



EDB

EDB Postgres Distributed Command Line Interface

Release 1.0.0

EDB

Jun 04, 2022

CONTENTS

1	PGD CLI Release notes	3
1.1	Release notes for PGD CLI version 1	3
2	Requirements	5
2.1	Privileges	5
2.2	Supported version	5
3	Installing the EDB Postgres Distributed Command Line Interface	7
3.1	Installing with TPAexec	7
3.2	Installing manually	7
3.2.1	Specifying database connection strings	7
3.2.2	Using a configuration file	7
4	Using the EDB Postgres Distributed Command Line Interface	9
4.1	Specifying a configuration file	9
4.2	Passing a database connection string	9
5	pgd	11
5.1	Synopsis	11
5.2	Options	11
6	check-health	13
6.1	Synopsis	13
6.2	Examples	13
6.3	Options	14
6.4	Options inherited from parent commands	14
7	show-camo	15
7.1	Synopsis	15
7.2	Examples	15
7.3	Options	16
7.4	Options inherited from parent commands	16
8	show-clockskew	17
8.1	Synopsis	17
8.2	Examples	17
8.3	Options	18
8.4	Options inherited from parent commands	18
9	show-events	19
9.1	Synopsis	19

9.2	Examples	19
9.3	Options	20
9.4	Options inherited from parent commands	20
10	show-nodes	21
10.1	Synopsis	21
10.2	Examples	22
10.3	Options	22
10.4	Options inherited from parent commands	23
11	show-raft	25
11.1	Synopsis	25
11.2	Examples	25
11.3	Options	26
11.4	Options inherited from parent commands	26
12	show-replslots	27
12.1	Synopsis	27
12.2	Examples	27
12.3	Options	29
12.4	Options inherited from parent commands	30
13	show-subscriptions	31
13.1	Synopsis	31
13.2	Examples	31
13.3	Options	32
13.4	Options inherited from parent commands	33
14	show-version	35
14.1	Synopsis	35
14.2	Examples	35
14.3	Options	36
14.4	Options inherited from parent commands	36

The EDB Postgres Distributed Command Line Interface (PGD CLI) is a tool to manage your BDR cluster. It allows you to run commands against BDR clusters.

You can use it to inspect and manage cluster resources and to get information on the following:

- node summary
- BDR/Postgres version on each node
- replication and subscription details
- membership and worker events
- raft consensus details
- CAMO (Commit at Most Once) details
- clock skew
- cluster health summary

PGD CLI RELEASE NOTES

PGD CLI is a tool to manage your BDR cluster. It allows you to run commands against BDR clusters.

The PGD CLI documentation describes the latest version of PGD CLI including minor releases and patches. The release notes in this section provide information on what was new in each release.

Version	Release Date
<i>Release notes for PGD CLI version 1</i>	2022 May 17

1.1 Release notes for PGD CLI version 1

This is an initial release of EDB Postgres Distributed (PGD) Command Line Interface (CLI). PGD CLI version 1 includes:

Type	Description
Feature	Initial release of PGD CLI. Includes the following commands:
	<i>pgd check-health</i> - Checks the health of the BDR cluster.
	<i>pgd show-camo</i> - Shows BDR CAMO (Commit at Most Once) details.
	<i>pgd show-clockskew</i> - Shows the status of clock skew between each BDR node pair.
	<i>pgd show-events</i> - Shows events such as background worker errors and node membership changes.
	<i>pgd show-nodes</i> - Shows all nodes in the BDR cluster and their summary.
	<i>pgd show-raft</i> - Shows BDR Raft (consensus protocol) details.
	<i>pgd show-replslots</i> - Shows the status of BDR replication slots.
	<i>pgd show-subscriptions</i> - Shows BDR subscription (incoming replication) details.
	<i>pgd show-version</i> - Shows the version of BDR and Postgres installed on each node.

REQUIREMENTS

2.1 Privileges

The PGD CLI requires postgres superuser privileges to run.

2.2 Supported version

The PGD CLI supports BDR 4.1 and later.

INSTALLING THE EDB POSTGRES DISTRIBUTED COMMAND LINE INTERFACE

3.1 Installing with TPAexec

If you are using TPAexec (v22.14 or later) to deploy EDB Postgres Distributed, it installs the PGD CLI package (`edb-pgd-cli`) on each BDR node, by default. It also creates the configuration file (`/etc/edb/pgd-config.yml`) prepopulated with the endpoints of each node in the cluster so no additional configuration is necessary.

