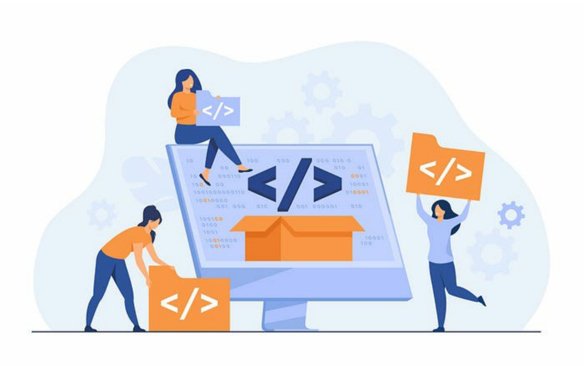
Building your first Flask REST API with MongoDB and JWT

S Shay Pepper · Follow
7 min read · Dec 7, 2020

84 1



Designed by pch.vector / Freepik

In this article, I am going to explain you step-by-step how to build your first Web API app. I assumed you already know the basic of python and you already have Python and an IDE installed.

The following topics will be covered in this article:
-Flask
-MongoDB
-RESTful API
-JWT

. . .

Flask

Flask is a lightweight and an easy-to-use web framework that provides a way to quickly build a backend for your application.
First, install Flask framework using pip with the following command:

```
pip install flask
```

Build a Basic Application

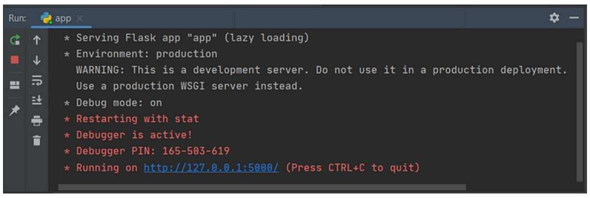
In order to build a basic flask application create a new file called “app.py” and use the following code:

our basic app.py

In the first lines of the code we imported the Flask class and then we create an instance of it. Next, we use the `@app.route` decorator, the parameter we send set what URL should trigger the following function. The “if” statements will be true when our model is being run directly (and not being imported by something else).

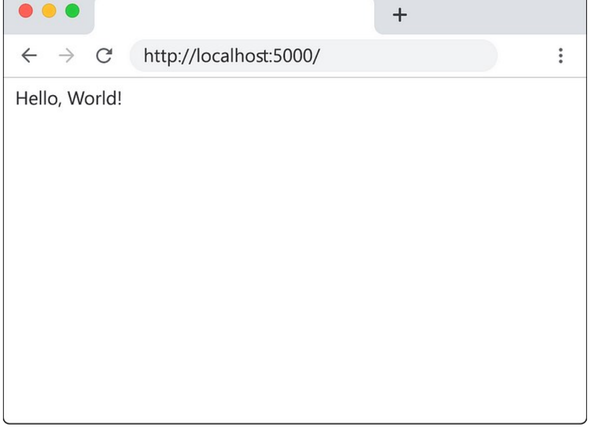
Finally, the `run()` function will run the application, set the debug parameter to true will reload the application on code changes which will help us during the development process.

