Intro to Python Backends

Day 4: 3DC Introduction to Programming

Raphael Yee

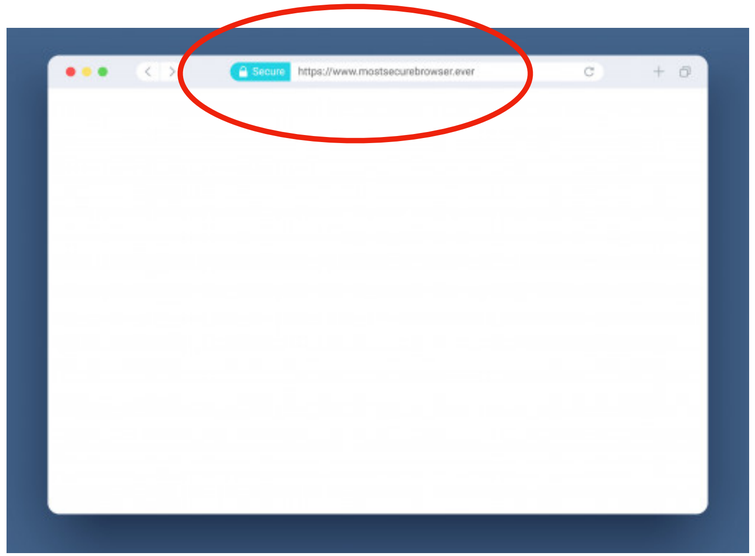
Recap:

What have we learnt?

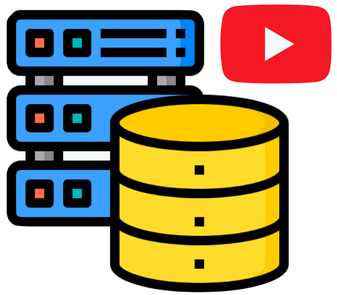
- Information needs to be exchanged for stuff to happen
- We can do so by making “push” or “pull” requests to/from servers.
- We can use the Python Requests library to play with data

… But what if we want to create the application that responds to these requests?

Previously…



GET https://www.youtube.com



HTML, Javascript, CSS

How does this **process a request** and and **return a response**?

How do application backends work?

How do they parse requests?

Where does information get stored?

How do we retrieve this information?

How do servers work?

Essentially what's happening is that your app server "listens" on a port for incoming activity.

When it does, it takes the data and does some processing and returns the response.

Think: What is a **port**? USB-ports, shipping ports…?

Note: There are different kinds of ways for client and server to "talk"…ultimately, it depends on what your use-case. Again, do you need to push information from both sides or just push from one?

Read more: https://superuser.com/questions/837933/how-do-web-servers-listen-to-ip-addresses-interrupt-or-polling

Specifying our API endpoints

GET "/" returns "Hello World!"

