Pgpool-II Guide

Version 4.2.3

Built at 2021-07-23T10:38:41

1	Pgpool-II Guide	3
2	Installing Pgpool-II	3
3	Installing and Managing Extensions	13
4	Configuring Pgpool-II	21
5	Connecting a Client to Pgpool-II	29
6	Upgrading Pgpool-II and Extensions	30
7	Uninstalling Pgpool-II and Extensions	31

1 Pgpool-II Guide

Pgpool-II acts as middleware between client applications and a PostgreSQL database server.

Using Pgpool-II adds the following benefits to your application connection infrastructure:

- Transparently reroutes and ensures load balancing of Read Only transactions to Standby database servers
- Reuses connections to prevent reconnects
- Brings down Postgres connections by queuing stale connections
- Integrates with Failover managers to follow the primary

EDB supports the following Pgpool-II functionality:

- Load balancing
- Connection pooling
- Replication
- High availability
- Connection limits
- Watchdog
- Limiting Exceeding Connections
- In Memory Query Cache

Conventions Used in this Guide

The term Postgres refers to either PostgreSQL or EDB Postgres Advanced Server.

What's New

This release contains a merge with upstream, which includes the following bug fixes:

- Fix race condition between detach_false_primary and follow_primary_command.
- Fix broken database/app redirect preference in statement level load balancing mode.
- Fix pgpool crash when query cache enabled for non-streaming and logical replication mode.
- Fix query cache not being created in other than streaming and logical replication mode.
- Fix scenario where no primary node is found when detach_false_primary and follow_primary_command operation are running concurrently
- Fix hang when using asyncpg (Python frontend driver with asynchronous I/O)
- Enhance debug message upon receiving startup packet. Now it will print all the GUC variables in the log instead of just username, database name and application name. It will help clients to see why cached connections are not used.

For more information, please refer to the Upstream release notes.

!!! Note 4.2 is a major release. For more details on migrating from earlier versions to version 4.2, see the Migration Section.

2 Installing Pgpool-II

!!! Note Pgpool-II runs as a service on Linux systems. Pgpool-II is not supported on Windows.

The following table lists the Pgpool version and their corresponding EDB Postgres Advanced Server and PostgreSQL versions.

The Pgpool version required for your EDB Postgres Advanced Server and PostgreSQL installation is version-specific, but the documented and supported functionality of each version is the same.

Pgpool Version	Postgres Version	Supported Platforms
Pgpool 4.2	EDB Postgres Advanced Server and PostgreSQL 13	RHEL 7 - x86_64, RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch and 10x Buster Ubuntu 18.04 LTS Bionic Beaver and Ubuntu 20.04 LTS Focal Fossa
Pgpool 4.2	EDB Postgres Advanced Server and PostgreSQL 12	RHEL 7 - x86_64, and RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch and 10x Buster Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 4.2	EDB Postgres Advanced Server and PostgreSQL 11	RHEL 7 - x86_64, RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 4.2	EDB Postgres Advanced Server and PostgreSQL 9.6 and 10	RHEL 7 - x86_64 and RHEL 7 - ppc64le
Pgpool 4.1	EDB Postgres Advanced Server 13	RHEL 7 - x86_64, RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch and 10x Buster Ubuntu 18.04 LTS Bionic Beaver and Ubuntu 20.04 LTS Focal Fossa
Pgpool 4.1	EDB Postgres Advanced Server 12	RHEL 7 - x86_64, and RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch and 10x Buster Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 4.1	EDB Postgres Advanced Server 11	RHEL 7 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 4.0	EDB Postgres Advanced Server 12	RHEL 7 - x86_64, and RHEL 8 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch and 10x Buster Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 4.0	EDB Postgres Advanced Server 11	RHEL 7 - x86_64 RHEL 7 - ppc64le Debian 9x Stretch Ubuntu 18.04 LTS Bionic Beaver SLES 12
Pgpool 3.7	EDB Postgres Advanced Server 10 and 11	RHEL 7 - x86_64 and RHEL 7 - ppc64le

Pgpool Version	Postgres Version	Supported Platforms
Pgpool 3.6	EDB Postgres Advanced Server 9.6 and 10	RHEL 7 - x86_64 RHEL 7 - ppc64le Linux graphical installer

!!! Note - Pgpool is no longer supported on CentOS/RHEL/OL 6.x platforms. It is strongly recommended that EDB products running on these platforms be migrated to a supported platform. - Pgpool is certified with the Pgpool-II extensions shipped with EDB Postgres Advanced Server and PostgreSQL.

This guide assumes that the user has some knowledge of installation and system administration procedures and has administrative privileges on the host.

Before installing the repository configuration, you must have credentials that allow access to the EDB repository. For information about requesting credentials, visit the EDB website.

Installing Pgpool-II on a CentOS Host

Perform the following steps to install Pgpool-II on a CentOS host:

1. To install the repository configuration, assume superuser privileges, and invoke the platform-specific command:

   On CentOS 7:

   ```
   yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

   On CentOS 8:

   ```
   dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

2. Replace the `USERNAME:PASSWORD` variable in the following command with the username and password of a registered EDB user:

   ```
   sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
   ```

3. Invoke the following command to install the Extra Packages for Enterprise Linux (EPEL):

   On CentOS 7:

   ```
   yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
   7.noarch.rpm
   ```

   On CentOS 8:

   ```
   dnf -y install epel-release
   ```

4. The following steps are applicable only for CentOS 8:

   a. Enable the PowerTools repository to satisfy additional package dependencies:

   ```
   dnf config-manager --set-enabled PowerTools
   ```

   b. Disable the built-in PostgreSQL module:

```
dnf -qy module disable postgresql
```

5. Invoke the platform-specific command to install Pgpool-II:

   On CentOS 7:

   ```
   yum install edb-pgpool<xx>
   ```

   On CentOS 8:

   ```
   dnf install edb-pgpool<xx>
   ```

   Where <xx> is the Pgpool release version.

   For example, to install the latest Pgpool Version 4.2, invoke the following command:

   On CentOS 7:

   ```
   yum install edb-pgpool42
   ```

   On CentOS 8:

   ```
   dnf install edb-pgpool42
   ```

   When you install an RPM package signed by a source that is not recognized by your system, your permission to import the key to your local server may be asked. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y` and press `Return` to continue.

   Pgpool-II is installed in the `/usr/edb/pgpool<x.y>/` directory, where <x.y> is the installed Pgpool-II version number.

Installing Pgpool-II on an RHEL Host

Before creating the repository configuration file, you must have credentials that allow access to the EDB repository. For information about requesting credentials, visit the EDB website.

Perform the following steps to install Pgpool-II:

1. To create the repository configuration file, assume superuser privileges, and invoke the platform-specific command:

   On RHEL 7:

   ```
   yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

   On RHEL 8:

   ```
   dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

2. Replace the `USERNAME:PASSWORD` variable in the following command with the username and password of a registered EDB user:

   ```
   sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
   ```

3. Invoke the following command to install the Extra Packages for Enterprise Linux (EPEL):

On RHEL 7:

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

On RHEL 8:

```
dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

4. Enable the repository:

   On RHEL 7, enable the `optional, extras`, and `HA` repositories to satisfy additional package dependencies:

   ```
   subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
   ```

   On RHEL 8, enable the `codeready-builder-for-rhel-8-*-rpms` repository to satisfy additional package dependencies:

   ```
   ARCH=$( /bin/arch )

   subscription-manager repos --enable "codeready-builder-for-rhel-8-${ARCH}-rpms"
   ```

5. For RHEL 8, disable the built-in PostgreSQL module:

   ```
   dnf -qy module disable postgresql
   ```

6. Invoke the platform-specific command to install Pgpool-II:

   On RHEL 7:

   ```
   yum install edb-pgpool<xx>
   ```

   On RHEL 8:

   ```
   dnf install edb-pgpool<xx>
   ```

   Where <xx> is the Pgpool release version.

   For example, to install the latest Pgpool Version 4.2, invoke the following command:

   On RHEL 7:

   ```
   yum install edb-pgpool42
   ```

   On RHEL 8:

   ```
   dnf install edb-pgpool42
   ```

   When you install an RPM package that is signed by a source that is not recognized by your system, your permission to import the key to your local server may be asked for. If prompted, and you are satisfied that the packages come from a trustworthy source, enter `y`, and press `Return` to continue.

   Pgpool-II will be installed in the `/usr/edb/pgpool<x.y>/` directory, where <x.y> is the installed Pgpool-II version number.

Installing Pgpool-II on an RHEL/CentOS 7 PPCLE Host

Before creating the repository configuration file, you must have credentials that allow access to the EDB repository. For information about requesting credentials, visit the EDB website.

Perform the following steps to install Pgpool-II on an RHEL/CentOS 7 PPC64LE Host:

1. Install Advance Toolchain:

```
rpm --import
https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/RHEL7/gp
-pubkey-6976a827-5164221b

cat > /etc/yum.repos.d/advance-toolchain.repo <<EOF

# Beginning of the configuration file
[advance-toolchain]
name=Advance Toolchain IBM FTP
baseurl=https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/
7
failovermethod=priority
enabled=1
gpgcheck=1
gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/RHE
-pubkey-6976a827-5164221b
# End of the configuration file
```

2. To create the repository configuration file, assume superuser privileges and invoke the following command:

```
yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
```

3. Replace the `USERNAME:PASSWORD` variable in the following command with the username and password of a registered EDB user:

```
sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
```

4. Invoke the following command to install the Extra Packages for Enterprise Linux (EPEL):

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

5. On RHEL 7, enable the `optional, extras`, and `HA` repositories to satisfy additional package dependencies:

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-
extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
```

6. Invoke the following command to install Pgpool-II:

```
yum -y install edb-pgpool<xx>
```

   Where <xx> is the Pgpool-II version you want to install.

Installing Pgpool-II on a Debian/Ubuntu Host

To install Pgpool-II on a Debian or Ubuntu host, you must have credentials that allow access to the EDB repository. To request credentials for the repository, visit the EDB website.

Perform the following steps to install a Debian package using the EDB apt repository.

1. Assume superuser privileges:

```
sudo su -
```

2. Configure access to the EDB repository on your system:

   On Debian 9, Ubuntu 18, and Ubuntu 20, replace the `username` and `password` with your EDB credentials:

```
sh -c 'echo "deb https://<username>:<password>@apt.enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
```

   On Debian 10:

   a. Set up the EDB repository:

```
sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -cs).list'
```

   b. Replace the `username` and `password` with your EDB credentials:

```
sh -c 'echo "machine apt.enterprisedb.com login <username> password <password>" > /etc/apt/auth.conf.d/edb.conf'
```

3. Add support to your system for secure APT repositories:

```
apt-get install apt-transport-https
```

4. Add the EDB signing key:

```
wget -q -O - https://apt.enterprisedb.com/edb-deb.gpg.key  | sudo apt-key add -
```

5. Update the repository metadata:

```
apt-get update
```

6. Install the Debian package:

```
apt-get install -y edb-pgpool<xx>
```

Where <xx> is the Pgpool release version.

After installing Pgpool-II on a Debian host, the configuration files are located in the `/etc/edb/edb-pgpool<x.y>` directory, where <x.y> is the Pgpool release version.

Installing Pgpool-II on a SLES 12 Host

You can use the Zypper package manager to install Pgpool-II on a SLES 12 host. Zypper will attempt to satisfy package

dependencies but requires access to specific repositories that EDB does not host.

1. Assume superuser privileges.

```
sudo su -
```

2. Invoke the following command to add the EDB repository to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo
```

3. Invoke the following command to refresh the metadata:

```
zypper refresh
```

4. Install `SUSEConnect` to register the host with SUSE to allow access to SUSE repositories:

```
zypper install SUSEConnect
```

5. Register the host with SUSE to allow access to SUSE repositories and replace `'REGISTRATION_CODE'` and `'EMAIL'` with your SUSE registration information:

```
SUSEConnect -r 'REGISTRATION_CODE' -e 'EMAIL'
SUSEConnect -p PackageHub/12.4/x86_64
SUSEConnect -p sle-sdk/12.4/x86_64
```

6. Install the following repository for PEM dependencies:

```
zypper addrepo
https://download.opensuse.org/repositories/Apache:/Modules/SLE_12_SP4/Apache:Module
o
```

7. Refresh the metadata:

```
zypper refresh
```

8. Install OpenJDK (version 1.8) for Java-based components:

```
zypper -n install java-1_8_0-openjdk
```

9. Use the Zypper utility to install Pgpool-II:

```
zypper install -n edb-pgpool<xx>
```

   Where <xx> is the Pgpool version you wish to install.

Installing Pgpool-II Using the Linux Graphical Installer

Graphical installers for Pgpool-II are available via StackBuilder Plus (on EDB Postgres Advanced Server hosts) or Stack Builder (on PostgreSQL hosts).

!!! Note Pgpool does not support Windows systems.

Perform the following steps to install Pgpool-II by accessing StackBuilder Plus through your Linux start menu:

1. Open StackBuilder Plus and select your EDB Postgres Advanced Server installation from the drop-down list. Click `Next` to continue to the application selection page.
2. Expand the `Add-ons, tools and utilities` node, and check the box next to the Pgpool-II version you want to install and download the Pgpool-II installer.
3. Click `Next` to continue. Provide the credentials and click `Next`.
4. The selected packages and the default download directory are displayed. Click `Next`.
5. Once you have downloaded the installation files, a confirmation message is displayed. Click `Next` to start the Pgpool-II installation.
6. Select an installation language and click `OK`.
7. The Pgpool installer welcomes you to the setup wizard.

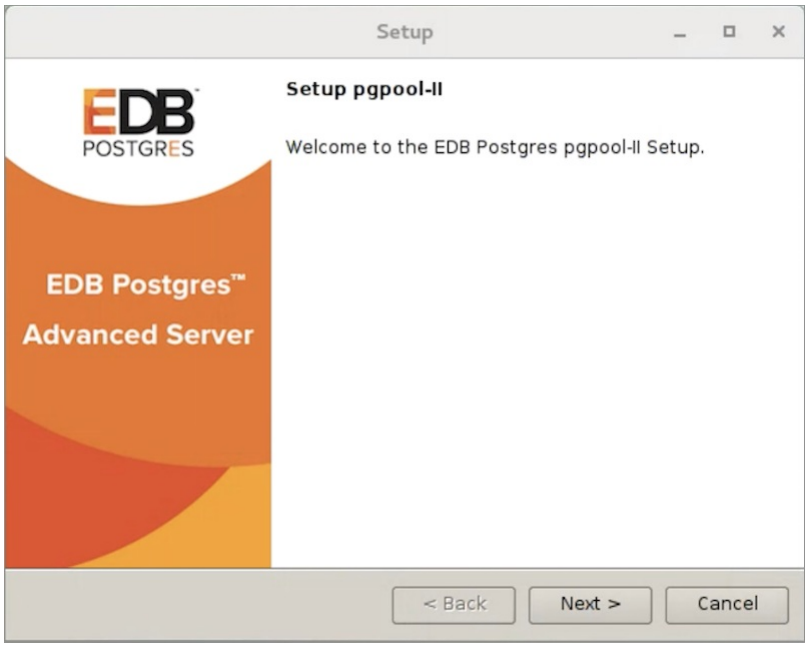

Fig. 1: The Pgpool-II Welcome window

8. Use the `Installation Directory` field to specify the directory where you would install the Pgpool-II software (the default installation directory is `/opt/edb`). Then, click `Next` to continue.

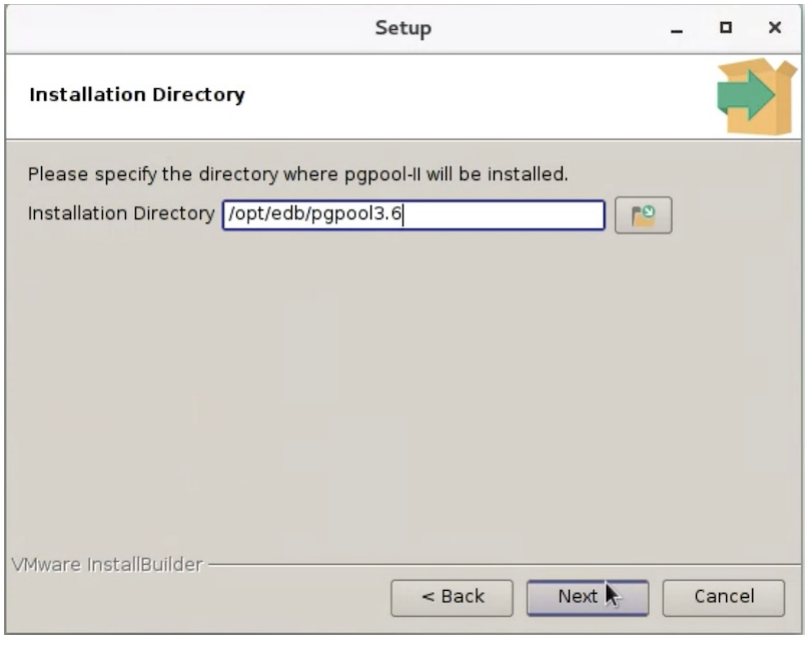

Fig. 2: The Pgpool-II Installation Details Window

9. Use the `Operating System User` field to specify the Linux operating system user's name that Pgpool-II will change to after startup. Then, click `Next` to continue.

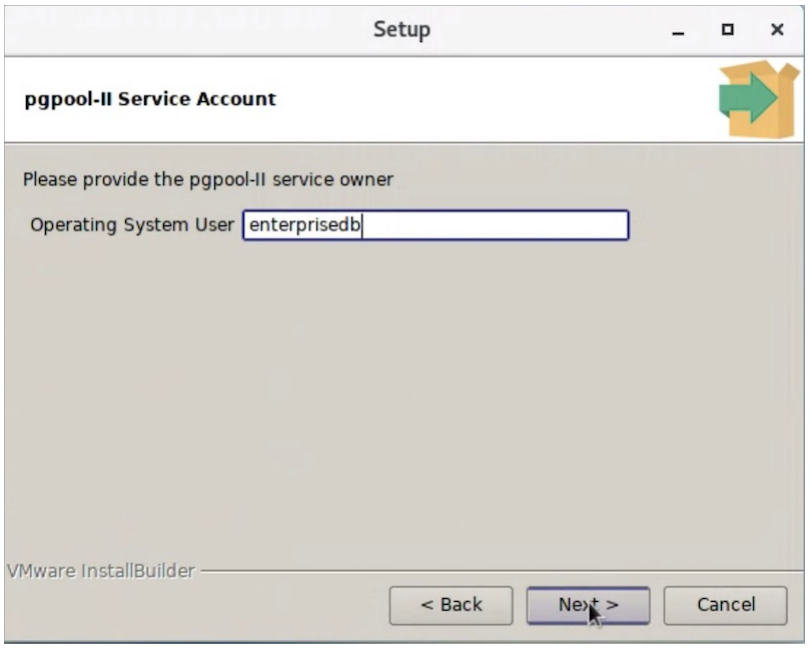

Fig. 3: The Pgpool-II Operating User window

10. The `Ready to Install` window notifies you when the installer has all of the information needed to install Pgpool-II on your system. Click `Next` to install Pgpool-II.

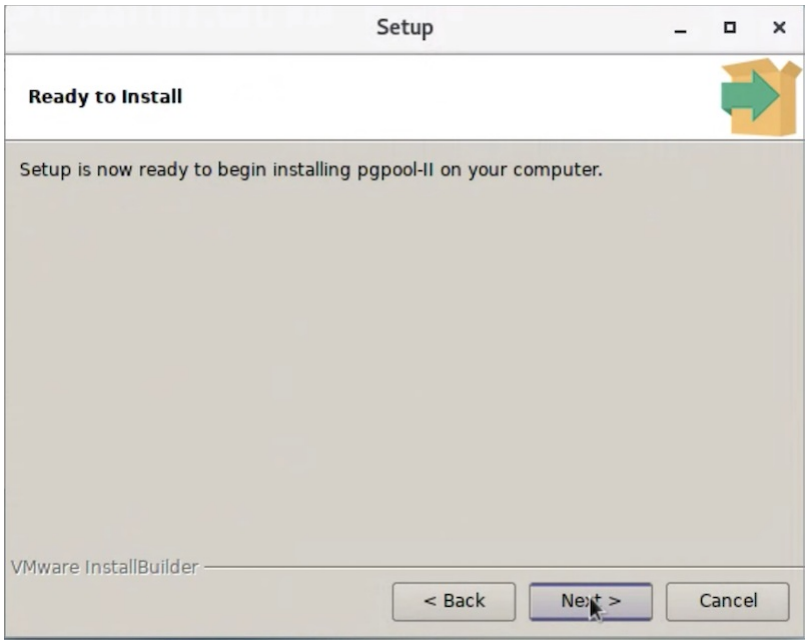

Fig. 4: The Ready to Install window

11. Progress bars inform you as the installation progresses.

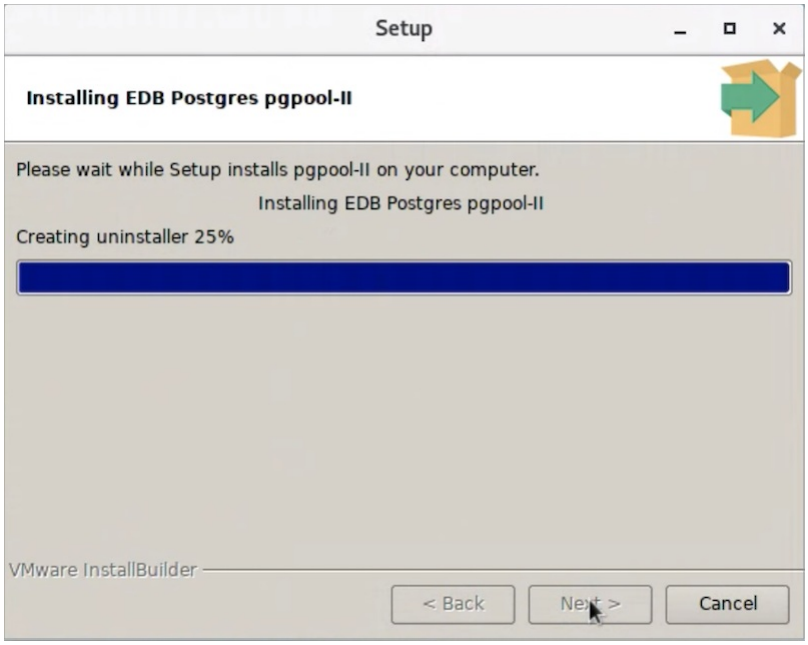

Fig. 5: The installation progresses

12. The installer notifies you when the setup wizard has completed the Pgpool-II installation. Click `Finish` to exit the installer.

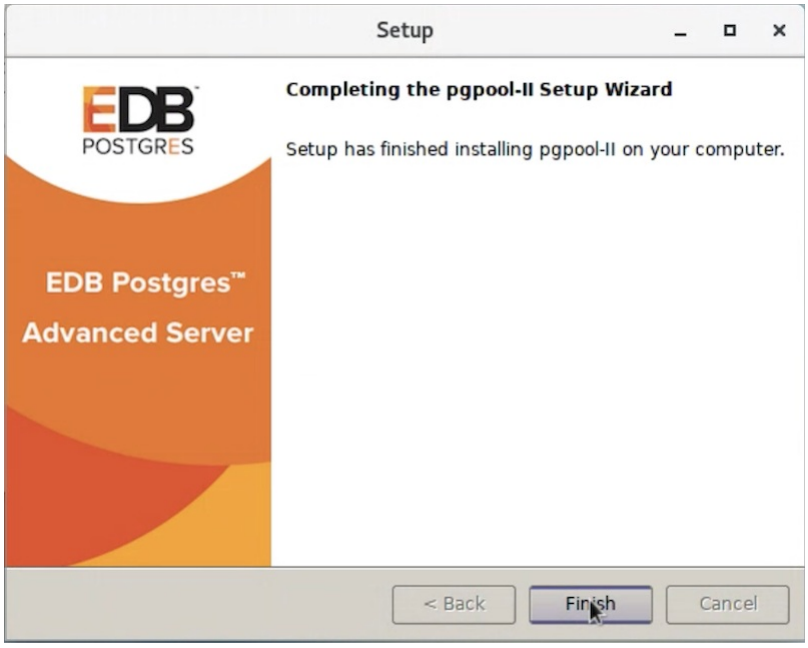

Fig. 6: The installation is complete

3 Installing and Managing Extensions

Modules in the extensions directory are additional features to EDB Postgres Advanced Server, which are generally not included in the core database. Once loaded in a database, they can function just like built-in features. They allow you to use simple `SELECT` commands to use PCP remotely.

!!! Note Pgpool-II extensions are only delivered for supported combinations of EDB Postgres Advanced Server versions and operating systems.

Before installing Pgpool-II extensions, install the EDB Postgres Advanced Server on your host system.

Installing Pgpool-II Extensions

The following section walks you through the steps of installing Pgpool-II extensions. To request credentials for the repository, visit the EDB website.

Installing Pgpool-II Extension on a CentOS Host

Assume superuser privileges and perform the following steps to install Pgpool-II extensions on a CentOS host:

1. To install the repository configuration, assume superuser privileges, and invoke the platform-specific command:

   On CentOS 7:

   ```
   yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