If you wish to install the PGD CLI on any non-BDR instance in the cluster, you simply attach the `pgdcli` role to that instance in TPAexec's configuration file before deploying. See *Installing with TPAexec* for more information.

3.2 Installing manually

You can manually install the PGD CLI on any Linux machine using `.deb` and `.rpm` packages available from the [BDR repository](#). The package name is `edb-pgd-cli`. For example:

```
# for Debian
sudo apt-get install edb-pgd-cli
```

When the PGD CLI is configured by TPAexec, it connects automatically, but with a manual installation you need to provide a connection string to the BDR cluster.

3.2.1 Specifying database connection strings

You can either use a configuration file to specify the database connection strings for your cluster (see following section) or pass the connection string directly to a command (see the sample use case).

3.2.2 Using a configuration file

Use the `pgd-config.yml` configuration file to specify the database connection string for your cluster. The configuration file should contain the database connection string for at least one BDR node in the cluster. The cluster name is optional and not validated.

For example:

```
cluster:
  name: cluster-name
  endpoints:
```

(continues on next page)

(continued from previous page)

```
- "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "  
- "host=bdr-b1 port=5432 dbname=bdrdb user=postgres "  
- "host=bdr-c1 port=5432 dbname=bdrdb user=postgres "
```

The `pgd-config.yml` , is located in the `/etc/edb` directory, by default. The PGD CLI searches for `pgd-config.yml` in the following locations (precedence order - higher to lower):

1. `/etc/edb` (default)
2. `$HOME/.edb`
3. `.` (working directory)

If you rename the file or move it to another location, specify the new name and location using the optional `-f` or `--config-file` flag when entering a command. See the sample use case .

USING THE EDB POSTGRES DISTRIBUTED COMMAND LINE INTERFACE

pgd is the command name for the PGD command line interface.

The syntax for the command is:

```
pgd -f, --config-file string    configuration file (default
                                "/etc/edb/pgd-config.yml")
    --dsn string                database connection string
    -h, --help                  help for pgd
    -L, --log-level string      logging level: debug, info,
                                warn, error (default "error")
    -o, --output string         output format: json, yaml
```

See sample use cases for the following flags:

- -f, --config-file
- -dsn
- -o, --output
- -h, --help

4.1 Specifying a configuration file

If you rename the file or move it to another location, specify the new name and location using the optional -f or --config-file flag. For example:

```
pgd show-nodes -f /opt/my-config.yml
```

4.2 Passing a database connection string

Use the --dsn flag to pass a database connection string directly to a command. You don't need a configuration file if you pass the connection string with this flag. The flag takes precedence if a configuration file is present. For example:

```
pgd show-nodes --dsn "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
``` ## Specifying the output format
```

The PGD CLI supports the following output formats:

(continues on next page)

(continued from previous page)

```
.. csv-table::
:header: Format,Considerations
:widths: 10,30
:class: longtable

tabular,Default format. Presents the data in tabular form.
json,"Presents the raw data with no formatting. For some commands, the json output may
↳ show more data than shown in the tabular output such as extra fields and more detailed
↳ messages."
yaml,Same as json except field order is alphabetical. Experimental and may not be fully
↳ supported in future versions.

Use the `-o` or `--output` flag to change the default output format to json or yaml.
↳ For example:

```sh
pgd show-nodes -o json
``` ## Accessing the command line help

To list the supported commands, enter:

```sh
pgd help
```

For help for a specific command and its parameters, enter `pgd help <command_name>` . For example:

```
pgd help show-nodes
```

pgd is the command name for the PGD command line interface.

5.1 Synopsis

The EDB Postgres Distributed Command Line Interface (PGD CLI) is a tool to manage your BDR cluster. It allows you to run commands against BDR clusters. You can use it to inspect and manage cluster resources.

5.2 Options

<code>-f, --config-file string</code>	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
<code>--dsn string</code>	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
<code>-h, --help</code>	help for pgd
<code>-L, --log-level string</code>	logging level: debug, info, warn, error (default "error")
<code>-o, --output string</code>	output format: json, yaml

CHECK-HEALTH

Checks the health of the BDR cluster.

