




ОНЛАЙН-ОБРАЗОВАНИЕ

# Онлайн-образование





# Меня хорошо видно && слышно?

Ставьте  , если все хорошо  
Напишите в чат, если есть проблемы





Защита проекта

# Отказоустойчивый корпоративный кластер PostgreSQL





# Постановка задачи. Инфраструктура

1/6

У нас используется несколько (<10) экземпляров MS SQL Server в инфраструктуре MS, при этом:

- экземпляры СУБД (разных версий) работают под MS Hyper-V в нескольких VM Windows;
- в каждом экземпляре несколько БД;
- доступ к БД осуществляется из приложений на компьютерах пользователей под их правами (SSO на базе MS Active Directory (AD));
- права доступа к объектам БД розданы ролям, на которые назначены группы пользователей из MS AD;
- сами пользователи (поштучно) в СУБД не прописаны, но все получают доступ к серверу как члены группы «Пользователи домена», а далее на каждую БД уже через членство в специфических группах;
- между экземплярами есть возможность выполнять кросс-базные запросы в контексте безопасности подключенного пользователя, изменение критических таблиц логируется, при этом в журналах
- кол-во пользователей работающих в системе – тысячи.
- для управления группами доступа в AD используется внешняя система IdM

# Постановка задачи. Инфраструктура.

1/6

Задача повторить сложившуюся практику и для PostgreSQL (и добиться, чтоб PostgreSQL вошел в эту среду как еще один инстанс БД для обеспечения плавной миграции):

- Kerberos аутентификация;
- Настройка возможности запросов к MS SQL Server из PostgreSQL в контексте пользователя и обратно (если это вообще возможно);
- Динамическое назначение ролей пользователям согласно их членству в группах AD;
- Обеспечение возможности подключения большого одновременного кол-ва пользователей;

# Постановка задачи. Инфраструктура.

1/6

Инфраструктура:

- MS Hyper-V 2016
- MS Windows Server 2016
- MS SQL Server 2016 / 2012 / 2008
- MS Active Directory
- Изолированная среда - интернета нет совсем

Добавляем:

- Debian 11
- Postgres Pro Enterprise 15

Задачи:

- Обеспечить **бесшовную интеграцию** инстансов postgres в сложившуюся инфраструктуру MS
- Сделать доступ разработчиков к postgres такой же привычный как к SQL Server
- Минимизировать кол-во дополнительных сущностей вроде (логинов, паролей, адресов, ...)
- **Обеспечить отказоустойчивость**

# Постановка задачи. Инфраструктура.

1/6

```
wget --user xxxxx --ask-password https://repoe.postgrespro.ru/ent-15/keys/pgpro-repo-add.sh
```

```
sh pgpro-repo-add.sh
```

```
apt -y install postgrespro-ent-15
```

```
# Установится в каталог
```

```
ls /var/lib/pgpro/ent-15/data/
```

```
sudo -u postgres psql
```

```
# Разрешаем внешние подключения
```

```
sed -i
```

```
's/host      all                all                127.0.0.1\32          md5/host      all                all                0.0
```

```
.0.0\0          md5/g' /var/lib/pgpro/ent-15/data/pg_hba.conf
```

```
sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*'>/g" /var/lib/pgpro/ent-15/data/postgresql.conf
```

```
systemctl restart postgrespro-ent-15
```

# Active Directory authentication

2/6

## Задача:

- Обеспечить возможность текущим пользователям и сервисным учетным записям "ходить" в БД postgres аналогично MS SQL:
- Пользователь входит в домен AD при старте рабочей станции
- SSO (Single-Sign-On)
- Централизованная система управления доступом на базе групп AD
- Credential "проброшены" до уровня БД, хранимые процедуры работают с учетом контекста пользователя

## План:

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD для postgres
- Прикрutим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres



# Active Directory authentication

2/6

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD
- Прикрutим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres

```
# установим krb5
apt install -y krb5-user

cat << EOF >> /etc/krb5.conf
[libdefaults]
    default_realm = XXXXX.RU

# The following krb5.conf variables are only for MIT Kerberos.
    kdc_timesync = 1
    ccache_type = 4
    forwardable = true
    proxiable = true

# The following libdefaults parameters are only for Heimdal Kerberos.
    fcc-mit-ticketflags = true

[realms]
    XXXXX.RU= {
        kdc = vidc2.xxxxx.ru
        admin_server = vidc2.xxxxx.ru
    }

[domain_realm]
    .xxxxx.ru = XXXXX.RU
    xxxxx.ru = XXXXX.RU
EOF
```

# Active Directory authentication

2/6

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD
- Прикрutим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres

```
kinit IvanovII XXXXX.RU
Password for IvanovII XXXXX.RU:

klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: IvanovII XXXXX.RU

Valid starting      Expires            Service
principal
06/07/2023 19:01:06 06/08/2023 05:01:06 krbtgt
XXXXX.RU
renew until 06/08/2023 19:00:58
```



# Active Directory authentication

2/6

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD
- Прикрutим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres

```
# выполняем в Powershell на Windows Server
# делаем пользователя в AD для службы PG Pro
New-ADUser -Name "pgpro" -GivenName "pgpro" -SamAccountName "pgpro" -UserPrincipalName
"pgpro@xxxxx.ru" -AccountPassword (ConvertTo-SecureString "cKFjW5b6LqkSTnp" -AsPlainText -
force) -Enabled $true
Get-ADUser "pgpro" | Set-ADUser -PasswordNeverExpires:$True -CannotChangePassword:$true

# create DNS record A
Add-DnsServerResourceRecordA -Name vpgpro -IPv4Address 172.16.20.1 -ZoneName xxxxx.ru

# create SPN
setspn -A POSTGRES/vpgpro.xxxxx.ru XXXXX.RU pgpro

# генерируем keytab
# если в пути содержится кириллица, то на время генерации keytab перенести в Users, потом
можно обратно
ktpass.exe -princ postgres/vpgpro.xxxxx.ru XXXXX.RU -ptype KRB5_NT_PRINCIPAL -crypto ALL -
mapuser pgpro@xxxxx.ru -pass cKFjW5b6LqkSTnp -out C:\krb5.keytab
```

# Active Directory authentication

2/6

- Настроим kerberos клиента linux

```
# копируем файл на linux
scp krb5.keytab ae@vpgpro:/home/ae
```
- Проверим kerberos на linux

```
# смотрим в psql где postgres ждет файл keytab
show krb_server_keyfile;
# /opt/pgpro/ent-15/etc/krb5.keytab
```
- Создадим объекты в AD
- Прикроем keytab файл в linux

```
# скопируем туда keytab и защитим его
cp pgpro-krb5.keytab /opt/pgpro/ent-15/etc/krb5.keytab
```
- Настроим postgresql на авторизацию GSSAPI

```
# проверим
kinit -V -k -t /opt/pgpro/ent-15/etc/krb5.keytab POSTGRES/vpgpro.xxxxx.ru XXXXX.RU
# klist покажет тикет
```
- Проверим с windows доступ к postgres

```
# keytab защитим
chmod 600 /opt/pgpro/ent-15/etc/krb5.keytab
chown postgres:postgres /opt/pgpro/ent-15/etc/krb5.keytab
```



# Active Directory authentication

2/6

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD
- Прикрutим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres

# правим разрешения подключения

```
echo "host all all 0.0.0.0/0 gss include_realm=0 krb_realm XXXXX.RU" >> pg_hba.conf
```

# если доменов много, то можно усложнить

# не обрезать REALM, а парсить их regex-ами

```
echo "host all all 0.0.0.0/0 gss include_realm=1 krb_realm XXXXX.RU map=gssmap" >> pg_hba.conf
```

# формула как выкусывать имя пользователя из его полного имени

```
echo "gssmap /^(.*)@XXXXX\.RU$ \1" >> pg_ident.conf
```

-- для AD users не важен регистр

```
ALTER SYSTEM SET krb_caseins_users=on;
```

-- Создадим пользователя под которым будет приходить из AD

```
CREATE ROLE ivanovii login WITH superuser;
```

```
```
```

```
``` sh
```

# postgres готов

```
systemctl restart postgrespro-ent-15
```

# Active Directory authentication

2/6

- Настроим kerberos клиента linux
- Проверим kerberos на linux
- Создадим объекты в AD
- Прикрутим keytab файл в linux
- Настроим postgresql на авторизацию GSSAPI
- Проверим с windows доступ к postgres

```
# ставим на Windows клиента psql
postgresql-15.3-1-windows-x64.exe
```

```
# запускаем
```

```
C:\Users\ivanovii>psql -h vpgpro.xxxxx.ru -d postgres
psql (15.3, server 15.2)
Type "help" for help.
```

```
# логин и пароль у меня не спросили
```

```
postgres=# \c
psql (15.3, server 15.2)
You are now connected to database "postgres" as user "ivanovii".
postgres=# select version();
```

```
version
```

```
-----
PostgreSQL 15.2 on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.1-6) 10.2.1 20210110,
64-bit
(1 row)
```



# Запросы между СУБД

3/6

- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

- **Установим tds-fdw**

TDS Foreign data wrapper

This is a PostgreSQL foreign data wrapper that can connect to databases that use the Tabular Data Stream (TDS) protocol, such as Sybase databases and Microsoft SQL server.

