



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)
- 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: "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"
```

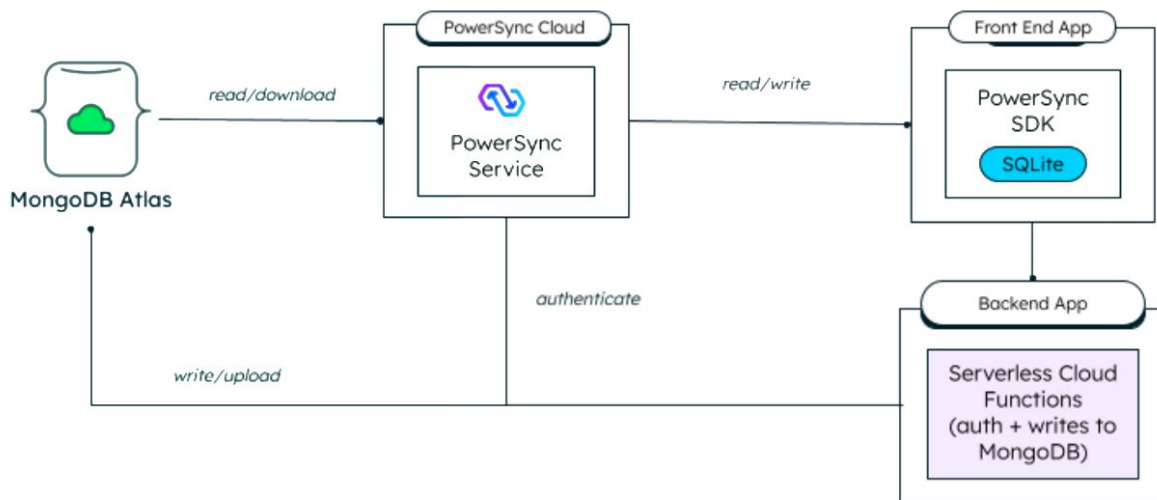
```
_id: "9427ea1c-6f04-407f-a310-0bbf805919a1"  
completed: false  
created_at: 2024-12-19T09:02:06.000+00:00  
created_by: "05f76e09-1f15-4dd9-83a8-166bb09278a7"  
description: "task1"  
list_id: "36cc0e12-d869-456b-9dd3-805f5ae7c576"
```

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

Architecture Overview

JourneyApps Platform Hosted Backend Architecture

PowerSync Integration with MongoDB Atlas with serverless cloud functions



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.

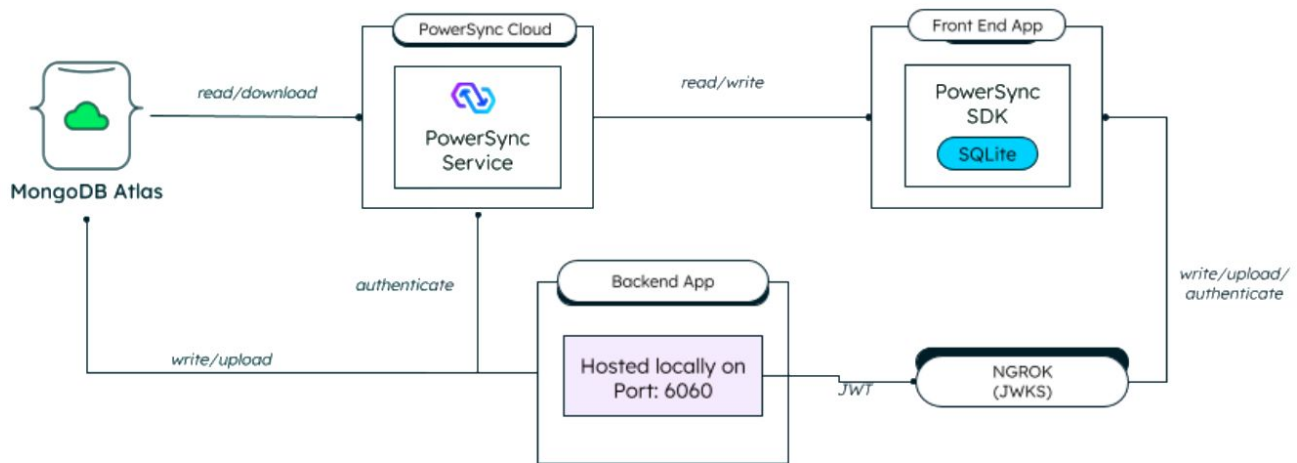
Backend App: Hosted on JourneyApps Platform using its serverless cloud functions, providing API functionalities. It handles both uploading client writes and generating JWTs for the client application.

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

Architecture Overview

Self-Hosted Custom Backend Architecture

PowerSync Integration with MongoDB Atlas with local backend



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

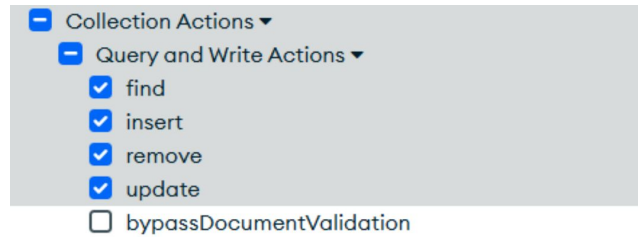
For the purposes of this guide, we will demonstrate running the backend service locally. This is what would be used during development and testing, whereas for production use, the backend service would be deployed in a suitable production environment.

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

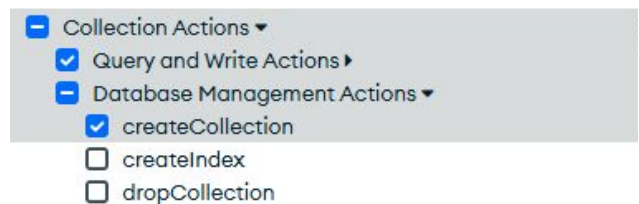
Initial Configuration Steps

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`, `createCollection`, `changeStream`, `collMod`, `dbStats`, `listCollections`
4. Network Access:
 - a. Depending on the region where your cluster is deployed, you'll need to add the corresponding PowerSync Cloud IP addresses. You can do this in the Network Access section under the IP Access List. [Here](#) is a detailed list of IP addresses categorized by region for your reference.
5. Connection String:
 - a. Obtain and save the connection string from the MongoDB Atlas cluster “Connect” option.

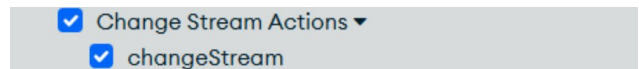
Tip: The operations for `find`, `insert`, `remove`, and `update` are categorized under Collection Actions, specifically within Query and Write Actions.



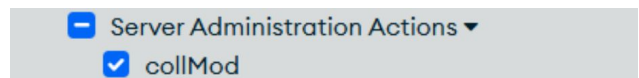
The `createCollection` option can be found under Collection Actions/ Database Management Actions



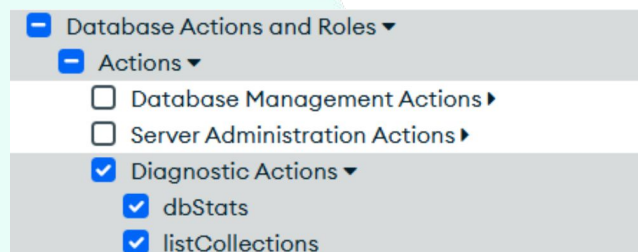
The `changeStream` option falls under Collection Actions/Change Stream Actions.