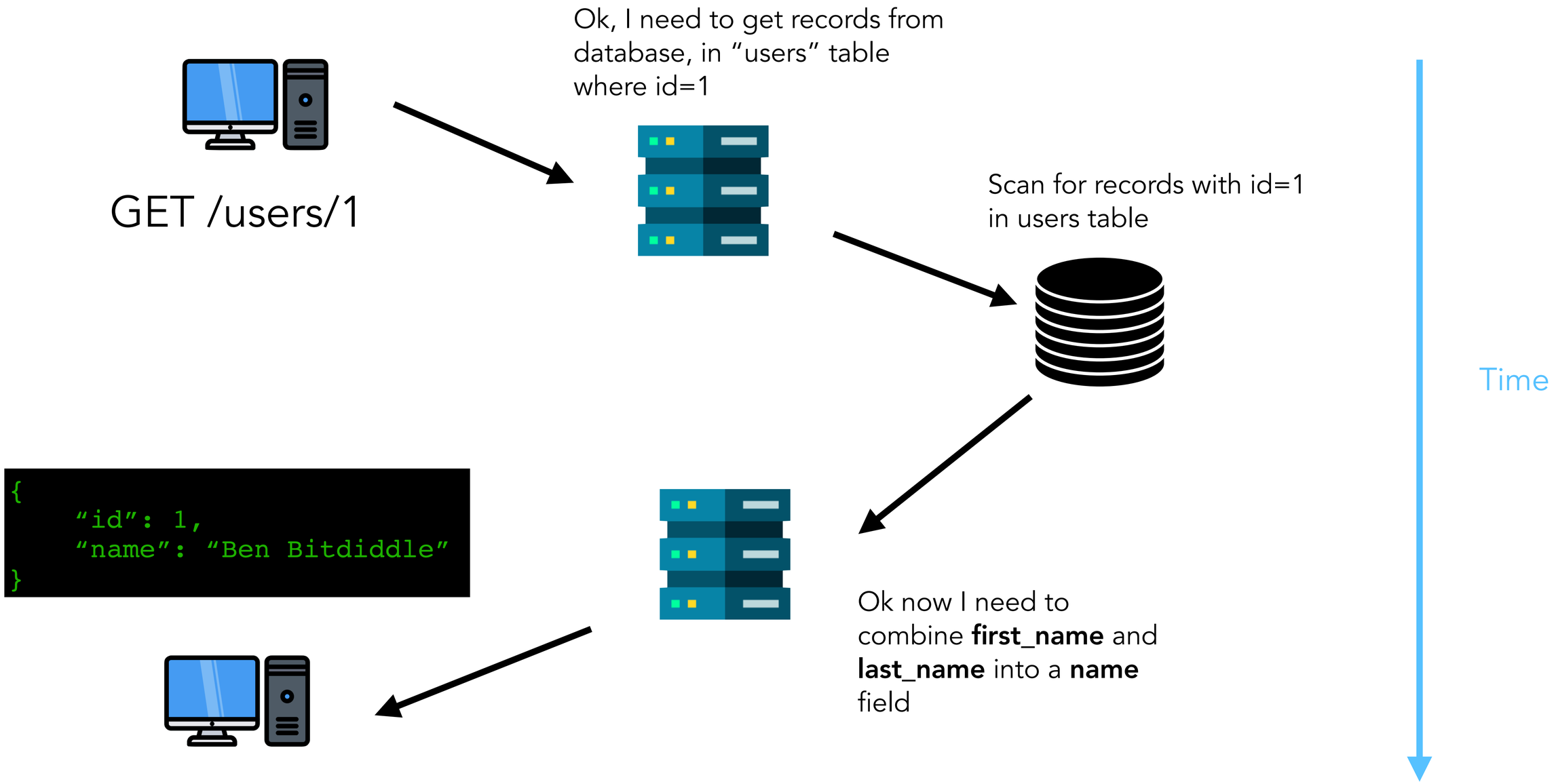
GET "/get_data" returns "Your data"

```
@app.route("/")
def helloworld():
    return "Hello World!"

@app.route("/get_data")
def getdata():
    return "Your data"

if __name__ == "__main__":
    app.run()
```

Think: In the previous session, we learned that you could return JSON… how will we do that here?



Think: What happens when calls are made over the internet? How do they differ from in-process calls?

What are databases?

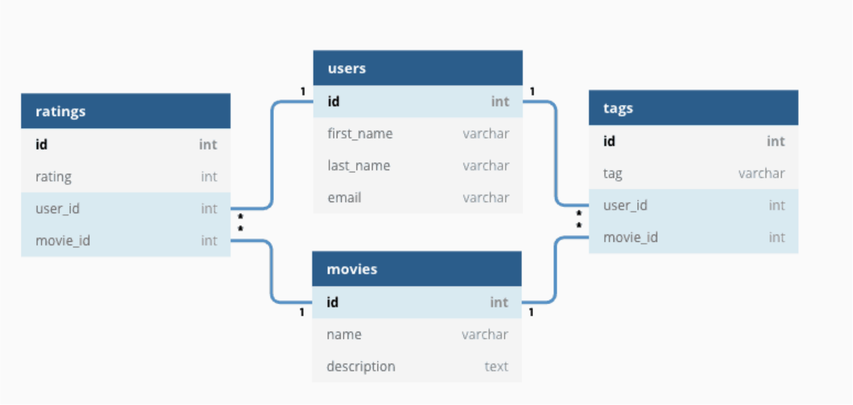
Databases are collections of information that allow you to store, update, delete and manage data.

Basically, if you want to **persist** data after your application ends, you will most likely need a database (or some place to store that data).

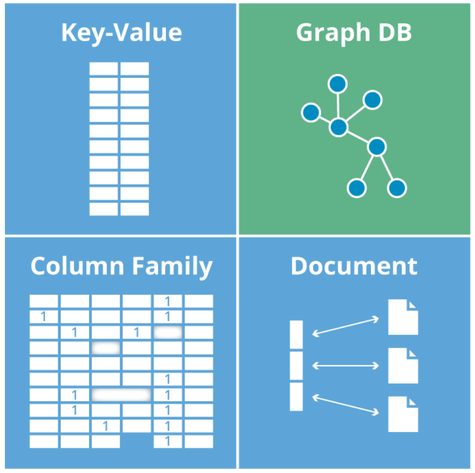
Note: There are different kinds of databases for different purposes. You have non-relational and relational databases.

