API Testing – Software testing

API testing, or application programming interface testing, is a type of software testing that focuses on the

testing of individual API methods and the
interactions between different APIs.

This type of testing is typically performed

at the integration level,
after unit testing is completed, and
before user interface testing begins.

It is used to validate that the API behaves correctly and that it meets the requirements of the system.

API testing can be performed manually or using automated testing tools. Some common tasks that are performed during API testing include:

- Testing the functionality of the API to ensure it behaves as expected
- Verifying that the API returns the correct response for different input values Checking for error handling and validation of input
  Testing for security vulnerabilities
- Checking for performance and scalability of the API
- API testing is important because it ensures that the different components of a system can communicate with each other correctly and that the system can handle a large volume of requests.

It is also used to ensure that the API is compatible with different platforms and operating systems, and can be integrated with other systems and applications.

API Testing : As we know API stands for Application Programming Interface which acts as an intermediate of communication between two applications. Due to this intermediary role of API (Application Programming Interface) two applications talk to each other and performs the required actions efficiently.

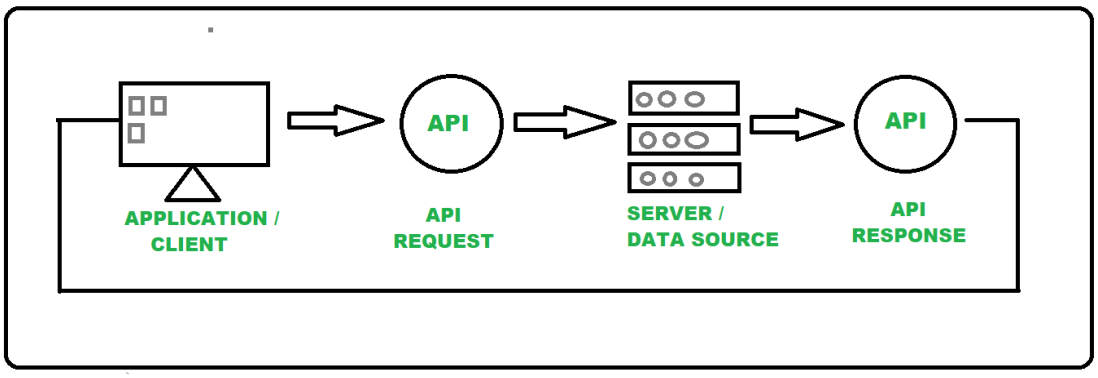
API contains a set of rules and guidelines based on which the applications are developed. So in simple we can say an API acts as an interface between two software applications so that two software applications can communicate with each other.

API testing typically includes the following steps:

- Reviewing the API documentation to understand the functionality and expected inputs and outputs
- Writing test cases that exercise the different functionality of the API
- Executing the test cases and comparing the expected results with the actual results
- Analysing the results and identifying any issues that need to be fixed

There are several types of API testing, including:

- Functional testing: Testing the functionality of the API to ensure it behaves as expected
- Security testing: Testing the security of the API to ensure it is protected against common vulnerabilities
- Performance testing: Testing the performance of the API to ensure it can handle the expected load
- Interoperability testing: Testing the compatibility of the API with other systems
- Usability testing: Testing the usability of the API for developers
- Tools such as Postman, SoapUI, and Runscope can be used to automate and simplify the process of API testing.



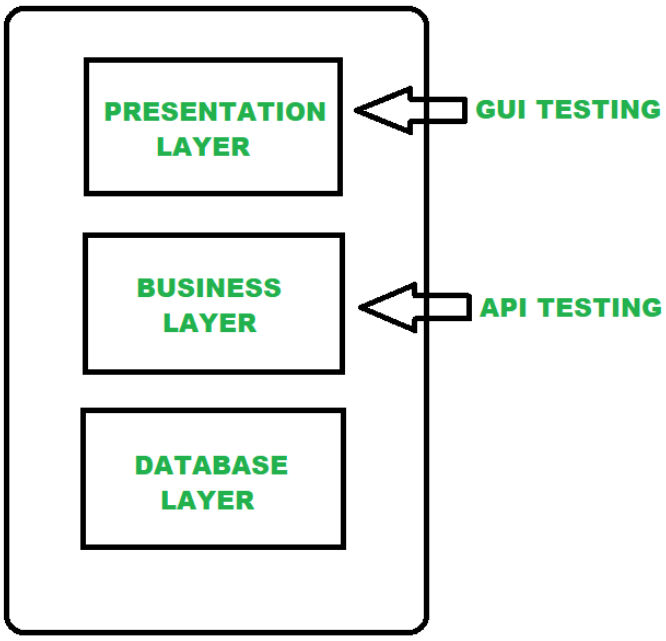
Types of API testing

API Testing refers to test the APIs which are used in the application just to validate that the APIs are working fine. When a system has a collection of APIs, these needs to be tested to know that the system is working perfectly or not. Mostly we can say that API testing confirms system's performance, reliability, security and functionality. **Below list represents some of the tools which are used for API Testing :**

- Postman
- Katalon Studio
- Soap UI
- Parasoft
- REST assured
- Tricentis Tosca

- Ping API
- Assertible

GUI testing is different from the API testing as GUI testing is present at Presentation layer where the API testing is present at Business layer. If we take an example of a typical app then API is the middle layer in between UI layer and Data base layer and due to this API communication and data exchange between the applications occur. The below **figure** represents the layer at which API testing is performed :



Layers of API Testing

API testing Types : There are multiple types of testing which are most often used as form of API testing which means during multiple types of testing simultaneously API can be tested. So below list represents the types of API testing i.e.

1. Unit Testing
2. Integration Testing
3. End-to-End Testing
4. Performance Testing
5. Functional testing
6. Security Testing
7. Load testing
8. Penetration testing
9. Reliability testing
10. Fuzz testing

What exactly we check during API testing :

- Data accuracy.

- Response time.
- Duplicate or missing functionality.
- Authorization checks.
- Multithreaded issues.
- Security and performance issues.
- Error codes if API returns.
- Reliability issues.

Benefits of API Testing :

Like we get a lot of advantages by using APIs in application, similarly by performing API testing we achieve a lot of things towards the success of the developed application. Below are some benefits i.e.

- Earlier validation of correctness in response and data.
- Earlier test maintenance.
- Better speed and coverage of testing.
- GUI independent testing.
- Reduced testing cost.
- Language independent test.
- Helpful in testing core functionality.
- API testing has several benefits that make it an important aspect of software testing:
- Improved functionality: API testing helps ensure that the functionality of the API is working as expected and that the data being exchanged is accurate and complete.
- Increased security: API testing helps identify and fix security vulnerabilities such as SQL injection and cross-site scripting. This helps ensure that the API is protected against common threats and that sensitive data is secure.
- Improved performance: API testing helps identify and fix performance bottlenecks, such as slow response times or high error rates. This helps ensure that the API can handle the expected load and that users have a positive experience when using it.
- Better integration: API testing helps ensure that the different systems that make up an application are working together correctly and that the data being exchanged is accurate and secure.
- Reduced risk: By identifying and fixing issues before the application is deployed to production, API testing helps reduce the risk of system failure or poor performance in production.
- Cost-effective: API testing is more cost-effective than fixing problems that occur in production. It is much cheaper to identify and fix issues during the testing phase than after deployment.
- Improved developer experience: By making sure that the API is easy to use, well-documented, and provides useful error messages, API testing helps improve the developer experience and encourage adoption.

- Greater flexibility: API testing allows teams to test the application without a user interface, which can be useful when testing microservices or when the user interface is not yet developed.

Disadvantages of API Testing:

API testing can have some disadvantages, including:

- Complexity: API testing can be complex, especially when testing multiple APIs or when testing APIs that are integrated with other systems.
- Limited Visibility: Since API testing is performed at the integration level, it can be difficult to see how the API is interacting with other components of the system. This can make it difficult to identify and troubleshoot issues.
- Security: APIs can introduce security vulnerabilities if they are not properly tested and secured. This can be a significant concern for organizations that handle sensitive data.
- Difficulty in testing non-functional requirements: Non-functional requirements such as performance, scalability and security are difficult to test with functional testing
- Time consuming: The time required to develop and execute test scripts for APIs can be longer than other types of testing.
- Limited documentation: Limited or poor documentation of the API can make it difficult for testers to understand how the API should behave.
- Limited test coverage: It is difficult to test all possible scenarios and edge cases with API testing.
- Cost: Automated API testing tools can be expensive and require a significant investment.

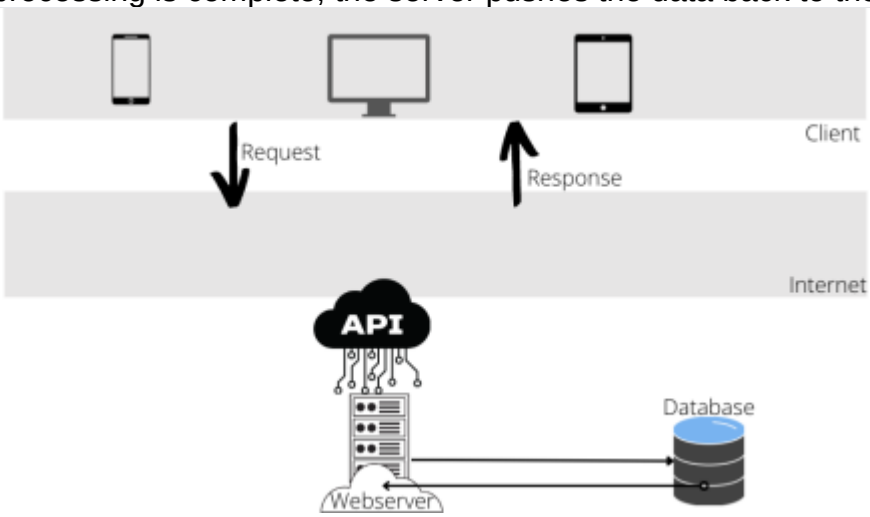
Types of Bugs that may occur in API Testing:

- Performance Issues – API response Time can be very high, and they may have latency.
- Response data may not structure correctly (JSON or XML)
- Security Issues
- Incorrect handling of valid argument values
- Improper errors/warning to caller
- Missing or Duplicate Functionality
- Reliability Issues : Difficulty in connecting and getting a response from API

What is API Testing?

API testing is integral to software development to provide optimal application performance. API, a specialized application programming interface, is a collection of protocols, tools, and functions that facilitate seamless communication between software applications.

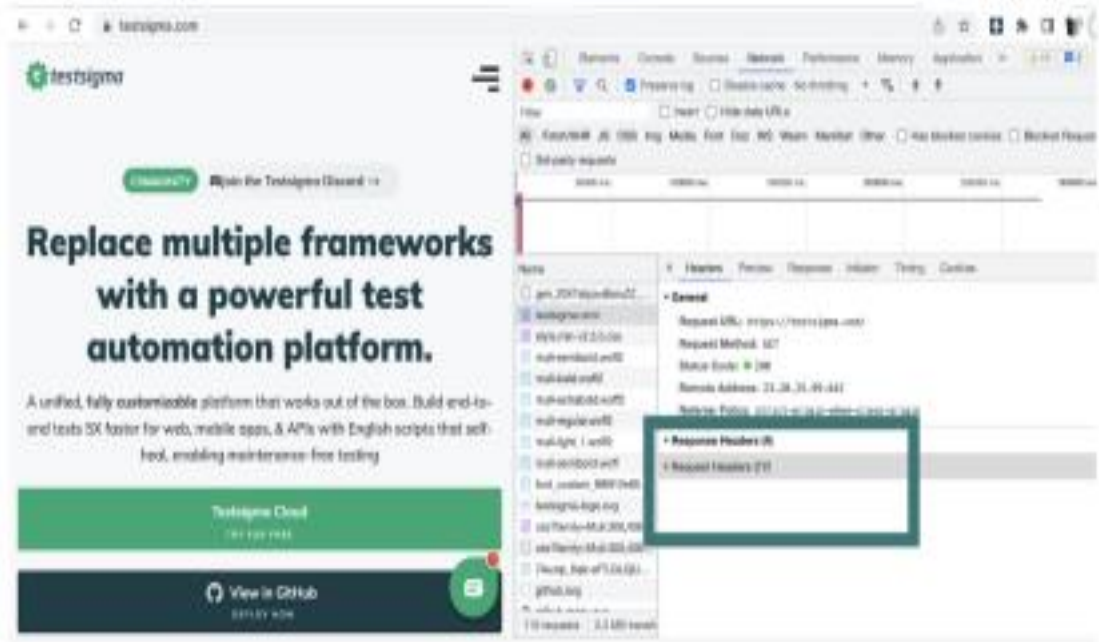
Client/ client-side- includes all the devices used to interact with the application.
The Internet is an enabler; it assists in transporting client requests to the Web server along with relevant details filled in by the user on the client side.
Web-server/server-side is the holy place where all processing takes place. Once the processing is complete, the server pushes the data back to the client.



API Facets

APIs comes with 2 broad facets
1. Request– An encapsulation of action performed along with pertinent details to be sent to the server.
2. Response– An encapsulation of processed data along with supplement details sent back to client by the server.

Both request and response come with header attached. Take a look at any network call, you should be able to see headers quite evidently.



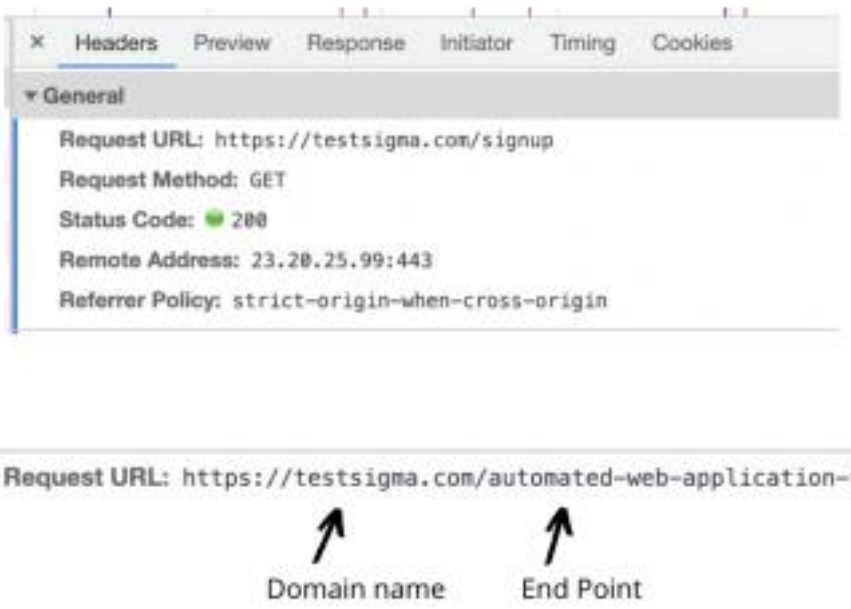
The Request header passes all the additional details to the server to process the Request. In the example below, we see a set of keys and values. The key names are case insensitive.

Sometimes Request also carries payload details; in the below example, I am trying to join the Testsigma discord community. To make that happen, I entered some details through UI and clicked on the “Send” button. In the backend call, we can see that the details passed to the server when the event is triggered fill the payload.

A web server processes the Request and prepares the response body and additional header details. For example, when I chose to join the discord community, the API responded with further information about filling captcha. We plan to perform regressive testing on the response body during API testing to verify that the sent details match our expectations.

How are API called?

Each user interaction on UI is associated with an API that will be accessed through a URL. Let us say a user fills up a form and clicks on “Send” button, on trigger of this event respective API will be invoked. In the below picture we can see the Request URL is pointing to the endpoint ‘Signup’.



API Request Method

Every request is associated with the “Request Method”. This method specifies the desired action to be performed by the server. Below are the frequently used methods

1. GET: This method is used to retrieve the details, this performs a read-only operation.
2. POST: This method submits data by creating an entry into database. This method should be used wisely as it can change the state causing side effects if not handled properly.
3. PUT: This method is used to update any data, the data that needs to be replaced is sent as part of payload.
4. DELETE: This method just deletes the data.
5. HEAD: This is similar to get method, however there is no response body sent.
6. PATCH: This method specifies how to update the data, this can can cause side effects if not handled with care.

Few FAQs about request methods

1. How will I get to know which API follows what method?

Ideally software development teams maintain API documentations. Every API is preceded with the respective method in the documentation.

2. Can one API endpoint perform multiple actions?

Yes, sometimes same endpoint is designed to do multiple methods.

3. Can I use API without specifying method?

No, you have to suffice the API call with request method

4. Can all APIs be accessed through URL

Yes and NO, some APIs are open for public and may not need authentication. However most of the business and enterprise APIs are behind API gateway and one may have to authenticate before calling the API.

5. Can same endpoint have multiple method associated?

Yes, it is completely possible to define multiple actions/verbs for same endpoint. Hence it is important to know before hand what actions are supported by the API.

API Response

Every API request is associated with response. The response have 2 facets

1. Status code : Numeric representation of the web server response. There are pre-defined set of status code which can be reused. Or, teams can create their own status code as per convenience. However, appropriate status code range should be used.

