

**POSTGRESQL SETUP GUIDE**

**The Below requirements are needed to establish a connection postgresql.**

**Prerequisites**

Prerequisiteslink  
To connect your generic PostgreSQL database to Fivetran, you need:  
  
PostgreSQL version 8.4 - 13.x  
Your database host's IP (e.g., 1.2.3.4) or domain (your.server.com)  
Your database's port (usually 5432)  
TLS enabled on your database

**Setup Guide**

1. Choose incremental sync mechanism

* To keep your data up to date after the initial sync, we use one of the following incremental sync methods:
* logical replication with the test\_decoding plugin
* logical replication with the pgoutput plugin Beta
* XMIN
* Each of these methods keeps a record of recent data changes, which allows Fivetran to update only the data that has changed since our last sync.
* TIP: We recommend using logical replication with the test\_decoding plugin as your incremental update mechanism. Learn more in our Updating data documentation.

1. Logical replication

* IMPORTANT: You can only enable logical replication if your PostgreSQL server version is 9.4.15 or later. You can only enable logical replication with the pgoutput plugin if your PostgresSQL server version is 10+. Prior minor versions have breaking bugs.
* Logical replication is based on logical decoding of the PostgreSQL write-ahead log (WAL). Fivetran reads the WAL to detect any new or changed data. We recommend using logical replication with the test\_decoding plugin because it is faster than XMIN replication and allows Fivetran to detect deleted rows for tables with primary keys.
* Learn more in our logical replication documentation.

1. XMIN

* The XMIN method is based on the hidden xmin system column that is present in all PostgreSQL tables. With XMIN, Fivetran must scan every table in full to detect updated data. We do not recommend XMIN for near real-time data needs because XMIN replication is slower than logical replication and doesn't allow Fivetran to detect deleted rows.
* Learn more in our XMIN documentation.

1. Choose connection method (TLS required)

* IMPORTANT: You must have TLS enabled on your database to connect to Fivetran.
* Decide whether to connect your generic PostgreSQL database directly or using an SSH tunnel.

1. Connect directly

* Fivetran connects directly to your database instance. This is the simplest and most secure method.
* To connect directly, do the following to configure your firewall and/or other access control systems:
* Allow incoming connections to your PostgreSQL port (usually 5432) from Fivetran's IPs for your database's region.
* Allow outgoing connections from all ports (1024 to 65535) to Fivetran's IPs for your database's region.
* How you do this will vary based on how your PostgreSQL database is hosted (cloud platform, on-premises, etc.).

1. Connect using SSH

* Fivetran connects to a separate server in your network that provides an SSH tunnel to your database. You must connect through SSH if your database is in an inaccessible subnet.
* To connect using SSH, follow our SSH connection instructions to give Fivetran access to your SSH tunnel.

1. Create user

* Create a database user for Fivetran's exclusive use.
* Open a connection to your PostgreSQL database in a PostgreSQL console (such as a SQL workbench or psql).
* Create a user for Fivetran by executing the following SQL command. Replace <username> and some-password with a username and password of your choice.
* CREATE USER <username> PASSWORD 'some-password';
* content\_copy

1. Grant read-only access

* Grant the Fivetran user read-only access to all tables by running the following commands. To grant access to a schema other than PostgreSQL's default public schema, replace public with the schema name.
* GRANT USAGE ON SCHEMA "public" TO <username>;
* GRANT SELECT ON ALL TABLES IN SCHEMA "public" TO <username>;
* ALTER DEFAULT PRIVILEGES IN SCHEMA "public" GRANT SELECT ON TABLES TO <username>;
* content\_copy
* NOTE: The last command makes sure that any future tables will be accessible to Fivetran.
* If you want to grant access to multiple schemas, you must run these three commands for each schema.

