PowerSync DevRel Technical Guide

In the age of real-time applications, seamless data synchronization across different environments is crucial. This guide provides a comprehensive walkthrough for integrating PowerSync (a product of JourneyApps) with MongoDB Atlas to manage real-time synchronization efficiently. This process ensures that your application remains up-to-date with the latest data dynamically, regardless of what backend you are using and where it is hosted—be it a self-hosted custom backend or using JourneyApps Platform serverless cloud functions (a sibling product of PowerSync). In the event of a connection loss, once the connection is restored, all data will automatically sync to ensure consistency and keep everything updated.

PowerSync & MongoDB Atlas

PowerSync, combined with MongoDB Atlas, provides a robust solution for syncing data in real-time. Unlike Atlas Device Sync, PowerSync provides the developer the option of sending local mutations to their own backend where validations can be applied before writing the changes into MongoDB Atlas. This guide outlines two configuration paths for this backend:

1. Cloud Hosting Backend Service Using JourneyApps Platform

2. Self-Hosting of a Custom Backend Service

For both scenarios, this guide will take you through the required configurations, utilizing a demo to-do list application provided by PowerSync. We conclude with insights into the diagnostics tool offered by PowerSync.

Data Model Overview

The demo to-do list application utilizes a straightforward data model with two collections: "lists" and "todos.

Collections

- Lists Collection
  - Document ID
  - Creation Time of the List
  - Name of the List
  - Owner ID (who created the list)

```
_id: "36cc0e12-d869-456b-9dd3-805f5ae7c576"
created_at : 2024-12-17T09:59:27.000+00:00
name : "fruits basket"
owner_id : "73f3672d-f57a-4838-adbc-3f7e80cabbb5"
```

- Todos Collection
  - Document ID
  - Task Completion Status
  - Time the Task was Created
  - Owner ID (who created the task)
  - Task Name
  - List ID (identifies which list it belongs to)

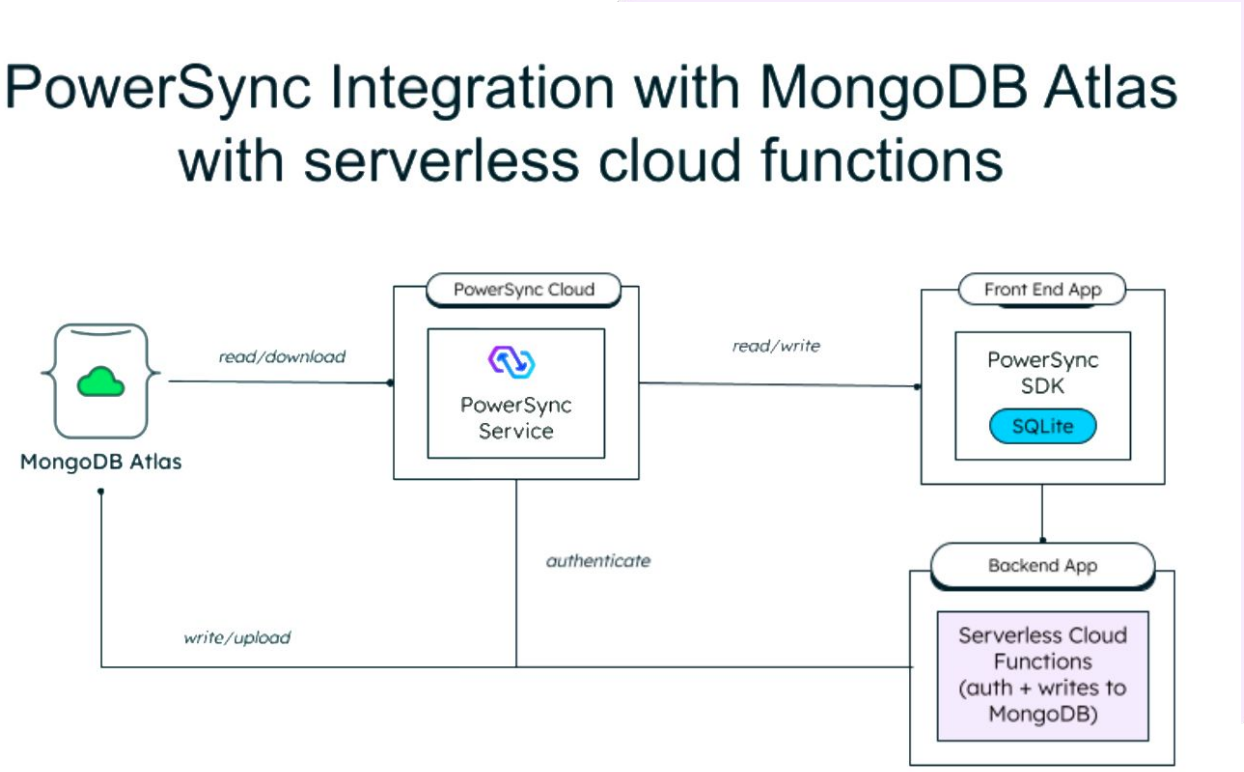
```
_id: "9427ea1c-6f04-407f-a310-0bbf805919a1"
completed : false
created_at : 2024-12-19T09:02:06.000+00:00
created_by : "05f76e09-1f15-4dd9-83a8-166bb09278a7"
description : "task1"
list_id : "36cc0e12-d869-456b-9dd3-805f5ae7c576"
```

This model offers a clear structure to manage tasks and lists, making it easy to track and organize your to-dos.

MongoDB.

Architecture Overview

JourneyApps Platform Hosted Backend Architecture



MongoDB Atlas: A fully managed cloud database service that stores and handles your application's core data.

PowerSync Service: A dedicated synchronization layer that manages real-time data updates and replication, which can be configured and monitored through the PowerSync Dashboard.

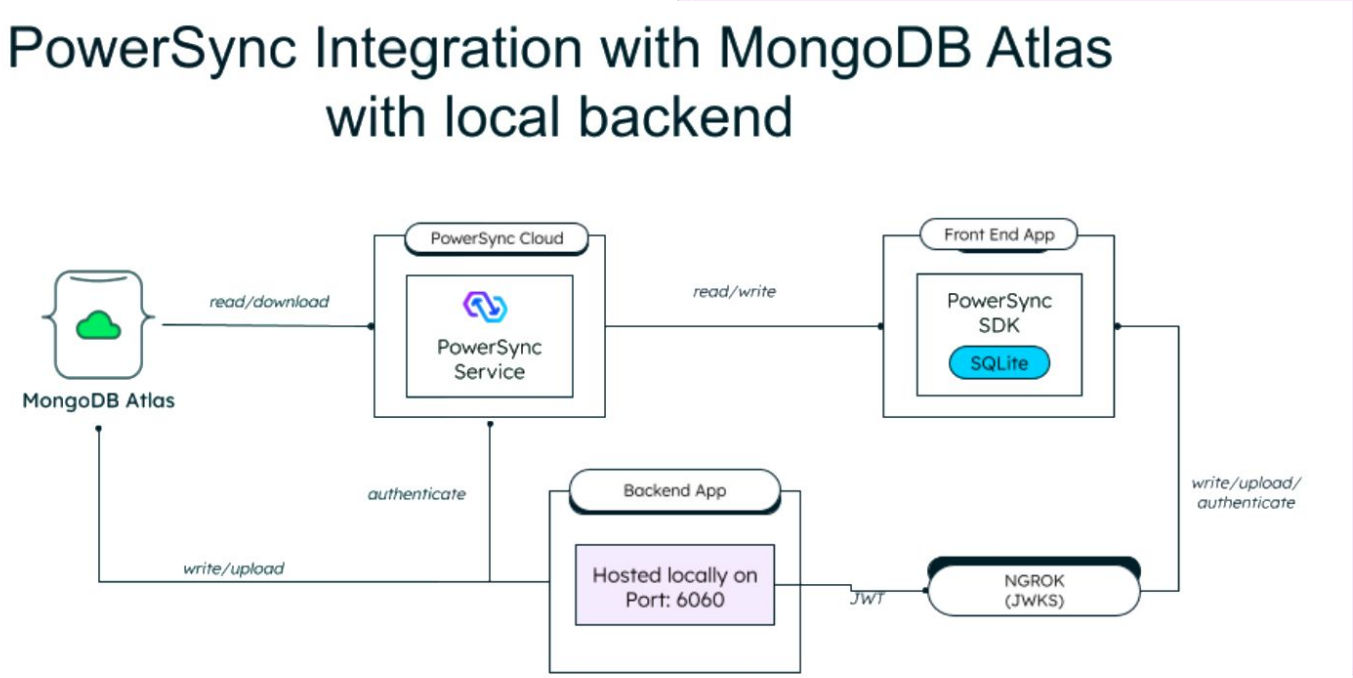
Backend App: Hosted on JourneyApps Platform using its serverless cloud functions, providing API functionalities. It handles both uploading client writes and generating JWTs for the client application.

Front End App: A client-side application that provides the user interface and handles Create, Read, Update, and Delete (CRUD) operations locally.

MongoDB.

Architecture Overview

Self-Hosted Custom Backend Architecture



Similar architecture, however with a custom backend service self-hosted instead of using JourneyApps Platform.

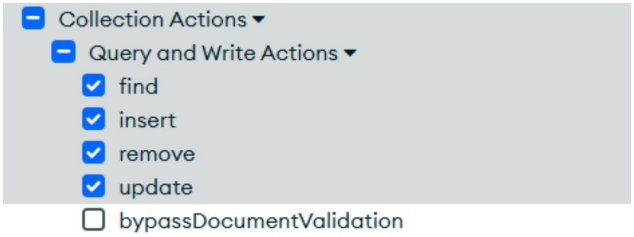
For the purposes of this guide, we will demonstrate running the backend service locally. This is what would be used during development and testing, whereas for production use, the backend service would be deployed in a suitable production environment.

NGROK: Connects the PowerSync Service to the locally hosted backend.

Initial Configuration Steps

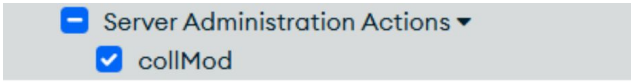
1. Database and Cluster:
   a. Create or select a cluster.
   b. Configure a database (note: the name must be lowercase)
2. Collections:
   a. Create the collections your application needs (e.g., “lists” and “todos” for the demo).
3. User Access:
   a. Create a user with a custom role to ensure it can only access the necessary database.
   b. The custom role should have the following permissions: `find`, `insert`, `remove`, `update`, `changeStream`, `collMod`, `dbStats`, `listCollections`
4. Connection String:
   a. Obtain and save the connection string from the MongoDB Atlas cluster “Connect” option.

Tip: The operations for `find`, `insert`, `remove`, and `update` are categorized under Collection Actions, specifically within Query and Write Actions.



The `changeStream` option falls under Collection Actions/Change Stream Actions.

The `collMod` option is part of Collection Actions/Server Administration Actions.

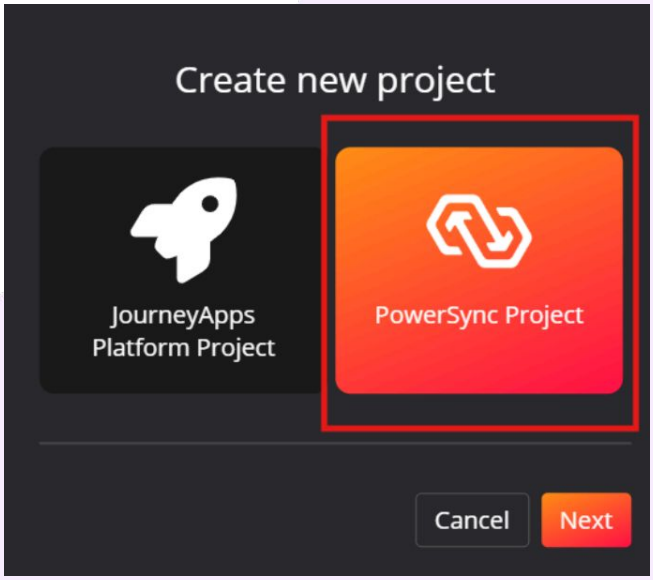


Additionally, the `dbStats` and `listCollections` options are classified under Database Actions and Roles/Actions/Diagnostic Actions.

