createuser

- Description
- Options
- Environment
- Diagnostics
- Examples
- See Also

createuser

createuser

1

Application

createuser

define a new

PostgreSQL

user account

createuser

connection-option

option

username

Description

createuser creates a new PostgreSQL user (or more precisely, a role). Only superusers and users with `CREATEROLE` privilege can create new users, so createuser must be invoked by someone who can connect as a superuser or a user with `CREATEROLE` privilege.

If you wish to create a role with the `SUPERUSER`, `REPLICATION`, or `BYPASSRLS` privilege, you must connect as a superuser, not merely with `CREATEROLE` privilege. Being a superuser implies the ability to bypass all access permission checks within the database, so superuser access should not be granted lightly. `CREATEROLE` also conveys very extensive privileges.

createuser is a wrapper around the SQL command `CREATE ROLE`. There is no effective difference between creating users via this utility and via other methods for accessing the server.

Options

createuser accepts the following command-line arguments:

<username>
Specifies the name of the PostgreSQL user to be created. This name must be different from all existing roles in this PostgreSQL installation.

`-a role`; `--with-admin=role`
Specifies an existing role that will be automatically added as a member of the new role with admin option, giving it the right to grant membership in the new role to others. Multiple existing roles can be specified by writing multiple `-a` switches.

`-c number`; `--connection-limit=number`
Set a maximum number of connections for the new user. The default is to set no limit.

`-d`; `--createdb`
The new user will be allowed to create databases.

`-D`; `--no-createdb`
The new user will not be allowed to create databases. This is the default.

`-e`; `--echo`
Echo the commands that createuser generates and sends to the server.

`-E`; `--encrypted`
This option is obsolete but still accepted for backward compatibility.

`-g role`; `--member-of=role`; `--role=role` (deprecated)
Specifies the new role should be automatically added as a member of the specified existing role. Multiple existing roles can be specified by writing multiple `-g` switches.

`-i`; `--inherit`
The new role will automatically inherit privileges of roles it is a member of. This is the default.

`-I`; `--no-inherit`
The new role will not automatically inherit privileges of roles it is a member of.

`--interactive`
Prompt for the user name if none is specified on the command line, and also prompt for whichever of the options `-d`/`-D`, `-r`/`-R`, `-s`/`-S` is not specified on the command line. (This was the default behavior up to PostgreSQL 9.1.)

`-l`; `--login`
The new user will be allowed to log in (that is, the user name can be used as the initial session user identifier). This is the default.

`-L`; `--no-login`
The new user will not be allowed to log in. (A role without login privilege is still useful as a means of managing database permissions.)

`-m role`; `--with-member=role`
Specifies an existing role that will be automatically added as a member of the new role. Multiple existing roles can be specified by writing multiple `-m` switches.

`-P`; `--pwprompt`
If given, createuser will issue a prompt for the password of the new user. This is not necessary if you do not plan on using password authentication.

`-r`; `--createrole`
The new user will be allowed to create, alter, drop, comment on, change the security label for other roles; that is, this user will have `CREATEROLE` privilege. See ??? for more details about what capabilities are conferred by this privilege.

`-R`; `--no-createrole`
The new user will not be allowed to create new roles. This is the default.

`-s`; `--superuser`
The new user will be a superuser.

`-S`; `--no-superuser`
The new user will not be a superuser. This is the default.

`-v timestamp`; `--valid-until=timestamp`
Set a date and time after which the role's password is no longer valid. The default is to set no password expiry date.

`-V`; `--version`
Print the createuser version and exit.

`--bypassrls`
The new user will bypass every row-level security (RLS) policy.

`--no-bypassrls`
The new user will not bypass row-level security (RLS) policies. This is the default.

`--replication`
The new user will have the `REPLICATION` privilege, which is described more fully in the documentation for ???.

`--no-replication`
The new user will not have the `REPLICATION` privilege, which is described more fully in the documentation for ???. This is the default.

`-?`; `--help`
Show help about createuser command line arguments, and exit.

createuser also accepts the following command-line arguments for connection parameters:

`-h host`; `--host=host`
Specifies the host name of the machine on which the server is running. If the value begins with a slash, it is used as the directory for the Unix domain socket.

`-p port`; `--port=port`
Specifies the TCP port or local Unix domain socket file extension on which the server is listening for connections.

`-U username`; `--username=username`
User name to connect as (not the user name to create).

`-w`; `--no-password`
Never issue a password prompt. If the server requires password authentication and a password is not available by other means such as a `.pgpass` file, the connection attempt will fail. This option can be useful in batch jobs and scripts where no user is present to enter a password.

`-W`; `--password`
Force createuser to prompt for a password (for connecting to the server, not for the password of the new user).

This option is never essential, since createuser will automatically prompt for a password if the server demands password authentication. However, createuser will waste a connection attempt finding out that the server wants a password. In some cases it is worth typing `-W` to avoid the extra connection attempt.

Environment

`PGHOST`; `PGPORT`; `PGUSER`
Default connection parameters

`PG_COLOR`
Specifies whether to use color in diagnostic messages. Possible values are `always`, `auto` and `never`
.

This utility, like most other PostgreSQL utilities, also uses the environment variables supported by libpq (see ???).

Diagnostics

In case of difficulty, see ??? and ??? for discussions of potential problems and error messages. The database server must be running at the targeted host. Also, any default connection settings and environment variables used by the libpq front-end library will apply.

Examples

To create a user `joe` on the default database server:

```
$ createuser joe
```

To create a user `joe` on the default database server with prompting for some additional attributes:

```
$ createuser --interactive joe
Shall the new role be a superuser? (y/n) n
Shall the new role be allowed to create databases? (y/n) n
Shall the new role be allowed to create more new roles? (y/n) n
```

To create the same user `joe` using the server on host `eden`, port 5000, with attributes explicitly specified, taking a look at the underlying command:

```
$ createuser -h eden -p 5000 -S -D -R -e joe
CREATE ROLE joe NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT LOGIN;
```

To create the user `joe` as a superuser, and assign a password immediately:

```
$ createuser -P -s -e joe
Enter password for new role: xyzzy
Enter it again: xyzzy
CREATE ROLE joe PASSWORD 'SCRAM-SHA-
256$4096:44560wPMLfjqiAzyPDZ/eQ==$4CA054rZlSFEq8Z3FEhToBTa2X6KnWFxFkPwIbKoDe0=:L/nbSZRCjp6RhOh
KK56GoR1zibCCSePKshCJ9lnl3yw=' SUPERUSER CREATEDB CREATEROLE INHERIT LOGIN NOREPLICATION
NOBYPASSRLS;
```

In the above example, the new password isn't actually echoed when typed, but we show what was typed for clarity. As you see, the password is encrypted before it is sent to the client.

See Also

□□□ #2
□ □□□ □□□ 2026-10-31 14:10:18 UTC
□ □□□ □□□ 2026-10-31 14:10:19 UTC