1. Restrict access to tables (optional)

* If you want to limit Fivetran's access to your tables, grant the Fivetran user access to only the tables that you would like to sync. You must individually grant access for each table that you want to sync. You cannot grant access to all tables and then revoke access for a subset of tables.
* Ensure that the Fivetran user has access to the schema that contains your table(s).
* GRANT USAGE ON SCHEMA "some\_schema" TO <username>;
* content\_copy
* Revoke any previously granted permission to all tables in that schema.
* ALTER DEFAULT PRIVILEGES IN SCHEMA "some\_schema" REVOKE SELECT ON TABLES FROM <username>;
* REVOKE SELECT ON ALL TABLES IN SCHEMA "some\_schema" FROM <username>;
* content\_copy
* Repeat the following command for each table you want Fivetran to sync.
* GRANT SELECT ON "some\_schema"."some\_table" TO <username>;
* content\_copy
* By default, any tables that you create in the future will be excluded from the Fivetran user's access. To grant access to new tables, run the following command.
* ALTER DEFAULT PRIVILEGES IN SCHEMA "some\_schema" GRANT SELECT ON TABLES TO <username>;
* content\_copy

1. Restrict access to columns (optional)

* If you want to limit Fivetran's access to the columns in your tables, grant the Fivetran user access to only certain columns. You must individually grant access for each column that you want to sync.
* Revoke any previously granted permission to read all columns in the table.
* REVOKE SELECT ON "some\_schema"."some\_table" FROM <username>;
* content\_copy
* Grant permission to the specific columns you want to sync (for example, some\_column and other\_column).
* NOTE: If you chose XMIN as your incremental update mechanism, you must grant us access to the hidden system column xmin.
* GRANT SELECT (xmin, "some\_column", "other\_column") ON "some\_schema"."some\_table" TO <username>;
* content\_copy
* Once you restrict access to columns within a table, the Fivetran user will not have access to any new columns added to that table in the future. To grant access to new columns, you must rerun the command above.

1. Configure incremental update mechanism

* Configure your chosen incremental update mechanism.

1. Logical replication with the test\_decoding plugin

* To enable logical replication with the test\_decoding plugin, follow these steps:
* Go to your PostgreSQL database.
* Ensure that your server has ample free space for the logs. As soon as Fivetran processes a log, we delete it. However, we don't delete logs if the sync is interrupted (for example, if we lose access to your database). In this case, logs may accumulate on your server and consume additional storage. The amount of additional disk space that these logs consume is proportional to the number of changes committed on the server. If we can't resume a lost connection quickly enough and you need more disk space, you can drop the replication slot, which deletes its unconsumed logs.
* Ensure that the statement\_timeout setting on your server is either 0 (the default value to disable the timeout) or greater than 5 minute.
* Set the wal\_level parameter in your database configuration to logical. For a standard PostgreSQL database, do this by adding a wal\_level = logical line to the postgresql.conf1 file. Restart the server for this change to take effect.
* Ensure that your max\_replication\_slots value is equal to or higher than the number of PostgreSQL connectors that use WAL plus the number of other replication slots your database uses.
* Set the wal\_sender\_timeout parameter in your database configuration to 0 to disable the timeout.
* Add a record to your pg\_hba.conf file that allows your database to authenticate Fivetran's connection to the WAL.
* Go to the postgresql.conf file and ensure that the max\_wal\_senders parameter, which specifies the maximum number of concurrent connections to the WAL, is at least twice the total number of logical replication slots. For example, if your database uses 11 replication slots in total, then the max\_wal\_senders value must be 22 or greater.
* Log in to a PostgreSQL console (such as a SQL workbench or psql) as a superuser.
* Create a logical replication slot for the database you want to sync by running the following command. You must use the output plugin test\_decoding supplied in the postgresql-contrib subpackage.
* IMPORTANT: The replication slot name fivetran\_replication\_slot quoted throughout this guide is used purely as an example. The actual replication slot name should be unique for every connector using the same PostgreSQL cluster. Replication slot names cannot start with a number.
* SELECT pg\_create\_logical\_replication\_slot('fivetran\_replication\_slot', 'test\_decoding');
* content\_copy
* Grant permission to the Fivetran user for reading the replication slot.
* ALTER ROLE <username> WITH REPLICATION;
* content\_copy
* Log in as the Fivetran user.
* Verify that the Fivetran user can read the replication slot by running the following command:
* SELECT count(\*) FROM pg\_logical\_slot\_peek\_changes('fivetran\_replication\_slot', null, null);
* content\_copy
* If the query succeeds, then permissions are sufficient.
* IMPORTANT: You must periodically tune the checkpoint\_timeout and max\_wal\_size parameters based on your PostgreSQL database operations. If you do not, you may experience replication failures. To learn how to tune, read this tuning checkpoints documentation.

