



Advanced Server Upgrade Guide

Version 12

0	Advanced Server Upgrade Guide	3
1	Requirements Overview	3
1.1	Supported Platforms	3
!!!Note Advanced Server is no longer supported on RHEL/CentOS/OEL 6.x platforms. It is strongly recommended that EDB products running on these platforms be migrated to a supported platform.		
2	Limitations	44
3	Upgrading an Installation With pg_upgrade	5
3.1	Performing an Upgrade Overview	6
3.1.1	Linking versus Copying	8
3.2	Invoking pg_upgrade	9
3.2.1	Command Line Options - Reference	11
3.3	Upgrading to Advanced Server 12 - Step-by-Step	14
3.4	Upgrading a pgAgent Installation	27
3.5	pg_upgrade Troubleshooting	28
3.6	Reverting to the Old Cluster	29
4	Performing a Minor Version Update of an RPM Installation	30
5	Using StackBuilder Plus to Perform a Minor Version Update	31
6	Migration to Version 12	36

0 Advanced Server Upgrade Guide

The EDB Postgres Advanced Server Upgrade Guide is a comprehensive guide about upgrading EDB Postgres Advanced Server (Advanced Server). In this guide you will find detailed information about using:

- `pg_upgrade` to upgrade from an earlier version of Advanced Server to Advanced Server 12.
 - `yum` to perform a minor version upgrade on a Linux host.
 - `StackBuilder Plus` to perform a minor version upgrade on a Windows host.
-

1 Requirements Overview

The following sections detail the supported platforms for EDB Postgres Advanced Server 12.

1.1 Supported Platforms

For information about the platforms and versions supported by Advanced Server, visit the EDB website at:

<https://www.enterprisedb.com/services-support/edb-supported-products-and-platforms#epas>

!!!Note Advanced Server is no longer supported on RHEL/CentOS/OEL 6.x platforms. It is strongly recommended that EDB products running on these platforms be migrated to a supported platform.

2 Limitations

The following limitations apply to EDB Postgres Advanced Server:

- The `data` directory of a production database should not be stored on an NFS file system.
- The `pg_upgrade` utility cannot upgrade a partitioned table if a foreign key refers to the partitioned table.
- If you are upgrading from the version 9.4 server or a lower version of Advanced Server, and you use partitioned tables that include a `SUBPARTITION BY` clause, you must use `pg_dump` and `pg_restore` to upgrade an existing Advanced Server installation to a later version of Advanced Server. To upgrade, you must:
 1. Use `pg_dump` to preserve the content of the subpartitioned table.
 2. Drop the table from the Advanced Server 9.4 database or a lower version of Advanced Server database.
 3. Use `pg_upgrade` to upgrade the rest of the Advanced Server database to a more recent version.
 4. Use `pg_restore` to restore the subpartitioned table to the latest upgraded Advanced Server database.
- If you perform an upgrade of the Advanced Server installation, you

must rebuild any hash-partitioned table on the upgraded server.

3 Upgrading an Installation With `pg_upgrade`

While minor upgrades between versions are fairly simple and require only the installation of new executables, past major version upgrades has been both expensive and time consuming. `pg_upgrade` facilitates migration between any version of Advanced Server (version 9.0 or later), and any subsequent release of Advanced Server that is supported on the same platform.

Without `pg_upgrade`, to migrate from an earlier version of Advanced Server to Advanced Server 12, you must export all of your data using `pg_dump`, install the new release, run `initdb` to create a new cluster, and then import your old data.

`pg_upgrade` can reduce both the amount of time required and the disk space required for many major-version upgrades.

The `pg_upgrade` utility performs an in-place transfer of existing data between Advanced Server and any subsequent version.

Several factors determine if an in-place upgrade is practical:

- The on-disk representation of user-defined tables must not change between the original version and the upgraded version.
- The on-disk representation of data types must not change between the original version and the upgraded version.
- To upgrade between major versions of Advanced Server with

`pg_upgrade`, both versions must share a common binary representation for each data type. Therefore, you cannot use `pg_upgrade` to migrate from a 32-bit to a 64-bit Linux platform.

Before performing a version upgrade, `pg_upgrade` will verify that the two clusters (the old cluster and the new cluster) are compatible.

If the upgrade involves a change in the on-disk representation of database objects or data, or involves a change in the binary representation of data types, `pg_upgrade` will be unable to perform the upgrade; to upgrade, you will have to `pg_dump` the old data and then import that data into the new cluster.

The `pg_upgrade` executable is distributed with Advanced Server 12, and is installed as part of the `Database Server` component; no additional installation or configuration steps are required.

3.1 Performing an Upgrade Overview

To upgrade an earlier version of Advanced Server to the current version, you must:

- Install the current version of Advanced Server. The new installation must contain the same supporting server components as the old installation.
- Empty the target database or create a new target cluster with `initdb`.
- Place the `pg_hba.conf` file for both databases in `trust` authentication mode (to avoid authentication conflicts).
- Shut down the old and new Advanced Server services.

- Invoke the `pg_upgrade` utility.

When `pg_upgrade` starts, it performs a compatibility check to ensure that all required executables are present and contain the expected version numbers. The verification process also checks the old and new `$PGDATA` directories to ensure that the expected files and subdirectories are in place. If the verification process succeeds, `pg_upgrade` starts the old `postmaster` and runs `pg_dumpall --schema-only` to capture the metadata contained in the old cluster. The script produced by `pg_dumpall` is used in a later step to recreate all user-defined objects in the new cluster.

Note that the script produced by `pg_dumpall` recreates only user-defined objects and not system-defined objects. The new cluster will *already* contain the system-defined objects created by the latest version of Advanced Server.

After extracting the metadata from the old cluster, `pg_upgrade` performs the bookkeeping tasks required to sync the new cluster with the existing data.

`pg_upgrade` runs the `pg_dumpall` script against the new cluster to create (empty) database objects of the same shape and type as those found in the old cluster. Then, `pg_upgrade` links or copies each table and index from the old cluster to the new cluster.

If you are upgrading to Advanced Server 12 and have installed the `edb_dblink_oci` or `edb_dblink_libpq` extension, you must drop the extension before performing an upgrade. To drop the extension, connect to the server with the psql or PEM client, and invoke the commands:

```
DROP EXTENSION edb_dblink_oci;  
DROP EXTENSION edb_dblink_libpq;
```

When you have completed upgrading, you can use the `CREATE`

EXTENSION command to add the current versions of the extensions to your installation.

3.1.1 Linking versus Copying

When invoking **pg_upgrade**, you can use a command-line option to specify whether **pg_upgrade** should *copy* or *link* each table and index in the old cluster to the new cluster.

Linking is much faster because **pg_upgrade** simply creates a second name (a hard link) for each file in the cluster; linking also requires no extra workspace because **pg_upgrade** does not make a copy of the original data. When linking the old cluster and the new cluster, the old and new clusters share the data; note that after starting the new cluster, your data can no longer be used with the previous version of Advanced Server.