6.1 Synopsis

Performs various checks such as if all nodes are accessible, all replication slots are working, and CAMO pairs are connected.

```
pgd check-health [flags]
```

6.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down; bdr-a1 and bdr-b1 are CAMO partners)

```
$ pgd check-health
```

Check	Status	Message
----	-----	-----
CAMO	Critical	At least 1 CAMO partner is not connected
ClockSkew	Critical	Clockskew cannot be determined for at least 1 BDR node pair
Connection	Critical	The node bdr-b1 is not accessible
Raft	Warning	There is no RAFT_LEADER, an election might be in progress
Replslots	Critical	There is at least 1 BDR replication slot which is inactive
Version	Warning	There is at least 1 node that is not accessible

Example 2 (3 node cluster, all nodes are up but system clocks are not in sync)

```
$ pgd check-health
```

Check	Status	Message
----	-----	-----
CAMO	Ok	All CAMO pairs are connected
ClockSkew	Warning	At least 1 BDR node pair has clockskew greater than 2 seconds
Connection	Ok	All BDR nodes are accessible
Raft	Ok	Raft Consensus is working correctly
Replslots	Ok	All BDR replication slots are working correctly

(continues on next page)

(continued from previous page)

```
Version    Ok      All nodes are running same BDR versions
```

Example 3 (3 node cluster, all nodes are up and all checks are Ok)

```
$ pgd check-health
```

Check	Status	Message
-----	-----	-----
CAMO	Ok	All CAMO pairs are connected
ClockSkew	Ok	All BDR node pairs have clockskew within permissible limit
Connection	Ok	All BDR nodes are accessible
Raft	Ok	Raft Consensus is working correctly
Replslots	Ok	All BDR replication slots are working correctly
Version	Ok	All nodes are running same BDR versions

6.3 Options

```
-h, --help    help for check-health
```

6.4 Options inherited from parent commands

-f, --config-file string	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
--dsn string	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
-L, --log-level string	logging level: debug, info, warn, error (default "error")
-o, --output string	output format: json, yaml

SHOW-CAMO

Shows BDR CAMO (Commit at Most Once) details.

7.1 Synopsis

Shows BDR CAMO (Commit at Most Once) details such as the name of the CAMO partner, connection and readiness status, any pending and unresolved CAMO transactions, and differences between `apply_lsn` and `receive_lsn`. This command is available only for EDB Postgres Extended and EDB Postgres Advanced Server (v14 and later).

```
pgd show-camo [flags]
```

7.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-b1 are CAMO partner but bdr-b1 is down)

```
$ pgd show-camo
```

Node	CAMO Partner	Connected	Ready	Transactions Resolved	Apply LSN	Receive LSN	Apply Queue Size
bdr-a1 bdr-b1	false	false	true		0/E42C99B0	0/E42C99B0	0

Example 2 (3 node cluster, bdr-b1 was down and it has just been restarted)

```
$ pgd show-camo
```

Node	CAMO Partner	Connected	Ready	Transactions Resolved	Apply LSN	Receive LSN	Apply Queue Size
bdr-a1 bdr-b1	true	true	true		0/E533DAB8	0/E533DAB8	0
bdr-b1 bdr-a1	true	false	true		3/7AE81A28	3/7AE81A28	0

Example 3 (3 node cluster, all nodes are up and in streaming state)

```
$ pgd show-camo
```

(continues on next page)

(continued from previous page)

Node	CAMO	Partner	Connected	Ready	Transactions	Resolved	Apply LSN	Receive LSN	Apply
↪ Queue Size									

↪ -----									
bdr-a1	bdr-b1		true	true	true		0/E56AE520	0/E56AE520	0
bdr-b1	bdr-a1		true	true	true		3/7B180BA8	3/7B180BA8	0

7.3 Options

-h, --help help for show-camo

7.4 Options inherited from parent commands

-f, --config-file string	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
--dsn string	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
-L, --log-level string	logging level: debug, info, warn, error (default "error")
-o, --output string	output format: json, yaml

SHOW-CLOCKSKEW

Shows the status of clock skew between each BDR node pair.

8.1 Synopsis

Shows the status of clock skew between each BDR node pair in the cluster.