INFORMATIONAL RESPONSES	(100-199)
SUCCESSFUL RESPONSES	(200-299)
REDIRECTION MESSAGES	(300-399)
CLIENT ERROR RESPONSES	(400-499)
SERVER ERROR RESPONSES	(500-599)

2. Status Message : Each status code is associated with detailed response message.

for example:

Status code 200 → Success

Status code 401 → Unauthorized

Status code 404 → Not found

API Gateway

API usage is not as simple in real world, when companies are hosting large scale APIs protecting them from misuse is their primary concern. API gateway helps establishing authentication to verify the calls before transferring it to further execution. The

capability of API gateway is not just restricted to authentication, it also provides multitude of services like

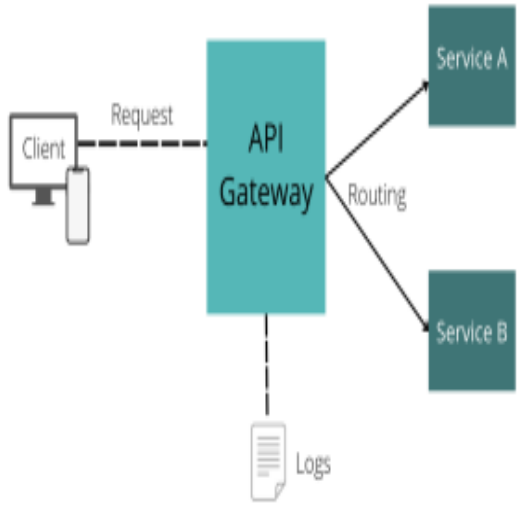
I. Routing

ii. Rate Limiting

iii. Analytics

iv. Security

v. Policies etc



API Authentication

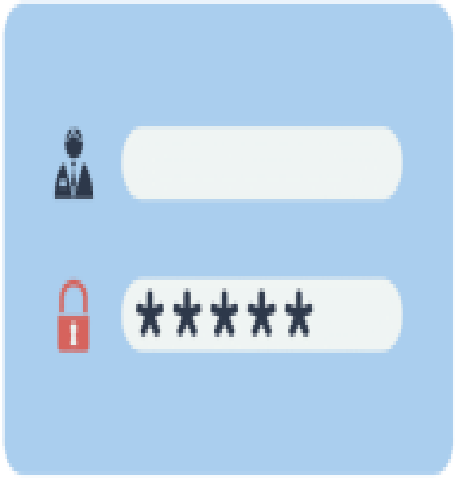
Since APIs deal with protected resources, the request processing should be backed with authentication process to ensure the access is provided only to the intended user. There is a difference between authentication and authorisation.

Authentication is a process of verifying the identity of the user it solely answers who you are, whereas authorisation mostly deals with access management and comes into play only after the user is identified and verified successfully.

Most commonly used authentication are

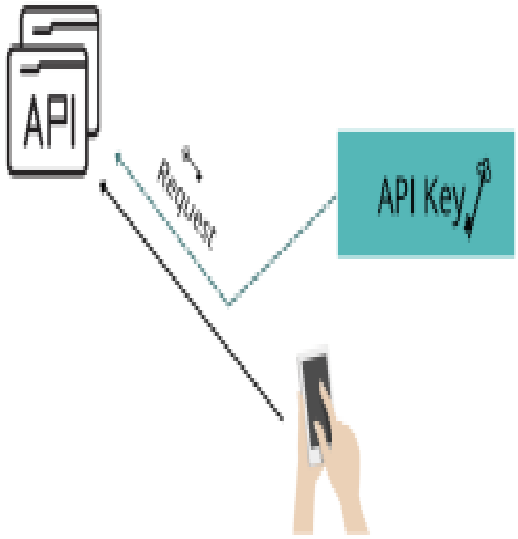
1. Basic Authentication:

This technique involve providing username and password for user verification. Authentication sounds little flimsy isn't it?, but this is how it is going to work. When user enters their credentials the details are encoded in Base64 generating a Key which will be bundled in request header and sent to server for verification. Server Verifies the key with the stored
username and password. If the identity is verified the request is fulfilled else an error is sent back denying request sent.



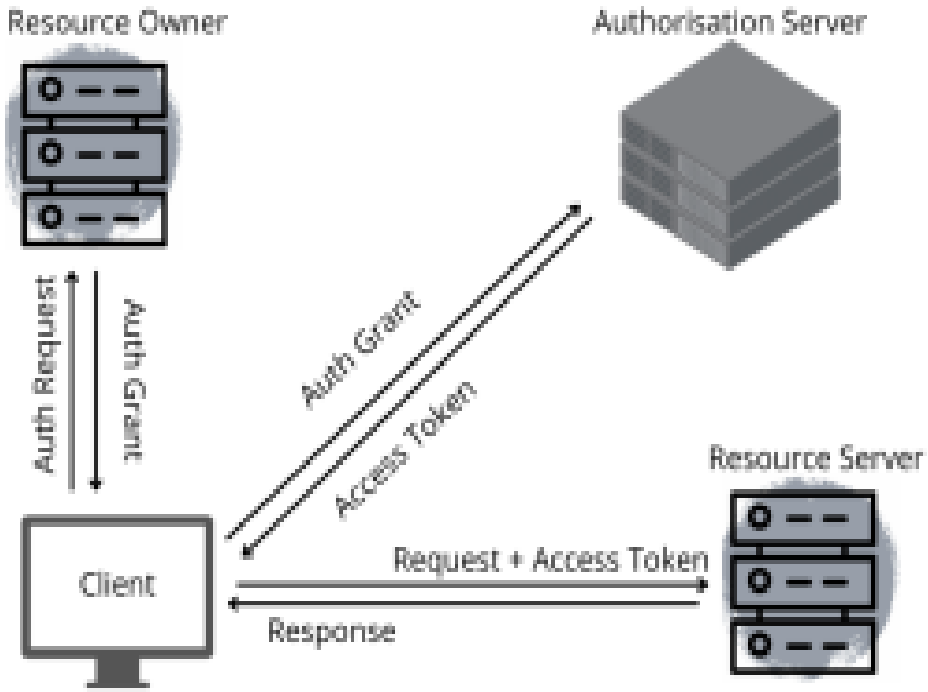
2. API Key Authentication :

API key is a long encrypted string which identifies the application without any principal. These are sent either as a part of request header or URL. When client recognises the API key server will process the request.



3. OAUTH Authentication :

This technique is considered quite powerful and secure way of authenticating the users. oAuth technique can also be used for authorisation. Initially a user may have to login to the oAuth application using the credentials to generate a token. The generated token is attached as part of request header, which will be sent to authentication server in order to verify. If the token is recognised the API request will be processed



API testing basics for beginners

Generating Test cases for API testing

API testing is a black box testing technique, one don't really delve into what is implemented within an API. Having said that, if you would like to particularly walk through the code to test the API that is fine too. The main task in testing APIs is to figure out the test cases just like any other tests out there.
For example consider the below REST API

GET https://gorest.co.in/public/v2/users
Looking at the API method we understand that this is going to perform a read operation and fetch the details requested which is list of all users. The response returned will be in the JSON format. REST Apis generally use JSON structure to represent the response body.

Let us go ahead churning the test cases for this API
1. On Successful execution the API should return status code 200 (message : Success)
2. Specific user details can be fetched by passing user id in URL (https://gorest.co.in/public/v2/users/25)

3. The JSON response should match the below schema

{

“id”:integer,

“name”:string,

“email”:string,

“gender”:string,

“status”:string

}

4. Wrong URL should respond with 404 (Not found)

5. Multiple User ID in the URL should respond with 404 error

API Testing Approach

There are various approaches to API testing, and the following points highlight some of the standard testing approaches:

1. Functional testing: This involves testing the functionality of the API to ensure that it meets the specified requirements.
2. Security testing: This involves testing the security of the API to ensure that it is not vulnerable to security threats.
3. Load testing: This involves testing the performance of the API under different loads to ensure that it can handle the expected traffic.
4. Integration testing: This involves testing the integration of the API with other software components to ensure that it works seamlessly.

Overall, a comprehensive API testing approach should cover all these testing approaches to ensure the quality and reliability of the software application.

How to Test API?

- Understand the API’s Functionality

Before you start testing an API, you need to understand its functionality. This includes the data it exchanges with the client, the expected format of the data, the input parameters, and the expected output. Understanding the API’s functionality will help you design your test strategy and ensure that your test cases cover all the required scenarios.

- Test the API Endpoints

Once you understand the API well, you can begin testing the endpoints. This involves sending requests to the API using various HTTP methods (GET, POST, PUT, DELETE).

Best Practices of API Testing

The best practices of API testing are essential to ensure the functionality and reliability of an application programming interface.

1. Plan and design the API tests before starting the testing process.
2. Use a testing framework that provides robust features to automate API testing.
3. Create test data covering a wide range of scenarios and edge cases.
4. Verify the API's functional behavior, security, and performance.
5. Conduct regression testing to ensure that changes and updates do not impact the existing API functionality.
6. Integrate API testing with continuous integration and delivery pipelines.
7. Collaborate with developers and other stakeholders to identify and resolve issues early.
8. Use monitoring tools to track API performance and identify issues before they impact end-users.

Challenges of API Testing

Here are some of the most common challenges:

1. Complex API architecture and integration with other systems can make testing difficult and time-consuming.
2. API testing requires programming skills and knowledge of HTTP protocols and methods.
3. Test data management can be challenging, especially when dealing with large data sets.
4. Ensuring API security and authentication is essential, but testing can be challenging.
5. API testing requires close collaboration between developers, testers, and stakeholders to ensure that the APIs meet the business requirements and user expectations.

Conclusion

Hey, it's time to wrap things up! API testing is super important when it comes to software development. It ensures that APIs play nice with other systems and do what they should. If you're new to the game, getting a handle on the API testing basics,

including the different tests, tools, and tricks involved, is key. This guide should give you a solid overview of API testing basics, so you can confidently start testing your APIs. But listen up: API testing isn't a one-and-done deal; it's an ongoing process that you must keep up with to keep your APIs in tip-top shape. Keep learning, keep testing, and keep improving those APIs!

Try out Testsigma for our API test automation needs

Happy Testing !

Frequently Asked Questions

How will I get to know which API follows what method?

The request header carries all the supplemental details to be passed on to the server to process the Request. In the example below, we see a set of keys and values. The key names are case insensitive.

Can one API endpoint perform multiple actions?

Yes, sometimes the same endpoint is designed to do multiple methods.

Can I use API without specifying the method?

No, you have to suffice the API call with the request method

Can all APIs be accessed through a URL?

Yes, and NO, some APIs are open to the public and may not need authentication. However, most business and enterprise APIs are behind API gateway, and one may have to authenticate before calling the API.

Can the same endpoint have multiple methods associated?

It is possible to define multiple actions/verbs for the same endpoint. Hence, it is essential to know beforehand what actions the API supports.

What are the three layers of API testing?

People commonly refer to the three layers of API testing as unit testing, integration testing, and end-to-end testing.

What is the process of API testing?

API testing typically involves identifying the requirements for the API, designing test cases, executing the tests, and analyzing the results.

What is Postman API?

Postman API is a platform that promotes the development, testing, and documentation of APIs.

API testing typically involves the following practices:

- **Unit testing:** To test the functionality of individual operation
- **Functional testing:** To test the functionality of broader scenarios by using a block of unit test results tested together
- **Load testing:** To test the functionality and performance under load
- **Runtime/Error Detection:** To monitor an application to identify problems such as exceptions and resource leaks
- **Security testing:** To ensure that the implementation of the API is secure from external threats
- **UI testing:** It is performed as part of end-to-end integration tests to make sure every aspect of the user interface functions as expected
- **Interoperability and WS Compliance testing:** Interoperability and WS Compliance Testing is a type of testing that applies to SOAP APIs. Interoperability between SOAP APIs is checked by ensuring conformance to the Web Services Interoperability profiles. WS-* compliance is tested to ensure standards such as WS-Addressing, WS-Discovery, WS-Federation, WS-Policy, WS-Security, and WS-Trust are properly implemented and utilized
- **Penetration testing:** To find vulnerabilities of an application from attackers
- **Fuzz testing:** To test the API by forcibly input into the system in order to attempt a forced crash

HTTP Protocol

HTTP stands for Hyper Text Transfer Protocol. Communication between client and web servers is done by sending HTTP Requests and receiving HTTP Responses.

HTTP Protocol is used to perform

CRUD operations (Create, Read, Update, Delete) by sending HTTP requests with different HTTP methods.

HTTP methods are also called as HTTP verbs.

HTTP Request Structure

Every HTTP request should have a URL or URI address and a method.

(Highlighted in purple)
Example URI

http://api.example.com/products?name=laptop&available=true

The format of a request message includes

- A request-line
- Zero or more header field(s) followed by CRLF (Carriage Return, Line Feed)
- An empty line
- A message-body (optional)

HTTP Methods in REST

REST
REpresentational State Transfer

It is a set of architectural principles for designing web services.

Web services based on REST Architecture are known as RESTful web services.

REST makes it easy to share data between clients and servers.

REST applications use HTTP methods like GET, POST, DELETE, PUT, etc., to do CRUD operations.

Categories of HTTP Methods

You can divide HTTP methods into two main categories :

- Safe HTTP Methods
- Idempotent Methods

Safe HTTP Methods

It doesn't change data on the server. It always returns the same response, no matter how many times it gets called.

Example - GET, HEAD

Idempotent Methods

It may change data on the server. It always returns the same response, no matter how many times it gets called.

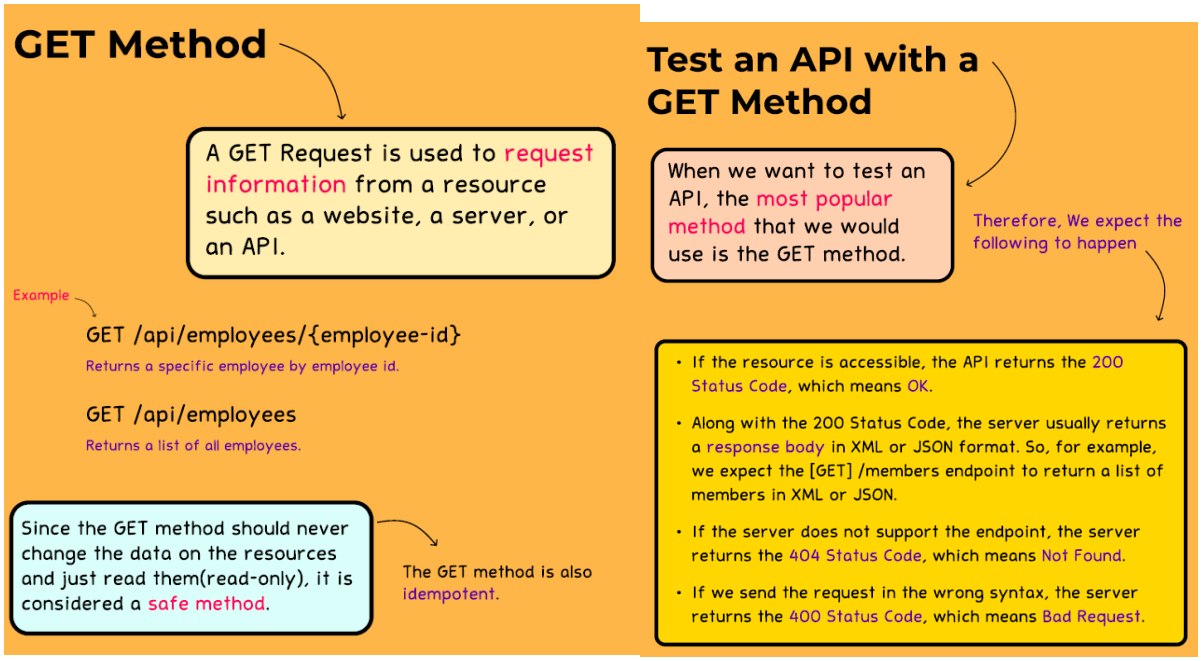
Example - GET, HEAD, PUT, DELETE, TRACE

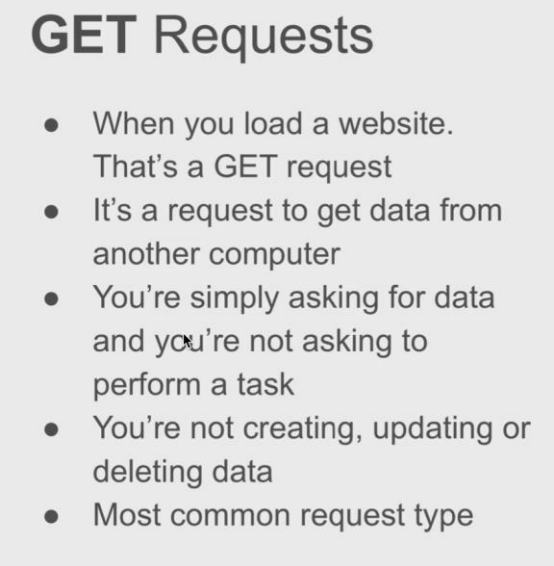
All safe methods are also idempotent, but not all idempotent methods are safe.

The 9 HTTP Methods

- GET Method
- POST Method
- PUT Method
- PATCH Method
- DELETE Method
- HEAD Method
- OPTIONS Method
- TRACE Method
- CONNECT Method

Let us now see them in detail.





Sending data through url

Caching: GET requests can be cached by browsers and proxy servers, which can improve performance and reduce server load.

Security: While GET requests are generally considered safe, sensitive data should not be passed in the URL query string to avoid exposing it in server logs or browser history.

Sure, here are some examples of GET requests:

1. **Retrieve a List of Resources:**
   - Request: `GET /api/users`
   - Response: Returns a list of users in JSON format.