If you choose to copy data from the old cluster to the new cluster, **pg_upgrade** will still reduce the amount of time required to perform an upgrade compared to the traditional **dump/restore** procedure. **pg_upgrade** uses a file-at-a-time mechanism to copy data files from the old cluster to the new cluster (versus the row-by-row mechanism used by **dump/restore**). When you use **pg_upgrade**, you avoid building indexes in the new cluster; each index is simply copied from the old cluster to the new cluster. Finally, using a **dump/restore** procedure to upgrade requires a great deal of workspace to hold the intermediate text-based dump of all of your data, while **pg_upgrade** requires very little extra workspace.

Data that is stored in user-defined tablespaces is not copied to the new cluster; it stays in the same location in the file system, but is copied into a

subdirectory whose name reflects the version number of the new cluster. To manually relocate files that are stored in a tablespace after upgrading, move the files to the new location and update the symbolic links (located in the `pg_tblspc` directory under your cluster's `data` directory) to point to the files.

3.2 Invoking pg_upgrade

When invoking `pg_upgrade`, you must specify the location of the old and new cluster's `PGDATA` and executable `(/bin)` directories, as well as the name of the Advanced Server superuser, and the ports on which the installations are listening. A typical call to invoke `pg_upgrade` to migrate from Advanced Server 11 to Advanced Server 12 takes the form:

```
pg_upgrade
--old-datadir <path_to_11_data_directory>
--new-datadir <path_to_12_data_directory>
--user <superuser_name>
--old-bindir <path_to_11_bin_directory>
--new-bindir <path_to_12_bin_directory>
--old-port <11_port> --new-port <12_port>
```

Where:

```
--old-datadir path_to_11_data_directory
```

Use the `--old-datadir` option to specify the complete path to the `data` directory within the Advanced Server 11 installation.

```
--new-datadir path_to_12_data_directory
```

Use the `--new-datadir` option to specify the complete path to the `data` directory within the Advanced Server 12 installation.

```
--username superuser_name
```

Include the `--username` option to specify the name of the Advanced Server superuser. The superuser name should be the same in both versions of Advanced Server. By default, when Advanced Server is installed in Oracle mode, the superuser is named `enterprisedb`. If installed in PostgreSQL mode, the superuser is named `postgres`.

If the Advanced Server superuser name is not the same in both clusters, the clusters will not pass the `pg_upgrade` consistency check.

```
--old-bindir path_to_11_bin_directory
```

Use the `--old-bindir` option to specify the complete path to the `bin` directory in the Advanced Server 11 installation.

```
--new-bindir path_to_12_bin_directory
```

Use the `--new-bindir` option to specify the complete path to the `bin` directory in the Advanced Server 12 installation.

```
--old-port 11_port
```

Include the `--old-port` option to specify the port on which Advanced Server 11 listens for connections.

```
--new-port 12_port
```

Include the `--new-port` option to specify the port on which Advanced Server 12 listens for connections.

3.2.1 Command Line Options - Reference

`pg_upgrade` accepts the following command line options; each option is available in a long form or a short form:

```
-b path_to_old_bin_directory  
--old-bindir path_to_old_bin_directory
```

Use the `-b` or `--old-bindir` keyword to specify the location of the old cluster's executable directory.

```
-B path_to_new_bin_directory  
--new-bindir path_to_new_bin_directory
```

Use the `-B` or `--new-bindir` keyword to specify the location of the new cluster's executable directory.

```
-c  
--check
```

Include the `-c` or `--check` keyword to specify that `pg_upgrade` should perform a consistency check on the old and new cluster without performing a version upgrade.

```
-d path_to_old_data_directory  
--old-datadir path_to_old_data_directory
```

Use the `-d` or `--old-datadir` keyword to specify the location of the old cluster's `data` directory.

```
-D path_to_new_data_directory  
--new-datadir path_to_new_data_directory
```

Use the `-D` or `--new-datadir` keyword to specify the location of the new cluster's `data` directory.

Data that is stored in user-defined tablespaces is not copied to the new cluster; it stays in the same location in the file system, but is copied into a subdirectory whose name reflects the version number of the new cluster. To manually relocate files that are stored in a tablespace after upgrading, you must move the files to the new location and update the symbolic links (located in the `pg_tblspc` directory under your cluster's `data` directory) to point to the files.

```
-j
--jobs
```

Include the `-j` or `--jobs` keyword to specify the number of simultaneous processes or threads to use during the upgrade.

```
-k
--link
```

Include the `-k` or `--link` keyword to create a hard link from the new cluster to the old cluster. See [Linking versus Copying](#) for more information about using a symbolic link.

```
-o options
--old-options options
```

Use the `-o` or `--old-options` keyword to specify options that will be passed to the old `postgres` command. Enclose options in single or double quotes to ensure that they are passed as a group.

```
-O options
--new-options options
```

Use the `-O` or `--new-options` keyword to specify options to be passed to the new `postgres` command. Enclose options in single or double quotes to ensure that they are passed as a group.

```
-p old_port_number
```

```
--old-port old_port_number
```

Include the `-p` or `--old-port` keyword to specify the port number of the Advanced Server installation that you are upgrading.

```
-P new_port_number
--new-port new_port_number
```

Include the `-P` or `--new-port` keyword to specify the port number of the new Advanced Server installation.

!!! Note If the original Advanced Server installation is using port number `5444` when you invoke the Advanced Server 12 installer, the installer will recommend using listener port `5445` for the new installation of Advanced Server.

```
-r
--retain
```

During the upgrade process, `pg_upgrade` creates four append-only log files; when the upgrade is completed, `pg_upgrade` deletes these files. Include the `-r` or `--retain` option to specify that the server should retain the `pg_upgrade` log files.

```
-U user_name
--username user_name
```

Include the `-U` or `--username` keyword to specify the name of the Advanced Server database superuser. The same superuser must exist in both clusters.

```
-v
--verbose
```

Include the `-v` or `--verbose` keyword to enable verbose output during the upgrade process.

```
-V  
--version
```

Use the `-V` or `--version` keyword to display version information for `pg_upgrade`.

```
-?  
-h  
--help
```

Use `-, -h,` or `--help` options to display `pg_upgrade` help information.

3.3 Upgrading to Advanced Server 12 - Step-by-Step

You can use `pg_upgrade` to upgrade from an existing installation of Advanced Server into the cluster built by the Advanced Server 12 installer or into an alternate cluster created using the `initdb` command. In this section, we will provide the details of upgrading into the cluster provided by the installer.

The basic steps to perform an upgrade into an empty cluster created with the `initdb` command are the same as the steps to upgrade into the cluster created by the Advanced Server 12 installer, but you can omit Step 2 (`Empty the edb database`), and substitute the location of the alternate cluster when specifying a target cluster for the upgrade.

If a problem occurs during the upgrade process, you can revert to the previous version. See [Reverting to the old cluster](#) Section for detailed

information about this process.

You must be an operating system superuser or Windows Administrator to perform an Advanced Server upgrade.

Step 1 - Install the New Server

Install Advanced Server 12, specifying the same non-server components that were installed during the previous Advanced Server installation. The new cluster and the old cluster must reside in different directories.

Step 2 - Empty the target database

The target cluster must not contain any data; you can create an empty cluster using the `initdb` command, or you can empty a database that was created during the installation of Advanced Server 12. If you have installed Advanced Server in PostgreSQL mode, the installer creates a single database named `postgres`; if you have installed Advanced Server in Oracle mode, it creates a database named `postgres` and a database named `edb`.