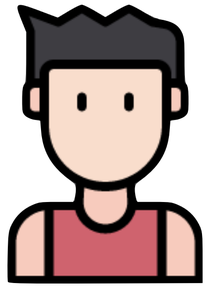
Think: What's the difference between a relational and non-relational database?

This is a relational database (RDBMS)



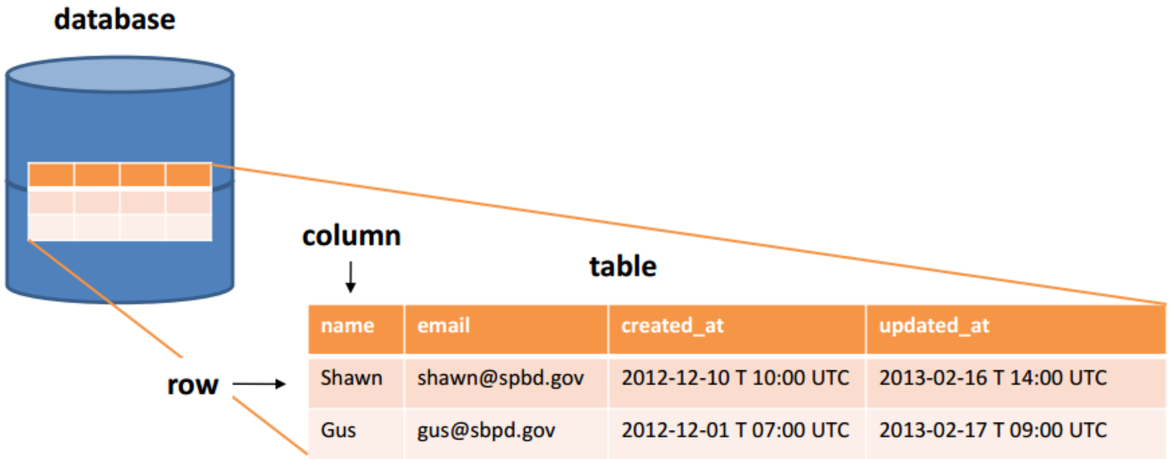
These are non-relational types of DBs.





"Hey man, I need the email of every person named Shawn"

```
SELECT
    email
FROM users
WHERE
    name = "Shawn"
```



name	email	created_at	updated_at
Shawn	shawn@spbd.gov	2012-12-10 T 10:00 UTC	2013-02-16 T 14:00 UTC
Gus	gus@sbpd.gov	2012-12-01 T 07:00 UTC	2013-02-17 T 09:00 UTC

Document-based storage

A document database looks very much like JSON. Instead of tables and rows, we have nested “dictionary”-like structures that allow us to query for data using key-values.

Note: Notice that the first book has a lot more information than the second book and yet both entries are **valid**. One benefit of document-based storage is that we can define (and similarly leave out) pieces of information we don’t need or don’t have.

In a relational database, your data is very structured and defined by the columns you have (or do not).

JSON

```
[
    {
        "year" : 2013,
        "title" : "Turn It Down, Or Else!",
        "info" : {
            "directors" : [ "Alice Smith", "Bob Jones"],
            "release_date" : "2013-01-18T00:00:00Z",
            "rating" : 6.2,
            "genres" : ["Comedy", "Drama"],
            "image_url" : "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RLlo5JF90EWR7LJKQ7@@._V1_SX400_.jpg",
            "plot" : "A rock band plays their music at high volumes, annoying the neighbors.",
            "actors" : ["David Matthewman", "Jonathan G. Neff"]
        }
    },
    {
        "year": 2015,
        "title": "The Big New Movie",
        "info": {
            "plot": "Nothing happens at all.",
            "rating": 0
        }
    }
]
```

Think: When do we prefer document-based storage over relational databases?

What is JSON?

```
{
    “id”: 1,
    “name”: “Ben Bitdiddle”
}
```

Javascript Object Notation (JSON) is a text format of data used when sending information from client to server and vice-versa.

You can think of using JSON like Python Dictionaries, both are syntactically similar and use key-value pairs to store/access information.

eg. to print “Ben Bitdiddle”: **print(data[“name”])**

Think: What is the data type of JSON? (strings… integers… etc.)