- Настроим SQL Server

- Настроим extension tds\_fdw в postgresql

```
# https://github.com/tds-fdw/tds\_fdw
```

- Проверка: запрос из postgres в SQL Server

```
apt install -y postgrespro-ent-15-dev  
apt install -y gcc gnupg make git  
apt install -y libsybdb5 freetds-dev freetds-common
```

- Настроим postgres для подключения из SQL Server

```
git clone https://github.com/tds-fdw/tds_fdw.git
```

- Установим на windows ODBC driver

```
cd tds_fdw  
make USE_PGXS=1  
make USE_PGXS=1 install
```

- Настроим linked server в SQL Server

```
cat << EOF >> /etc/freetds/freetds.conf  
[mssql_test]  
    host = sql-2016  
    #port = 1433  
    tds version = 7.3  
    instance = dev
```

```
EOF
```

- Проверка: запрос из SQL Server в postgres



- Установим tds-fdw
- **Настроим SQL Server**
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

```
-- Пользователь для подключения из postgres
CREATE LOGIN [pguser_test] WITH PASSWORD='P@$word'

-- демо таблички
CREATE TABLE [dbo].[table1_test](
    [id] [uniqueidentifier] NOT NULL,
    [name] [nchar](20) NULL
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[table1_test] ADD CONSTRAINT
[DF_table1_test_id] DEFAULT (newid()) FOR [id]
GO

CREATE TABLE [dbo].[table2_test](
    [name] [nchar](20) NULL
) ON [PRIMARY]
GO

INSERT INTO table1_test (name) values ('First')
INSERT INTO table1_test (name) values ('Second')
```

- Установим tds-fdw
- Настроим SQL Server
- **Настроим extension tds\_fdw в postgresql**
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

-- СУБД postgres

```
CREATE EXTENSION if not exists tds_fdw schema public;
```

```
CREATE SERVER mssql FOREIGN DATA WRAPPER tds_fdw OPTIONS  
(servername 'mssql_test', database 'integration_test');
```

```
CREATE USER MAPPING FOR CURRENT_USER SERVER mssql OPTIONS  
(username 'pguser_test', password 'P@$w0rd');
```

```
create schema ms;
```

```
IMPORT FOREIGN SCHEMA schema_test FROM SERVER mssql INTO ms  
OPTIONS (import_default 'true');
```

- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- **Проверка: запрос из postgres в SQL Server**
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

```
postgres=# select * from ms.table1_test;
NOTICE: tds_fdw: Query executed correctly
NOTICE: tds_fdw: Getting results
```

id	name
f4fd2847-7007-46e5-8f5f-995d1ab656a3	First
d71bf2ec-d8f5-431a-9cde-c98bd71c1aca	Second

(2 rows)

```
postgres=# select * from ms.table2_test;
NOTICE: tds_fdw: Query executed correctly
NOTICE: tds_fdw: Getting results
```

id	name
----	------

(0 rows)



- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- **Настроим postgres для подключения из SQL Server**
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

-- Пользователь для SQL Server

```
CREATE USER msuser_test WITH PASSWORD 'P@$word'
```

```
ALTER USER msuser_test WITH superuser;
```

-- Сделаем табличек

```
postgres=# create table table3 (name char(10));
```

```
CREATE TABLE
```

```
postgres=# insert into table3 (name) values ('1111');
```

```
INSERT 0 1
```

```
postgres=# insert into table3 (name) values ('2222');
```

```
INSERT 0 1
```

```
postgres=#
```

# проверим что работает хотя бы psql из под Windows

```
PS C:\Program Files\PostgreSQL\15\bin> .\psql -h 172.16.20.1 -d postgres -p 5432 -U
```

```
msuser_test -W
```

```
Password:
```

```
psql (15.3, server 15.2)
```

```
WARNING: Console code page (866) differs from Windows code page (1251)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
```

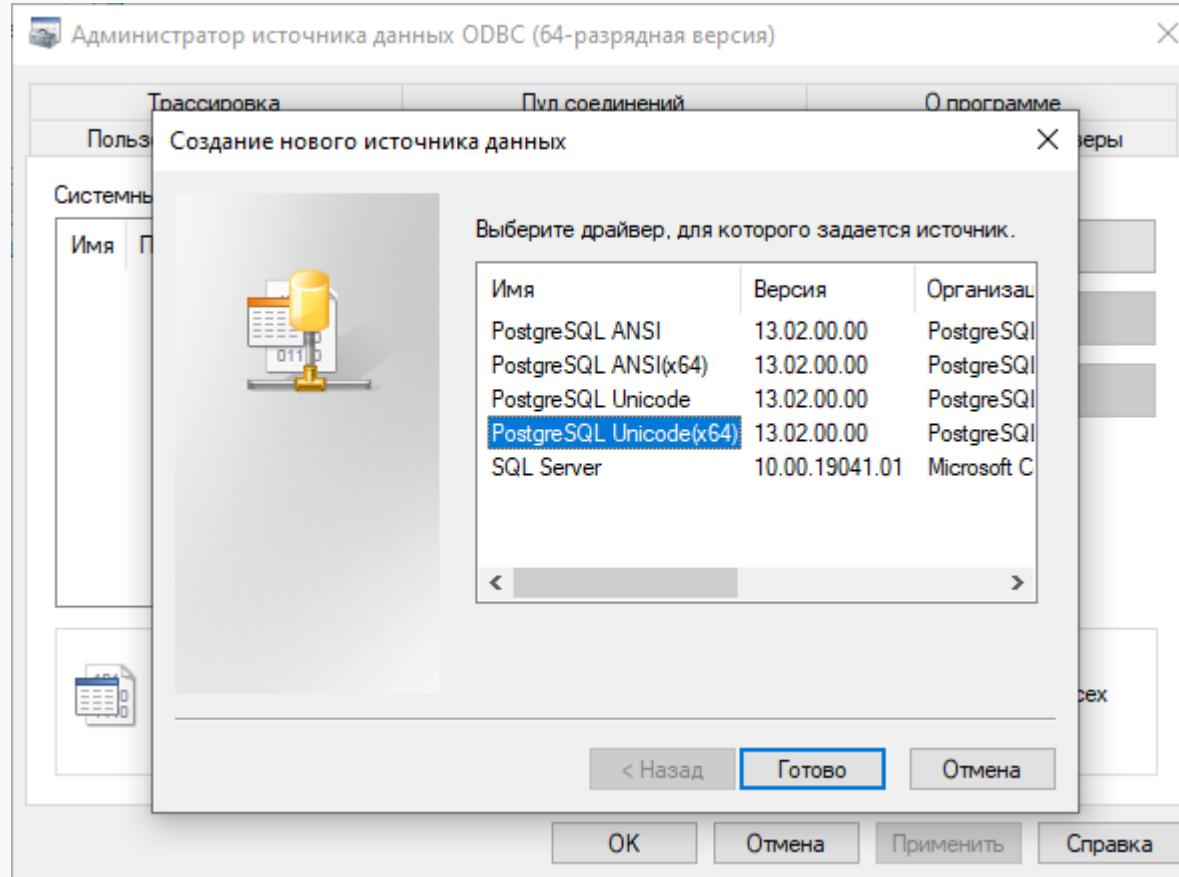
```
Type "help" for help.
```

# Запросы между СУБД

3/6

- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- **Установим на windows ODBC driver**
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

<https://www.postgresql.org/ftp/odbc/versions/msi/>

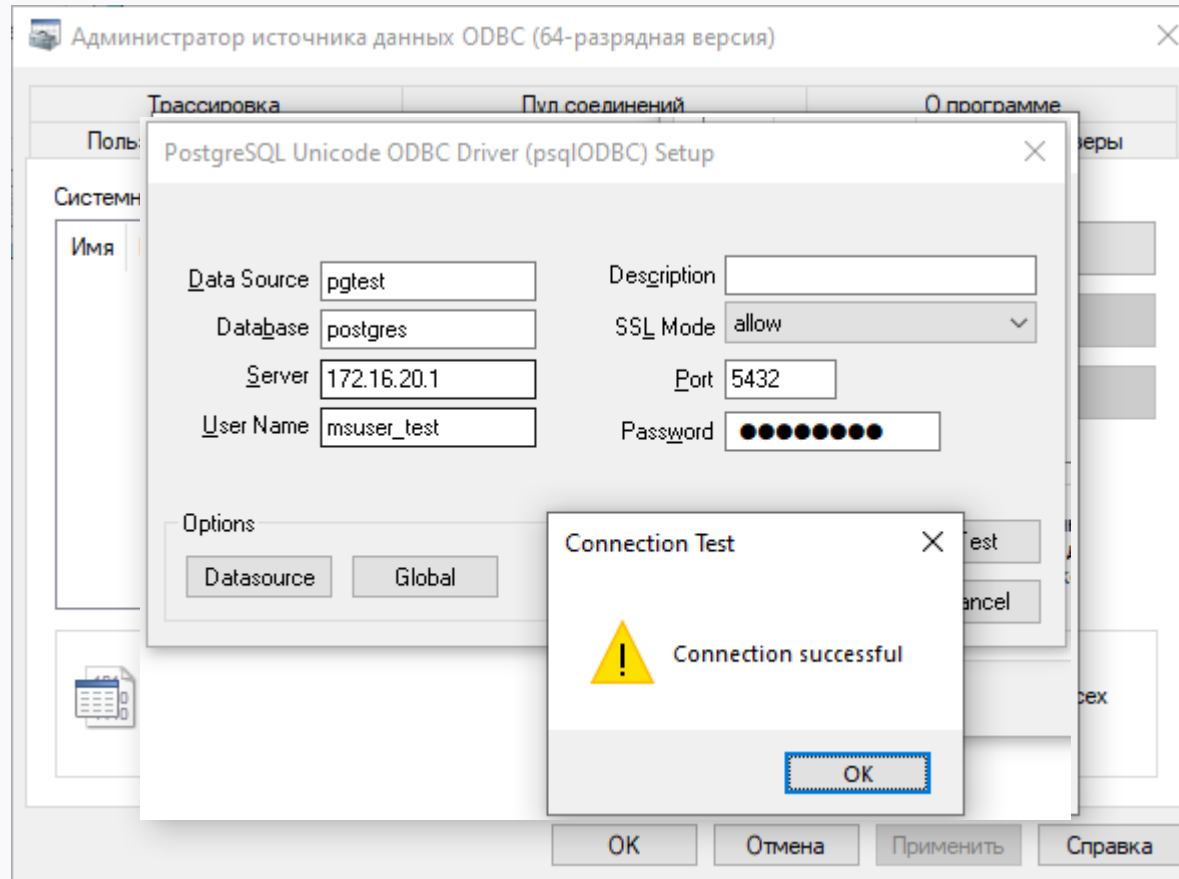


# Запросы между СУБД

3/6

- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- **Установим на windows ODBC driver**
- Настроим linked server в SQL Server
- Проверка: запрос из SQL Server в postgres