   On CentOS 8:

   ```
   dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

2. Replace the `USERNAME:PASSWORD` variable with the username and password of a registered EDB user:

   ```
   sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
   ```

3. Before installing Pgpool, execute the following command to install the Extra Packages for Enterprise Linux (EPEL):

   On CentOS 7:

   ```
   yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
   ```

   On CentOS 8:

   ```
   dnf -y install epel-release
   ```

4. For CentOS 8, enable the PowerTools repository to satisfy EPEL package dependencies:

   ```
   dnf config-manager --set-enabled PowerTools
   ```

5. Use the platform-specific command to install Pgpool-II extensions:

   On RHEL/CentOS 7:

   ```
   yum -y install edb-as<xx>-pgpool<yy>-extensions
   ```

   For example, to install Pgpool42 extensions for EDB Postgres Advanced Server 13, execute the following command:

   ```
   yum -y install edb-as13-pgpool42-extensions
   ```

On RHEL/CentOS 8:

```
dnf install edb-as<xx>-pgpool<yy>-extensions
```

In the above command, `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version. The extensions will be available in the `/usr/edb/as<xx>/share/extension` directory.

Installing Pgpool-II Extensions on an RHEL Host

Before installing the repository configuration, you must have credentials that allow access to the EDB repository. For information about requesting credentials, visit the EDB website.

Perform the following steps to install Pgpool-II extensions on an RHEL host:

1. To install the repository configuration, assume superuser privileges, and invoke the platform-specific command:

   On RHEL 7:

   ```
   yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

   On RHEL 8:

   ```
   dnf -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

2. Replace the `USERNAME:PASSWORD` variable in the following command with the username and password of a registered EDB user:

   ```
   sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
   ```

3. Before installing Pgpool, execute the following command to install the Extra Packages for Enterprise Linux (EPEL):

   On RHEL 7:

   ```
   yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
   7.noarch.rpm
   ```

   On RHEL 8:

   ```
   dnf -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
   8.noarch.rpm
   ```

4. Enable the repository:

   On RHEL 7, enable the `optional, extras`, and `HA` repositories to satisfy EPEL package dependencies:

   ```
   subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-
   extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
   ```

   On RHEL 8, enable the `codeready-builder-for-rhel-8-*-rpms` repository to satisfy EPEL packages dependency:

   ```
   ARCH=$( /bin/arch )

   subscription-manager repos --enable "codeready-builder-for-rhel-8-${ARCH}-rpms"
   ```

5. Use the platform-specific command to install Pgpool-II extensions:

   On RHEL/CentOS 7:

   ```
   yum -y install edb-as<xx>-pgpool<yy>-extensions
   ```

   For example, to install Pgpool42 extensions for EDB Postgres Advanced Server 13, execute the following command:

   ```
   yum -y install edb-as13-pgpool42-extensions
   ```

   On RHEL/CentOS 8:

   ```
   dnf install edb-as<xx>-pgpool<yy>-extensions
   ```

In the above command, `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version. The extensions will be available in the `/usr/edb/as<xx>/share/extension` directory.

Installing Pgpool-II Extensions on an RHEL/CentOS 7 PPCLE Host

Before installing the repository configuration, you must have credentials that allow access to the EDB repository. For information about requesting credentials, visit the EDB website.

Perform the following steps to install Pgpool-II extensions on an RHEL/CentOS 7 PPC64LE Host:

1. Install Advance Toolchain:

   ```
   rpm --import
   https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/RHEL7/gp
   -pubkey-6976a827-5164221b

   cat > /etc/yum.repos.d/advance-toolchain.repo <<EOF

   # Beginning of the configuration file
   [advance-toolchain]
   name=Advance Toolchain IBM FTP
   baseurl=https://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/
   7
   failovermethod=priority
   enabled=1
   gpgcheck=1
   gpgkey=ftp://public.dhe.ibm.com/software/server/POWER/Linux/toolchain/at/redhat/RHI
   -pubkey-6976a827-5164221b
   # End of the configuration file
   ```

2. To install the repository configuration, assume superuser privileges, and invoke the following command:

   ```
   yum -y install https://yum.enterprisedb.com/edbrepos/edb-repo-latest.noarch.rpm
   ```

3. Replace the `USERNAME:PASSWORD` variable in the following command with the username and password of a registered EDB user:

   ```
   sed -i "s@<username>:<password>@USERNAME:PASSWORD@" /etc/yum.repos.d/edb.repo
   ```

4. Before installing Pgpool-II, execute the following command to install the Extra Packages for Enterprise Linux (EPEL):

```
yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-
7.noarch.rpm
```

5. On RHEL 7, enable the `optional, extras`, and `HA` repositories to satisfy EPEL package dependencies:

```
subscription-manager repos --enable "rhel-*-optional-rpms" --enable "rhel-*-
extras-rpms"  --enable "rhel-ha-for-rhel-*-server-rpms"
```

6. Invoke the following command to install Pgpool-II extensions:

```
yum -y install edb-as13-pgpool<xx>-extensions
```

   Where <xx> is the Pgpool-II version you want to install.

Installing Pgpool-II Extensions on a Debian/Ubuntu Host

To install Pgpool-II extensions on a Debian or Ubuntu host, you must have credentials that allow access to the EDB repository. To request credentials for the repository, visit the EDB website.

The following steps walk you through using the EDB apt repository to install a DEB package.

1. Assume superuser privileges:

```
sudo su -
```

2. Configure the EDB repository. Substitute your EDB credentials for the `username` and `password` placeholders in the following command:

   On Debian 9 and Ubuntu 18:

```
sh -c 'echo "deb https://username:password@apt.enterprisedb.com/$(lsb_release -
cs)-edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -
cs).list'
```

   On Debian 10:

   a. Set up the EDB repository:

```
sh -c 'echo "deb [arch=amd64] https://apt.enterprisedb.com/$(lsb_release -cs)-
edb/ $(lsb_release -cs) main" > /etc/apt/sources.list.d/edb-$(lsb_release -
cs).list'
```

   b. Substitute your EDB credentials for the `username` and `password` placeholders in the following command:

```
sh -c 'echo "machine apt.enterprisedb.com login <username> password <password>" >
/etc/apt/auth.conf.d/edb.conf'
```

3. Add support to your system for secure APT repositories:

```
apt-get install apt-transport-https
```

4. Add the EDB signing key:

```
wget -q -O - https://apt.enterprisedb.com/edb-deb.gpg.key | apt-key add -
```

5. Update the repository metadata:

```
apt-get update
```

6. Install the Debian package:

```
apt-get install -y edb-as<xx>-pgpool<yy>-extensions
```

In the above command, `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version you want to install.

Installing Pgpool-II Extension on a SLES 12 Host

You can use the Zypper package manager to install the Pgpool-II extension on a SLES 12 host. Zypper will attempt to satisfy package dependencies as it installs a package but requires access to specific repositories that are not hosted at EDB.

1. Assume superuser privileges.