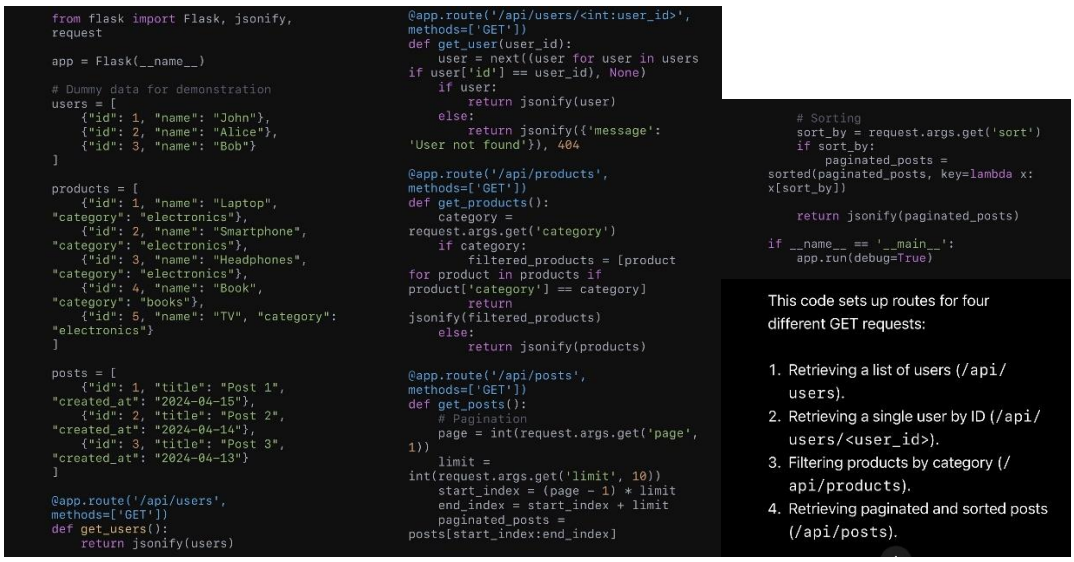
2. **Fetch a Single Resource by ID**:

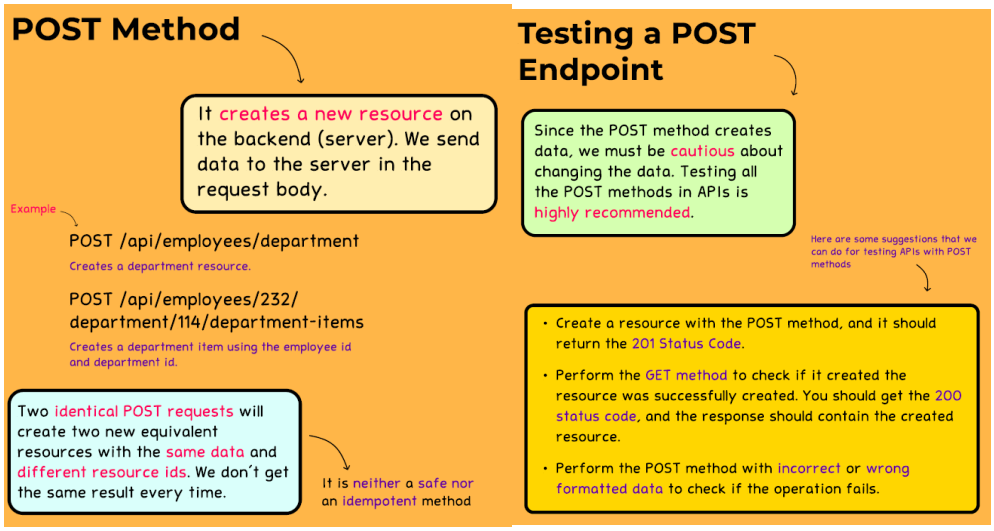
- Request: `GET /api/users/123`
- Response: Returns the user with ID 123 in JSON format.

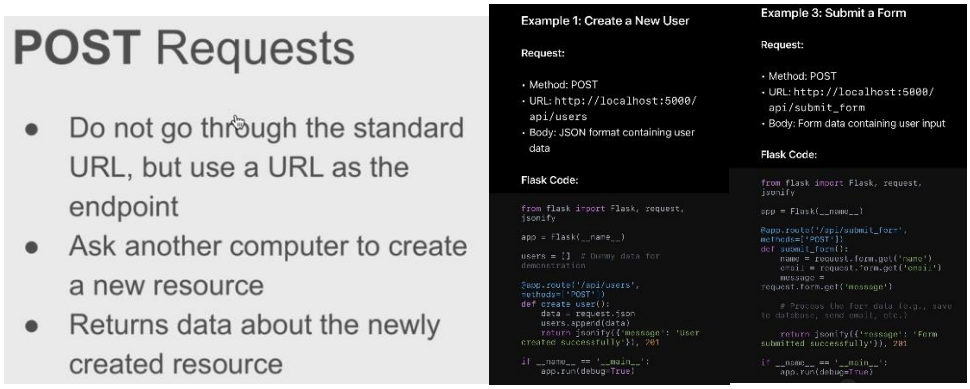
3. **Filter Data Based on Parameters**:
- Request: `GET /api/products?category=electronics`
- Response: Returns a list of products belonging to the "electronics" category in JSON format.

4. **Pagination**:
- Request: `GET /api/posts?page=2&limit=10`
- Response: Returns the second page of posts with a limit of 10 posts per page in JSON format.

5. **Sorting**:
- Request: `GET /api/posts?sort=created_at`
- Response: Returns a list of posts sorted by the creation date in ascending order in JSON format.







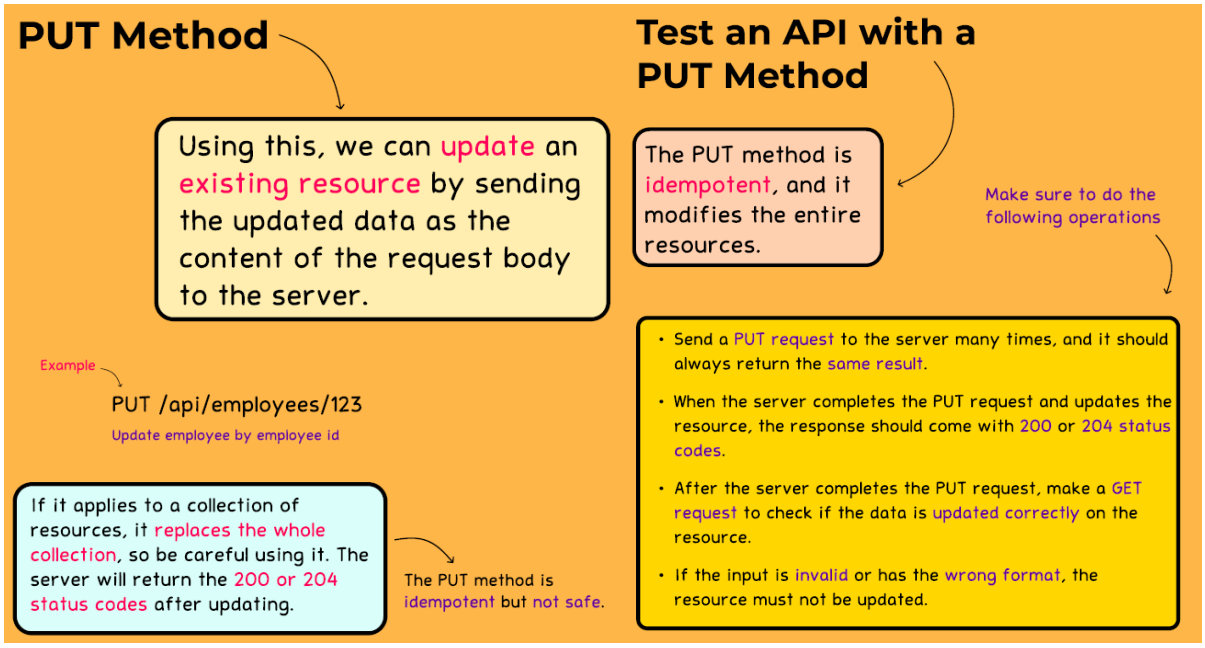
Sumbmitting any form or field ,creating user in fb by filling form

1. **Definition**: The POST method is used to submit data to be processed to a specified resource. It's commonly used for creating new resources.

3. **Request Body**: Data to be sent to the server is included in the body of the request. This can be in various formats like JSON, XML, or form data.

4. **Security**: POST requests can contain sensitive data, so it's important to use HTTPS to encrypt the data in transit. Additionally, input validation and sanitization should be performed on the server-side to prevent security vulnerabilities like SQL injection and XSS attacks.

8. **Examples**: Provide examples of POST requests for creating resources, such as creating a new user, adding a product to a database, or submitting a form.



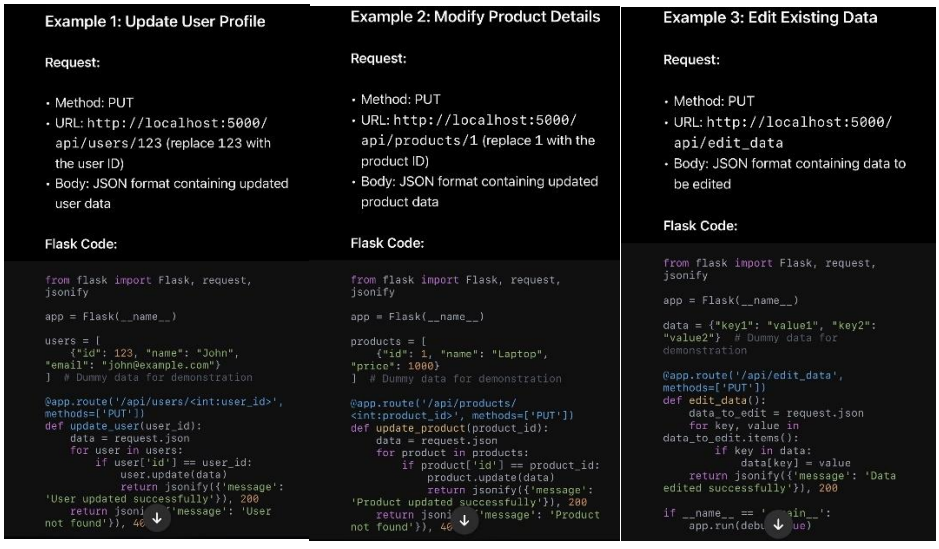
1. **Definition**: The PUT method is used to update a resource or create it if it doesn't exist at a specified URL.

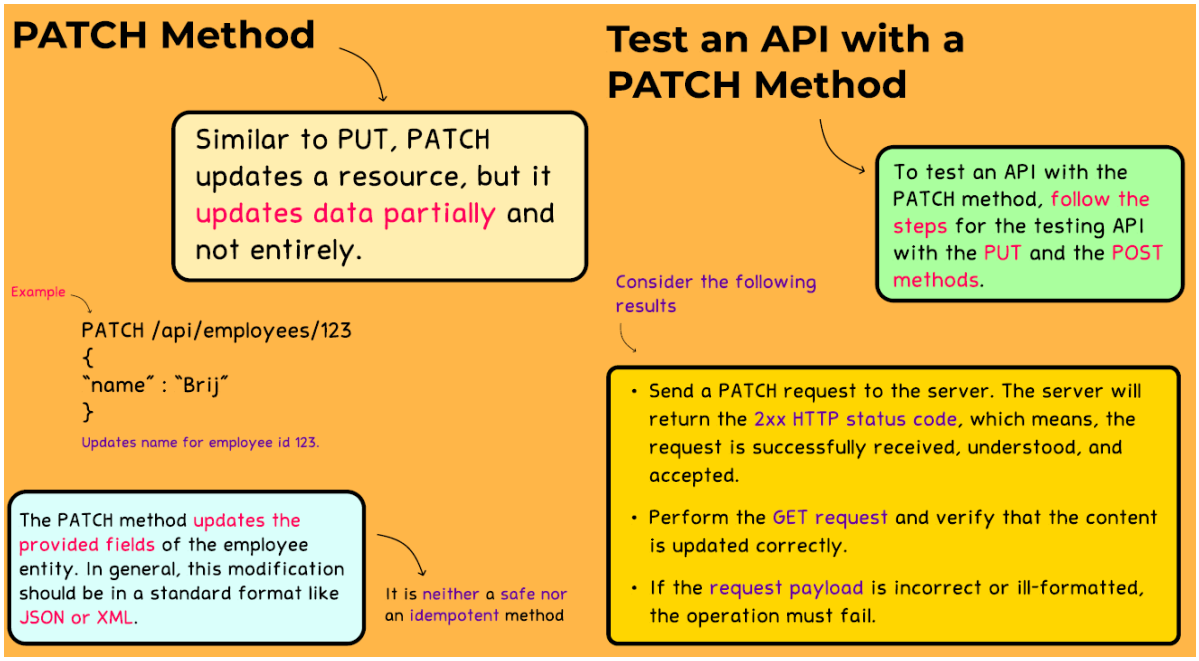
3. **Request Body**: Data to be updated or created is included in the body of the request. This can be in various formats like JSON, XML, or form data.

4. **Idempotent**: PUT requests are idempotent, meaning making the same request multiple times should have the same result. If the resource already exists, it will be updated with the provided data. If it doesn't exist, a new resource will be created.

5. **Security**: Similar to POST requests, PUT requests can contain sensitive data, so it's important to use HTTPS to encrypt the data in transit. Input validation and sanitization should be performed on the server-side to prevent security vulnerabilities.

7. **Examples**: Provide examples of PUT requests for updating resources, such as updating a user's profile information, modifying product details, or editing existing data.





The PATCH method in API is similar to the PUT method, but it's typically used for partial updates to an existing resource, rather than replacing the entire resource. Here's an explanation of the PATCH method:

1. **Partial Updates**: PATCH requests are used when you want to update only specific fields or properties of a resource, rather than replacing the entire resource. This can be useful when you only need to modify certain attributes of an object without affecting others.

2. **HTTP Verb**: Like PUT and POST, PATCH is an HTTP method used for making requests to a server. It's typically used in situations where a PUT request might be too heavy-handed, especially when dealing with large or complex resources.

3. **Request Body**: Data to be updated is included in the body of the PATCH request. This data typically contains only the fields that need to be modified, rather than the entire resource representation.

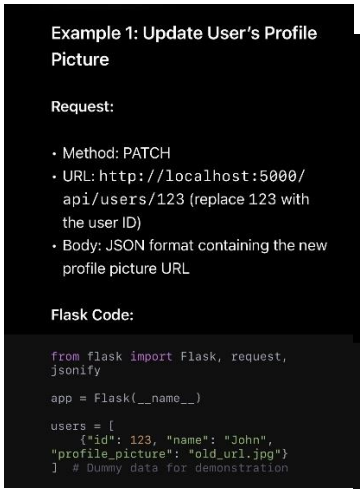
4. **Idempotent**: PATCH requests are not necessarily idempotent. While they can be designed to be idempotent in certain cases, it's not a requirement like it is for PUT requests.

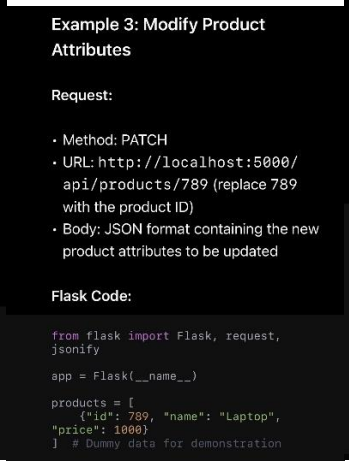
5. **Response**: After processing the PATCH request, the server responds with a status code indicating the success or failure of the operation, similar to other HTTP methods.

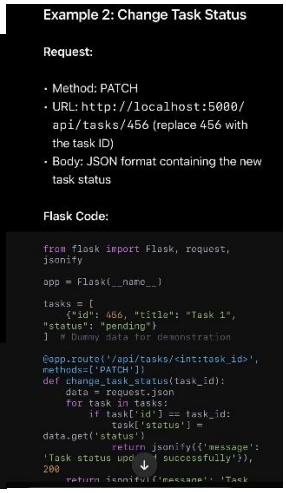
6. **Examples**: PATCH requests can be used for various scenarios, such as updating a user's profile picture, changing the status of a task, or modifying specific attributes of a product.

7. **Best Practices**: It's important to design PATCH requests carefully to ensure that they accurately reflect the intended partial updates and that they're implemented in a way that is consistent with the application's data model and business logic.

In summary, the PATCH method is used for making partial updates to existing resources in a flexible and efficient manner, making it a valuable tool in API design and development.