Symbol Meaning ——— ——— * ok ~ warning (skew > 2 seconds) ! critical (skew > 5 seconds) x down / unreachable
? unknown - n/a

```
pgd show-clockskew [flags]
```

8.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-clockskew
```

Node	bdr-a1	bdr-b1	bdr-c1	Current Time
----	-----	-----	-----	-----
bdr-a1	*	?	*	2022-03-30 07:02:21.334472
bdr-b1	x	*	x	x
bdr-c1	*	?	*	2022-03-30 07:02:21.186809

Example 2 (3 node cluster, all nodes are up)

```
$ pgd show-clockskew
```

Node	bdr-a1	bdr-b1	bdr-c1	Current Time
----	-----	-----	-----	-----
bdr-a1	*	*	*	2022-03-30 07:04:54.147017
bdr-b1	*	*	*	2022-03-30 07:04:54.340543
bdr-c1	*	*	*	2022-03-30 07:04:53.90451

8.3 Options

<code>-h, --help</code> help for show-clockskew

8.4 Options inherited from parent commands

<code>-f, --config-file string</code>	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
<code>--dsn string</code>	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
<code>-L, --log-level string</code>	logging level: debug, info, warn, error (default "error")
<code>-o, --output string</code>	output format: json, yaml

SHOW-EVENTS

Shows events such as background worker errors and node membership changes.

9.1 Synopsis

Shows events such as background worker errors and node membership changes. Output is sorted by Time column in descending order. Message column is truncated after a few lines. To view complete message use json output format ('-o json').

For more details on each node state, see show-nodes command help ('pgd show-nodes -h').

```
pgd show-events [flags]
```

9.2 Examples

Example 1 (3 node cluster)

```
$ pgd show-events --lines 10
```

Time	Observer Node	Subject Node	Type	Message
----	-----	-----	----	-----
2022-04-19 19:45:43.077712+00	bdr-b1	bdr-c1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:43.066804+00	bdr-c1	bdr-a1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:43.057598+00	bdr-b1	bdr-a1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:43.046515+00	bdr-c1	bdr-b1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:43.033369+00	bdr-a1	bdr-c1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:43.013203+00	bdr-a1	bdr-b1	receiver worker error	pglogical↵
↵				worker received fast finish request, exiting
2022-04-19 19:45:40.024662+00	bdr-c1	bdr-c1	node state change	ACTIVE
2022-04-19 19:45:40.024575+00	bdr-b1	bdr-c1	node state change	ACTIVE
2022-04-19 19:45:40.022788+00	bdr-a1	bdr-c1	node state change	ACTIVE
2022-04-19 19:45:38.961424+00	bdr-c1	bdr-c1	node state change	PROMOTING

9.3 Options

<code>-h, --help</code>	help for show-events
<code>-n, --lines int</code>	show top n lines

9.4 Options inherited from parent commands

<code>-f, --config-file string</code>	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
<code>--dsn string</code>	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
<code>-L, --log-level string</code>	logging level: debug, info, warn, error (default "error")
<code>-o, --output string</code>	output format: json, yaml

SHOW-NODES

Shows all nodes in the BDR cluster and their summary.

10.1 Synopsis

Shows all nodes in the BDR cluster and their summary such as name, node id, group, and current/target state etc.

Node States

- **NONE:** Node state is unset when the worker starts, expected to be set quickly to the current known state.
- **CREATED:** `bdr.create_node()` has been executed, but the node is not a member of any BDR cluster yet.
- **JOIN_START:** `bdr.join_node_group()` begins to join the local node to an existing BDR cluster.
- **JOINING:** The node join has started and is currently at the initial sync phase, creating the schema and data on the node.
- **CATCHUP:** Initial sync phase is completed; now the join is at the last step of retrieving and applying transactions that were performed on the upstream peer node since the join started.
- **STANDBY:** Node join has finished, but not yet started to broadcast changes. All joins spend some time in this state, but if defined as a Logical Standby, the node will continue in this state.
- **PROMOTE:** Node was a logical standby and we just called `bdr.promote_node` to move the node state to **ACTIVE**. These two **PROMOTE** states have to be coherent to the fact, that only one node can be with a state higher than **STANDBY** but lower than **ACTIVE**.
- **PROMOTING:** Promotion from logical standby to full BDR node is in progress.
- **ACTIVE:** The node is a full BDR node and is currently **ACTIVE**. This is the most common node status.
- **PART_START:** Node was **ACTIVE** or **STANDBY** and we just called `bdr.part_node` to remove the node from the BDR cluster.
- **PARTING:** Node disconnects from other nodes and plays no further part in consensus or replication.
- **PART_CATCHUP:** Non-parting nodes synchronize any missing data from the recently parted node.
- **PARTED:** Node parting operation is now complete on all nodes.

Only one node at a time can be in either of the states **PROMOTE** or **PROMOTING**.

Note that the read-only state of a node, as shown in the Current State or in the Target State columns, is indicated as **STANDBY**.