<https://www.postgresql.org/ftp/odbc/versions/msi/>





- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- **Настроим linked server в SQL Server**
- Проверка: запрос из SQL Server в postgres

```
EXEC master.dbo.sp_addlinkedserver @server = N'PGPRO', @srvproduct=N'Postgres PRO',  
@provider=N'MSDASQL', @datasrc=N'pgtest'
```

```
/* For security reasons the linked server remote logins password is changed with ##### */  
EXEC master.dbo.sp_addlinkedsrvlogin  
@rmtsrvname=N'PGPRO',@useself=N'False',@locallogin=NULL,@rmtuser=N'msuser_test',@rmtpassword  
='P@$w0rd'  
GO
```

# Запросы между СУБД

3/6

- Установим tds-fdw
- Настроим SQL Server
- Настроим extension tds\_fdw в postgresql
- Проверка: запрос из postgres в SQL Server
- Настроим postgres для подключения из SQL Server
- Установим на windows ODBC driver
- Настроим linked server в SQL Server
- **Проверка: запрос из SQL Server в postgres**

```
SELECT * FROM PGPRO.postgres.[public].table3
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to 'master' on a server named '(56)\*'. The Object Explorer on the left shows the server hierarchy, with the 'PGPRO' linked server expanded, showing its catalogs, including 'postgres', and its tables, including 'public.table3'. The main query window displays the SQL query: `SELECT * FROM PGPRO.postgres.[public].table3`. The Results pane at the bottom shows the output of the query, which consists of two rows of data.

	name
1	1111
2	2222

# Синхронизация AD и postgresql

4/6

Задача:

В AD существует структура групп обеспечивающих пользователей доступом в SQL. Необходимо эти группы отзеркалить в postgresql вместе с пользователями и их членством в группах.

План:

- Установим pg\_ldap\_sync
- Файл настройки yaml
- Запускаем синхронизацию и проверяем
- Правим AD, проверяем



- Установим  
pg ldap sync

This program helps to solve the issue by synchronizing users, groups and their memberships from LDAP to PostgreSQL.

<https://github.com/larskanis/pg-ldap-sync>

- Файл настройки  
yam1

```
# prerequisites
```

```
apt-get install -y ruby libpq-dev
```

- Запускаем  
синхронизацию  
и проверяем

```
# pg-ldap-sync install from git
```

```
git clone https://github.com/larskanis/pg-ldap-sync.git
```

```
cd pg-ldap-sync
```

```
gem install bundler
```

```
bundle install
```

```
bundle exec rake install
```

- Правим AD,  
проверяем

```
-- создаем служебные user & group
```

```
postgres=# CREATE GROUP ldap_groups;
```

```
CREATE ROLE
```

```
postgres=# CREATE USER ldap_users;
```

```
CREATE ROLE
```

# Синхронизация AD и postgresql

4/6

- Установим `pg_ldap_sync`
  - Файл настройки  
`yaml`
  - Запускаем синхронизацию и проверяем
  - Правим AD, проверяем
- ```
# With this sample config the distinction between LDAP-synchronized
# groups/users from manually created PostgreSQL users is done by the
# membership in ldap_user and ldap_group.
# These two roles have to be defined manually before pg_ldap_sync can
# run and all synchronized users/groups will become member of them
# later on:
#     CREATE GROUP ldap_groups;
#     CREATE USER ldap_users;
#
# Connection parameters to LDAP server
# see also: http://net-ldap.rubyforge.org/Net/LDAP.html#method-c-new
ldap_connection:
  host: dc-1
  port: 389
  auth:
    method: :simple
    username: CN=pgpro,CN=Users,DC=xxxxx,DC=ru
    password: cKFj5b6LqkSTnp
```

# Синхронизация AD и postgresql

4/6

- Установим pg\_ldap\_sync
- Файл настройки yaml
- Запускаем синхронизацию и проверяем
- Правим AD, проверяем

```
# Search parameters for LDAP users which should be synchronized
ldap_users:
  base: OU=users,OU=_Services,DC=xxxxx,DC=ru
  # LDAP filter (according to RFC 2254)
  # defines to users in LDAP to be synchronized
  filter: (&(objectClass=person)(objectClass=organizationalPerson)(givenName=*)(sn=*)(sAMAccountName=*))
  # this attribute is used as PG role name
  name_attribute: sAMAccountName
  # lowercase name for use as PG role name
  lowercase_name: true
  # Add lowercase name *and* original name for use as PG role names (useful for migrating between case types)
  bothcase_name: false
# Search parameters for LDAP groups which should be synchronized
ldap_groups:
  base: OU=pgsql,OU=_Services,DC=xxxxx,DC=ru
  filter: (cn=*)
  # this attribute is used as PG role name
  name_attribute: cn
  # lowercase name for use as PG role name
  lowercase_name: false
  # this attribute must reference to all member DN's of the given group
  member_attribute: member
```



# Синхронизация AD и postgresql

4/6

- Установим  
pg\_ldap\_sync
- Файл настройки  
yaml
- Запускаем  
синхронизацию и  
проверяем
- Правим AD,  
проверяем

```
# Connection parameters to PostgreSQL server
# see also: http://rubydoc.info/gems/pg/PG/Connection#initialize-instance\_method
pg_connection:
  host: localhost
  dbname: postgres
  user: msuser_test
  password: P@$w0rd

pg_users:
  # Filter for identifying LDAP generated users in the database.
  # It's the WHERE-condition to "SELECT rolname, oid FROM pg_roles"
  filter: oid IN (SELECT pam.member FROM pg_auth_members pam JOIN pg_roles pr ON
pr.oid=pam.roleid WHERE pr.rolname='ldap_users')
  # Options for CREATE RULE statements
  create_options: LOGIN IN ROLE ldap_users

pg_groups:
  # Filter for identifying LDAP generated groups in the database.
  # It's the WHERE-condition to "SELECT rolname, oid FROM pg_roles"
  filter: oid IN (SELECT pam.member FROM pg_auth_members pam JOIN pg_roles pr ON
pr.oid=pam.roleid WHERE pr.rolname='ldap_groups')
  # Options for CREATE RULE statements
  create_options: NOLOGIN IN ROLE ldap_groups
  # Options for GRANT <role> TO <group> statements
  grant_options:
```

# Синхронизация AD и postgresql

4/6

- Установим `pg_ldap_sync`  
`pg_ldap_sync -c ad.yaml -vv -t`  
# реальные изменения
- Файл настройки `yaml`  
`pg_ldap_sync -c ad.yaml -vv`
- Запускаем синхронизацию и проверяем  

