sshd

На Primary сервере поднимем postgresql

sudo apt update

sudo apt upgrade

sudo apt install postgresql

# Скрипт

Создадим следующий скрипт на witness

#!/bin/bash

rep\_user="replica"

witnessIp=192.168.0.106

#primaryIp=192.168.0.105

#standbyIp=192.168.0.107

# Read IP address

read -p "Enter primary ip: " primaryIp

read -p "Enter standby ip: " standbyIp

ssh root@$primaryIp << EOT

echo "Connect to primary server"

systemctl enable postgresql

echo "Change postgresql.conf"

sed -i "s/#listen\_addresses = 'localhost'/listen\_addresses = '\*'/; s/#wal\_level = replica/wal\_level = replica/; s/#synchronous\_commit = on/synchronous\_commit = on/; s/#max\_wal\_senders = 10/max\_wal\_senders = 10/; s/#wal\_keep\_segments = 0/wal\_keep\_segments = 10/; s/#synchronous\_standby\_names = ''/synchronous\_standby\_names = '\*'/" /etc/postgresql/12/main/postgresql.conf

echo "Add rull to pg\_sql.conf"

if grep -Fxq 'host replication $rep\_user $standbyIp/32 trust' /etc/postgresql/12/main/pg\_hba.conf

then echo 'Rule exist'

else echo 'host replication $rep\_user $standbyIp/32 trust' >> /etc/postgresql/12/main/pg\_hba.conf

fi

if grep -Fxq 'host all all 0.0.0.0/0 md5' /etc/postgresql/12/main/pg\_hba.conf

then echo 'Rule exist'

else echo 'host all all 0.0.0.0/0 md5' >> /etc/postgresql/12/main/pg\_hba.conf

fi

echo "Create $rep\_user"

sudo -S -u postgres psql -c "CREATE USER $rep\_user REPLICATION PASSWORD 'A';"

systemctl restart postgresql

EOT

echo "Primary server finished"

ssh root@$standbyIp << EOT

echo "Connect to standby server"

apt update

apt upgrade -y

apt install -y postgresql

systemctl enable postgresql

echo "Posstgresql installed"

systemctl stop postgresql

rm -rf /var/lib/postgresql/12/main/

sudo -S -u postgres pg\_basebackup -R -h $primaryIp -U $rep\_user -D /var/lib/postgresql/12/main -P -w

echo "Backup is succsessful"

sed -i "s/#listen\_addresses = 'localhost'/listen\_addresses = '\*'/; s/#hot\_standby = on/hot\_standby = on/" /etc/postgresql/12/main/postgresql.conf

echo "Changed postgresql.conf"

systemctl start postgresql

echo "Create agent"

cat > /etc/init.d/script << AUF

#!/bin/bash

while true

do

if pg\_isready -h $primaryIp -q

then

echo "Primary is available"

else

if ping -c 1 -q $witnessIp

then

wall "Primary is not available. Promote StandbyIp"

sudo -S -u postgres psql -c "SELECT pg\_promote(false);"

break;

else

wall "Primary and witness is not available"

fi

fi

sleep 1

done

wall "final"

AUF

cat > /etc/systemd/system//script.service << AUF

[Unit]

Description=My service

After=network.target

[Service]

Type=oneshot

User=root

ExecStart=/etc/init.d/script

[Install]