We'll be creating a quick server using Flask and storing the data on Google Firebase.

Flask

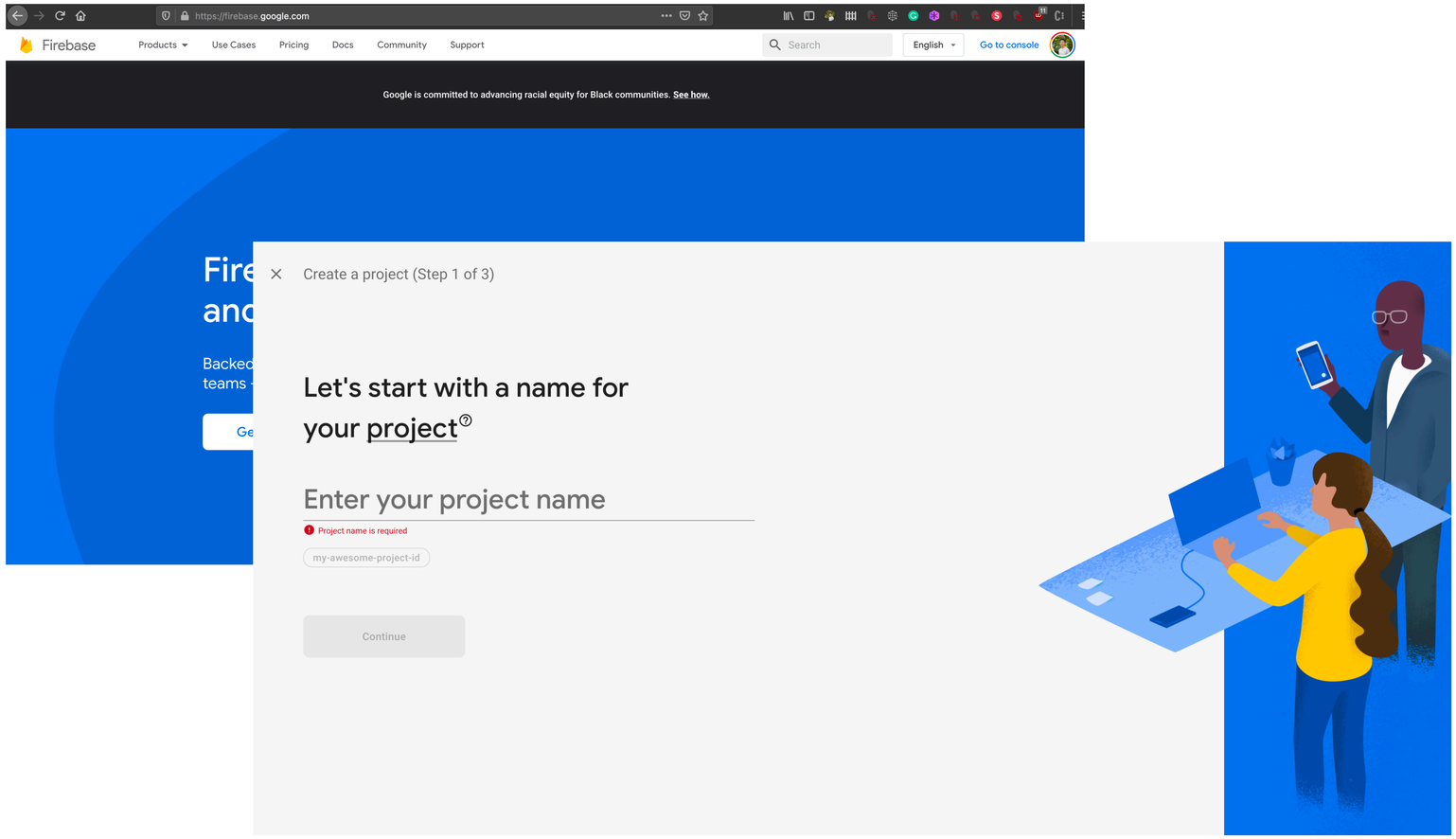
Firebase

Let’s write our first Flask app!

Activity #1

Setting up Firebase

Go to firebase.google.com



Think: What kind of database is Firebase's Realtime-Database?

Hello World

```python
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'

if __name__ == '__main__':
    app.run()
```

Think: Line-by-line, what's happening?

Let’s save your scraped data

Let’s access your scraped data

Does everything work?

