

**POSTGRESQL HEROKU SETUP GUIDE**

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

**Prerequisites**

Heroku requires you to have a heroku account.

**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.
* Choose logical replication with the test\_decoding plugin, logical replication with the pgoutput plugin, or XMIN as your incremental update mechanism. You will configure your incremental update mechanism in later steps.

1. Logical replication

* IMPORTANT: Because of AWS limitations, you can only use logical replication if you connect Fivetran to your Amazon RDS PostgreSQL master instance. You can only enable logical replication with the test\_decoding plugin if your PostgreSQL server version is 9.4 - 13.2. You can only enable logical replication with the pgoutput plugin if your PostgreSQL version is 10.4 or later. 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

* NOTE: You can use XMIN for any Amazon RDS PostgreSQL deployment, including a read replica.
* 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 database to Fivetran directly or using an SSH tunnel. How you configure your security groups in later steps will differ depending on this decision.

1. Connect directly

* Fivetran connects directly to your Amazon RDS PostgreSQL database. This is the simplest and most secure method.
* If you connect directly, you will 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 will configure your SSH tunnel server's security group to allow Fivetran access and configure your database's security to allow access from the tunnel.
* Before you proceed to the next step, you must follow our SSH connection instructions to give Fivetran access to your SSH tunnel.

1. Create read replica (XMIN only)

* IMPORTANT: You can only use a read replica if you chose XMIN as your incremental update mechanism.
* If you'd like, create a read replica for Fivetran's exclusive use. Using a read replica allows Fivetran to integrate your data without putting unnecessary load on or interrupting the queries running on your master server. We recommend that you connect a read replica to Fivetran, but it's not required.
* If you chose logical replication as your incremental update mechanism, want to connect Fivetran to your master database, or already have a read replica, skip ahead to Step 4.
* In your Amazon RDS Dashboard, select the PostgreSQL instance you want to replicate.
* Click Actions, then select Create read replica from the drop-down menu.
* In the Instance specifications section, specify the instance type for the read replica. It does not need to be as large as your master instance.
* In the Network Security section, set the Publicly accessibility setting to Yes to ensure that the read replica is accessible from outside your VPC.
* In the Settings section, enter your chosen instance ID.
* Click Create read replica.
* The replica's status should now be creating.
* It will take a few minutes for the read replica to finish being created. The status will change to available when it is done.

1. Enable database access

* Grant Fivetran's data processing servers access to your database server. How you grant access depends on whether or not your database instance is in a VPC.
* If your instance is in a VPC, you must configure the two mechanisms that control access: VPC security groups and network access control lists (ACLs). If your instance is not in a VPC, you only need to configure security groups.

1. Configure security group

* NOTE: These instructions assume that your database instance is in a VPC. If your database instance is not in a VPC, you can still use these instructions because configuring a non-VPC security group is an almost identical process.
* In your Amazon RDS dashboard, click on the database instance you want to connect to Fivetran.
* In the Connectivity & security section, find the database's port number and make a note of it. You will need the port number to configure Fivetran.

1. In the Security column, click the to the database instance's security group.

* In the Security Group panel, click Actions, then select Edit inbound rules from the drop-down menu.
* Click Add Rule. This creates a new Custom TCP Rule at the bottom of the list.
* Fill in the new Custom TCP Rule.
* In the Port Range field, enter your database instance's port number. If you created a new read replica for Fivetran, this is the port number that you found in step 3 of this section. (The port will be 5432 for direct connections, unless you changed the default.)
* What you enter in the Custom IP field depends on whether you're connecting directly or using an SSH tunnel.
* If you're connecting directly, enter Fivetran's IPs for your database's region.
* If you're connecting using an SSH tunnel, enter {your-ssh-tunnel-server-ip-address}/32.
* (Optional) Enter a brief description in the Description field.
* Click Save rules.

1. Configure network ACLs (VPC only)

* If your database instance is not in a VPC, skip ahead to Step 5.
* Return to the RDS dashboard.
* Click on your database instance.

