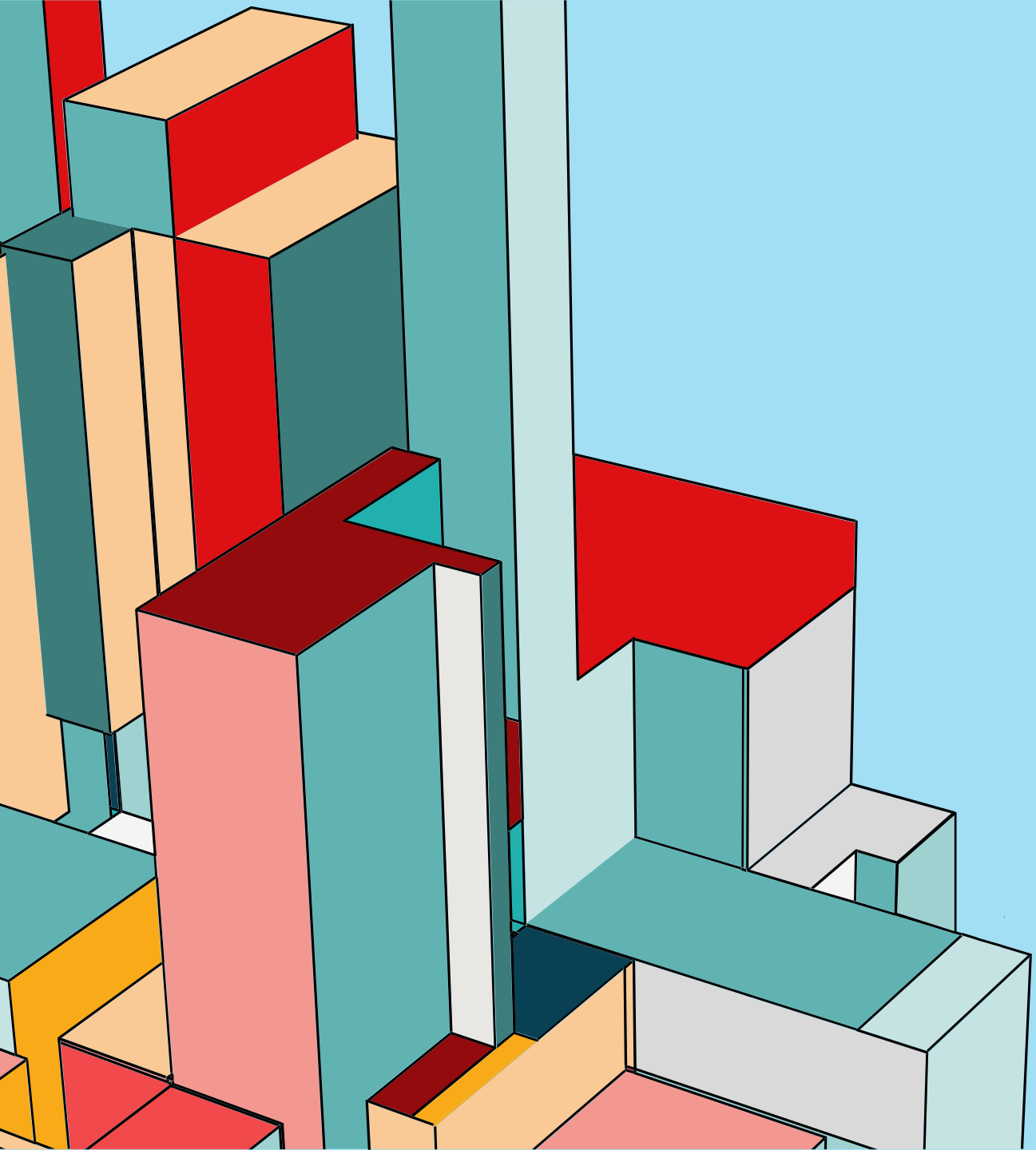
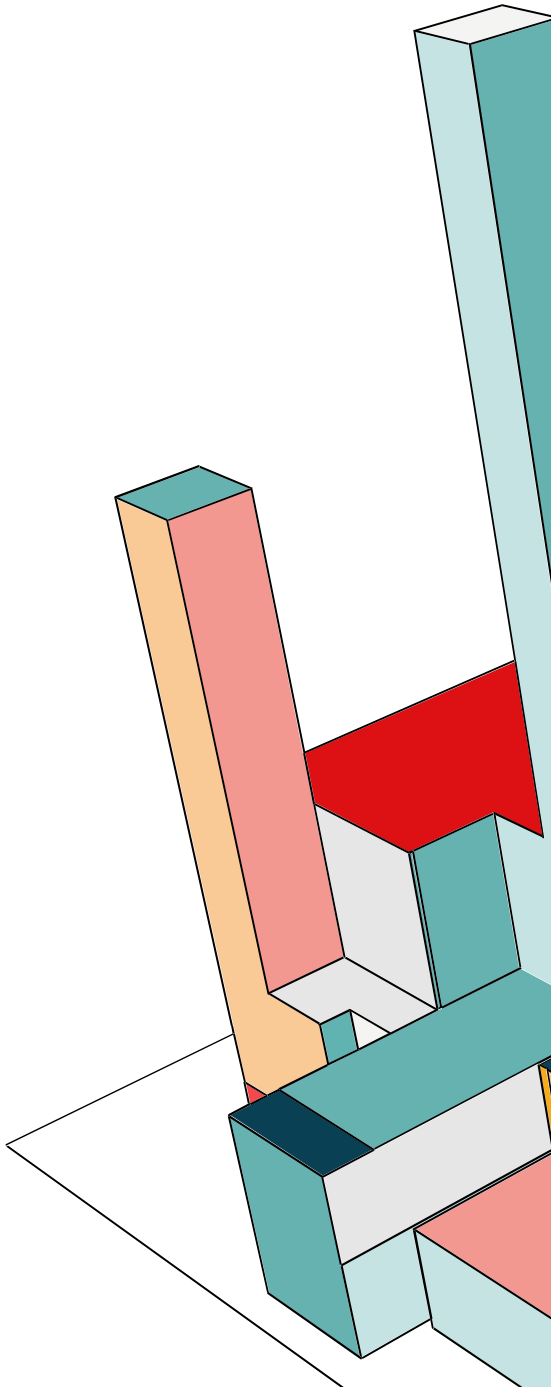


MICROSERVICES USING FLASK & DJANGO (NOT “SERVERLESS”)