```
root@debian:/home/ae/pg-ldap-sync# pg_ldap_sync -c ad.yaml -vv
I, [2023-06-07T18:28:19.052281 #18211] INFO -- : found user-dn: CN=Ivanov
Ivan,OU=users,OU=_Services,DC=xxxxx,DC=ru
I, [2023-06-07T18:28:19.065129 #18211] INFO -- : found group-dn:
CN=sqlgroup_test_developers,OU=pgsql,OU=_Services,DC=xxxxx,DC=ru
I, [2023-06-07T18:28:19.065320 #18211] INFO -- : found group-dn:
CN=sqlgroup_test_users,OU=pgsql,OU=_Services,DC=xxxxx,DC=ru
I, [2023-06-07T18:28:19.072834 #18211] INFO -- : user stat: create: 1 drop: 0 keep: 0
I, [2023-06-07T18:28:19.072930 #18211] INFO -- : group stat: create: 2 drop: 0 keep: 0
I, [2023-06-07T18:28:19.072974 #18211] INFO -- : membership stat: grant: 1 revoke: 0 keep:
0
I, [2023-06-07T18:28:19.072994 #18211] INFO -- : SQL: CREATE ROLE "ivanovii" LOGIN IN ROLE
ldap_users
I, [2023-06-07T18:28:19.074962 #18211] INFO -- : SQL: CREATE ROLE
"sqlgroup_test_developers" NOLOGIN IN ROLE ldap_groups
I, [2023-06-07T18:28:19.076218 #18211] INFO -- : SQL: CREATE ROLE "sqlgroup_test_users"
NOLOGIN IN ROLE ldap_groups
I, [2023-06-07T18:28:19.077110 #18211] INFO -- : SQL: GRANT "sqlgroup_test_users" TO
"ivanovii"
```
- Правим AD, проверяем

# Синхронизация AD и postgresql

4/6

- Установим pg\_ldap\_sync
- Файл настройки yam1
- Запускаем синхронизацию и проверяем
- Правим AD, проверяем

postgres=# \du

| Role name                | List of roles<br>Attributes                                | Member of                        |
|--------------------------|------------------------------------------------------------|----------------------------------|
| ivanovii                 |                                                            | {ldap_users,sqlgroup_test_users} |
| ldap_groups              | Cannot login                                               | {}                               |
| ldap_users               |                                                            | {}                               |
| msuser_test              | Superuser                                                  | {}                               |
| postgres                 | Superuser, Create role, Create DB, Replication, Bypass RLS | {}                               |
| sqlgroup_test_developers | Cannot login                                               | {ldap_groups}                    |
| sqlgroup_test_users      | Cannot login                                               | {ldap_groups}                    |

# Синхронизация AD и postgresql

4/6

- Установим `pg_ldap_sync`  
root@debian:/home/ae/pg-ldap-sync# `pg_ldap_sync -c ad.yaml -vv`  
I, [2023-06-07T18:32:52.806432 #18248] INFO -- : found user-dn: CN=Ivanov Ivan,OU=users,OU=\_Services,DC=xxxxx,DC=ru
- Файл настройки `yaml`  
I, [2023-06-07T18:32:52.806918 #18248] INFO -- : found user-dn: CN=Petr Petrov,OU=users,OU=\_Services,DC=xxxxx,DC=ru
- Запускаем синхронизацию и проверяем  
I, [2023-06-07T18:32:52.818396 #18248] INFO -- : found group-dn: CN=sqlgroup\_test\_developers,OU=pgsql,OU=\_Services,DC=xxxxx,DC=ru  
I, [2023-06-07T18:32:52.818551 #18248] INFO -- : found group-dn: CN=sqlgroup\_test\_users,OU=pgsql,OU=\_Services,DC=xxxxx,DC=ru  
I, [2023-06-07T18:32:52.826221 #18248] INFO -- : found pg-user: "ivanovii"  
I, [2023-06-07T18:32:52.828017 #18248] INFO -- : found pg-group: "sqlgroup\_test\_developers" with members: []  
I, [2023-06-07T18:32:52.828350 #18248] INFO -- : found pg-group: "sqlgroup\_test\_users" with members: ["ivanovii"]  
I, [2023-06-07T18:32:52.828474 #18248] INFO -- : user stat: create: 1 drop: 0 keep: 1  
I, [2023-06-07T18:32:52.828544 #18248] INFO -- : group stat: create: 0 drop: 0 keep: 2  
I, [2023-06-07T18:32:52.828616 #18248] INFO -- : membership stat: grant: 2 revoke: 0 keep: 1  
I, [2023-06-07T18:32:52.828676 #18248] INFO -- : SQL: CREATE ROLE "petrovpp" LOGIN IN ROLE ldap\_users  
I, [2023-06-07T18:32:52.829118 #18248] INFO -- : SQL: GRANT "sqlgroup\_test\_developers" TO "petrovpp"  
I, [2023-06-07T18:32:52.829342 #18248] INFO -- : SQL: GRANT "sqlgroup\_test\_users" TO "petrovpp"
- Правим AD, проверяем



# Синхронизация AD и postgresql

4/6

- Установим `pg_ldap_sync`  
root@debian:/home/ae/pg-ldap-sync# `pg_ldap_sync -c ad.yaml -vv`  
I, [2023-06-07T18:33:18.703076 #18250] INFO -- : found user-dn: CN=Ivanov Ivan,OU=users,OU=\_Services,DC=xxxxx,DC=ru
- Файл настройки `yaml`  
I, [2023-06-07T18:33:18.703271 #18250] INFO -- : found user-dn: CN=Petr Petrov,OU=users,OU=\_Services,DC=xxxxx,DC=ru
- Запускаем синхронизацию и проверяем  
I, [2023-06-07T18:33:18.716282 #18250] INFO -- : found group-dn: CN=sqlgroup\_test\_developers,OU=pgsql,OU=\_Services,DC=xxxxx,DC=ru  
I, [2023-06-07T18:33:18.716442 #18250] INFO -- : found group-dn: CN=sqlgroup\_test\_users,OU=pgsql,OU=\_Services,DC=xxxxx,DC=ru
- Правим AD, проверяем  
I, [2023-06-07T18:33:18.724416 #18250] INFO -- : found pg-user: "ivanovii"  
I, [2023-06-07T18:33:18.724584 #18250] INFO -- : found pg-user: "petrovpp"  
I, [2023-06-07T18:33:18.726264 #18250] INFO -- : found pg-group: "sqlgroup\_test\_developers" with members: ["petrovpp"]  
I, [2023-06-07T18:33:18.727171 #18250] INFO -- : found pg-group: "sqlgroup\_test\_users" with members: ["ivanovii", "petrovpp"]  
I, [2023-06-07T18:33:18.727386 #18250] INFO -- : user stat: create: 0 drop: 0 keep: 2  
I, [2023-06-07T18:33:18.727467 #18250] INFO -- : group stat: create: 0 drop: 0 keep: 2  
I, [2023-06-07T18:33:18.727543 #18250] INFO -- : membership stat: grant: 0 revoke: 1 keep: 2  
I, [2023-06-07T18:33:18.727606 #18250] INFO -- : SQL: REVOKE "sqlgroup\_test\_users" FROM "ivanovii"

Задача:

Обеспечить большое кол-во подключений и минимизировать издержки, связанные с установлением новых подключений.

Будем пробовать встроенный в Postgres Pro Enterprise экспериментальный пул соединений:

<https://postgrespro.ru/docs/enterprise/11/connection-pooling>

И сравнивать его с pgbouncer для ванильного postgresql.

План:

- Установка postgresql
- Настройка pgbouncer
- Настройка Postgres Pro
- и его пулинга
- Результаты

Для OLTP на 2 CPU с 2GB RAM PG Tune насоветовал такие параметры:

```
max_connections = 300
shared_buffers = 512MB
effective_cache_size = 1536MB
maintenance_work_mem = 128MB
checkpoint_completion_target = 0.9
wal_buffers = 16MB
default_statistics_target = 100
random_page_cost = 1.1
effective_io_concurrency = 200
work_mem = 873kB
min_wal_size = 2GB
max_wal_size = 8GB
```

Поправим `max_connections` = **500** чтоб было наглядней.

Для тестирования подготовлено три VM Debian 11:

- postgresql
- Postgres Pro
- Client pgbench (не Windows)

Запускаем все это в VM под Hyper-V 2016 на Xeon Gold 6248R.

- Установка  
postgresql

- Настройка  
pgbouncer

- Настройка  
Postgres Pro

- и его пулинга

- Результаты

```
# postgresql 15
sudo apt-get install gnupg2 wget curl -y
echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list
wget --no-check-certificate --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |
sudo apt-key add -
sudo apt-get update -y && sudo apt-get upgrade -y

sudo apt-get install postgresql-15 -y
```



## - Установка postgresql

- Настройка  
pgbouncer

- Настройка  
Postgres Pro

- и его пулинга

- Результаты

```
cat << EOF >> /var/lib/postgresql/15/main/postgresql.auto.conf
```

```
max_connections = 500
```

```
shared_buffers = 512MB
```

```
effective_cache_size = 1536MB
```

```
maintenance_work_mem = 128MB
```

```
checkpoint_completion_target = 0.9
```

```
wal_buffers = 16MB
```

```
default_statistics_target = 100
```

```
random_page_cost = 1.1
```

```
effective_io_concurrency = 200
```

```
work_mem = 873kB
```

```
min_wal_size = 2GB
```

```
max_wal_size = 8GB
```

```
EOF
```

```
sed -i 's/host      all              all              127.0.0.1\|/32      scram-sha-256/host      all              all              0.0.0.0\|/0      scram-sha-256/g' /etc/postgresql/15/main/pg_hba.conf
```

```
sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*' /g" /etc/postgresql/15/main/postgresql.conf
```

```
systemctl restart postgresql
```

## - Установка postgresql

- Настройка  
pgbouncer

- Настройка  
Postgres Pro

- и его пулинга

- Результаты

# Создадим пользователя и БД для тестов

```
sudo -u postgres psql -c"CREATE USER pooltest WITH PASSWORD 'KJheysd123@';ALTER USER pooltest WITH superuser;"
```

```
sudo -u postgres psql -c"CREATE DATABASE pooltest;"
```

# иницилируем БД для тестов

```
pgbench -i -s 100 -h localhost -U pooltest -d pooltest
```

# запустим серию тестов

```
for i in {10..600..10}
```

```
do
```

```
    pgbench -c $i -P30 -T 150 -h xx.xx.xx.xx -p 5432 -U pooltest pooltest >> Result.txt
```

```
done
```

- Установка  
postgresql

```
apt install -y pgbouncer
```

- Настройка  
pgbouncer

```
cat << EOF >> /etc/pgbouncer# cat pgbouncer.ini
```

```
[databases]
```

```
* = host=localhost port=5432
```

```
[pgbouncer]
```

```
logfile = /var/log/postgresql/pgbouncer.log
```

```
pidfile = /var/run/postgresql/pgbouncer.pid
```

- Настройка  
Postgres Pro

- и его пулинга

```
listen_addr = *
```

```
listen_port = 6432
```

```
unix_socket_dir = /var/run/postgresql
```

- Результаты

```
auth_type = md5
```

```
auth_file = /etc/pgbouncer/userlist.txt
```

```
pool_mode = transaction
```

```
max_client_conn = 1000
```

```
default_pool_size = 70
```

```
EOF
```

```
cat << EOF >> /etc/pgbouncer/userlist.txt
```

```
"pooltest" "KJheysd123@"
```

```
EOF
```

```
systemctl restart pgbouncer
```

```
# и снова запустим тест
```

- Установка postgresql

- Настройка pgbouncer

- Настройка Postgres Pro

- и его пулинга

- Результаты

`session_pool_size (integer)`

>Включает пул соединений и определяет максимальное количество обслуживающих процессов, которые могут использоваться клиентскими сеансами для отдельно взятой базы данных.

`max_sessions (integer)`

>Максимальное количество клиентских сеансов, которые могут обслуживаться одним процессом при включении пула соединений. От этого параметра не зависит нагрузка процессора или использование памяти, так что и при большом значении `max_sessions` производительность не должна снизиться. При достижении предела `max_sessions` обслуживающий процесс прекращает принимать подключения и пока минимум одно из соединений не будет завершено, попытки подключиться к этому процессу будут вызывать ошибку. Значение по умолчанию – 1000. Этот параметр можно задать только при запуске сервера.



- Установка postgresql

- Настройка pgbouncer

- Настройка Postgres Pro

- и его пулинга

- Результаты

# Postgres Pro 15. Установка

```
wget --user xxyyzz --ask-password https://repoe.postgrespro.ru/ent-15/keys/pgpro-repo-add.sh
```

```
sh pgpro-repo-add.sh
```

```
apt -y install postgrespro-ent-15
```

```
ls /var/lib/pgpro/ent-15/data
```

```
cat << EOF >> /var/lib/pgpro/ent-15/data/postgresql.auto.conf
```

```
max_connections = 500
```

```
shared_buffers = 512MB
```

```
effective_cache_size = 1536MB
```

```
maintenance_work_mem = 128MB
```

```
checkpoint_completion_target = 0.9
```

```
wal_buffers = 16MB
```

```
default_statistics_target = 100
```

```
random_page_cost = 1.1
```

```
effective_io_concurrency = 200
```

```
work_mem = 873kB
```

```
min_wal_size = 2GB
```

```
max_wal_size = 8GB
```

```
EOF
```

- Установка postgresql

```
sed -i 's/host all all 127.0.0.1\32 md5/host all all 0.0.0.0\0 md5/g' /var/lib/pgpro/ent-15/data/pg_hba.conf
sed -i "s/#listen_addresses = 'localhost'/listen_addresses = '*' /g" /var/lib/pgpro/ent-15/data/postgresql.conf
```

- Настройка Postgres Pro

