# Introduction to Backend Using Flask & MongoDB Atlas

We’ll be getting started around 10.40am

## What we’ll be making today

- A web app!
- To add and store recipes
- Frontend is in React.js, a JavaScript library
- Backend is using Flask, a Python framework
- Demo!
- Github repo: https://github.com/stella0304/CatalystWebappDemp
- 
- Access the slides here:
- Shorturl: shorturl.at/bJTY7
-

## What is the frontend?

- Everything you see on the website e.g. colours, buttons, links, animations, and more…
- Frontend developer job is to take the vision and design concept from the client and implement it though code
- Skillset - HTML, CSS and JavaScript
- Frontend is where the “client-side” processes run
- Frontend frameworks: React.js, Vue.js, AngularJS

*

## What is the backend?

- Focuses on everything you cannot see in the website
- Developers ensure the website performs correctly
- Backend
  - Deals with reading and writing data from the database server
  - Is where all API (Application Programming Interface) calls are defined so that the frontend and backend can communicate with each other
  - Also referred to as “server-side” software
- Backend frameworks can be: Flask, Node.js, Django, Express.js etc
- Databases: MongoDB, MySQL, Oracle etc

*

## What is Flask?

- Flask is a Python web app framework
- Doesn't require particular tools or libraries
- Very popular for backend web development due to its simplistic nature.

- Flask cheat sheet (Reference: Pretty Printed)
    - https://s3.us-east-2.amazonaws.com/prettyprinted/flask_cheatsheet.pdf
    - May be useful for setting up a barebones Flask app

*

## What is MongoDB?

- MongoDB is a NoSQL database program
  - NoSQL databases are non-tabular and have flexible schemas
  - Lots of document types can be stored including JSON files
- MongoDB Atlas is a cloud database which we'll be using today
- PyMongo
  - Library used to connect and interact with MongoDB using Python

## What is a JSON file format?

- JSON - JavaScript Object Notation
- Lots of datatypes e.g. number, string, array, objects

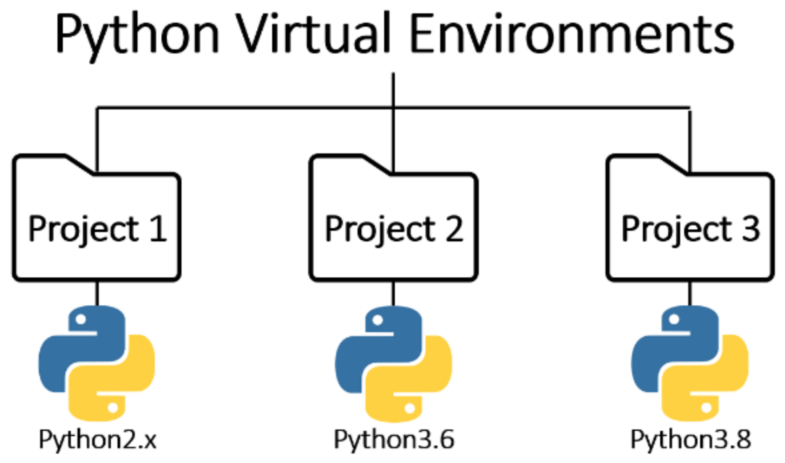
- Object data-type is very similar to Python dictionaries
  - Key-value pairs e.g. {“event” : “Catalyst”, “food” : “pizza”, “day” : “Saturday”}
- Array datatype is similar to Python arrays e.g. [“Apple”, “Dragonfruit”, “Guava”]
- Can have an array of objects. Can even nest objects within objects

- A lightweight format for storing and transporting data
- Often used when data is sent from a server to a webpage
- We’ll be storing our data as JSON files in this demo

- JSON cheat sheet here with examples of datatypes:
  - https://cheatography.com/mackan90096/cheat-sheets/json/

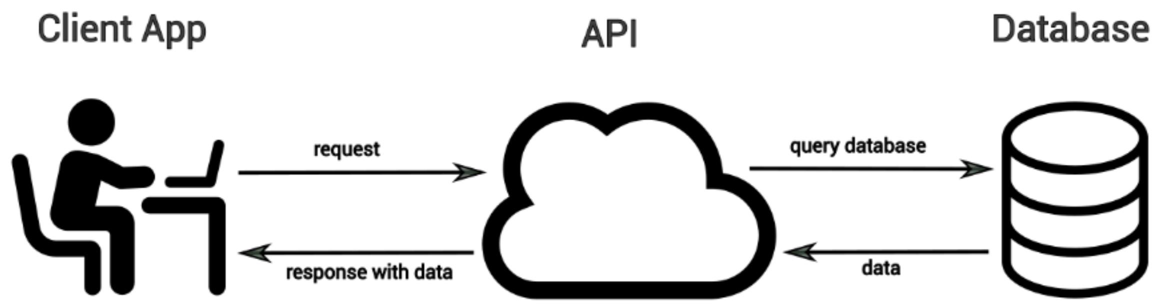
## What is a virtual environment?