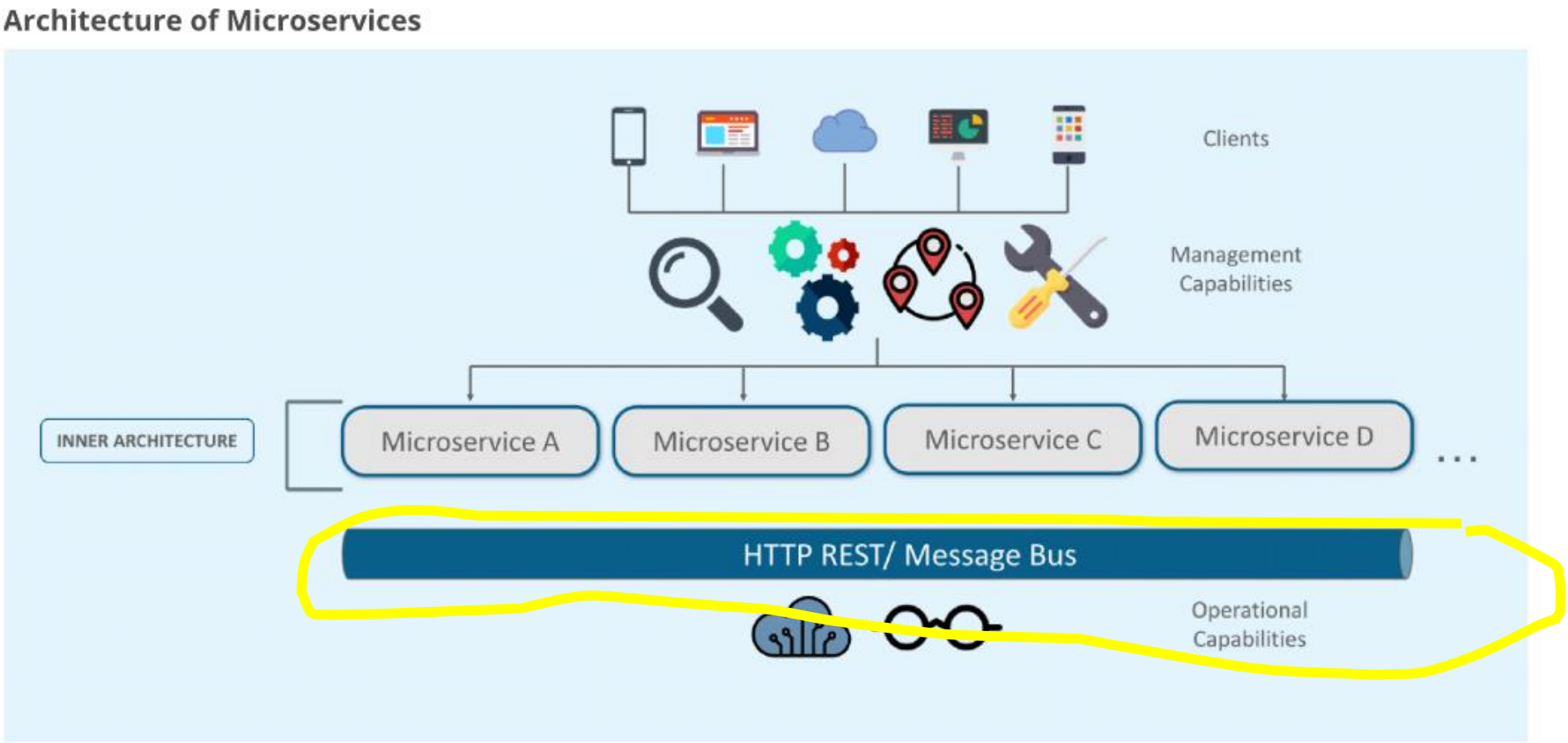
ECE 4150
SPRING 2024
VIJAY MADISETTI

TOPICS THIS WEEK

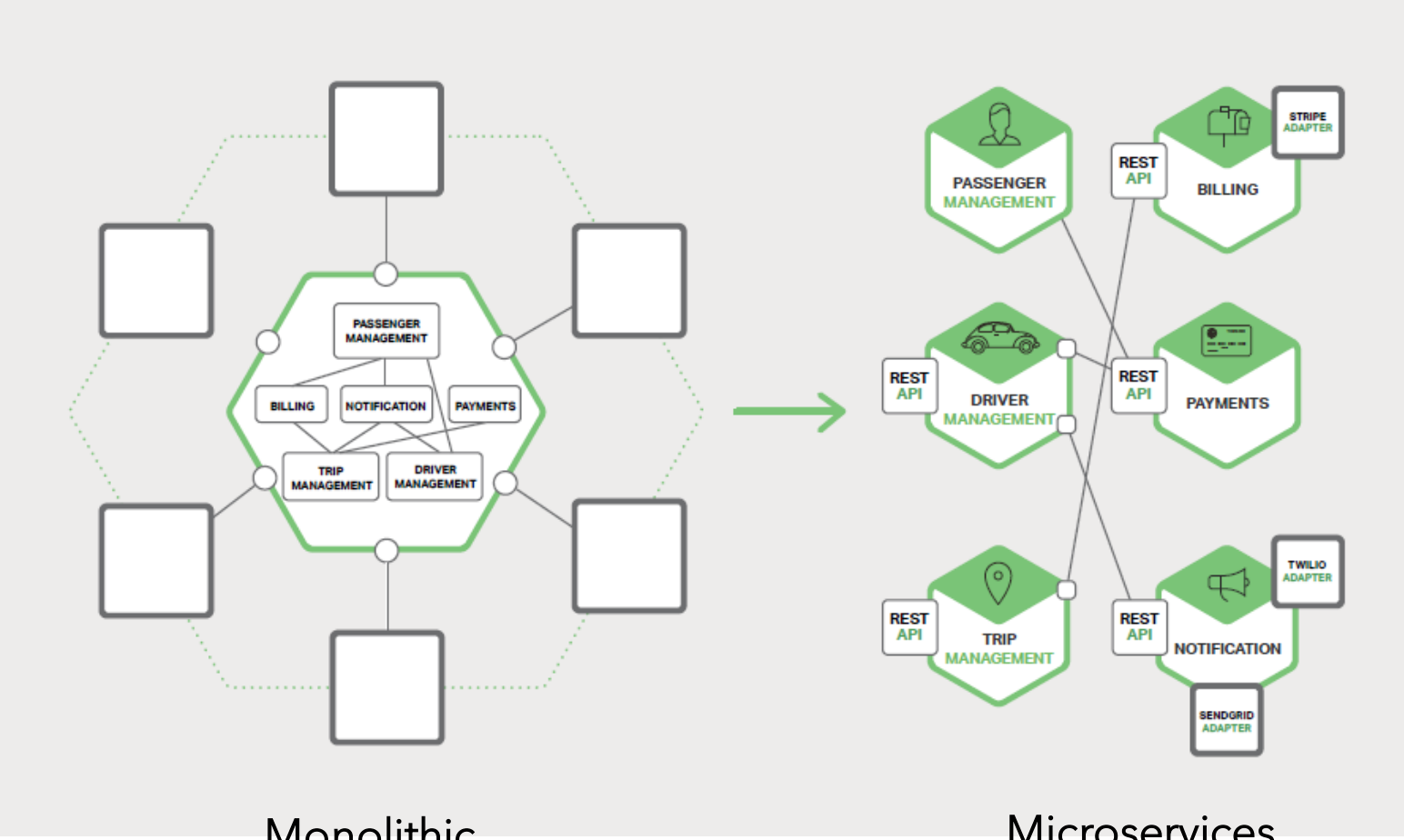
- Microservices using Lambda Services (Serverless)
- Microservices where you use your own servers (Flask/Django)
- Microservices Communications Models (using gRPC)
- Entity Relationship Models (ER Models) for Software
- Conclusions



MICROSERVICES-BASED DECOMPOSITION



HOW DO MICROSERVICES INTERACT WITH EACH OTHER ?