The easiest way to empty the target database is to drop the database and then create a new database. Before invoking the `DROP DATABASE` command, you must disconnect any users and halt any services that are currently using the database.

On Windows, navigate through the `Control Panel` to the `Services` manager; highlight each service in the `Services` list, and select `Stop`.

On Linux, open a terminal window, assume superuser privileges, and manually stop each service; for example, invoke the following:

```
service edb-pgagent-12 stop
```

to stop the pgAgent service.

After stopping any services that are currently connected to Advanced

Server, you can use the EDB-PSQL command line client to drop and create a database. When the client opens, connect to the `template1` database as the database superuser; if prompted, provide authentication information. Then, use the following command to drop your database:

```
DROP DATABASE <database_name>;
```

Where `database_name` is the name of the database.

Then, create an empty database based on the contents of the `template1` database.

```
CREATE DATABASE <database_name>;
```

Step 3 - Set both servers in trust mode

During the upgrade process, `pg_upgrade` will connect to the old and new servers several times; to make the connection process easier, you can edit the `pg_hba.conf` file, setting the authentication mode to `trust`. To modify the `pg_hba.conf` file, navigate through the `Start` menu to the `EDB Postgres` menu; to the `Advanced Server` menu, and open the `Expert Configuration` menu; select the `Edit pg_hba.conf` menu option to open the `pg_hba.conf` file.

You must allow trust authentication for the previous Advanced Server installation, and Advanced Server 12 servers. Edit the `pg_hba.conf` file for both installations of Advanced Server as shown in the following figure.

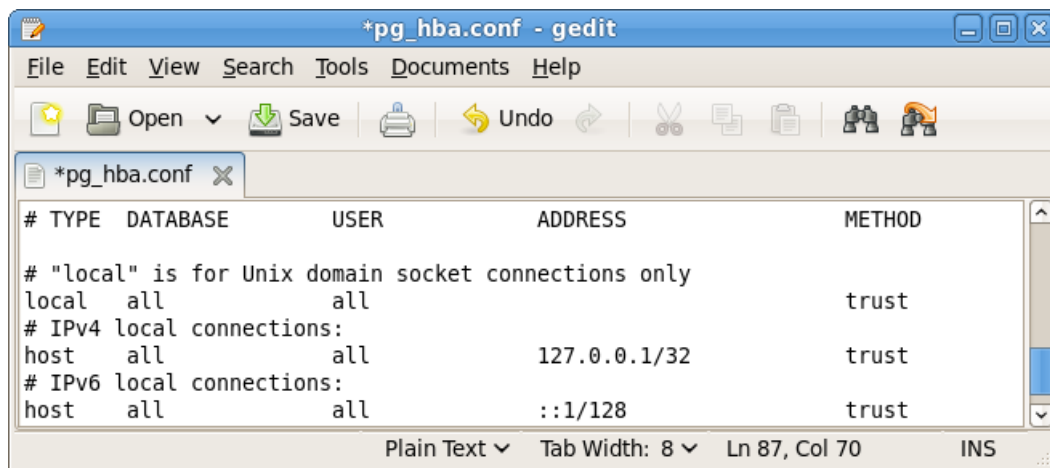


Fig. 1: *Configuring Advanced Server to use trust authentication*

After editing each file, save the file and exit the editor.

If the system is required to maintain **md5** authentication mode during the upgrade process, you can specify user passwords for the database superuser in a password file (**pgpass.conf** on Windows, **.pgpass** on Linux). For more information about configuring a password file, see the PostgreSQL Core Documentation, available at:

<https://www.postgresql.org/docs/current/static/libpq-pgpass.html>

Step 4 - Stop All Component Services and Servers

Before you invoke **pg_upgrade**, you must stop any services that belong to the original Advanced Server installation, Advanced Server 12, or the supporting components. This ensures that a service will not attempt to access either cluster during the upgrade process.

The services that are most likely to be running in your installation are:

Service:	On Linux:	On Windows:
EnterpriseDB Postgres Advanced Server 9.6	edb-as-9.6	edb-as-9.6

Service:	On Linux:	On Windows:
EnterpriseDB Postgres Advanced Server 10	edb-as-10	edb-as-10
EnterpriseDB Postgres Advanced Server 11	edb-as-11	edb-as-11
EnterpriseDB Postgres Advanced Server 12	edb-as-12	edb-as-12
Advanced Server 9.6 Scheduling Agent (pgAgent)	edb-pgagent-9.6	EnterpriseDB Postgres Advanced Server 9.6 Scheduling Agent
Infinite Cache 9.6	edb-icache	N/A
Infinite Cache 10	edb-icache	N/A
PgBouncer	Pgbouncer	Pgbouncer
PgBouncer 1.6	ppas-pgbouncer- 1.6 or ppas- pgbouncer16	ppas-pgbouncer-1.6
PgBouncer 1.7	edb-pgbouncer- 1.7	edb-pgbouncer-1.7
PgPool	ppas-pgpool	N/A
PgPool 3.4	ppas-pgpool-3.4 or ppas-pgpool34	N/A
pgPool-II	edb-pgpool-3.5	N/A
Slony 9.6	edb-slony- replication-9.6	edb-slony-replication-9.6
xDB Subscription Server	edb-xdbsubserver- 90	Subscription Service 90
xDB Subscription Server	edb-xdbsubserver- 91	Subscription Service 91

Service:	On Linux:	On Windows:
EDB Replication Server v6.x	edb-xdbpubserver	Publication Service for xDB Replication Server
EDB Subscription Server v6.x	edb-xdbsubserver	Subscription Service for xDB Replication Server

To stop a service on Windows:

Open the **Services** applet; highlight each Advanced Server or supporting component service displayed in the list, and select **Stop**.

To stop a service on Linux:

Open a terminal window and manually stop each service at the command line.

Step 5 for Linux only - Assume the identity of the cluster owner

If you are using Linux, assume the identity of the Advanced Server cluster owner. (The following example assumes Advanced Server was installed in the default, compatibility with Oracle database mode, thus assigning **enterprisedb** as the cluster owner. If installed in compatibility with PostgreSQL database mode, **postgres** is the cluster owner.)

```
su - enterprisedb
```

Enter the Advanced Server cluster owner password if prompted. Then, set the path to include the location of the **pg_upgrade** executable:

```
export PATH=$PATH:/usr/edb/as12/bin
```

During the upgrade process, **pg_upgrade** writes a file to the current working directory of the **enterprisedb** user; you must invoke **pg_upgrade** from a directory where the **enterprisedb** user has **write** privileges. After performing the above commands, navigate to a

directory in which the `enterprisedb` user has sufficient privileges to write a file.

```
cd /tmp
```

Proceed to Step 6.

Step 5 for Windows only - Assume the identity of the cluster owner

If you are using Windows, open a terminal window, assume the identity of the Advanced Server cluster owner and set the path to the `pg_upgrade` executable.