- A virtual environment (`env`) is an isolated environment where we can install libraries, scripts etc
- Keeps dependencies (e.g. Flask) required by different projects separate by creating an isolated python environment for them
- Networked application that allows a user to interact with both the computing environment and the work of other users.
- There are no limits to the number of environments it can have.



*

## What is an API?



- An application programming interface (API) is a software intermediary that allows two applications to talk to each other.
- When using an application on your phone, the application connects to the internet and sends data to the server.
- The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone.

- We'll write APIs in the backend which accesses the database in some way e.g. retrieves all records from the database.
- The frontend will be able to use those API functions to communicate with the backend and the database

*

## HTTP Methods

- APIs send requests to the server. Hypertext Transfer Protocol  (HTTP) are request-response protocols to communicate between clients and servers.
- These are the most commonly-used HTTP methods:
- GET
  - Read-only
  - Used to retrieve data
- POST
  - Used to create data
  - Writing to the database
- PUT
  - Used to update data
- DELETE
  - Used to delete data from the database

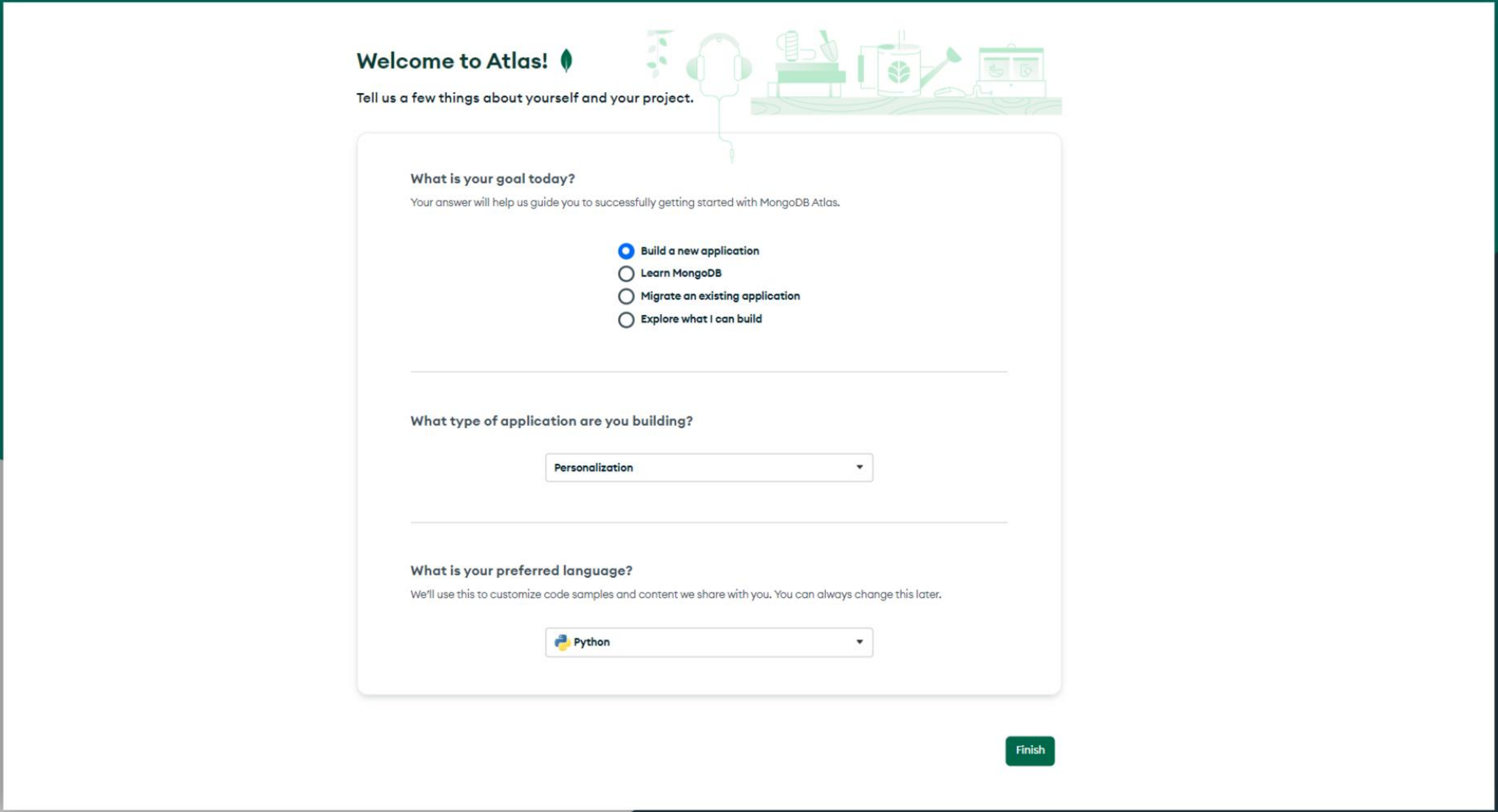
## API Testing Tool - Postman

- There are many tools that can be used to test your APIs are working as expected
- We'll be using Postman today for demo (others include Testim, SoapUI etc)

## Demo!

## MongoDB Atlas Account

- Sign up for a (free) MongoDB Atlas Account : https://www.mongodb.com/cloud/atlas/register
- Create a “Shared” free-tier database
- Choose defaults and create your cluster



## Welcome to Atlas!

Tell us a few things about yourself and your project.

**What is your goal today?**

Your answer will help us guide you to successfully getting started with MongoDB Atlas.

[✓ Build a new application]
Learn MongoDB
Migrate an existing application
Explore what I can build

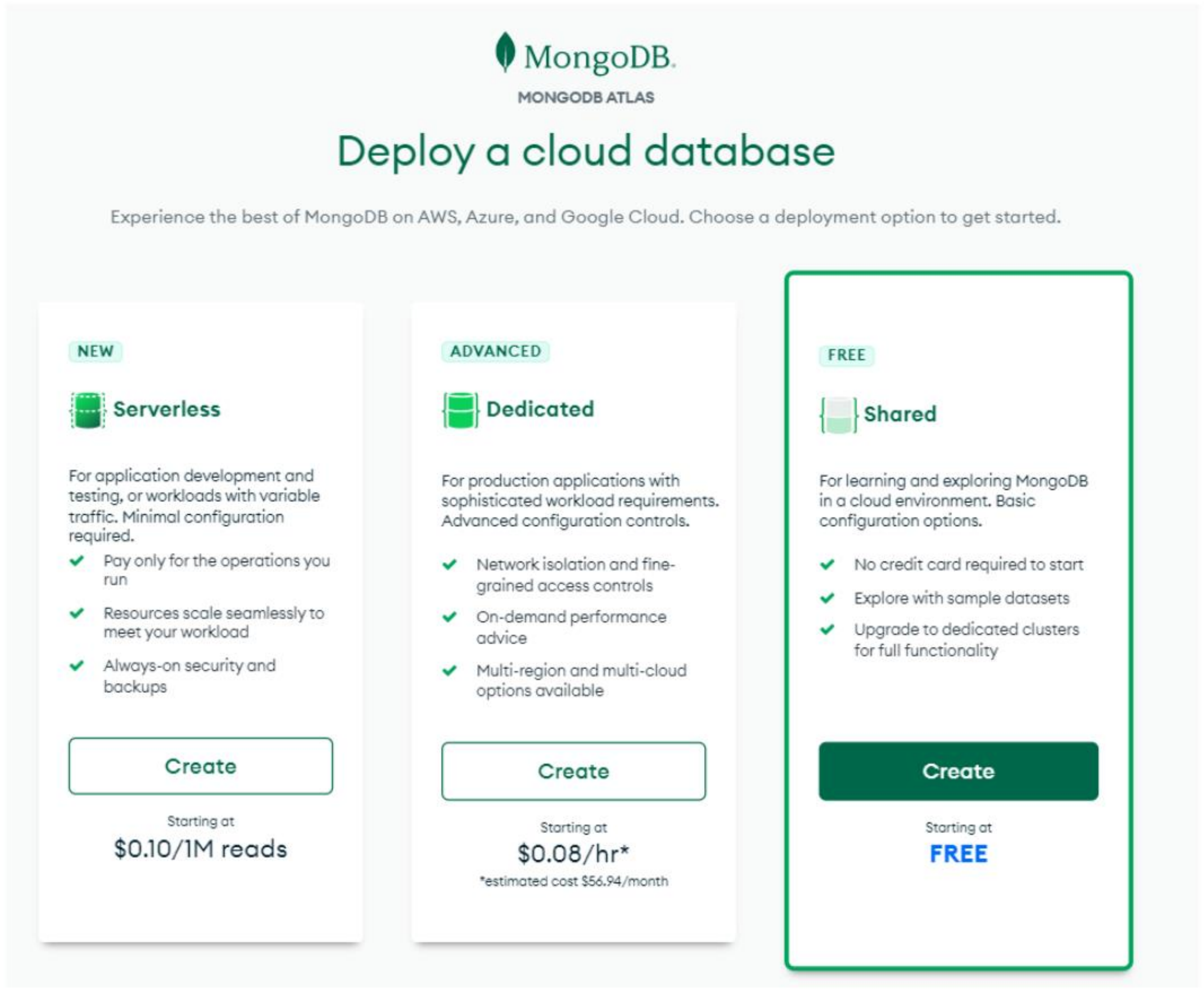
**What type of application are you building?**

Personalization

**What is your preferred language?**

We'll use this to customize code samples and content we share with you. You can always change this later.