Monolithic

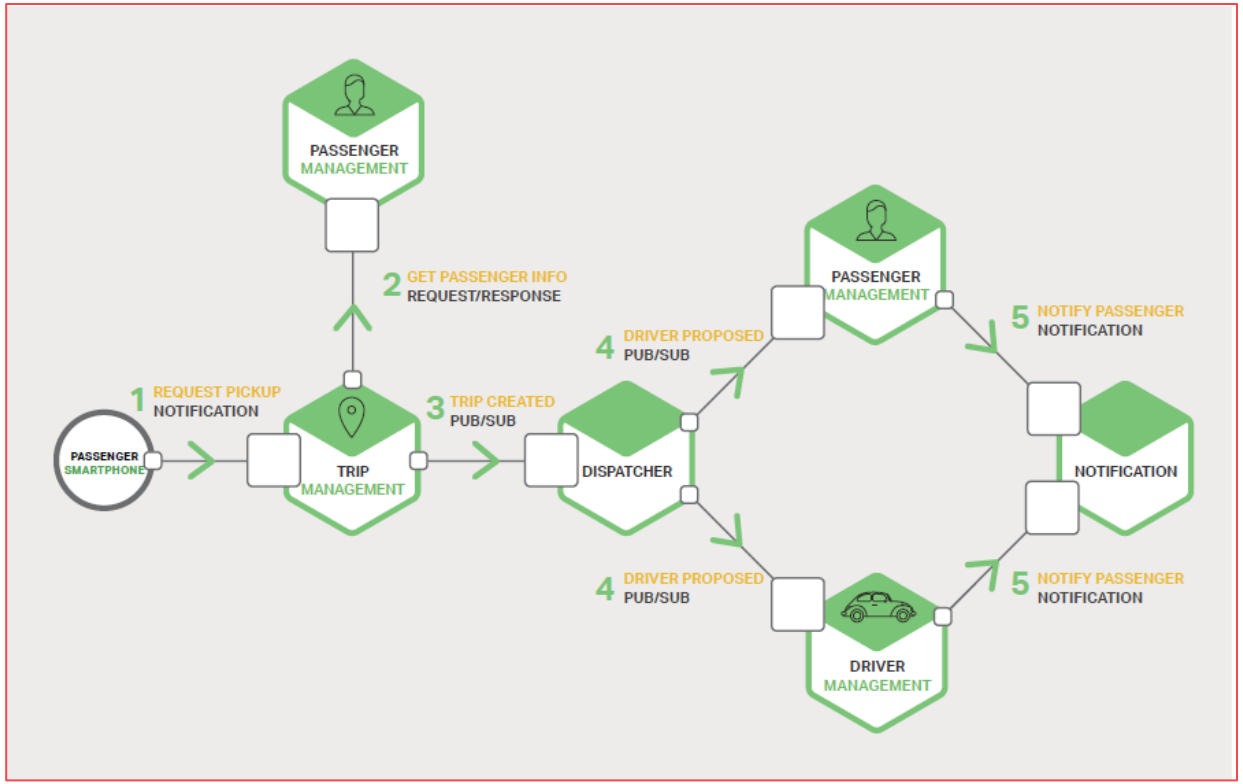
Microservices

MICROSERVICE INTERACTION STYLES

1. Note: REST can be asynchronous. There is no reason a client need wait for a response
2. In REST, it can do other things as well. It is a programming choice. A HTTP web server is usually Request/Response

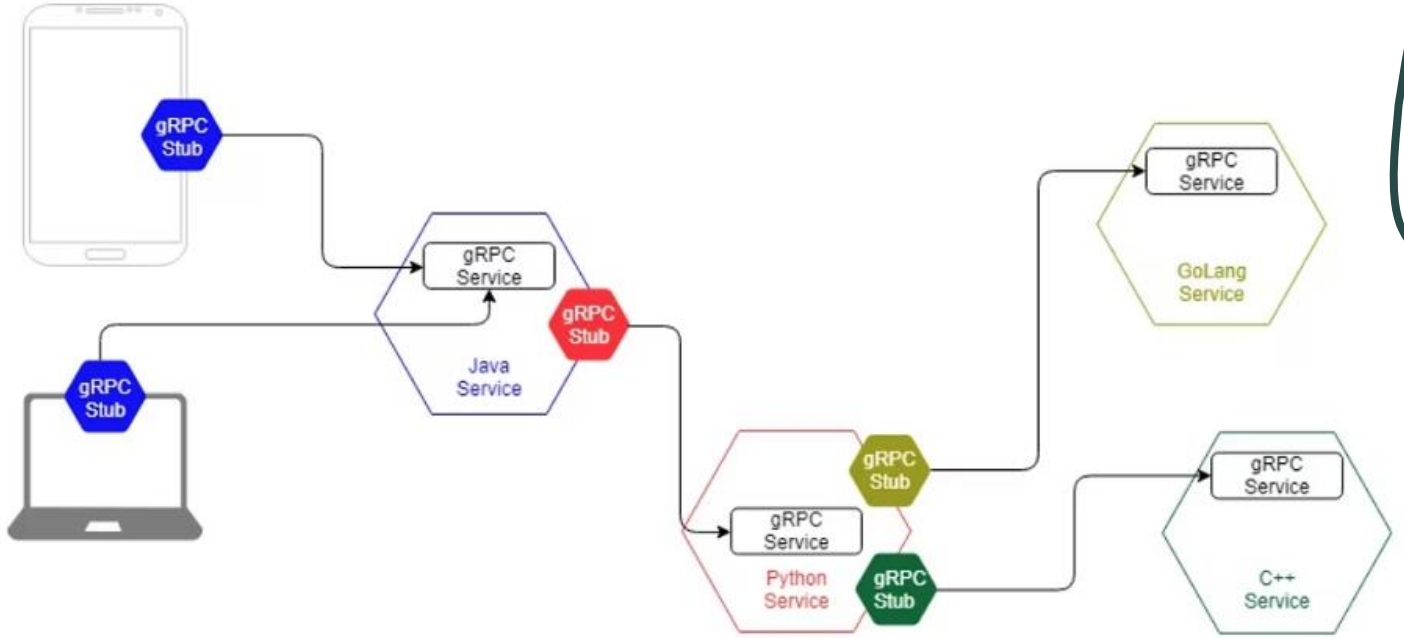
	ONE-TO-ONE	ONE-TO-MANY
SYNCHRONOUS	Request/response	—
ASYNCHRONOUS	Notification	Publish/subscribe
	Request/async response	Publish/async responses