Run the application by running the “app.py” file, then you should see:



```
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 165-503-619
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Let’s open the http://localhost:5000/ URL in browser to make sure everything work, you should see:



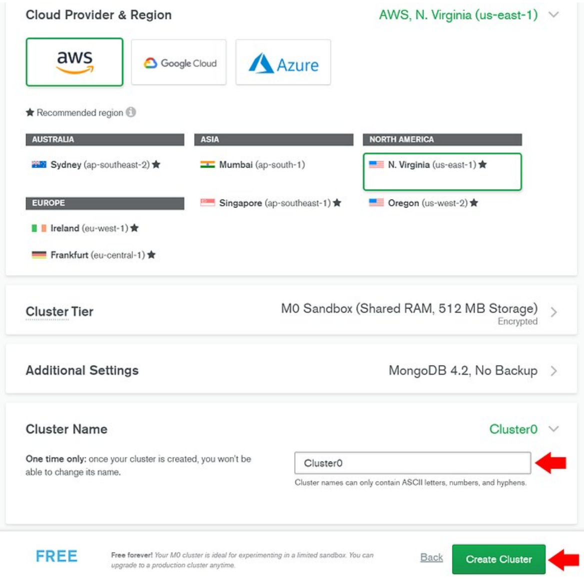
```
Hello, World!
```

MongoDB

MongoDB is one of the most popular NoSQL databases. In SQL databases the data stored in table format. In MongoDB the data stores in JSON format. JSON forms a tree data structure and the individual records are known as 'documents'.

We choose to work with MongoDB because the data represent in the same way that application do. It is also more flexible and easy to scale.

Do the following steps to create your MongoDB database:
1. Go to https://cloud.mongodb.com and create an account.
2. Click on "Create a cluster" button.
3. The default settings should be fine. Scroll down and type in a name for your cluster.



Cloud Provider & Region: AWS, N. Virginia (us-east-1)

Cluster Tier: M0 Sandbox (Shared RAM, 512 MB Storage) Encrypted

Additional Settings: MongoDB 4.2, No Backup

Cluster Name: Cluster0

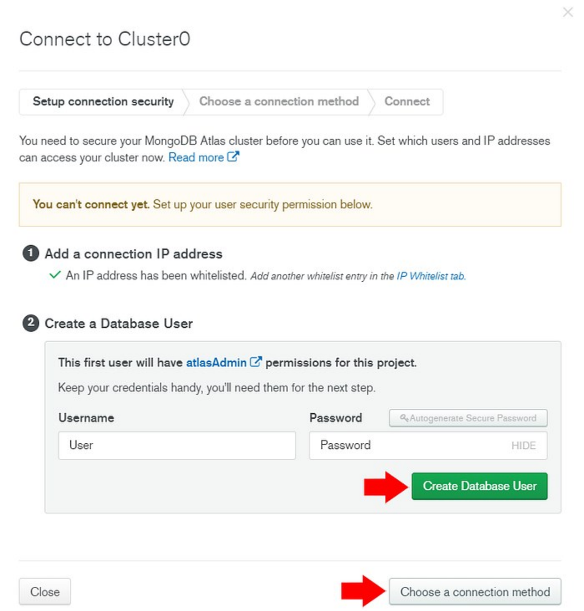
One time only: once your cluster is created, you won't be able to change its name.

Cluster0

Cluster names can only contain ASCII letters, numbers, and hyphens.

FREE **Free forever!** *Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.*

4. Click on "Connect" and then click "Add Your Current IP Address" to be able to connect MongoDB from your computer.
5. Set a username and password and click on "Create Database User".


Setup connection security
Choose a connection method
Connect
Autogenerate Secure Password
HIDE
Create Database User
Close
Choose a connection method

Connect to Cluster0

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. Read more

You can't connect yet. Set up your user security permission below.

1 Add a connection IP address

✓ An IP address has been whitelisted. *Add another whitelist entry in the IP Whitelist tab.*

2 Create a Database User

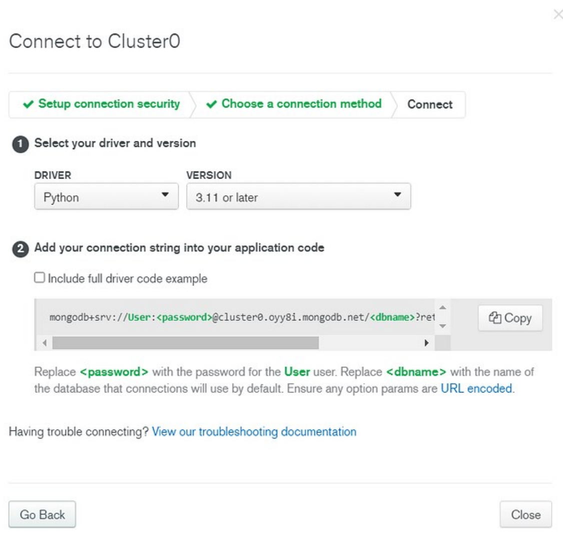
This first user will have atlasAdmin permissions for this project.

Keep your credentials handy, you'll need them for the next step.

Username: User

Password: Password

6. Click on “Choose a connection method” and then “connect your application”.
7. Choose python and copy your connection string, save it for later.



Connect to Cluster0

1 Select your driver and version

DRIVER: Python
VERSION: 3.11 or later

2 Add your connection string into your application code

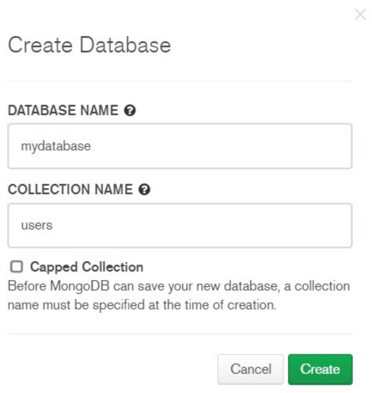
☐ Include full driver code example