Python



MongoDB.

MONGODB ATLAS

Deploy a cloud database

Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started.

NEW

**Serverless**

For application development and testing, or workloads with variable traffic. Minimal configuration required.

- ✓ Pay only for the operations you run
- ✓ Resources scale seamlessly to meet your workload
- ✓ Always-on security and backups

Create

Starting at

$0.10/1M reads

ADVANCED

**Dedicated**

For production applications with sophisticated workload requirements. Advanced configuration controls.

- ✓ Network isolation and fine-grained access controls
- ✓ On-demand performance advice
- ✓ Multi-region and multi-cloud options available

Create

Starting at

$0.08/hr*

*estimated cost $56.94/month

FREE

**Shared**

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

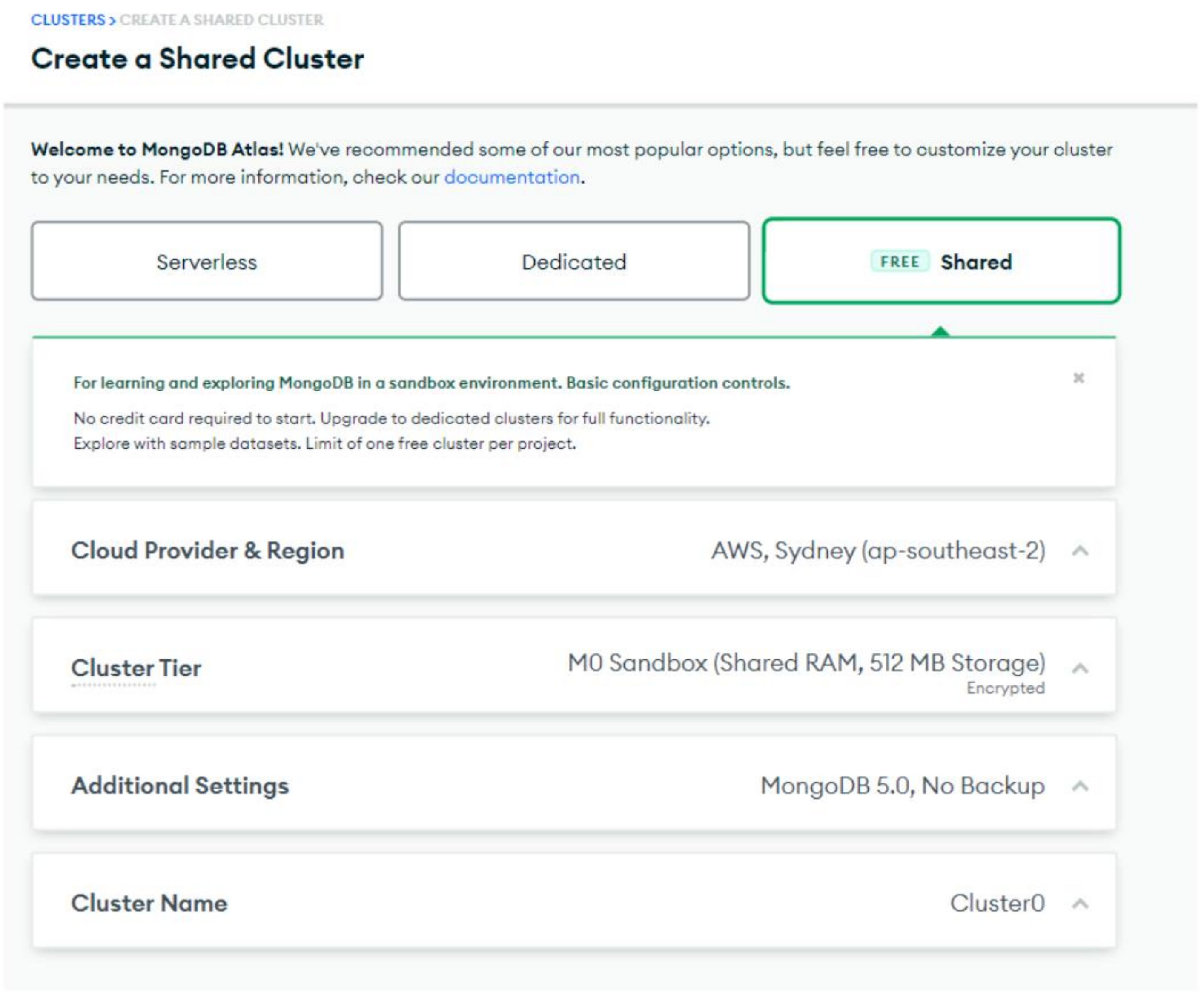
- ✓ No credit card required to start
- ✓ Explore with sample datasets
- ✓ Upgrade to dedicated clusters for full functionality