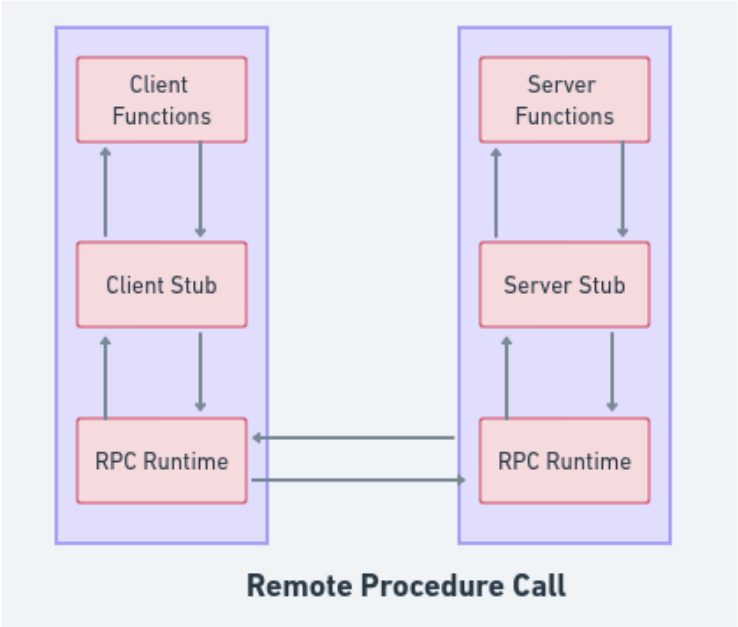
REST-BASED COMMUNICATIONS

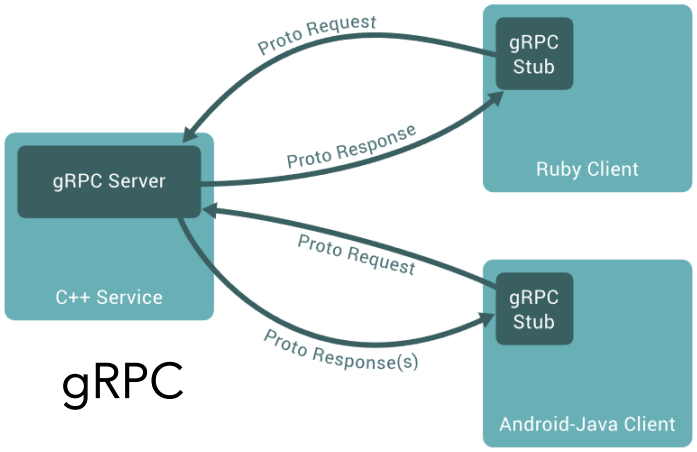


Applying Interaction Styles to Microservices Architecture of Uber-like App

gRPC (ASYNCHRONOUS!) – MORE POPULAR TODAY DUE TO USE BY GOOGLE AND BANKS

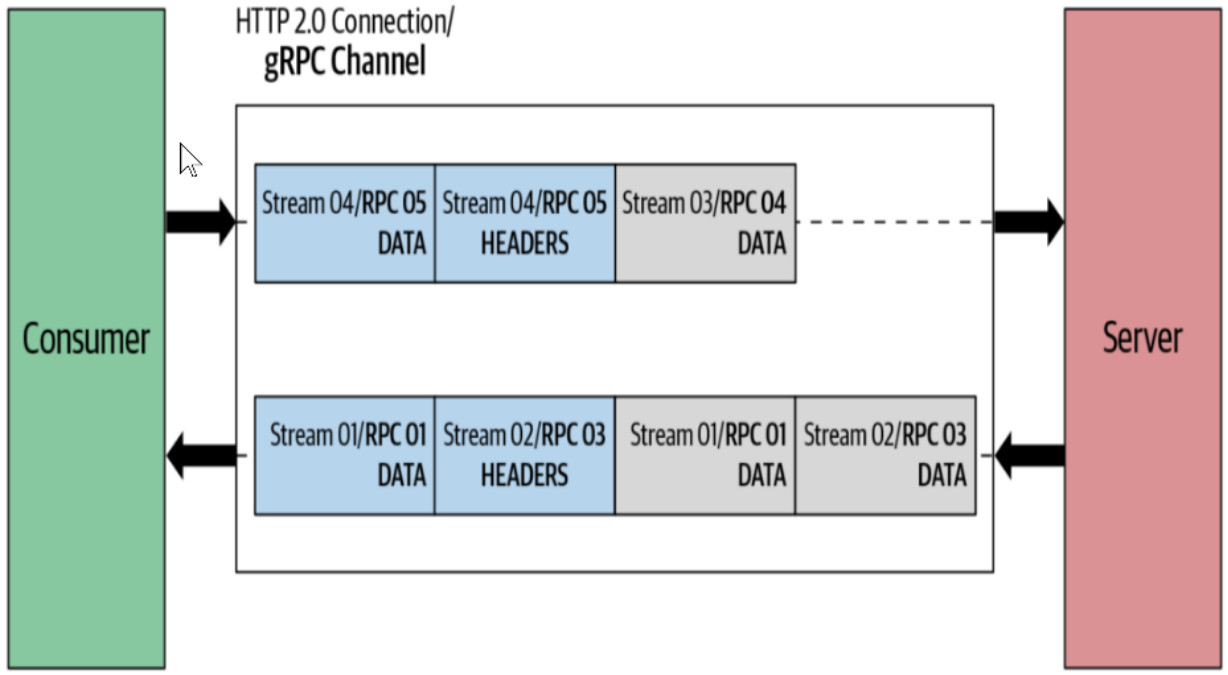






gRPC

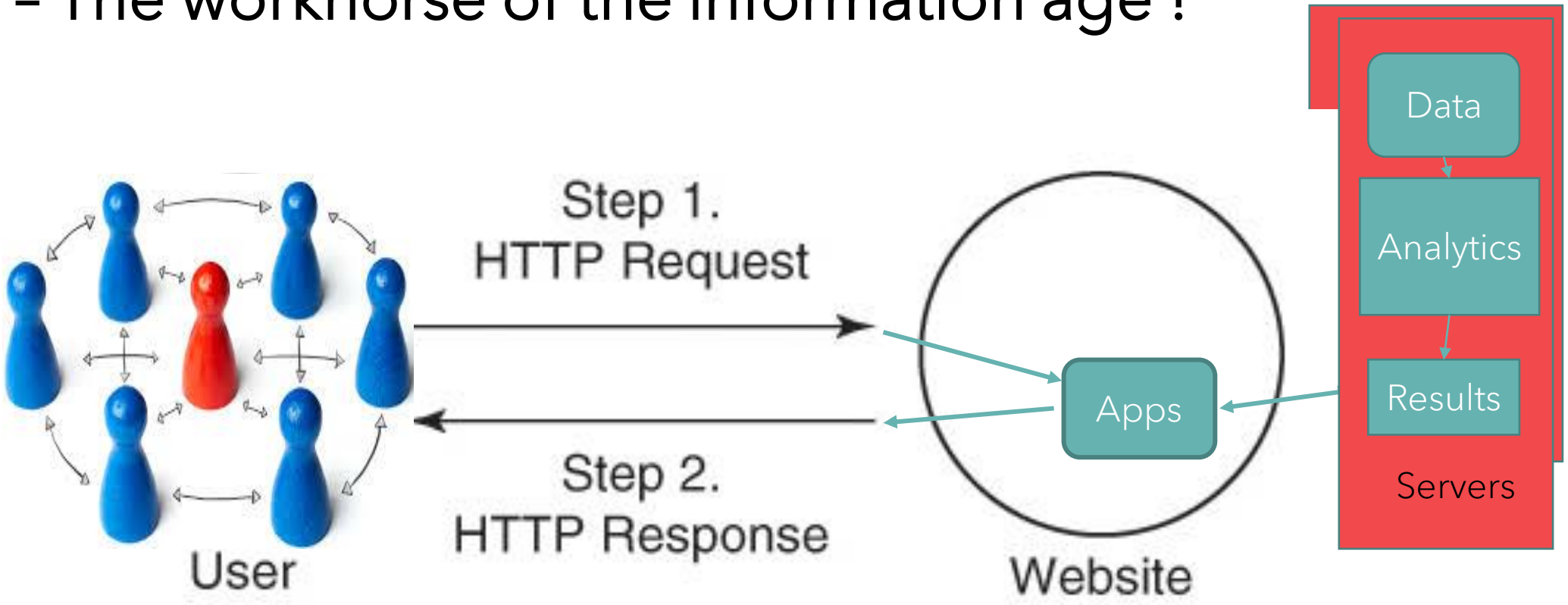
gPRC



1. Unary gRPC – Request and Response (like REST)
2. Server Streaming RPC – Server responds with streams
3. Client Streaming RPC – Client transmits streams and server responds to the batch
4. Bidirectional Streaming – Both client and server send each other streams

WHAT IS FLASK/DJANGO SERVER*FULL* MODEL?