```
pgd show-nodes [flags]
```

10.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-nodes
```

Node	Node ID	Node Group	Current State	Target State	Status	Seq ID
bdr-a1	3136956818	bdrgroup	ACTIVE	ACTIVE	Up	1
bdr-b1	2380210996	bdrgroup	ACTIVE	ACTIVE	Unreachable	2
bdr-c1	1804769977	bdrgroup	ACTIVE	ACTIVE	Up	3

Example 2 (3 node cluster, all nodes are up)

```
$ pgd show-nodes
```

Node	Node ID	Node Group	Current State	Target State	Status	Seq ID
bdr-a1	3136956818	bdrgroup	ACTIVE	ACTIVE	Up	1
bdr-b1	2380210996	bdrgroup	ACTIVE	ACTIVE	Up	2
bdr-c1	1804769977	bdrgroup	ACTIVE	ACTIVE	Up	3

Example 3 (cluster with witness, logical standby and subscriber-only nodes)

Note: In contrast to logical standby, the subscriber-only nodes are fully joined node to the cluster

```
$ pgd show-nodes
```

Node	Node ID	Node Group	Current State	Target State	Status	Seq ID
bdr-a1	3136956818	bdrgroup	ACTIVE	ACTIVE	Up	1
bdr-b1	2380210996	bdrgroup	ACTIVE	ACTIVE	Up	2
witness-c1	1450365472	bdrgroup	ACTIVE	ACTIVE	Up	3
logical-standby-a1	1140256918	bdrgroup	STANDBY	STANDBY	Up	4
logical-standby-b1	3541792022	bdrgroup	STANDBY	STANDBY	Up	5
subscriber-only-c1	2448841809	subscriber-only	ACTIVE	ACTIVE	Up	6

10.3 Options

```
-h, --help  help for show-nodes
```

10.4 Options inherited from parent commands

<code>-f, --config-file string</code>	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
<code>--dsn string</code>	database connection string e.g. "host=bdr-al port=5432 dbname=bdrdb user=postgres "
<code>-L, --log-level string</code>	logging level: debug, info, warn, error (default "error")
<code>-o, --output string</code>	output format: json, yaml

SHOW-RAFT

Shows BDR Raft (consensus protocol) details.

11.1 Synopsis

Shows BDR Raft (consensus protocol) details such as Raft state (leader, follower), Raft election id, and number of voting nodes.

Note: In some cases such as network partition, output may vary based on the node to which the CLI is connected.

```
pgd show-raft [flags]
```

11.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-raft
```

Node	Raft State	Raft Term	Commit Index	Nodes	Voting Nodes	Protocol Version
bdr-c1	RAFT_FOLLOWER	29	6081272	3	3	4002
bdr-a1	RAFT_LEADER	29	6081272	3	3	4002
bdr-b1						

Example 2 (3 node cluster, all nodes are up)

```
$ pgd show-raft
```

Node	Raft State	Raft Term	Commit Index	Nodes	Voting Nodes	Protocol Version
bdr-c1	RAFT_FOLLOWER	38	6132327	3	3	4002
bdr-a1	RAFT_LEADER	38	6132331	3	3	4002
bdr-b1	RAFT_FOLLOWER	38	6132336	3	3	4002

Example 3 (3 node cluster, all nodes are up but bdr-a1 is not able to connect to other nodes; following is the output when cli is connected to bdr-a1)

```
$ pgd show-raft
```

(continues on next page)

(continued from previous page)

```

Node      Raft State      Raft Term Commit Index Nodes Voting Nodes Protocol Version
-----
bdr-c1
bdr-a1 RAFT_FOLLOWER 40          6176769      3      3          4002
bdr-b1

```

Example 4 (cluster with witness, logical standby and subscriber-only nodes)
 Note: Unlike full-bdr (or witness node), logical standby and subscriber-only nodes don't have raft voting rights.

