jupyter2hashnode

Tiago Patricio Santos

Feb 23, 2023

CONTENTS

CHAPTER

ONE

ABOUT

1.1 About Hashnode

https://hashnode.com/

Citations from Hashnode:

> Helping developers, and people in tech connect and share knowledge easily! Hassle-free blogging and community experience for the creators of tomorrow.
>
> Hashnode is a Free Medium Alternative for Developers! Medium was simply not built with the tech industry in mind.
>
> Hashnode was created to meet the needs of developers who want to plug into the global dev community while retaining ownership of their content on their own custom domains.
>
> And we will never, ever make your audience pay us for access to your articles.

1.2 About Jupyter2Hashnode

As a data scientist, I aim to publish and share my work with others. I have recently started exploring Medium as a platform for sharing my work due to its popularity among many users. However, I soon realized that Medium does not support tables, which are a crucial component of many of my Jupyter notebooks.

As a result, I had to look for alternative platforms, and I discovered Hashnode. It offers a powerful editor that supports markdown text, including tables, and it provides a convenient Amazon AWS S3 bucket for uploading images. Hashnode also has an API that can be used to create blog articles.

Given the ability to export a Jupyter Notebook as a markdown file, I decided to create a tool called Jupyter2Hashnode. This tool simplifies the process of converting Jupyter Notebooks into Hashnode stories by compressing images, uploading them to the Hashnode server, updating image URLs in the markdown file, and finally, publishing the story article with a single command.

In conclusion, Jupyter2Hashnode is a useful tool for converting Jupyter Notebooks into Hashnode stories by simplifying the process of compressing images, uploading images, and publishing the story article.

CHAPTER

TWO

INSTALLATION

2.1 Prerequisites

2.1.1 Python

There are many ways to get Python. If you don't know which one is best for you, you can try Anaconda.

2.1.2 pip

Recent versions of Python already come with `pip` pre-installed. If you don't have it, you can install it manually.

2.1.3 Virtual environments

Use a virtual environment to manage the dependencies for your project, both in development and in production.

What problem does a virtual environment solve? The more Python projects you have, the more likely it is that you need to work with different versions of Python libraries, or even Python itself. Newer versions of libraries for one project can break compatibility in another project.

Virtual environments are independent groups of Python libraries, one for each project. Packages installed for one project will not affect other projects or the operating system's packages.

Python comes bundled with the venv module to create virtual environments.

Create a project folder and a venv folder within:

macOS/Linux:

```
$ mkdir myproject
$ cd myproject
$ python3 -m venv venv
```

Windows:

```
> mkdir myproject
> cd myproject
> py -3 -m venv venv
```

Activate the environment Before you work on your project, activate the corresponding environment:

macOS/Linux:

```
$ . venv/bin/activate
```

Windows:

```
> venv\Scripts\activate
```

2.2 jupyter2hashnode

Install `jupyter2hashnode` with `pip`:

```
$ pip install jupyter2hashnode
```

Depending on your Python installation, you may have to use `python` instead of `python3`. If you have installed the module already, you can use the `--upgrade` flag to get the newest release.

2.3 Other Prerequisites

This prerequisites will be installed automatically when installing the `jupiter2hashnode` package.

2.3.1 nbconvert

The nbconvert is used to convert the notebook to Markdown content. It will be installed with the `jupyter2hashnode` package by default.

If not installed by default have a look at `nbconvert`'s installation instructions.

2.3.2 Pillow

The Pillow is used to compress images before upload them. It will be installed with the `jupyter2hashnode` package by default.

If not installed by default have a look at `Pillow`'s installation instructions.

CHAPTER

THREE

USAGE

3.1 Setup

Create a Python environment and install `jupyter2hashnode` package, check the instructions *here*.

Then is necessary to get the JWT token, Hashnode Api Token and the Publication ID.

JWT To obtain JWT:

1. Open https://hashnode.com, your account must be logged in
2. Open DevTools of chrome browser (F12)
3. Go to Application tab, go to Cookies, find and copy value of "jwt" cookie (245 characters)

Setting the environment variable:

- macOS/Linux:

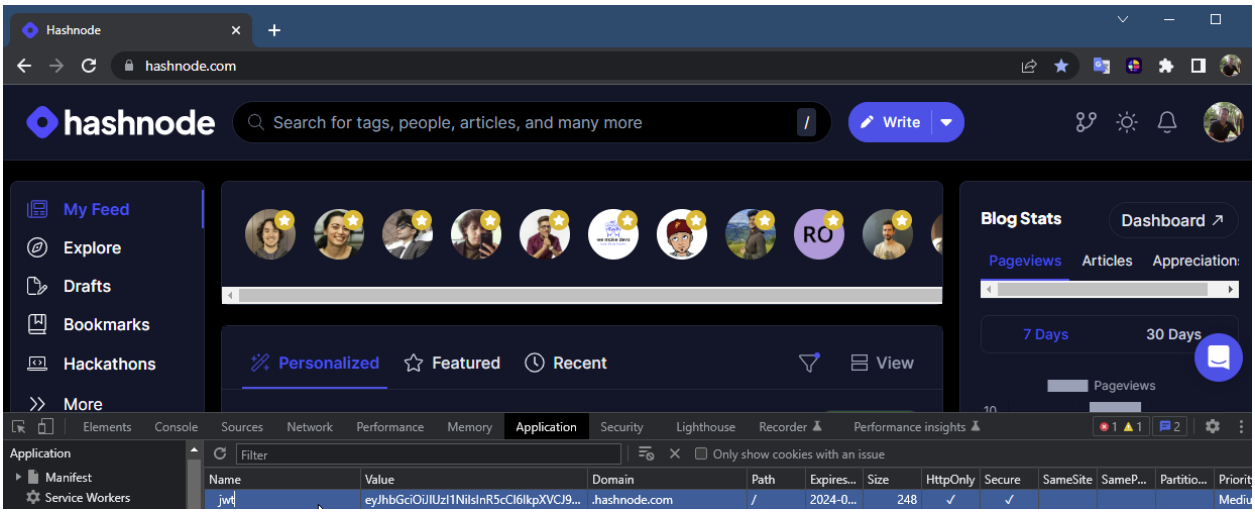
```
$ export HASHNODE_JWT=<your_jwt>
```

- Windows command line:

```
> set HASHNODE_JWT=<your_jwt>
```

- Windows Powershell:

```
> $env:HASHNODE_JWT="<your_jwt>"
```



API TOKEN To obtain Hashnode API token:

1. Open https://hashnode.com/settings/developer
2. Click on "Generate New Token" button or use the existing one

Setting the environment variable:

- macOS/Linux:

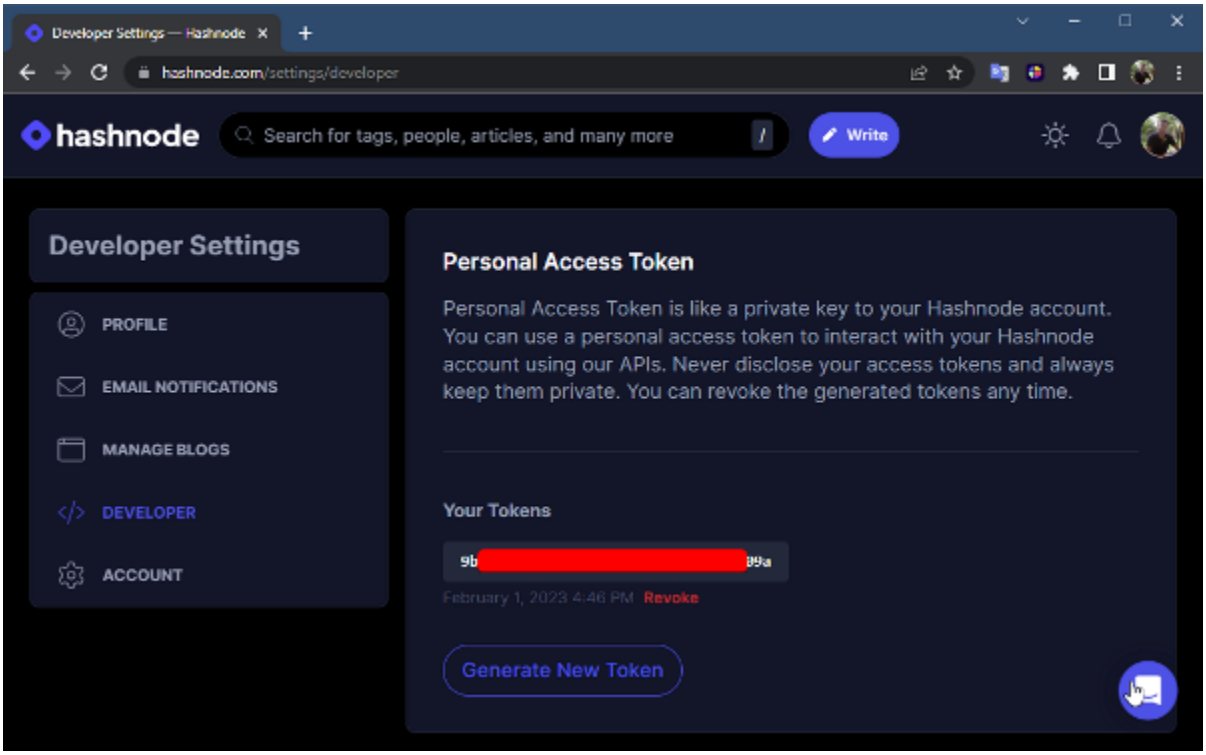
```
$ export HASHNODE_TOKEN=<your_token>
```

