Upload to Pinata Script

`uploadToPinata.js` is a Node.js script designed to automate the uploading of files (such as images or metadata JSON files) to Pinata, a service that pins files to IPFS (InterPlanetary File System). This script reads files from a specified directory and uploads each to Pinata, retrieving the IPFS hash for each file.

Table of Contents

Features

- **Automated Upload**: Uploads multiple files to Pinata in one run.
- **IPFS Integration**: Utilizes Pinata to pin files to IPFS, ensuring decentralized storage.
- **Error Handling**: Logs errors if uploads fail, facilitating debugging.
- **Scalable**: Easily handles large numbers of files.
- **Flexible**: Can upload any file type, making it versatile for various use cases.

Prerequisites

- **Node.js**: Ensure you have Node.js installed. Download it from nodejs.org.

- **npm**: Node.js typically comes with npm. Verify installation with:

  ```
  npm -v
  ```

Installation

1. **Clone the Repository**

   ```
   git clone https://github.com/yourusername/your-repo.git
   cd your-repo
   ```

2. **Initialize `package.json`**

   Initialize a `package.json` file to manage your project's metadata and dependencies.

```
   npm init -y
   ```

   This command creates a `package.json` file with default settings.

3. **Install Dependencies**

   This script requires external packages: `axios` and `form-data`.

   ```
   npm install axios form-data
   ```

4. **(Optional) Install `dotenv` for Environment Variables**

   To enhance security by managing API credentials through environment variables, install the `dotenv` package.

   ```
   npm install dotenv
   ```

Configuration

API Credentials

To interact with Pinata's API, you need to provide your Pinata API Key and Secret API Key.

1. **Obtain Pinata API Keys**

   - Sign up for a Pinata account at pinata.cloud.
   - Navigate to your Pinata dashboard to obtain your API Key and Secret API Key.

2. **Secure Your API Keys**

   Security Best Practice: It's highly recommended to store your API credentials in environment variables or a `.env` file instead of hardcoding them into your scripts.

   - **Using Environment Variables:**

     ```
     export PINATA_API_KEY='your_pinata_api_key'
     export PINATA_SECRET_API_KEY='your_pinata_secret_api_key'
     ```

   - **Using a `.env` File:**

     Create a `.env` file in your project root:

     ```
     PINATA_API_KEY=your_pinata_api_key
     PINATA_SECRET_API_KEY=your_pinata_secret_api_key
     ```

     Update `uploadToPinata.js` to load environment variables:

     ```
     require('dotenv').config();

     const PINATA_API_KEY = process.env.PINATA_API_KEY;
     const PINATA_SECRET_API_KEY = process.env.PINATA_SECRET_API_KEY;
     ```

Ensure that `.env` is added to your `.gitignore` to prevent accidental commits:

```
.env
```

Input Files

1. **Images Directory (`images/`)**

   - **Path**: Located in the root directory (`./images`).

   - **Contents**: Files you wish to upload to Pinata (e.g., images, metadata JSON files).

   - **Format**: Can include any file type, such as `.png` , `.jpg` , `.json` , etc.

   - *Example Directory Structure:**

   ```
   your-repo/
   ├── uploadToPinata.js
   ├── images/
   │   ├── art_0001.png
   │   ├── art_0002.png
   │   └── metadata_0001.json
   ├── package.json
   ├── .env (optional)
   └── README.md
   ```

Directory Structure

Ensure your project directory has the following structure:

```
your-repo/
├── uploadToPinata.js
├── images/
│   ├── art_0001.png
│   ├── art_0002.png
│   └── metadata_0001.json
├── package.json
├── .env (optional)
└── README.md
```

Usage

1. **Prepare Files for Upload**

   - Place all files you wish to upload into the `images/` directory.
   - Ensure that each file is correctly named and formatted as needed.

2. **Configure API Credentials**

   - **If Using Environment Variables:**

     - Ensure your `.env` file contains your Pinata API credentials.

   - **If Hardcoding API Keys:**
      - Open `uploadToPinata.js`.
      - Replace `'YOUR_PINATA_API_KEY'` and `'YOUR_PINATA_SECRET_API_KEY'` with your actual Pinata API credentials.
   - *Note:** Hardcoding API keys is not recommended due to security risks.

3. **Run the Script**

   Execute the script using Node.js:

   ```
   node uploadToPinata.js
   ```

   Output:

   - The script will upload each file in the `images/` directory to Pinata.
   - For each successful upload, it will log the response containing the IPFS hash.