WantedBy=multi-user.target

AUF

chmod +x /etc/init.d/script

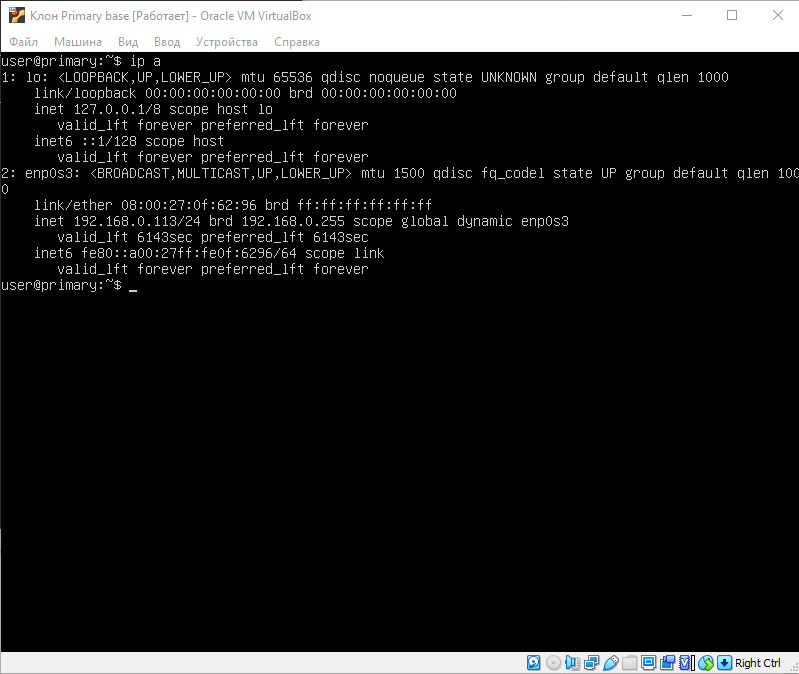
systemctl enable script.service

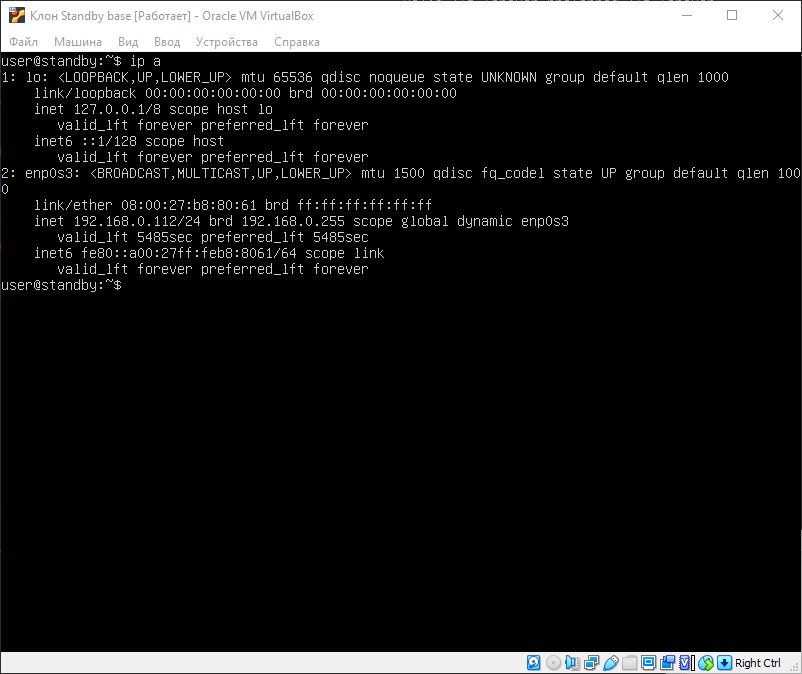
systemctl restart script.service

EOT

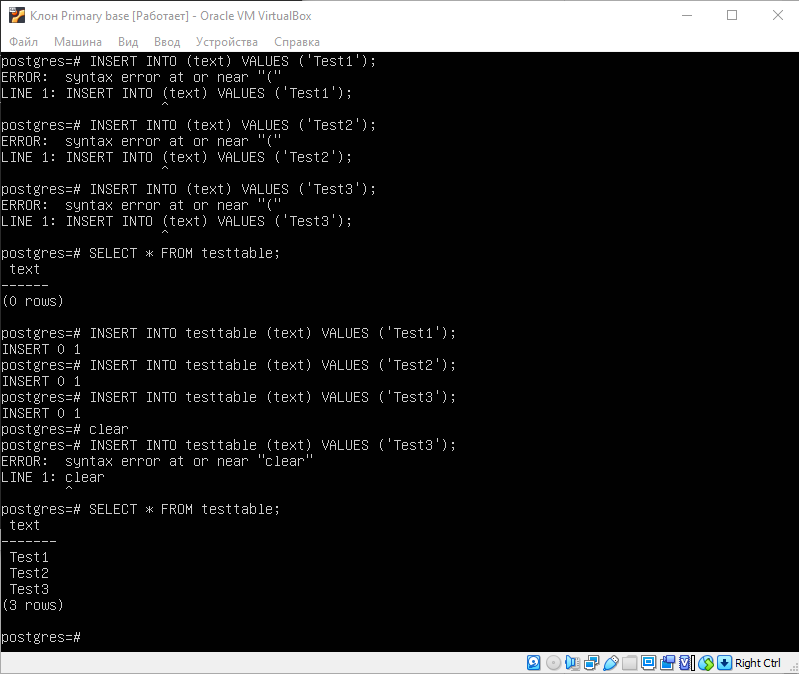
echo "StandBy server finished"

