Step-by-Step Guide - ProcureHub

This guide explains how to set up and deploy the ProcureHub project on Railway.

---

1) Prerequisites

Railway uses GitHub integration to deploy services. Make sure both the backend and frontend code are available in separate repositories on GitHub.

- Confirm both repositories are visible in your GitHub account
- You can keep the repositories private – Railway will still be able to access them after authorization

You need a remote PostgreSQL database provider like Supabase for production use.

- Create an account at Supabase (or another provider)
- Set up a new project and database
- Keep your database credentials ready (host, port, name, user, password)

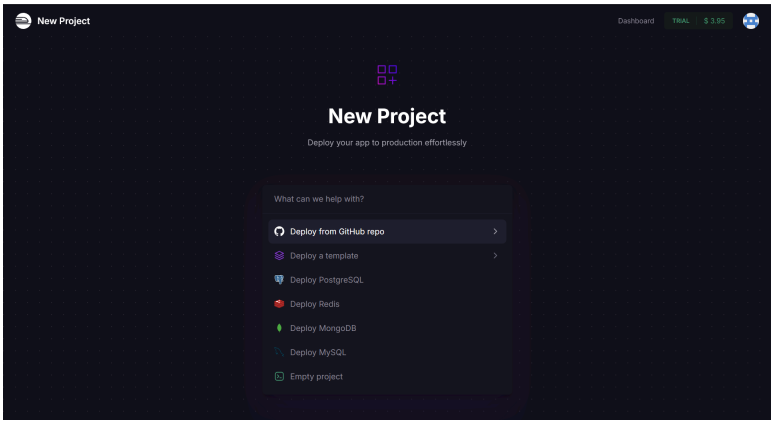
---

2) Create a Railway Account and Project

Railway allows you to manage multiple services (frontend, backend, Redis, etc.) in one project.

Instructions:

- Go to: https://railway.app
- Sign up or log in using your GitHub account
- Click **"+ New"** → **"Deploy from GitHub repo"**



---

3) Deploy the Backend Service

Instructions:

- Choose the `procure-hub-be` repository when prompted
- Railway will auto-detect it as a Node.js project and install dependencies
- Leave the default build and start commands
- Railway will automatically detect all variables from your `.env.example` file and ask you to populate the values
    - `FRONTEND_URL` - you will generate this value in step 4
    - `REDIS_URL` - you will get this value in step 5
- After deployment go to **"Settings"** → **"Networking"** and under public networking generate a new domain. The generated domain will be used as an environment variable value in the frontend service

---

4) Deploy the Frontend Service

Instructions:

- Create a new service inside the same project **"+ Create"** → **"GitHub repo"**
- Choose the `procure-hub-fe` repository when prompted
- Railway will auto-detect it as a Node.js project and install dependencies
- Leave the default build and start commands
- Railway will automatically detect all variables from your `.env.example` file and ask you to populate the values:
    - `VITE_API_URL=https://<your-backend-domain>`
    - `VITE_WEBSOCKET_URL=ws://<your-backend-domain>`

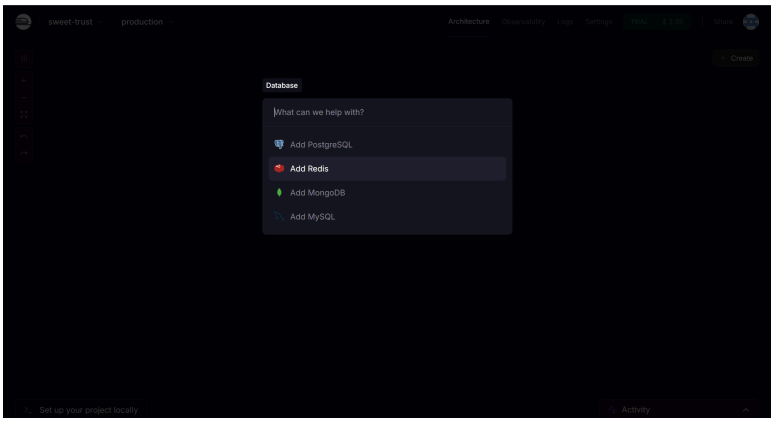
- After deployment go to **"Settings"** → **"Networking"** and under public networking generate a new domain
- Go back to your backend service and add the value for the environment variable `FRONTEND_URL`

---

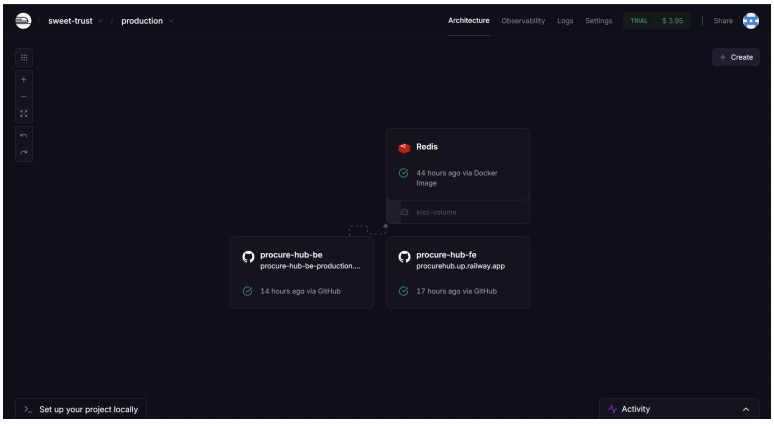
5) Add Redis Service

Instructions:

- Create a new service inside the project **"+ Create"** → **"Database"** → **“Add Redis”**
- In the **“Variables”** tab find `REDIS_URL` and copy the connection string
- Go to the backend service and under the **“Variables”** find the variable `REDIS_URL` and paste



This is how your project dashboard should look like after following these steps:



---

5) Run Migrations and Add Admin User

To prepare your database in the cloud, you'll need to run Sequelize migrations and optionally create an admin account. This is done by accessing the backend service using the Railway CLI.

Instructions:

- Install **Railway CLI** (if not already installed) https://docs.railway.com/guides/cli
    - For example, using nmp, you can install with `npm i -g @railway/cli`
- Login to Railway in your terminal: `railway login`
- Link your local terminal to the deployed backend service: `railway link`
- Run all migrations: `railway run npx sequelize-cli db:migrate`
- Run seeders: `railway run npx sequelize-cli db:seed:all`
- To create the admin user run this command and follow instructions:
  `railway npm run add-admin`

---

6) Open the App in Your Browser

Open the deployed frontend URL in your browser.

You can now:

- Register as a buyer or seller
- Log in with your admin credentials

Final Notes

- Redis, backend, and frontend should **all exist within the same Railway project** for best performance and environment management
- Use **Railway variables** to securely manage credentials and config