Google Developer Student Clubs

Back-end Development

Intro to AWS and Flask

Rough Plan (please add to this, it is just some rough ideas I had)

Me:
intro to AWS slides

- What it is
- why should you learn about it
- AWS services
- pricing
- what is RDS

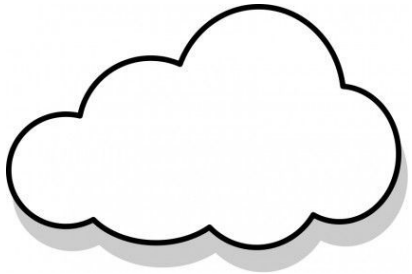
- Setting up an RDS database
- me connecting to it
- set up AWS user and permissions for Milind to connect

Milind stuff

- slides talking about what flask is, what it's used for, flask basics, quick recap on HTTP methods (don't really need to go into detail just maybe say what for example a POST request is used for generally and give an example of a complete POST request (an example payload, and response) , maybe just list out all the SQL commands we are going to use ahead of time on the slides just so they don't get overwhelmed when they see them in the code). They should have some exposure to SQL because of the WISC sql workshops though), etc
- Make a REST api with endpoints GET api/articles/:id, GET api/articles/, POST api/articles, /PUT api/articles/id (let's use 'articles' instead of posts to make it less confusing (HTTP post vs an actual post)
  - a post consists of three things: title, body, author (let's avoid using images to avoid unnecessary complexity)
- push your stuff to a git repository

What is the cloud?

- Basically just a bunch of servers that can be accessed through the internet
- For example.
  - Ex. you could be storing documents in the cloud if you use OneDrive/Google Drive/etc
  - If you run your python program on Google's servers, you can say that you are running your python program in the 'cloud'



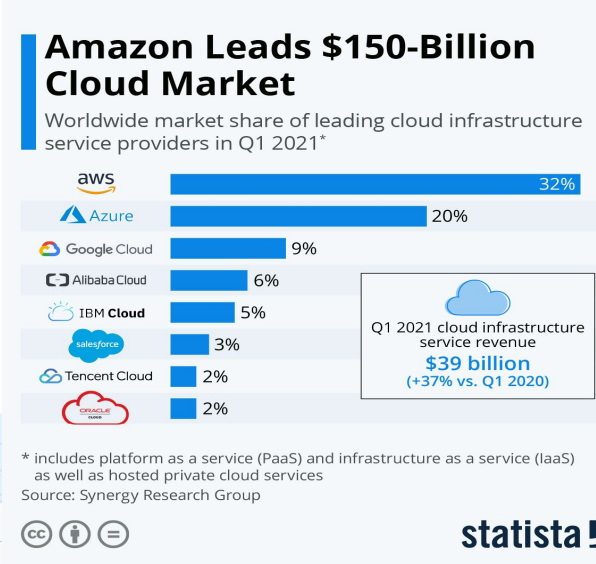
Google Developer Student Clubs

What is AWS?

- Amazon Web Services (AWS) is a **cloud service provider**
- on-demand delivery of IT resources over the internet
- Examples of it's services include:
  - Elastic Compute Engine **(EC2)** - *virtual machine*
  - Relational Database Service **(RDS) -** *database*
  - Amazon S3 - *cloud object storage*
  - Over 200 more!
- Anything your app needs, AWS probably has a service to help you!
  - youtube.com/watch?v=JIbIYCM48to

Why learn AWS?

- AWS is the leading cloud service provider in the world
- Used by companies like Netflix, Sony, Disney, Nasa, Epic Games, Reddit, etc .
- It's an extremely useful skill to have on your resume!