```
systemctl restart postgrespro-ent-15
```

```
# и снова запустим тест
```

- и его пулинга

- Результаты

- Установка postgresql

```
cat << EOF >> /var/lib/pgpro/ent-15/data/postgresql.auto.conf
```

```
session_pool_size = 70
```

```
EOF
```

- Настройка pgbouncer

```
# Примечательно, что подключение к пулу в Postgres Pro нативное, т.е. используются те же pg_hba.conf и работает Active Directory SSO!
```

- Настройка Postgres Pro

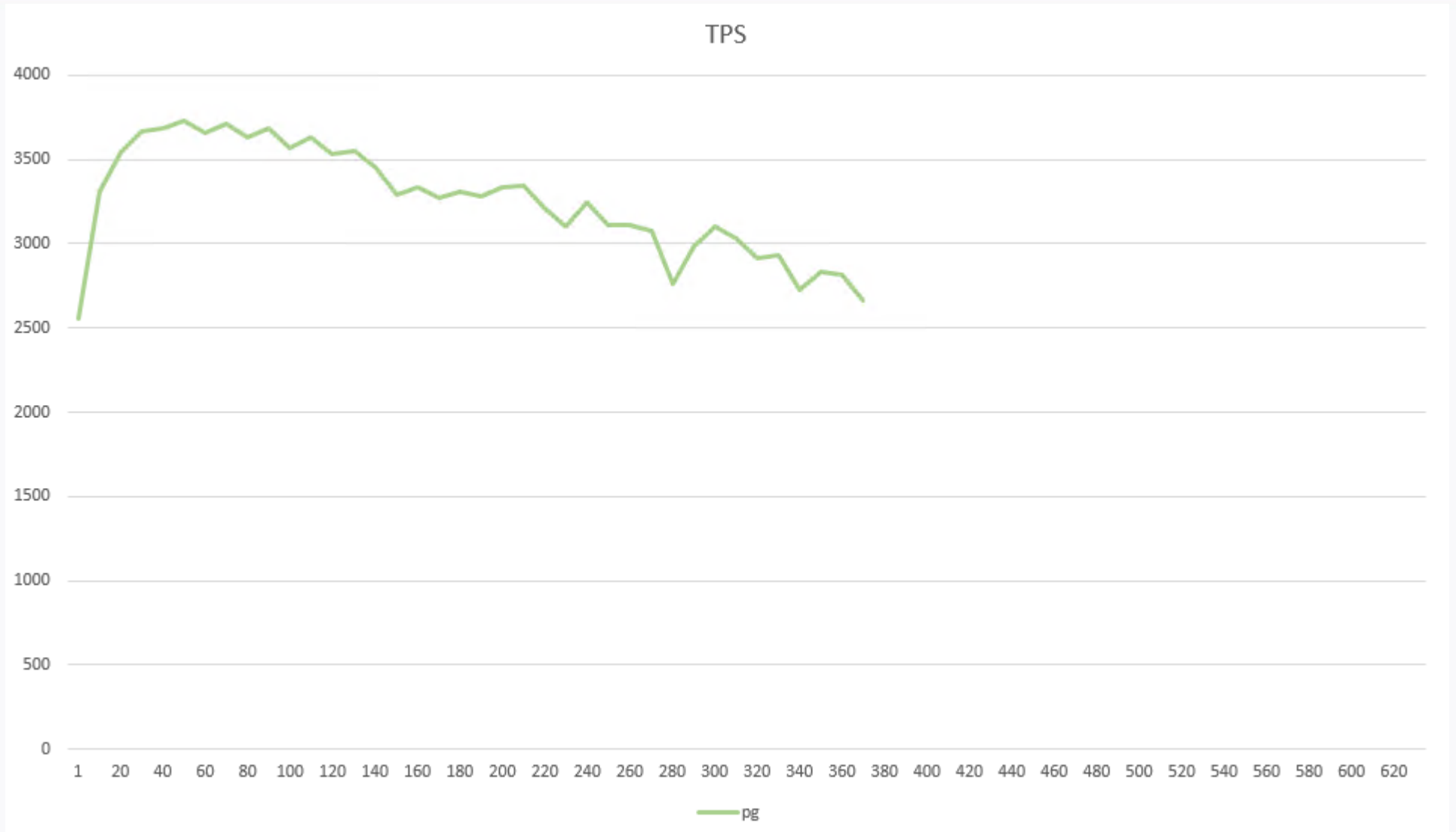
- и его пулинга

