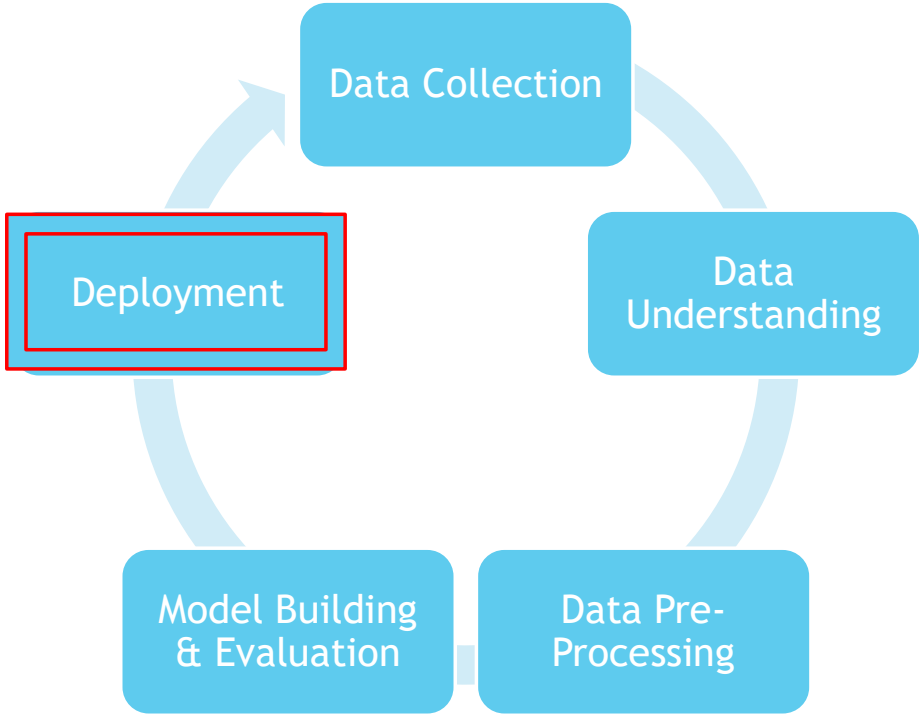
API

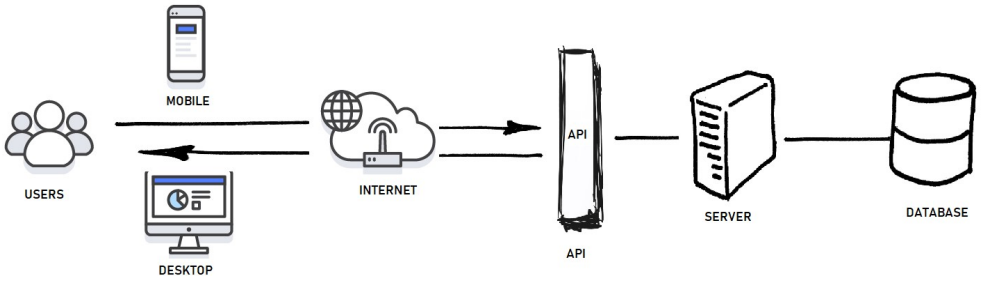
DataDNA

Typical Data Science Cycle



Introduction

- API (Application Programming Interface) is a connection between two applications/services.
- These are used to manipulate/fetch data to/from a online server using some requests by external script.
- API requests act like a mediator between client and server.



- Web APIs are different than Web-apps :

1. Web-apps are like websites where user can open it in browser and interact it but APIs return only data not views.

2. API has an endpoint set, which can be hit by other systems to get data which it provides. So APIs are meant for system to system interaction but web-apps are made for human interactions.

3. Web-apps can use APIs to retrieve information from external servers.

DataDNA

Introduction

Need of API:

1. Users need to **access** data in **real time**, such as for display on another website or as part of an application.
2. Data **changes** or is updated **frequently**.
3. If data size is huge and users only need **access to a part of the data** at any one time, then APIs are used. But if we have small data size, then we can try other traditional methods to transfer complete data altogether.
4. Users will need to perform actions other than retrieve data, such as contributing, updating, or deleting data.
5. Providing data through API keeps **database safe** as users don't directly access complete database.