Google Developer Student Clubs

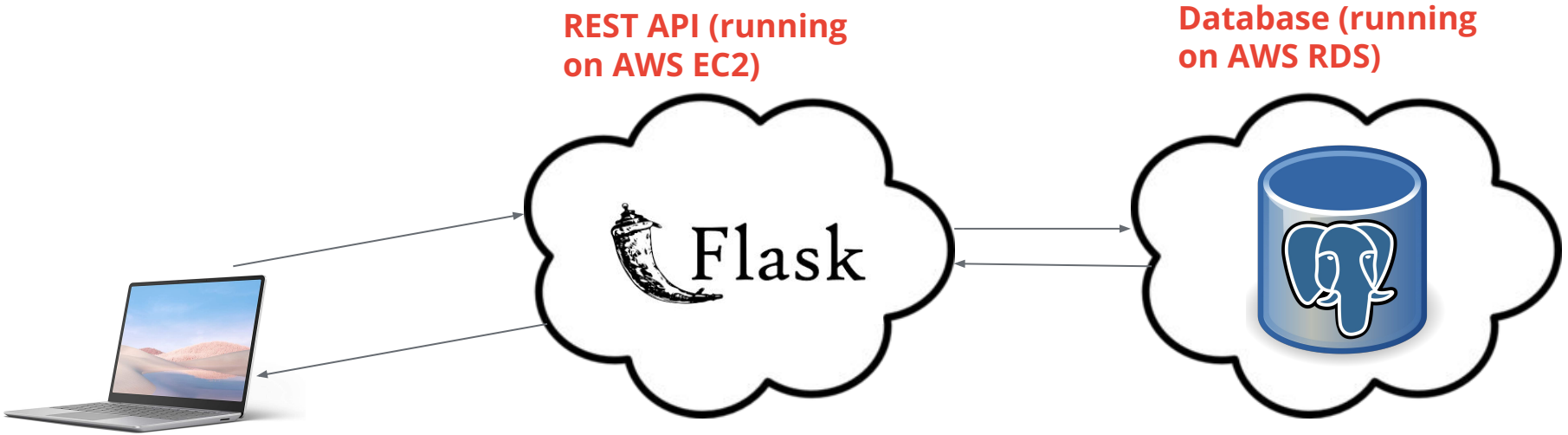
source:https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/:

Benefits of using the cloud

From a software engineering POV

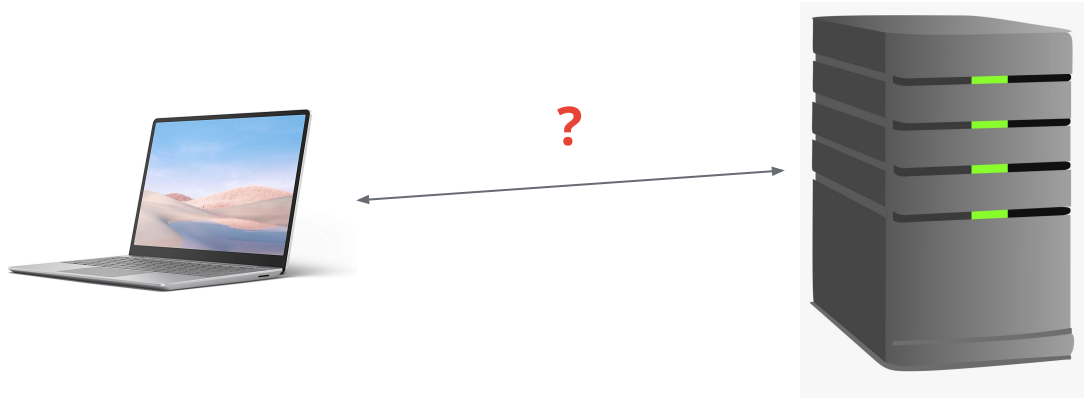
- Trade capital expense for **variable expense**
- **Don’t need** to know how much **hardware** you need upfront
- Increase **Agility** ex. only takes a couple clicks to launch a website
- Extremely easy to deploy your products **globally**
- Benefit from massive economies of scale -> **costs less** to do stuff in the cloud!
- **Don’t** need to **maintain hardware** anymore!

What we'll be doing in today's workshop



What is an API

- “A software intermediary that allows two applications to talk to each other.”
- How can the front-end get data from the back-end?



What is an API? continued

What's an API's purpose?

- We *could* let the front-end deal with all the complexity of retrieving the data
  - i.e connecting to database, querying the database, structuring/filtering the data that came from the database, etc

What is an API? continued

What's an API's purpose?