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

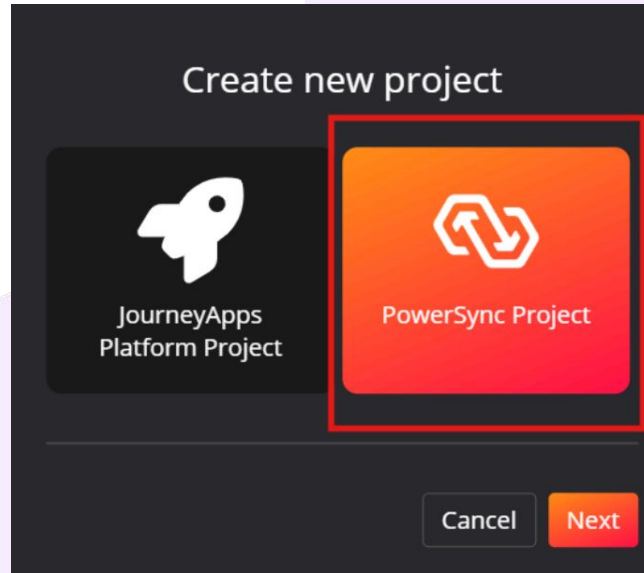


Additionally, the `dbStats` and `listCollections` options are classified under Database Actions and Roles/Actions/Diagnostic Actions.

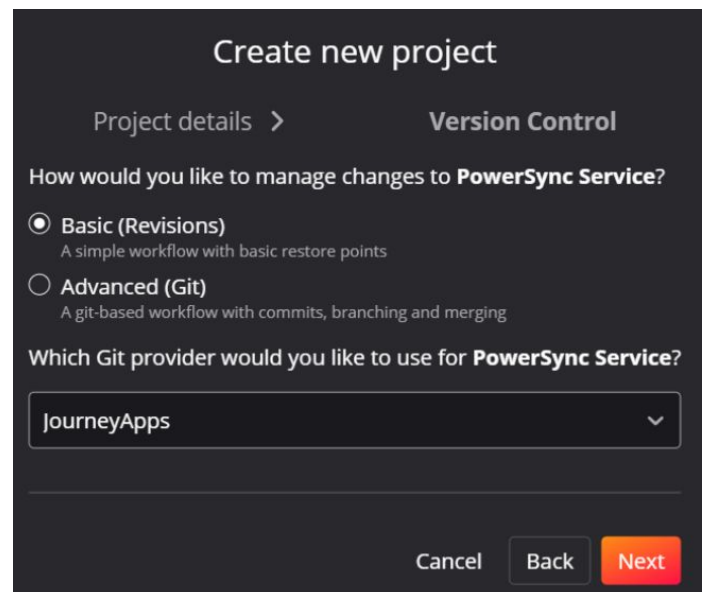
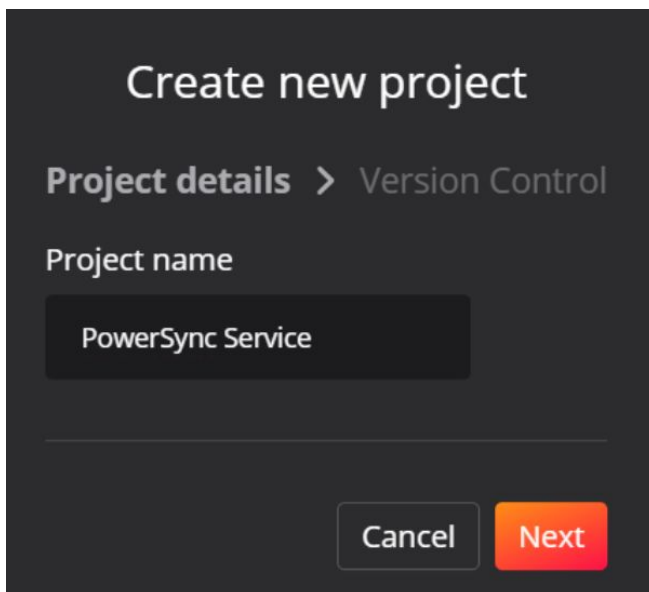


PowerSync Service

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

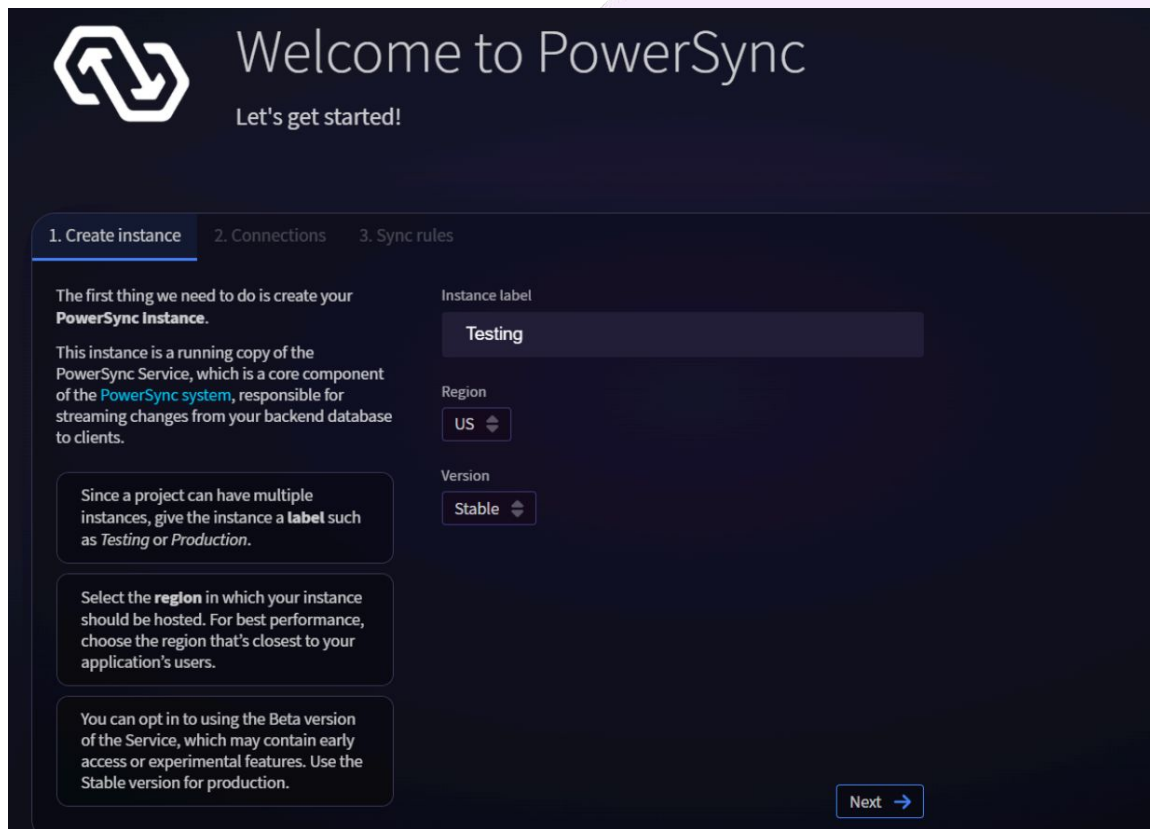


- c. Name the project, click "Next," and then choose JourneyApps as the Git provider:



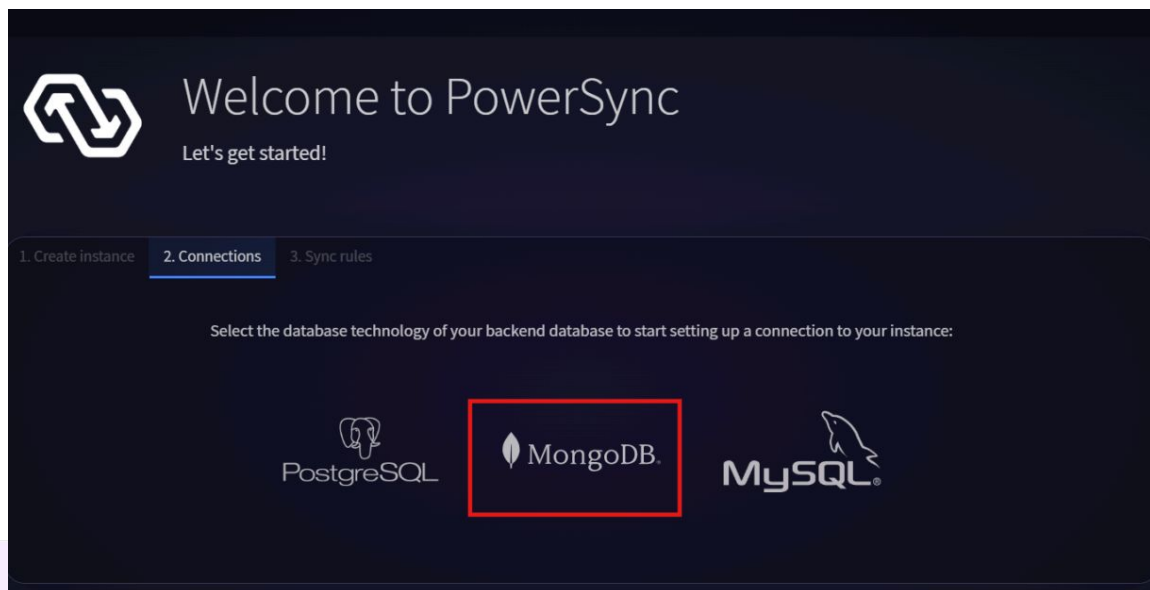
PowerSync Service

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



The screenshot shows the 'Welcome to PowerSync' interface with the 'Let's get started!' message. The navigation bar at the top has three tabs: '1. Create instance' (active), '2. Connections', and '3. Sync rules'. The main content area is divided into two columns. The left column contains instructional text: 'The first thing we need to do is create your PowerSync Instance. This instance is a running copy of the PowerSync Service, which is a core component of the PowerSync system, responsible for streaming changes from your backend database to clients.' Below this, there are three informational boxes: 'Since a project can have multiple instances, give the instance a label such as Testing or Production.', 'Select the region in which your instance should be hosted. For best performance, choose the region that's closest to your application's users.', and 'You can opt in to using the Beta version of the Service, which may contain early access or experimental features. Use the Stable version for production.' The right column contains configuration fields: 'Instance label' with a text input containing 'Testing', 'Region' with a dropdown menu showing 'US', and 'Version' with a dropdown menu showing 'Stable'. A 'Next' button with a right arrow is located at the bottom right of the form.

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



The screenshot shows the 'Welcome to PowerSync' interface with the 'Let's get started!' message. The navigation bar at the top has three tabs: '1. Create instance', '2. Connections' (active), and '3. Sync rules'. The main content area has a heading 'Select the database technology of your backend database to start setting up a connection to your instance:'. Below this heading, there are three database logos: PostgreSQL, MongoDB, and MySQL. The MongoDB logo is highlighted with a red rectangular box.

PowerSync Service

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