DataDNA

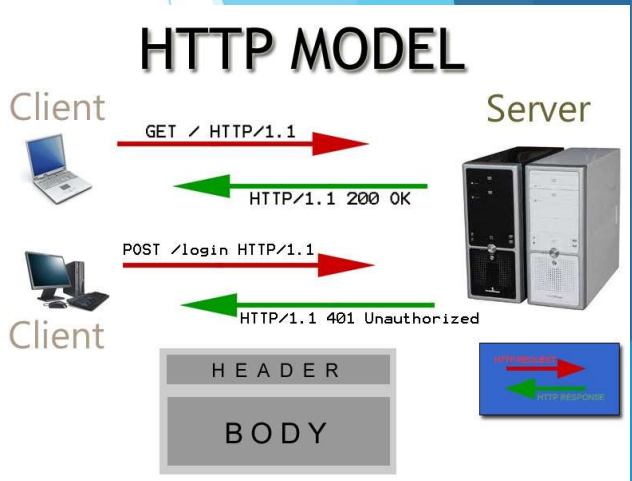
Components

HTTP:

- HTTP (Hypertext Transfer Protocol) is the primary means of **communicating data** on the web.
- HTTP implements a number of “methods,” which tell which **direction data** is moving and what should happen to it.

Following are the key http methods :

Purpose (CRUD)	Request	Details
Create	POST	Sent data to server (add a new detail)
Read	GET	Get data from server or retrieve information
Update	PUT	Update/modify an existing entity
Delete	DELETE	Remove file from server



Components

URL :

- URL (Uniform Resource Locator) - An address/reference for a resource on the web

- A URL consists of a protocol, domain, and optional path.

  - Protocol (http:// or https://)
  - Domain (amazon.in)
  - Optional path (/home).

- A URL describes the **location/address** of a specific resource such as a web page.
- In context of APIs URL, request, URI, or endpoint are used interchangeable .
- These url can be **accessed** either through **browser** or from tool such as **python IDE**

Components

JSON:

- JSON (JavaScript Object Notation) is a text-based data format that is designed to be easy to read for both humans and machines. Efficient to use.
- JSON is generally the most common format for returning data through an API, XML (Extensible Markup Language) being the second most common.

REST:

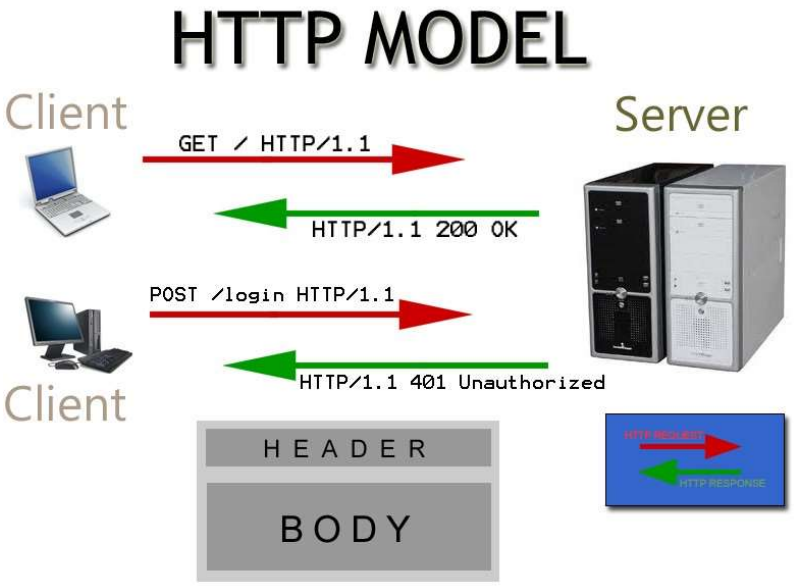
- REST (Representational State Transfer) are set of **guidelines** that describes some best practices for implementing APIs to provide some advantages such as low internet bandwidth, reuse responses by enabling cache responses.
- A web API that obeys the REST guidelines is informally described as REST API.