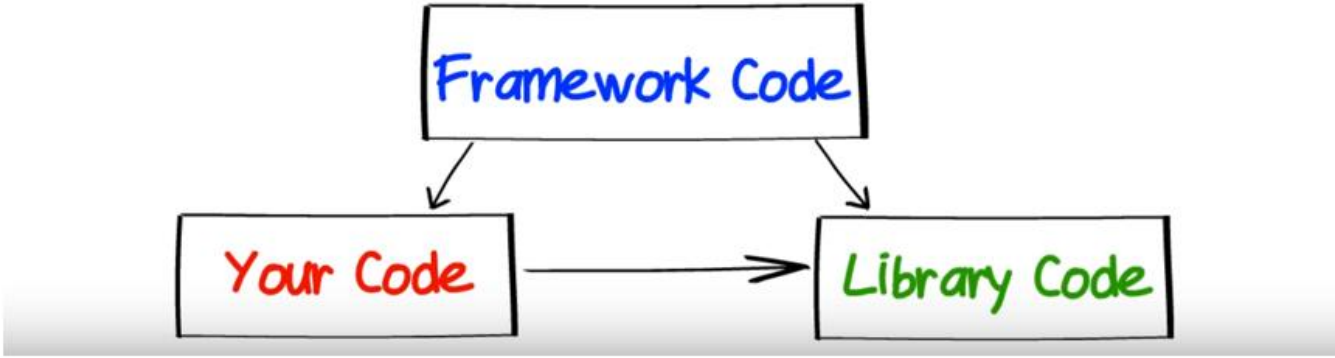
Web-based Cloud Services
– The workhorse of the information age !



Cloud Computing converts a web service to a highly scalable, reliable, available, and on-demand type of service (Well-Architectured Framework)

THE CLOUD OR PLATFORM FRAMEWORK CALLS YOUR CODE

YOUR CODE <u>MUST</u> BE WRITTEN IN A *PRESCRIBED* WAY AND PRESCRIBED STRUCTURE OR ARCHITECTURE PATTERN



Library versus Framework

```
# Using a Library

def my_function(*args):

    ...

    library.library_function(*args)

    ...
```

```
# Using a framework

def my_function(*args):

    ...

    framework.run(my_function)
```

Don't call us, we will call you !