Посмотрим ip адреса

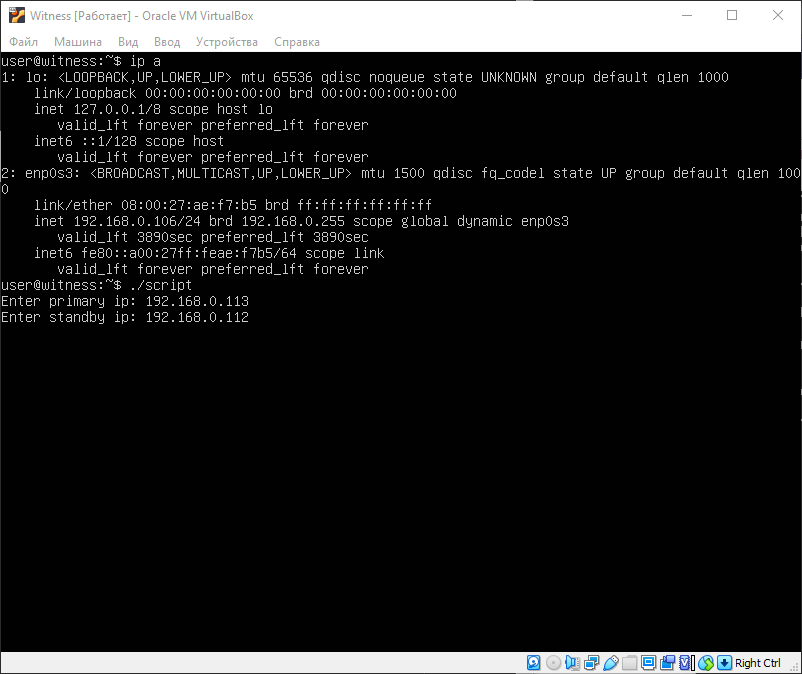




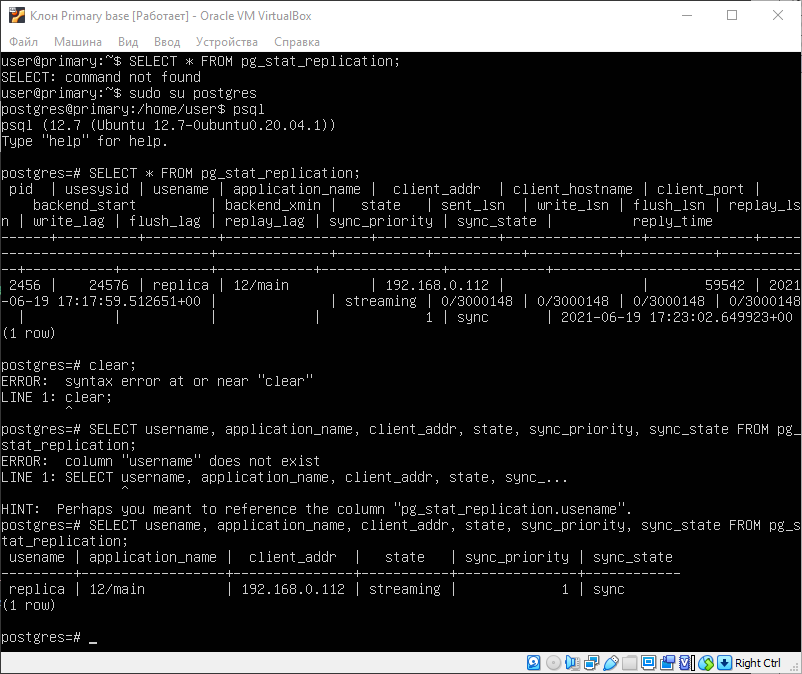
Так же добавим несколько строк в таблицу на primary сервере