Components

Requests Status Code:

Requests status code are 3 digit code that tells us status of http requests.

Status Code	Description
1XX	Informational message such as request has been received and processing
2XX	Successful operations such as 200
3XX	Redirection means further actions requires
4XX	Client Error such as 404 tell that client is entering invalid URL, 401 tells unauthorized operations
5XX	Server side Error such as error in api output



Some Public APIs

1. https://datausa.io/api/data?drilldowns=Nation&measures=Population : Population data
2. https://open.er-api.com/v6/latest/USD : USD exchange rate
3. https://v2.jokeapi.dev/joke/Any : Joke API
4. https://www.thecocktaildb.com/api/json/v1/1/search.php?s=margarita : Cocktail recipe

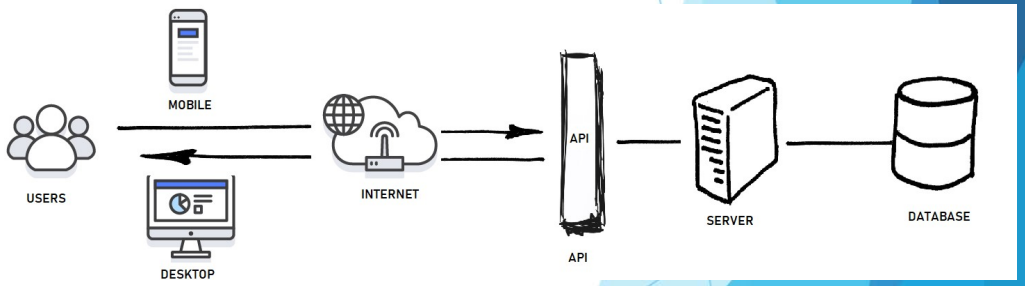
Test through Postman/RestMan extension and python

API using Flask

- Flask is a **web framework** for Python, meaning that it provides functionality for building web applications, including managing HTTP requests and rendering templates.

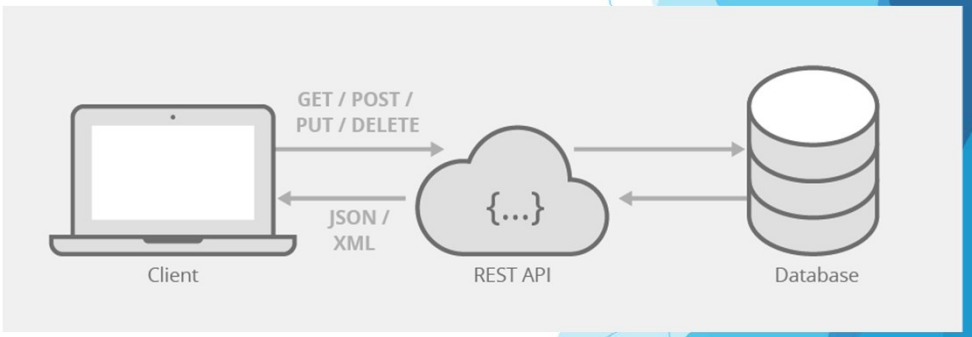
- Flask maps HTTPS requests (requests send by users using any URL) to some associated functions also called **routing** and return results based upon definition of **function**.

- This returned output could be a data or it could be a web page by rendering an **html template**.



API using Flask

- We can create flask web-apps that can contain APIs as well.
- These APIs can be created similar way except in APIs we **return** a **json data** rather than rendering any html template.
- API endpoints will be URL which is mentioned in @app.route.
- Data is returned in json format.



DEMO : flask_basic_api.py

- Data can be fetched from database as well

DEMO : flask_data_api.py

- We can consume/use API using request module.

DEMO Flask API Consumption.py

We can test APIs using external platform such as Restman Chrome Extension

Machine Learning Model API

- Deployment of machine learning models is simply putting models into production that makes models available to other systems within the organization or the web.

- End user can utilize ML model in two ways :

1. Send data and retrieve Prediction on web-browser
2. Send data and retrieve Prediction using rest API in any tool (e.g. python) through http request (POST method)

- Let's deploy ML model using flask .

Demo flask_ml_api.py