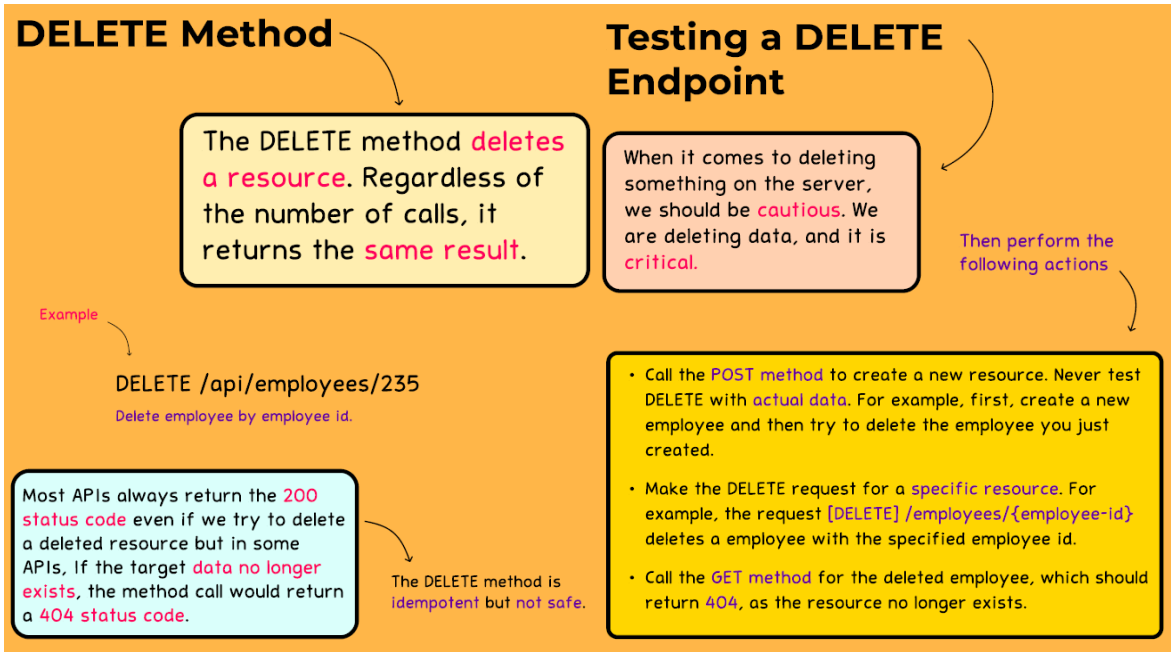


The main difference between the PUT and PATCH methods in API lies in how they are used for updating resources:

1. **PUT Method**:
 - The PUT method is used to update or replace an entire resource at a specific URL.
 - When making a PUT request, the client sends a complete representation of the resource, including all fields, even those that are not being updated.
 - If a resource exists at the specified URL, the entire resource is replaced with the new representation provided in the request body.
 - PUT requests are idempotent, meaning making the same request multiple times should have the same result.

2. **PATCH Method**:
 - The PATCH method is used to make partial updates to an existing resource.
 - When making a PATCH request, the client sends only the fields that need to be updated, rather than the entire resource representation.
 - If a resource exists at the specified URL, only the specified fields in the request body are updated, leaving the rest of the resource unchanged.
 - PATCH requests are not necessarily idempotent, as subsequent requests may produce different results based on the initial state of the resource and the applied patches.

In summary, PUT is typically used for full updates where the client sends the complete representation of the resource, while PATCH is used for partial updates where the client sends only the modified fields.



The DELETE method in API is used to request the removal of a resource from the server at a specified URL. Here's an overview of the DELETE method:

1. **HTTP Verb**: DELETE is one of the standard HTTP methods, alongside GET, POST, PUT, and PATCH.

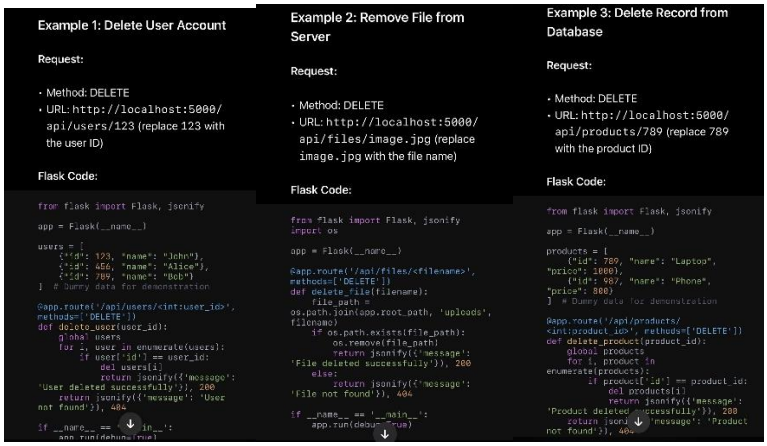
2. **Resource Removal**: DELETE requests are used to instruct the server to remove the resource identified by the URL from its storage.

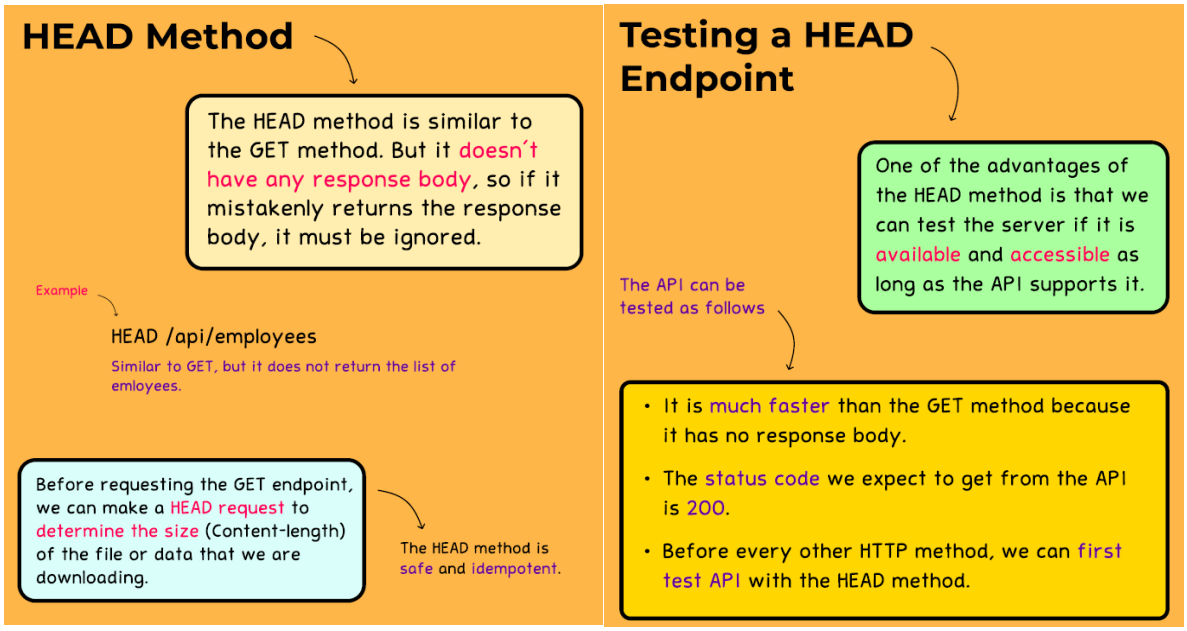
3. **Idempotent**: DELETE requests are idempotent, meaning making the same request multiple times should have the same result. Once a resource is deleted, subsequent DELETE requests on the same resource will have no effect.

4. **Response**: After processing the DELETE request, the server typically responds with a status code indicating the success or failure of the operation. Common status codes include 200 (OK) for successful deletion and 404 (Not Found) if the resource does not exist.

5. **Security**: DELETE requests can have significant consequences as they permanently remove data from the server. Proper authentication and authorization mechanisms should be in place to ensure that only authorized users can perform DELETE operations.

6. **Examples**: DELETE requests are commonly used in scenarios such as deleting a user account, removing a file from a server, or deleting a record from a database.





The HEAD method in API is similar to the GET method, but it's used to retrieve metadata about a resource rather than the resource itself. Here's an overview of the HEAD method:

1. **HTTP Verb**: HEAD is one of the standard HTTP methods, alongside GET, POST, PUT, DELETE, and PATCH.

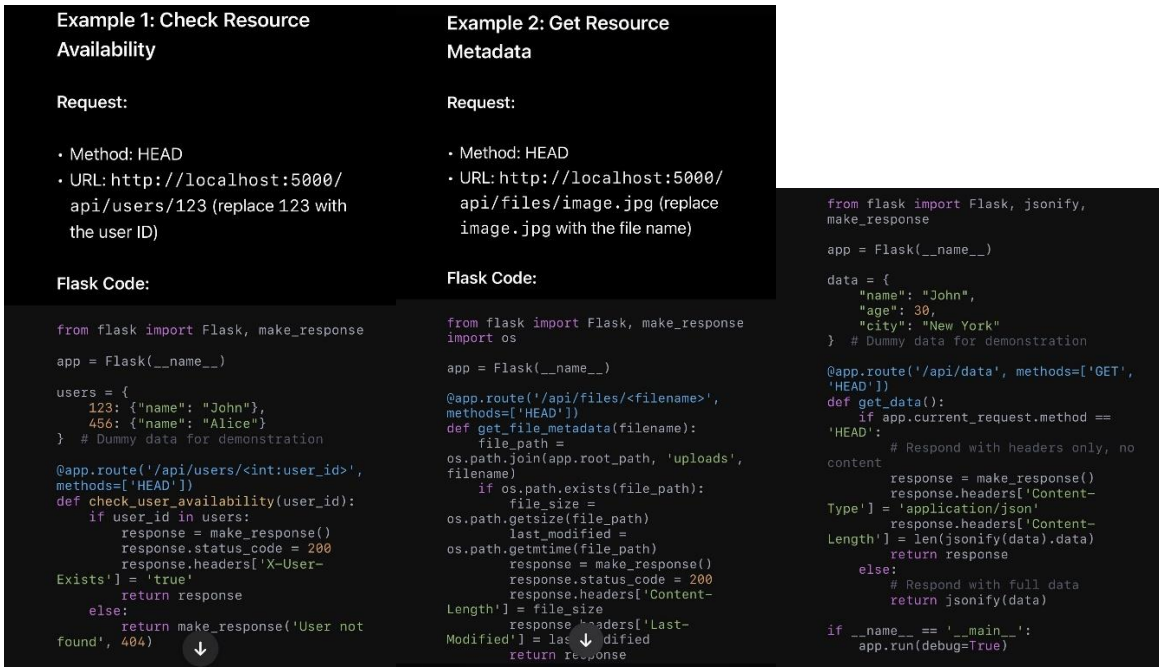
2. **Metadata Retrieval**: HEAD requests are used to retrieve metadata about a resource, such as its headers, content length, content type, etc.

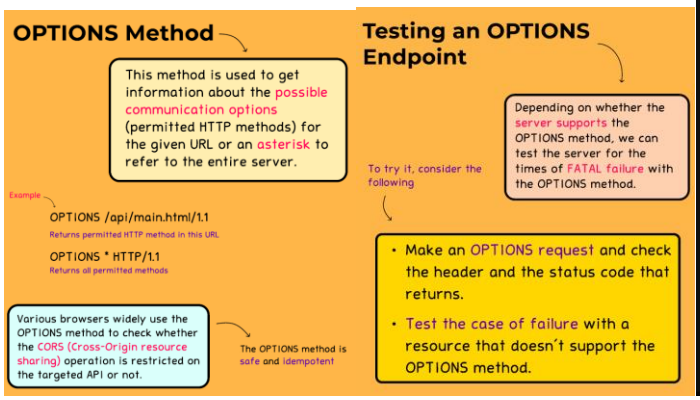
3. **Response**: When a server receives a HEAD request, it responds with the headers that would be returned for a corresponding GET request to the same URL, but without the actual content of the resource. This allows clients to retrieve important metadata without downloading the entire resource, which can be useful for checking resource availability, content characteristics, or server status.

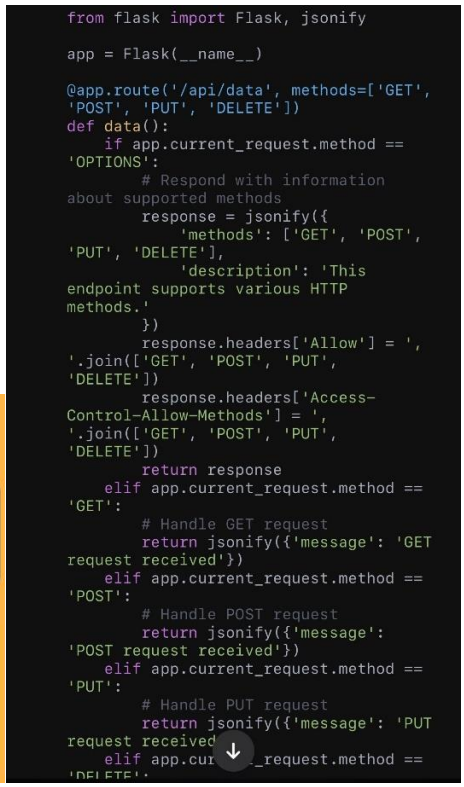
4. **Idempotent**: HEAD requests are idempotent, meaning making the same request multiple times should have the same result. However, they do not affect the server's state or cause any changes to the requested resource.

5. **Examples**: HEAD requests are commonly used in scenarios where clients need to check the status or characteristics of a resource without actually downloading its content, such as checking if a file exists, verifying the last modified timestamp of a document, or determining the size of a file.

In summary, the HEAD method provides a lightweight way to retrieve metadata about resources, making it a useful tool for efficient communication between clients and servers.







The OPTIONS method in API is used to retrieve information about the communication options available for a particular resource or server. Here's an overview of the OPTIONS method:

1. **HTTP Verb**: OPTIONS is one of the standard HTTP methods, alongside GET, POST, PUT, DELETE, and PATCH.

2. **Resource Information**: OPTIONS requests are used to inquire about the communication options available for a resource or server, such as supported HTTP methods, allowed headers, supported authentication methods, etc.

3. **Response**: When a server receives an OPTIONS request, it responds with a list of the supported HTTP methods, headers, and other relevant information for the requested resource. This allows clients to determine the available communication options and make appropriate requests.

4. **Usage**: OPTIONS requests are commonly used during preflight CORS (Cross-Origin Resource Sharing) requests, where the client sends an OPTIONS request to the server to determine if a cross-origin request is allowed.

5. **Idempotent**: OPTIONS requests are idempotent, meaning making the same request multiple times should have the same result. They do not cause any changes to the server's state or affect the requested resource.

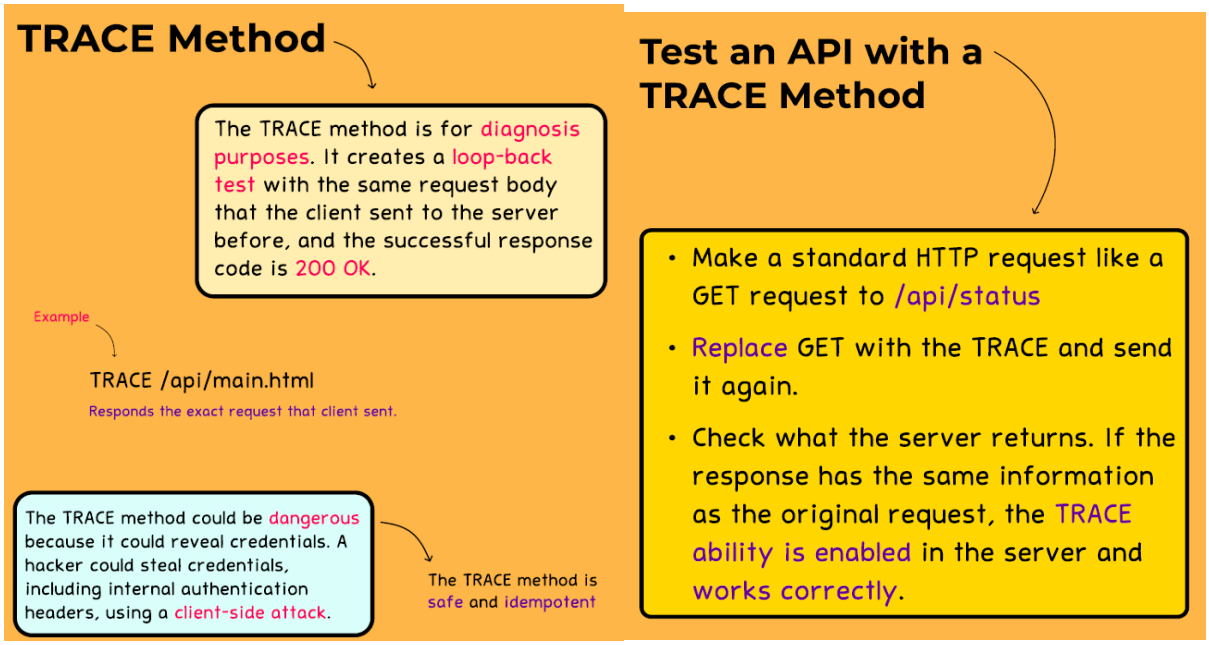
6. **Examples**: OPTIONS requests can be used in various scenarios, such as determining the allowed methods for a particular endpoint, checking CORS policies, or retrieving server capabilities.

In summary, the OPTIONS method provides a standardized way for clients to inquire about the communication options available for a resource or server, making it a useful tool for understanding server capabilities and making appropriate requests.

it's not mandatory to have an OPTIONS method implementation for every endpoint in your API. The OPTIONS method is primarily used for providing information about the communication options available for a particular resource or server.

However, if your API is designed to support CORS (Cross-Origin Resource Sharing), the OPTIONS method becomes more relevant. In CORS, browsers may make a preflight OPTIONS request to check if the server allows the intended cross-origin request. In such cases, you should handle OPTIONS requests appropriately to provide the required CORS headers and information about supported methods.

In general, if your API doesn't require CORS support or if there are no specific communication options that need to be exposed to clients, you may not need to implement the OPTIONS method for all endpoints. It's typically implemented for endpoints where it's necessary to provide information about supported methods or CORS policies.