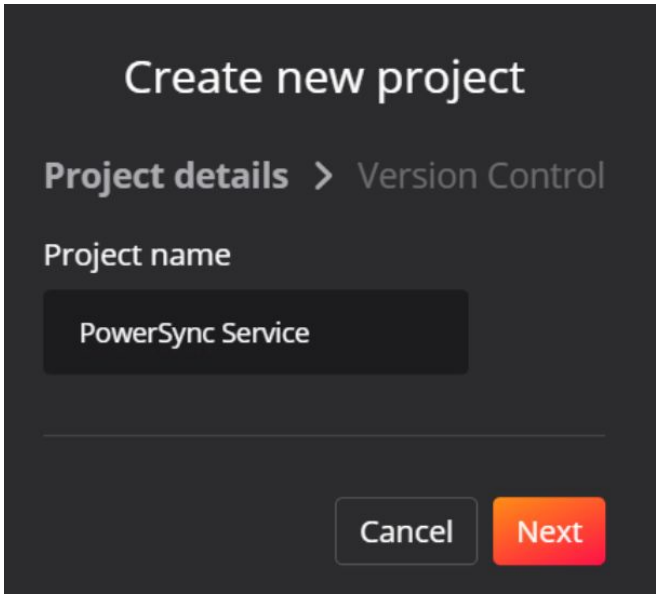
MongoDB.

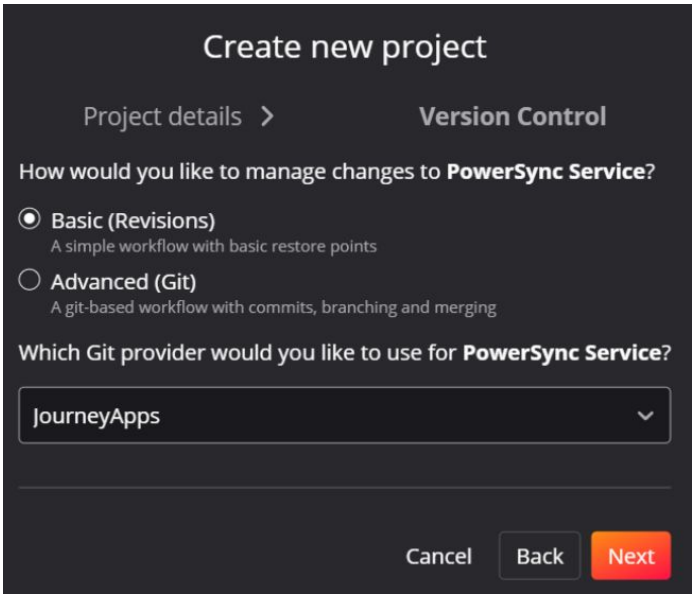
PowerSync Service

1. PowerSync Project Setup:
   a. Login and create a new PowerSync project in the JourneyApps Admin Portal.
   b. Select the PowerSync Project option:

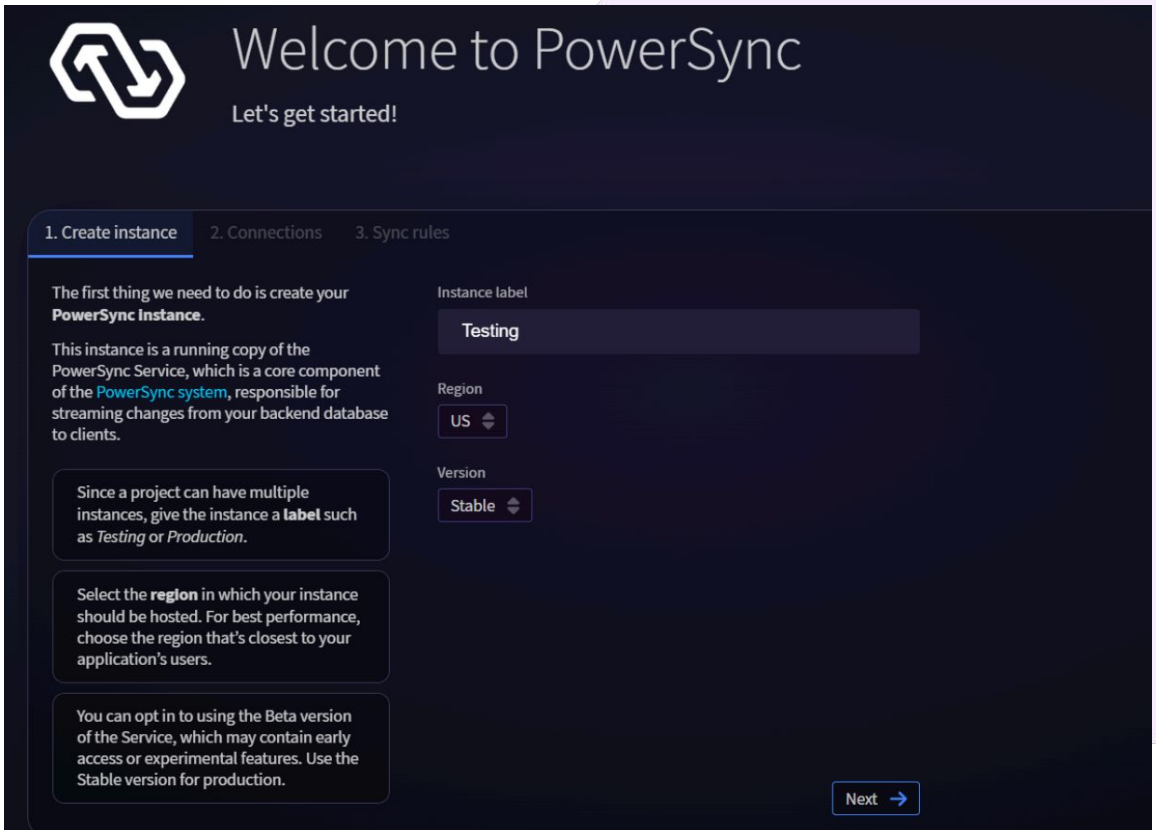


   c. Name the project, click "Next," and then choose JourneyApps as the Git provider:

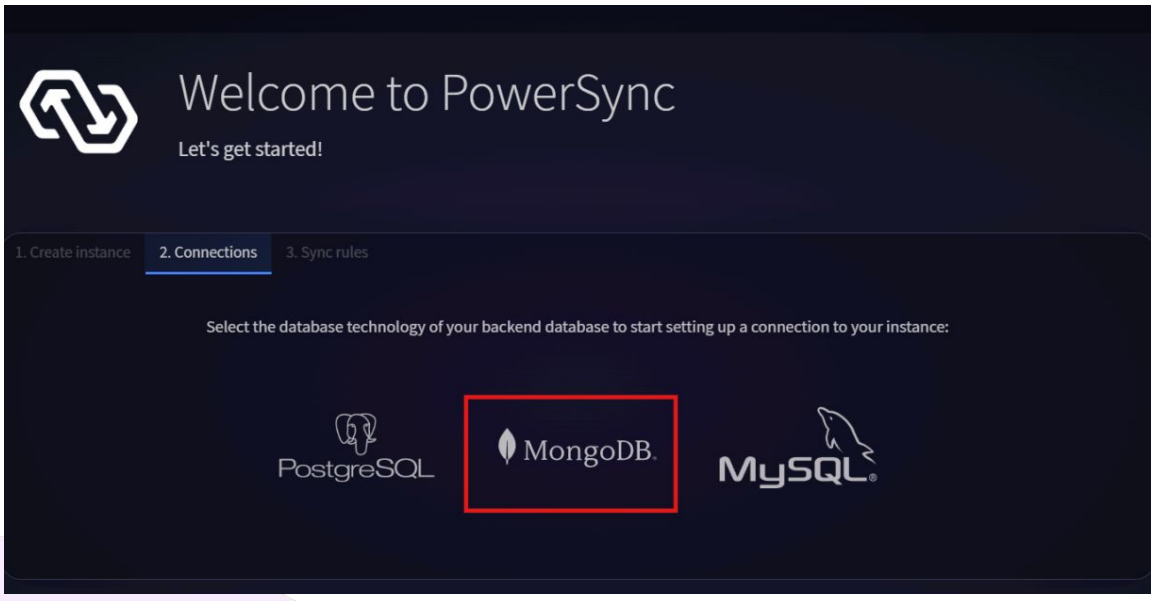




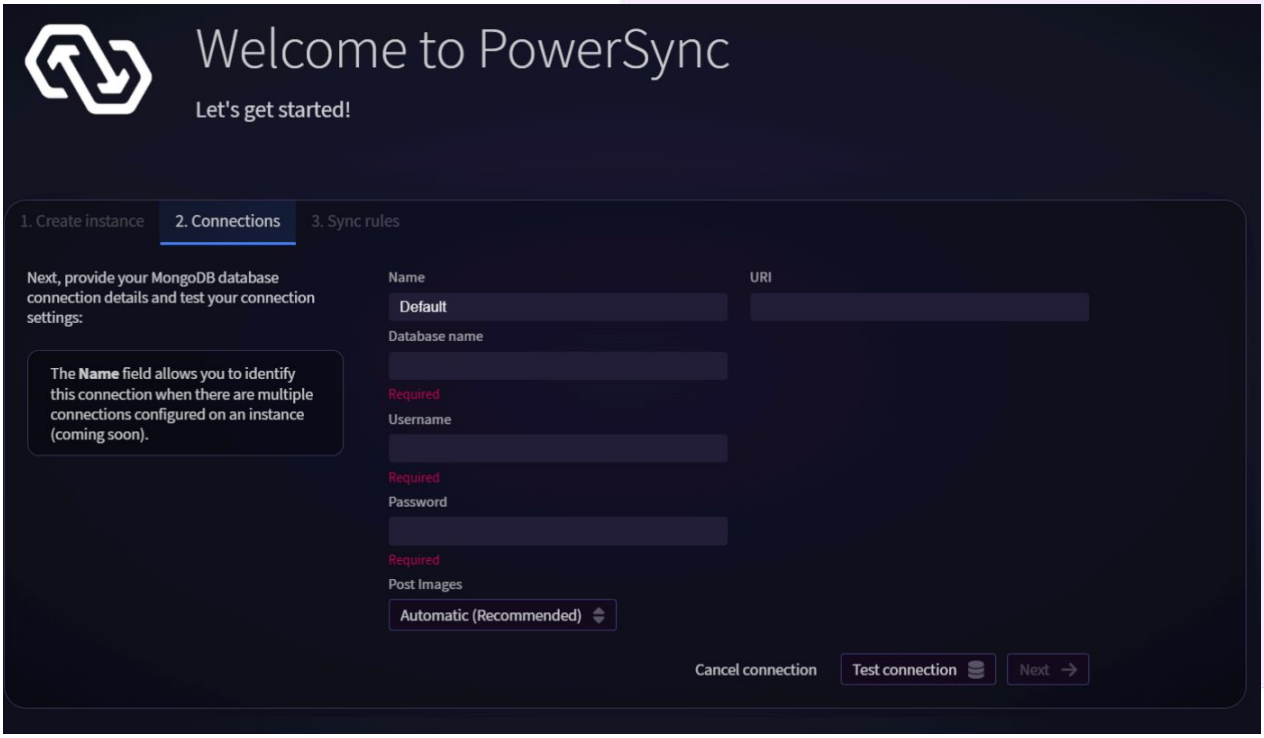
d. Proceed with the setup by configuring a new Instance:



e. In the following step, choose the MongoDB option to connect to the database that was configured earlier:



f. Complete the required fields by entering the database name, username, password, and the connection string you obtained earlier:



g. In the Post Images setting, users can choose from three options:

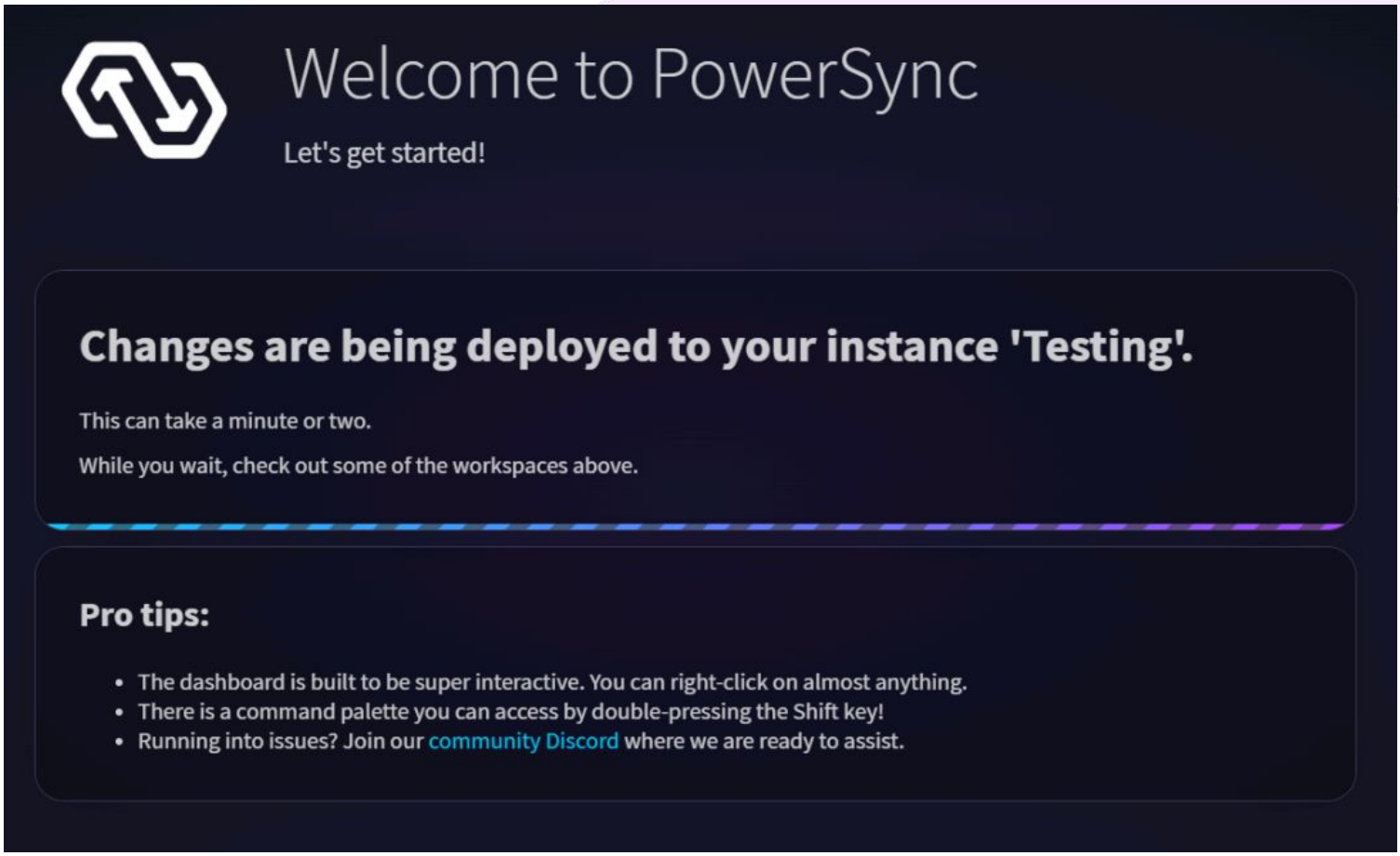
i. Automatic (Recommended): Automatically configures the changeStreamPreAndPostImages option on collections as needed.
ii. Read-only: Utilizes `fullDocument: 'required'` and mandates that `changeStreamPreAndPostImages: { enabled: true }` be set for each collection referenced in sync rules. Replication will fail if this configuration is missing, making it ideal when permissions are limited.
iii. Off (not recommended): Uses `fullDocument: 'updateLookup'` for backward compatibility.

For more detailed information about the Post Images options, please refer to the PowerSync documentation here.

h. Before proceeding, test the connection
i. If the connection is established successfully, proceed by clicking the Next button.

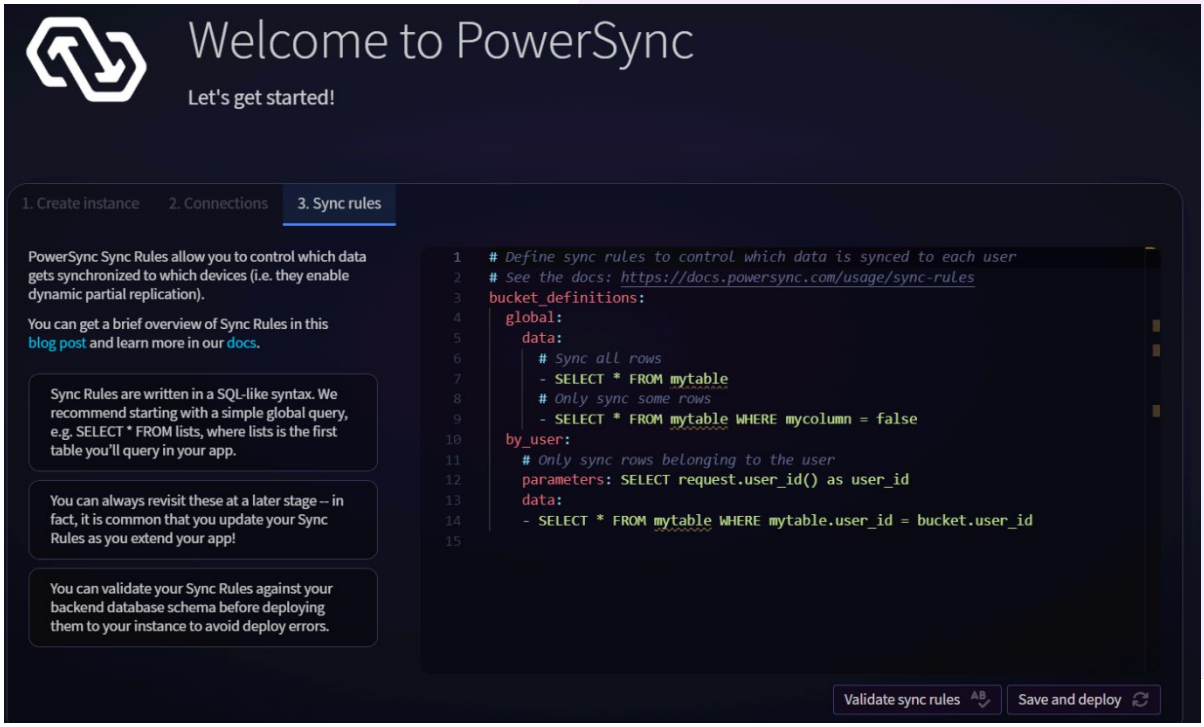
Tip: Should you need to modify the database connection in the future, you can do so by editing the Instance.

j. The changes will then be deployed to the instance:



PowerSync Service

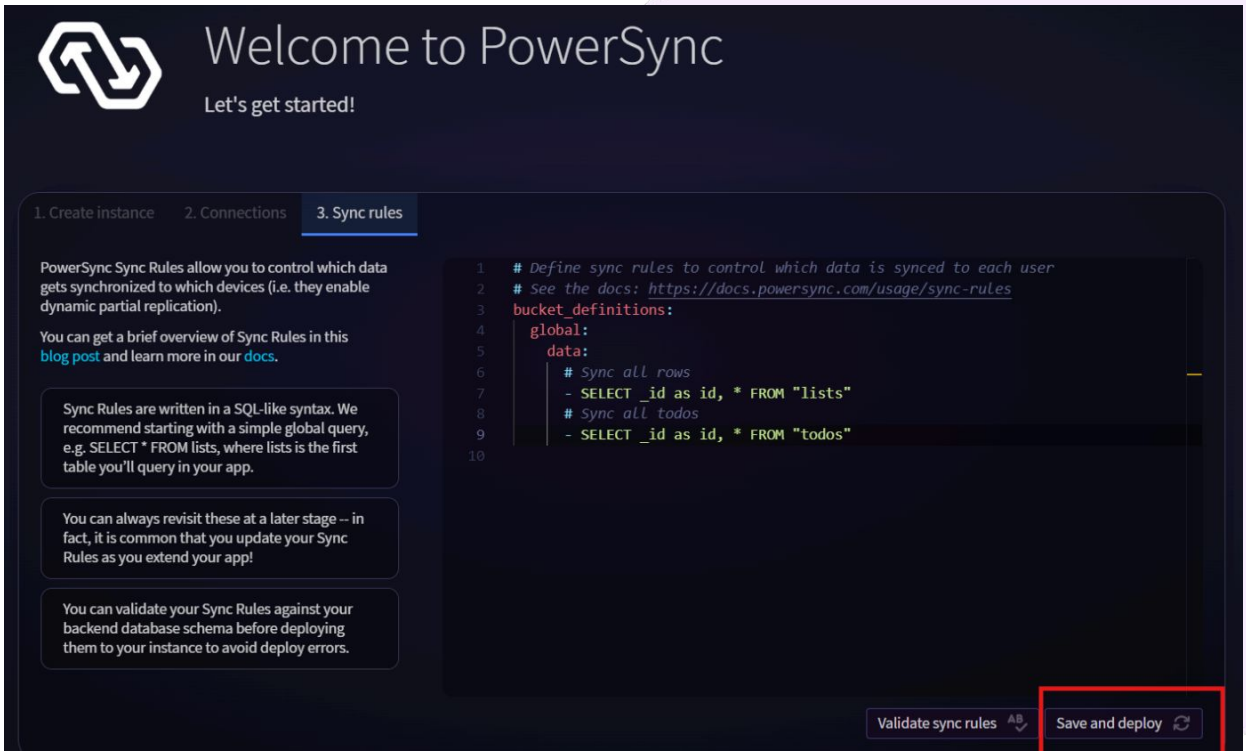
1. Sync Rules Configuration:
    a. Set up the sync rules to determine how the data will be synchronized and identify the clusters that need to be updated:



    i. For the demo backend application, the "todos" and "lists" collections are the ones that need to be updated:

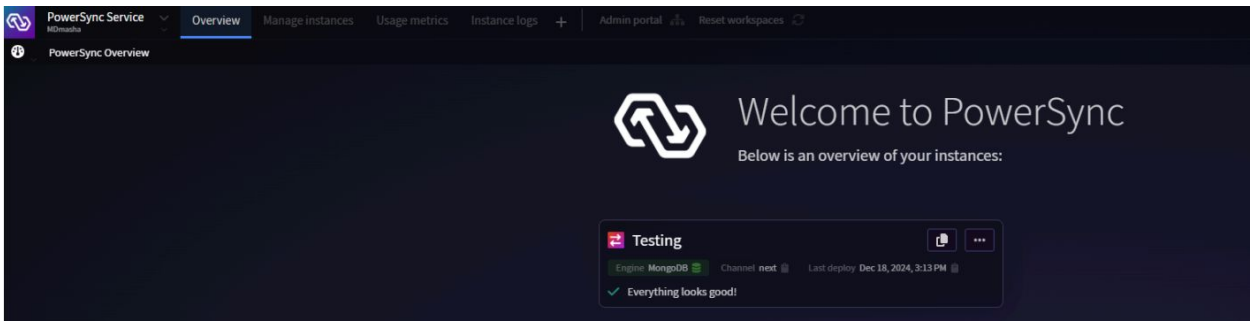
```
Unset
bucket_definitions:
 global:
   data:
     # Sync all rows
     # Sync all lists
     - SELECT _id as id, * FROM "lists"
     # Sync all todos
     - SELECT _id as id, * FROM "todos"
```

b. After configuring the sync rules, they must be deployed to take effect:

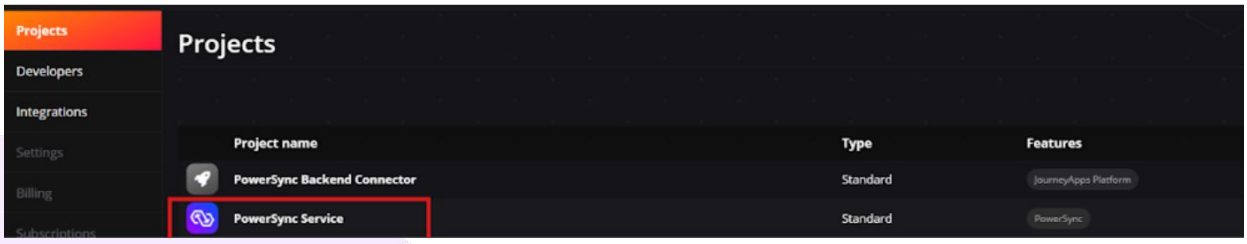


c. Modifying the sync rules will initiate a redeployment of the instance

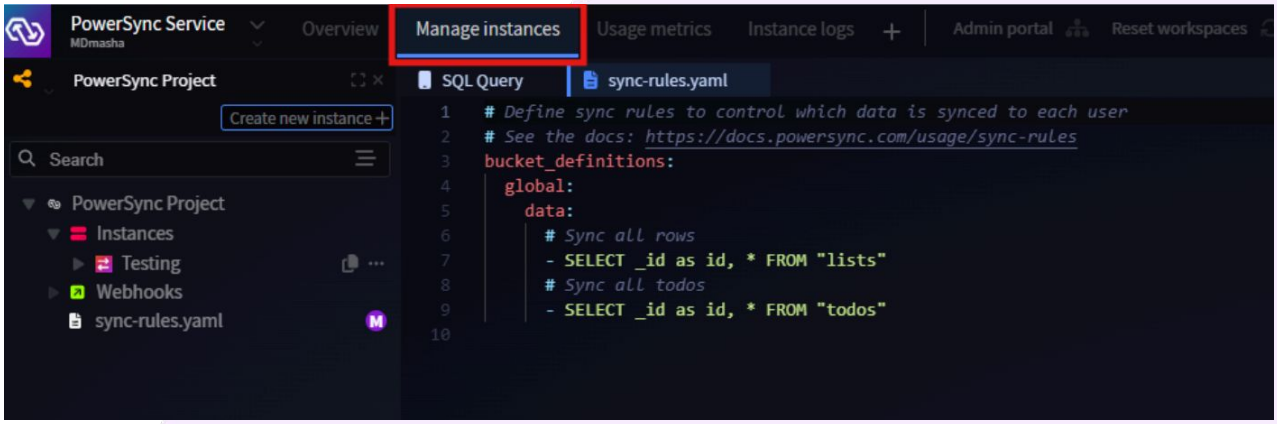
d. If the rules are deployed successfully, the following screen will appear:



e. By clicking the icon  in the top right corner of the screen above to return to the Projects view, you will be able to see the newly created instance:



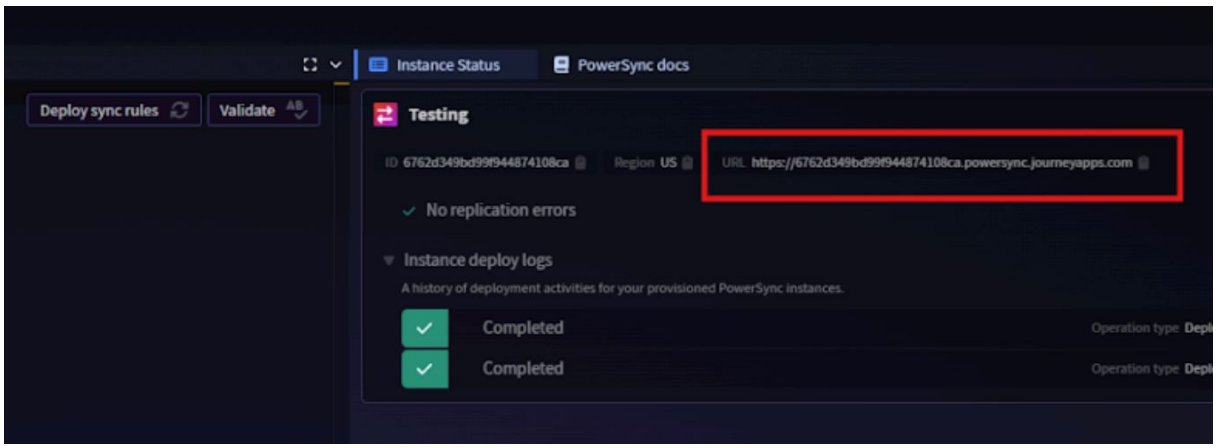
f. To view more details, click on the project and select "Manage Instances" from the top navigation menu:



g. In the left-hand menu can be found the instance called "Testing" together with the `sync-rules.yaml` file which allows to define the rules for data synchronization on the instance. The above document represents a basic configuration of the sync rules, while PowerSync supports more advanced ones.

For instructions on more complex configurations, please consult the PowerSync documentation.

h. In the subsequent setup steps, you will need the PowerSync Service URL and ID, which can be located in the PowerSync Dashboard:

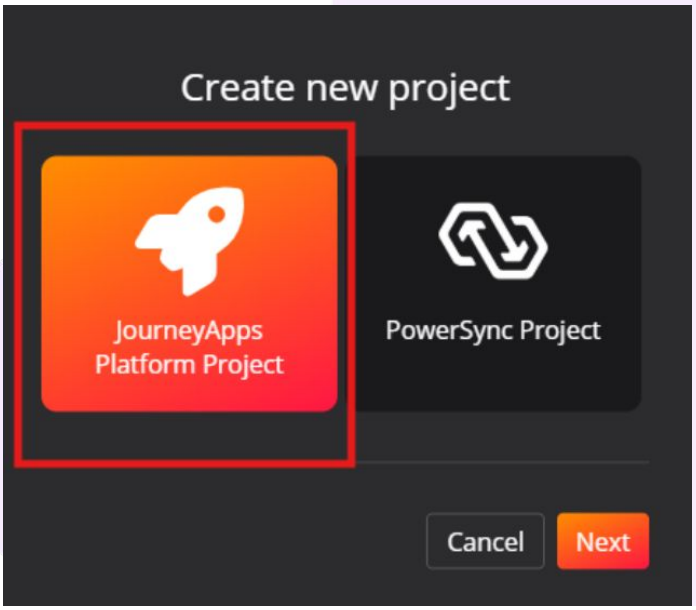


MongoDB.

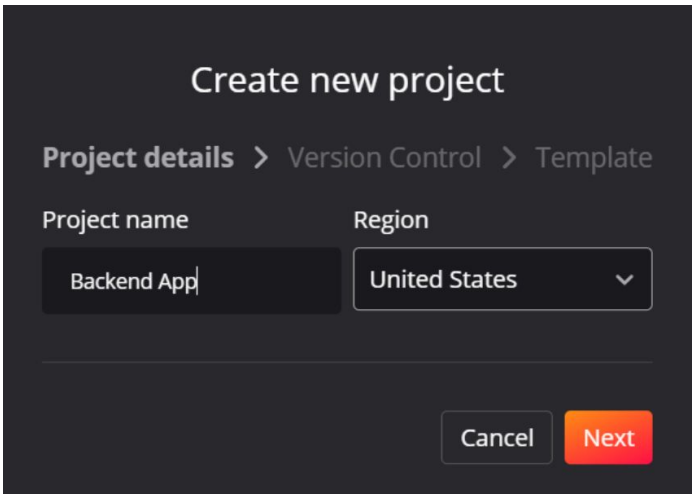
Backend App Setup

JourneyApps Platform Cloud Deployment

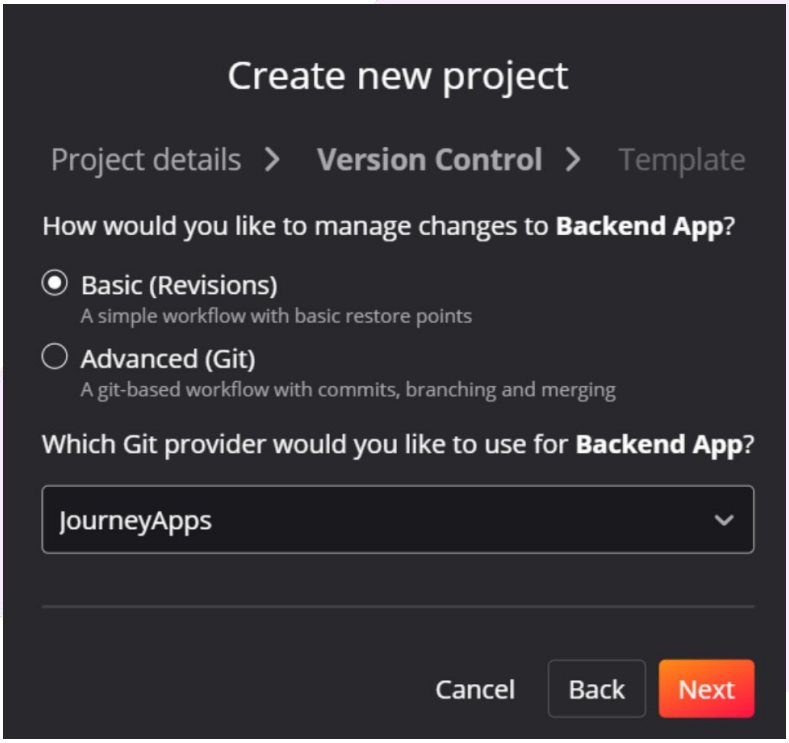
1. New JourneyApps Platform Project:
   a. Go to JourneyApps Admin Portal and set up a new JourneyApps Platform Project.



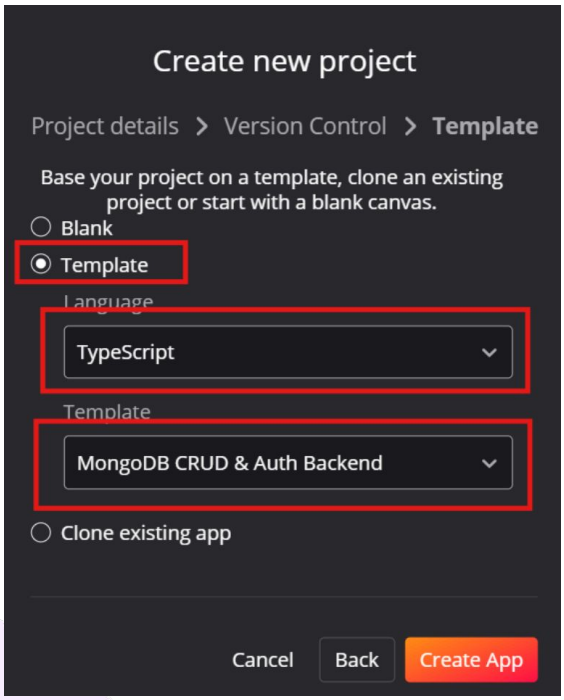
   b. Choose a name for the Backend App and select the deployment region.



c. Next, you can configure the version control system that best suits your requirements. For demonstration purposes, I have chosen the basic option, as shown below:

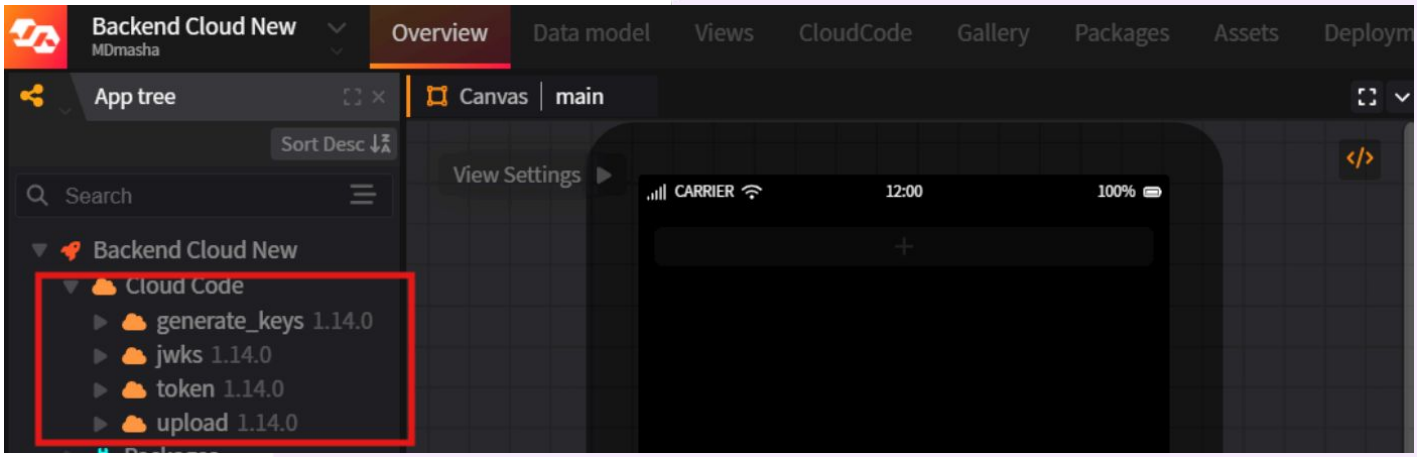


d. Finally, choose the "Template" option. Within this section, set the "Language" to TypeScript and select the template titled "MongoDB CRUD & Auth Backend":



Backend App Setup

e. After completing the setup, you'll be redirected to the Project Overview page. Under the "Cloud Code" section in the left-hand menu, you can find the app's implementation:



d. Below is a breakdown of each of the components:

i. `generate_keys`: This is a task that can be used to generate a private/public key pair which the `jwks` and `token` tasks require. This task does not expose an HTTP endpoint and should only be used for development and getting started.

ii. `token`: This task exposes an HTTP endpoint which has a `GET` function. This task is used by the Frontend App to generate a token to validate against the PowerSync Service. For more information about custom authentication setups for PowerSync, please see this page from the PowerSync docs.

iii. `jwks`: This exposes an HTTP endpoint which has a `GET` function which returns the public JWKS details.

iv. `upload`: This task exposes an HTTP endpoint which has a `POST` function which is used to process the write events from the Frontend App and writes it back to the source MongoDB database.

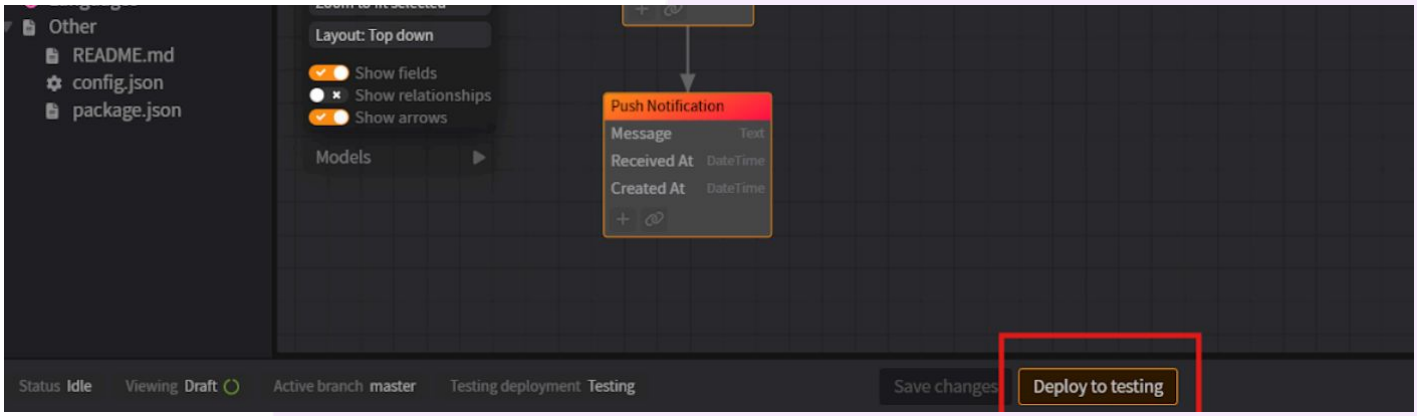
MongoDB.

Backend App Setup

2. Generate Key Pair

Before using the serverless functions you need to generate a public/private key pair. Follow the below steps to generate the key pair:

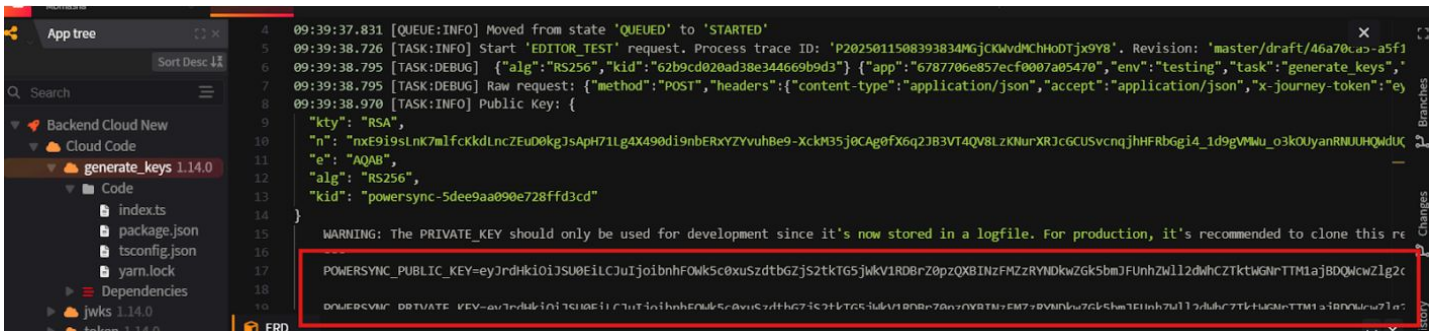
a. Initially, click on the Deploy to Testing button. For the time being, there's no requirement to set up any environment variables for the deployment.



b. Once the Deploy has succeeded, open the `generate_keys` CloudCode task.

c. Click on the `Test CloudCode Task` button at the top right.

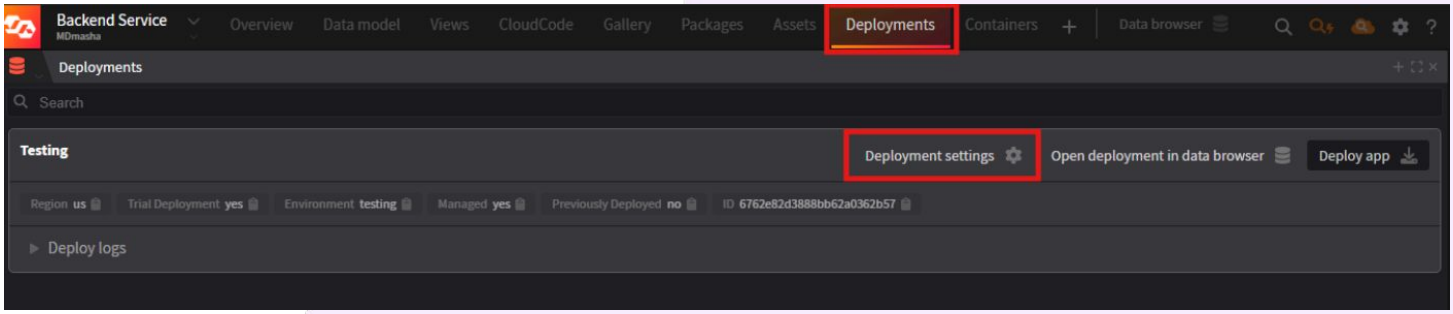
d. This will print the public and private key in the task logs window.



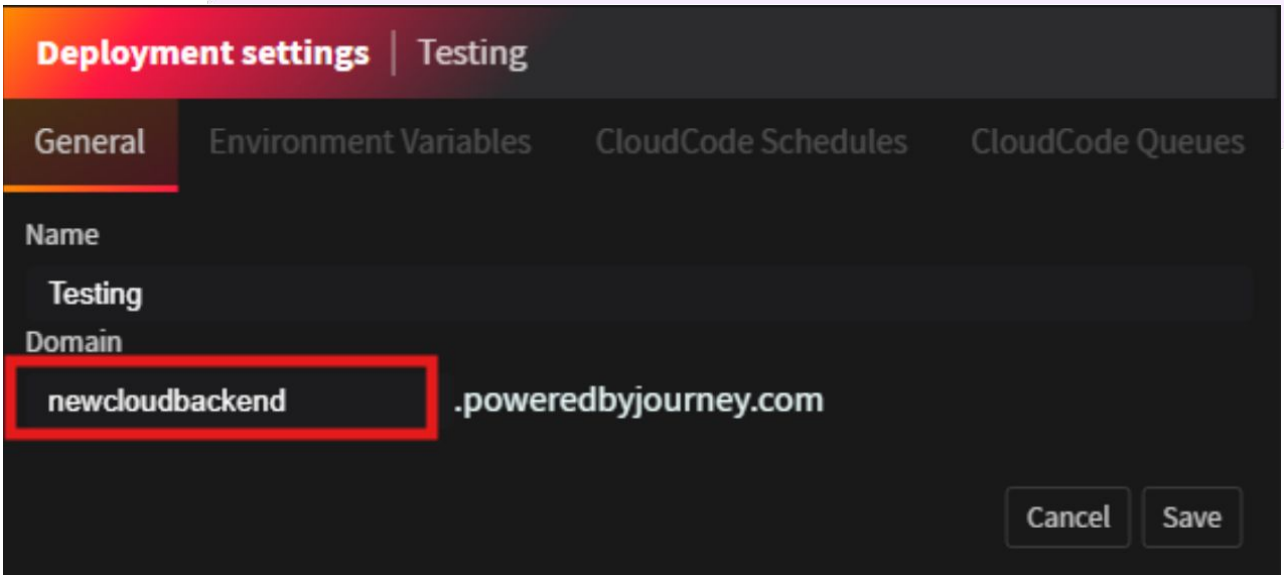
e. Copy and paste the `POWERSYNC_PUBLIC_KEY` and `POWERSYNC_PRIVATE_KEY` to a file — we'll need this in the next step.

Backend App Setup

3. Set Environment Variables and Deploy:
    a. At this stage, the application has been created but not yet deployed to the cloud. In order to deploy the app, the environment variables need to be configured, which can be done by clicking on the "Deployments" tab in the top navigation menu and selecting “Deployment settings”:



    b. In the “General” tab enter a Domain Name (e.g: “newcloudbackend”).



Important: The domain configured here will be used later in the PowerSync Service to set up client authentication.