Create

Starting at

FREE

Click ‘Shared’



CLUSTERS > CREATE A SHARED CLUSTER

Create a Shared Cluster

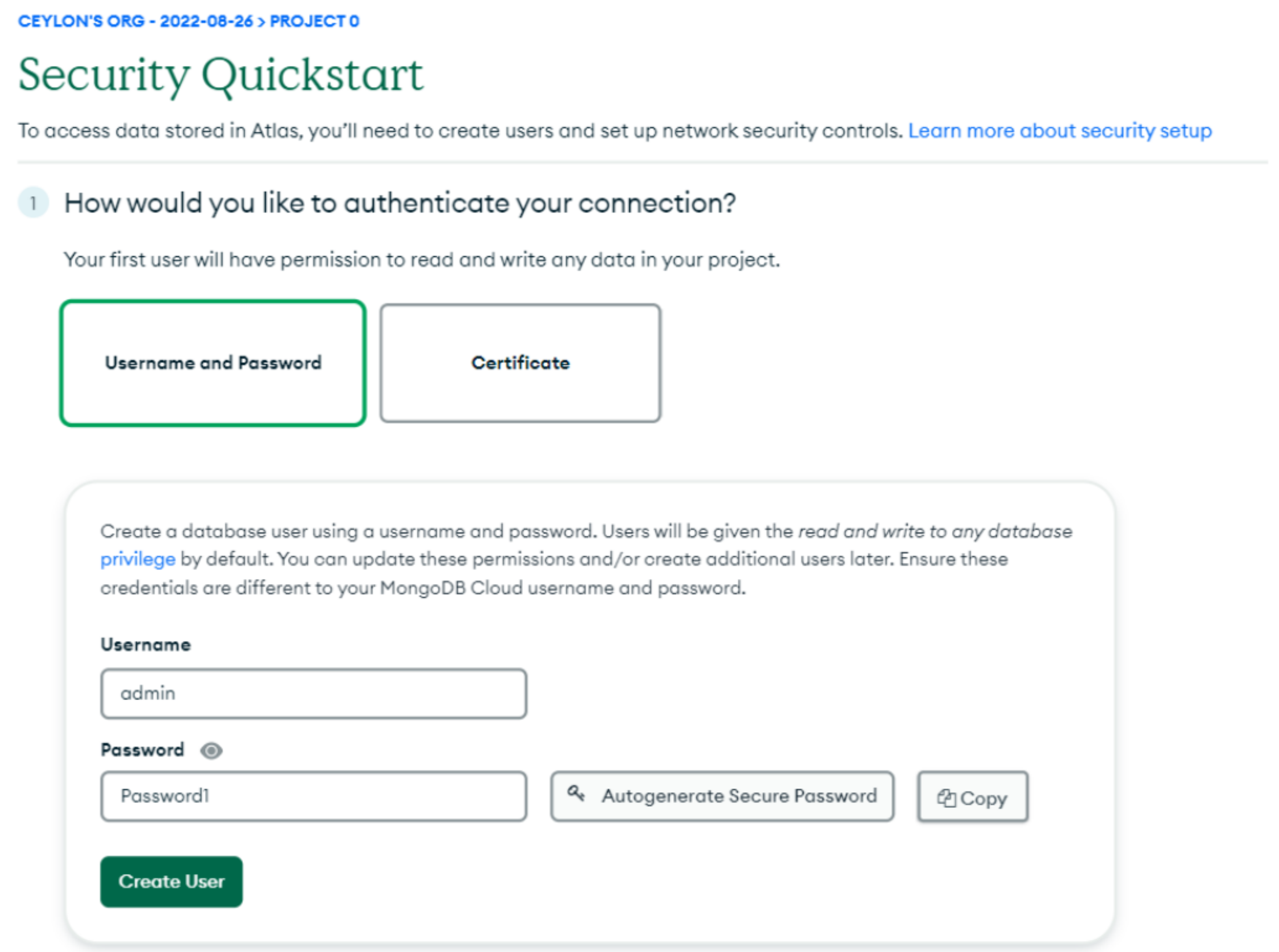
**Welcome to MongoDB Atlas!** We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our documentation.