```
# и снова запустим тест
```

- Результаты

- Установка postgresql
- Настройка pgbouncer
- Настройка Postgres Pro
- и его пулинга
- Результаты

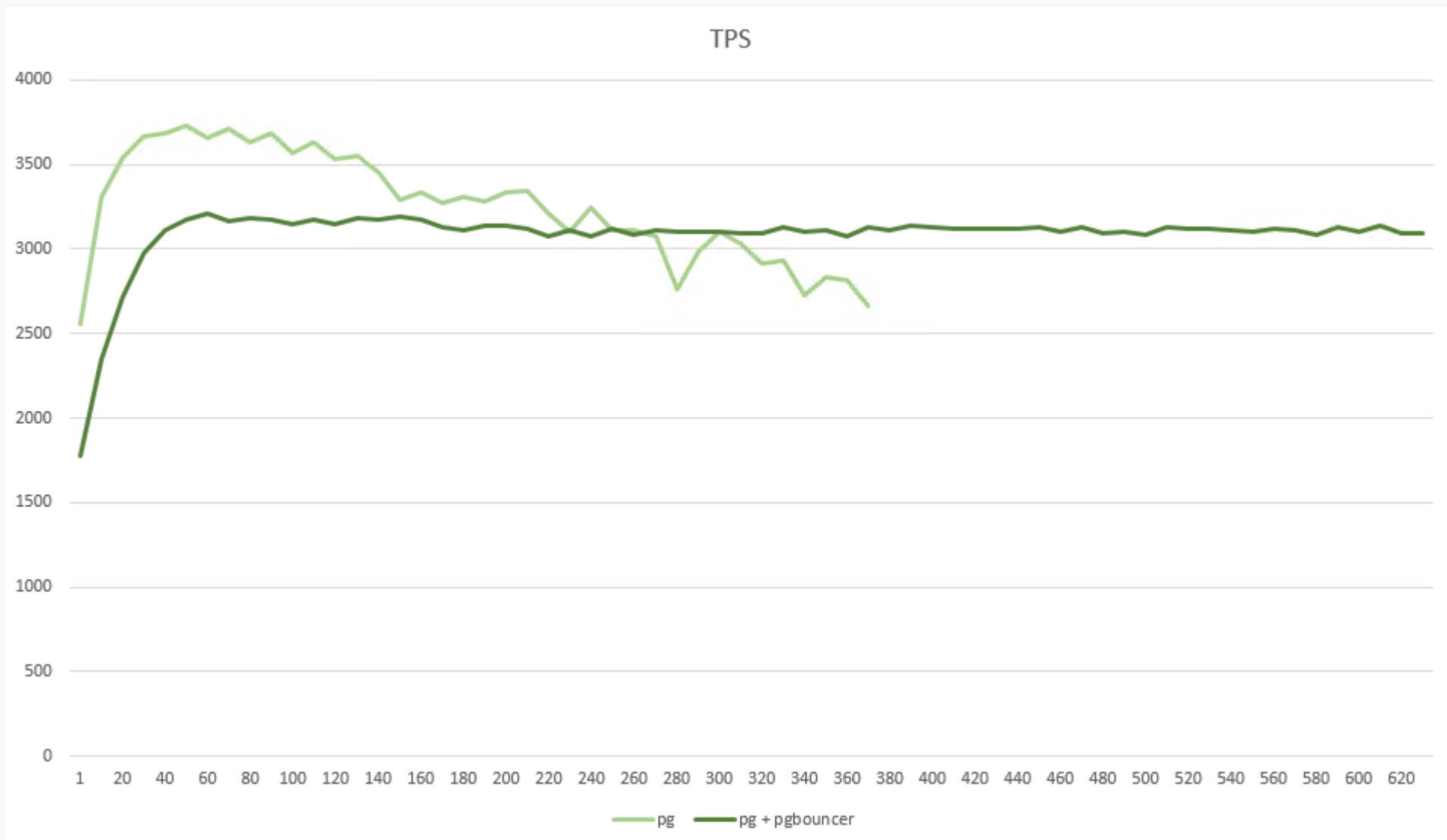
2 x Xeon Gold 6248R  
2 GB RAM





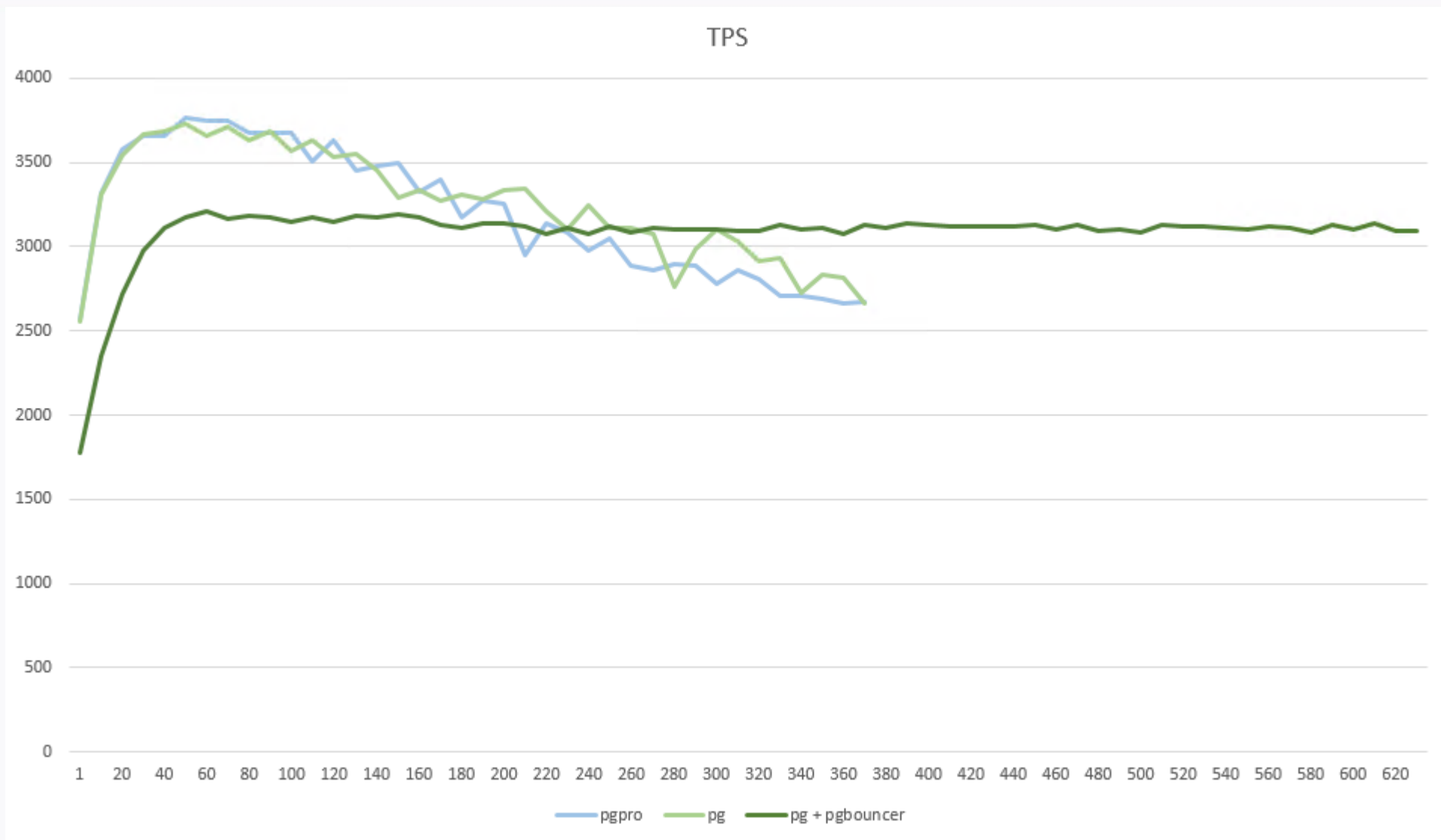
- Установка postgresql
- Настройка pgbouncer
- Настройка Postgres Pro
- и его пулинга
- Результаты

2 x Xeon Gold 6248R  
2 GB RAM



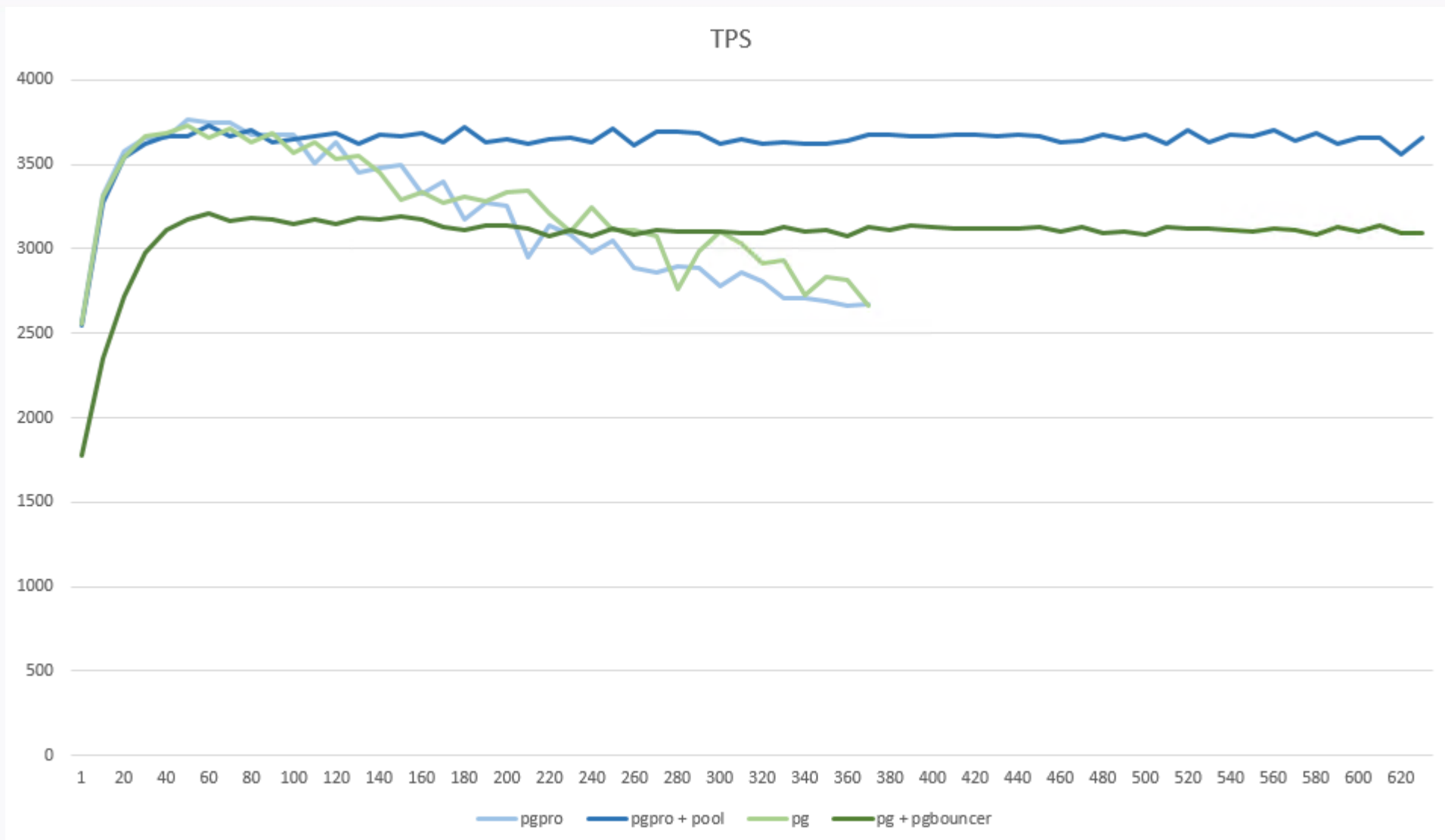
- Установка postgresql
- Настройка pgbouncer
- Настройка Postgres Pro
- и его пулинга
- Результаты

2 x Xeon Gold 6248R  
2 GB RAM



- Установка postgresql
- Настройка pgbouncer
- Настройка Postgres Pro
- и его пулинга
- Результаты

2 x Xeon Gold 6248R  
2 GB RAM



- Что это и какие ограничения

multimaster – это расширение Postgres Pro Enterprise, которое в сочетании с набором доработок ядра превращает Postgres Pro Enterprise в синхронный кластер без разделения ресурсов, который обеспечивает масштабируемость OLTP для читающих транзакций, а также высокую степень доступности с автоматическим восстановлением после сбоев.

- Настройка

- Результаты

<https://postgrespro.ru/docs/enterprise/11/multimaster>

Операционная система Microsoft Windows не поддерживается.

Решения 1С по ряду причин не поддерживаются.

multimaster может реплицировать только одну базу данных в кластере. Если требуется реплицировать содержимое нескольких баз данных, вы можете либо перенести все данные в разные схемы одной базы данных, либо создать для каждой базы отдельный кластер и настроить multimaster в каждом из этих кластеров.



- Что это и какие ограничения

- Настройка

- Результаты

```
shared_preload_libraries = 'multimaster'
```

```
wal_level = logical
```

```
max_connections = 100
```

```
max_prepared_transactions = 300 # max_connections * N
```

```
max_wal_senders = 10 # как минимум N
```

```
max_replication_slots = 10 # как минимум 2N
```

```
wal_sender_timeout = 0
```

```
max_worker_processes = 250 # (N - 1) * (max_connections + 3) + 3
```

```
```
```

```
``` sql
```

```
\c pooltest
```

```
CREATE EXTENSION multimaster;
```

```
SELECT mtm.init_cluster('dbname=pooltest user=pooltest host=b-pg-01',  
'{"dbname=pooltest user=pooltest host=b-pg-02", "dbname=pooltest user=pooltest host=b-pg-03"}');
```