The screenshot shows the 'Welcome to PowerSync' interface with the 'Connections' tab selected. The form is titled 'Next, provide your MongoDB database connection details and test your connection settings:'. A tooltip explains that the 'Name' field is used to identify the connection. The form fields are: 'Name' (Default), 'URI' (empty), 'Database name' (empty), 'Username' (empty), 'Password' (empty), and 'Post Images' (Automatic (Recommended)). A 'Test connection' button is at the bottom right.

- g. In the Post Images setting, users can choose from three options:
- Automatic (Recommended): Automatically configures the `changeStreamPreAndPostImages` option on collections as needed.
 - 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.
 - 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](#).

PowerSync Service

- 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:



Welcome to PowerSync

Let's get started!

Changes are being deployed to your instance 'Testing'.

This can take a minute or two.

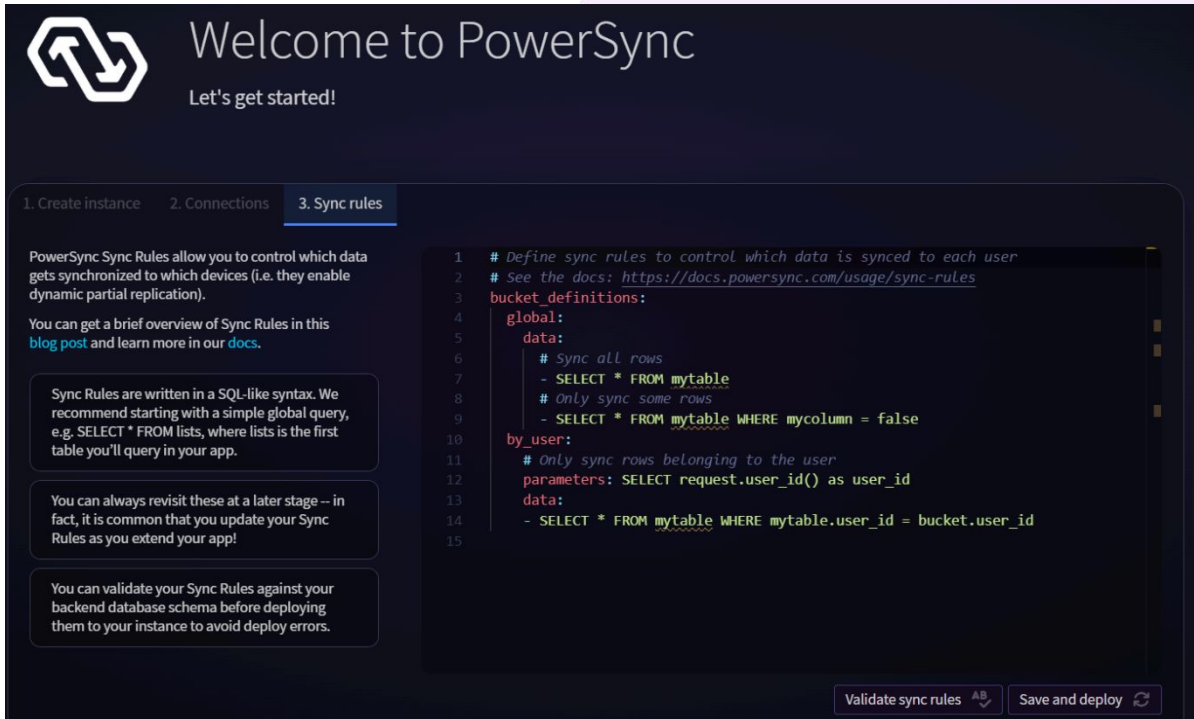
While you wait, check out some of the workspaces above.

Pro tips:

- The dashboard is built to be super interactive. You can right-click on almost anything.
- There is a command palette you can access by double-pressing the Shift key!
- Running into issues? Join our [community Discord](#) where we are ready to assist.

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:



The screenshot shows the PowerSync 'Welcome to PowerSync' interface. The '3. Sync rules' tab is active. On the left, there are three informational boxes: one explaining that sync rules control data synchronization, another stating that rules are written in SQL-like syntax, and a third about validating rules against a backend schema. The main area displays a code editor with a sample sync rule configuration. At the bottom right, there are buttons for 'Validate sync rules' and 'Save and deploy'.

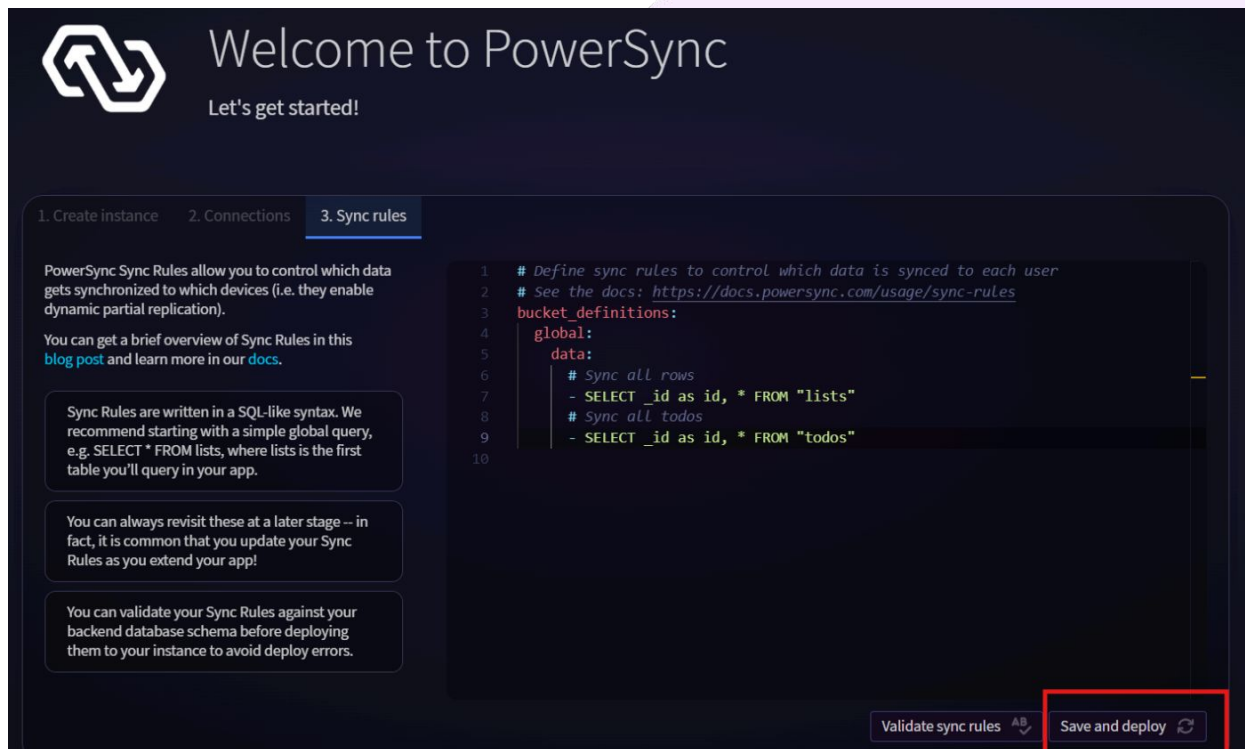
```
1 # Define sync rules to control which data is synced to each user
2 # See the docs: https://docs.powersync.com/usage/sync-rules
3 bucket_definitions:
4   global:
5     data:
6       # Sync all rows
7       - SELECT * FROM mytable
8       # Only sync some rows
9       - SELECT * FROM mytable WHERE mycolumn = false
10   by_user:
11     # Only sync rows belonging to the user
12     parameters: SELECT request.user_id() as user_id
13     data:
14       - SELECT * FROM mytable WHERE mytable.user_id = bucket.user_id
15
```

- 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"
```