**For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.**

No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

|  |  |
|---|---|
| Cloud Provider & Region | AWS, Sydney (ap-southeast-2) |
| Cluster Tier | M0 Sandbox (Shared RAM, 512 MB Storage) Encrypted |
| Additional Settings | MongoDB 5.0, No Backup |
| Cluster Name | Cluster0 |

Choose the defaults and scroll down and click Next


CEYLON'S ORG - 2022-08-26 > PROJECT 0

Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. Learn more about security setup

1 How would you like to authenticate your connection?

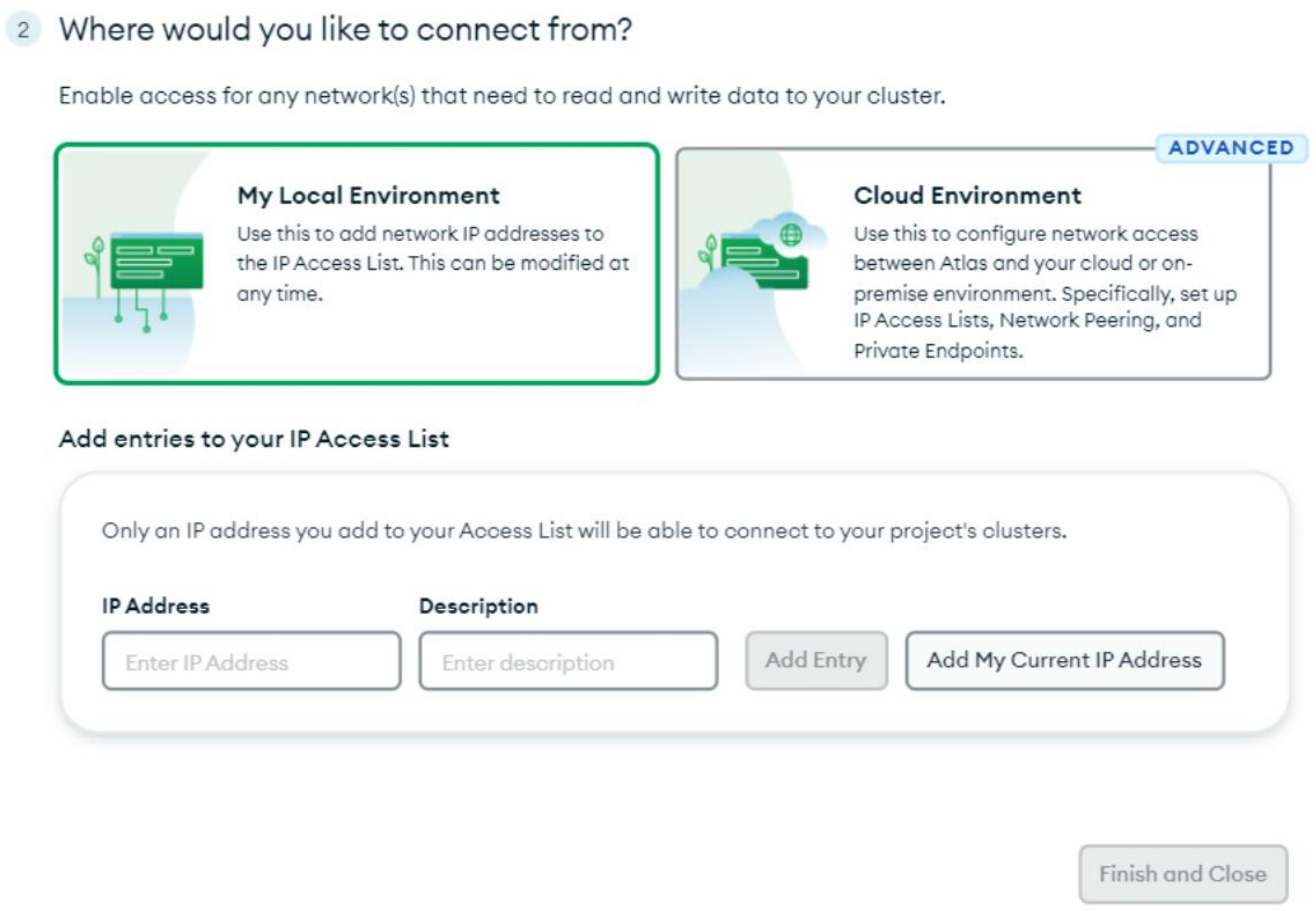
Your first user will have permission to read and write any data in your project.

Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username: admin

Password: Password1

Create a user so you can connect to the database from Flask (this is the one created for Demo)



2 Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

**My Local Environment**
Use this to add network IP addresses to the IP Access List. This can be modified at any time.

ADVANCED
**Cloud Environment**
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

**Add entries to your IP Access List**

Only an IP address you add to your Access List will be able to connect to your project's clusters.

| IP Address | Description |
|---|---|
| Enter IP Address | Enter description |

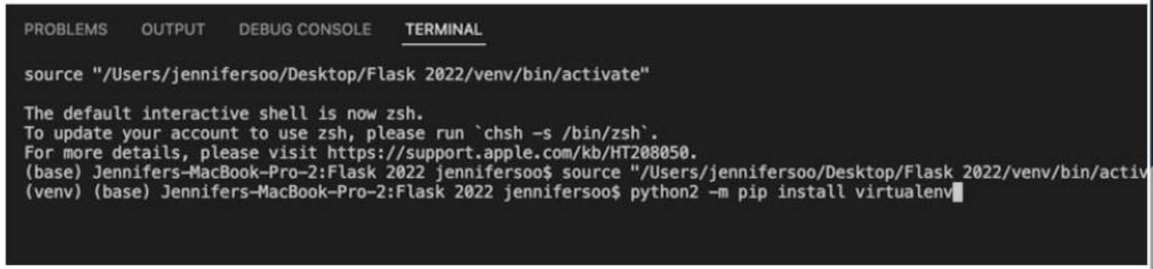
Add your current IP address

## Setting up the environment

**Step 1:** Install virtual environment (Skip this step if python3 installed)

macOS: `sudo python2 -m pip install virtualenv`

windows: `py -2 -m pip install virtualenv`

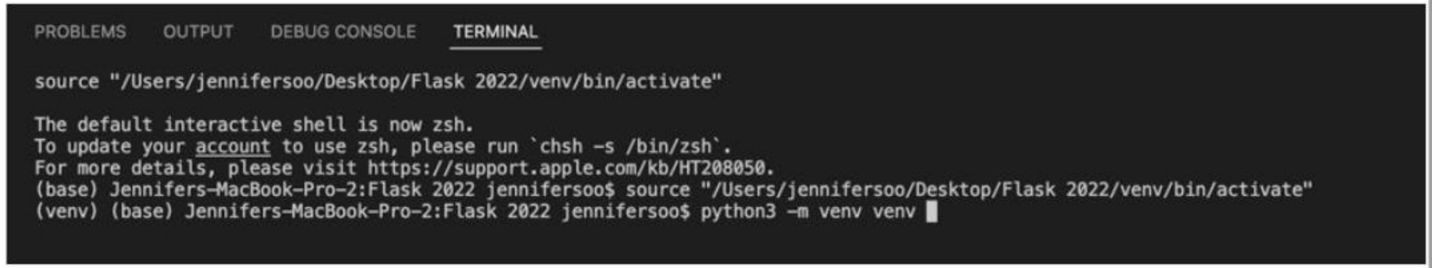


```
source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activate"

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activ
(venv) (base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ python2 -m pip install virtualenv
```

**Step 2:** Create an environment (Only need to do this once)

macOS: `python3 -m venv env`

windows: `python -m venv env`



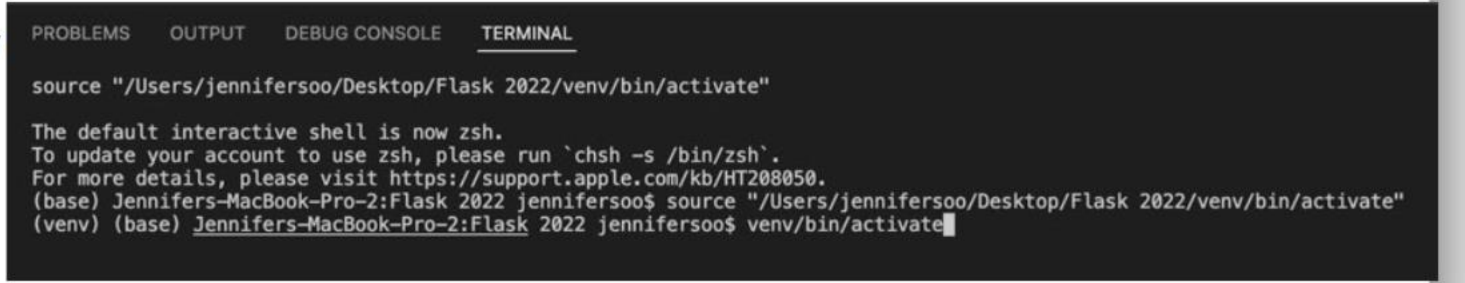
```
source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activate"

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activate"
(venv) (base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ python3 -m venv venv
```