1. Logical replication with the pgoutput plugin Beta

* To enable logical replication with the pgoutput plugin, follow these steps:
* Go to your PostgreSQL database.
* Ensure that your server has ample free space for the logs. As soon as Fivetran processes a log, we delete it. However, we don't delete logs if the sync is interrupted (for example, if we lose access to your database). In this case, logs may accumulate on your server and consume additional storage. The amount of additional disk space that these logs consume is proportional to the number of changes committed on the server. If we can't resume a lost connection quickly enough and you need more disk space, you can drop the replication slot, which deletes its unconsumed logs.
* Ensure that the statement\_timeout setting on your server is either 0 (the default value to disable the timeout) or greater than 5 minute.
* Set the wal\_level parameter in your database configuration to logical. For a standard PostgreSQL database, do this by adding a wal\_level = logical line to the postgresql.conf1 file. Restart the server for this change to take effect.
* Ensure that your max\_replication\_slots value is equal to or higher than the number of PostgreSQL connectors that use WAL plus the number of other replication slots your database uses.
* Set the wal\_sender\_timeout parameter in your database configuration to 0 to disable the timeout.
* Add a record to your pg\_hba.conf file that allows your database to authenticate Fivetran's connection to the WAL.
* Go to the postgresql.conf file and ensure that the max\_wal\_senders parameter, which specifies the maximum number of concurrent connections to the WAL, is at least twice the total number of logical replication slots. For example, if your database uses 11 replication slots in total, then the max\_wal\_senders value must be 22 or greater.
* Log in to a PostgreSQL console (such as a SQL workbench or psql) as a superuser.
* Create a logical replication slot for the database you want to sync by running the following command. You must use the standard output plugin pgoutput.
* IMPORTANT: The replication slot name fivetran\_pgoutput\_slot quoted throughout this guide is used purely as an example. The actual replication slot name should be unique for every connector using the same PostgreSQL cluster. Replication slot names cannot start with a number.
* SELECT pg\_create\_logical\_replication\_slot('fivetran\_pgoutput\_slot', 'pgoutput');
* content\_copy
* Create a publication for your tables. If you want, you can create a publication for only certain tables so that you add or remove tables from the publication later on. Only changes from tables in the publication are replicated to Fivetran. Each database can have multiple distinct publications. You must have CREATE privileges or above to run this command.
* IMPORTANT: The publication name fivetran\_pub quoted throughout this guide is used purely as an example. The actual publication name should be unique for every database and cannot start with a number.
* CREATE PUBLICATION fivetran\_pub FOR TABLE table2, table4, table8;
* content\_copy
* To add or remove a table from a publication, run the following command. You must have ownership rights over the table(s).
* ALTER PUBLICATION fivetran\_pub ADD/DROP TABLE table\_name;
* content\_copy
* Alternatively, you can create a publication for all of your tables. However, you cannot remove any table from this publication later on. You must have superuser privileges to run this command.
* CREATE PUBLICATION fivetran\_pub FOR ALL TABLES;
* content\_copy
* (Optional) You can choose which operations to include in the publication. For example, the following publication includes only INSERT and UPDATE operations.
* CREATE PUBLICATION insert\_only\_pub FOR TABLE table1 WITH (publish = 'INSERT, UPDATE');
* content\_copy
* Verify that your chosen tables are in the publication.
* SELECT \* FROM pg\_publication\_tables.
* content\_copy
* Grant the Fivetran user permission to read the replication slot.
* ALTER ROLE <username> WITH REPLICATION;
* content\_copy
* Log in as the Fivetran user.
* Verify that the Fivetran user can read the replication slot by running the following command. Replace fivetran\_pgoutput\_slot with your replication slot name and fivetran\_pub with the publication name.
* SELECT count(\*) FROM pg\_logical\_slot\_peek\_binary\_changes('fivetran\_pgoutput\_slot', null, null, 'proto\_version', '1', 'publication\_names', 'fivetran\_pub');
* content\_copy
* If the query succeeds, then permissions are sufficient.
* IMPORTANT: You must periodically tune the checkpoint\_timeout and max\_wal\_size parameters based on your PostgreSQL database operations. If you do not, you may experience replication failures. To learn how to tune, read this tuning checkpoints documentation.