```
$ pgd show-raft
```

```

Node      Raft State      Raft Term Commit Index Nodes Voting Nodes Protocol Version
-----
bdr-a1     RAFT_LEADER    0           10268        6      3          4003
bdr-b1     RAFT_FOLLOWER  0           10279        6      3          4003
witness-c1 RAFT_FOLLOWER  0           10281        6      3          4003
logical-standby-a1 RAFT_FOLLOWER  0           10281        6      3          4003
logical-standby-b1 RAFT_FOLLOWER  0           10281        6      3          4003
subscriber-only-c1 RAFT_FOLLOWER  0           10281        6      3          4003

```

11.3 Options

```
-h, --help    help for show-raft
```

11.4 Options inherited from parent commands

```

-f, --config-file string  config file; ignored if
                           --dsn flag is present (default "/etc/edb/pgd-config.yml")
--dsn string              database connection string
                           e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
-L, --log-level string    logging level: debug, info, warn, error (default "error")
-o, --output string       output format: json, yaml

```

SHOW-REPLSLOTS

Shows the status of BDR replication slots.

12.1 Synopsis

Shows the status of BDR replication slots. Output with the verbose flag gives details such as is slot active, replication state (disconnected, streaming, catchup), and approximate lag.

Symbol Meaning ——— * ok ~ warning (lag > 10M) ! critical (lag > 100M OR slot is ‘inactive’ OR ‘disconnected’) x down / unreachable - n/a

In matrix view, sometimes byte lag is shown in parentheses. It is a maxOf(WriteLag, FlushLag, ReplayLag, SentLag).

```
pgd show-replslots [flags]
```

12.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-replslots
```

```
Node    bdr-a1 bdr-b1 bdr-c1
-----
bdr-a1 *      ! (6.6G) *
bdr-b1 x      *      x
bdr-c1 *      ! (6.9G) *
```

```
$ pgd show-replslots --verbose
```

Origin Node	Target Node	Status (active/state)	Write Lag (bytes/duration)	Replay Lag (bytes/duration)	Sent Lag (bytes)	Flush Lag
bdr-a1	bdr-b1	f / disconnected	6.6G / 8 days 02:58:36.243723	6.6G / 8 days 02:58:36.243723	6.6G	6.6G / 8
bdr-a1	bdr-c1	t / streaming	0B / 00:00:00	0B / 00:00:00	0B	0B /
bdr-c1	bdr-a1	t / streaming	0B / 00:00:00.000812	0B / 00:00:00.000812	0B	0B /

(continues on next page)

(continued from previous page)

```
bdr-c1      bdr-b1      f / disconnected      6.9G / 8 days 02:58:36.004415 6.9G / 8
↳ days 02:58:36.004415 6.9G / 8 days 02:58:36.004415 6.9G
```

Example 2 (3 node cluster, bdr-b1 was down and it has just been restarted)

```
$ pgd show-replslots
```

```
Node      bdr-a1 bdr-b1 bdr-c1
-----
bdr-a1 *          ! (6.9G) *
bdr-b1 *          *      *
bdr-c1 *          ! (5.8G) *
```

```
$ pgd show-replslots --verbose
```

```
Origin Node Target Node Status (active/state) Write Lag (bytes/duration) Flush Lag
↳ (bytes/duration) Replay Lag (bytes/duration) Sent Lag (bytes)
-----
↳ -----
bdr-a1      bdr-b1      t / catchup      6.9G / 00:00:00.000778      6.9G / 00:00:00.
↳ 000778      6.9G / 00:00:00.000778      6.9G
bdr-a1      bdr-c1      t / streaming      0B / 00:00:00.104121      0B / 00:00:00.
↳ 104133      0B / 00:00:00.104133      0B
bdr-b1      bdr-a1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳      0B / 00:00:00      0B
bdr-b1      bdr-c1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳      0B / 00:00:00      0B
bdr-c1      bdr-a1      t / streaming      6.8K / 00:00:00      6.8K / 00:00:00
↳      6.8K / 00:00:00      6.8K
bdr-c1      bdr-b1      t / catchup      5.5G / 00:00:00.008257      5.5G / 00:00:00.
↳ 008257      5.5G / 00:00:00.008257      5.5G
```