4. **Sample Output**

   ```
   Uploaded art_0001.png: { IpfsHash: 'QmX...', ... }
   Uploaded art_0002.png: { IpfsHash: 'QmY...', ... }
   Uploaded metadata_0001.json: { IpfsHash: 'QmZ...', ... }
   ```

Output

- **Uploaded Files**

  Each file in the `images/` directory is uploaded to Pinata, and the response includes the IPFS hash. You can use these hashes to reference your files on IPFS.

  Example Response:

  ```
  {
      "IpfsHash": "QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco",
      "PinSize": 12345,
      "Timestamp": "2023-10-13T12:34:56.789Z",
      "isDuplicate": false,
      "PinataMetadata": {
          "name": "art_0001.png",
          "keyvalues": {}
      },
      "UserCreated": true
  }
  ```

Customization

Handling Different File Types

You can modify the script to handle specific file types differently or to add additional metadata during upload.

Example: Adding Metadata Fields

```
// Function to upload a file to Pinata with metadata
async function uploadFileWithMetadata(filePath, metadata) {
    const url = 'https://api.pinata.cloud/pinning/pinFileToIPFS';
    const data = new FormData();
    data.append('file', fs.createReadStream(filePath));

    // Add metadata
    if (metadata) {
        data.append('pinataMetadata', JSON.stringify(metadata));
    }

    const config = {
        headers: {
            ...data.getHeaders(),
            pinata_api_key: PINATA_API_KEY,
            pinata_secret_api_key: PINATA_SECRET_API_KEY,
        },
    };

    try {
        const response = await axios.post(url, data, { headers: config.headers });
        console.log(`Uploaded ${path.basename(filePath)}: ${response.data.IpfsHash}`);
    } catch (error) {
        console.error(`Failed to upload ${path.basename(filePath)}:`, error.response ?
error.response.data : error);
    }
}
```

Enhancing Error Handling

You can improve error handling by adding retries or more detailed logging mechanisms to handle upload failures gracefully.

Example: Adding Retries

```
const MAX_RETRIES = 3;

async function uploadFileWithRetries(filePath, retries = 0) {
    try {
        await uploadFile(filePath);
    } catch (error) {
        if (retries < MAX_RETRIES) {
            console.warn(`Retrying upload for ${filePath} (${retries +
1}/${MAX_RETRIES})`);
            await uploadFileWithRetries(filePath, retries + 1);
        } else {
            console.error(`Failed to upload ${filePath} after ${MAX_RETRIES}
attempts.`);
        }
    }
}
```

Using Environment Variables for API Keys

To enhance security, store your Pinata API keys in environment variables.

1. **Install `dotenv` Package**

   ```
   npm install dotenv
   ```

2. **Create a `.env` File**

   ```
   PINATA_API_KEY=your_pinata_api_key
   PINATA_SECRET_API_KEY=your_pinata_secret_api_key
   ```

3. **Update `uploadToPinata.js`**

   ```
   require('dotenv').config();

   const PINATA_API_KEY = process.env.PINATA_API_KEY;
   const PINATA_SECRET_API_KEY = process.env.PINATA_SECRET_API_KEY;
   ```

4. **Add `.env` to `.gitignore`**

   Ensure your `.env` file is not committed to version control.

   ```
   .env
   ```

Troubleshooting

- **Failed to Read Images Directory**

  Ensure the `images/` directory exists and contains files to upload. Verify the script has the necessary permissions to access the directory.

- **Invalid API Credentials**

  Double-check your Pinata API key and secret. Incorrect credentials will result in authentication errors.

- **Network Issues**

  Ensure you have a stable internet connection. Network issues can cause upload failures.

- **File Formatting Errors**

  Ensure all files in the `images/` directory are correctly formatted and not corrupted.

- **Quota Limits**

  Pinata may have rate limits or storage quotas. Check your Pinata account to ensure you haven't exceeded these limits.

Contributing

Contributions are welcome! Please follow these steps:

1. **Fork the Repository**
2. **Create a Feature Branch**

   ```
   git checkout -b feature/YourFeature
   ```

3. **Commit Your Changes**

   ```
   git commit -m "Add Your Feature"
   ```

4. **Push to the Branch**

   ```
   git push origin feature/YourFeature
   ```

5. **Open a Pull Request**

   Describe your changes and submit the pull request for review.