```
SELECT * FROM mtm.status();
```

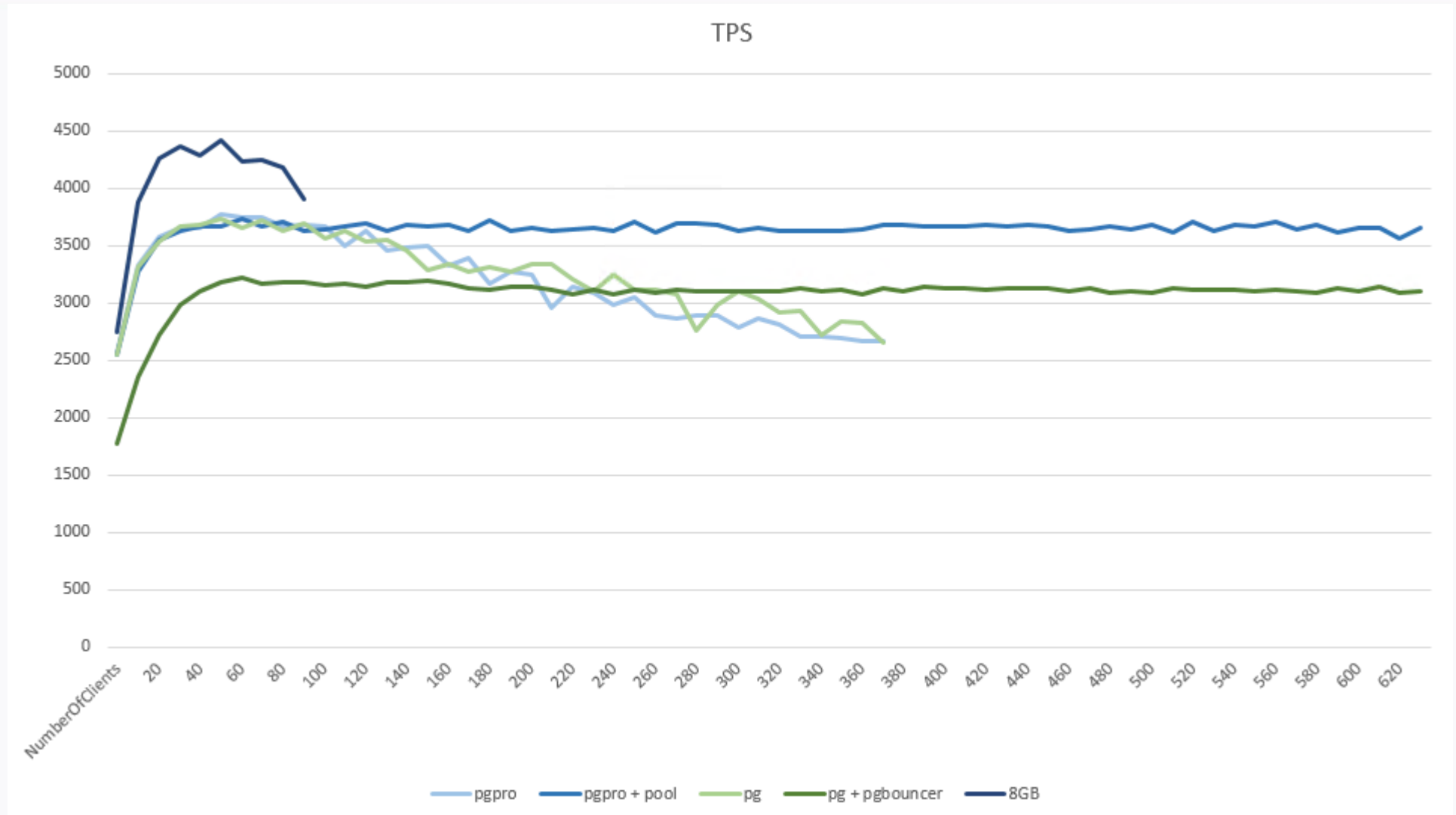
```
SELECT * FROM mtm.nodes();
```

# PostgreSQL Pro Multimaster

5/6

- Что это и какие ограничения
- Настройка
- Результаты

2 x Xeon Gold 6248R  
8 GB RAM

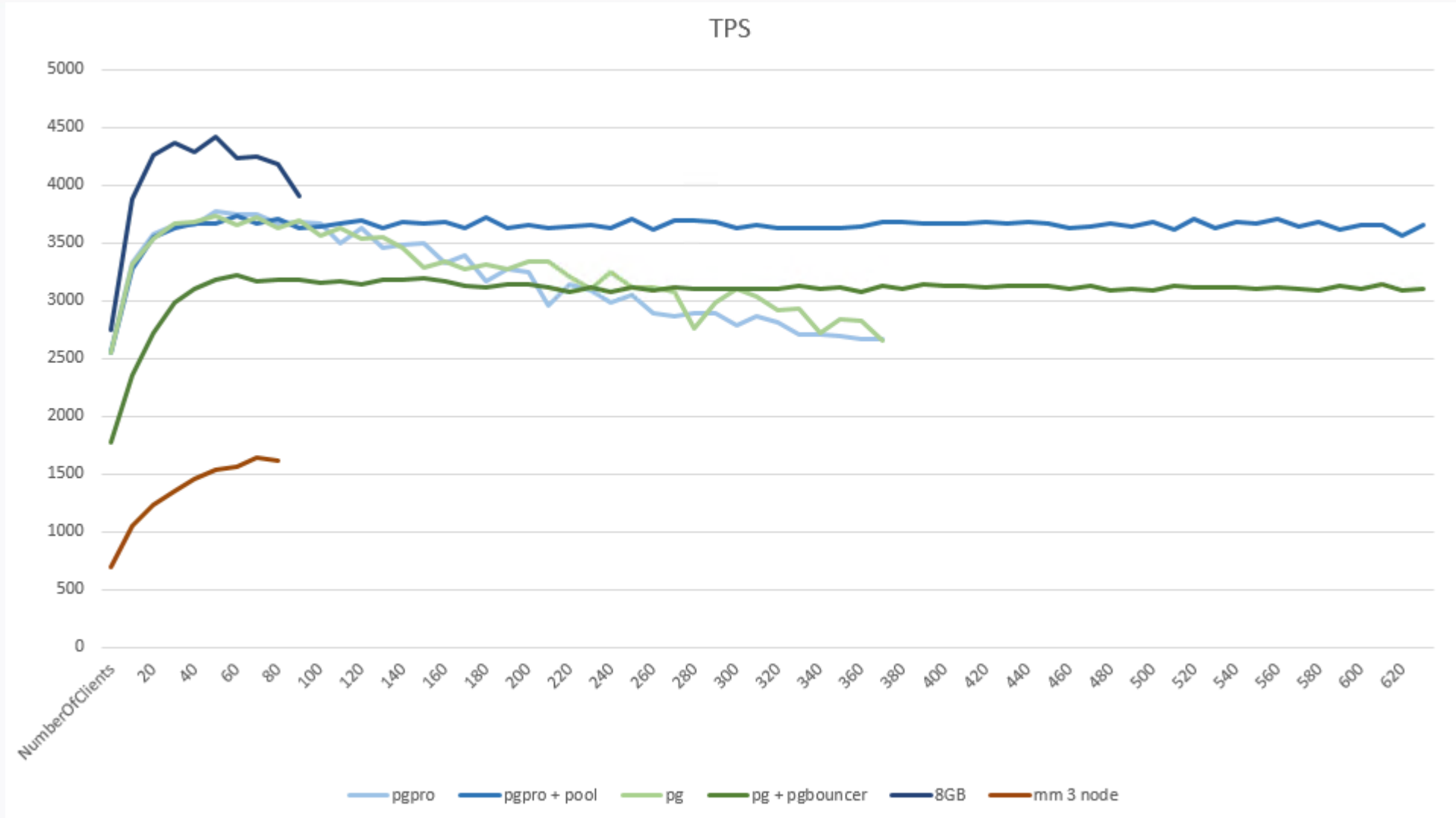


# PostgreSQL Pro Multimaster

5/6

- Что это и какие ограничения
- Настройка
- Результаты

2 x Xeon Gold 6248R  
8 GB RAM

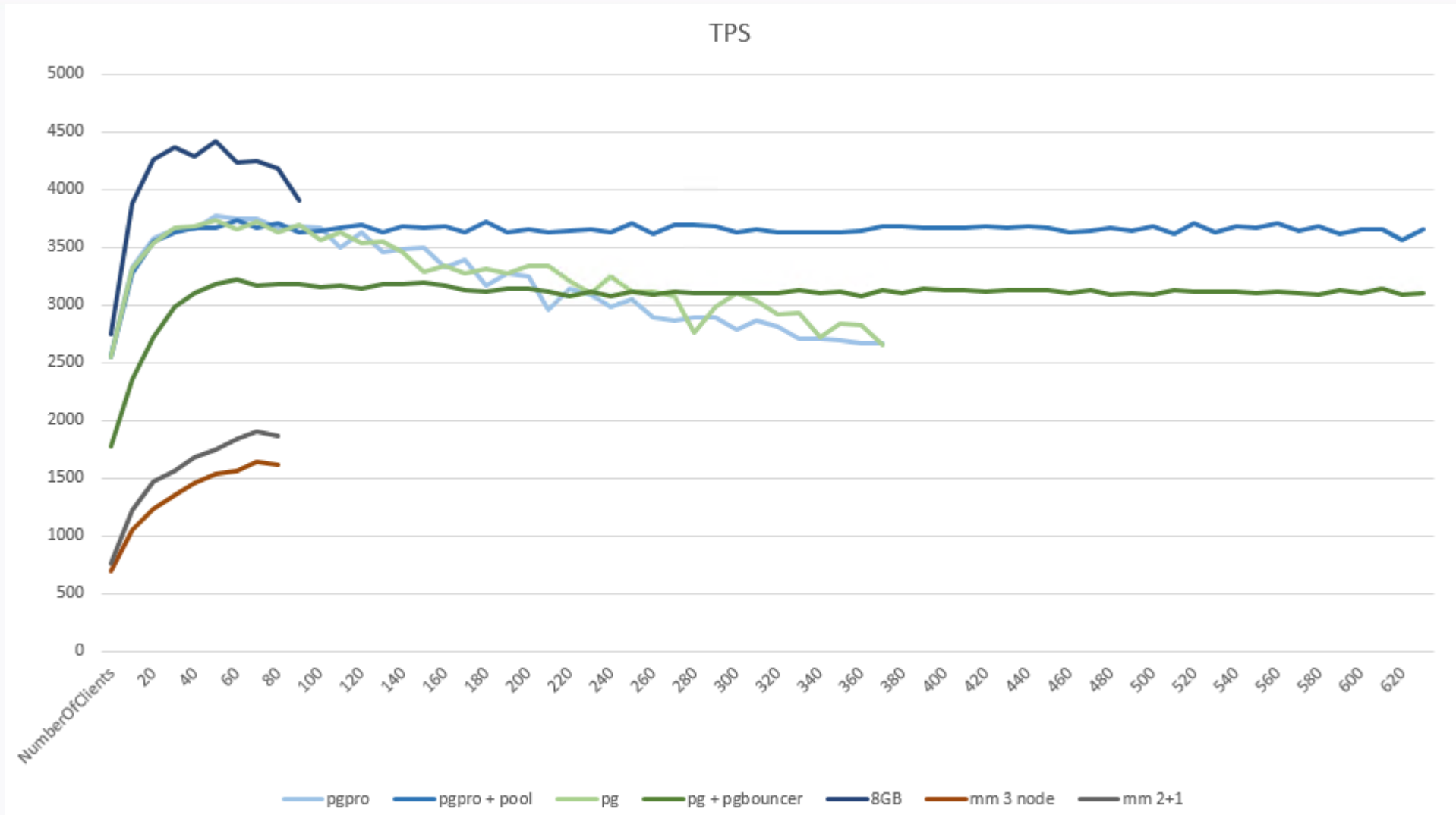


# PostgreSQL Pro Multimaster

5/6

- Что это и какие ограничения
- Настройка
- Результаты

2 x Xeon Gold 6248R  
8 GB RAM



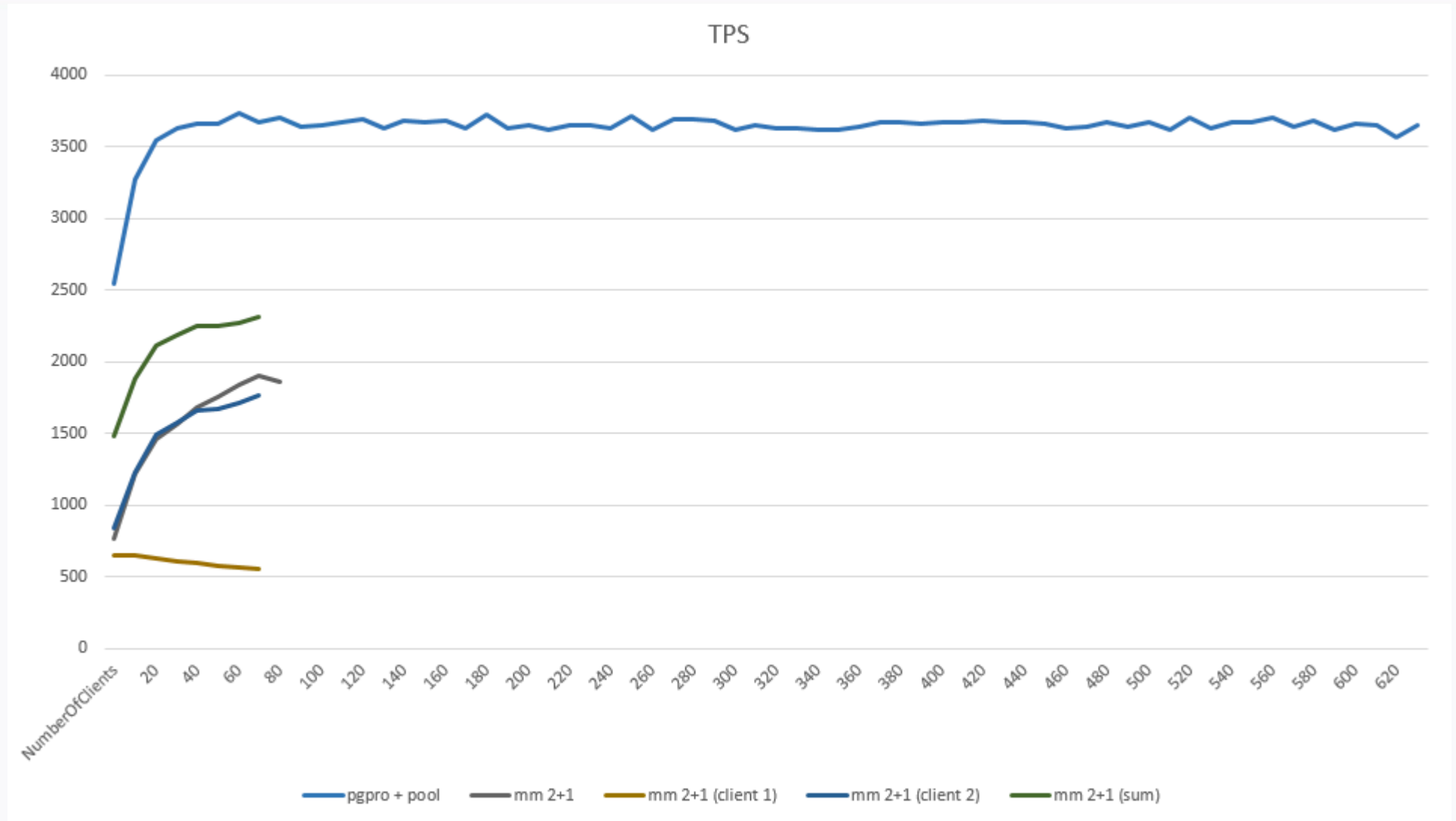


# PostgreSQL Pro Multimaster

5/6

- Что это и какие ограничения
- Настройка
- Результаты

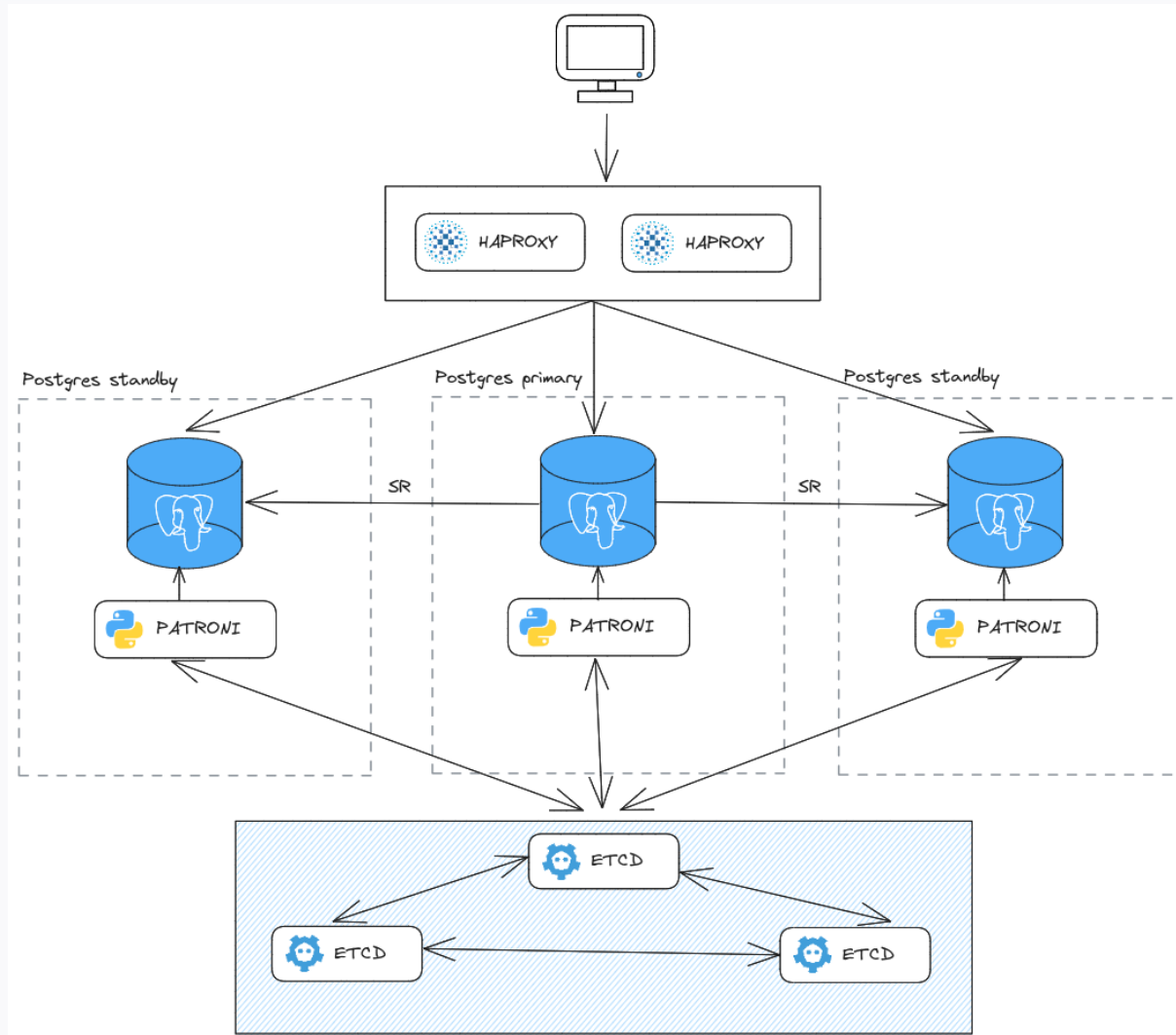
2 x Xeon Gold 6248R  
8 GB RAM



# PostgreSQL Pro + patroni + etcd + HAProxy

5/6

- Схема
- Настройка
- Результат



# PostgreSQL Pro + patroni + etcd + HAProxy

5/6

- Схема

cluster is healthy

9f79bd8cbb090fed: name=etcd-03 peerURLs=http://192.168.0.13:2380

clientURLs=http://192.168.0.13:2379 isLeader=false

ab28f612fac9cd5e: name=etcd-02 peerURLs=http://192.168.0.12:2380

clientURLs=http://192.168.0.12:2379 isLeader=false

ccf520eb1d8aac51: name=etcd-01 peerURLs=http://192.168.0.11:2380