- Better way:
  - Front-end send requests to specific 'endpoints' on the back-end and waits for data to be sent back
    - Ex. http://some-server**/api/cats**
    - Ex. http://some-server**/api/users/2**
  - Let the back-end deal with all the internal complexity!
  - Similar to how we create public interfaces with classes in languages like Python/Java/etc (users need to know what's going on internally in class, we can just use its methods and attributes)

For example...

- Say I want to retrieve some data about all the users of my application
- I can just send a GET request to **http://my-server/api/users** and I will get back some data like this:
-

```
[
        {
                first_name: “Bobby”,
                last_name: “Daniels”
        },
        {
                first_name: “Paul”,
                last_name: “Smith”
        }
[
```

HTTP Request Methods

and when to use each

GET: *Retrieve* a resource (some data) on the server

POST: *Create* a new resource on the server

PUT: *Update* a resource on the server

DELETE: *Delete* a resource

etc.

Quick Demo of what we will be making

AWS Services that we will be using today to make it

AWS EC2

EC2 = Elastic Compute Engine

- A virtual machine that runs on AWS' physical servers
- You can run any OS you want on it, including Ubuntu, Amazon Linux, MacOS, WIndows, etc.
- You can upgrade an EC2 instance to whatever specs you'd like (including memory, cpu, storage)
- Examples of what you can do:
  - Run demanding tasks (maybe training a ML model, performing large scale data analysis, etc)
  - Host a minecraft server
  - Do (almost) anything you could with a normal computer!

AWS RDS

RDS = Relational Database Service

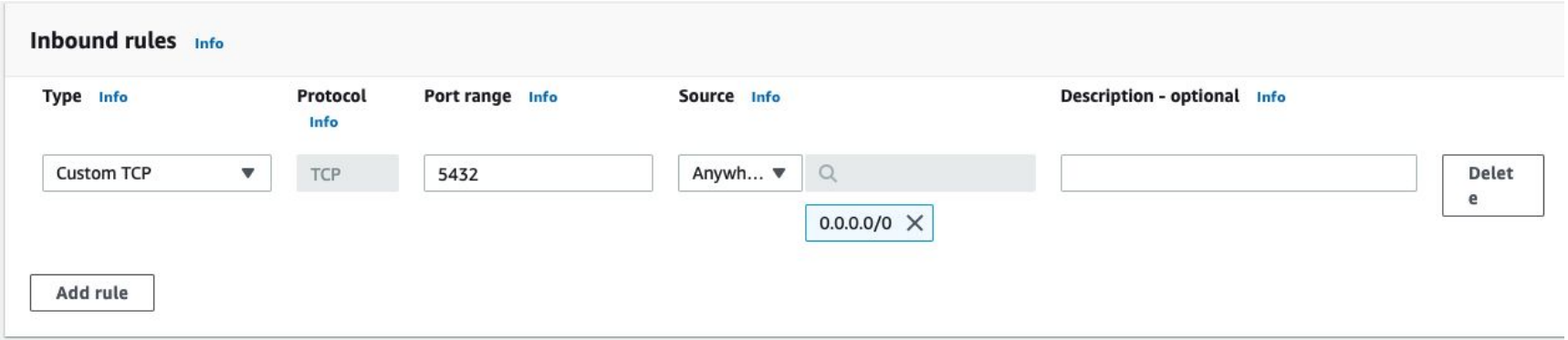
- A managed PostgreSQL/MySQL/Oracle/etc data
- Some cool features of this service
  - Offers automatic database backups
  - Auto-scaling (the database can scale up and down in terms of its processing power depending on the current demand)
  - etc.

Additional AWS Topics

AWS Security Group

A security group acts as a **firewall** that controls what gets **in and out** of your EC2/RDS/etc instances.