```
sudo su -
```

2. Use the following command to add the EDB repository to your SLES host:

```
zypper addrepo https://zypp.enterprisedb.com/suse/edb-sles.repo
```

3. Invoke the following command to refresh the metadata:

```
zypper refresh
```

4. Install `SUSEConnect` to register the host with SUSE to allow access to SUSE repositories:

```
zypper install SUSEConnect
```

5. Register the host with SUSE to allow access to SUSE repositories and replace `'REGISTRATION_CODE'` and `'EMAIL'` with your SUSE registration information:

```
SUSEConnect -r 'REGISTRATION_CODE' -e 'EMAIL'
SUSEConnect -p PackageHub/12.4/x86_64
SUSEConnect -p sle-sdk/12.4/x86_64
```

6. Install the following repository for PEM dependencies:

```
zypper addrepo
https://download.opensuse.org/repositories/Apache:/Modules/SLE_12_SP4/Apache:Module
o
```

7. Refresh the metadata:

```
zypper refresh
```

8. Install OpenJDK (version 1.8) for Java based components:

```
zypper -n install java-1_8_0-openjdk
```

9. Then, use the Zypper utility to install Pgpool-II extension:

```
zypper -n install edb-as<xx>-pgpool<yy>-extensions
```

Where <xx> is the EDB Postgres Advanced Server version and <yy> is the Pgpool version you wish to install.

Installing Pgpool-II Extension Using the Linux Graphical Installer

Graphical installers for Pgpool-II extensions are available via StackBuilder Plus (for EDB Postgres Advanced Server hosts) or Stack Builder (on PostgreSQL hosts). You can access StackBuilder Plus through your Linux start menu. It is not supported on Windows.

Perform the following steps to install Pgpool-II extensions:

1. Open StackBuilder Plus and select your EDB Postgres Advanced Server installation from the drop-down list on the `Welcome` window. Click `Next` to continue to the application selection page.
2. Expand the `Add-ons, tools and utilities` node, and check the box next to the Pgpool-II extension to select and download the installer. Click `Next` to continue.
3. Provide the credentials and click `Next`.
4. The selected packages and the default download directory where the package will be installed are displayed; change the download directory location if required. Click `Next`.
5. Once you have downloaded the installation files, a confirmation message is displayed. Click `Next` to start the installation.
6. Select an installation language and click `OK`.
7. The Pgpool-II extensions installer welcomes you to the setup wizard.

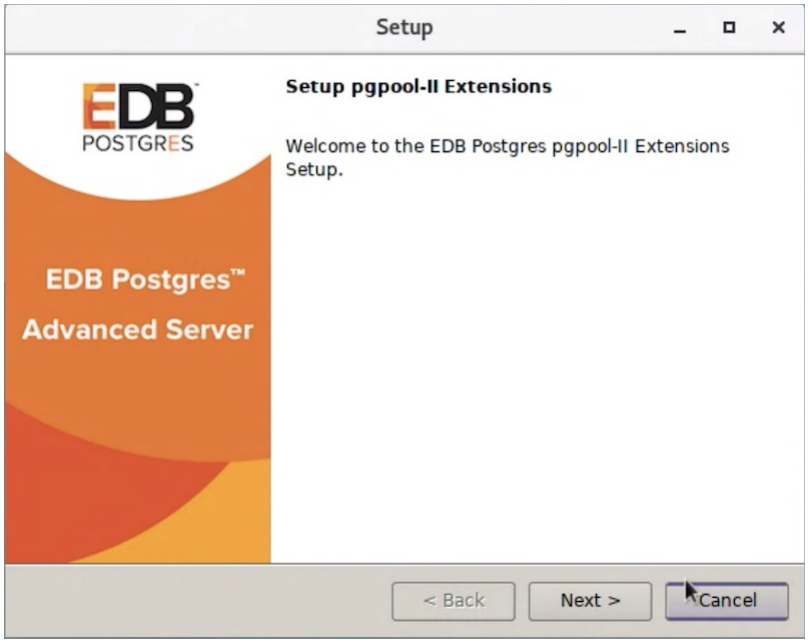

Fig. 1: The Pgpool-II Extensions Welcome window

8. Use the `Installation Directory` field to specify the directory in which you wish to install the Pgpool-II extensions software (the default installation directory is `/opt/edb/as<xx>`) Then, click `Next` to continue.

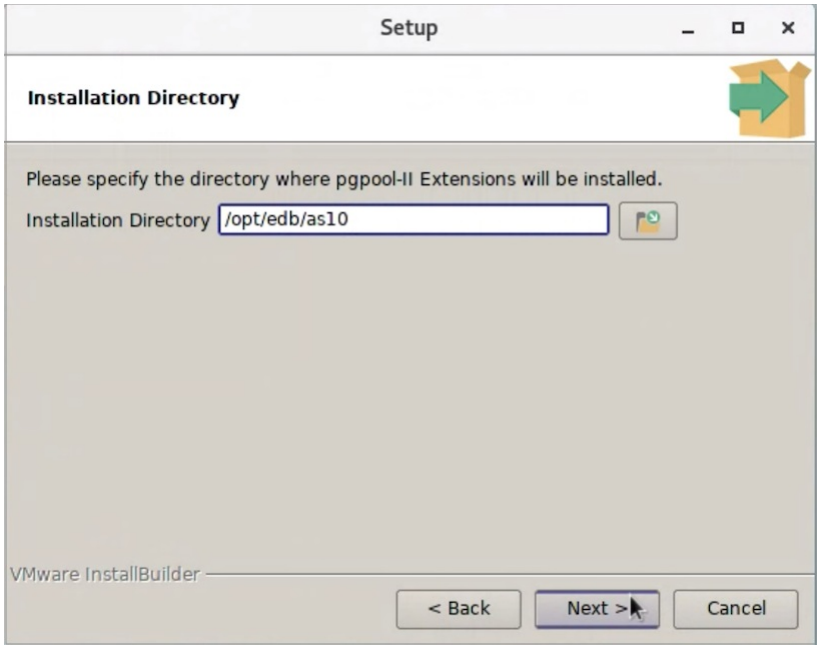

Fig. 2: The Pgpool-II Extensions Installation Details Window

9. The `Ready to Install` window notifies you when the installer has all of the information needed to install Pgpool-II extensions on your system. Click `Next` to install Pgpool-II extensions.

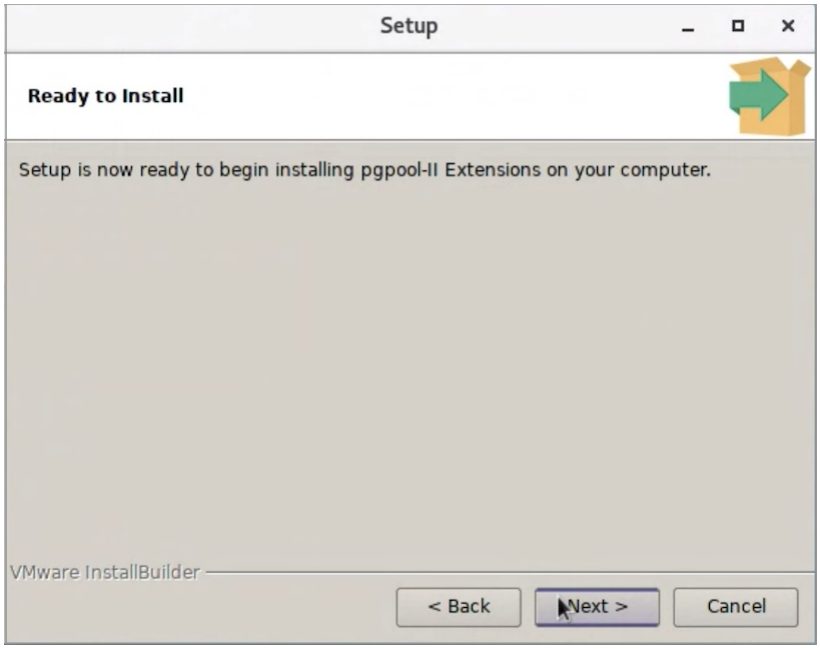

Fig. 3: The Ready to Install window

10. Progress bars inform you as the installation progresses.

```
![The installation progresses](images/pg4.png)

Fig. 4: The installation progresses
```

11. The installer notifies you when the setup wizard has completed the Pgpool-II installation. Click `Finish` to exit the installer.

