

**POSTGRESQL AZURE SETUP GUIDE**

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

**Prerequisites**

Prerequisiteslink  
To connect your 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 (e.g., your.server.com)  
Your database's port (usually 5432)  
TLS enabled on your database

**Setup Guide**

* IMPORTANT: You must have TLS enabled on your database to connect to Fivetran.
* Decide whether to connect your Azure PostgreSQL database directly or using an SSH tunnel. How you configure your security groups will differ depending on this decision.

1. Connect directly

* Fivetran connects directly to your database instance. This is the simplest and most secure method.
* If you connect directly, you must create a rule in a security group that allows Fivetran access to your database instance.

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.
* If you connect using SSH, you must follow our SSH tunnel instructions before proceeding to the next step. You must then configure your tunnel server's security group to allow Fivetran access and configure the database instance's security to allow access from the tunnel.

1. Enable access

* You must configure the firewall to grant Fivetran's data processing servers access to your database server.
* In the Azure console, open the SQL database firewall settings.
* Select the Azure PostgreSQL database that you want to connect to Fivetran.
* In Settings, click Connection security.
* Add a new firewall rule. Enter Fivetran's IP in both the Start IP and End IP fields to define the firewall rule's range.
* Click Save.

1. Create user

* Open a connection to your master PostgreSQL database.
* 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 data, grant the Fivetran user access to only the tables that you would like to sync. You need to individually grant access for each table that you want to sync. It is not possible to achieve exclusion by granting access to all tables and then revoking 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)

* You can also grant the Fivetran user access to only certain columns within a table. You need to individually grant access for each column that you want to sync.
* NOTE: We need access to the hidden system column xmin for incremental updates.
* Ensure that you have revoked 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).
* 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. 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. We support logical replication with the test\_decoding plugin and the XMIN method on Azure PostgreSQL versions 9.5 or later, and logical replication with the pgoutput plugin for 10.6 or later. For earlier versions, we only support the XMIN method. As a result, our Azure PostgreSQL connector does not support replicating deleted data for versions before 9.5.
* TIP: We recommend using logical replication with the test\_decoding plugin as your incremental update mechanism. Learn more in our Updating data documentation.
* Choose logical replication with the test\_decoding plugin, logical replication with the pgoutput plugin, or XMIN as your incremental update mechanism.
* To enable logical replication with the test\_decoding plugin, proceed to the test\_decoding section.
* To enable logical replication with the pgoutput plugin, proceed to the pgoutput section.
* To enable XMIN, proceed to the XMIN section.

1. Logical replication with the test\_decoding plugin

* IMPORTANT: You can only enable logical replication with the test\_decoding plugin if your Azure PostgreSQL version is 10 or later.
* To enable logical replication with the test\_decoding plugin, follow these steps:
* Connect to your master database. You cannot enable logical replication on a read replica.
* Ensure that your server has ample free space for the logs. Logs that Fivetran has already processed are released. However, logs are not released if replication stops (for example, if we lose access). In this case, logs may accumulate on your server and consume additional storage. The amount of additional disk space consumed by these logs is proportional to the amount of changes committed on the server. If a lost connection can't be resumed quickly enough, you can drop the replication slot, which releases the storage of unconsumed logs. You would then need to do a full re-sync of your connector to reset the cursor in the replication slot.
* In your Azure portal, do the following:
* i. Set Azure replication support to logical, then click Save.
* ii. Click Yes to restart the server to apply the change.
* iii. If you are running Postgres 9.5 or 9.6 and use public network access, add a firewall rule to include the public IP address of the client used to run the logical replication. The firewall rule name must include \_replrule. For example, fivetran\_replrule. Add the rule and click Save.
* Log into a PostgreSQL console as a superuser (one that has the rds\_superuser role).
* 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 fivetran WITH REPLICATION;
* content\_copy
* Log in as the Fivetran user.
* Verify that the Fivtran 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.

1. Logical replication with the pgoutput plugin Beta

* IMPORTANT: You can only enable logical replication with the pgoutput plugin if your Azure PostgreSQL version is 11 or later.
* To enable logical replication with the pgoutput plugin, follow these steps:
* Connect to your master database. You cannot enable logical replication on a read replica.
* Ensure that your server has ample free space for the logs. Logs that Fivetran has already processed are released. However, logs are not released if replication stops (for example, if we lose access). In this case, logs may accumulate on your server and consume additional storage. The amount of additional disk space consumed by these logs is proportional to the amount of changes committed on the server. If a lost connection can't be resumed quickly enough, you can drop the replication slot, which releases the storage of unconsumed logs. You would then need to do a full re-sync of your connector to reset the cursor in the replication slot.
* In your Azure portal, do the following:
* i. Set Azure replication support to logical, then click Save.
* ii. Click Yes to restart the server to apply the change.
* iii. If you are running Postgres 9.5 or 9.6 and use public network access, add a firewall rule to include the public IP address of the client used to run the logical replication. The firewall rule name must include \_replrule. For example, fivetran\_replrule. Add the rule and click Save.
* Log into a PostgreSQL console as a superuser (one that has the rds\_superuser role).
* 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 fivetran 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.

1. XMIN

* You do not need to do any additional configuration for the XMIN method. You must use the XMIN method if your Azure PostgreSQL version is earlier than 10.
* Enter user, password, and database in setup formlink
* In your Fivetran setup form, enter your user, password, and database name.
* For the User, enter <username>@<servername>, where <servername> is part of your Azure host URL: <servername>.database.windows.net.
* For the Password, enter the password you set when you created the Fivetran user.
* For the Database, enter the database you want to replicate from.
* 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, which is in alpha phase as of now, 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. Choose schema prefix

* This is the last step of the integration. Each schema from the source database will be mapped to a schema in the destination by adding a prefix to the original schema name. For example, if your original database contains schemas "foo" and "bar" and if you choose the prefix "pre", then you will get schemas "pre\_foo" and "pre\_bar" in the output.

1. Setup tests

* Fivetran performs the following tests to ensure that we can connect to your Azure 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 pgoutput update method, or the test\_decoding plugin if you selected WAL with 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 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.