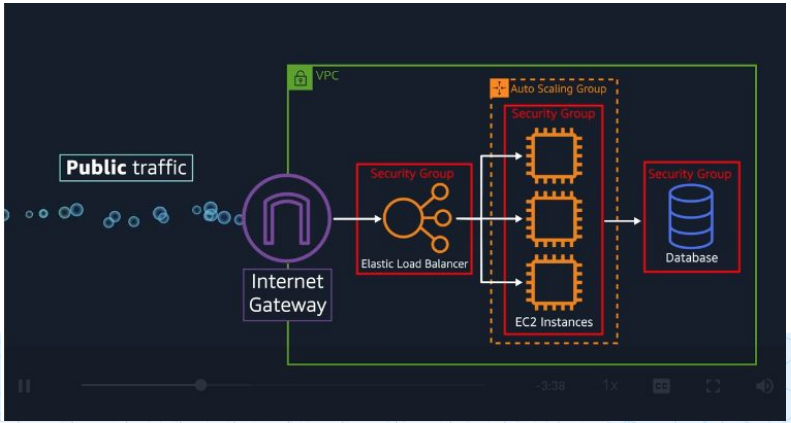
Example:



AWS VPC

VPC - Virtual Private Cloud

- Your own private network within AWS
- A logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define
- Don’t worry about this for this workshop!



Google Developer Student Clubs

AWS RDS Pricing

[AWS Free Tier]

Let’s set up a Database on AWS!

Python Flask

Flask is Python based lightweight framework used to build web applications.

We will be using Flask to make an API today.

API (Application Programming Interface) is an application that provides an interface so that the database and the frontend can interact. API uses **HTTP request** to communicate with frontend.

HTTP Request

200 OK: The response has succeeded!

201 Created: The request has succeeded, and the resource has been created (usually for POST)

400 Bad Request: The server could not understand the request due to invalid syntax

401 Unauthorized: The client is not allowed to get the requested response

404 Not Found: The server cannot find the requested resource

418 Unprocessable Entity: The server unable to process the contained instructions.

500 Internal Server Error: The server has encountered an issue while processing your request

Making a request

We can make different types of requests: GET, POST, PUT, PATCH, etc

Example:

```
POST /articles
{
        author: ‘gdsc’,
        title: ‘welcome to apis’,
        body: ‘blah blah blah’
}
```

Flask SQLAlchemy

Allows you to use ORM (Object Relational mapping), which provides an interface for using OOPs to interact with database.

This allows you to not write raw SQL

db.Model	Ability to create and manipulate models/tables
db.session	Ability to create and manipulate transactions/queries
db.session.add(person)	It is used as an insert SQL Query
db.session.commit()	It is used to run the transaction

Model and Serialization

Model is a description of the database table in ‘class’ form. This helps SQLAlchemy map the database table to an object.

Serialization is converting the object in a more readable form.
For example:
<Article 1> is an object of article but you don’t want to send the object itself so you convert this object to something like:

```
{
    title: ‘welcome to gdsc’,
    body: ‘blah blah blah’
}
```

Useful functions

MyModel.query	Used to return the query table
MyModel.query.filter_by(<expression>) **eg : name = 'Milind'**	Used to filter the query object and return only the one/many you asked for
MyModel.query.filter(<expression>) **eg: Person.name ="Milind", Team.name = "Pod7"**	More flexible to use like filter_by method
MyModel.query.first()	Used to get the first result of a query/list of responses
MyModel.query.all()	Used to get all the results of a query
MyModel.query.limit(<number>).all()	Used to get a limited number of response