License

This project is licensed under the MIT License.

Happy NFT Creating!

Complete `uploadToPinata.js` Code

Below is the complete `uploadToPinata.js` script. You can copy and paste it into your project.

```
const axios = require('axios');
const FormData = require('form-data');
const fs = require('fs');
const path = require('path');
require('dotenv').config(); // Only if using environment variables

const PINATA_API_KEY = process.env.PINATA_API_KEY || 'YOUR_PINATA_API_KEY';
const PINATA_SECRET_API_KEY = process.env.PINATA_SECRET_API_KEY ||
'YOUR_PINATA_SECRET_API_KEY';

async function uploadFile(filePath) {
    const url = 'https://api.pinata.cloud/pinning/pinFileToIPFS';
    const data = new FormData();
    data.append('file', fs.createReadStream(filePath));

    const config = {
        headers: {
            ...data.getHeaders(),
            pinata_api_key: PINATA_API_KEY,
            pinata_secret_api_key: PINATA_SECRET_API_KEY,
        },
    };

    try {
```

```
        const response = await axios.post(url, data, config);
        console.log(`Uploaded ${path.basename(filePath)}: ${response.data.IpfsHash}`);
    } catch (error) {
        console.error(`Failed to upload ${path.basename(filePath)}:`, error.response ?
error.response.data : error);
    }
}

function uploadDirectory(directoryPath) {
    fs.readdir(directoryPath, (err, files) => {
        if (err) {
            console.error('Could not list the directory.', err);
            process.exit();
        }

        files.forEach(file => {
            const filePath = path.join(directoryPath, file);
            uploadFile(filePath);
        });
    });
}

const imagesDirectory = './images'; // Change to the path of your images directory
uploadDirectory(imagesDirectory);
```

`package.json` File

Below is the complete `package.json` file tailored to your project. You can copy and paste this into your project directory.

```
{
  "name": "upload-to-pinata",
  "version": "1.0.0",
  "description": "A Node.js script to upload files to Pinata for NFT projects.",
  "main": "uploadToPinata.js",
  "scripts": {
    "start": "node uploadToPinata.js"
  },
  "keywords": [
    "NFT",
    "upload",
    "Pinata",
    "IPFS",
    "Node.js"
  ],
  "author": "Your Name",
  "license": "MIT",
  "dependencies": {
    "axios": "^1.5.0",
    "form-data": "^4.0.0",
    "dotenv": "^16.3.1"
  }
}
```

Example `images` Directory

Ensure your `images` directory contains the files you wish to upload. Below is an example of how your `images/` directory should look:

```
images/
├── art_0001.png
├── art_0002.png
├── metadata_0001.json
└── metadata_0002.json
```

Each file should be correctly named and formatted.

Converting Markdown to PDF with Dark Theme

To convert this Markdown document to a PDF with a dark theme, follow these steps:

1. **Save the Markdown Content:**
   - Copy the entire content from the code block above.
   - Paste it into a text editor and save the file with a `.md` extension, e.g., `README.md`.
2. **Choose a Markdown to PDF Converter:**
   - **Pandoc**:
     - Install Pandoc from here.
     - Use the following command to convert:

       ```
       pandoc -s README.md -o README.pdf
       ```

     - To apply a dark theme, you can create a custom CSS file and use it with Pandoc:

       ```
       pandoc -s README.md -o README.pdf --css=dark-theme.css
       ```

   - **VS Code Extension**:
     - Install the **Markdown PDF** extension from the VS Code marketplace.
     - Open your `README.md` file in VS Code.
     - Press `Ctrl+Shift+P` and select `Markdown PDF: Export (pdf)`.
3. **Customize for Dark Theme:**
   - If using Pandoc, create a `dark-theme.css` with your desired styles.
   - If using VS Code, refer to the extension's documentation to apply custom CSS or themes.
4. **Verify the PDF:**
   - Open the generated PDF to ensure all sections and code blocks are properly formatted within dark boxes.

    - Ensure that code snippets are easily readable and can be copied without issues.

Additional Resources

- Node.js Documentation
- Pinata Documentation
- GitHub Markdown Guide
- Axios Documentation
- Form-Data Documentation
- dotenv Documentation
- Pandoc Documentation
- Markdown PDF Extension for VS Code

For any further assistance, feel free to open an issue or contact the maintainer.