1. Test your functions
2. Does it do what you intended for it to do?
3. Use Postman to test your APIs
4. Any bugs?

Deploying our app to the web

Activity #2

What is deployment?

Now, your app is available locally on your computer or over the WIFI network. But what if you want it to be available over the internet?

Deployment is the process of putting your app on cloud servers, allowing your app to be accessed over the web.

HEROKU

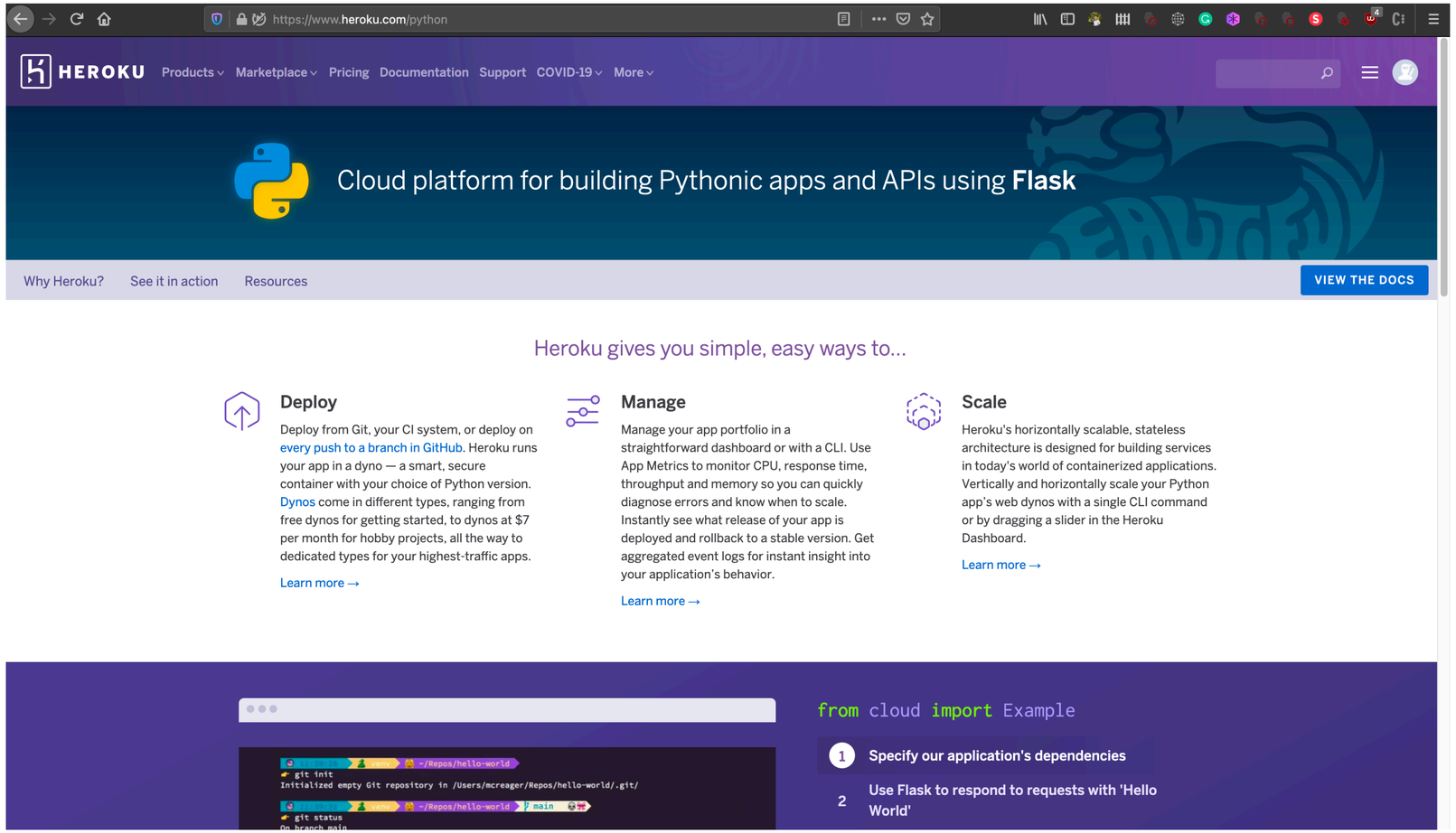
Amazon EC2

DigitalOcean

Google Cloud

Let’s go, Heroku!

Go to heroku.com



Recap:

What have we learnt?