Note that "MyModel" is just some arbitrary model that we made up for this slide

Basics

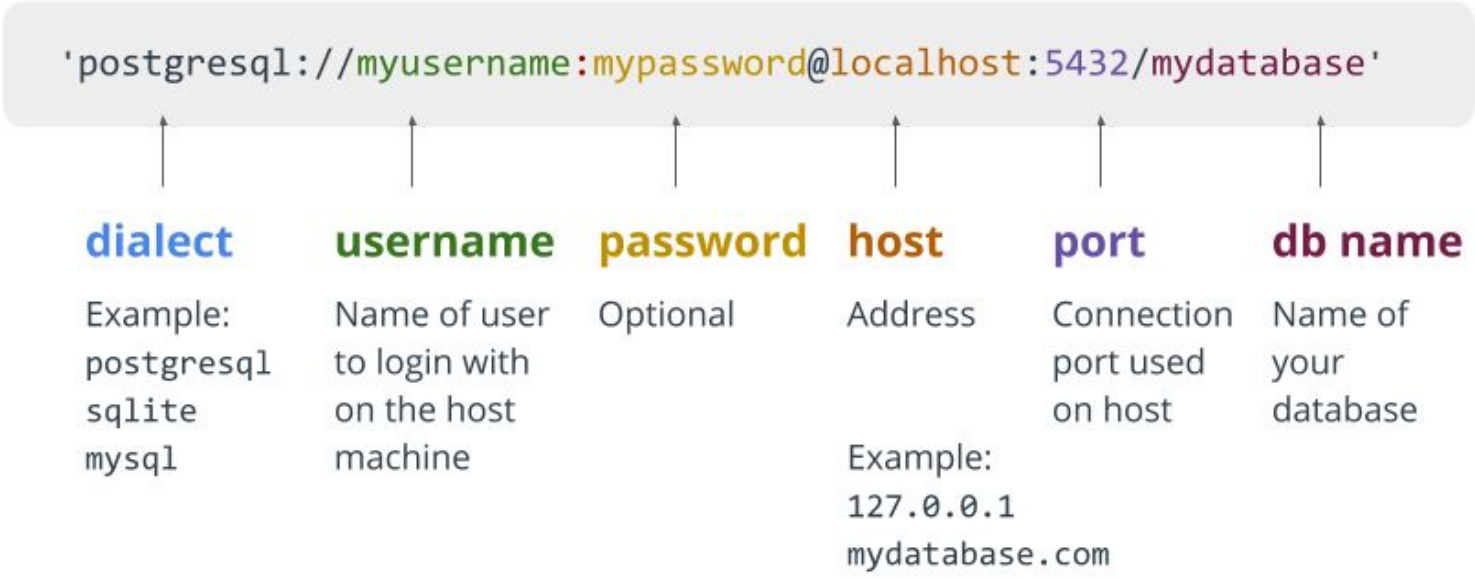
api.py (used to create the endpoints)

```
# Initializing the application
app = Flask(__name__)

# Connecting to the postgres server
app.config['SQLALCHEMY_DATABASE_URI'] = f'postgresql://{user_name}:{password}@{host}:5432/{database}'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

# connecting the db with the application
db.init_app(app)
```

model.py (used to describe the database)

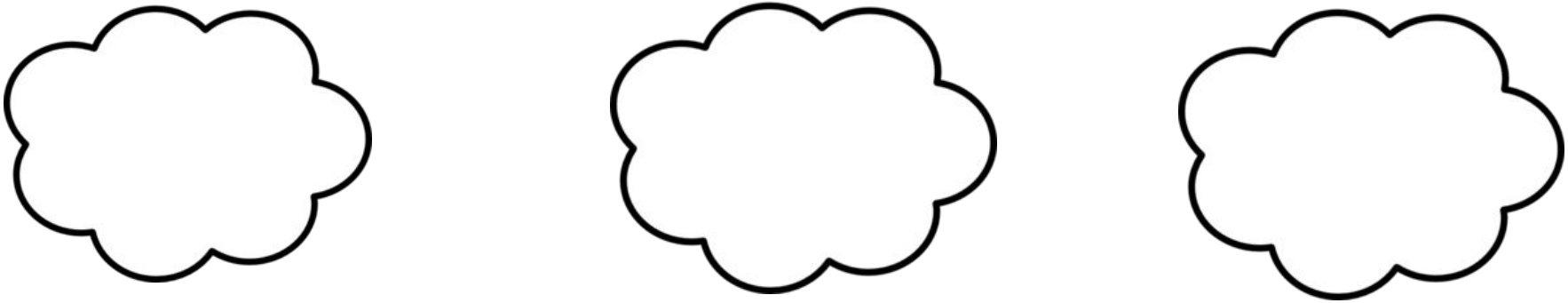
```
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()
```

Connecting to Database



Let’s create an API !!

IF YOU START UP ANY AWS RESOURCES, PLEASE REMEMBER TO DELETE THEM WHEN YOU ARE DONE USING THEM. WE ARE NOT RESPONSIBLE FOR ANY FINANCIAL LOSSES if you forget to do this :)))



Thanks for coming today!