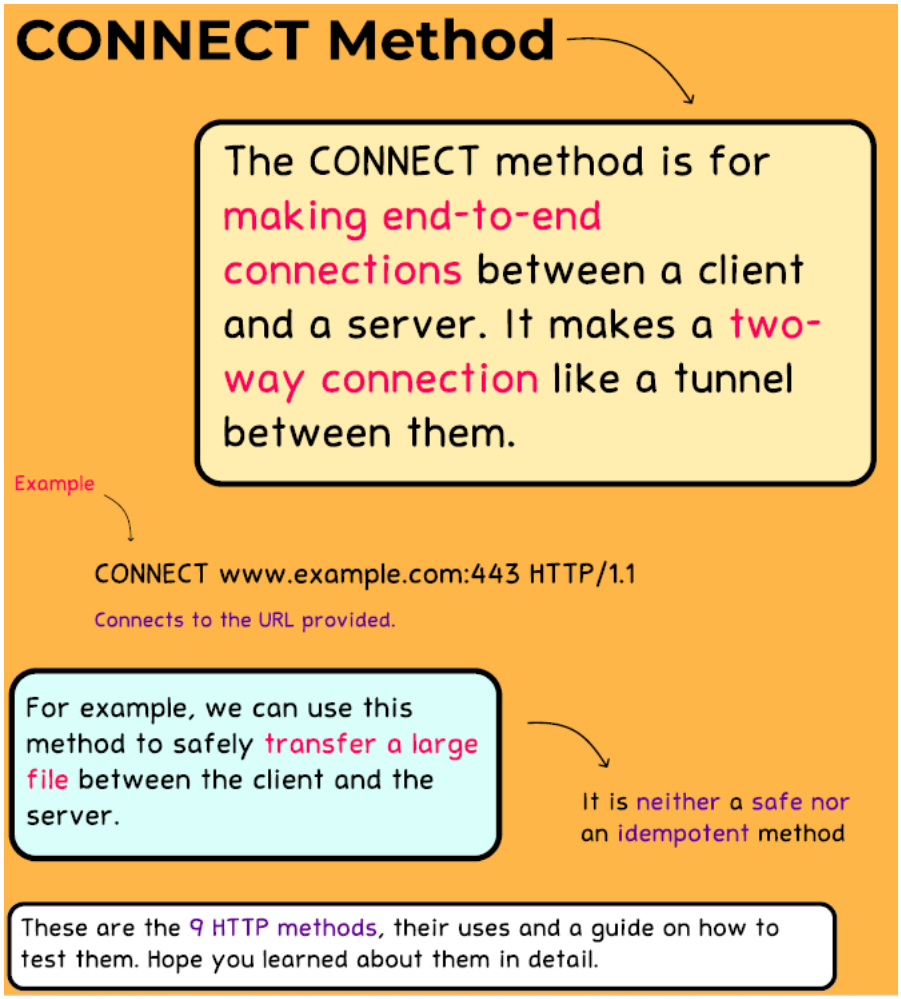
The TRACE method in HTTP is used primarily for diagnostic purposes. When a server receives a TRACE request, it echoes back the received request to the client, allowing the client to see what changes, if any, were made by intermediate servers along the request chain.

However, the TRACE method is often disabled or restricted in modern web servers due to security concerns, as it can potentially expose sensitive information or be exploited in cross-site tracing (XST) attacks.

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/api/trace', methods=['TRACE'])
def trace_request():
    # Echo back the received request information
    request_info = {
        'method': request.method,
        'url': request.url,
        'headers': dict(request.headers),
        'data': request.get_data(as_text=True)
    }
    return jsonify(request_info)

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

```python
import requests

url = 'http://localhost:5000/api/trace'
response = requests.request('TRACE', url)

print(response.json())
```

```
{
  "method": "TRACE",
  "url": "http://localhost:5000/api/trace",
  "headers": {
    "Host": "localhost:5000",
    "User-Agent": "python-requests/2.26.0",
    "Accept-Encoding": "gzip, deflate",
    "Accept": "*/*",
    "Connection": "keep-alive"
  },
  "data": ""
}
```


CONNECT Method
The CONNECT method is for making end-to-end connections between a client and a server. It makes a two-way connection like a tunnel between them.
Example
CONNECT www.example.com:443 HTTP/1.1
Connects to the URL provided.
For example, we can use this method to safely transfer a large file between the client and the server.
It is neither a safe nor an idempotent method
These are the 9 HTTP methods, their uses and a guide on how to test them. Hope you learned about them in detail.

DELETE Requests

- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to delete a single resource or a list of resources
- **Use with caution**

Deleting anything

PATCH Requests

- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to **update a piece** of a resource
- Are not fully supported by all browsers or frameworks, so we typically fall back on PUT requests
- Example: Updating a users first name

Python Django not support

PUT Requests

- Do not go through the standard URL, but use a URL as the endpoint
- Ask another computer to **update an entire** resource
- If the resource doesn't exist, the API might decide to CREATE (CRUD) the resource

HTTP Methods for RESTful Requests		
HTTP Method	CRUD Operation	Example URL(s)
GET	Read	HTTP GET http://website.com/api/users/ HTTP GET http://website.com/api/users/1/
POST	Create	HTTP POST http://website.com/api/users/
DELETE	Delete	HTTP DELETE http://website.com/api/user/1/
PUT	Update/Replace	HTTP PUT http://website.com/api/user/1/
PATCH	Partial Update/Modify	HTTP PATCH http://website.com/api/user/1/

More details at https://restfulapi.net/http-methods/

JSON Example

```
{
  "key_val_example": "value",
  "array_example": [
    'array item 1',
    'array item 2',
  ],
  "object_example": {
    "key1": "value1",
    "key2": "value2"
  }
}
```

XML Example

```
<example>
  <field>
    Value
  </field>
  <secondField>
    Value
  </secondField>
  <nestedExample>
    <nestedField>
      Value
    </nestedField>
    <nestedSecondField>
      Value
    </nestedSecondField>
  </nestedExample>
</example>
```

Browsers use JavaScript for their API requests.

Servers use any language that runs on that computer.

Healthy Responses (2--)

- 200 — OK.
  Request accepted.
- 201 — Created.
  POST requests often return 201s when a resource is created.
- 202 — Accepted.
  When a request is accepted but its not done processing.
  Maybe the task goes into a queue.

Redirect Responses (3--)

- 301 — Moved Permanently.
  When the endpoint has permanently changed. Update your endpoint.
- 302 — Found.
  The endpoint you're accessing is temporarily moved to somewhere else.

Client Responses (4--)

- 400 — Bad Request.
  Server cannot or will not process your request. Often this is due to malformed API keys or an invalid payload.
- 401 — Unauthorized.
  You're not allowed here. Usually this is because you're missing authentication credentials (API keys)
- 403 — Forbidden.
  The servers understands your request but won't execute it. Your API keys might not have the right permissions or your trying to use an endpoint that you don't have access to.
- 404 — Not Found.
  There's nothing here. Move along, move along.
- 405 — Method Not Allowed.
  You're using the wrong HTTP Method. The endpoint might only accept GET requests and you might be POSTing to it, for example.

Server Responses (5--)

- 500 — Internal Server Error.
  The server had a problem and couldn't process the request. This is **the only time you are out of control**.

RESTful API Cheat Sheet

REpresentational State Transfer

Request Methods

GET — Used to get data only and does not modify the data at all
Should return 200, 400 or 404 responses

POST — Create a new resource (ie. creating a new user)
Should return 200, 201 or 204 responses

DELETE — Delete a resource (ie. delete a user)
Should return 200, 202 or 204 responses

PUT — Update a resource. If the resource doesn't exist, the api might decide to create it and return a 201 response
Should return 200, 201 or 204 responses

Common Status Codes

2-- – Success Codes

200	OK
201	Created
202	Accepted
204	No Content

3-- – Redirection Codes

301	Moved Permanently
302	Found

4-- – Client Error Codes

400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed

5-- – Server Error Codes

500	Internal Server Error

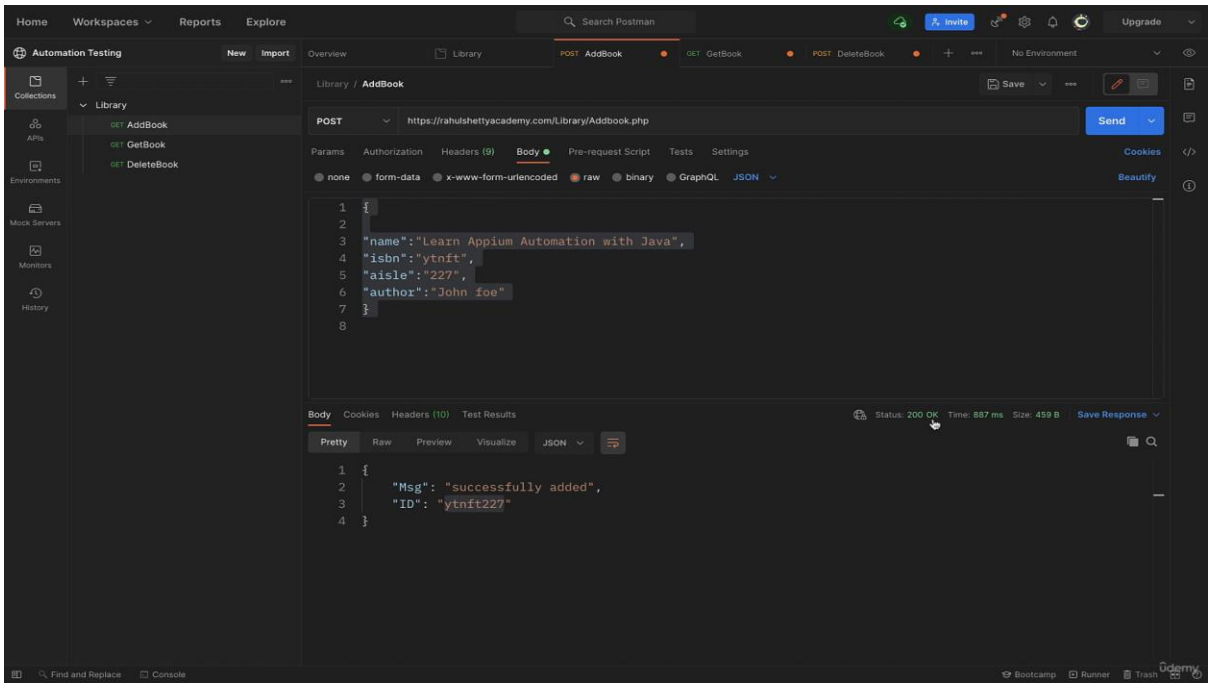
Authentication

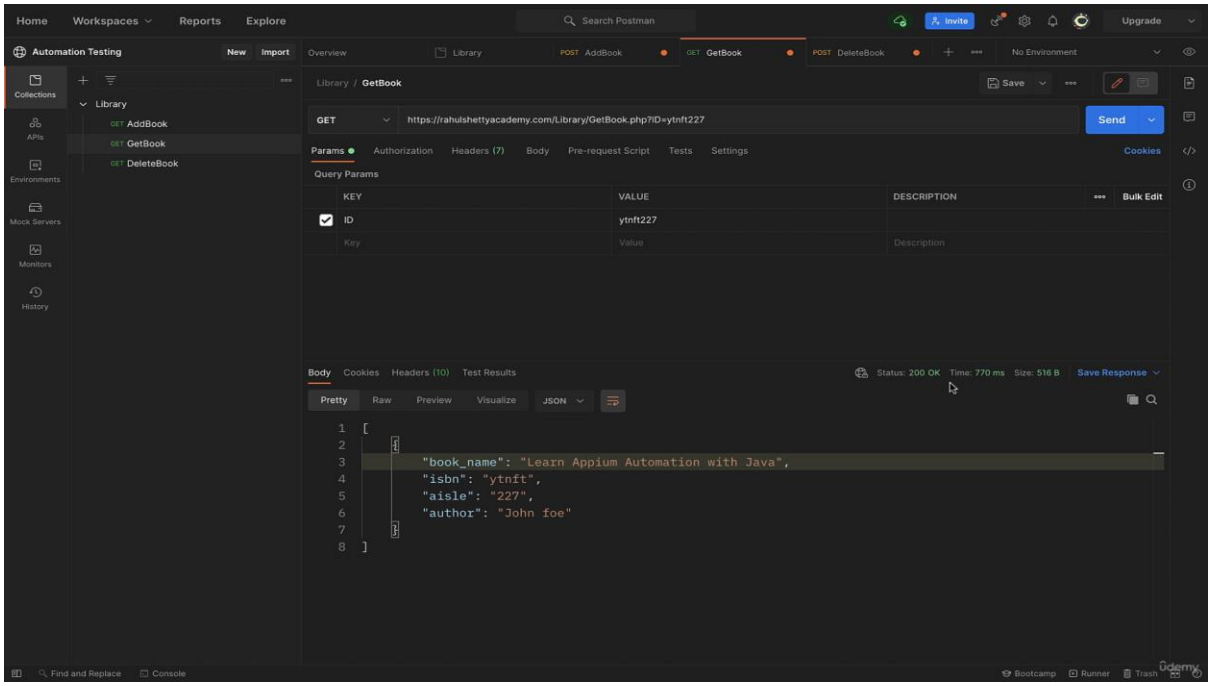
API keys are used as authentication credentials