Backend App Setup

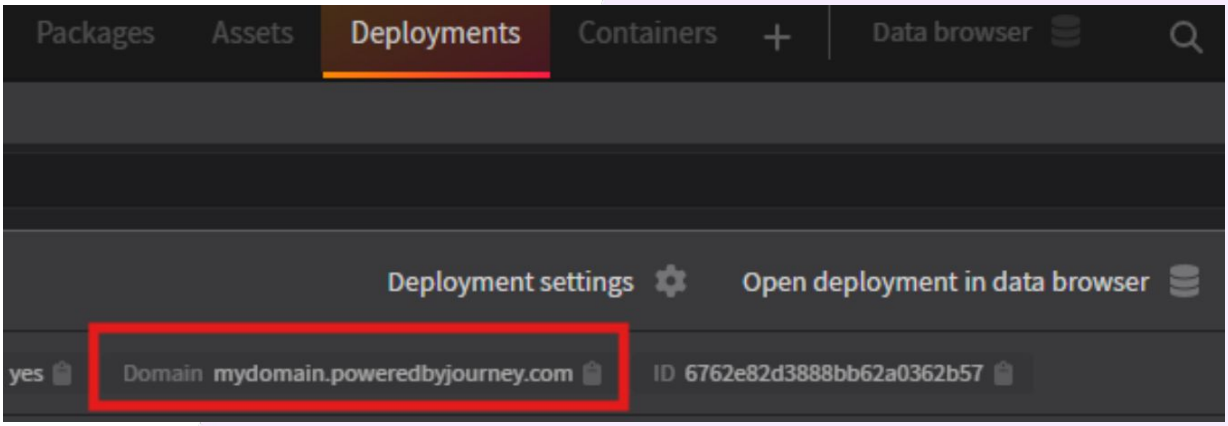
c. On the "Environment Variables" tab, you should configure the following variables to ensure all necessary parameters are in place:
   i. `POWERSYNC_PRIVATE_KEY` and `POWERSYNC_PUBLIC_KEY`: These were produced in the prior step utilizing the key generator built into the JourneyApps Platform.
   ii. `POWERSYNC_URL`: This is the PowerSync Service URL, identical to the one configured in the Front End App.
   iii. `MONGO_URI`: This is the MongoDB connection string, it needs to contain the username, password and the database name in the following format:

```
mongodb+srv://<username>:<password>@<deployment>.mongodb.net/<database_name>
```

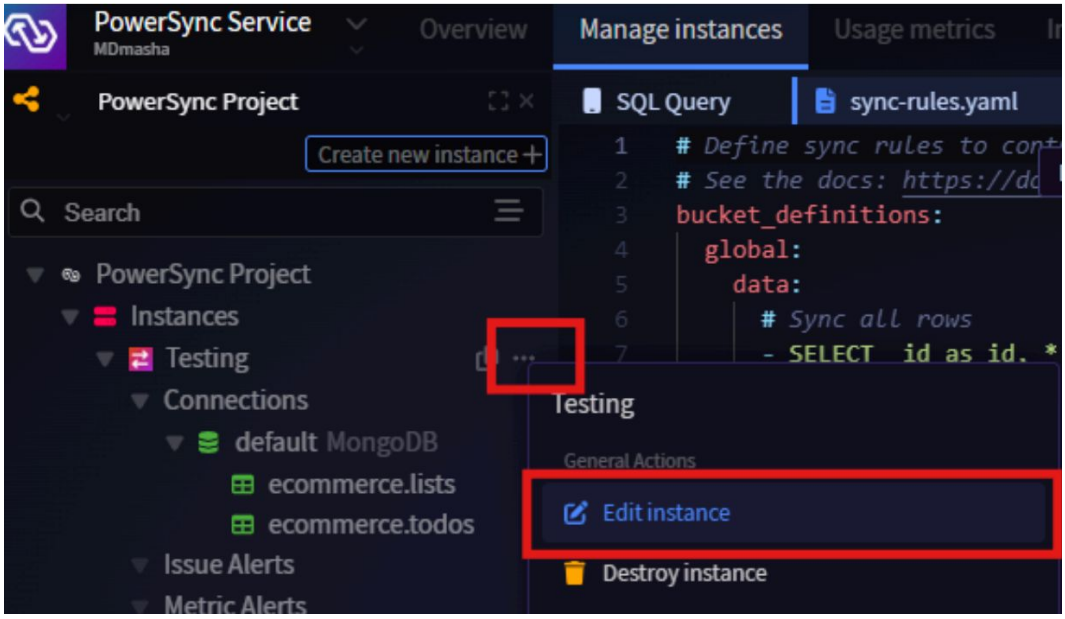
d. After entering all the necessary information, save the changes. At the bottom of the page, you will find the "Deploy to testing" button. Click it to get the Backend App up and running.

Backend App Setup

3. Link with PowerSync
   a. To link the backend app with PowerSync, you'll need the domain from the "Deployments" page



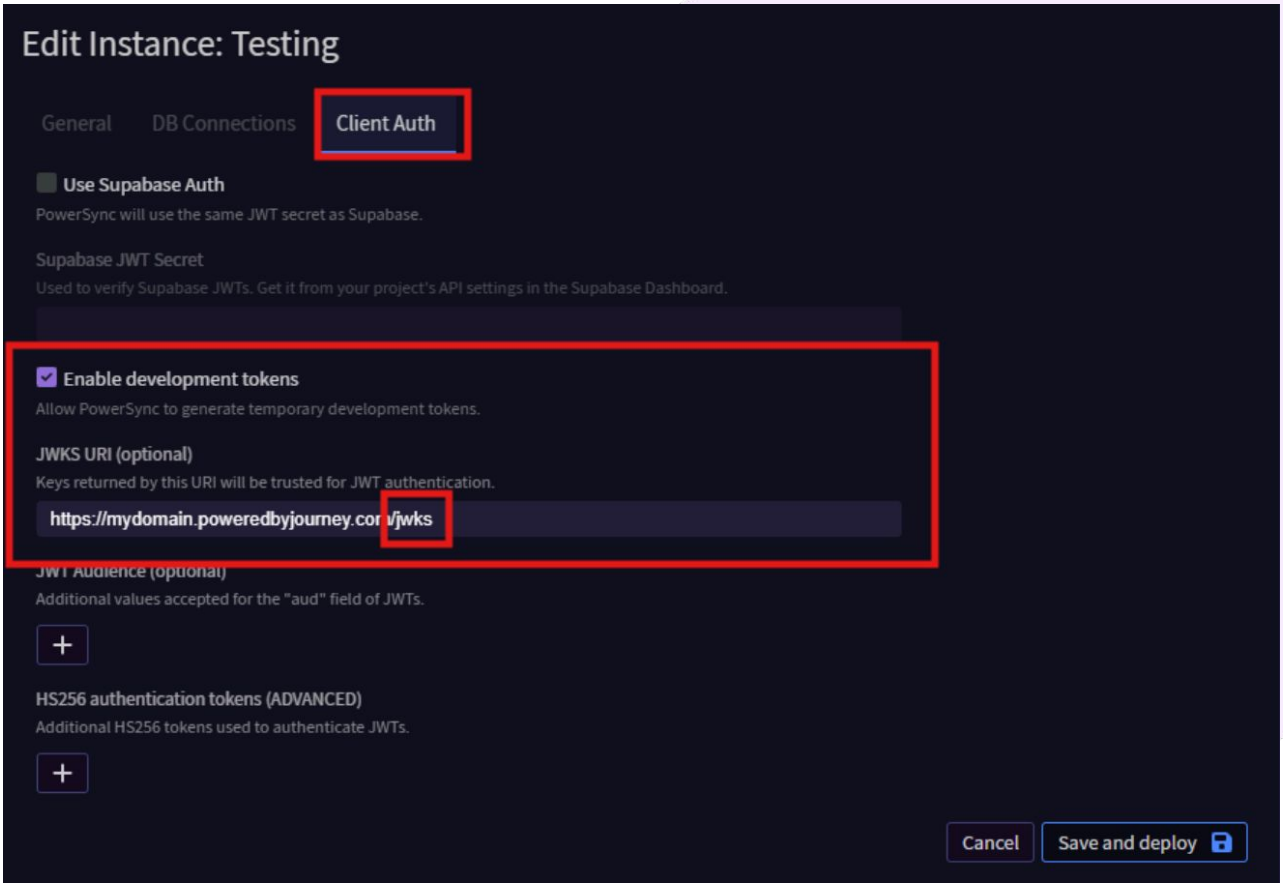
   b. To set up the PowerSync, click the  icon in the top left corner to return to the Projects, then choose the PowerSync project.
   c. On the PowerSync page, click the three dots next to the Instance name and select "Edit Instance":



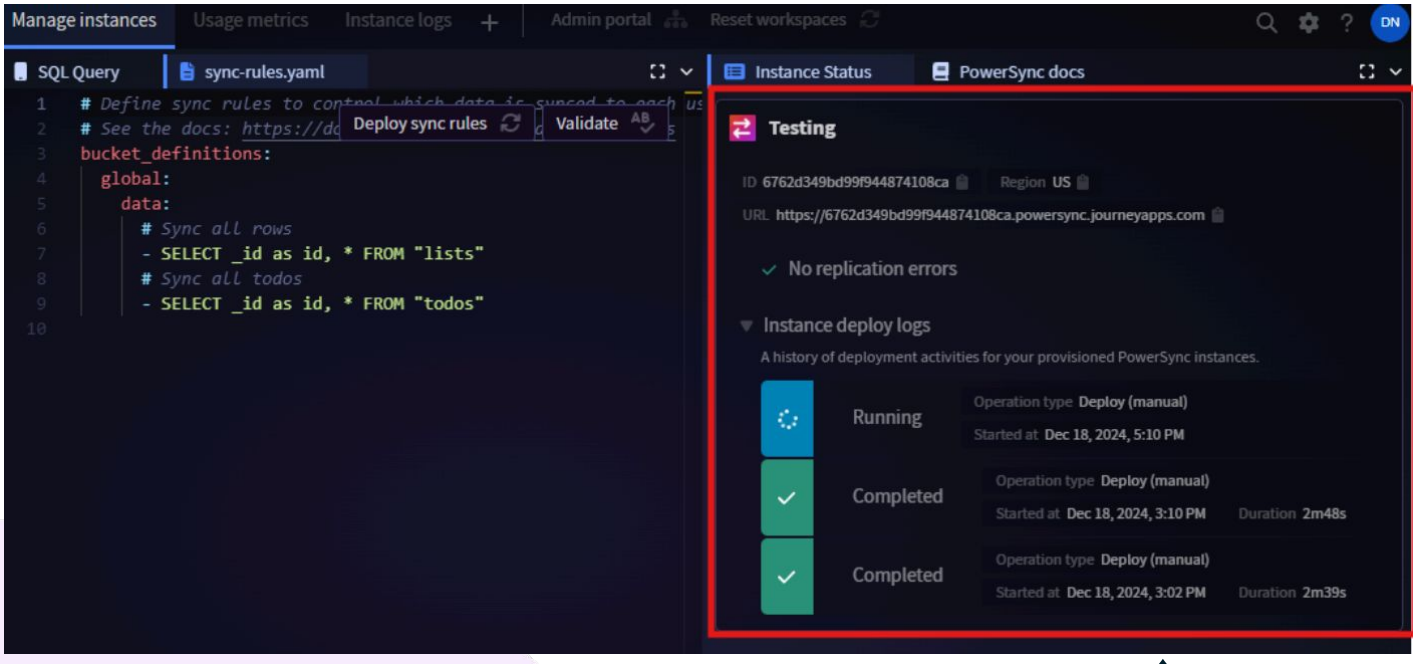
   d. In the Edit Instance window, navigate to the "Client Auth" tab and enable the "Enable development tokens" option. Paste the previously saved Backend App domain URL into the JWKS URI field.