Example 3 (3 node cluster, all nodes are up and in streaming state)

```
$ pgd show-replslots
```

```
Node      bdr-a1 bdr-b1 bdr-c1
-----
bdr-a1 *          *      *
bdr-b1 *          *      *
bdr-c1 *          *      *
```

```
$ pgd show-replslots --verbose
```

```
Origin Node Target Node Status (active/state) Write Lag (bytes/duration) Flush Lag
↳ (bytes/duration) Replay Lag (bytes/duration) Sent Lag (bytes)
-----
↳ -----
bdr-a1      bdr-b1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳      0B / 00:00:00      0B
bdr-a1      bdr-c1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳      0B / 00:00:00      0B
```

(continues on next page)

(continued from previous page)

```

bdr-b1      bdr-a1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳          0B / 00:00:00      0B
bdr-b1      bdr-c1      t / streaming      0B / 00:00:00      0B / 00:00:00
↳          0B / 00:00:00      0B
bdr-c1      bdr-a1      t / streaming      0B / 00:00:00      528B / 00:00:00
↳          528B / 00:00:00      0B
bdr-c1      bdr-b1      t / streaming      528B / 00:00:00      528B / 00:00:00
↳          528B / 00:00:00      0B

```

Example 4 (cluster with witness, logical standby and subscriber-only nodes;
upstream for logical-standby-a1 is bdr-a1 and for logical-standby-b1 it is bdr-b1)

Note:

1. A logical standby is sent data only by one source node, but no other nodes receive replication changes from it
2. Subscriber-only node subscribes to replication changes from other nodes in the cluster, but no other nodes receive replication changes from it

\$ pgd show-replslots

```

Node      bdr-a1 bdr-b1 logical-standby-a1 logical-standby-b1 subscriber-only-
↳ c1 witness-c1
-----
↳ -----
bdr-a1      *      *      *      -      *
↳ *
bdr-b1      *      *      -      *      *
↳ *
logical-standby-a1 -      -      *      -      -
↳ -
logical-standby-b1 -      -      -      *      -
↳ -
subscriber-only-c1 -      -      -      -      *
↳ -
witness-c1   *      *      -      -      *
↳ *

```

12.3 Options

```

-h, --help      help for show-replslots
-v, --verbose    verbose output (default true)

```

12.4 Options inherited from parent commands

-f, --config-file string	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
--dsn string	database connection string e.g. "host=bdr-al port=5432 dbname=bdrdb user=postgres "
-L, --log-level string	logging level: debug, info, warn, error (default "error")
-o, --output string	output format: json, yaml

SHOW-SUBSCRIPTIONS

Shows BDR subscription (incoming replication) details.

13.1 Synopsis

Shows BDR subscription (incoming replication) details such as origin/target node, timestamp of the last replayed transaction, and lag between now and the timestamp of the last replayed transaction.

```
pgd show-subscriptions [flags]
```

13.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-subscriptions
```

Origin Node	Target Node	Last Transaction Replayed At	Lag Duration (seconds)
bdr-a1	bdr-c1	2022-04-23 13:13:40.854433+00	0.514275
bdr-b1	bdr-a1		
bdr-b1	bdr-c1		
bdr-c1	bdr-a1	2022-04-23 13:13:40.852233+00	0.335464

Example 2 (3 node cluster, bdr-b1 was down and it has just been restarted)

```
$ pgd show-subscriptions
```

Origin Node	Target Node	Last Transaction Replayed At	Lag Duration (seconds)
bdr-a1	bdr-b1	2022-04-23 13:14:45.669254+00	0.001686
bdr-a1	bdr-c1	2022-04-23 13:14:46.157913+00	-0.002009
bdr-b1	bdr-a1		
bdr-b1	bdr-c1		
bdr-c1	bdr-a1	2022-04-23 13:14:45.698472+00	0.259521
bdr-c1	bdr-b1	2022-04-23 13:14:45.667979+00	0.002961

Example 3 (3 node cluster, all nodes are up and in streaming state)

(continues on next page)

(continued from previous page)