clientURLs=http://192.168.0.11:2379 isLeader=true

- Результат

# patroni

```
+ Cluster: pg_patroni (7325984515973847232) ----+-----+-----+
| Member   | Host           | Role    | State    | TL | Lag in MB |
+-----+-----+-----+-----+-----+-----+
| pgsql-01 | 192.168.0.21   | Leader  | running  | 1  |           |
| pgsql-02 | 192.168.0.22   | Replica | streaming| 1  | 0         |
| pgsql-03 | 192.168.0.23   | Replica | streaming| 1  | 0         |
+-----+-----+-----+-----+-----+-----+

```

Что сделано:

- Настроена Active Directory аутентификация (SSO)
- Настроена возможность кросс-базных (кросс-инстансных) запросов в обе стороны
- Настроена возможность синхронизации пользователей и групп (с членством пользователей) из AD
- Настроен пулинг
- Протестирована конфигурация PostgreSQL Pro multimaster
- Настроена отказоустойчивая конфигурация (etcd + patroni)



The background of the slide is an aerial photograph of a dense city skyline, likely New York City, with numerous skyscrapers. The image is overlaid with a semi-transparent blue layer. A network of thin, light blue lines connects various points across the blue area, creating a digital or technological aesthetic. The text "Спасибо за внимание!" is centered in white, bold, sans-serif font.

Спасибо за внимание!