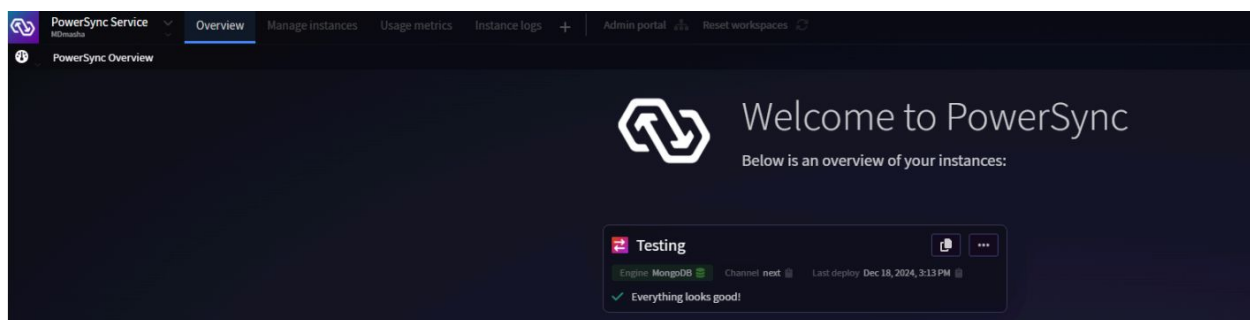
PowerSync Service


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

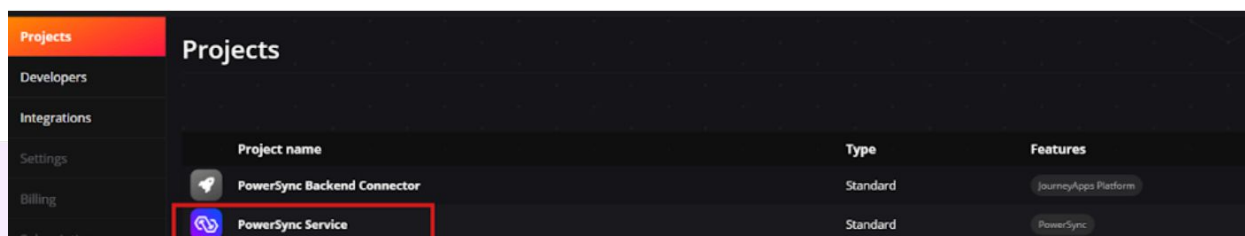


c. Modifying the sync rules will initiate a redeployment of the instance

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

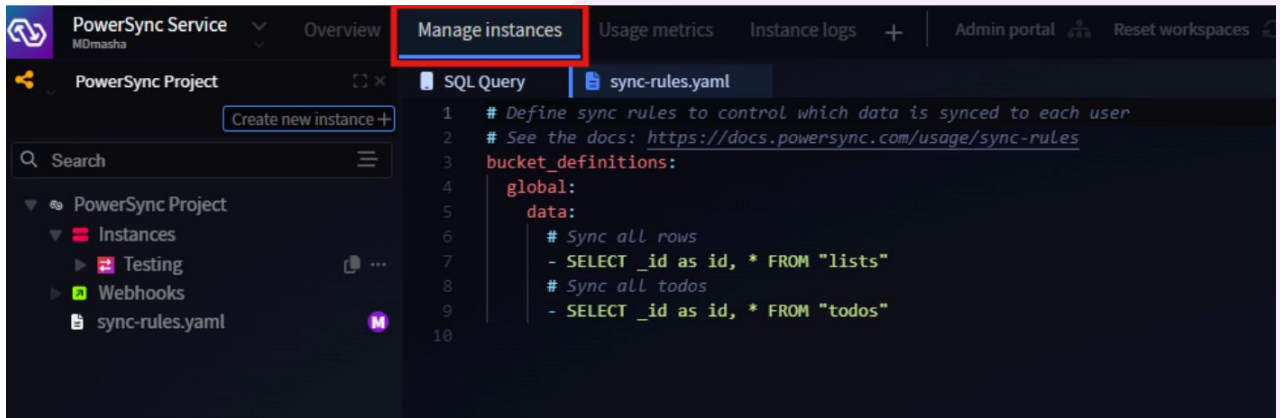


e. By clicking the icon  in the top right corner of the screen above to return to the Projects view, you will be able to see the newly created instance:



PowerSync Service

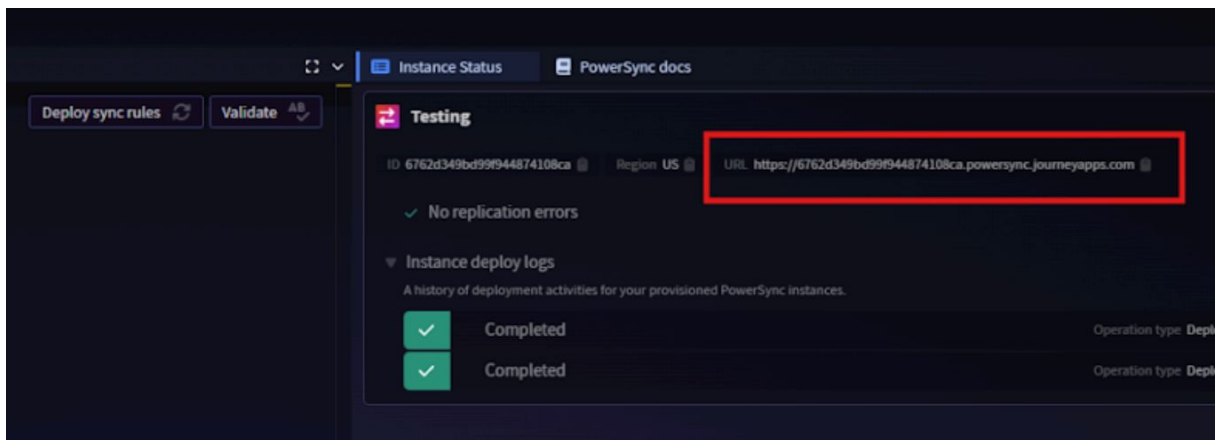
- f. To view more details, click on the project and select "Manage Instances" from the top navigation menu:



- 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](#).

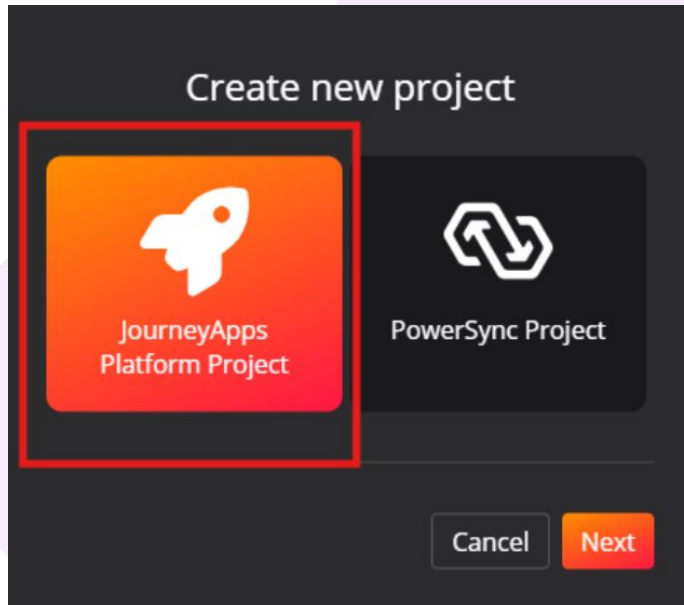
- h. In the subsequent setup steps, you will need the PowerSync Service URL and ID, which can be located in the [PowerSync Dashboard](#):



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.