If the `--serviceaccount service_account_user` parameter was specified during the initial installation of Advanced Server, then `service_account_user` is the Advanced Server cluster owner and is the user to be given with the `RUNAS` command.

```
RUNAS /USER:service_account_user "CMD.EXE"
SET PATH=%PATH%;C:\Program Files\edb\as12\bin
```

During the upgrade process, `pg_upgrade` writes a file to the current working directory of the service account user; you must invoke `pg_upgrade` from a directory where the service account user has `write` privileges. After performing the above commands, navigate to a directory in which the service account user has sufficient privileges to write a file.

```
cd %TEMP%
```

Proceed to Step 6.

If the `--serviceaccount` parameter was omitted during the initial installation of Advanced Server, then the default owner of the Advanced Server service and the database cluster is `NT`

`AUTHORITY\NetworkService`.

When `NT AUTHORITY\NetworkService` is the service account user, the `RUNAS` command may not be usable as it prompts for a password and the `NT AUTHORITY\NetworkService` account is not assigned a password. Thus, there is typically a failure with an error message such as, “Unable to acquire user password”.

Under this circumstance a Windows utility program named `PsExec` must be used to run `CMD.EXE` as the service account `NT AUTHORITY\NetworkService`.

The `PsExec` program must be obtained by downloading `PsTools`, which is available at the following site:

<https://technet.microsoft.com/en-us/sysinternals/bb897553.aspx>.

You can then use the following command to run `CMD.EXE` as `NT AUTHORITY\NetworkService`, and then set the path to the `pg_upgrade` executable.

```
psexec.exe -u "NT AUTHORITY\NetworkService" CMD.EXE  
SET PATH=%PATH%;C:\Program Files\edb\as12\bin
```

During the upgrade process, `pg_upgrade` writes a file to the current working directory of the service account user; you must invoke `pg_upgrade` from a directory where the service account user has `write` privileges. After performing the above commands, navigate to a directory in which the service account user has sufficient privileges to write a file.

```
cd %TEMP%
```

Proceed with Step 6.

Step 6 - Perform a consistency check

Before attempting an upgrade, perform a consistency check to assure that the old and new clusters are compatible and properly configured. Include the `--check` option to instruct `pg_upgrade` to perform the consistency check.

The following example demonstrates invoking `pg_upgrade` to perform a consistency check on Linux:

```
pg_upgrade -d /var/lib/edb/as11/data
-D /var/lib/edb/as12/data -U enterprisedb
-b /usr/edb/as11/bin -B /usr/edb/as12/bin -p 5444 -
P 5445 --check
```

If the command is successful, it will return `*Clusters are compatible*`.

If you are using Windows, you must quote any directory names that contain a space:

```
pg_upgrade.exe
-d "C:\Program Files\ PostgresPlus\11AS\data"
-D "C:\Program Files\edb\as12\data" -U enterprisedb
-b "C:\Program Files\PostgresPlus\11AS\bin"
-B "C:\Program Files\edb\as12\bin" -p 5444 -P 5445
--check
```

During the consistency checking process, `pg_upgrade` will log any discrepancies that it finds to a file located in the directory from which `pg_upgrade` was invoked. When the consistency check completes, review the file to identify any missing components or upgrade conflicts. You must resolve any conflicts before invoking `pg_upgrade` to perform a version upgrade.

If `pg_upgrade` alerts you to a missing component, you can use StackBuilder Plus to add the component that contains the component. Before using StackBuilder Plus, you must restart the Advanced Server 12

service. After restarting the service, open StackBuilder Plus by navigating through the **Start** menu to the **Advanced Server 12** menu, and selecting **StackBuilder Plus**. Follow the onscreen advice of the StackBuilder Plus wizard to download and install the missing components.

When **pg_upgrade** has confirmed that the clusters are compatible, you can perform a version upgrade.

Step 7 - Run pg_upgrade

After confirming that the clusters are compatible, you can invoke **pg_upgrade** to upgrade the old cluster to the new version of Advanced Server.

On Linux:

```
pg_upgrade -d /var/lib/edb/as11/data
-D /var/lib/edb/as12/data -U enterprisedb
-b /usr/edb/as11/bin -B /usr/edb/as12/bin -p 5444 -
P 5445
```

On Windows:

```
pg_upgrade.exe -d "C:\Program
Files\PostgresPlus\11AS\data"
-D "C:\Program Files\edb\as12\data" -U enterprisedb
-b "C:\Program Files\PostgresPlus\11AS\bin"
-B "C:\Program Files\edb\as12\bin" -p 5444 -P 5445
```

pg_upgrade will display the progress of the upgrade onscreen:

```
$ pg_upgrade -d /var/lib/edb/as11/data -D
/var/lib/edb/as12/data -U
enterprisedb -b /usr/edb/as11/bin -B
/usr/edb/as12/bin -p 5444 -P 5445
Performing Consistency Checks
```

Checking current, bin, and data directories

ok

Checking cluster versions

ok

Checking database user is a superuser

ok

Checking for prepared transactions

ok

Checking for reg* system OID user data types

ok

Checking for contrib/ism with bigint-passing
mismatch ok

Creating catalog dump

ok

Checking for presence of required libraries

ok

Checking database user is a superuser

ok

Checking for prepared transactions

ok

If pg_upgrade fails after this point, you must re-initdb the
new cluster before continuing.

Performing Upgrade

Analyzing all rows in the new cluster

ok

Freezing all rows on the new cluster

ok

Deleting files from new pg_clog

ok

Copying old pg_clog to new server


```
ok
Setting next transaction ID for new cluster
ok
Resetting WAL archives
ok
Setting frozenxid counters in new cluster
ok
Creating databases in the new cluster
ok
Adding support functions to new cluster
ok
Restoring database schema to new cluster
ok
Removing support functions from new cluster
ok
Copying user relation files
ok
```

```
Setting next OID for new cluster
ok
Creating script to analyze new cluster
ok
Creating script to delete old cluster
ok
```

Upgrade Complete

Optimizer statistics are not transferred by
pg_upgrade so,
once you start the new server, consider running:
 analyze_new_cluster.sh

Running this script will delete the old cluster's
data files:
 delete_old_cluster.sh

While `pg_upgrade` runs, it may generate SQL scripts that handle special circumstances that it has encountered during your upgrade. For example, if the old cluster contains large objects, you may need to invoke a script that defines the default permissions for the objects in the new cluster. When performing the pre-upgrade consistency check `pg_upgrade` will alert you to any script that you may be required to run manually.

You must invoke the scripts after `pg_upgrade` completes. To invoke the scripts, connect to the new cluster as a database superuser with the EDB-PSQL command line client, and invoke each script using the `\i` option:

```
\i complete_path_to_script/script.sql
```

It is generally unsafe to access tables referenced in rebuild scripts until the rebuild scripts have completed; accessing the tables could yield incorrect results or poor performance. Tables not referenced in rebuild scripts can be accessed immediately.

If `pg_upgrade` fails to complete the upgrade process, the old cluster will be unchanged, except that `$PGDATA/global/pg_control` is renamed to `pg_control.old` and each tablespace is renamed to `tablespace.old`. To revert to the pre-invocation state:

1. Delete any tablespace directories created by the new cluster.
2. Rename `$PGDATA/global/pg_control`, removing the `.old` suffix.
3. Rename the old cluster tablespace directory names, removing the `.old` suffix.
4. Remove any database objects (from the new cluster) that may have been moved before the upgrade failed.

After performing these steps, resolve any upgrade conflicts encountered before attempting the upgrade again.