Backend App Setup

Important: Append "/jwks" to the end of the URL, as shown in the example below:



c. After completing the setup, click "Save and deploy." The deployment information is available on the right side of the window:



Self-Hosted Backend Deployment Setup

1. Clone the Backend Demo:

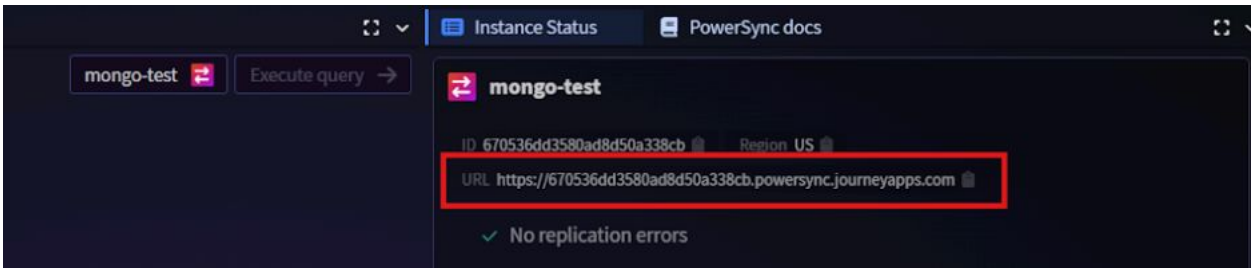
This will clone a Node.js custom backend demo/example provided by PowerSync:

```
Unset
git clone
https://github.com/powersync-ja/powersync-nodejs-backend-todolist
-demo.git
```

2. Environment Configuration:
   a. Copy the `.env.template file` in the root directory and rename the copied file to `.env`.

```
Unset
cp .env.template .env
```

   b. Fill out the .env file with the necessary database and synchronization parameters.
      i. `POWERSYNC_PRIVATE_KEY` and `POWERSYNC_PUBLIC_KEY`: These fields can be left empty as they will be autogenerated
      ii. `POWERSYNC_URL`: This information is available in the PowerSync Dashboard



      iii. `PORT`: The backend application will be deployed on this port
      iv. `JWT_ISSUER`: This will be used later when setting up client authentication in the Instance on the PowerSync Dashboard
      v. `DATABASE_TYPE`: Specifies the database the application will sync with, which in this case is MongoDB
      vi. `DATABASE_URI`: The connection string including the username and password

Self-Hosted Backend Deployment Setup

Important: Ensure the database URI includes the database name configured earlier in the MongoDB connection on the PowerSync Dashboard. If not specified, it will default to looking for a database named “test.”

```
Unset
POWERSYNC_PRIVATE_KEY=
POWERSYNC_PUBLIC_KEY=
POWERSYNC_URL=<PowerSync URL>
PORT=6060
JWT_ISSUER=powersync
# Either 'mongodb', 'mysql' or 'postgres'. This defaults to
Postgres
DATABASE_TYPE=mongodb
DATABASE_URI=mongodb+srv://<username>:<password>@<clusterName>.mo
ngodb.net/<databaseName>?retryWrites=true&w=majority&appName=<Mon
goDB Cluster Name>
```

Tip: Replace username and password with your MongoDB Atlas username and password, clusterName with the name of your MongoDB Atlas cluster, and databaseName with the name of your database.

3. Run Backend App Locally
    a. Install and start via npm:

```
Unset
npm install
npm start
```

Self-Hosted Backend Deployment Setup

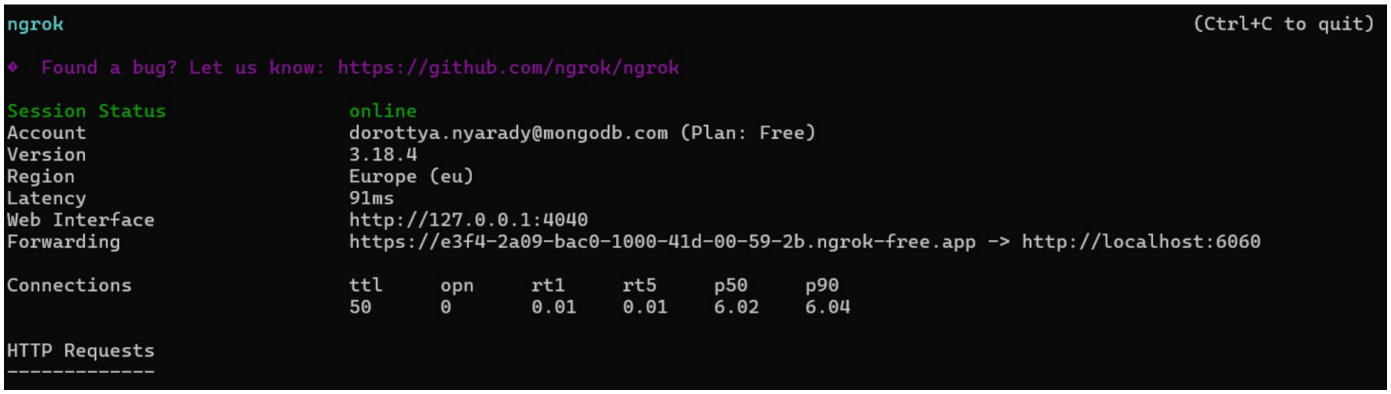
4. NGROK Configuration:
    a. Use NGROK to expose your local server.
    b. Download and install ngrok. If you don't have an account, you'll need to create a new one.
    c. Run the command below to add your authtoken to the default `ngrok.yml` configuration file.

```
Unset
ngrok config add-authtoken <Auth token>
```

5. NGROK Execution:
    a. Executing the command below will bring the previously configured backend app online at a temporary domain. For instance, if the provided example is used, it will be accessible at http://localhost:6060:

```
Unset
ngrok http 6060
```

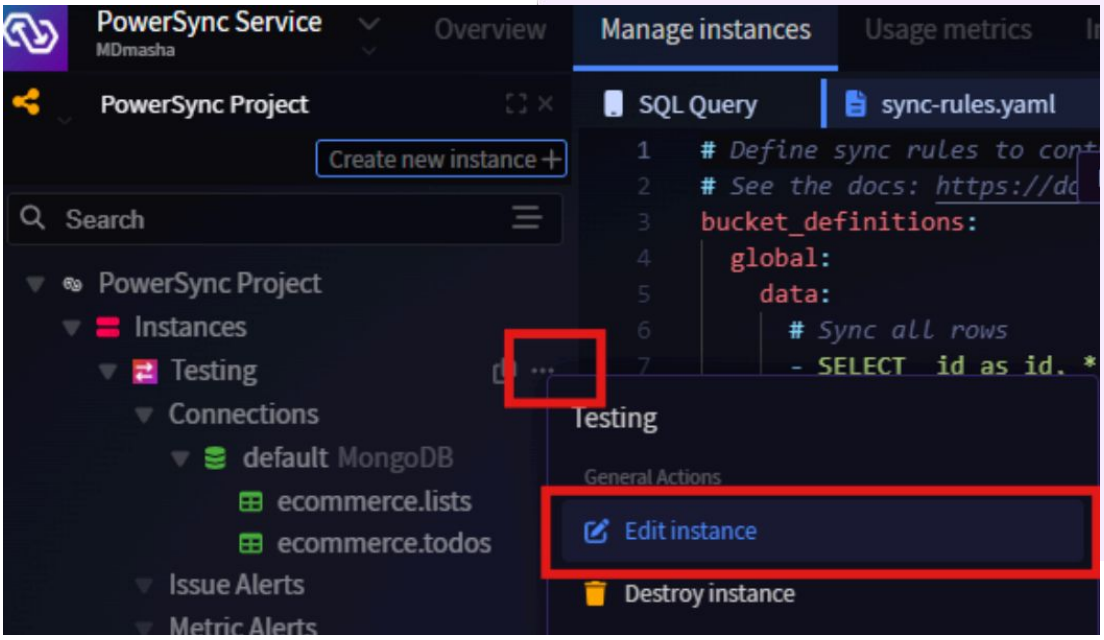
    b. The Terminal will display the following message:

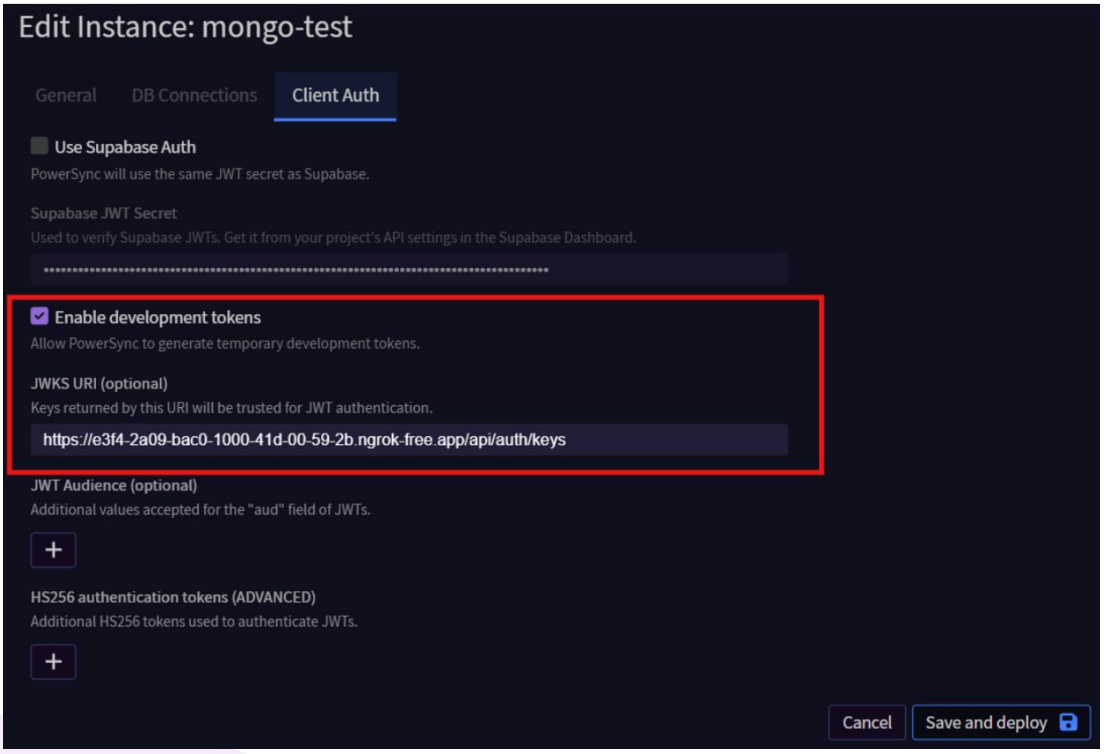


Self-Hosted Backend Deployment Setup

6. Link with PowerSync:
    a. Open the PowerSync Dashboard, edit the Instance, and paste the Forwarding URL that begins with HTTPS into the Client Auth tab.

Important: Ensure you append the authentication endpoint `/api/auth/keys` to the end of the URL.





b. After entering the value, be sure to click Save and deploy.

Frontend App Setup (Demo To-Do Application)

1. Clone the demo code:

```
Unset
git clone https://github.com/powersync-ja/self-host-demo.git
```

2. Environment Variables:
   a. Navigate to the `self-host-demo` folder, then proceed to `demos -> nodejs -> demo-app`, and copy the `.env.template` using the following command:

```
Unset
cp .env.template .env.local
```

   b. In the `.env.local` file, configure the following variables:
      i. `VITE_BACKEND_URL`: Enter the Instance URL from the PowerSync Dashboard.
      ii. `VITE_POWERSYNC_URL`: This information is available in the PowerSync Dashboard
      iii. `VITE_CHECKPOINT_MODE`: This can remain as the default value, "managed."
3. Modify Connector Configuration for JourneyApps Platform:
   a. If you are using the cloud deployment of the backend app with JourneyApps Platform, you need to update the specific endpoints in the `DemoConnector.ts` file.
   b. In the `self-host-demo/demos/nodejs/demo-app/library/powersync/DemoConnector.ts` file, update the `tokenEndpoint` variable to the one shown below (changing from `api/auth/token` to `token`):

```
Unset
 async fetchCredentials() {

    const tokenEndpoint = 'token';

...
```

   c. Additionally, within the same file, modify the uploadData function to the one below (switching from /api/data to /upload):

```
Unset
 const response = await fetch(`${this.config.backendUrl}/upload`
....
```

Frontend App Setup (Demo To-Do Application)

