

 Store with IPFS – Decentralized File Upload

**Objective/Aim:**  
  
 To develop a secure, full-stack web application that allows users to upload files to the InterPlanetary File System (IPFS) through a user-friendly interface. The system uses a private backend to manage secret API keys, ensuring that credentials are never exposed on the client-side.

**Apparatus/Software Used:**

* Laptop/PC
* Pinata
* Pinata docs for reserch

**Theory/Concept:**

**1. How IPFS Changes File Storage**

Think of regular cloud storage (like Google Drive) as a street address. To get a file, you have to go to a specific server location, like drive.google.com/folder/myfile.txt. If that server is down, you can't get your file.

This project uses the **InterPlanetary File System (IPFS)**, which works differently. Instead of asking *where* a file is, IPFS asks *what* the file is.

* When you upload a file, IPFS gives it a unique fingerprint based on its content, called a **Content Identifier (CID)**. It's like a book's ISBN number—no two are alike.
* To get the file, you just ask the network for that "fingerprint." The network finds it from any computer that has a copy, not just one central server.
* This means there's no single point of failure. As long as someone on the network has the file, it's available.



### 2. Why We Use Pinata (The Pinning Service)

Imagine you share a file on the IPFS network from your laptop. If you shut your laptop down, that file disappears from the network. For a file to stay available, someone has to be online to share it.

This is where **Pinata** comes in. Think of Pinata as a friendly service that promises to keep a copy of your file online 24/7. When you "pin" a file with Pinata, you're asking their powerful, always-on computers to hold onto your file and share it with the IPFS network, so you don't have to.

### 3. Keeping Your Keys Safe (The App's Architecture)

Your Pinata account is protected by secret credentials (your API Keys and JWT). These are like the keys to your house—you wouldn't just leave them on the front doormat of your website for anyone to see. If you put them in your frontend code, they could be easily stolen.

To prevent this, our app is split into two parts:

* **The Frontend:** This is the beautiful React website you see and interact with. It's the public-facing "storefront" and contains no secret keys.
* **The Backend:** This is a private "back office" (a Node.js server) that you run separately. It's the only part of the system that knows your secrets because it reads them from a private .env file.

### 4. How the Frontend and Backend Talk (The "Proxy")

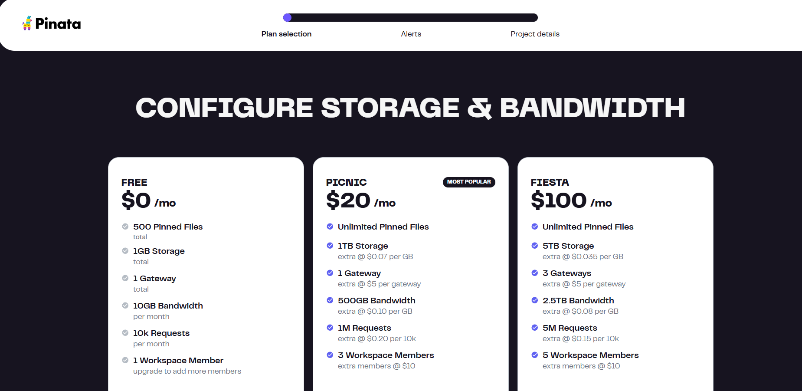
During development, your frontend and backend are like two people in different rooms trying to talk—they're running on different ports (e.g., 3000 and 3001). A web browser's security rules would normally block this communication.