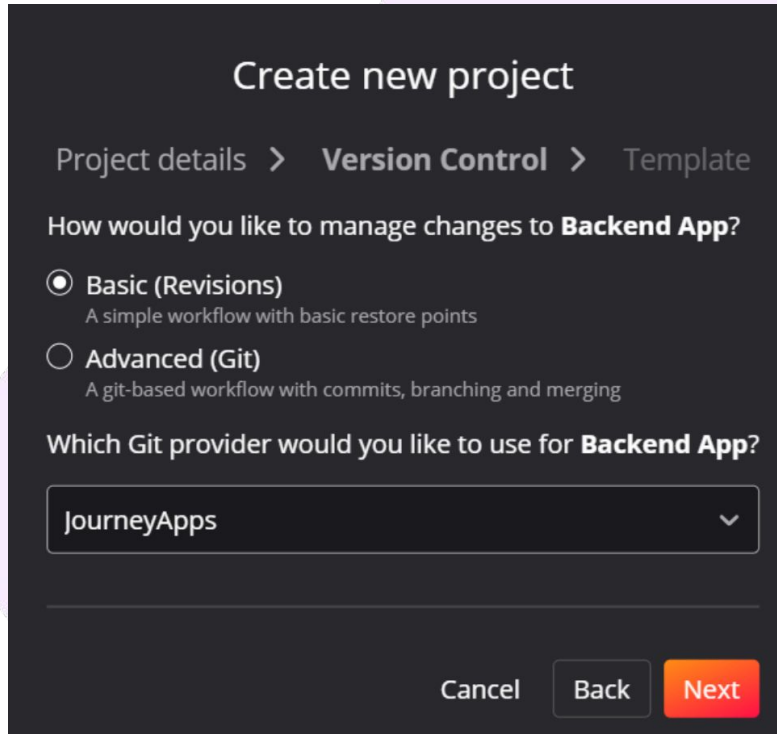


- b. Choose a name for the Backend App and select the deployment region.

A screenshot of the 'Create new project' dialog box, specifically the 'Project details' tab. The breadcrumb navigation at the top shows 'Project details > Version Control > Template'. Below this, there are two input fields. The 'Project name' field is a text input containing 'Backend App'. The 'Region' field is a dropdown menu currently showing 'United States' with a downward arrow. At the bottom right, there are two buttons: a grey 'Cancel' button and an orange 'Next' button.

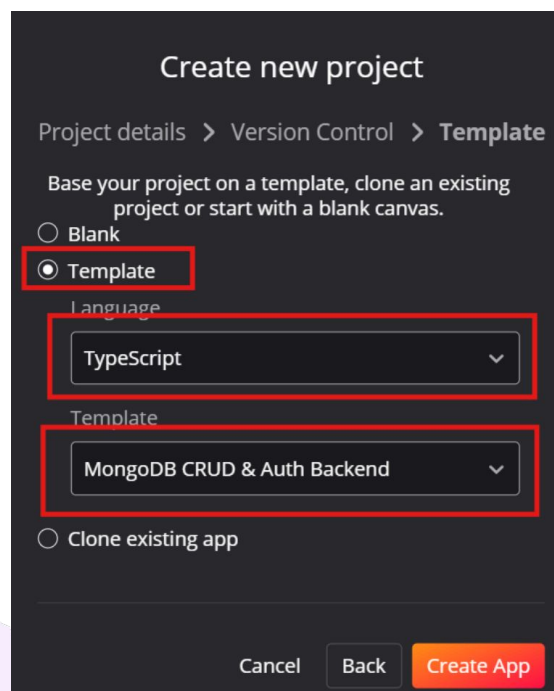
Backend App Setup

- 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:



The screenshot shows the 'Create new project' dialog with the 'Version Control' tab selected. The breadcrumb trail is 'Project details > Version Control > Template'. The question is 'How would you like to manage changes to Backend App?'. There are two radio button options: 'Basic (Revisions)' which is selected, and 'Advanced (Git)'. Below 'Basic (Revisions)' is the text 'A simple workflow with basic restore points'. Below 'Advanced (Git)' is the text 'A git-based workflow with commits, branching and merging'. The next question is 'Which Git provider would you like to use for Backend App?'. A dropdown menu shows 'JourneyApps' with a downward arrow. At the bottom are three buttons: 'Cancel', 'Back', and 'Next' (highlighted in orange).

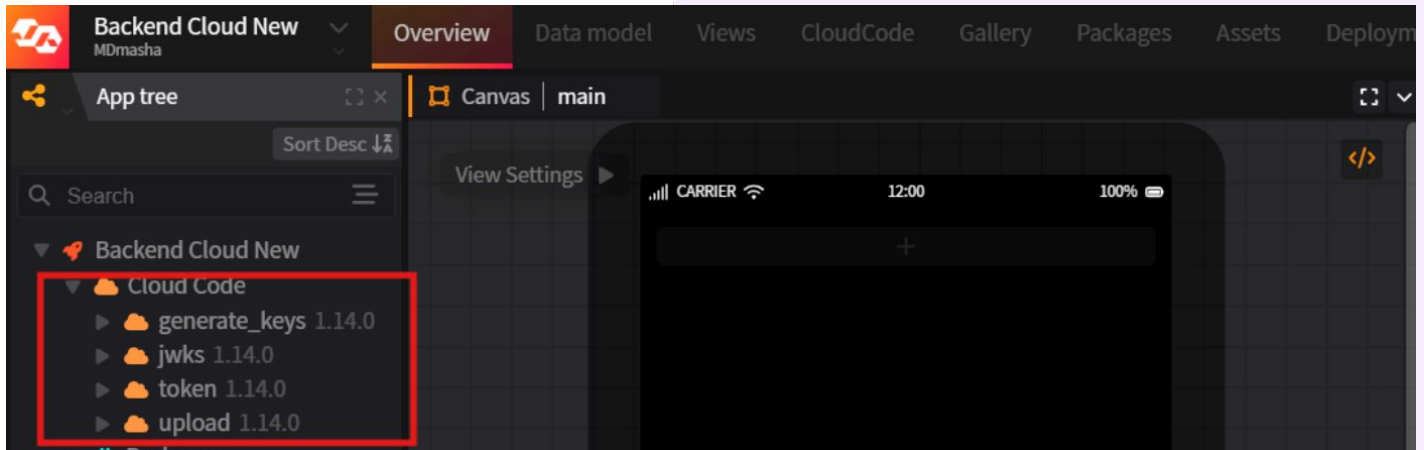
- d. Finally, choose the "Template" option. Within this section, set the "Language" to TypeScript and select the template titled "MongoDB CRUD & Auth Backend":



The screenshot shows the 'Create new project' dialog with the 'Template' tab selected. The breadcrumb trail is 'Project details > Version Control > Template'. The text says 'Base your project on a template, clone an existing project or start with a blank canvas.' There are three radio button options: 'Blank', 'Template' (which is selected and highlighted with a red box), and 'Clone existing app'. Below the 'Template' option is a 'Language' dropdown menu with 'TypeScript' selected (highlighted with a red box). Below that is a 'Template' dropdown menu with 'MongoDB CRUD & Auth Backend' selected (highlighted with a red box). At the bottom are three buttons: 'Cancel', 'Back', and 'Create App' (highlighted in orange).

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.