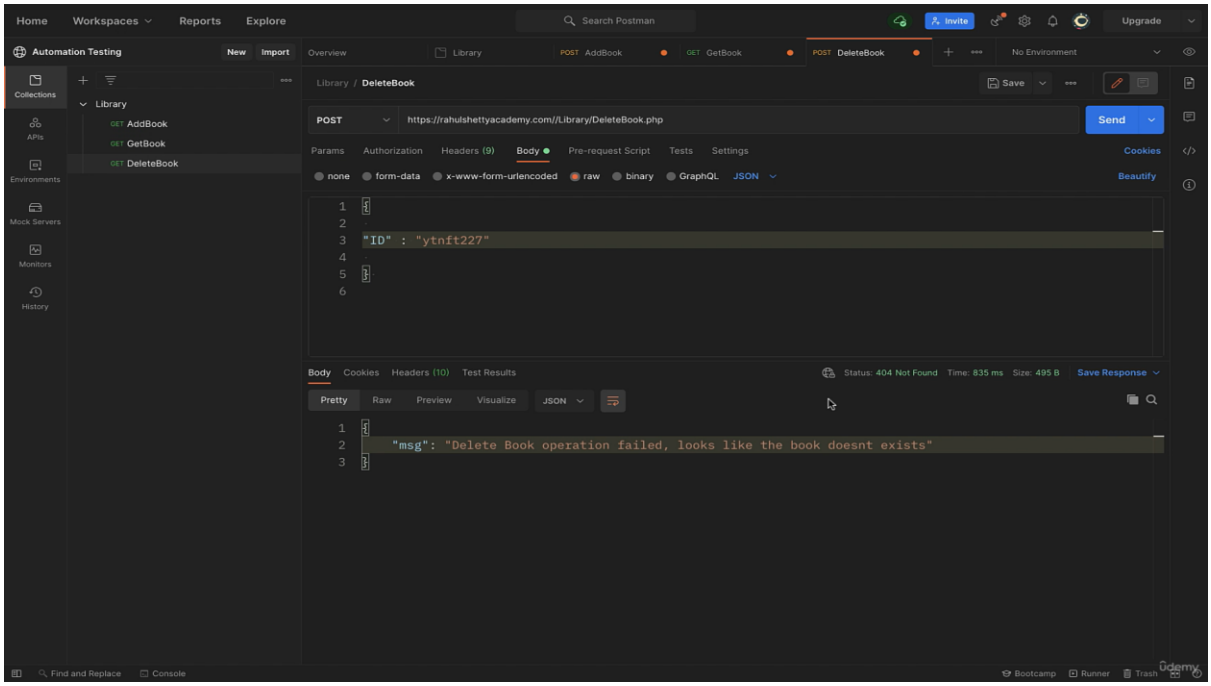
CRUD Operations

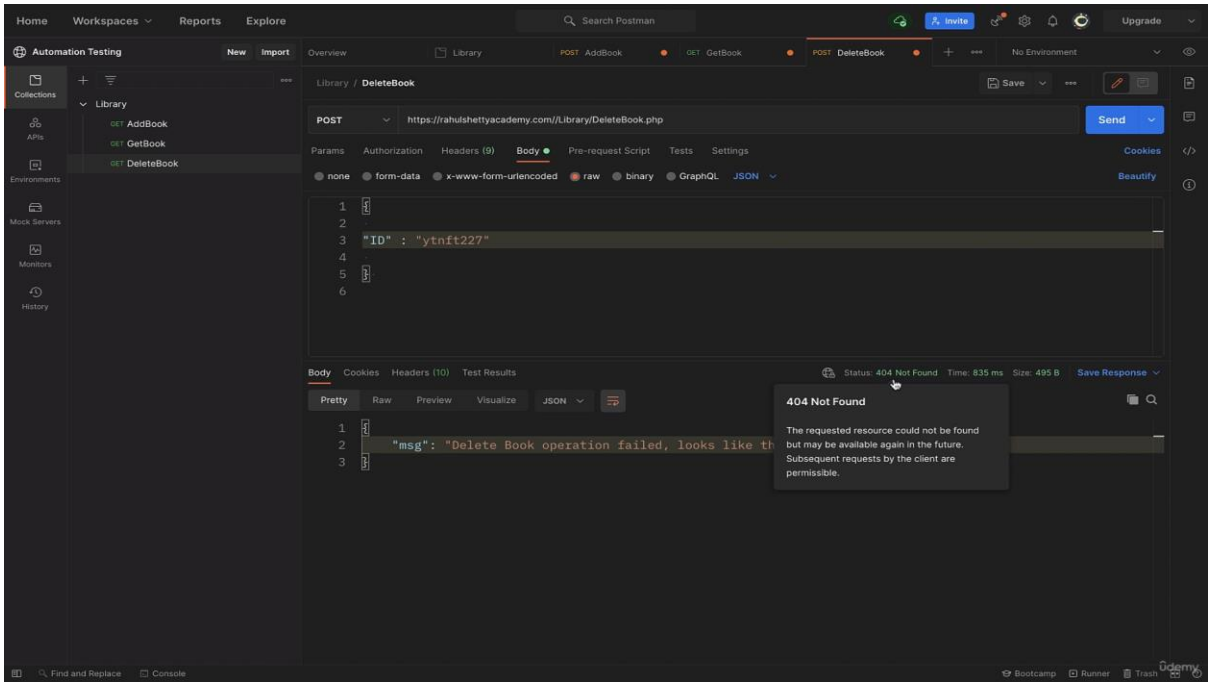
Create, Read, Update, Delete

@KalobTaulien

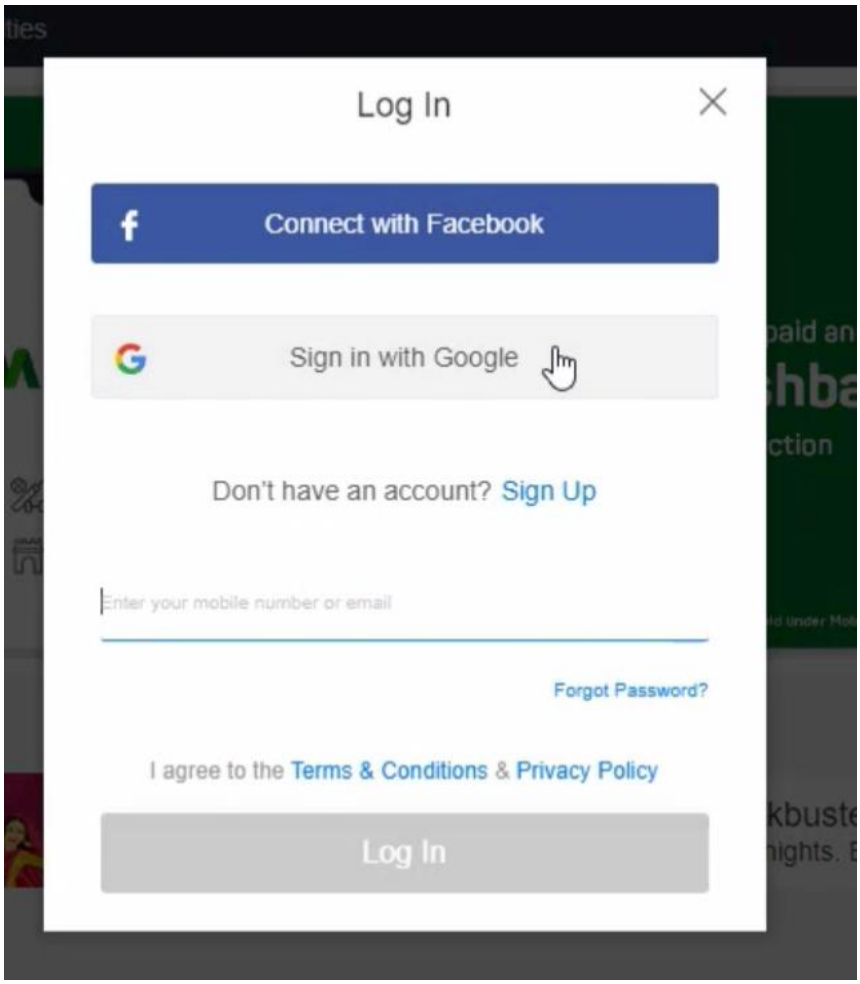



Home
Workspaces
Reports
Explore
Search Postman
Invite
Upgrade
Automation Testing
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
Library
GET AddBook
GET GetBook
GET DeleteBook
Overview
POST AddBook
GET GetBook
POST DeleteBook
No Environment
Library / GetBook
Save
GET
https://rahulshettyacademy.com/Library/GetBook.php?ID=ytnft227
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
ID
ytnft227
Key
Value
Description
Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 770 ms
Size: 516 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"book_name": "Learn Appium Automation with Java",
"isbn": "ytnft",
"aisle": "227",
"author": "John foe"
}
]
Find and Replace
Console
Bootcamp
Runner
Trash


Home
Workspaces
Reports
Explore
Search Postman
Invite
Upgrade
Automation Testing
New
Import
Collections
APIs
Environments
Mock Servers
Monitors
History
Library
GET AddBook
GET GetBook
GET DeleteBook
Overview
POST AddBook
GET GetBook
POST DeleteBook
No Environment
Library / DeleteBook
Save
POST
https://rahulshettyacademy.com//Library/DeleteBook.php
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"ID" : "ytnft227"
}
Body
Cookies
Headers (10)
Test Results
Status: 404 Not Found
Time: 835 ms
Size: 495 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"msg": "Delete Book operation failed, looks like the book doesnt exists"
}
Find and Replace
Console
Bootcamp
Runner
Trash


Home
Workspaces
Reports
Explore
Search Postman
Invite
Upgrade
Automation Testing
New
Import
Overview
Library
POST AddBook
GET GetBook
POST DeleteBook
No Environment
Collections
APIs
Environments
Mock Servers
Monitors
History
Library
GET AddBook
GET GetBook
GET DeleteBook
Library / DeleteBook
Save
POST
https://rahulshettyacademy.com//Library/DeleteBook.php
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"ID" : "ytnft227"
}
Body
Cookies
Headers (10)
Test Results
Status: 404 Not Found
Time: 835 ms
Size: 495 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"msg": "Delete Book operation failed, looks like th
}
404 Not Found
The requested resource could not be found but may be available again in the future. Subsequent requests by the client are permissible.
Find and Replace
Console
Bootcamp
Runner
Trash

OAuth2.0

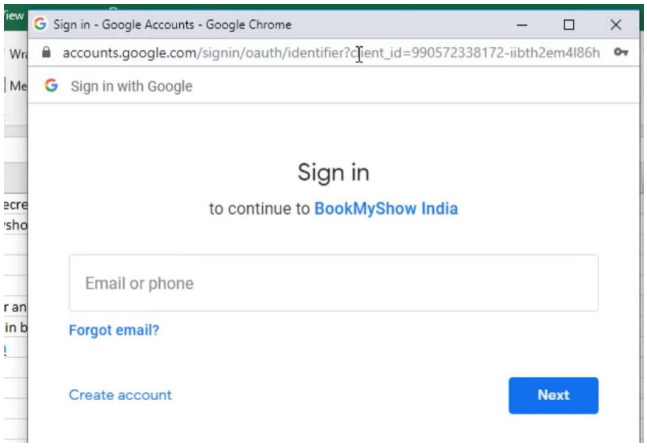

Log In
Connect with Facebook
Sign in with Google
Don't have an account? Sign Up
Enter your mobile number or email
Forgot Password?
I agree to the Terms & Conditions & Privacy Policy
Log In

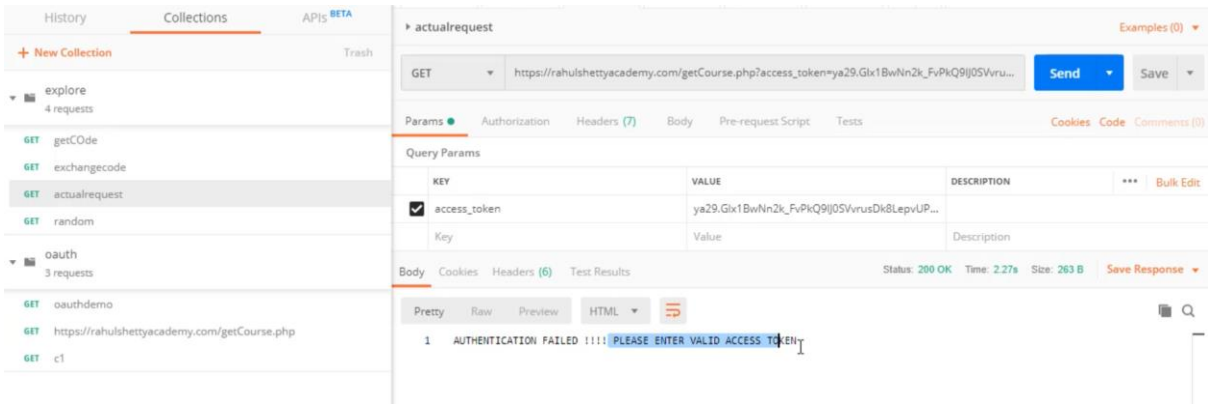
Why Applications rely on Other (Google or facebook) Authentications?
No Data breach Headaches for Application
Need not maintain user profile data
This also allows richer websites by allowing disparate applications to talk to each other.

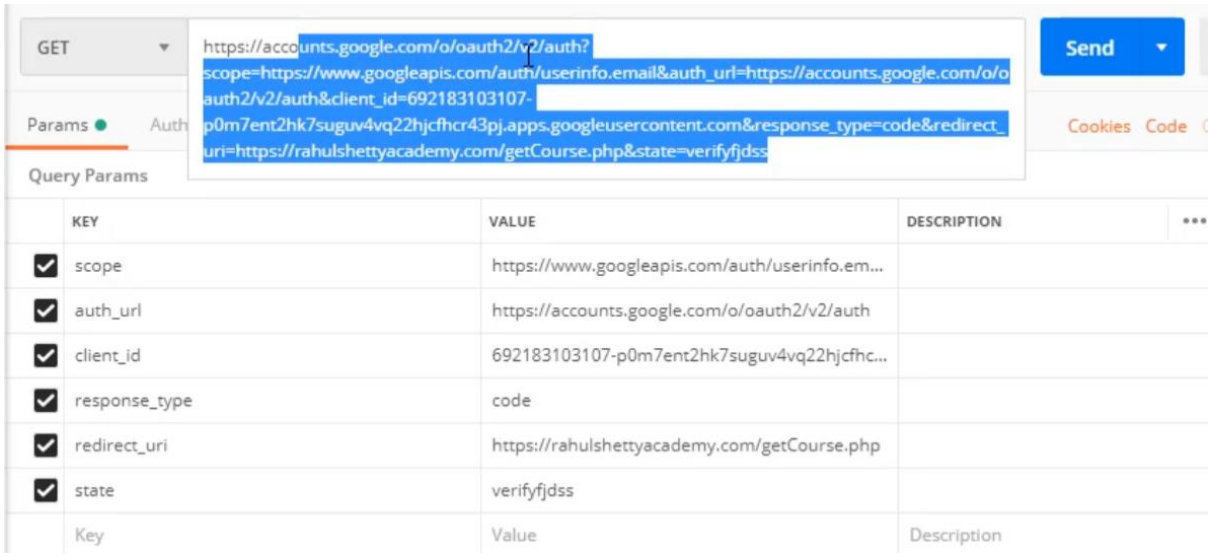
User sign into Google by hitting google Authorization server and get code
Application will use this code to hit google resource server in back end to get Access token
Application grants access to User by validating Access token

User sign into Google by hitting google Authorization server and get code
Application will use this code to hit google resource server in back end to get (Access token,First name, last name, email,d)
Application grants access to User by validating Access token

B	C	D	E	F
Client	ClientID	Client secret ID	Resource owner	Resource/Authorization Server
BookMyShow	ID that identifies the client	Bookmyshow registers this with google	Human	Google





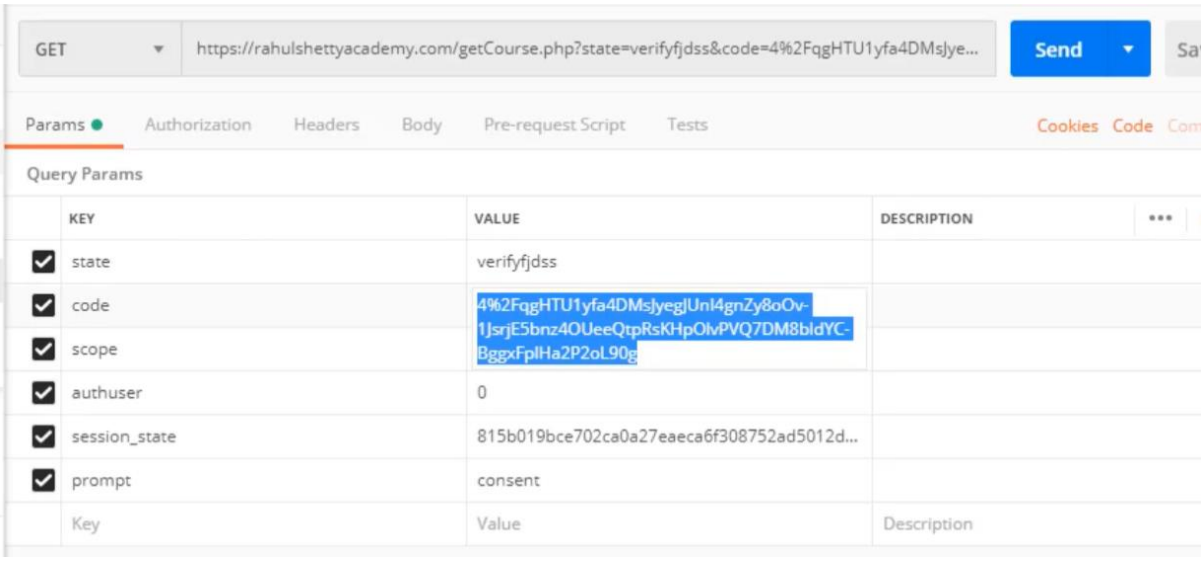


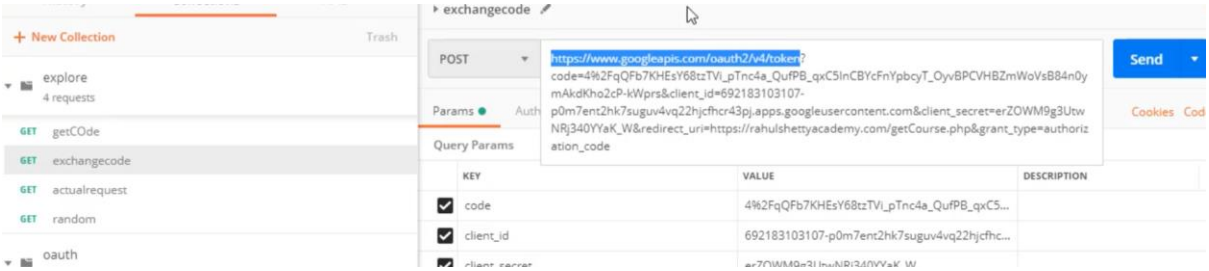
Scope what information is required from google

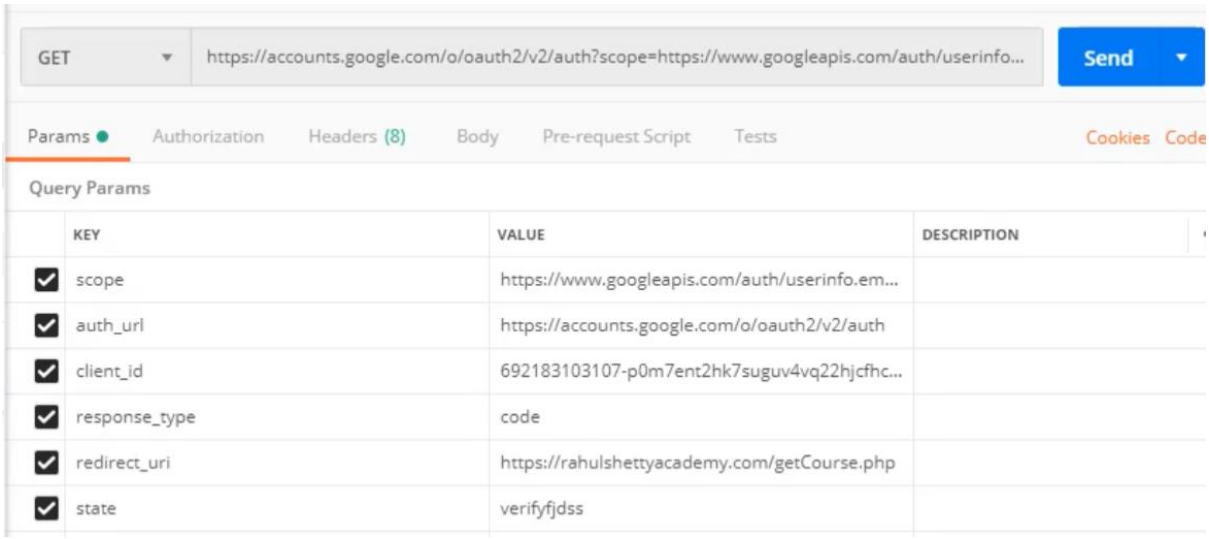
State is optional it should be available in Rahul server for security purposes

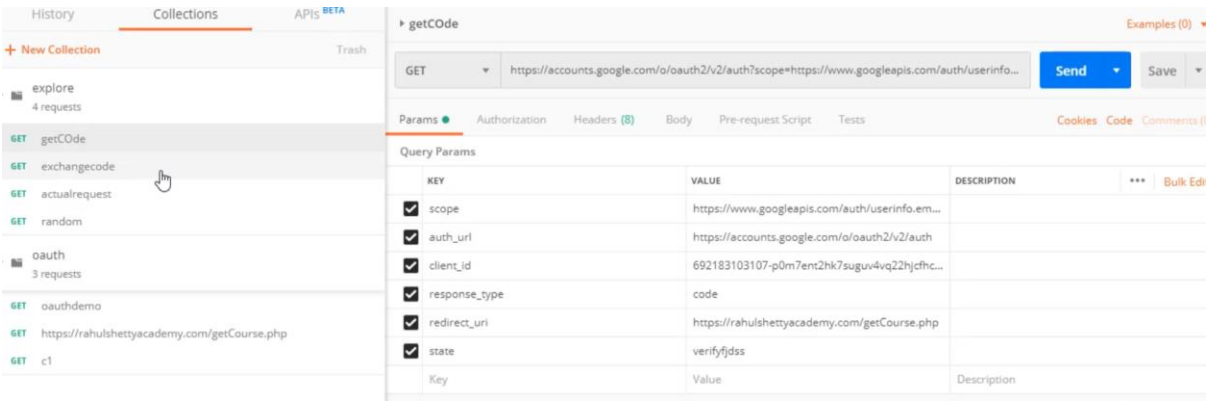
After authorisation:

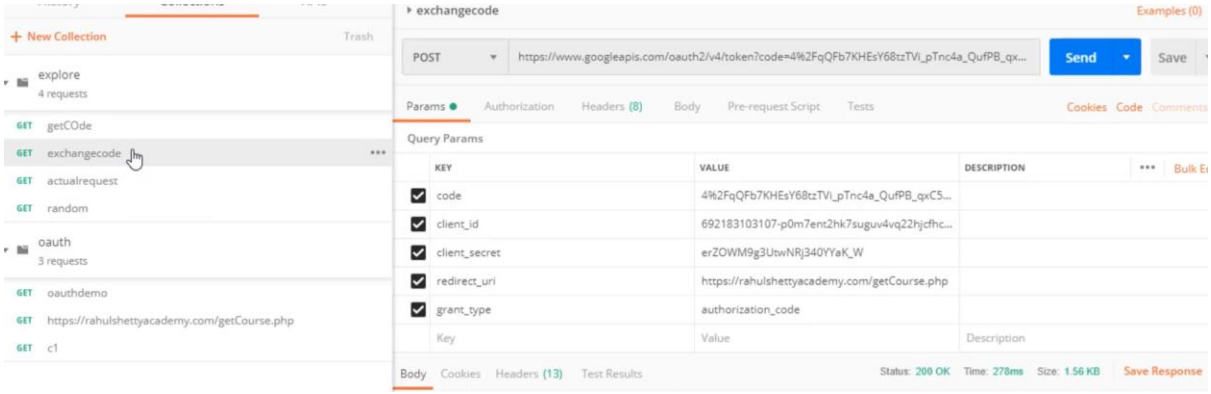
Copy url and paste and it formatted like below, in state its giving whats we given and what we received both are same

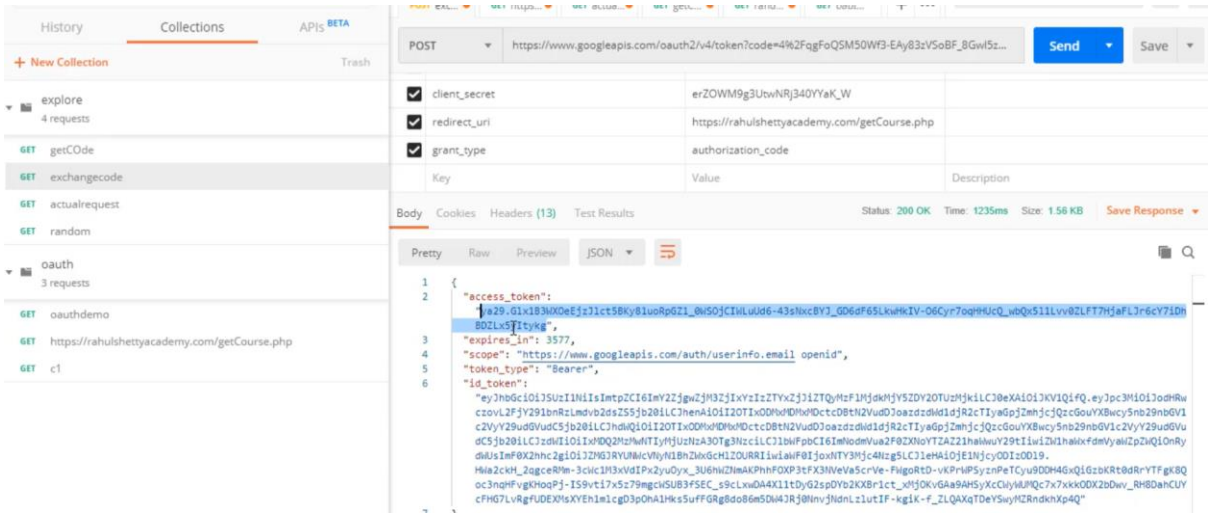


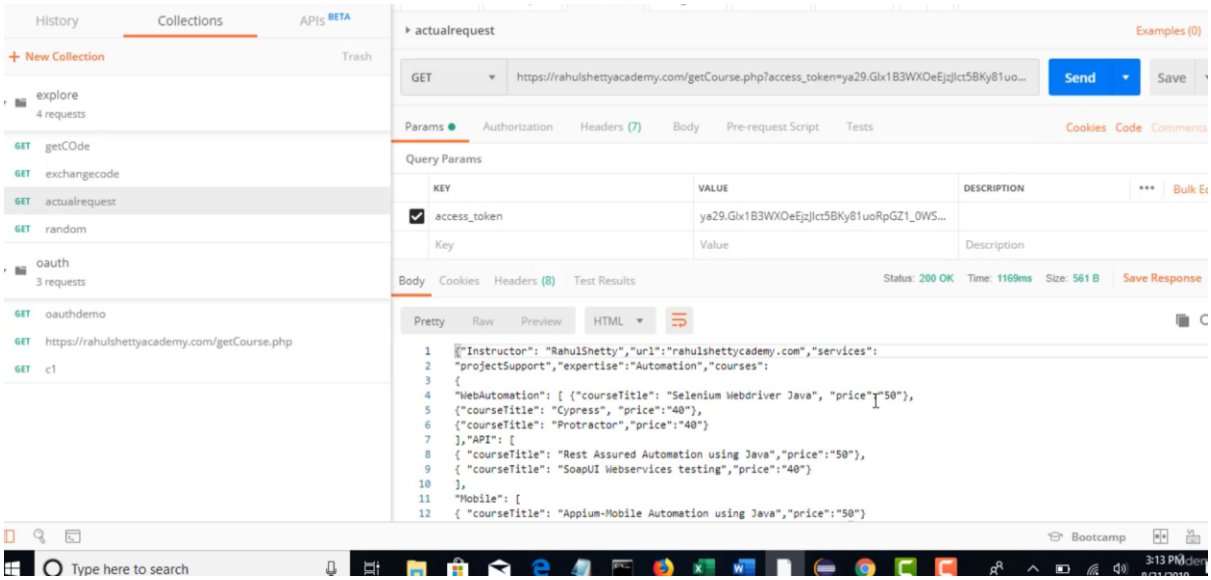

exchangecode
New Collection
Trash
explore
4 requests
GET getCOde
GET exchangecode
GET actualrequest
GET random
oauth
POST
https://www.googleapis.com/oauth2/v4/token?
code=4%2FqQFb7KHEsY68tzTVi_pTnc4a_QufPB_qxC5InCBYcFnYpbcyT_OyvBPCVHBZmWoVsB84n0y
mAkdKho2cP-kWprs&client_id=692183103107-
p0m7ent2hk7suguv4vq22hjcfhcr43pj.apps.googleusercontent.com&client_secret=erZOWM9g3Utw
NRj340YYaK_W&redirect_uri=https://rahulshettyacademy.com/getCourse.php&grant_type=authoriz
ation_code
Send
Params
Query Params
KEY
VALUE
DESCRIPTION
code
4%2FqQFb7KHEsY68tzTVi_pTnc4a_QufPB_qxC5...
client_id
692183103107-p0m7ent2hk7suguv4vq22hjcfhc...

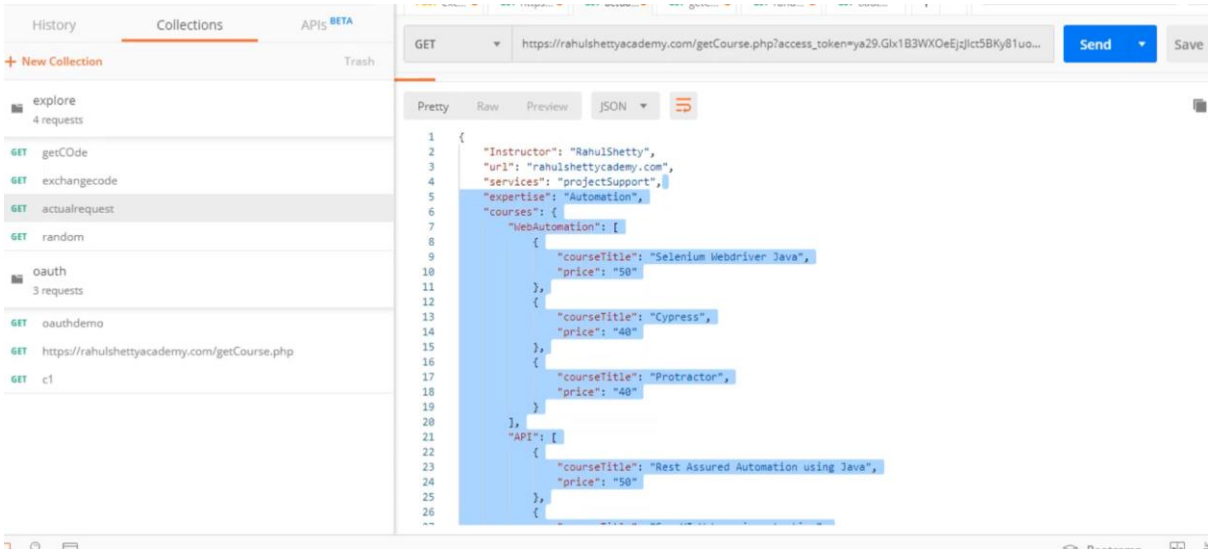

GET
https://accounts.google.com/o/oauth2/v2/auth?scope=https://www.googleapis.com/auth/userinfo...
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
scope
https://www.googleapis.com/auth/userinfo.em...
auth_url
https://accounts.google.com/o/oauth2/v2/auth
client_id
692183103107-p0m7ent2hk7suguv4vq22hjcfhc...
response_type
code
redirect_uri
https://rahulshettyacademy.com/getCourse.php
state
verifyfjdss


History
Collections
APIs BETA
New Collection
Trash
explore
4 requests
GET getCOde
GET exchangecode
GET actualrequest
GET random
oauth
3 requests
GET oauthdemo
GET https://rahulshettyacademy.com/getCourse.php
GET c1
getCOde
Examples (0)
GET
https://accounts.google.com/o/oauth2/v2/auth?scope=https://www.googleapis.com/auth/userinfo...
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
scope
https://www.googleapis.com/auth/userinfo.em...
auth_url
https://accounts.google.com/o/oauth2/v2/auth
client_id
692183103107-p0m7ent2hk7suguv4vq22hjcfhc...
response_type
code
redirect_uri
https://rahulshettyacademy.com/getCourse.php
state
verifyfjdss
Key
Value
Description


New Collection
Trash
explore
4 requests
GET getCOde
GET exchangecode
GET actualrequest
GET random
oauth
3 requests
GET oauthdemo
GET https://rahulshettyacademy.com/getCourse.php
GET c1
exchangecode
Examples (0)
POST
https://www.googleapis.com/oauth2/v4/token?code=4%2FqQFb7KHEsY68tzTVi_pTnc4a_QufPB_qx...
Send
Save
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
code
4%2FqQFb7KHEsY68tzTVi_pTnc4a_QufPB_qxC5...
client_id
692183103107-p0m7ent2hk7suguv4vq22hjcfhc...
client_secret
erZOWM9g3UtwNRj340YYaK_W
redirect_uri
https://rahulshettyacademy.com/getCourse.php
grant_type
authorization_code
Key
Value
Description
Body
Cookies
Headers (13)
Test Results
Status: 200 OK
Time: 278ms
Size: 1.56 KB
Save Response


History
Collections
APIs BETA
New Collection
Trash
explore
4 requests
GET getCOde
GET exchangecode
GET actualrequest
GET random
oauth
3 requests
GET oauthdemo
GET https://rahulshettyacademy.com/getCourse.php
GET c1
POST
https://www.googleapis.com/oauth2/v4/token?code=4%2FqgFoQSM50Wf3-EAy83zVSoBF_8Gwl5z...
Send
Save
client_secret
erZOWM9g3UtwNRj340YYaK_W
redirect_uri
https://rahulshettyacademy.com/getCourse.php
grant_type
authorization_code
Key
Value
Description
Body
Cookies
Headers (13)
Test Results
Status: 200 OK
Time: 1235ms
Size: 1.56 KB
Save Response
Pretty
Raw
Preview
JSON
{
"access_token":
"ya29.Glx1B3WXOeEjzJlct5BKy81uoRpGZ1_0WSOjCIWLuUd6-43sNxcBYJ_GD6dF6LkwHkIV-O6Cyr7oqHHUcQ_wbQx511Lvv0ZLFT7HjaFLJr6cY7iDM
BDZLx5JItyKg",
"expires_in": 3577,
"scope": "https://www.googleapis.com/auth/userinfo.email openid",
"token_type": "Bearer",
"id_token":
"eyJhbGciOiJSUzI1NiIsImtpZCI6ImY2ZjgwZjM3ZjIxYzIzZTYxZjJiZTQyMzF1MjdkMjY5ZDY2OTUzMjkiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRw
czovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiI2OTIxODMxMDMxMDctcDBtN2VudDJoazdzdWd1djR2cTIyaGpjZmhjcjQzcGouYXBwcy5nb29nbGV1
c2VyY29udGVudC5jb20iLCJhdWQiOiI2OTIxODMxMDMxMDctcDBtN2VudDJoazdzdWd1djR2cTIyaGpjZmhjcjQzcGouYXBwcy5nb29nbGV1c2VyY29udGVu
dC5jb20iLCJzdWIiOiIxMDQ2MzMwNTIyMjUzNzA3OTg3NzciLCJlbWFpbCI6ImNodmVua2F0ZXNoYTZAZ21haWwuY29tIiwiZW1haWxfdmVyaWZpZWQiOnRy
dWUsImF0X2hhc2giOiJZMGJRYUN4cVNyN1BhZWxGcH1ZOURRIiwiaWF0IjoxNTY3Mjc4Nzg5LCJleHAiOjE1NjcyODIzOD19.
Hwa2ckH_2qgceRMm-3cWc1M3xVdIPx2yuOyx_3U6hWZNmAKPhhFOXP3tFX3NVeVa5crVe-FWgoRtD-vKPrWPSyznPeTCyu9DDH4GxQiGzbKRt0dRrYTFgK8Q
oc3nqHFvgKHoqPj-IS9vti7x5z79mgcWSUB3f5EC_s9cLxwDA4X11tDyG2spOYb2KXBr1ct_xMjOKvGAa9AHSyXcCWyMJMQc7x7xkkODX2bDwv_RH8DahCUY
cFHG7LvRgfUDEXMsXYEh1mlcgD3pOhA1Hks5ufFGRg8do86m5DW4JRj0NnvjNdnLz1utIF-kgiK-f_ZLQAXqTDeYSwyMZRndkhXp4Q"


History
Collections
APIs BETA
New Collection
Trash
explore
4 requests
GET getCOde
GET exchangecode
GET actualrequest
GET random
oauth
3 requests
GET oauthdemo
GET https://rahulshettyacademy.com/getCourse.php
GET c1
actualrequest
Examples (0)
GET
https://rahulshettyacademy.com/getCourse.php?access_token=ya29.Glx1B3WXOeEjzJlct5BKy81uo...
Send
Save
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Cookies
Code
Query Params
KEY
VALUE
DESCRIPTION
access_token
ya29.Glx1B3WXOeEjzJlct5BKy81uoRpGZ1_0WS...
Key
Value
Description
Body
Cookies
Headers (8)
Test Results
Status: 200 OK
Time: 1169ms
Size: 561 B
Save Response
Pretty
Raw
Preview
HTML
{"Instructor": "RahulShetty","url":"rahulshettycademy.com","services":
"projectSupport","expertise":"Automation","courses":
{
"WebAutomation": [{"courseTitle": "Selenium Webdriver Java", "price":"50"},
{"courseTitle": "Cypress", "price":"40"},
{"courseTitle": "Protractor","price":"40"}
],"API": [
{ "courseTitle": "Rest Assured Automation using Java","price":"50"},
{ "courseTitle": "SoapUI Webservices testing","price":"40"}
],
"Mobile": [
{ "courseTitle": "Appium-Mobile Automation using Java","price":"50"}
Bootcamp
Type here to search
3:13 PM



OAuth 2.0 Contract Details:

GrantType	Authorization code
redirect URL/Callback URL	https://rahulshettyacademy.com/getCourse.php
Authorization server url	https://accounts.google.com/o/oauth2/v2/auth
Access token url	https://www.googleapis.com/oauth2/v4/token
Client ID	692183103107-p0m7ent2hk7suguv4vq22hjcfhcr43pj.apps.googleusercontent.com
Client Secret	erZOWM9g3UtwNRj340YYaK_W
Scope	https://www.googleapis.com/auth/userinfo.email
State	Any random string
How to pass oauth in request	Headers

Mandatory fields for GetAuthorization Code Request ;