When the upgrade is complete, `pg_upgrade` may also recommend vacuuming the new cluster, and will provide a script that allows you to

delete the old cluster.

!!! Note Before removing the old cluster, ensure that the cluster has been upgraded as expected, and that you have preserved a backup of the cluster in case you need to revert to a previous version.

Step 8 - Restore the authentication settings in the `pg_hba.conf` file

If you modified the `pg_hba.conf` file to permit `trust` authentication, update the contents of the `pg_hba.conf` file to reflect your preferred authentication settings.

Step 9 - Move and identify user-defined tablespaces (Optional)

If you have data stored in a user-defined tablespace, you must manually relocate tablespace files after upgrading; move the files to the new location and update the symbolic links (located in the `pg_tblspc` directory under your cluster's `data` directory) to point to the files.

3.4 Upgrading a pgAgent Installation

If your existing Advanced Server installation uses pgAgent, you can use a script provided with the Advanced Server 12 installer to update pgAgent. The script is named `dbms_job.upgrade.script.sql`, and is located in the `/share/contrib/` directory under your Advanced Server installation.

If you are using `pg_upgrade` to upgrade your installation, you should:

1. Perform the upgrade.
2. Invoke the `dbms_job.upgrade.script.sql` script to update the

catalog files. If your existing pgAgent installation was performed with a script, the update will convert the installation to an extension.

3.5 pg_upgrade Troubleshooting

The troubleshooting tips in this section address problems you may encounter when using `pg_upgrade`.

Upgrade Error - There seems to be a postmaster servicing the cluster

If `pg_upgrade` reports that a postmaster is servicing the cluster, please stop all Advanced Server services and try the upgrade again.

Upgrade Error - fe_sendauth: no password supplied

If `pg_upgrade` reports an authentication error that references a missing password, please modify the `pg_hba.conf` files in the old and new cluster to enable `trust` authentication, or configure the system to use a `pgpass.conf` file.

Upgrade Error - New cluster is not empty; exiting

If `pg_upgrade` reports that the new cluster is not empty, please empty

the new cluster. The target cluster may not contain any user-defined databases.

Upgrade Error - Failed to load library

If the original Advanced Server cluster included libraries that are not included in the Advanced Server 12 cluster, `pg_upgrade` will alert you to the missing component during the consistency check by writing an entry to the `loadable_libraries.txt` file in the directory from which you invoked `pg_upgrade`. Generally, for missing libraries that are not part of a major component upgrade, perform the following steps:

1. Restart the Advanced Server service.

Use StackBuilder Plus to download and install the missing module.
Then:

2. Stop the Advanced Server service.
3. Resume the upgrade process: invoke `pg_upgrade` to perform consistency checking.
4. When you have resolved any remaining problems noted in the consistency checks, invoke `pg_upgrade` to perform the data migration from the old cluster to the new cluster.

3.6 Reverting to the Old Cluster

The method used to revert to a previous cluster varies with the options

specified when invoking `pg_upgrade`:

- If you specified the `--check` option when invoking `pg_upgrade`, an upgrade has not been performed, and no modifications have been made to the old cluster; you can re-use the old cluster at any time.
- If you included the `--link` option when invoking `pg_upgrade`, the data files are shared between the old and new cluster after the upgrade completes. If you have started the server that is servicing the new cluster, the new server has written to those shared files and it is unsafe to use the old cluster.
- If you ran `pg_upgrade` without the `--link` specification or have not started the new server, the old cluster is unchanged, except that the `.old` suffix has been appended to the `$PGDATA/global/pg_control` and tablespace directories.
- To reuse the old cluster, delete the tablespace directories created by the new cluster and remove the `.old` suffix from `$PGDATA/global/pg_control` and the old cluster tablespace directory names and restart the server that services the old cluster.

4 Performing a Minor Version Update of an RPM Installation

If you used an RPM package to install Advanced Server or its supporting components, you can use `yum` to perform a minor version upgrade to a more recent version. To review a list of the package updates that are available for your system, open a command line, assume root privileges, and enter the command:

```
yum check-update <package_name>
```

Where `package_name` is the search term for which you wish to search for updates. Please note that you can include wild-card values in the search term. To use `yum update` to install an updated package, use the command:

```
yum update <package_name>
```

Where `package_name` is the name of the package you wish to update. Include wild-card values in the update command to update multiple related packages with a single command. For example, use the following command to update all packages whose names include the expression `edb`:

```
yum update edb*
```

!!! Note The `yum update` command will only perform an update between minor releases; to update between major releases, you must use `pg_upgrade`.

For more information about using yum commands and options, enter `yum --help` on your command line, or visit:

https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Deployment_Guide/ch-yum.html

5 Using StackBuilder Plus to Perform a Minor Version Update

StackBuilder Plus is supported only on Windows systems.

The StackBuilder Plus utility provides a graphical interface that simplifies the process of updating, downloading, and installing modules that complement your Advanced Server installation. When you install a module with StackBuilder Plus, StackBuilder Plus automatically resolves any software dependencies.

You can invoke StackBuilder Plus at any time after the installation has completed by selecting the **StackBuilder Plus** menu option from the **Apps** menu. Enter your system password (if prompted), and the StackBuilder Plus welcome window opens.



Fig. 1: *The StackBuilder Plus welcome window*

Use the drop-down listbox on the welcome window to select your Advanced Server installation.

StackBuilder Plus requires Internet access; if your installation of Advanced Server resides behind a firewall (with restricted Internet access), StackBuilder Plus can download program installers through a proxy server. The module provider determines if the module can be accessed through an HTTP proxy or an FTP proxy; currently, all updates are transferred via an HTTP proxy and the FTP proxy information is not used.

If the selected Advanced Server installation has restricted Internet access, use the **Proxy Servers** on the **Welcome** window to open the **Proxy servers** dialog (shown in the following figure).

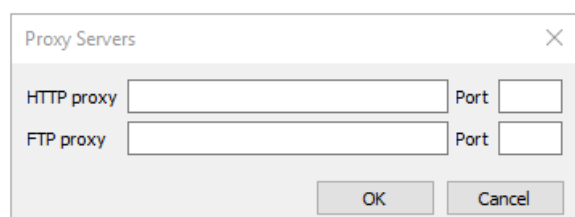


Fig. 2: *The Proxy servers dialog*

Enter the IP address and port number of the proxy server in the **HTTP proxy** on the **Proxy servers** dialog. Currently, all StackBuilder Plus modules are distributed via HTTP proxy (FTP proxy information is ignored). Click **OK** to continue.