1. XMIN

* The XMIN method is based on the hidden xmin system column that is present in all PostgreSQL tables. To enable XMIN, ensure that the statement\_timeout setting on your server is either 0 (the default value to disable the timeout) or greater than 5 minute.

1. Finish Fivetran configuration

* In your connector setup form, enter a destination schema prefix. This prefix applies to each replicated schema and cannot be changed once your connector is created.
* In the Host field, enter your database host's IP (for example, 1.2.3.4) or domain (for example, your.server.com).
* Enter your database instance's port number. The port number is usually 5432.
* Enter the Fivetran-specific user that you created in Step 3.
* Enter the password for the Fivetran-specific user that you created in Step 3.
* Enter the name of your database (for example, your\_database).
* Choose your connection method. If you selected Connect via an SSH tunnel, provide the following information:
* SSH hostname (do not use a load balancer's IP address/hostname)
* SSH port
* SSH user
* Choose your update method. If you selected Logical replication of the WAL using the test\_decoding plugin, enter the name of your database's replication slot. If you selected Logical replication of the WAL using the pgoutput plugin, enter both the name of your database's replication slot and publication name accordingly.
* Click Save & Test. Fivetran tests and validates our connection to your PostgreSQL database. Upon successful completion of the setup tests, you can sync your data using Fivetran.

1. Setup tests

* Fivetran performs the following tests to ensure that we can connect to your generic PostgreSQL database and that it is properly configured:
* The Connecting to SSH Tunnel Test validates the SSH tunnel details you provided in the setup form. It then checks that we can connect to your database using the SSH Tunnel. (We skip this test if you aren't connecting using SSH.)
* The Connecting to Host Test validates the database credentials you provided in the setup form. The test verifies that the host is not private and then checks the connectivity to the host.
* The Validating Certificate Test generates a pop-up window where you must choose which certificate you want Fivetran to use. It then validates that certificate and checks that we can connect to your database using TLS. (We skip this test if you aren't connecting directly.)
* The Connecting to Database Test checks that we can access your database.
* The Connecting to WAL Replication Slot Test confirms that the database associated with the replication slot matches the name you supplied in the setup form. It then verifies that the replication slot uses the pgoutput if you selected WAL with the pgoutput update method, or test\_decoding plugin if you selected WAL with the test\_decoding update method. Lastly, it makes sure that the Fivetran user has replication privileges. (We skip this test if you selected XMIN as your incremental update mechanism)
* The Publication Test verifies that the supplied publication name exists in your database. (We skip this test if you selected XMIN or WAL with the test\_decoding plugin as your incremental update mechanism.)
* The Checking wal\_sender\_timeout Value Test checks that your database's wal\_sender\_timeout value is 0. (We skip this test if you selected XMIN as your incremental update mechanism.)
* The Checking statement\_timeout Value Test checks that we can access your database's pg\_settings table. It then verifies that the statement\_timeout value is either 0 or greater than 5 minutes.
* NOTE: The tests may take a few minutes to finish running.

