



EDB

PG Failover Slots

Release 1.0.0

EnterpriseDB Corporation

Apr 06, 2023

CONTENTS

- 1 PG Failover Slots release notes 3**
 - 1.1 Release notes for PG Failover Slots version 1.0.0 3
- 2 Installing PG Failover Slots 5**
 - 2.1 Prerequisites 5
 - 2.2 Install the package 5
- 3 Configuring PG Failover Slots 7**
 - 3.1 synchronize_slot_names 7
 - 3.2 drop_extra_slots 8
 - 3.3 primary_dsn 8
 - 3.4 standby_slot_names 8
 - 3.5 standby_slots_min_confirmed 9
- 4 Using PG Failover Slots 11**

PG Failover Slots (`pg_failover_slots`) is an extension that makes logical replication slots practically usable across a physical failover.

This extension does the following:

- Copies any missing slots from the primary to the standby
- Removes any slots from the standby that aren't found on the primary
- Periodically synchronizes the position of slots on the standby based on the primary
- Ensures that selected standbys receive data before any of the logical slot walsenders can send data to consumers

PG FAILOVER SLOTS RELEASE NOTES

The PG Failover Slots documentation describes the latest version of PG Failover Slots, including minor releases and patches. These release notes cover what was new in each release. For new functionality introduced in a minor or patch release, there are also indicators in the content about the release that introduced the feature.

Version	Release Date
<i>Release notes for PG Failover Slots version 1.0.0</i>	2023 Mar 31

1.1 Release notes for PG Failover Slots version 1.0.0

This release of PG Failover Slots includes:

Type	Description
Feature	This is the initial release.

INSTALLING PG FAILOVER SLOTS

PG Failover Slots is supported on the same platforms as the Postgres distribution you're using. Support for PG Failover Slots starts with Postgres 12. For details, see:

- [EDB Postgres Advanced Server Product Compatibility](#)
- [PostgreSQL Product Compatibility](#)
- [EDB Postgres Extended Server Product Compatibility](#)

2.1 Prerequisites

Before you begin the installation process:

- Install Postgres. See: [- Installing EDB Postgres Advanced Server](#)
- [Installing PostgreSQL](#)
- Installing EDB Postgres Extended Server
- Set up the repository.

Setting up the repository is a one-time task. If you've already set up your repository, you don't need to perform this step.

To set up the repository, go to [EDB repositories](#) , and follow the instructions provided there.

2.2 Install the package

The syntax for the RPM package install command is:

```
sudo <package-manager> -y install pg-<postgres><postgres_version>-pg-failover_slots  
↪<major_version>
```

The syntax for the Debian package install command is:

```
sudo <package-manager> -y install pg-<postgres><postgres_version>-pg-failover_slots-  
↪<major_version>
```

Where:

- `<package-manager>` is the package manager used with your operating system:

Package manager	Operating system
dnf	RHEL 8 and derivatives
yum	RHEL 7 and derivatives, CentOS 7
zypper	SLES
apt-get	Debian and derivatives

- `<postgres>` is the distribution of Postgres you're using:

Postgres distribution	Value
PostgreSQL	pg
EDB Postgres Advanced Server	as
EDB Postgres Extended	postgresextended

- `<postgres_version>` is the version of Postgres you're using.
- `<major_version>` is the major version of PG Failover Slots you're installing.

For example, to install PG Failover Slots 1.0.0 for EDB Postgres Advanced Server 15 on a RHEL 8 platform:

```
sudo dnf -y install pg-as15-pg-failover_slots
```

To install PG Failover Slots 1.0.0 for EDB Postgres Advanced Server 15 on a Debian 11 platform:

```
sudo apt-get -y install pg-as15-pg-failover_slots-1
```

CONFIGURING PG FAILOVER SLOTS

You must add the extension to `shared_preload_libraries` on both the primary instance as well as any standby that's used for high availability (failover or switchover) purposes.

You can configure the behavior of PG Failover Slots using the following configuration options in `postgresql.conf`. The configuration options have the prefix `pg_failover_slots.`, for example, `pg_failover_slots.synchronize_slot_names`.

3.1 `synchronize_slot_names`

Option that allows you to set the logical slots to synchronize to this physical standby.

Select slots to synchronize by using a slot filter. These slot filters can match a specifically named slot, slots with names matching an SQL LIKE expression, or the name of a plugin that a slot is using. Define slot filters as a `key:value` pair, where the key can be one of:

- `name` — Match an exact slot name, for example, `name:my_slot`.
- `name_like` — Match any slot name that matches an SQL LIKE expressions, for example, `name_like:change_%`.
- `plugin` — Match any slot using a named plugin, for example, `plugin:test_decoding`.