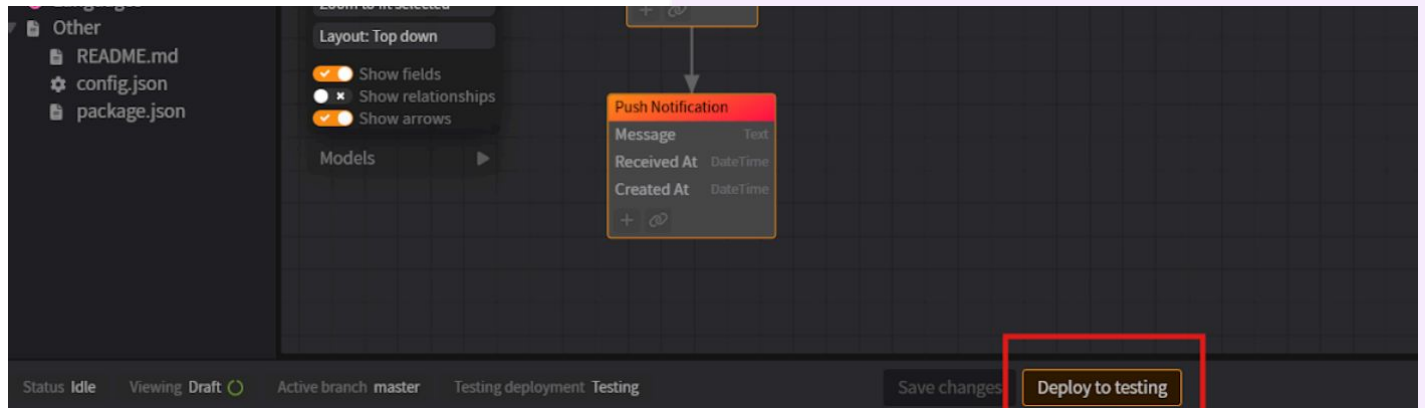
Backend App Setup

2. Generate Key Pair

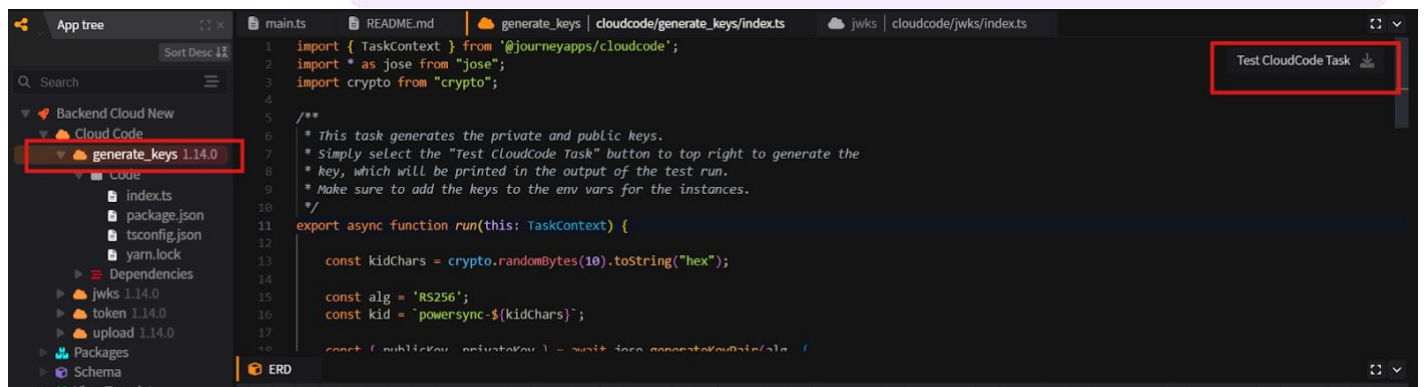
Before using the serverless functions you need to generate a public/private key pair.

Follow the below steps to generate the key pair:

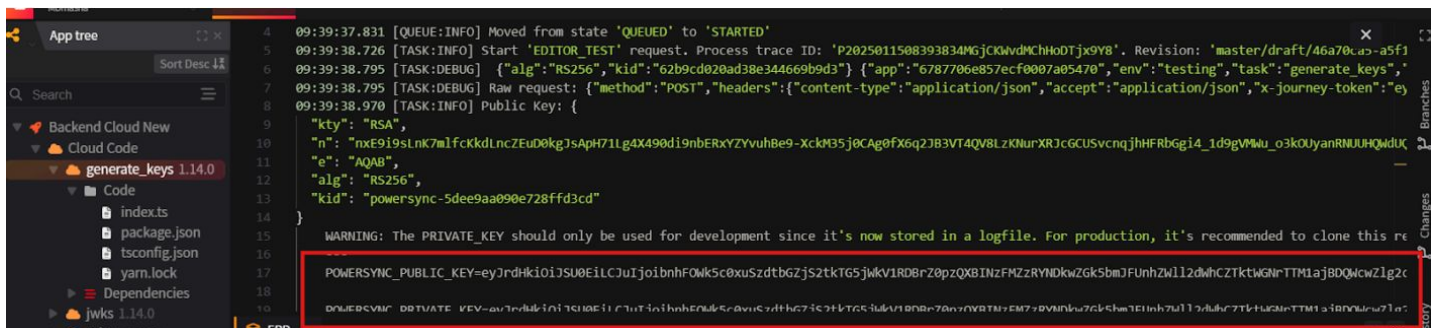
- Initially, click on the Deploy to Testing button. For the time being, there's no requirement to set up any environment variables for the deployment.



- Once the Deploy has succeeded, open the generate_keys CloudCode task.
- Click on the Test CloudCode Task button at the top right.



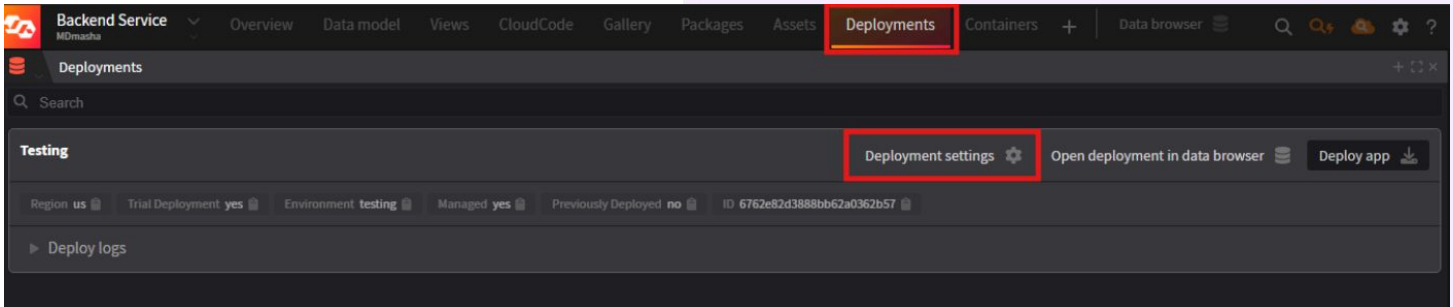
- This will print the public and private key in the task logs window.



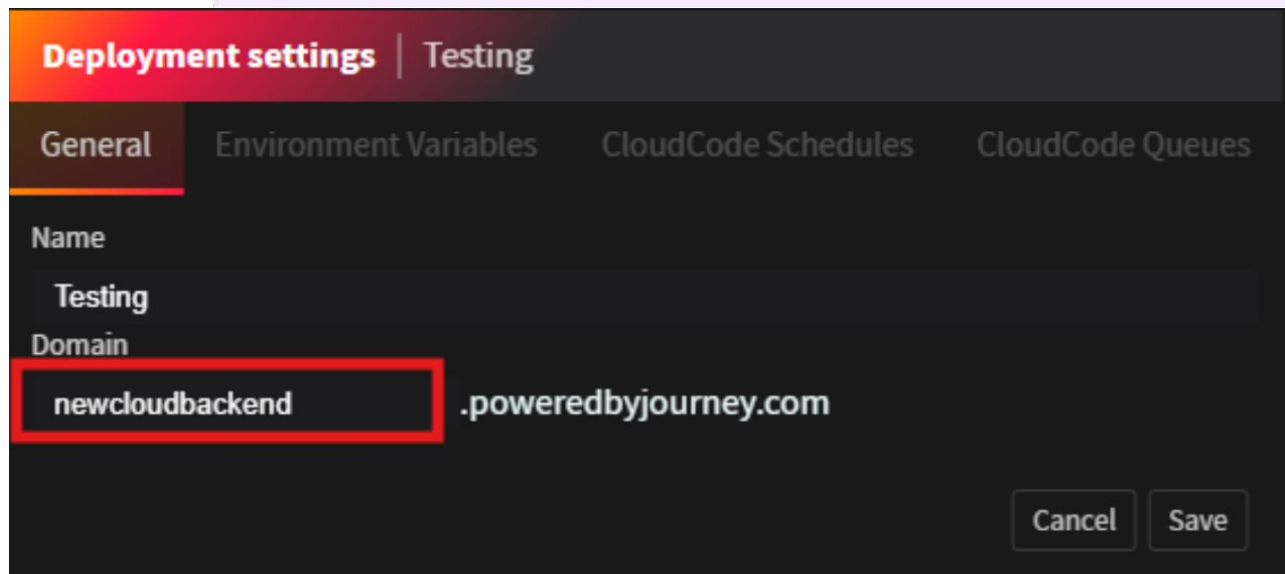
- Copy and paste the POWERSYNC_PUBLIC_KEY and POWERSYNC_PRIVATE_KEY to a file — we'll need this in the next step.