- Windows command line:

```
> set HASHNODE_TOKEN=<your_token>
```

- Windows Powershell:

```
> $env:HASHNODE_TOKEN="<your_token>"
```



PUBLICATION ID To obtain Publication ID:

1. Go to https://hashnode.com/settings/blogs
2. Click "Dashboard" of the blog you want to upload to
3. Copy the ID, e.g. https://hashnode.com/<id>/dashboard

Setting the environment variable:

- macOS/Linux:

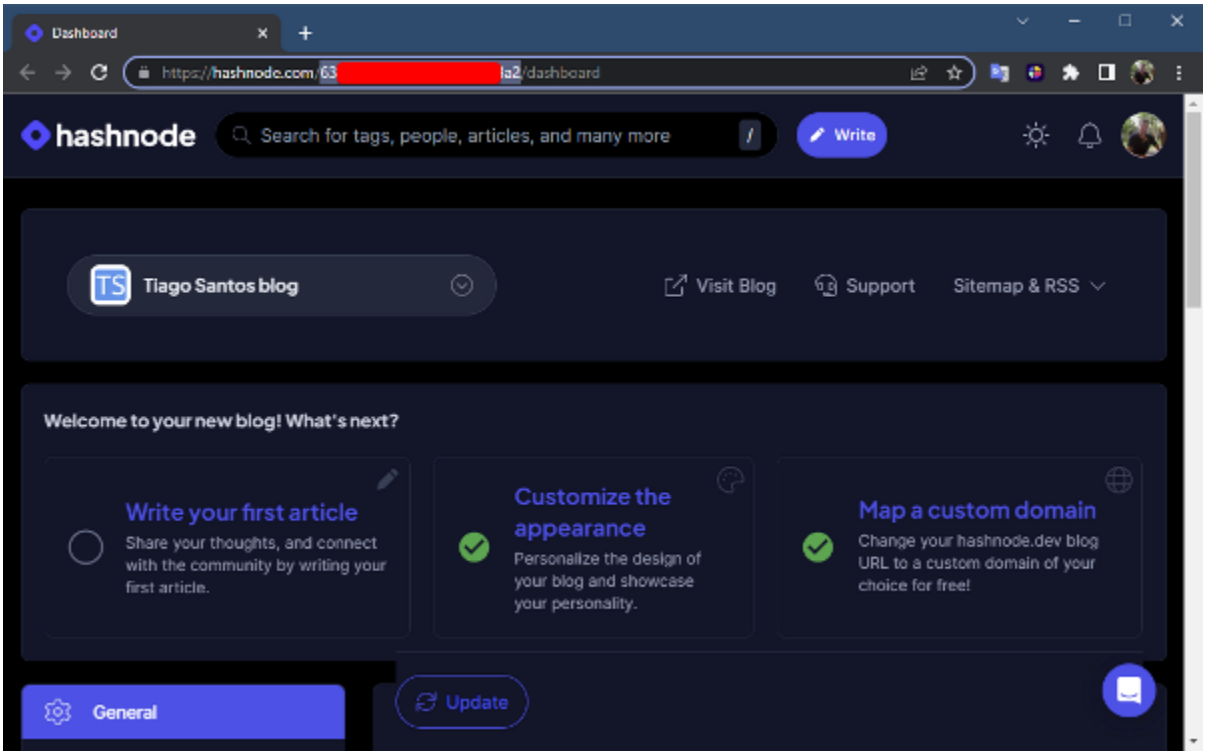
```
$ export HASHNODE_PUBLICATION_ID=<your_id>
```

- Windows command line:

```
> set HASHNODE_PUBLICATION_ID=<your_id>
```

- Windows Powershell:

```
> $env:HASHNODE_PUBLICATION_ID="<your_id>"
```



NOTE: We can set jwt, token and publication id values using environment variables or passing them when calling the `jupyter2hashnode` command.

3.2 As a command line tool

jupyter2hashnode converts the specified Jupyter Notebook to a Hashnode publication story, compressing images, uploading images to the Hashnode server, and replacing image URLs in the markdown file, then published.