Where you don't specify a key, the system defaults to interpreting it as a `name`. So `my_slot` is equivalent to `name:my_slot`.

You can combine slot filters in a comma-separated string value. Setting `my_slot,name_like:change_%,plugin:test_decoding` synchronizes the slot named `my_slot`, any slot whose name began with `change_`, and any slot that used the `test_decoding` plugin.

If `pg_failover_slots.synchronize_slot_names` is set to an empty string, no slots are synchronized.

The default value for `pg_failover_slots.synchronize_slot_names` is `name_like:%`, which matches all logical replication slots and synchronizes all slots.

3.2 drop_extra_slots

Controls what happens to extra slots on the standby that aren't found on the primary using the `pg_failover_slots.synchronize_slot_names` filter. When set to `true` (the default), they are dropped. Otherwise, they're retained.

3.3 primary_dsn

Specifies the connection string to use to connect to the primary when fetching slot information.

If empty (the default), then it uses the same connection string as `primary_conninfo`.

!!!Note You can't use `primary_conninfo` if there's a password field in the connection string. The password gets obfuscated by PostgreSQL, and PG Failover Slots can't actually see the password. In this case, you must configure `pg_failover_slots.primary_dsn`.

3.4 standby_slot_names

Ensures that the failover-candidate streaming physical replicas received and flushed all changes before they ever become visible to any subscribers. This option guarantees that a commit can't vanish on failover to a standby for the consumer of a logical slot.

Replication slots whose names are included in the comma-separated `pg_failover_slots.standby_slot_names` list are treated specially by the walsender on the primary.

Logical replication walsenders ensure that all local changes are sent and flushed to the replication slots in `pg_failover_slots.standby_slot_names` before the walsender sends those changes for the logical replication slots. Effectively, it provides a synchronous replication barrier between the named list of slots and all the consumers of logically decoded streams from walsender.

You can list any replication slot in `pg_failover_slots.standby_slot_names`. Both logical and physical slots work, but it's generally used for physical slots.

Without this safeguard, two anomalies are possible where a commit can be received by a subscriber and then vanish from the provider on failover because the failover candidate didn't receive it yet:

- For one or more subscribers, the subscriber might have applied the change, but the new provider might execute new transactions that conflict with the received change, as it never happened as far as the provider is concerned.
- For two or more subscribers, at the time of failover, not all subscribers applied the change. The subscribers now have inconsistent and irreconcilable states because the subscribers that didn't receive the commit have no way to get it now.

Setting `pg_failover_slots.standby_slot_names` causes subscribers to lag behind the provider if the provider's failover-candidate replicas aren't keeping up, by design. Monitoring is thus essential.

3.5 standby_slots_min_confirmed

Controls how many of the `pg_failover_slots.standby_slot_names` have to confirm before data is sent through the logical replication slots. Setting `-1` (the default) specifies to wait for all entries in `pg_failover_slots.standby_slot_names`.

USING PG FAILOVER SLOTS

During failover, the slots aren't synchronized to the standby immediately because of consistency reasons. The standby can be too behind logical slots or too ahead of logical slots on primary when the `pg_failover_slots` extension is activated. The extension does verification and synchronizes slots only when it's actually safe.

However, this brings a need to verify that the slots are synchronized and that the standby is actually ready to be a failover target with consistent logical decoding for all slots. You need to do this only initially. Once the slots are synchronized the first time, they're always consistent as long as the extension is active in the cluster.

The check for whether slots are fully synchronized with the primary is relatively simple. The slots just need to be present in the `pg_replication_slots` view on the standby and have active state `false`. An active state `true` means the slots is currently being initialized.

For example, consider the following psql session:

```
# SELECT slot_name, active FROM pg_replication_slots WHERE slot_type = logical;
```

__OUTPUT__

slot_name	active
regression_slot1	f
regression_slot2	f
regression_slot3	t

This example shows slots `regression_slot1` and `regression_slot2` are synchronized from the primary to the standby, and `regression_slot3` is still being synchronized. If failover happens at this stage, the `regression_slot3` is lost.

After waiting a little, you can query again:

```
```sql # SELECT slot_name, active FROM pg_replication_slots WHERE slot_type = logical;
```

**OUTPUT** slot\_name | active - +----- regression\_slot1 | f regression\_slot2 | f regression\_slot3 | f``` Now all three slots are synchronized. You can use the standby for failover without losing logical decoding state for any of them.