```
$ pgd show-subscriptions
```

Origin Node	Target Node	Last Transaction Replayed At	Lag Duration (seconds)
bdr-a1	bdr-b1	2022-04-23 13:15:39.732375+00	0.034462
bdr-a1	bdr-c1	2022-04-23 13:15:40.179618+00	0.002647
bdr-b1	bdr-a1	2022-04-23 13:15:39.719994+00	0.305814
bdr-b1	bdr-c1	2022-04-23 13:15:40.180886+00	0.001379
bdr-c1	bdr-a1	2022-04-23 13:15:39.714397+00	0.311411
bdr-c1	bdr-b1	2022-04-23 13:15:39.714397+00	0.052440

Example 4 (cluster with witness, logical standby and subscriber-only nodes; upstream for logical-standby-a1 is bdr-a1 and for logical-standby-b1 it is bdr-b1)
 Note: Logical standby and subscriber-only nodes receive changes but do not send changes made locally to other nodes

```
$ pgd show-subscriptions
```

Origin Node	Target Node	Last Transaction Replayed At	Lag Duration (seconds)
bdr-a1	bdr-b1	2022-04-23 13:40:49.106411+00	0.853665
bdr-a1	logical-standby-a1	2022-04-23 13:40:50.72036+00	0.138430
bdr-a1	logical-standby-b1		
bdr-a1	subscriber-only-c1	2022-04-23 13:40:50.72036+00	0.016226
bdr-a1	witness-c1	2022-04-23 13:40:50.470142+00	0.001514
bdr-b1	bdr-a1	2022-04-23 13:40:49.10174+00	1.095422
bdr-b1	logical-standby-a1		
bdr-b1	logical-standby-b1	2022-04-23 13:40:50.713666+00	0.271213
bdr-b1	subscriber-only-c1	2022-04-23 13:40:50.713666+00	0.022920
bdr-b1	witness-c1	2022-04-23 13:40:50.471789+00	-0.000133
witness-c1	bdr-a1	2022-04-23 13:40:49.107706+00	1.089456
witness-c1	bdr-b1	2022-04-23 13:40:49.107706+00	0.852370
witness-c1	logical-standby-a1		
witness-c1	logical-standby-b1		
witness-c1	subscriber-only-c1	2022-04-23 13:40:50.719844+00	0.016742

13.3 Options

```
-h, --help  help for show-subscriptions
```

13.4 Options inherited from parent commands

-f, --config-file string	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
--dsn string	database connection string e.g. "host=bdr-al port=5432 dbname=bdrdb user=postgres "
-L, --log-level string	logging level: debug, info, warn, error (default "error")
-o, --output string	output format: json, yaml

SHOW-VERSION

Shows the version of BDR and Postgres installed on each node.

14.1 Synopsis

Shows the version of BDR and Postgres installed on each node in the cluster.

```
pgd show-version [flags]
```

14.2 Examples

Example 1 (3 node cluster, bdr-a1 and bdr-c1 are up, bdr-b1 is down)

```
$ pgd show-version
```

Node	BDR Version	Postgres Version
bdr-c1	4.1.0	14.2 (EDB Postgres Extended Server 14.2.0) (Debian 2:14.2.0edbpge-1. ↳buster+1)
bdr-a1	4.1.0	14.2 (EDB Postgres Extended Server 14.2.0) (Debian 2:14.2.0edbpge-1. ↳buster+1)
bdr-b1		

Example 2 (3 node cluster, all nodes are up)

```
$ pgd show-version
```

Node	BDR Version	Postgres Version
bdr-c1	4.1.0	14.2 (EDB Postgres Extended Server 14.2.0) (Debian 2:14.2.0edbpge-1. ↳buster+1)
bdr-a1	4.1.0	14.2 (EDB Postgres Extended Server 14.2.0) (Debian 2:14.2.0edbpge-1. ↳buster+1)
bdr-b1	4.1.0	14.2 (EDB Postgres Extended Server 14.2.0) (Debian 2:14.2.0edbpge-1. ↳buster+1)

14.3 Options

<code>-h, --help</code> help for show-version

14.4 Options inherited from parent commands

<code>-f, --config-file string</code>	config file; ignored if --dsn flag is present (default "/etc/edb/pgd-config.yml")
<code>--dsn string</code>	database connection string e.g. "host=bdr-a1 port=5432 dbname=bdrdb user=postgres "
<code>-L, --log-level string</code>	logging level: debug, info, warn, error (default "error")
<code>-o, --output string</code>	output format: json, yaml