**Step 3:** Activate the environment

Need to do this every time you want to run virtual env

macOS: `source env\bin\activate`

Windows: `.\env\Scripts\activate`



```
source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activate"

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ source "/Users/jennifersoo/Desktop/Flask 2022/venv/bin/activate"
(venv) (base) Jennifers-MacBook-Pro-2:Flask 2022 jennifersoo$ venv/bin/activate
```

# Installing Flask

**Step 4**

Create a file app.py with barebones Flask app (example given on the right)

**Step 5**

Install Flask within virtual environment

Windows: `pip install flask`

Mac: `pip3 install flask`

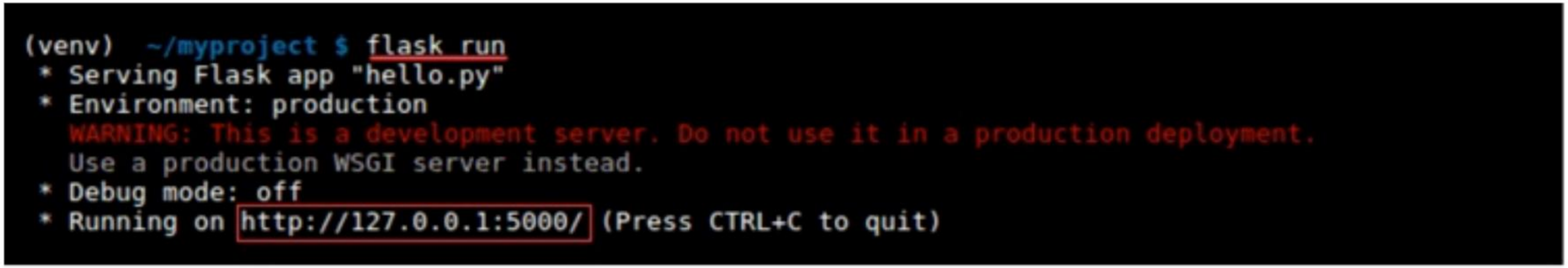
**Step 6**

Run the flask app

Windows: `python app.py`

Mac: `python3 app.py`

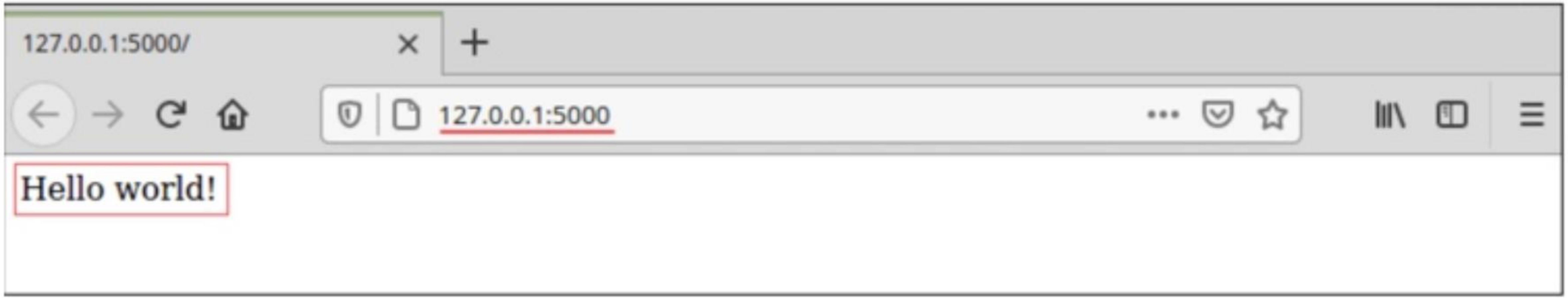
```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello world!'
```



```
(venv) ~/myproject $ flask run
 * Serving Flask app "hello.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The output prints out a confirmation message and the address.

8. Copy and paste the address into the browser to see the project running:



Hello world!

## Modules to install

- You should be installing these modules at some point (while the virtual environment is running!)
- Command:
    - Windows: `pip install <module>`
    - Mac: `pip3 install <module>`
- Modules
    - flask
    - flask_pymongo
    - (May also need flask-pymongo)
    - python-dotenv
    - dnspython

## What we did not cover in this demo

- Deployment
  - How to deploy your web app so that others can access it e.g. Heroku
  - Currently we’re running our web app locally on our own computer
- Security
  - Have checks on what people can pass into the web app
  - Authentication