If jwt, token, publication_id arguments not passed then will use environment variables HASHNODE_JWT, HASHNODE_TOKEN, HASHNODE_PUBLICATION_ID.

Usage:

```
$ python3 -m jupyter2hashnode [OPTIONS] NOTEBOOK_PATH [OUTPUT_PATH]
```

Arguments:

- `NOTEBOOK_PATH`: notebook file name or complete path [required]
- `[OUTPUT_PATH]`: output folder name or complete output path where the files will be written to, if none creates output folder with the same name as the notebook file name

Options:

- `-j, --jwt TEXT`: JWT token for authentication at https://hashnode.com/api/upload-image.

- `-t, --token TEXT`: Token for authentication at https://api.hashnode.com mutation createPublicationStory endpoint
- `-p, --publication TEXT`: ID of the Hashnode publication e.g. https://hashnode.com//dashboard
- `--title TEXT`: Article title [required]
- `--hide-from-feed / --no-hide-from-feed`: Hide this article from Hashnode and display it only on your blog [default: True]
- `--delete-files / --no-delete-files`: Delete all files after creating the publication story [default: True]
- `--upload / --no-upload`: Upload the publication story to the Hashnode server [default: True]
- `-v, --version`: Show the application's version and exit.
- `--install-completion`: Install completion for the current shell.
- `--show-completion`: Show completion for the current shell, to copy it or customize the installation.
- `--help`: Show this message and exit.

Help command:

```
$ python3 -m jupyter2hashnode --help
```

```
Usage: jupyter2hashnode [OPTIONS] NOTEBOOK_PATH [OUTPUT_PATH] [JWT] [TOKEN]
                        [PUBLICATION_ID]

jupyter2hashnode converts the specified Jupyter Notebook to a Hashnode publication story,  compressing images, uploading images to the Hashnode server, and replacing image URLs  in the markdown file,
then published.
If jwt, token, publication_id arguments not passed then will use environment variables HASHNODE_JWT, HASHNODE_TOKEN, HASHNODE_PUBLICATION_ID.
Notes:
To obtain JWT: Open https://hashnode.com, account must be logged in, open DevTools of chrome browser (F12), go to Application tab, go to Cookies, find and copy value of "jwt" cookie (245 characters)
To obtain Hashnode API token: Open https://hashnode.com/settings/developer, click on "Generate New Token" button or use the existing one
To obtain Publication ID: Go to https://hashnode.com/settings/blogs, click "Dashboard" of the blog you want to upload to, copy the ID, e.g. https://hashnode.com/<id>/dashboard

 Arguments
 *    notebook_path       TEXT             notebook file name or complete path [default: None] [required]
      output_path         [OUTPUT_PATH]    output folder name or complete output path where the files will be written to, if none creates output folder with the same name as the notebook file name
                                           [default: None]
      jwt                 [JWT]            JWT token for authentication at https://hashnode.com/api/upload-image. [env var: HASHNODE_JWT] [default: None]
      token               [TOKEN]          Token for authentication at https://api.hashnode.com  mutation createPublicationStory endpoint [env var: HASHNODE_TOKEN] [default: None]
      publication_id      [PUBLICATION_ID] ID of the Hashnode publication e.g. https://hashnode.com/<id>/dashboard [env var: HASHNODE_PUBLICATION_ID] [default: None]

 Options
 *  --title                      TEXT  Article title [default: None] [required]
    --hide                             Hide this article from Hashnode and display it only on your blog [default: True]
    --delete                -d         Delete all files after creating the publication story [default: True]
    --upload                -u         Upload the publication story to the Hashnode server [default: True]
    --version               -v         Show the application's version and exit.
    --install-completion               Install completion for the current shell.
    --show-completion                  Show completion for the current shell, to copy it or customize the installation.
    --help                             Show this message and exit.
```

Version:

```
$ python3 -m jupyter2hashnode --version
```

3.3 As a library

class Jupyter2Hashnode

The Jupyter2Hashnode class is used to convert Jupyter Notebooks to Hashnode publication stories by compressing images, uploading images to the Hashnode server, and replacing image URLs in the markdown file.