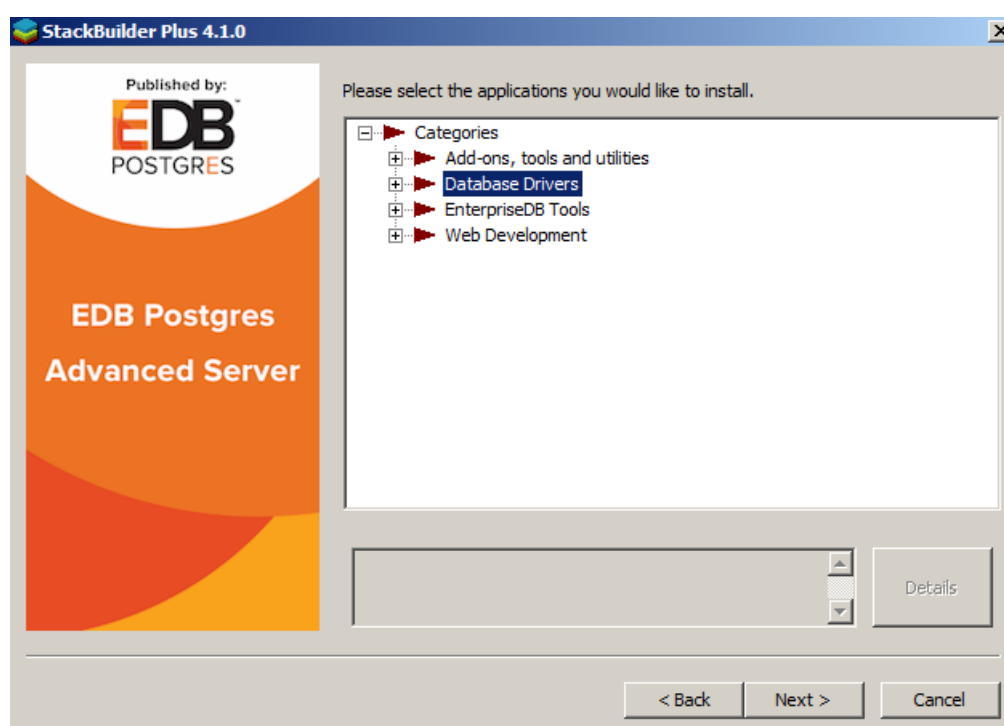
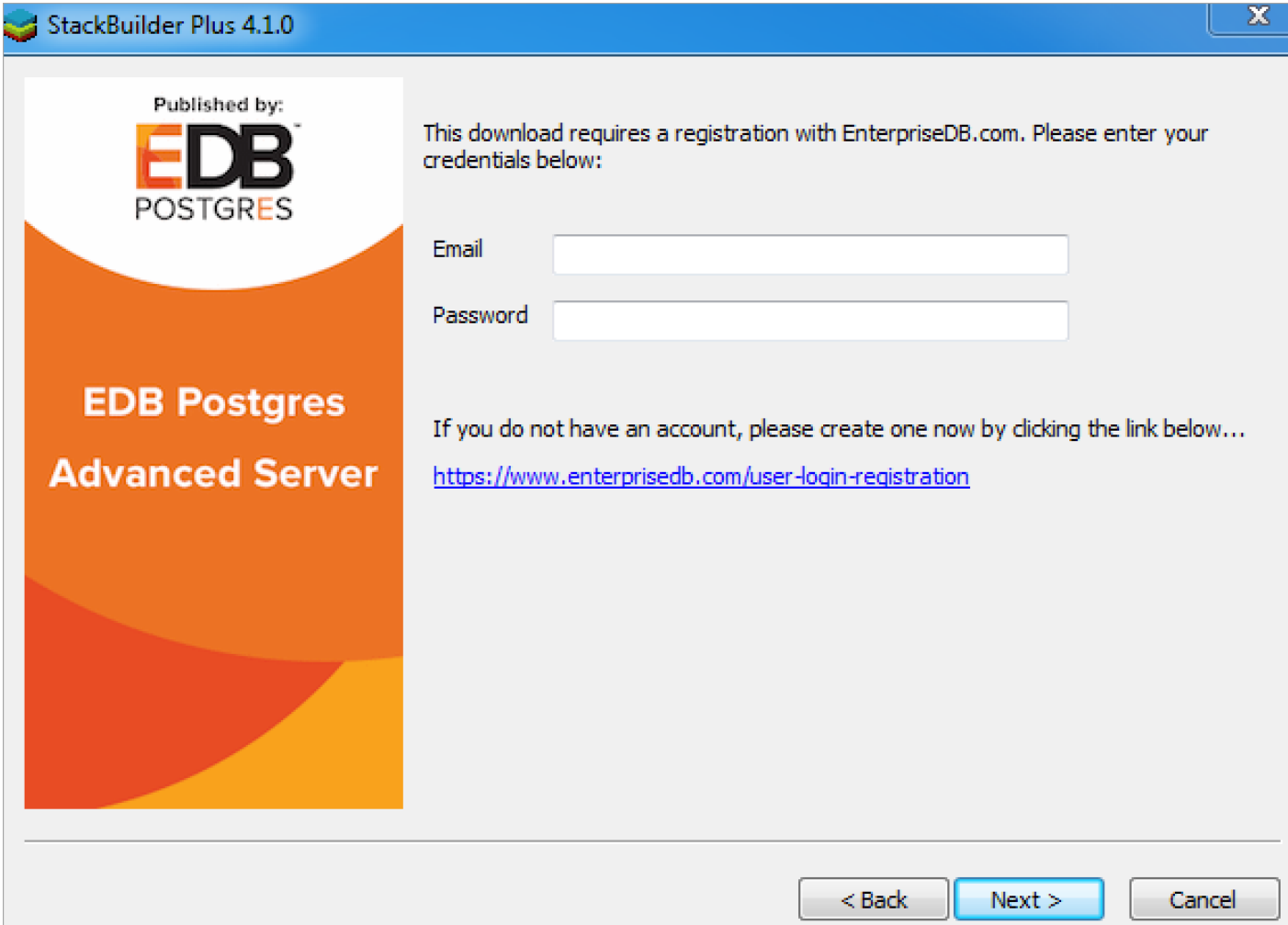


Fig. 3: *The StackBuilder Plus module selection window*

The tree control on the StackBuilder Plus module selection window (shown in the figure) displays a node for each module category. To perform an Advanced Server update, expand the Database Server module in the tree control and check the box to the left of the Advanced Server upgraded

version. Then, click Next.

If prompted, enter your email address and password on the StackBuilder Plus registration window.



The image shows a registration window titled "StackBuilder Plus 4.1.0". On the left is a graphic with the EDB PostgreSQL logo and the text "EDB Postgres Advanced Server". The main area contains a message: "This download requires a registration with EnterpriseDB.com. Please enter your credentials below:". Below this are two input fields labeled "Email" and "Password". Further down, it says "If you do not have an account, please create one now by clicking the link below..." followed by a blue hyperlink: <https://www.enterprisedb.com/user-login-registration>. At the bottom right are three buttons: "< Back", "Next >" (highlighted in blue), and "Cancel".