```
mongodb+srv://User:<password>@cluster0.oyy8i.mongodb.net/<dbname>?ret
```

Replace <password> with the password for the User user. Replace <dbname> with the name of the database that connections will use by default. Ensure any option params are URL encoded.

Having trouble connecting? View our troubleshooting documentation

8. Now let’s go to “Collection” and click on “Add My Own Data”.
9. Choose database and collection name and click on “ Create”.



Create Database

DATABASE NAME: mydatabase

COLLECTION NAME: users

☐ Capped Collection
Before MongoDB can save your new database, a collection name must be specified at the time of creation.

We are now going to use **PyMongo** library which allows interaction with the ***MongoDB*** database through Python.
So, let’s install it:

```
pip install pymongo
```

Now, let’s import pymongo and configurate our mongoDB database

We use the MongoClient class to make a connection, we then choose our relevant database and collection (as we defined at MongoDB Cloud).

BASIC OF REST:

Rest is a very common Architectural Style for an application program interface.

Rest offers a stateless protocol and standard operation which provides many advantages such as fast performance, reliability, scalability, separation between client and server and more.

In Rest the server data represented as resources, the customer can access the resources by using web URIs (Uniform Resource Identifiers) and HTTP.

Here are some basic things you should know:

HTTP Methods
POST — use to create a new resources.
GET — use to read (or retrieve) a representation of a resource.
PUT — use to update a resource.
DELETE — use to delete a resource.

Endpoints
GET Example.com/api/v1/users — will retrieve a list of users

POST Example.com/api/v1/users — will create a new user
DELETE Example.com/api/v1/users/1 — will delete user with id 1

Status code

RESTful APIs use the **Status-Line** part of an HTTP response message to inform clients of their request's overarching result.
1xx: Informational
2xx: Success — The client's request was successfully.
3xx: Redirection — The client must take some additional action in order to complete their request.
4xx: Client Error — Error at client request.
5xx: Server Error — Error at the server side.

Register

After we talked about RESTful we can now make our first function that will handle the registration logic and let us add new users to our database.

There are some serval basic things we want to do when we add new users. We need to make sure that the username not already registered. If the username is new we can create the new user, if not we will return an error code.

As we learn about RESTful, while register a new user we create a new resource, so this request should be a **POST method**. The HTTP Status code we use are: **Code 201** mean "created" and **Code 409** describe that was a conflict and the request could not be completed.

We also want to **hash the password** of the user before we save it in our database. This way, if someone will get access to our database data he won't be able to know the user's real password, but we still be able to confirm that the password the user send is the same as the one we have in the database.

Login And JWT

To be able to secure the transmitting to the API and identifying the user without any private credentials we will use JWT (Json Web Token)

How do JWT work?

JWT is basically a string that is encrypted on the server-side and holds information about the user, when the user successfully login with their credentials we can return a JWT. The user will attach the JWT to his next requests. Then, we decrypt the JWT on the server-side and we can identify the user.

Why JWT?

JWT has a lot of benefits, unlike Server-side sessions, You don't need to store the session identifier in a database and check them in each request. It's meant JWT let us identify users with less database request. Read more about JWT benefits here:

[add link]

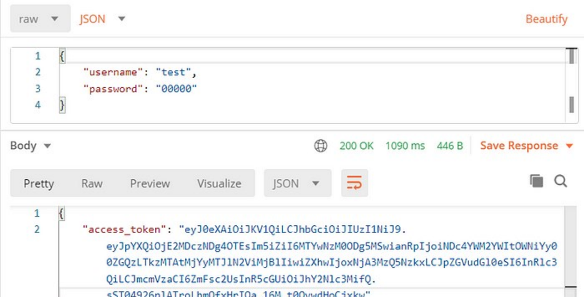
Install flask_jwt_extended in order to use JWT:

```
pip install flask_jwt_extended
```

Now let's write some code:

We define a secret key that will use to sign the token, Your secret key should never be given to anyone else. You need to keep this key a secret. We also need to define for how long the given token will be valid, for that we use the *JWT_ACCESS_TOKEN_EXPIRES* and set it for 1 day.
At first we check if the user exists and if the password are correct. Then, we use `create_access_token()` to create a JWT with the current username (There are ways to store more data inside the JWT).

On successful request you should get this:



```
{
    "username": "test",
    "password": "00000"
}
```

```
{
    "access_token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
        eyJpYXQiOjE2MDczNDg4OTEsIm5iZiI6MTYwNzM0ODg5MSwianRpIjoiNDc4YWM2YWItOWNiYy0
        0ZGQzLTkzMTAtMjYyMTJlN2ViMjBlIiwiZXhwIjoxNjA3MzQ5NzkxLCJpZGVudGl0eSI6InRlc3
        QiLCJmcmVzaCI6ZmFsc2UsInR5cGUiOiJhY2Nlc3MifQ.
        sST04926nlATroLbmOfxHrIOa_16M_t0QvwdHoCjxkw"
}
```

Getting the profile

After we logged in and got the JWT (access token), we need to use it so we could know what user make a request, it will help us to know what data we should return and if the user have the access to that data.

The `@jwt_required` decorator means that this function will work only if the user will attach the JWT to the request. `get_jwt_identity()` function takes the JWT check if it's valid and return the identity data we saved before. It allows us to get the username data from the database. If you try to go to profile route with out JWT, you will get a 401 status code which means unauthorized.

Full Code

This is the full code combined for our Flask web-api app using flask, MongoDB and JWT.

I hope this article was helpful, there are of course a lot of things could be improve in this code like data validation, more error checking and so on, I just want to show you the basic.

There are lots of other topics worth reading and getting to know more, you will find lots of documentation online (and here in Medium).

Flask Mongodb Jwt Rest Api Api Development

84 1

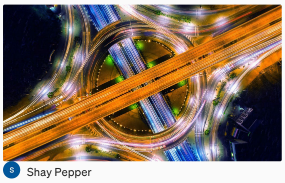
S

Written by Shay Pepper

9 Followers

Follow

More from Shay Pepper



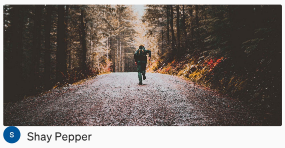
Shay Pepper

How Express Routing Works

Express is a very popular Node.js application framework that makes developing and...

4 min read · Nov 12, 2022

121



Shay Pepper

The Problem of Measuring your Node.js app Runtime

Measuring the runtime of a Node.js application is a critical aspect of performanc...

5 min read · Mar 7

110

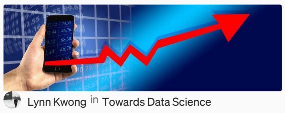
See all from Shay Pepper

Recommended from Medium

Lynn Kwong in Level Up Coding

Build an Autocomplete Search Project Using Elasticsearch,...

Autocomplete search (search as you type) is a very user-friendly feature implemented on...



Lynn Kwong in Towards Data Science

Build a WebSocket Application with FastAPI and Angular

Learn to build a two-way interactive communication application with the...

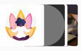
· 13 min read · Mar 14

139 1

· 8 min read · Jan 30

77

Lists



Stories to Help You Grow as a Software Developer

19 stories · 70 saves

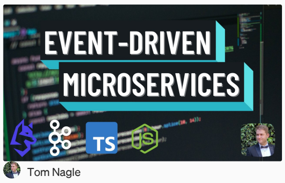
Farhad Malik in FinTechExplained

Securing FastAPI Services OAuth2

Explaining How To Authenticate FastAPI Services Using OAuth2

· 5 min read · Dec 15, 2022

93 1



Tom Nagle

Building Event-driven Microservices with Node.js & Kafka

Event-driven microservices are a design pattern for building scalable and resilient...

· 2 min read · Dec 21, 2022

60

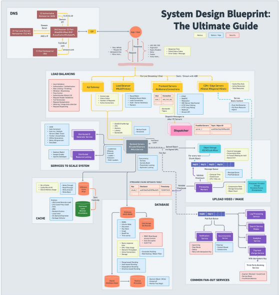
Lukas Wimhofer in Level Up Coding

API development with nodejs, express and typescript from...

This is the third part of the series "API development with nodejs, express and...

· 9 min read · Feb 8

61



Love Sharma in Dev Genius

System Design Blueprint: The Ultimate Guide

Developing a robust, scalable, and efficient system can be daunting. However,...

· 9 min read · Apr 20

1.6K 13

See more recommendations