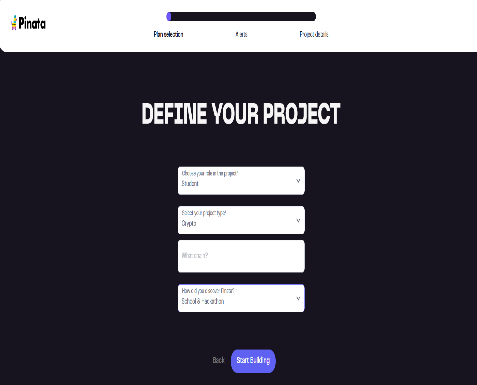
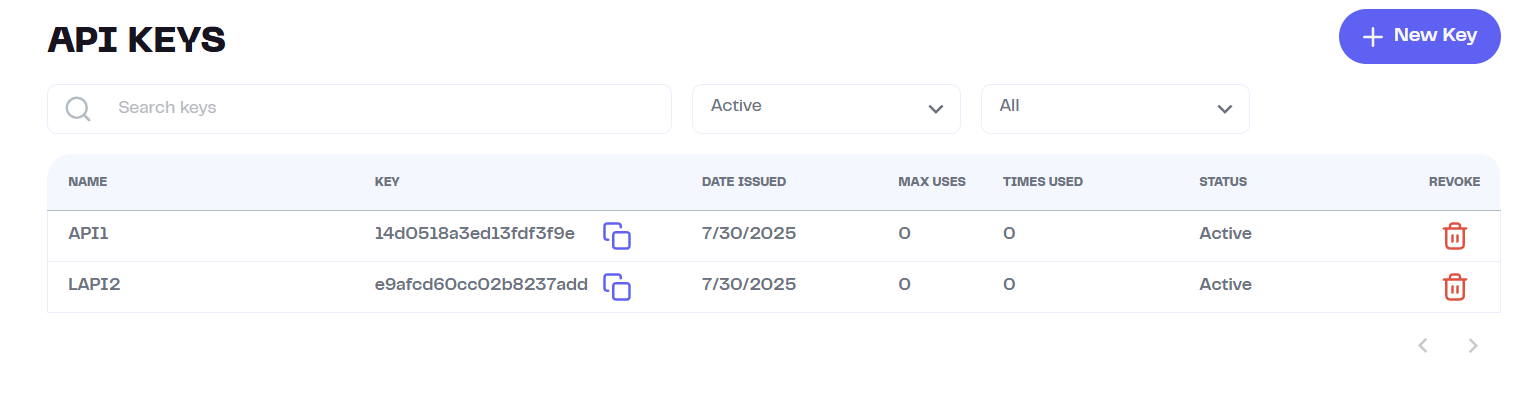
The "proxy": "http://localhost:3001" setting in your package.json acts like a helpful receptionist. It tells your frontend development server: "If you get a request you don't know how to handle (like for /api/upload), just forward the call to the back office at port 3001."

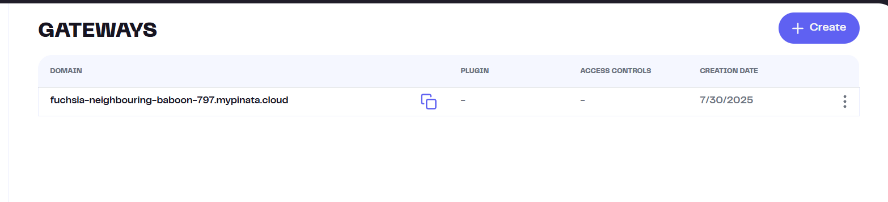
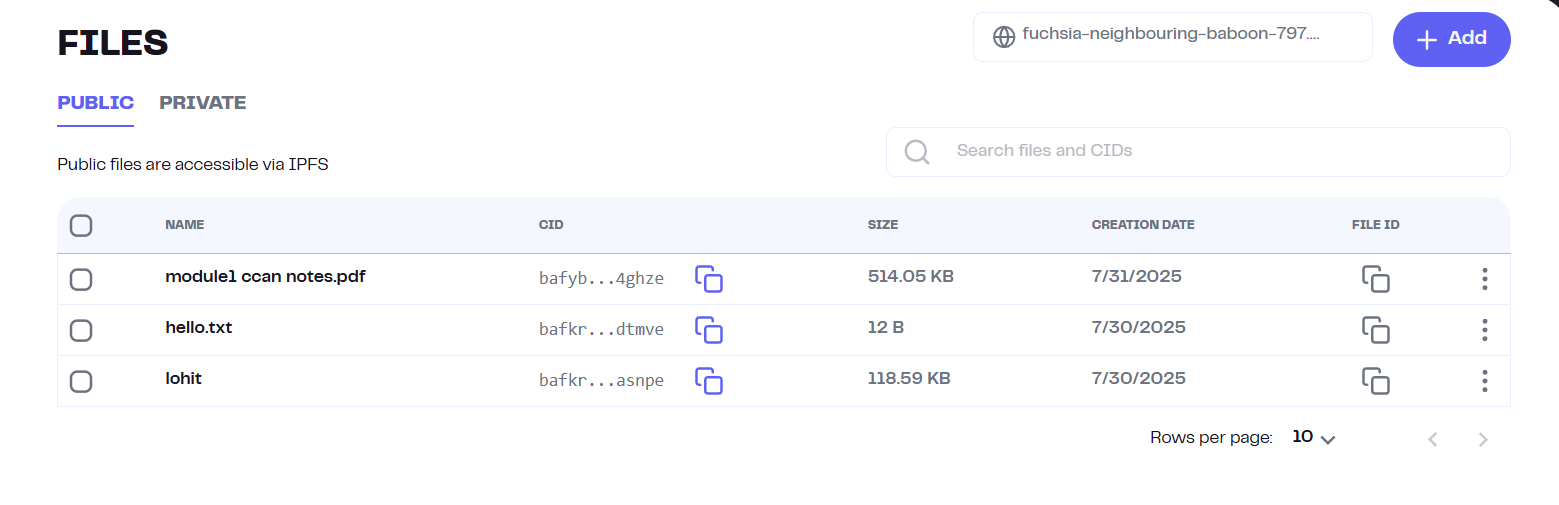
This creates a smooth and secure workflow:

1. **Upload:** You pick a file in the UI. The frontend sends it to the "receptionist," who forwards it to the backend. The backend takes the file, securely adds your secret key, and sends it to Pinata.
2. **View Files:** The frontend asks the "receptionist" to see all the files. The call gets forwarded to the backend, which securely fetches the list from Pinata and sends it back to the frontend to be displayed.

**Procedure:**

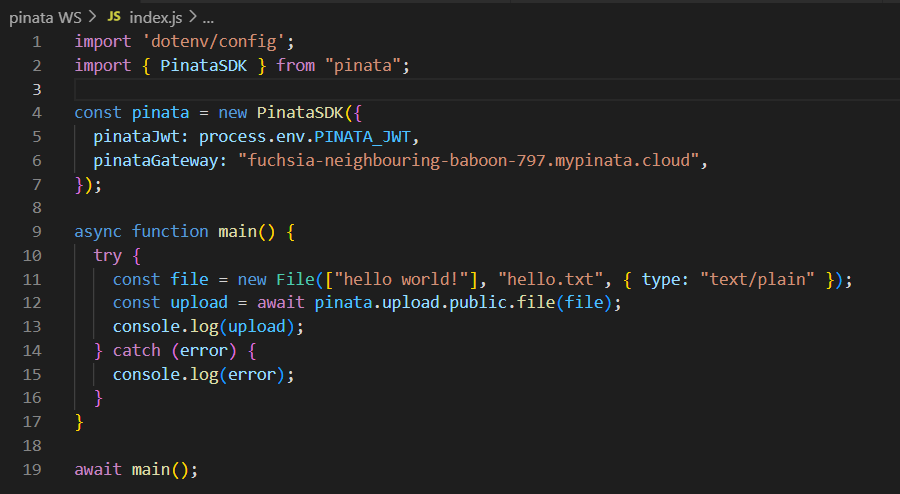


**Initial Setup & Manual Upload:** First, create a Pinata account and define your project details. You can test the connection by writing a simple Node.js script to upload a file directly, which confirms your environment is working.

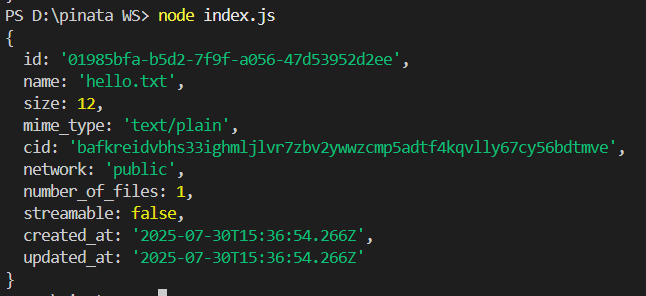
**Generate API Credentials:** Navigate to the "API Keys" section in your Pinata dashboard. Create a new key with Admin permissions to generate the JWT (JSON Web Token) needed for authentication.

**Create a Custom Gateway:** Go to the "Gateways" tab and create a new gateway. This provides a dedicated, high-speed URL for accessing your pinned files from the IPFS network.

**Develop the Secure Backend:** Create a Node.js server using the Express framework. Store your secret Pinata JWT in a .env file to keep it secure and hidden from the public.



**Build the Frontend Interface:** Develop the React application with two main pages: one for uploading new files and another for viewing all the files you've already pinned.



**Connect and Run:** Configure the frontend's package.json with a proxy to the backend server. Run both the frontend and backend servers simultaneously to launch the full, secure application.

**Observation Table:**

Looking at the final result, we can see how all the parts come together. The clean React website lets you upload and view files easily. Behind the scenes, the Node.js script shown in the terminal is doing the heavy lifting, securely

talking to Pinata using the secret keys. Finally, the Pinata dashboard confirms our success, showing the newly

uploaded files and the active gateway ready to serve them to the world.