TWO POPULAR OPTIONS FOR PAAS SERVICES FRAMEWORKS ON THE CLOUD

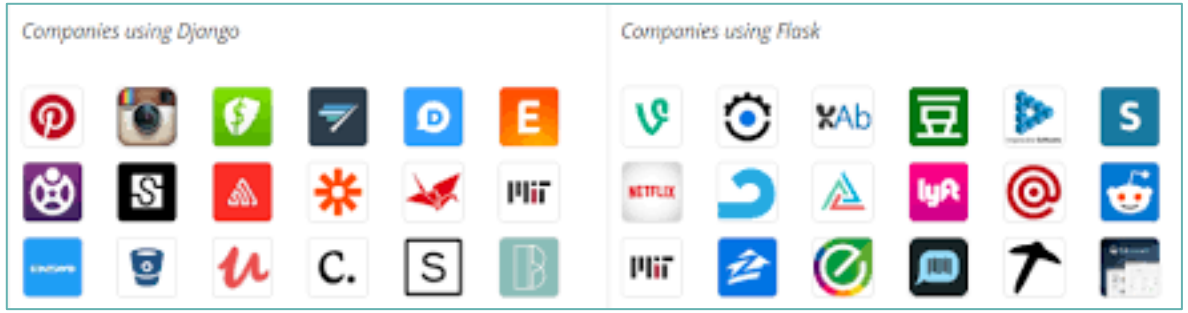
Flask is used in Lab 2

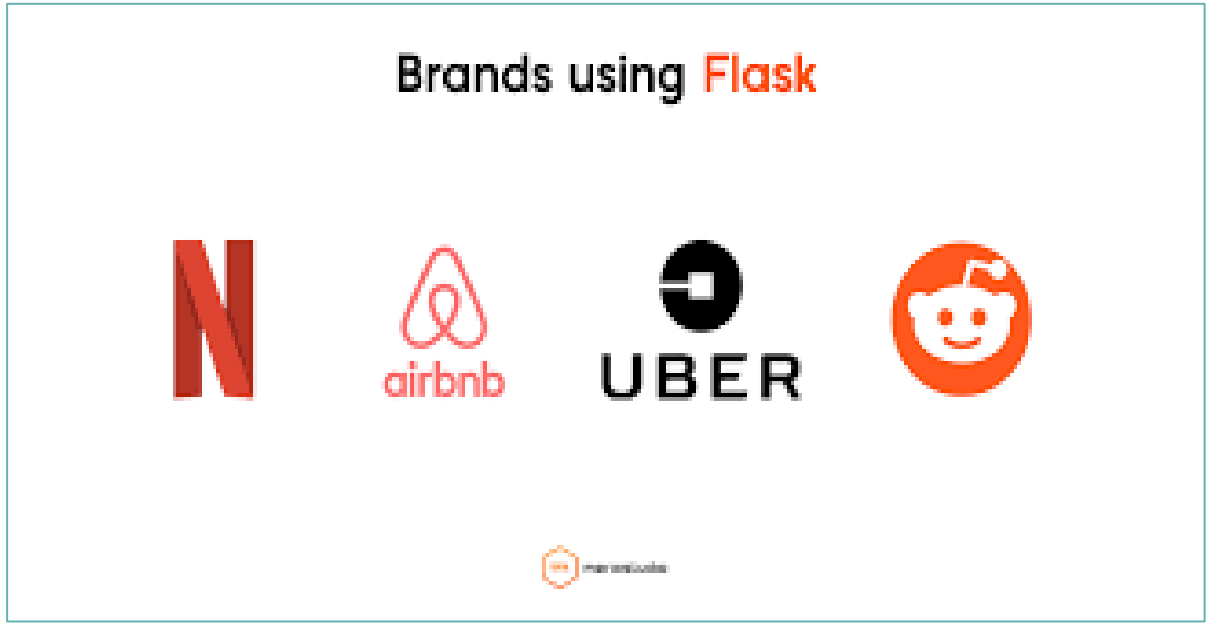
Django is used in Lab 3

Flask

(There are more interesting things than cloud computing!)

CORPORATIONS USING DJANGO & FLASK





TOP COMPANIES USING PYTHON/FLASK/DJANGO

WHAT IS FLASK FRAMEWORK?

```
vkm@LAPTOP-Q109T76F:~$ lynx http://127.0.0.1:5000/product/4150
```

```
vkm@LAPTOP-Q109T76F:~/hello$ more main.py
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hi ECE 4150"

@app.route ('/product/<name>')
def get_product(name):
    return "The product is "+ str(name)

app.run(debug=True)
```

vkm@LAPTOP-Q109T76F: ~

```
   The product is 4150

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

MORE ON FLASK

```
Sun Mar  7 17:59:20 EST 2021
vkm@LAPTOP-Q109T76F:~/hello$ more main3.py
from flask import Flask
from flask import render_template
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hi ECE 4150"

@app.route ('/product/<name>')
def get_product(name):
    return "The product is "+ str(name)

@app.route("/hello")
def hello_many():
    return render_template("hello.txt")

@app.route("/web")
def webmany():
    return render_template("index.html")



app.run(debug=True)
```

```
vkm@LAPTOP-Q109T76F:~/hello$ python main3.py
 * Serving Flask app "main3" (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
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 127-115-971
127.0.0.1 - - [07/Mar/2021 17:55:04] "GET /hello HTTP/1.0" 200 -
```

```
vkm@LAPTOP-Q109T76F:~$ lynx http://127.0.0.1:5000/hello
vkm@LAPTOP-Q109T76F:~$
```

vkm@LAPTOP-Q109T76F: ~

```
Hi there Hi There Great to see you !
```

```
vkm@LAPTOP-Q109T76F:~/hello$ ls
main.py  main2.py  main3.py  main4.py  main5.py  templates  virtualenv
vkm@LAPTOP-Q109T76F:~/hello$ ls templates
form.html  hello.txt  index.html  index2.html  index3.html
vkm@LAPTOP-Q109T76F:~/hello$
```

```
vkm@LAPTOP-Q109T76F:~/hello/templates$ more hello.txt

Hi there
Hi There

Great to see you !
```

MORE ON FLASK

vkm@LAPTOP-Q109T76F: ~/hello

```
import os
from flask import Flask
from flask import request, jsonify
from flask import render_template
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hi ECE 4150"

@app.route ('/product/<name>')
def get_product(name):
    return "The product is "+ str(name)

@app.route("/hello")
def hello_many():
    return render_template("hello.txt")

@app.route("/web")
def webmany():
    return render_template("index.html")


@app.route("/js")
def webmanyscript():
    return render_template("index2.html")

@app.route("/whoareyou",  methods=["GET", "POST"])
def whoareyou():
    if request.method == "POST":
        name = request.form.get("name")
        return render_template("form.html", name=name)

    if request.method == "GET":
        return render_template("form.html")



app.run(debug=True)
```

```
vkm@LAPTOP-Q109T76F:~/hello/templates$ more form.html
<body>
        <p> {%if name %} Hi, {{ name }}. Welcome to ECE 4150 App. {% endif %} <</p>

        <form action="/whoareyou" method="POST">
                <label for=""> Name:
                        <input typ"text" name="name" value="">
                </label>
                <inpout type="submit" value="Submit">
        </form>
</body>
vkm@LAPTOP-Q109T76F:~/hello/templates$ date
Sun Mar  7 18:22:49 EST 2021
```

```
vkm@LAPTOP-Q109T76F:~/hello/templates$ lynx http://127.0.0.1:5000/whoareyou
```

vkm@LAPTOP-Q109T76F: ~/hello/templates