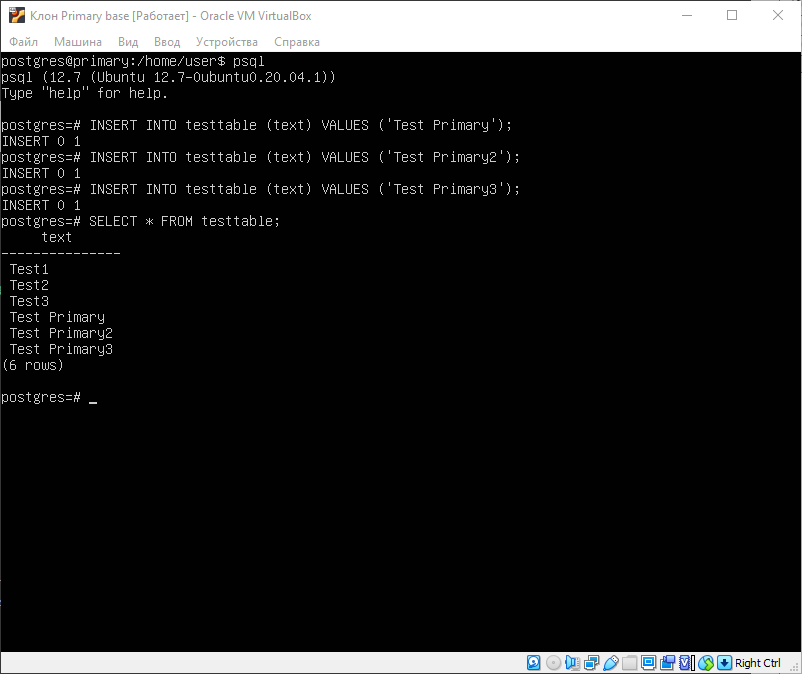
Запустим скрипт

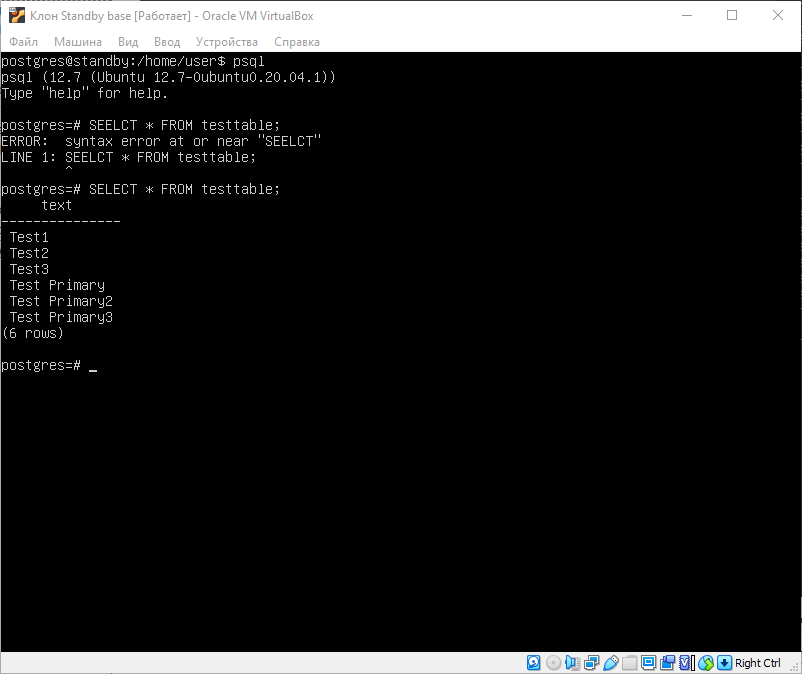


Проверим состояние репликации



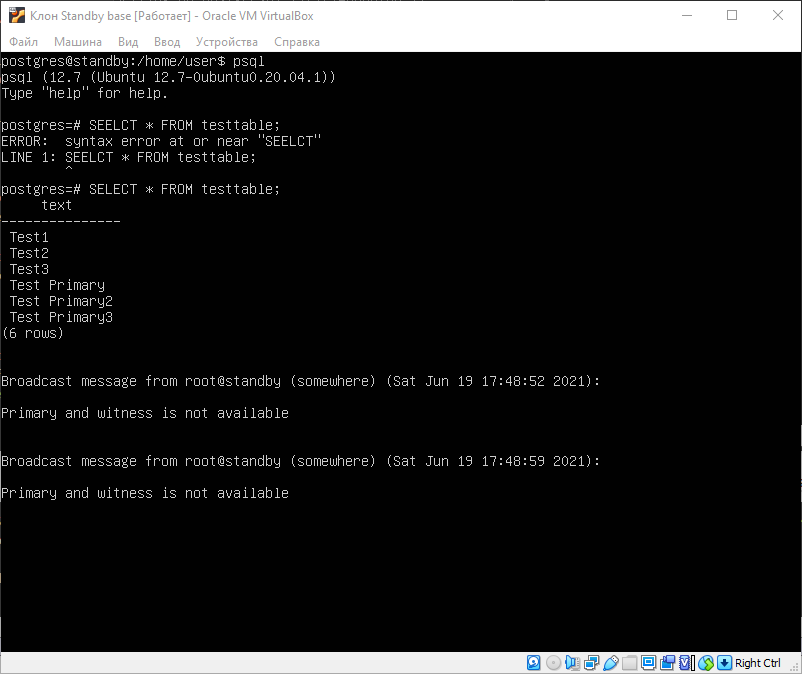
Так же проверим, добавив записей в БД





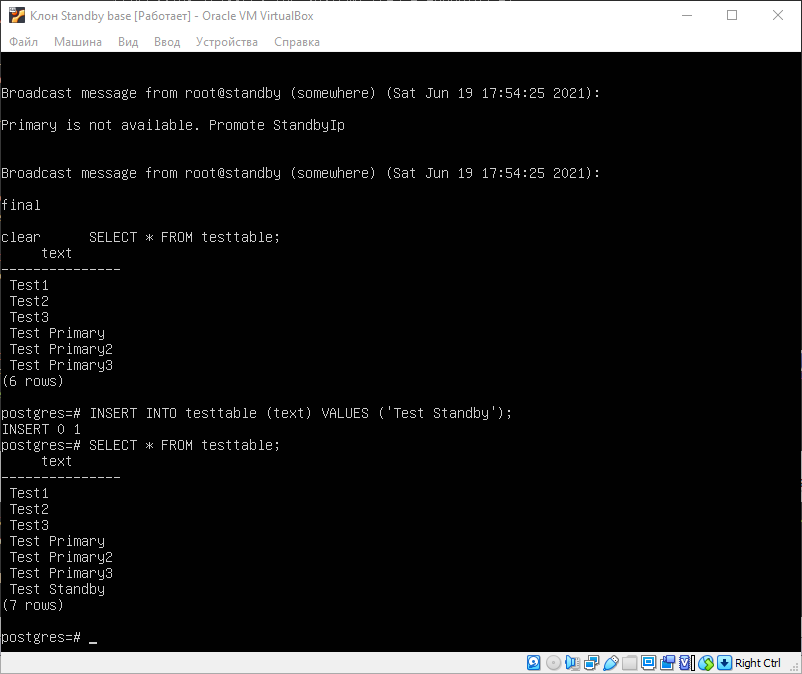
# Тестирование

Теперь проверим работу скрипта. Сперва отключим связь на standby.



Скрипт выполняется успешно.

Теперь проверим, отключив сеть на primary сервере



Как видим, standby сервер может выполнять пишущие запросы.