Notes: - To obtain JWT

```
1. Open https://hashnode.com, account must be logged in

2. Open DevTools of chrome browser (F12)
```

(continues on next page)

(continued from previous page)

```
3. Go to Application tab

4. Go to Cookies

5. Find and copy value of "jwt" cookie (245 characters)
```

- To obtain Hashnode API token
  1. Open https://hashnode.com/settings/developer
  2. Click on “Generate New Token” button or use the existing one
- To obtain Publication ID
  1. Go to https://hashnode.com/settings/blogs
  2. Click on “Dashboard” button of the blog you want to upload to
  3. Copy ID from the URL, e.g. https://hashnode.com//dashboard
-

Attributes:

```
HASHNODE_JWT (str): JWT token for authentication at Hashnode image uploader, https://
↪hashnode.com/api/upload-image.
HASHNODE_TOKEN (str): Token for authentication with the Hashnode server, to use https://
↪api.hashnode.com
                            mutation createPublicationStory endpoint
HASHNODE_PUBLICATION_ID (str): ID of the Hashnode publication e.g. https://hashnode.com/
↪<id>/dashboard
```

Methods:

```
create_publication_story(title:str, notebook_path: str, output_path:Optional[str]=None,
                            delete_files:bool=True, upload:bool=True):
    This function is used to create a publication story on the Hashnode blog platform by
    converting a Jupyter Notebook to a markdown file, compressing images, uploading␣
↪images
    to the Hashnode server, and replacing image URLs in the markdown file.

    Parameters:
        title (str): Title of the publication story.
        notebook_path (str): Path to the Jupyter Notebook file.
        hide_from_feed (bool): Hide this article from Hashnode and display it only on␣
↪your blog, Default is True.
        output_path (str, optional): Path to the output directory. Default is None.
        delete_files (bool, optional): Boolean value indicating whether to delete all␣
↪files after
                                            creating the publication story. Default is True.
        upload (bool, optional): Boolean value indicating whether to upload the␣
↪publication story
                                    to the Hashnode server. Default is True.

    Returns:
        None
```

Usage:

```
from jupyter2hashnode import Jupyter2Hashnode

j2h = Jupyter2Hashnode(jwt, token, publication_id)
j2h.create_publication_story(title, notebook_path, hide_from_feed, output_path, delete_
↪files, upload)
```

3.4 Example

When installing the package in a virtual environment some notebook examples are provided to test the story publication in the folder:

```
.venv\Lib\site-packages\jupyter2hashnode\examples
```

Let's publish the story of `example.ipynb`:

```
$ python3 -m jupyter2hashnode .\.venv\Lib\site-packages\jupyter2hashnode\examples\
↪example.ipynb
```

In this example we assume that environment variables have been created. If not then they could be passed as options in the command, check the "Using `jupyter2hashnode` as a command line tool" to view all options, for example passing the token:

```
$ python3 -m jupyter2hashnode .\.venv\Lib\site-packages\jupyter2hashnode\examples\
↪example.ipynb --token "<your_token>"
```

When executing will ask for the title:

```
Article title: My published example
```

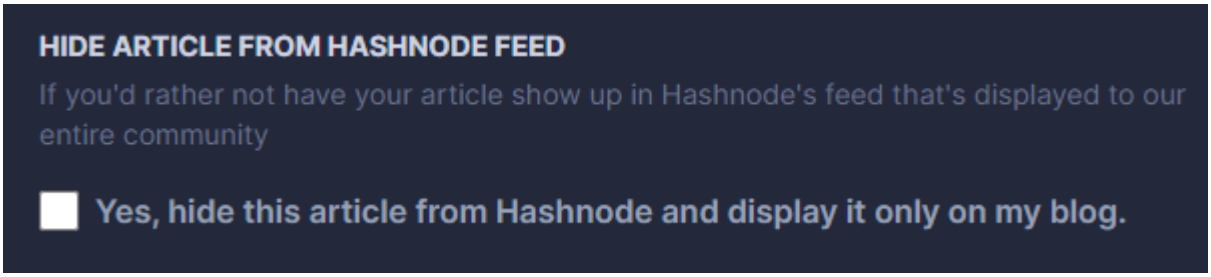
This will export the Jupyter notebook file notebook.ipynb into a markdown file and export all images, the images will be compressed and uploaded to the address https://hashnode.com/api/upload-image, Hashnode s3 amazon AWS bucket, that will generete the correspondent urls for the images. The local path of the images in the markdown file will be replaced by the urls generated and then all the markdown text will be uploaded into a Hashnode Article.

Execution output:

```
Starting the publication...
Creating markdown file at md_example ...
Compressing and uploading images from  md_example ...
Story publication from file md_example\example.md  with the title 'My published example'.
↪..
Result [200]: Publication article created successfully
Hashnode Post URL: https://hashnode.com/edit/clduiv6op0174d...
```

We could click in url provided in the output or we can check the hashnode dashboard:

To make the article visible to others making to show up in Hashnode's feed, edit the article push `Update` and unchecked the option `Yes, hide this article from Hashnode and display it only on my blog` and click update again.



Find more notebook examples here.