```
Hi, Vijay Madisetti. Welcome to ECE 4150 App. <
Name: _____
```

AWS AND FLASK

https://aws.amazon.com/getting-started/hands-on/serve-a-flask-app/

aws

Contact Us Support English My Account Create an AWS Account

Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Customer Enablement Events Expl

Step 1: Prerequisites

Complete the following prerequisites before you get started with your Flask app.

1.1 — You need an AWS account and must install Docker, the AWS Command Line Interface (CLI) tool and the Lightsail Control (lightsailctl) plugin on your system. Follow the provided links if you don't have some of those.

Sign up for AWS

Already have an account?
Log in to your account

Step 2: Create the Flask application

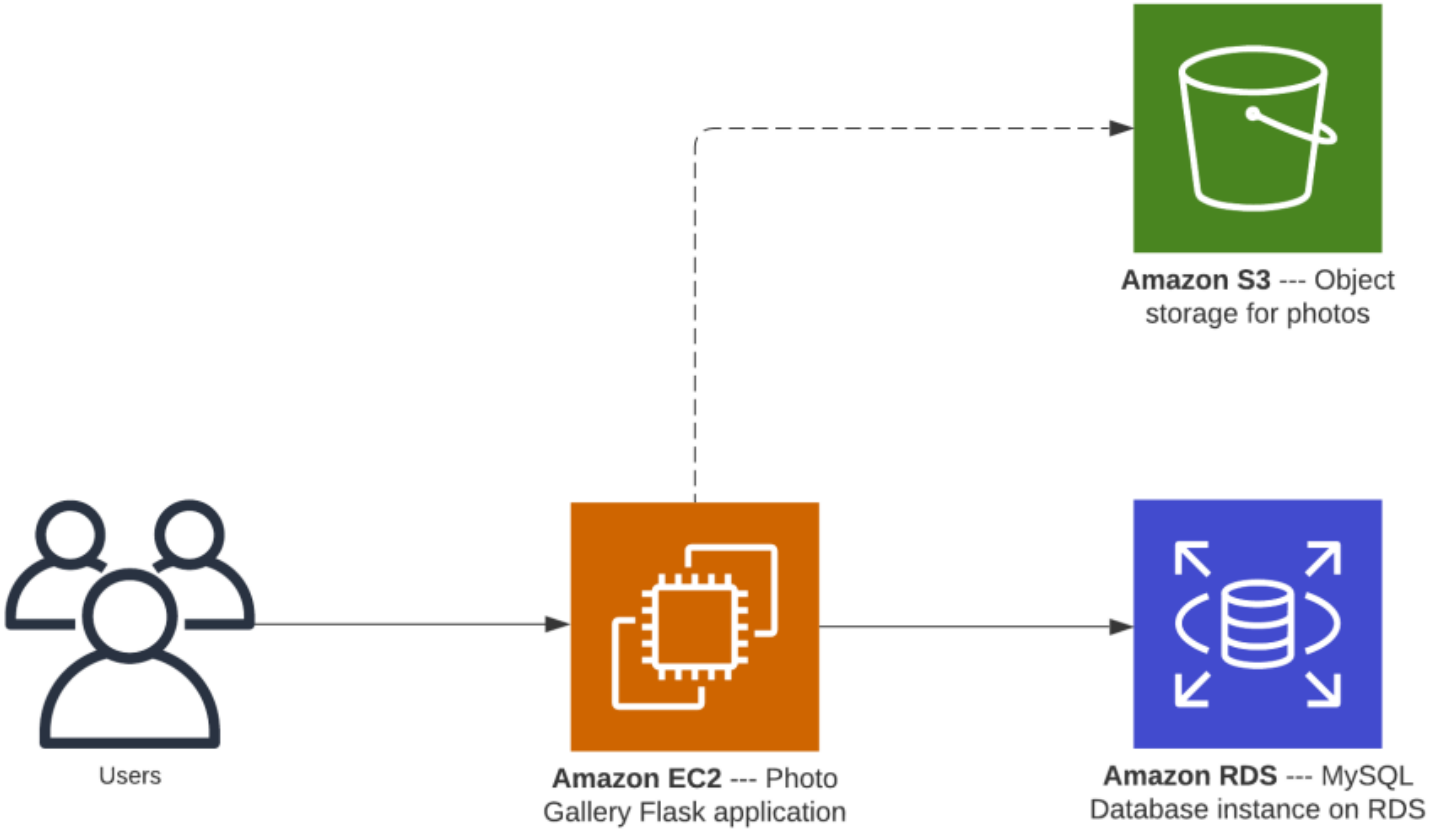