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:

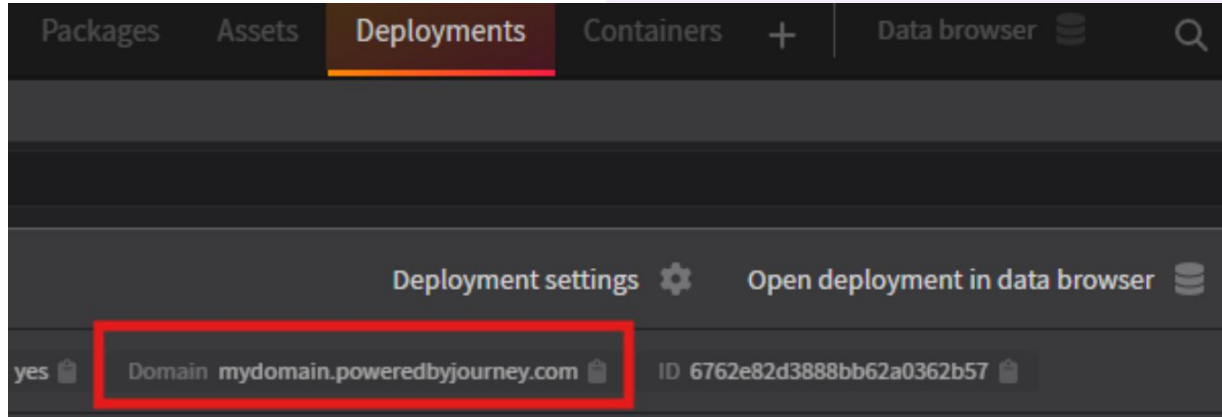
```
mongodb+srv://<username>:<password>@<deployment>.mongodb.net/<database_name>
```


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.

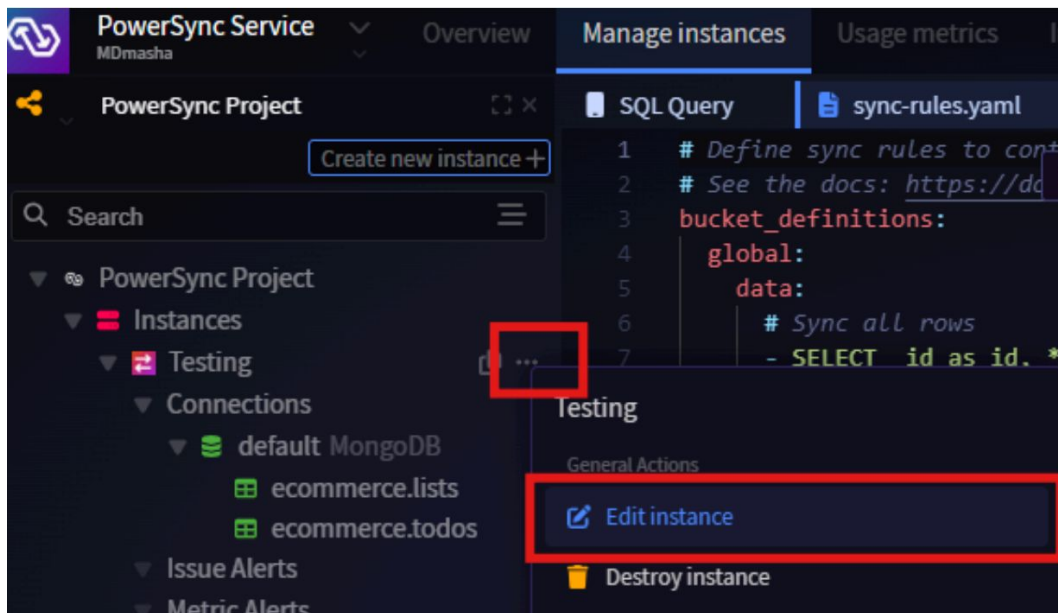
Backend App Setup

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:

Edit Instance: Testing

General DB Connections **Client Auth**

☐ Use Supabase Auth
PowerSync will use the same JWT secret as Supabase.

Supabase JWT Secret
Used to verify Supabase JWTs. Get it from your project's API settings in the Supabase Dashboard.

☒ Enable development tokens
Allow PowerSync to generate temporary development tokens.

JWKS URI (optional)
Keys returned by this URI will be trusted for JWT authentication.
`https://mydomain.poweredbyjourney.com/jwks`

JWT Audience (optional)
Additional values accepted for the "aud" field of JWTs.
+

HS256 authentication tokens (ADVANCED)
Additional HS256 tokens used to authenticate JWTs.
+

Cancel Save and deploy

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

Manage instances Usage metrics Instance logs + Admin portal Reset workspaces

SQL Query sync-rules.yaml Instance Status PowerSync docs

1 # Define sync rules to control which data is synced to each user
2 # See the docs: https://docs.mongodb.com/atlas/tutorial/sync-rules/
3 bucket_definitions:
4 global:
5 data:
6 # Sync all rows
7 - SELECT _id as id, * FROM "lists"
8 # Sync all todos
9 - SELECT _id as id, * FROM "todos"
10

Deploy sync rules Validate

Testing

ID 6762d349bd99f944874108ca Region US
URL https://6762d349bd99f944874108ca.powersync.journeyapps.com

✓ No replication errors

▼ Instance deploy logs
A history of deployment activities for your provisioned PowerSync instances.

Running	Operation type: Deploy (manual) Started at: Dec 18, 2024, 5:10 PM
Completed	Operation type: Deploy (manual) Started at: Dec 18, 2024, 3:10 PM Duration: 2m48s
Completed	Operation type: Deploy (manual) Started at: Dec 18, 2024, 3:02 PM Duration: 2m39s

Self-Hosted Backend Deployment Setup

1. Clone the [Backend Demo](#):

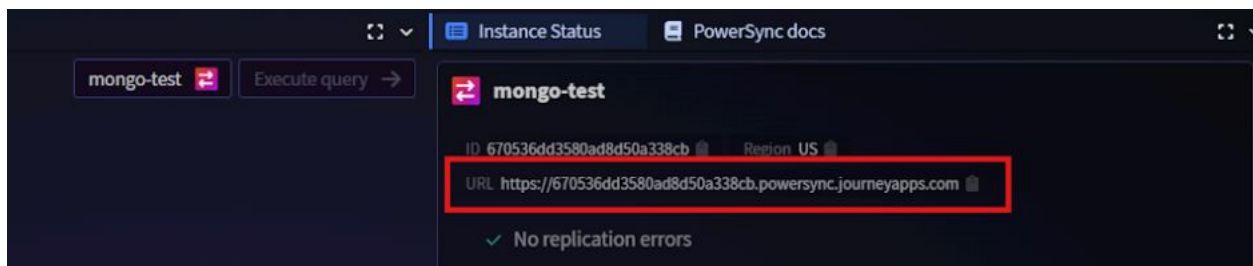
This will clone a Node.js custom backend demo/example provided by PowerSync:

```
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:

- Use NGROK to expose your local server.
- Download and install [ngrok](#). If you don't have an account, you'll need to create a new one.
- 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:

- 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
```

- The Terminal will display the following message:

```
ngrok (Ctrl+C to quit)
♦ Found a bug? Let us know: https://github.com/ngrok/ngrok

Session Status      online
Account             dorottya.nyarady@mongodb.com (Plan: Free)
Version             3.18.4
Region              Europe (eu)
Latency              91ms
Web Interface        http://127.0.0.1:4040
Forwarding            https://e3f4-2a09-bac0-1000-41d-00-59-2b.ngrok-free.app -> http://localhost:6060

Connections          ttl    opn    rt1    rt5    p50    p90
                    50     0      0.01   0.01   6.02   6.04

HTTP Requests
-----
```

Self-Hosted Backend Deployment Setup

6. Link with PowerSync:

- 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.