Fig. 4: *The StackBuilder Plus registration window.*

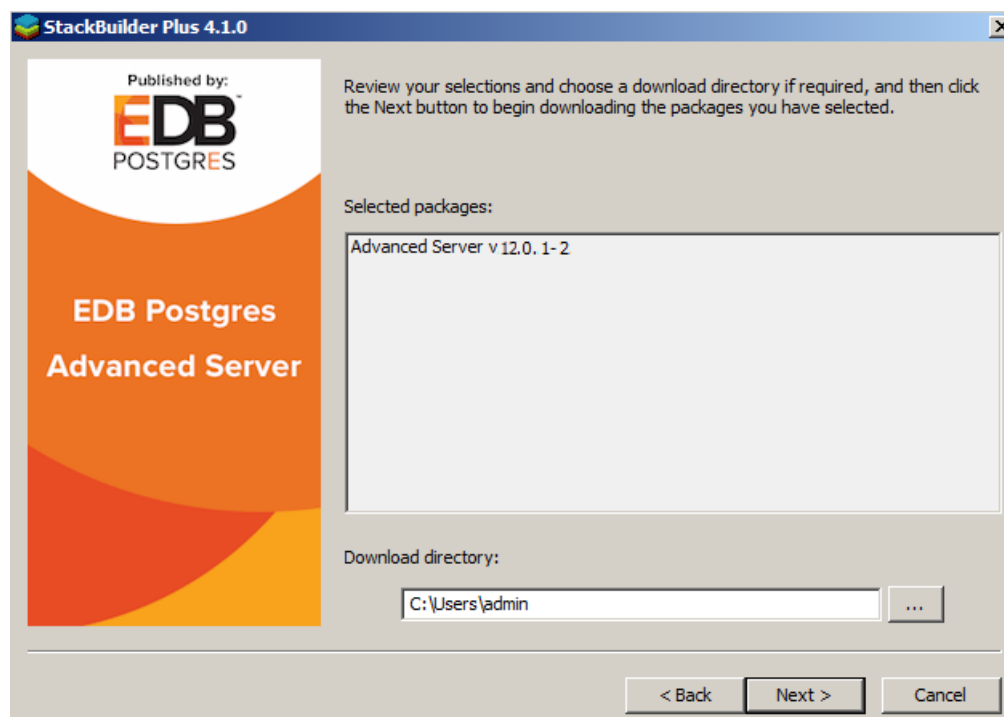


Fig. 5: A summary window displays a list of selected packages

StackBuilder Plus confirms the packages selected. The **Selected packages** dialog will display the name and version of the installer; click **Next** to continue.

When the download completes, a window opens that confirms the installation files have been downloaded and are ready for installation.

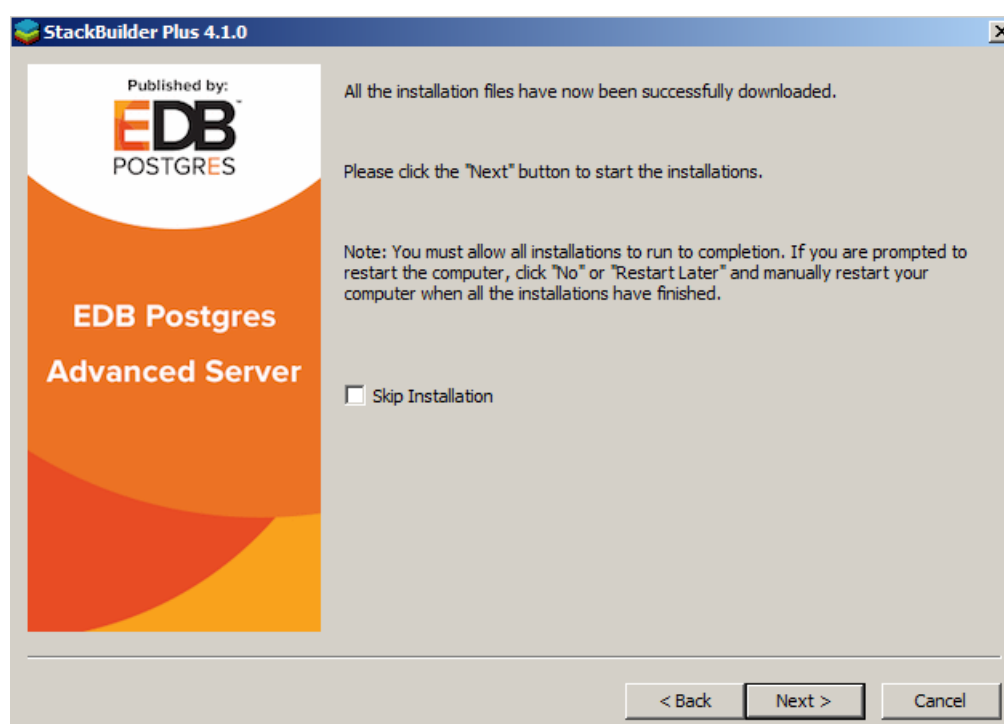


Fig. 6: *Confirmation that the download process is complete*

You can check the box next to **Skip Installation**, and select **Next** to exit StackBuilder Plus without installing the downloaded files, or leave the box unchecked and click **Next** to start the installation process.

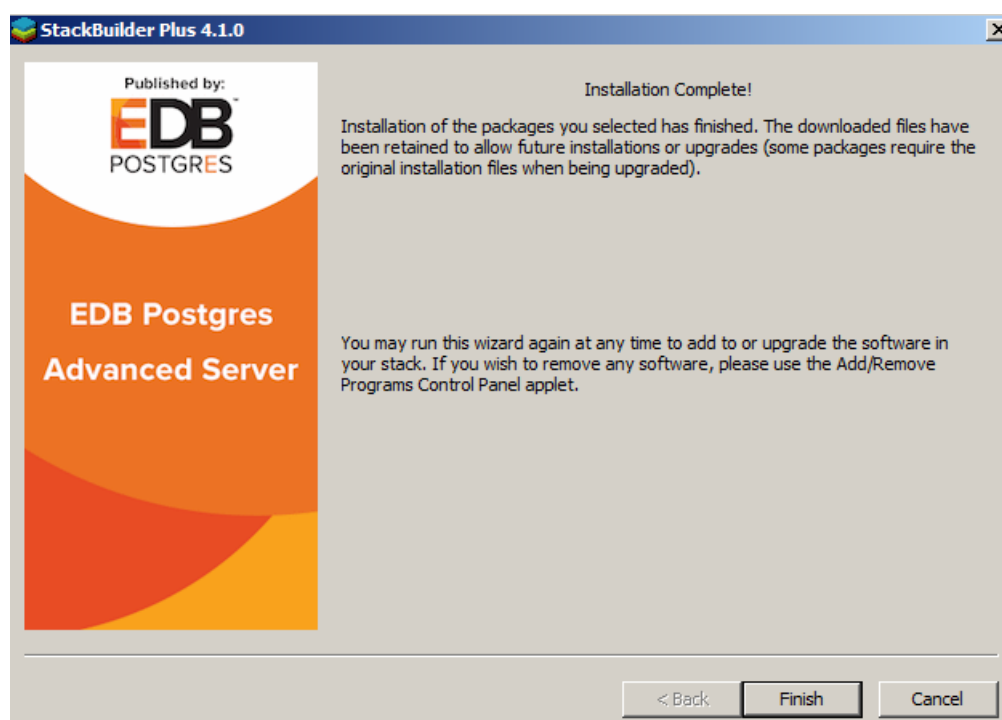


Fig. 7: *StackBuilder Plus confirms the completed installation*

When the upgrade is complete, StackBuilder Plus will alert you to the success or failure of the installation of the requested package. If you were prompted by an installer to restart your computer, reboot now.

!!! Note If the update fails to install, StackBuilder Plus will alert you to the installation error with a popup dialog and write a message to the log file at **%TEMP%**.