1. Click the to the instance's VPC.

* Select the VPC ID.

1. In the Summary tab, click the Network ACL .

* You will see tabs for Inbound Rules and Outbound Rules. You must edit both.
* Edit inbound rules
* Select Inbound Rules.
* If you have a default VPC that was automatically created by AWS, the settings already allow all incoming traffic. To verify that the settings allow incoming traffic, confirm that the Source value is 0.0.0.0/0 and that the ALLOW entry is listed above the DENY entry.
* If your inbound rules don't include ALL - 0.0.0.0/0 - ALLOW entry, edit the rules to allow the Source to access the port number of your database instance. (The port will be 5432 for direct connections, unless you changed the default.) For additional help, see Amazon's Network ACLs documentation.
* If you're connecting directly, enter Fivetran's IPs for your database's region.
* If you're connecting using an SSH tunnel, enter {your-ssh-tunnel-server-ip-address}/32.
* Edit outbound rules
* Select Outbound Rules.
* If you have a default VPC that was automatically created by AWS, the settings already allow all outbound traffic. To verify that the settings allow outbound traffic, confirm that the Destination value is 0.0.0.0/0 and that the ALLOW entry is listed above the DENY entry.
* If your outbound rules don't include an ALL - 0.0.0.0/0 - ALLOW entry, edit the rules to allow outbound traffic to all ports 1024-65535 for the following Destinations:
* If you're connecting directly, enter Fivetran's IPs for your database's region.
* If you're connecting using an SSH tunnel, enter {your-ssh-tunnel-server-ip-address}/32.

1. Create user

* Create a database user for Fivetran's exclusive use.
* Open a connection to your Amazon RDS 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 user 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 permissions 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.
* Ensure that you have revoked any previously granted permission to read all columns in the table.
* REVOKE SELECT ON "some\_schema"."some\_table" FROM fivetran;
* 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 fivetran;
* 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 updates

* 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 Amazon RDS PostgreSQL master database. You cannot enable logical replication on a read replica.
* 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.
* Set the rds.logical\_replication parameter to 1 by following these steps:
* i. Create a new parameter group (non-default group).
* ii. Enable the logical\_replication flag in that group by setting the value to 1.
* iii. Set wal\_sender\_timeout to 0 to disable the timeout mechanism.
* NOTE: Disabling the timeout mechanism ensures that the database won't end the wal\_sender process before Fivetran is able to establish a connection. This helps us stay up to date with the latest changes in the database’s WAL slot.
* iv. Apply the parameter group to the database.
* v. Wait until the status changes to pending-reboot, then reboot the database to apply the new parameter group.
* Log in to a PostgreSQL console (such as a SQL workbench or psql) as a superuser. Superusers have 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.
* GRANT rds\_replication TO <username>;
* content\_copy
* NOTE: The Fivetran user does not need the rds\_superuser role.
* 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 Amazon RDS PostgreSQL master database. You cannot enable logical replication on a read replica.
* 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.
* Set the rds.logical\_replication parameter to 1 by following these steps:
* i. Create a new parameter group (non-default group).
* ii. Enable the logical\_replication flag in that group by setting the value to 1.
* iii. Set wal\_sender\_timeout to 0.
* iv. Apply the parameter group to the database.
* v. Wait until the status changes to pending-reboot, then reboot the database to apply the new parameter group.
* Log in to a PostgreSQL console (such as a SQL workbench or psql) as a superuser. Superusers have 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 <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

* You do not need to do any additional configuration for the XMIN method.

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-database.cp0rdhwjbsae.us-east-1.rds.amazonaws.com)
* Enter your database instance's port number. The port will be 5432, unless you changed the default.
* Enter the Fivetran-specific user that you created in Step 5.
* Enter the password for the Fivetran-specific user that you created in Step 5.
* 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 host (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 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 Amazon RDS 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 PostgreSQL RDS 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 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.