The image shows two screenshots from the PowerSync Service dashboard. The top screenshot shows the 'Manage instances' tab with a list of instances. A red box highlights the 'Testing' section, and another red box highlights the 'Edit instance' button. The bottom screenshot shows the 'Edit Instance: mongo-test' form, specifically the 'Client Auth' tab. A red box highlights the 'Enable development tokens' section, which includes a checkbox, a description, and a text input field containing the URL `https://e3f4-2a09-bac0-1000-41d-00-59-2b.ngrok-free.app/api/auth/keys`. The 'Save and deploy' button is visible at the bottom right of the form.

PowerSync Service Overview Manage instances Usage metrics

PowerSync Project

Create new instance +

Search

PowerSync Project

- Instances
- Testing
- Connections
 - default MongoDB
 - ecommerce.lists
 - ecommerce.todos
- Issue Alerts
- Metric Alerts

SQL Query

```
1 # Define sync rules to con+
2 # See the docs: https://dc
3 bucket_definitions:
4   global:
5     data:
6       # Sync all rows
7       - SELECT id as id, *
```

Testing

General Actions

Edit instance

Destroy instance

Edit Instance: mongo-test

General DB Connections Client Auth

☐ Use Supabase Auth

PowerSync will use the same JWT secret as Supabase.

Supabase JWT Secret

Used to verify Supabase JWTs. Get it from your project's API settings in the Supabase Dashboard.

.....

☒ Enable development tokens

Allow PowerSync to generate temporary development tokens.

JWKS URI (optional)

Keys returned by this URI will be trusted for JWT authentication.

`https://e3f4-2a09-bac0-1000-41d-00-59-2b.ngrok-free.app/api/auth/keys`

JWT Audience (optional)

Additional values accepted for the "aud" field of JWTs.

+

HS256 authentication tokens (ADVANCED)

Additional HS256 tokens used to authenticate JWTs.

+

Cancel Save and deploy

- 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`: Set this to the URL of the Backend App. Ensure it matches the JWKS URI, excluding the `/jwks` segment.
 - 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/src/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:

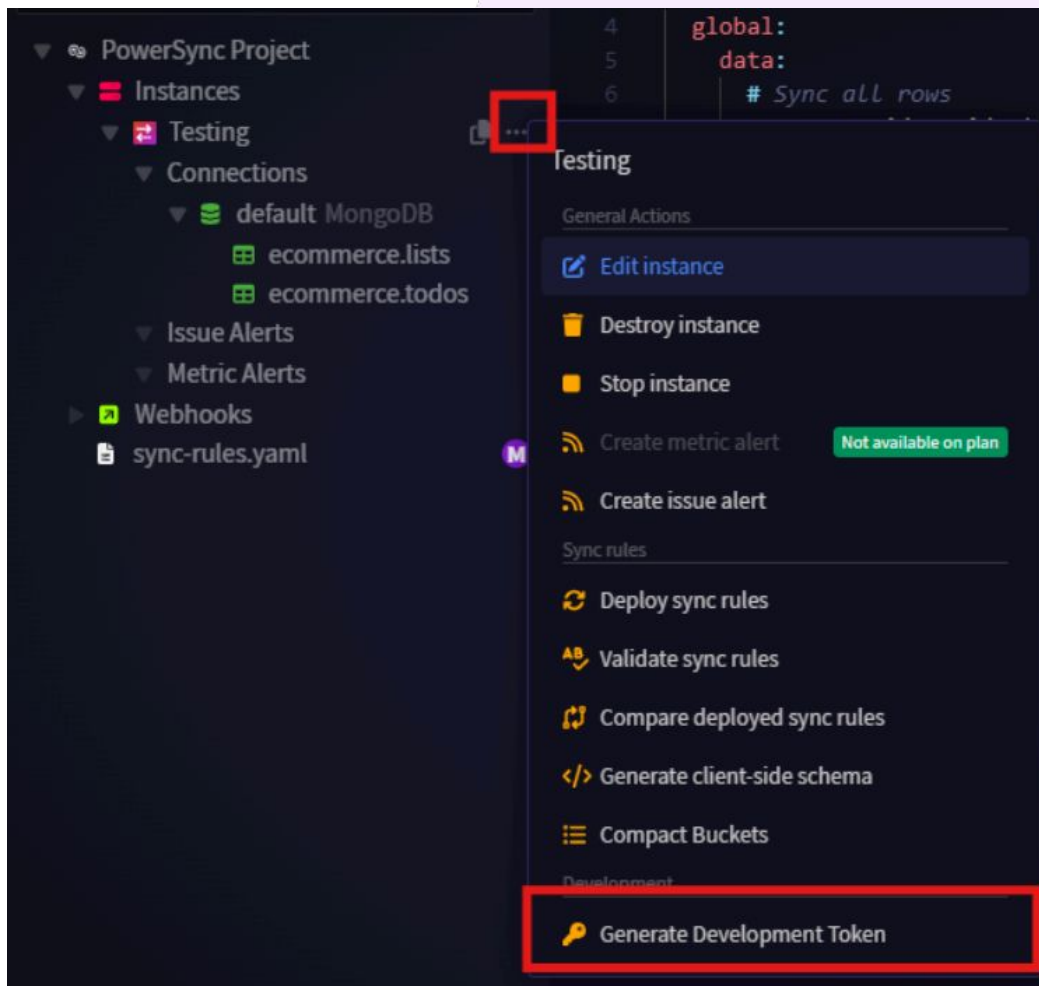
```
Unset  
pnpm install  
  
pnpm start
```

- 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:


The screenshot shows the MongoDB Compass interface for the 'ecommerce.lists' collection. The top bar indicates 'STORAGE SIZE: 36KB', 'LOGICAL DATA SIZE: 540B', 'TOTAL DOCUMENTS: 4', and 'INDEXES TOTAL SIZE: 36KB'. Below this, there are tabs for 'Find', 'Indexes', 'Schema', 'Anti-Patterns', 'Aggregation', and 'Search Indexes'. A search bar contains the query: 'Type a query: { field: 'value' }'. The 'QUERY RESULTS: 1-4 OF 4' section displays three document snippets. The first two documents have an 'owner_id' field with a value like '73f3672d-f57a-4838-adbc-3f7e80cabb55'. The third document has an 'owner_id' field with a value like '73f3672d-f57a-4838-adbc-3f7e80cabb55'. A 'Screenshot' dialog box is overlaid on the right, showing a code editor with a query: 'SELECT _id as list_id FROM lists where owner_id = request.'. Below the code editor, there is a 'Token subject / User ID' field with the value '73f3672d-f57a-4838-adbc-3f7e80cabb55' and 'Continue' and 'Cancel' buttons.

-
-
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

The screenshot shows the 'Diagnostics Config' form in the 'diagnostics-app.powersync.com' browser. The form has a dark background with the 'PowerSync' logo at the top. Below the logo, there are two input fields: 'PowerSync Token' and 'PowerSync Endpoint'. The 'PowerSync Token' field contains the value '.99Eby5tNAZ4nNdh3lBehjrULTgFgJ6FqcVvR6fe5bGfTygxOb2bD0l9sqkK0UNj6d'. The 'PowerSync Endpoint' field contains the value 'https://670536dd3580ad8d50a338cb.powersync.journeyapps.com'. A 'PROCEED' button is located at the bottom right of the form.

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:

PowerSync

Sync Overview

Dynamic Schema

SQL Console

Client Parameters

Sign Out

Sync Diagnostics

Number of buckets	Total Rows	Total Operations	Total Data Size	Total Metadata Size	Total Downloaded Size	Last Synced At
1	5	7	827 Bytes	806 Bytes	2.29 KiB	5:53:43 PM

CLEAR & REDOWNLOAD

Tables

Name	Row Count	Data Size
lists	4	605 Bytes
todos	1	222 Bytes

Rows per page: 10 1-2 of 2

Buckets

Name	Table(s)	Row Count	Total Oper...	Data Size	Metadata Size	Downloaded Size	Status
global[]	lists, todos	5	7	827 Bytes	806 Bytes	2.29 KiB	Ready

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.