6 Migration to Version 12

A dump/restore using `pg_dumpall` or `pg_upgrade` or logical replication is required for migrating data from any previous release. See [Upgrading an Installation With pg_upgrade](#) for information on migrating to new major releases.

Version 12 contains a number of changes that may affect compatibility with previous releases. Listed is the following incompatibilities:

- Remove the special behavior of `oid` columns.

Previously, a normally-invisible `oid` column could be specified during table creation using `WITH OIDS`; that ability has been removed. Columns can still be explicitly declared as type `oid`. Operations on tables that have columns created using `WITH OIDS` will need adjustment.

The system catalogs that previously had hidden `oid` columns now have ordinary `oid` columns. Hence, `SELECT *` will now output those columns, whereas previously they would be displayed only if selected explicitly.

- Remove data types `abstime`, `reltime`, and `tinterval`.

These are obsoleted by SQL-standard types such as `timestamp`.

- Move `recovery.conf` settings into `postgresql.conf`.

`recovery.conf` is no longer used, and the server will not start if that file exists. `recovery.signal` and `standby.signal` files are now used to switch into non-primary mode. The `trigger_file` setting has been renamed to `promote_trigger_file`. The `standby_mode` setting has been removed.

- Do not allow multiple conflicting ``recovery_target*` specifications.

Specifically, only allow one of `recovery_target`,

`recovery_target_lsn`, `recovery_target_name`, `recovery_target_time`, and `recovery_target_xid`.

Previously, multiple different instances of these parameters could be specified, and the last one was honored. Now, only one can be specified, though the same one can be specified multiple times and the last specification is honored.

- Cause recovery to advance to the latest timeline by default.

Specifically, `recovery_target_timeline` now defaults to `latest`. Previously, it defaulted to `current`.

- Refactor code for geometric functions and operators.

This could lead to more accurate, but slightly different, results compared to previous releases. Notably, cases involving `NaN`, underflow, overflow, and division by zero are handled more consistently than before.

- Improve performance by using a new algorithm for output of `real` and `double precision` values.

Previously, displayed floating-point values were rounded to 6 (for `real`) or 15 (for `double precision`) digits by default, adjusted by the value of `extra_float_digits`. Now, whenever `extra_float_digits` is more than zero (as it now is by default), only the minimum number of digits required to preserve the exact binary value are output. The behavior is the same as before when `extra_float_digits` is set to zero or less.

Also, formatting of floating-point exponents is now uniform across platforms: two digits are used unless three are necessary. In previous releases, Windows builds always printed three digits.

- `random()` and `setseed()` now behave uniformly across platforms.

The sequence of `random()` values generated following a `setseed()` call with a particular seed value is likely to be different now than before. However, it will also be repeatable, which was not previously guaranteed because of interference from other uses of random numbers inside the server. The SQL `random()` function now has its own private per-session state to forestall that.

- Change SQL-style `substring()` to have standard-compliant greediness behavior.

In cases where the pattern can be matched in more than one way, the initial subpattern is now treated as matching the least possible amount of text rather than the greatest; for example, a pattern such as `%#"aa*#"%` now selects the first group of a's from the input, not the last group.

- Do not pretty-print the result of `xpath()` or the `XMLTABLE` construct.

In some cases, these functions would insert extra whitespace (newlines and/or spaces) in nodeset values. This is undesirable since depending on usage, the whitespace might be considered semantically significant.

- Rename command-line tool `pg_verify_checksums` to `pg_checksums`.
- In `pg_restore`, require specification of `-f -` to send the dump contents to standard output.

Previously, this happened by default if no destination was specified, but that was deemed to be unfriendly.

- Disallow non-unique abbreviations in psql's `\pset format` command.

Previously, for example, `\pset format` chose `aligned`; it will now

fail since that could equally well mean `asciidoc`.

- In new `btree` indexes, the maximum index entry length is reduced by eight bytes, to improve handling of duplicate entries.

This means that a `REINDEX` operation on an index `pg_upgrade'd from a previous release could potentially fail.

- Cause `DROP IF EXISTS FUNCTION/PROCEDURE/AGGREGATE/ROUTINE` to generate an error if no argument list is supplied and there are multiple matching objects.

Also improve the error message in such cases.

- Split the `pg_statistic_ext` catalog into two catalogs, and add the `pg_stats_ext` view of it.

This change supports hiding potentially-sensitive statistics data from unprivileged users.

- Remove `obsolete pg_constraint.consrc` column.

This column has been deprecated for a long time, because it did not update in response to other catalog changes (such as column renamings). The recommended way to get a text version of a check constraint's expression from `pg_constraint` is `pg_get_expr(conbin, conrelid)`. `pg_get_constraintdef()` is also a useful alternative.

- Remove `obsolete pg_attrdef.adsrc` column.

This column has been deprecated for a long time, because it did not update in response to other catalog changes (such as column renamings). The recommended way to get a text version of a default-value expression from `pg_attrdef` is `pg_get_expr(adbin, adrelid)`.

- Mark table columns of type `name` as having `"C"` collation by default.

The comparison operators for data type `name` can now use any collation, rather than always using `"C"` collation. To preserve the previous semantics of queries, columns of type `name` are now explicitly marked as having `"C"` collation. A side effect of this is that regular-expression operators on `name` columns will now use the `"C"` collation by default, not the database collation, to determine the behavior of locale-dependent regular expression patterns (such as `\w`). If you want non-C behavior for a regular expression on a `name` column, attach an explicit `COLLATE` clause. (For user-defined `name` columns, another possibility is to specify a different collation at table creation time; but that just moves the nonbackwards-compatibility to the comparison operators.)

- Treat `object-name` columns in the `information_schema` views as being of type `name`, not `varchar`.

Per the SQL standard, object-name columns in the `information_schema` views are declared as being of domain type `sql_identifier`. In PostgreSQL, the underlying catalog columns are really of type `name`. This change makes `sql_identifier` be a domain over `name`, rather than `varchar` as before. This eliminates a semantic mismatch in comparison and sorting behavior, which can greatly improve the performance of queries on `information_schema` views that restrict an `object-name` column. Note however that inequality restrictions, for example:

```
SELECT ... FROM information_schema.tables WHERE
table_name < 'foo';
```

will now use `"C"-locale` comparison semantics by default, rather than the database's default collation as before. Sorting on these columns will also follow `"C"` ordering rules. The previous behavior (and inefficiency) can be enforced by adding a `COLLATE "default"` clause.

- Remove the ability to disable dynamic shared memory.

Specifically, `dynamic_shared_memory_type` can no longer be set to `none`.

- Parse `libpq` integer connection parameters more strictly.

In previous releases, using an incorrect integer value for connection parameters `connect_timeout`, `keepalives`, `keepalives_count`, `keepalives_idle`, `keepalives_interval` and `port` resulted in `libpq` either ignoring those values or failing with incorrect error messages.

- The `SELECT DISTINCT...ORDER BY` clause of the `SELECT DISTINCT` query behavior differs after upgrade.

If `SELECT DISTINCT` is specified or if a `SELECT` statement includes the `SELECT DISTINCT ...ORDER BY` clause then all the expressions in `ORDER BY` must be present in the select list of the `SELECT DISTINCT` query (applicable when upgrading from version 9.6 to any higher version of Advanced Server).