Upload Metadata to Pinata Script

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

Table of Contents

Features

- **Automated Upload**: Uploads multiple metadata JSON 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 metadata files.

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
   ```

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:**

     Install the `dotenv` package to manage environment variables.

     ```
     npm install dotenv
     ```

     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 `uploadMetadata.js` to load environment variables:

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

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

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

     ```
     .env
     ```

Input Files

1. **Metadata Directory (`metadata/`)**
    - **Path**: Located in the root directory (`./metadata`).
    - **Contents**: JSON files containing metadata for each NFT.
    - **Format**: Each JSON file should follow the structure:

```
  {
    "name": "TONS Universe NFT #0001",
    "description": "This is a unique piece from the TONS Universe
collection.",
    "image": "ipfs://<IPFS_HASH>",
    "attributes": [
        // Add any specific attributes here
        // Example:
        // { "trait_type": "Background", "value": "Blue" }
    ]
  }
```

Directory Structure

Ensure your project directory has the following structure:

```
your-repo/
├── uploadMetadata.js
├── metadata/
│   ├── 0001.json
│   ├── 0002.json
│   └── ...
├── package.json
├── .env (optional)
└── README.md
```

Usage

1. **Prepare Metadata Files**
    - Ensure all your metadata JSON files are placed inside the `metadata/` directory.
    - Each file should be named with the NFT ID, e.g., `0001.json`.
2. **Configure API Credentials**
    - Open `uploadMetadata.js`.
    - Replace `'YOUR_PINATA_API_KEY'` and `'YOUR_PINATA_SECRET_API_KEY'` with your actual Pinata API credentials if not using environment variables.
3. **Run the Script**

   Execute the script using Node.js:

```
node uploadMetadata.js
```

Output:

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

4. **Sample Output**

```
Uploaded 0001.json: { IpfsHash: 'Qm...', ... }
Uploaded 0002.json: { IpfsHash: 'Qm...', ... }
...
```

Output

- **Uploaded Metadata**

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

Customization

Handling Additional Attributes

To add specific attributes to your metadata, modify the `attributes` array within each JSON file.

Example:

```
"attributes": [
    { "trait_type": "Background", "value": "Blue" },
    { "trait_type": "Eyes", "value": "Green" },
    { "trait_type": "Accessory", "value": "Hat" }
]
```

Enhancing Error Handling

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

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 `uploadMetadata.js`**

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

const pinataApiKey = process.env.PINATA_API_KEY;
const pinataSecretApiKey = 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 Metadata Directory**

  Ensure the `metadata/` directory exists and contains JSON files. 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.

- **JSON Formatting Errors**

  Ensure all JSON files in the `metadata/` directory are properly formatted. Invalid JSON will prevent successful uploads.

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 `uploadMetadata.js` Code

Below is the complete `uploadMetadata.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 metadataDir = path.join(__dirname, 'metadata');

// Replace with your actual Pinata API key and secret if not using environment
variables
const pinataApiKey = process.env.PINATA_API_KEY || 'YOUR_PINATA_API_KEY';
const pinataSecretApiKey = process.env.PINATA_SECRET_API_KEY ||
'YOUR_PINATA_SECRET_API_KEY';
const pinataEndpoint = 'https://api.pinata.cloud/pinning/pinFileToIPFS';

// Function to upload a file to Pinata
async function uploadFileToPinata(filePath) {
    const fileName = path.basename(filePath);
    const formData = new FormData();
    formData.append('file', fs.createReadStream(filePath), fileName);

    const headers = {
        ...formData.getHeaders(),
        pinata_api_key: pinataApiKey,
        pinata_secret_api_key: pinataSecretApiKey
    };

    try {
        const response = await axios.post(pinataEndpoint, formData, { headers });
        console.log(`Uploaded ${fileName}:`, response.data);
        return response.data;
    } catch (error) {
        console.error(`Failed to upload ${fileName}:`, error.message);
    }
}

// Read all files from the metadata directory and upload them to Pinata
```

```
function uploadAllMetadata() {
    fs.readdir(metadataDir, (err, files) => {
        if (err) {
            console.error("Failed to read metadata directory:", err);
            return;
        }

        files.forEach(file => {
            if (file.endsWith('.json')) {
                const filePath = path.join(metadataDir, file);
                uploadFileToPinata(filePath).then(result => {
                    console.log(`Result for ${file}:`, result);
                }).catch(error => {
                    console.error(`Error uploading ${file}:`, error);
                });
            }
        });
    });
}

uploadAllMetadata();
```

`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-metadata",
  "version": "1.0.0",
  "description": "A Node.js script to upload metadata JSON files to Pinata for NFT projects.",
  "main": "uploadMetadata.js",
  "scripts": {
    "start": "node uploadMetadata.js"
  },
  "keywords": [
    "NFT",
    "metadata",
    "Pinata",
    "IPFS",
    "Node.js"
  ],
  "author": "Your Name",
  "license": "MIT",
  "dependencies": {
    "axios": "^1.5.0",
    "form-data": "^4.0.0",
    "dotenv": "^16.3.1" // Include if using environment variables
  }
}
```

Example `metadata` Directory

Ensure your `metadata` directory contains properly formatted JSON files. Below is an example of how your `metadata/0001.json` file should look:

```
{
    "name": "TONS Universe NFT #0001",
    "description": "This is a unique piece from the TONS Universe collection.",
    "image": "ipfs://QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco",
    "attributes": [
        // Add any specific attributes here
        // Example:
        // { "trait_type": "Background", "value": "Blue" }
    ]
}
```

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

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