Complete the following steps on your local machine that is running Docker. These steps walk you through the process of creating the Flask application files.

2.1 — Create a new project directory and switch to that

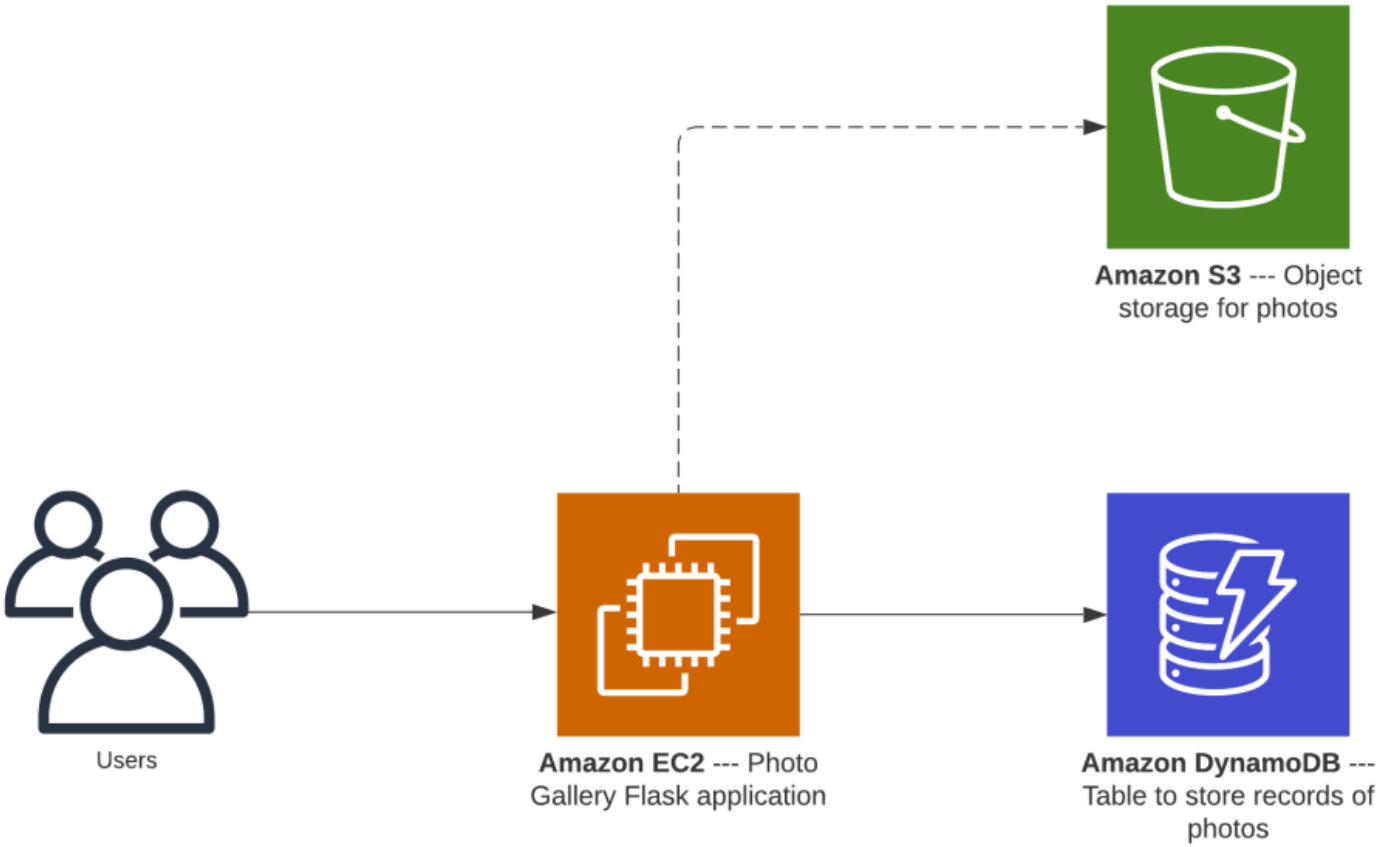
Bash

FLASK DEMO

LAB 2 USES FLASK (SQL AND NOSQL VARIANTS)

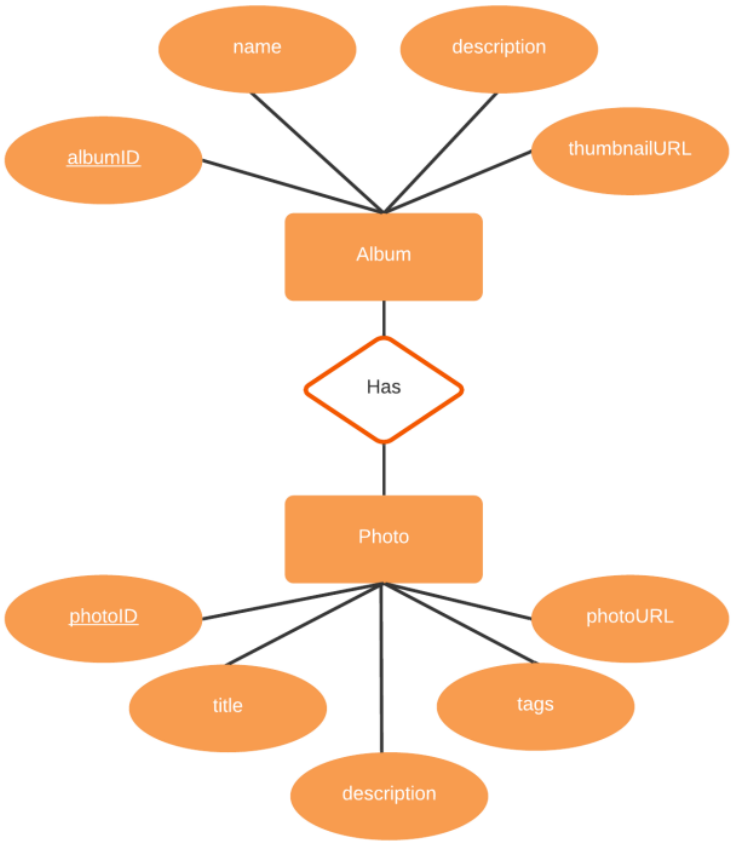


LAB 2 USES FLASK (SQL AND NOSQL VARIANT)



No payment to Amazon for every click on a lambda microservice!

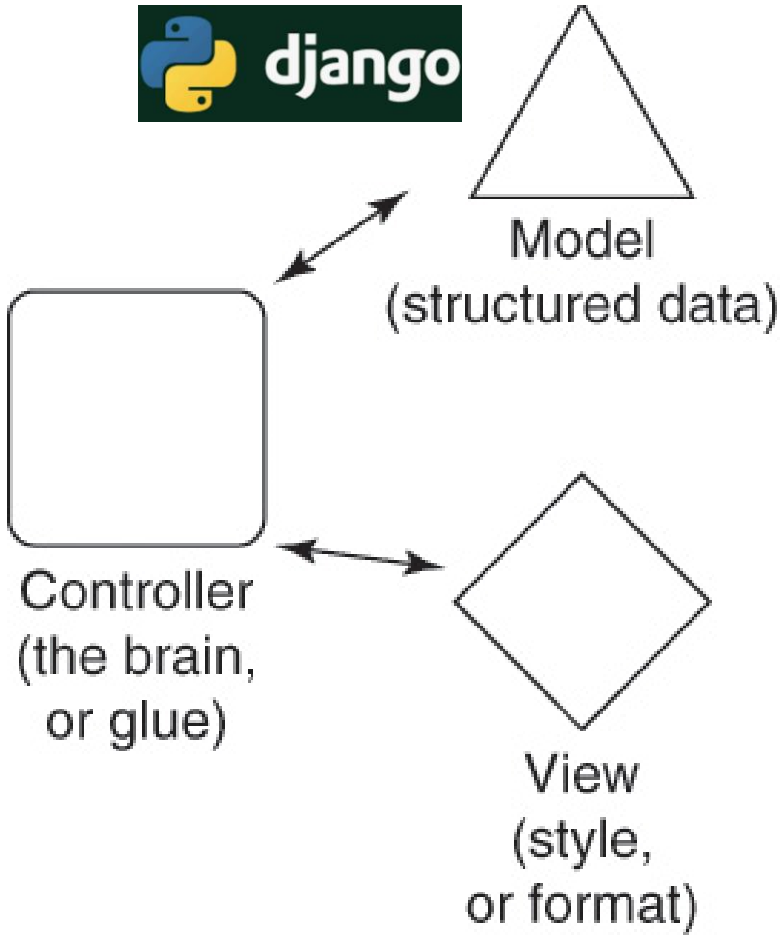
WHY ARE ER DIAGRAMS USED?



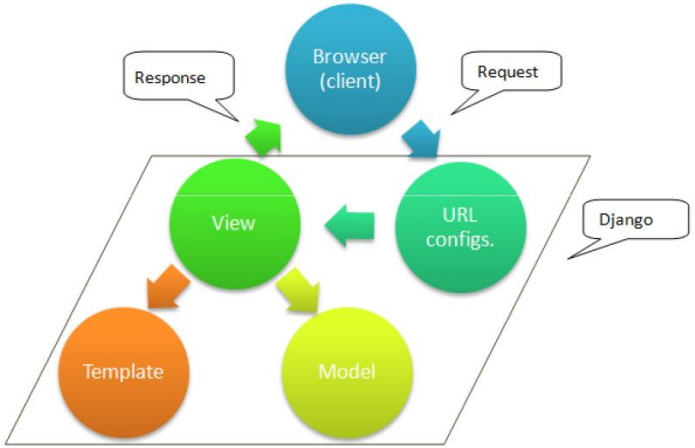
Entity Relationship Diagram that show the relation between albums and photos

MODEL VIEW CONTROLLER (MVC)

Django Reinhardt at the Aquarium jazz club in New York, NY



SUMMARY



We have been looking at "serverful" microservices – Flask

We will look at Django

We will also look at ER data modeling