```
![The installation is complete](images/pg5.png)
```

```
Fig. 5: The installation is complete
```

12. The extensions will be available in the `/opt/edb/as<xx>/share/extension/pgpool*` directory.

Creating Pgpool-II Extensions

You must install and create the extensions in each database where you will be using Pgpool-II functionality. To ensure all extensions are available for future databases, you can add the extension to the `template1` database; any extensions installed in the `template1` database will be created in each of the databases that uses `template1` as a template during creation.

Pgpool_adm Extension

`Pgpool_adm` is a set of extensions that allows SQL access to PCP commands. To view information about PCP commands, see https://www.pgpool.net/docs/latest/en/html/pcp-commands.html.

After installing the `Pgpool_adm` extension, use the psql client application to connect to the database, and execute the following SQL command:

```
CREATE EXTENSION pgpool_adm;
```

To view more information about `Pgpool_adm`, see https://www.pgpool.net/docs/latest/en/html/pgpool-adm.html.

Pgpool_recovery Extension

The `Pgpool_recovery` extension is required for online recovery and future fail-back mechanisms.

After installing the `Pgpool_recovery` extension, use psql to connect to the database, and execute the following SQL command to create a `Pgpool_recovery` extension:

```
CREATE EXTENSION pgpool_recovery;
```

For more information about using the `CREATE EXTENSION` command, see the PostgreSQL core documentation.

---

4 Configuring Pgpool-II

The configuration files are created in the `/etc/sysconfig/edb/pgpool<x.y>` directory, where *<x.y>* is the Pgpool release version. By default, `.sample` is appended to the configuration file name; remove the `.sample` from the configuration file after copying the file to create your custom configuration.

!!! Note The configuration options for Pgpool-II are extensive; consider the options listed below as a starting point only. For more information about configuring and using Pgpool-II, please consult the project website.

Commonly Used Parameters

The following table lists `pgpool.conf` parameters to use when implementing connection pooling:

Parameter Name	Description
listen_addresses	Hostname or IP address used by Pgpool-II to listen for connections. The default is localhost. Change to '*' for all addresses.
port	Port for Pgpool-II connections. The default is 9999.
pcp_port	Port for PCP connections. The default is 9898.
backend_hostname0	Hostname or IP address for backend 0. You can specify '' if the backend and Pgpool-II are running on the same host.
backend_port0	Port number for backend 0.
backend_weight0	Weight for backend 0 (only in load balancing mode). Specify 1 for each backend if you want to balance the load equally or decimal values (.9, .1, etc.) to weigh the load towards specific backends.
backend_data_directory0	Data directory for backend 0.
enable_pool_hba	Set to on to use pool_hba.conf for client authentication.
num_init_children	Number of pools. Default is 32.
max_pool	Number of connections per pool. Default is 4.
connection_cache	Set to on to enable connection pooling.
pool_conn_dbname	Database name to which Pgpool-II will connect. By default, Pgpool-II will connect with Postgres. Please note that the pool_conn_dbname parameter is now deprecated.

The following table lists pgpool.conf parameters to use when implementing replication and load balancing:

Parameter Name	Description
Allow_sql_comments	If on, ignore SQL comments; modifications to this parameter require a reload of the pgpool.conf file.
load_balance_mode	Set to on to activate load balancing mode. If load_balance_mode is on and replicate_select is off, SELECT statements are sent to one backend. The parameter backend_weight<N>.z determines the proportion of SELECT statements each backend receives.
ignore_leading_white_space	Ignore leading white spaces of each query. Certain APIs such as DBI/DBD::Pg for Perl add white space that the user cannot control. Default is on.

Configuring Connection Pooling

Pgpool-II provides a set of child processes that maintain cached connections to one or more database servers. When a client connects, Pgpool-II attempts to reuse a connection from its pool, thus avoiding the overhead of opening and closing client connections.

You can reuse a connection in the pool only if the target database and the connection user match a prior connection, which is currently in the pool. The pgpool.conf file specifies the connection pooling configuration options (such as the number of child processes and the maximum number of cached connections per child).

To configure connection pooling with one database server:

1. Configure the pg_hba.conf file on the Pgpool-II host to permit connections between the clients and the server.

2. Copy the pgpool.conf.sample file to pgpool.conf, modify the file, set the connection_cache parameter to on, and specify connection properties for your database server.

The following example shows how to connect with the EDB Postgres Advanced Server:

```
connection_cache = on

backend_hostname0 = 'localhost'

backend_port0 = 5444

backend_weight0 = 1

backend_data_directory0 = '/var/lib/edb/as13/data'
```

The following example shows how to connect with the PostgreSQL Server:

```
connection_cache = on

backend_hostname0 = 'localhost'

backend_port0 = 5432

backend_weight0 = 1

backend_data_directory0 = '/var/lib/pgsql/13/data'
```

!!! Note In the `pgpool.conf` file, connection parameters have an appended digit that specifies a cluster node identifier. Database node `0` specifies values for the primary node.

3. Optionally, configure Pgpool-II client authentication.

4. Optionally, configure the PCP administrative interface.

5. Start Pgpool-II:

   ```
   systemctl start edb-pgpool-<x.y>.service
   ```

   where <x.y> is the Pgpool release version.

6. Execute the following platform-specific command to connect to Pgpool42:

   On EDB Postgres Advanced Server for CentOS 7:

   ```
   ./psql -d edb -p 9999 -U enterprisedb -h /tmp
   ```

   On EDB Postgres Advanced Server for Debian:

   ```
   ./psql -d edb -p 9999 -U enterprisedb
   ```

   On PostgreSQL Server for CentOS 7:

   ```
   ./psql -d postgres -p 9999 -U postgres -h /tmp
   ```

   On PostgreSQL Server for Debian:

   ```
   ./psql -d postgres -p 9999 -U postgres
   ```

Configuring Load Balancing

EDB supports replication scenarios that use Pgpool-II load balancing with PostgreSQL streaming replication or Slony replication. The supported replication methods ensure that database updates made by client applications apply to multiple backend servers. For detailed information about the benefits of each replication method and configuration instructions, please review project documentation for each utility.

When load balancing is enabled, Pgpool-II distributes some types of `SELECT` statements to backend servers, allowing multiple database servers and hosts to share the processing load of `SELECT` statements issued by client applications.

When configuring Pgpool-II load balancing, the initial database environments in all backend servers must be identical:

- Tables must have the same name, definition, and row content.
- Schemas must exist in each backend application database.
- Roles and privileges on each backend server must be configured to ensure the result set of SQL statements are identical on all servers.

If you use `password` authentication, assign the same password to an associated user name on each database server. Use the same user name/password pair to connect Pgpool-II to each backend connection.

Within a replication scenario, each backend is uniquely identified by the hostname (or IP address) and the port number on which the database server instance is listening for connections. You must ensure that the `pool_hba.conf` and `pg_hba.conf` files allow a connection between that server and the host on which Pgpool-II will be running.

The following example demonstrates how to implement Pgpool-II load balancing with two servers (the primary and replica nodes) in a Streaming Replication scenario. Configuring Pgpool-II load balancing for a Slony replication scenario is similar; please see the Slony documentation for information about configuring Slony replication.

Configuring the Primary Node of the Replication Scenario

Open an SSH session with the primary node of the replication scenario, and modify the `pg_hba.conf` file (located in the `/var/lib/edb/as13/data` directory), adding connection information for the replication user (in the example that follows, `edbrepuser` resides on a standby node with an IP address of `107.178.217.178`):

```
   host replication edbrepuser 107.178.217.178/32 md5
```

The connection information should specify the address of the replication scenario's standby node and your preferred authentication method.

Modify the `postgresql.conf` file (located in `/var/lib/edb/as13/data`), adding the following replication parameter and values to the end of the file:

```
wal_level = replica
max_wal_senders = 10
checkpoint_segments = 8
wal_keep_segments = 0
```

Save the configuration file, and restart the server:

To restart on RHEL/CentOS 7 and 8 platforms:

```
systemctl restart edb-as-13
```

To restart on Debian 9.x | 10.x or Ubuntu 18.04 | 20.04 platforms:

```
/usr/edb/as13/bin/epas_ctlcluster 13 main restart
```

Use the `sudo su -` command to assume the identity of the `enterprisedb` database superuser:

```
sudo su - enterprisedb
```

Then, start a `psql` session, connecting to the `edb` database:

```
psql -d edb
```

At the `psql` command line, create a user with the `replication` attribute:

```
CREATE ROLE edbrepuser WITH REPLICATION LOGIN PASSWORD 'password';
```

Configuring the Standby Node of the Replication Scenario

Open an SSH session with the standby server, and assume the identity of the database superuser (`enterprisedb`):

```
sudo su - enterprisedb
```

With your choice of editor, create a `.pgpass` file in the home directory of the `enterprisedb` user. The `.pgpass` file holds the password of the replication user in plain-text form; if you are using a `.pgpass` file, you should ensure that only trusted users have access to the `.pgpass` file:

Add an entry that specifies connection information for the replication user:

```
*:5444:*:edbrepuser:password
```

The server will enforce restrictive permissions on the `.pgpass` file; use the following command to set the file permissions:

```
chmod 600 .pgpass
```

Relinquish the identity of the database superuser:

```
exit
```

Then, assume superuser privileges:

```
sudo su -
```

Use your platform-specific command to stop the database server before replacing the data directory on the standby node with the `data` directory of the primary node.

Then, delete the `data` directory on the standby node:

```
rm -rf /var/lib/edb/as13/data
```

After deleting the existing `data` directory, use the `pg_basebackup` utility to copy the `data` directory of the primary node to the standby:

```
pg_basebackup --format=p --label=standby --host=146.148.46.44 --username=edbrepuser
--password --wal-method=stream -R
```

The call to `pg_basebackup` specifies the IP address of the primary node and the name of the replication user created on the primary node.

Including the `-R` option creates the `standby.signal` file and appends connection settings to `postgresql.auto.conf` in the output directory (or into the base archive file when using tar format) to ease setting up a standby server.

For more information about the options available with the `pg_basebackup` utility, see the PostgreSQL core documentation.

When prompted by `pg_basebackup`, provide the password associated with the replication user.

After copying the `data` directory, change ownership of the directory to the database superuser (`enterprisedb`):

```
chown -R enterprisedb /var/lib/edb/as13/data
```

Modify the `postgresql.conf` file (located in `/var/lib/edb/as13/data`), specifying the following values at the end of the file:

```
wal_level = replica
hot_standby = on
```

The `data` file has been copied from the primary node, and contains the replication parameters specified previously.

Then, restart the server. At this point, the primary node will be replicating data to the standby node.

Configuring Pgpool-II Load Balancing

Use your choice of editor to modify the `pgpool.conf` file. Within the `pgpool.conf` file, modify the parameter settings to specify that load balancing is enabled:

```
load_balance_mode = on
```

Then, specify the connections settings for the primary database node in the parameter set that ends with a `0`. For example:

```
backend_hostname0 = '146.148.46.44'
backend_port0 = 5444
backend_weight0 = 1
backend_data_directory0 = '/var/lib/edb/as13/data'
```

Then, specify the connections settings for each node to which queries will be distributed. Increment the number that follows the parameter name for each node, and provide connection details:

```
backend_hostname1 = '107.178.217.178'
backend_port1 = 5444
backend_weight1 = 1
backend_data_directory1 = '/var/lib/edb/as13/data'
```

Use the `backend_weight` parameter to specify how queries will be distributed amongst the nodes. Specify a value of `1` to indicate that you wish (qualified) queries to be equally distributed across the nodes of the replication scenario.

Restart Pgpool-II

```
systemctl restart edb-pgpool-<x.y>.service
```

where <x.y> is the Pgpool release version.

Configuring Client Authentication

When Pgpool-II is enabled, client applications connect to Pgpool-II, which acts as a middleman for a Postgres server. A connecting client application is first authenticated with the Pgpool-II server and then with the Postgres server.

Parameter settings in the `pool_hba.conf` configuration file determine the Pgpool-II authentication properties. The `pool_hba.conf` file is similar in format and function to the Postgres `pg_hba.conf` configuration file. Please refer to the Pgpool-II documentation for detailed information about `pool_hba.conf` entries.

To enable Pgpool-II authentication:

1. Copy the `pool_hba.conf.sample` file to `pool_hba.conf`.
2. Modify the `pool_hba.conf` file, specifying authentication information for servers or users you want to connect. Entries must follow the same format used in the `pg_hba.conf` file.
3. Modify the `pgpool.conf` file, setting the `enable_pool_hba` parameter to `on`.
4. Restart Pgpool-II to reload the Pgpool-II configuration files.

!!! Note When authenticating with the database server, use the user names and passwords specified in the `pool_hba.conf` file; you must also specify these user names and passwords in the database server's `pg_hba.conf` file.

Configuring PCP

PCP is an administrative interface for Pgpool-II that allows you to retrieve information about database nodes, Pgpool-II child processes, and other information. You should issue PCP commands from the Linux command line.

`pcp.conf` is the password configuration file for the PCP client. Before using PCP commands, you must modify the `pcp.conf` file, providing the user names and passwords you provide when invoking a PCP command. The user names in the `pcp.conf` file are entirely independent of the database server user names and passwords.

Use the following steps to configure PCP:

1. Copy the `pcp.conf.sample` file to `pcp.conf`.

2. Add an entry to the `pcp.conf` file in the following form:

   ```
   username:md5_password
   ```

   where:

   `username` is a PCP user name.

   `md5_password` is the PCP password in `md5` format

   You can use the `pg_md5` program to generate the encrypted password from the clear-text form as shown below:

   ```
   $ pg_md5 mypassword

   34819d7beeabb9260a5c854bc85b3e44
   ```

   For example, the entry in the `pcp.conf` file for a PCP user named `pcpuser` with the password of `mypassword` is:

   ```
   # USERID:MD5PASSWD
   ```

```
pcpuser:34819d7beeabb9260a5c854bc85b3e44
```

3. Restart the Pgpool service.

4. When issuing a PCP command, specify the PCP user name and the unencrypted form of the password:

```
$ pcp_node_info 5 localhost 9898 pcpuser mypassword 0
localhost 5444 1 1.000000
```

After configuring PCP, you can use the following PCP commands to control Pgpool-II and retrieve information.

PCP Command	Description
`pcp_common_options`	Common options used in PCP commands
`pcp_node_count`	Displays the total number of database nodes
`pcp_node_info`	Displays the information on the given node ID
`pcp_health_check_stats`	Displays health check statistics data on given node ID
`pcp_watchdog_info`	Displays the watchdog status of the Pgpool-II
`pcp_proc_count`	Displays the list of Pgpool-II children process IDs
`pcp_proc_info`	Displays the information on the given Pgpool-II child process ID
`pcp_pool_status`	Displays the parameter values as defined in pgpool.conf
`pcp_detach_node`	Detaches the given node from Pgpool-II. Existing connections to Pgpool-II are forced to be disconnected.
`pcp_attach_node`	Attaches the given node to Pgpool-II.
`pcp_promote_node`	Promotes the given node as new main to Pgpool-II
`pcp_stop_pgpool`	Terminates the Pgpool-II process
`pcp_reload_config`	Reload pgpool-II config file
`pcp_recovery_node`	Attaches the given backend node with recovery

!!! Note `pcp_health_check_stats` and `pcp_reload_config` commands are available from Pgpool version 4.2 onwards.

To view more information about PCP command options, visit the Pgpool project site.

Configuring Number of Connections and Pooling

Pgpool has some configuration to tune the pooling and connection processing. Depending on this configuration, the Postgres configuration for `max_connections` must also be set to ensure all connections can be accepted as required. Furthermore, note that the Cloud Architecture works with active/active instances, which needs to spread `num_init_children` over all Pgpool instances (divide the normally used value by the number of active instances). The below text describes the effect of changing the configuration, and advises values for both the on-premise and the cloud architecture.

max_pool: Generally, advised to set `max_pool` to 1. Alternatively, for applications with many reconnects, `max_pool` can be set to the number of distinct combinations of users, databases, and connection options for the application connections. All but one connection in the pool would be stale connections, which consume a connection slot from Postgres without adding to the performance. It is, therefore, advised not to configure `max_pool` beyond 4 to preserve a healthy ratio between active and stale connections. As an example, for an application that constantly reconnects and uses two distinct users, both connecting to their own database, set it to 2. If both users would be able to connect to both databases, set it to 4. Note that increasing `max_pool` requires to tune down `num_init_children` in Pgpool, or

tune up `max_connections` in Postgres.

num_init_children: It is advised to set `num_init_children` to the number of connections that could be running active in parallel, but the value should be divided by the number of active Pgpool-II instances (one with the on-premise architecture and all instances for the cloud architecture). As an example: In an architecture with 3 Pgpool instances, to allow the application to have 100 active connections in parallel, set `num_init_children` to 100 for the on-premise architecture, and set `num_init_children` to 33 for the cloud architecture. Note that increasing `num_init_children` generally requires to tune up `max_connections` in Postgres.

listen_backlog_multiplier: Can be set to multiply the number of open connections (as perceived by the application) with the number of active connections (`num_init_children`). As an example, when the application might open 500 connections, of which 100 should be active in parallel, with the on-premise architecture, `num_init_children` should be set to 100, and `listen_backlog_multiplier` should be set to 4. This setup can process 100 connections active in parallel, and another 400 (`listen_backlog_multiplier` x `num_init_children`) connections will be queued before connections will be blocked. The application would perceive a total of 500 open connections, and Postgres would process the load of 100 connections maximum at all times. Note that increasing `listen_backlog_multiplier` only causes the application to perceive more connections but will not increase the number of parallel active connections (which is determined by `num_init_children`).

max_connections: It is advised to set `max_connections` in Postgres higher than [number of active pgpool instances] x [`max_pool`] x [`num_init_children`] + [`superuser_reserved_connections`] (Postgres). As an example: in the on-premise setup with 3 instances active/passive, `max_pool` set to 2, `num_init_children` set to 100, and `superuser_reserved_connections` (Postgres) set to 5, Postgres `max_connections` should be set equal to or higher than [1 x 2 x 100+5], which is 205 connections, or higher. A similar setup in the cloud setup would run with 3 active instances, `max_pool` set to 2, `num_init_children` set to 33, and `superuser_reserved_connections` (Postgres) set to 5, in which case Postgres `max_connections` should be set equal or higher than [3x 2 x 33+5] which is 203 or higher. Note that configuring below the advised setting can cause issues opening new connections, and in combination with `max_pool`, can cause unexpected behavior (low or no active connections but still connection issues due to stale pooled connections using connection slots from Postgres. For more information on the relation between `num_init_children`, `max_pool` and `max_connections`, see this background information.

Pgpool-II Host Setup

After modifying the parameter settings that implement Pgpool-II functionality for your installation, you must start the Pgpool-II service.

When Pgpool-II starts, it records its process ID in a `pgpool.conf` file whose name is determined by the `pid_file_name` configuration parameter. The initial value of the `pid_file_name` parameter in the sample file is:

```
pid_file_name = /var/run/edb/pgpool<x.y>/edb-pgpool-<x.y>.pid
```

Where <x.y> is the Pgpool release version.

!!! Note The operating system may remove the contents of the `/var/run` directory (including the `pgpool` directory) during a reboot. You should not use the `/var/run/edb/pgpool` directory as the location for the `pgpool.pid` file. Modify the `pid_file_name` parameter to specify a safer directory location.

5 Connecting a Client to Pgpool-II

Client applications should connect directly to the Pgpool-II listener port on the Pgpool-II host. For example, to connect to the `edb` database (while using Pgpool-II functionality), enter:

```
psql -d edb -U enterprisedb -h localhost -p 9999
```

When invoked at the `psql` prompt, the following `SHOW` command keywords display Pgpool-II information:

Command	Information Provided
`SHOW pool_status`	Displays Pgpool-II configuration parameters and their name, value, and description.
`SHOW pool_nodes`	Displays a list of all configured nodes.
`SHOW pool_processes`	Displays a list of all Pgpool-II processes waiting for connections or dealing with a connection.
`SHOW pool_pools`	Displays a list of pools.
`SHOW pool_version`	Displays the Pgpool-II release number.
`PGPOOL SHOW`	Displays the configuration parameter value.
`PGPOOL SET`	Changes a configuration parameter.
`PGPOOL RESET`	Restores the configuration parameter to the default value.
`SHOW POOL_CACHE`	Displays cache storage statistics.
`SHOW POOL_HEALTH_CHECK_STATS`	Displays health check statistics.
`SHOW POOL_BACKEND_STATS`	Displays backend SQL command statistics.

To view more information about `SHOW` command options, visit the Pgpool project site.

!!! Note

```
SHOW POOL_HEALTH_CHECK_STATS and SHOW POOL_BACKEND_STATS commands are available
from Pgpool version 4.2 onwards.
```

6 Upgrading Pgpool-II and Extensions

The following section outlines the Pgpool and Pgpool extensions upgrade process.

Upgrading Pgpool-II

The following section outlines the Pgpool-II minor version upgrade process (for example, to upgrade from 3.7.13 to 3.7.14):

Assume the identity of the root user and invoke the following command:

On RHEL/CentOS 7:

```
yum upgrade edb-pgpool<xx>
```

where <xx> is the Pgpool version you want to upgrade. For example, to upgrade from Pgpool 3.7.13 to 3.7.14, execute the following command:

```
yum upgrade edb-pgpool37
```

On RHEL/CentOS 8:

```
dnf upgrade edb-pgpool<xx>
```

On Debian/Ubuntu:

```
apt-get upgrade edb-pgpool<xx>
```

On SLES 12:

```
zypper upgrade edb-pgpool<xx>
```

Upgrading Pgpool-II Extensions

The following section outlines the Pgpool-II extensions upgrade process.

To upgrade from older versions of Pgpool extensions to the latest version, assume superuser privileges and execute the following command:

On RHEL/CentOS 7:

```
yum upgrade edb-as<xx>-pgpool<yy>-extensions
```

Where `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool extension version.

!!! Note Only minor version upgrade is supported (for example, you can upgrade from 3.6.20 to 3.6.21 extension, but not 3.7.14).

On RHEL/CentOS 8:

```
dnf upgrade edb-as<xx>-pgpool<yy>-extensions
```

On Debian/Ubuntu:

```
apt-get upgrade edb-as<xx>-pgpool<yy>-extensions
```

On SLES 12:

```
zypper upgrade edb-as<xx>-pgpool<yy>-extensions
```

7 Uninstalling Pgpool-II and Extensions

The following section outlines the process of uninstalling Pgpool-II and its extensions.

Uninstalling Pgpool-II

Uninstalling Pgpool-II on an RHEL/CentOS Host

To uninstall Pgpool-II, assume the identity of the root user and invoke the following command:

On RHEL/CentOS 7:

```
yum erase edb-pgpool<xx>
```

On RHEL/CentOS 8:

```
dnf erase edb-pgpool<xx>
```

Where `<xx>` is the Pgpool version.

Uninstalling Pgpool-II on a Debian/Ubuntu Host

To uninstall Pgpool-II on a Debian/Ubuntu host, invoke the following command:

```
apt-get remove -y edb-pgpool<xx>
```

Where `<xx>` is the Pgpool version you want to uninstall

Uninstalling Pgpool-II on a SLES 12 Host

To uninstall Pgpool-II on a SLES host, assume the identity of the root user and invoke the following command:

```
zypper remove edb-pgpool<xx>
```

Where <xx> is the Pgpool version you wish to uninstall.

Uninstalling Pgpool-II Linux Uninstaller

The Pgpool-II graphical installer creates an uninstaller in the installation directory. If you have used the default installation directory, i.e. `/opt/edb`, then uninstaller will be in the `/opt/edb/pgpool<x.y>` (where <x.y> is the Pgpool version you have installed).

1. Navigate into the directory that contains the uninstaller and assume superuser privileges. Open the uninstaller and click `Yes` to begin uninstalling Pgpool-II.

2. The uninstallation process begins. Click `OK` when the uninstallation completes.

Uninstalling Pgpool-II Extensions

The following section outlines the process of uninstalling Pgpool-II and its extensions.

Uninstalling Pgpool-II Extensions on an RHEL/CentOS Host

To remove extensions from the server, execute the following command:

On RHEL/CentOS 7:

```
yum erase edb-as<xx>-pgpool<yy>-extensions
```

On RHEL/CentOS 8:

```
dnf erase edb-as<xx>-pgpool<yy>-extensions
```

Where `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version.

Uninstalling Pgpool-II Extensions on a SLES 12 Host

To uninstall Pgpool-II extensions on a SLES host, assume the identity of the root user and invoke the following command:

```
zypper remove edb-as<xx>-pgpool<yy>-extensions
```

Where `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version you want to uninstall.

Uninstalling Pgpool-II Extensins on a Debian/Ubuntu Host

To uninstall Pgpool-II extensions on a Debian/Ubuntu host, invoke the following command:

```
apt-get remove -y edb-as<xx>-pgpool<yy>-extensions
```

Where `<xx>` is the EDB Postgres Advanced Server version, and `<yy>` is the Pgpool-II extension version you want to uninstall.

Uninstalling Pgpool-II Extensions Linux Graphical Uninstaller

The Pgpool-II extensions graphical installer creates an uninstaller in the installation directory. If you have used the default installation directory, i.e. `/opt/edb/as<xx>`, then uninstaller will be in the `/opt/edb/as<xx>` (where <xx> is the EDB Postgres Advanced Server version you have installed).

1. Navigate into the directory that contains the uninstaller and assume superuser privileges. Open the uninstaller and click `Yes` to begin uninstalling Pgpool-II extensions.
2. The uninstallation process begins. Click `OK` when the uninstallation completes.