4. Run Frontend App
    a. Install and start the local server:

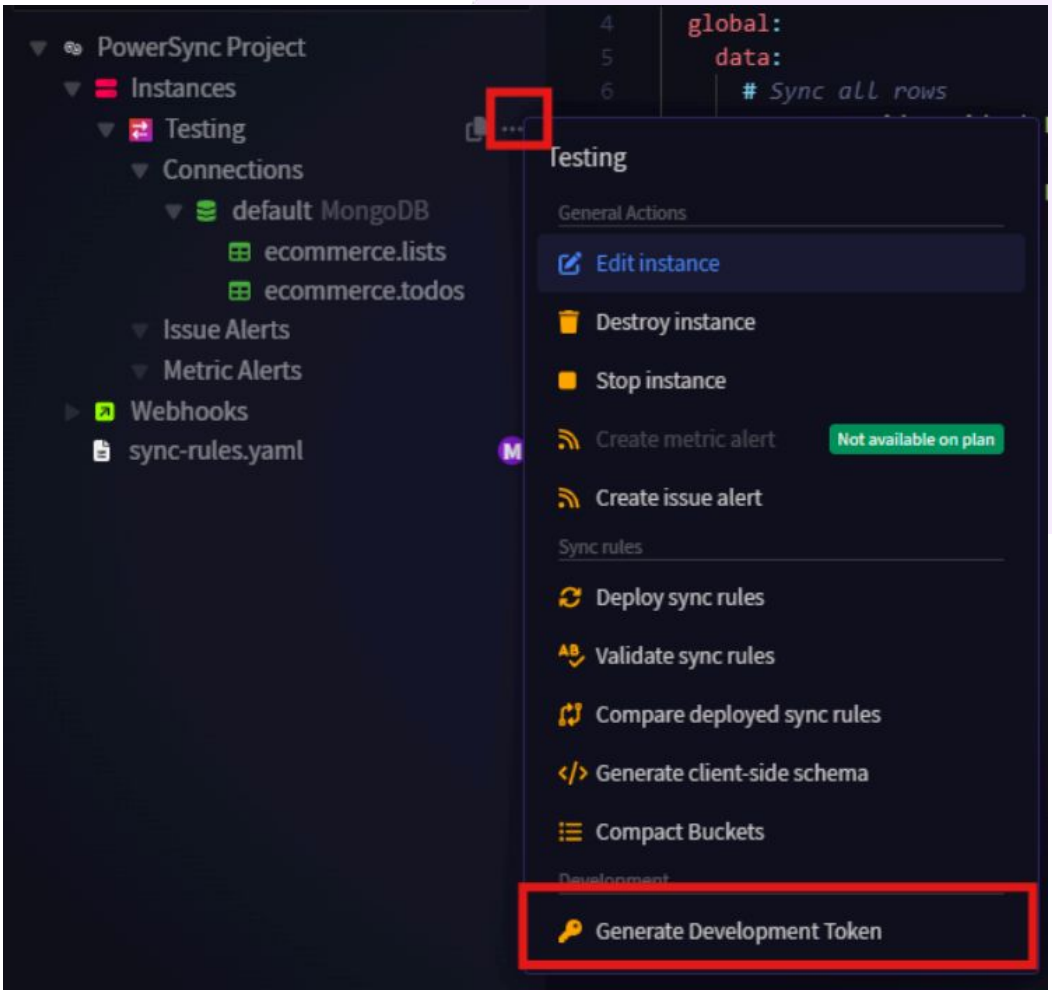
```
Unset
pnpm install

pnpm start
```

    b. You can access the to-do list UI (typically at the following URL: http://localhost:4173/)

Diagnostics Tool Overview

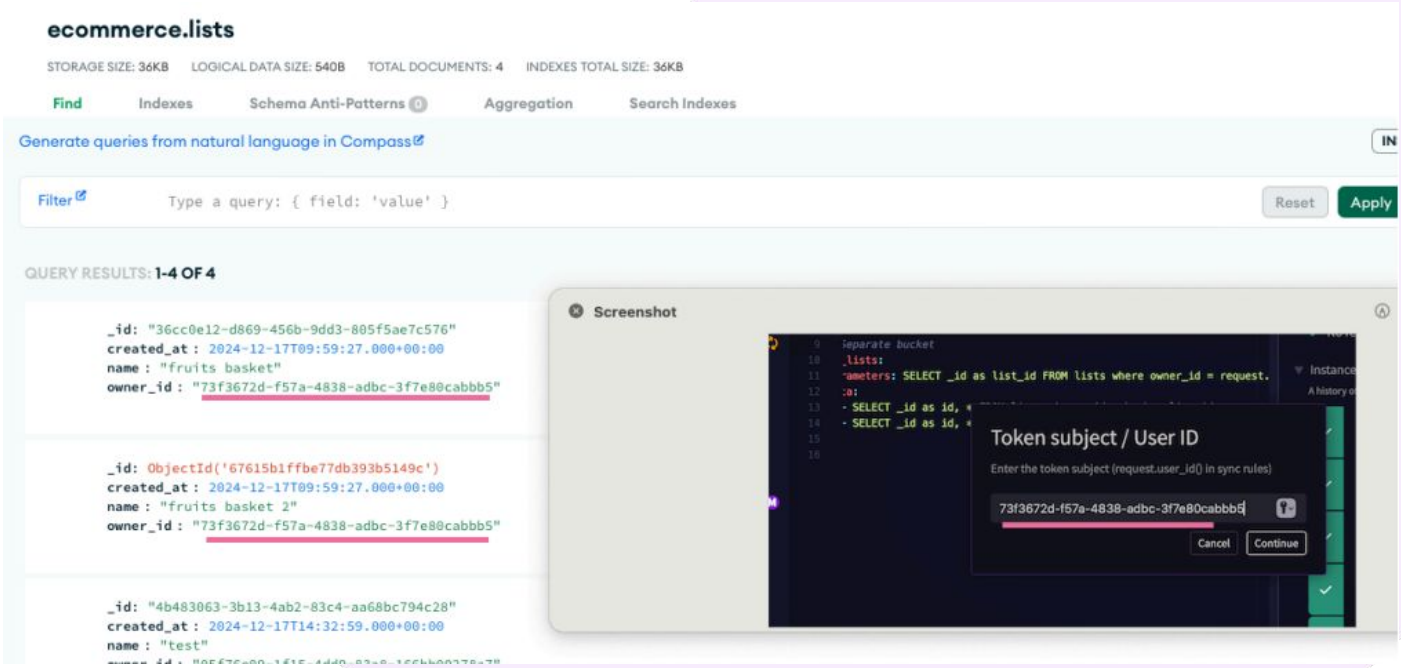
The diagnostics tool helps verify and visualize data synchronization processes. It can be accessed and utilized by generating a development token from the PowerSync Dashboard. To create a development token, follow these steps:

1. Navigate to the PowerSync Dashboard, click the three dots next to your instance, and choose "Generate Development Token":

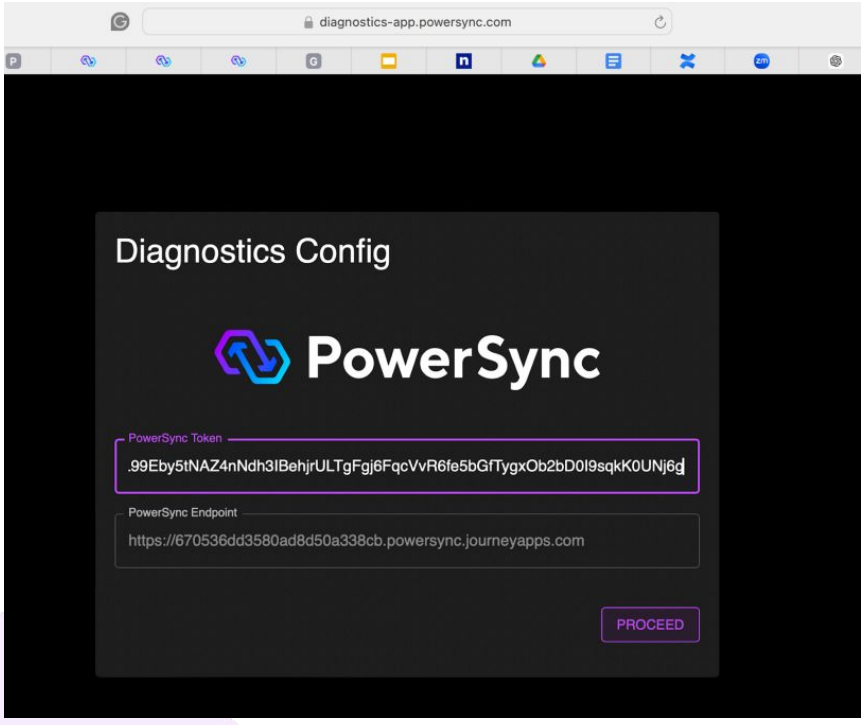


Diagnostics Tool Overview

2. If your sync rules involve any user-specific criteria, you can specify the user ID at this stage. You can find the user ID in the database, as shown in the example below:

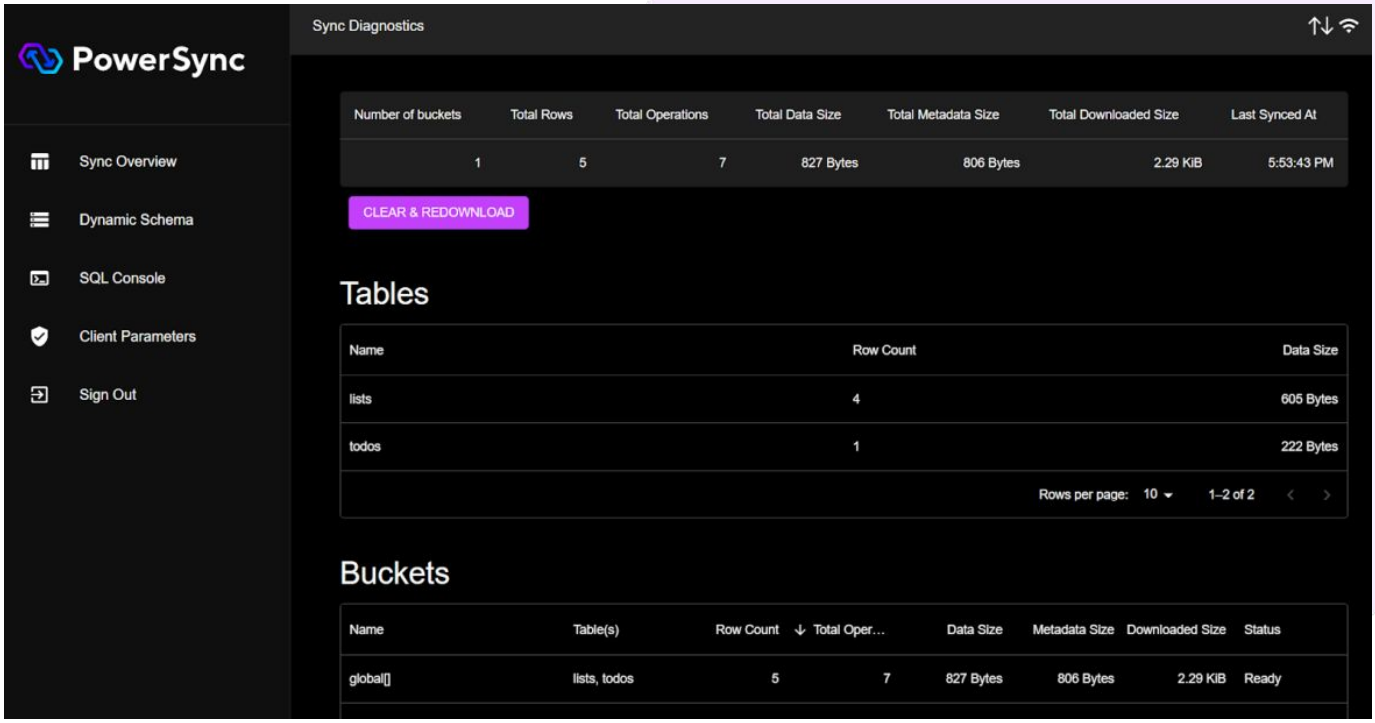


3. Copy the generated token, keeping in mind that it will expire in 12 hours and will need to be refreshed.
4. Open the Diagnostics app and input both the generated token and the PowerSync Service URL to establish a connection with the app



Diagnostics Tool Overview

Within the app, you'll be able to view the available tables and any configured buckets according to the sync rules. An example is provided below:



Real-time Data Made Easy

With the walkthrough provided, you are now equipped to set up a dynamic and reactive application that integrates PowerSync with MongoDB Atlas for real-time data synchronization. This setup enhances the flexibility and scalability of modern applications, ensuring data remains current and consistent across all fronts.

Continue Exploring:

- PowerSync Documentation
- MongoDB Atlas for Real-time Apps

By leveraging these technologies, your application can achieve a new level of dynamism and responsiveness, providing you a competitive edge in your field.

MongoDB.