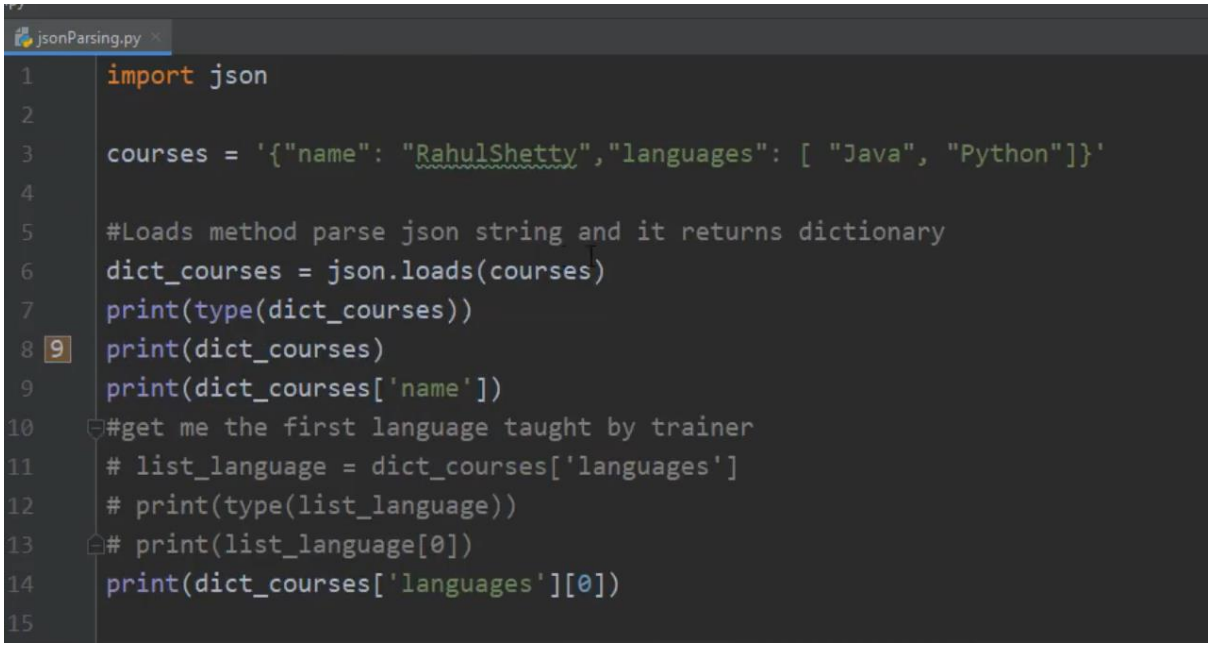
End Point : Authorization server url
Query Params:Scope, Auth_url, client_id, response_type, redirect_uri

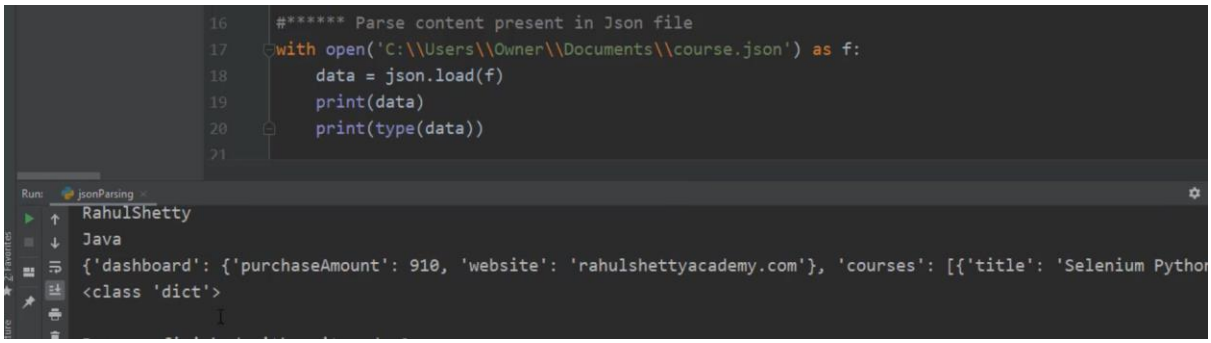
output : Code

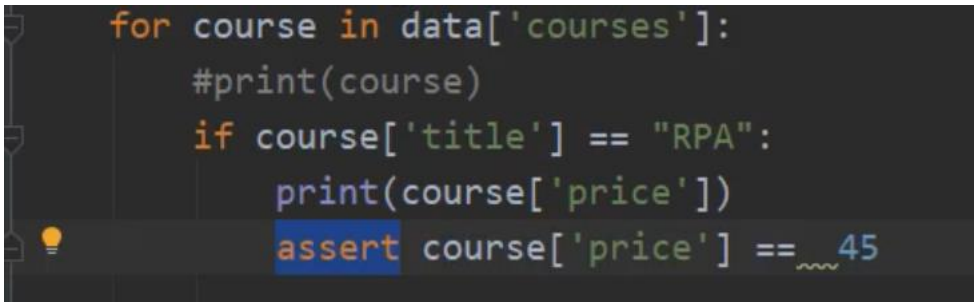
Mandatory fields for GetAccessToken Request :
End point : Access token url
Query Params :Code, client_id, client_secret, redirect_uri, grant_type

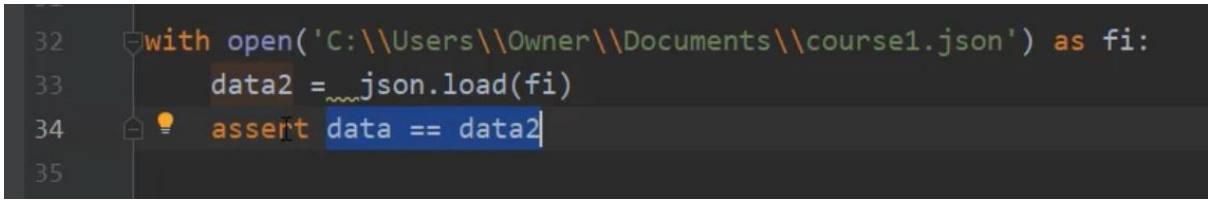
Output : Access token

Python Automation:

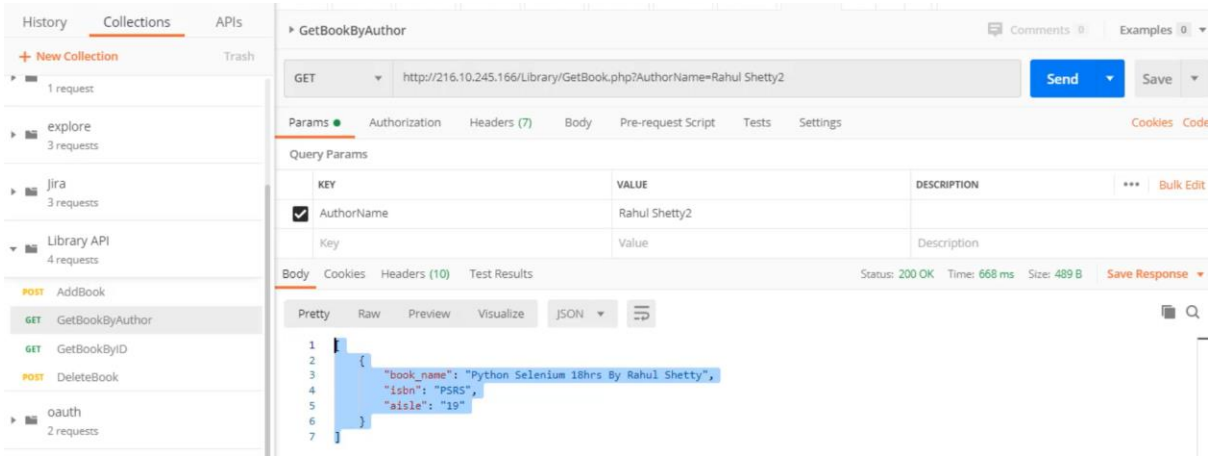


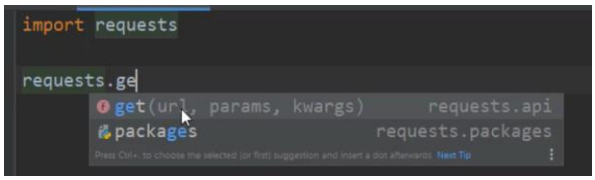




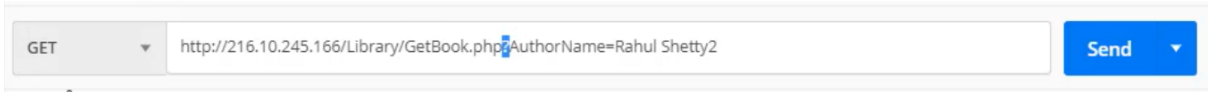


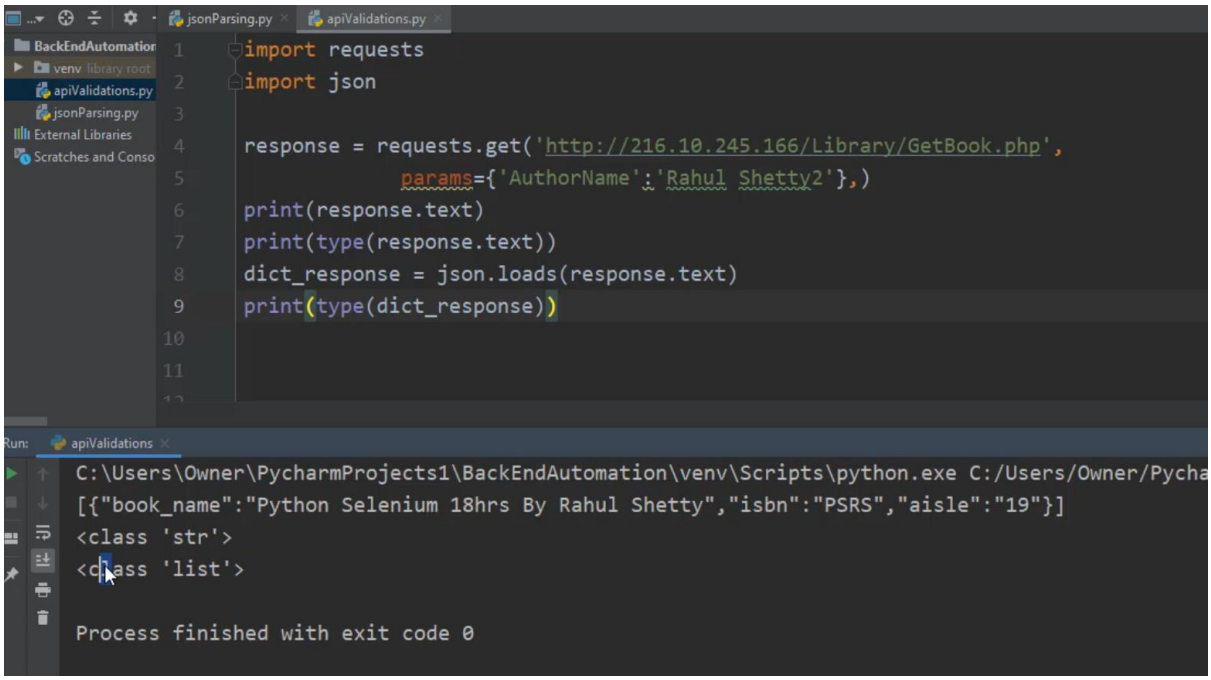
Get book by author name

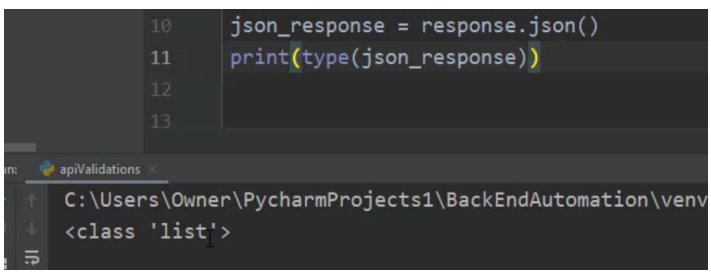




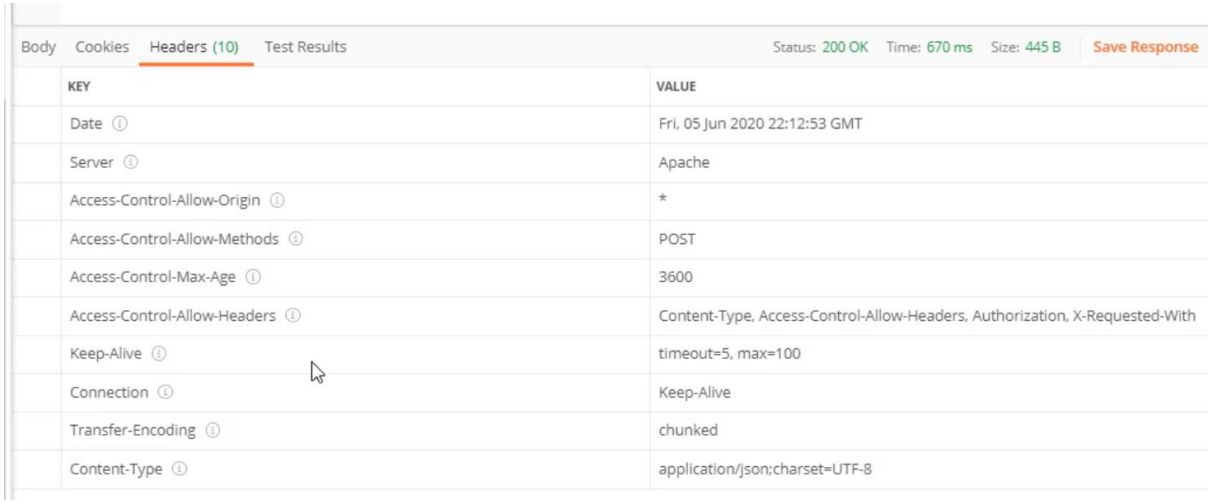
In get method url and parameters are separated by ?

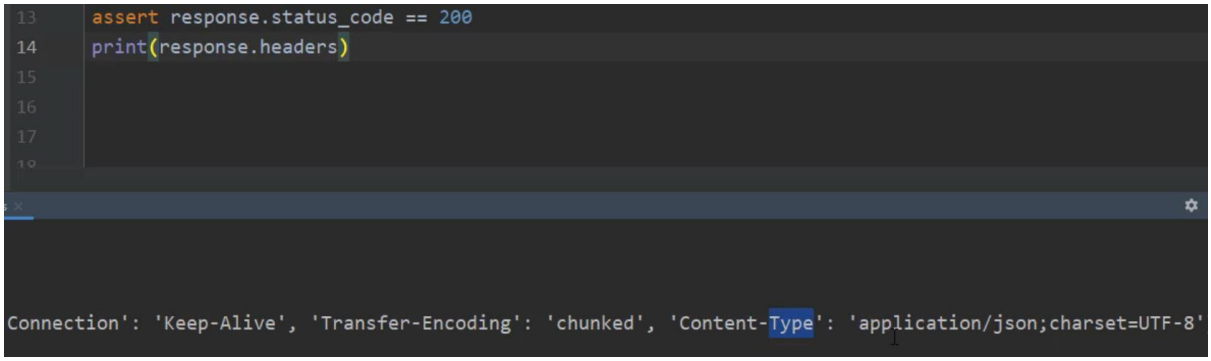






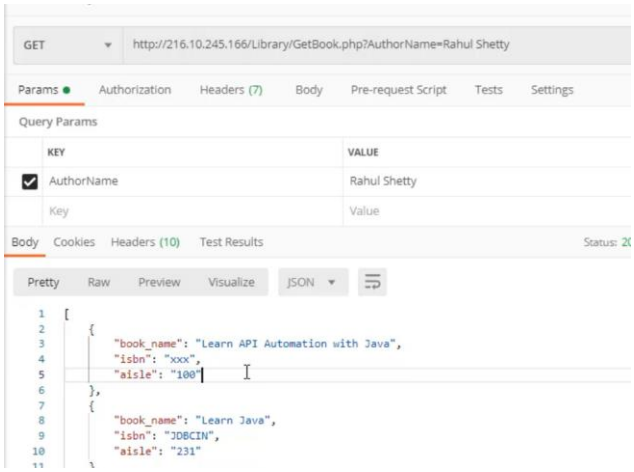
Headers in postman





Sometimes we need to verify that what is the content type we received

Getting book by author name and check isbn of the book and if exist retrieve the book details



json_response = response.json()

it gives list of dict.

```python
# Retrieve the book details with ISBN RGHCC
for actualBook in json_response:
    if actualBook['isbn'] == 'RGHCC':
        print(actualBook)
        break

expectedBook = {
        "book_name": "Learn API Automation with RestAssured",
        "isbn": "RGHCC",
        "aisle": "12239"
    }

assert actualBook == expectedBook
```

```
k_name': 'Learn API Automation with RestAssured', 'isbn': 'RGHCC', 'aisle': '12239'}

ss finished with exit code 0
```

POST API

Library API :

BaseURI : http://216.10.245.166

1. **Resource** : Library/AddBook.php **Method** : POST

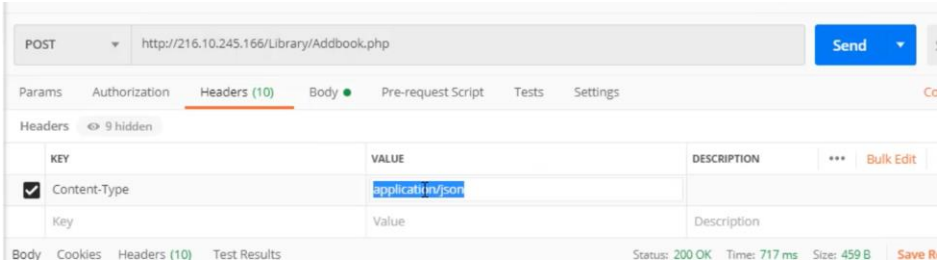
Input Payload : Json:

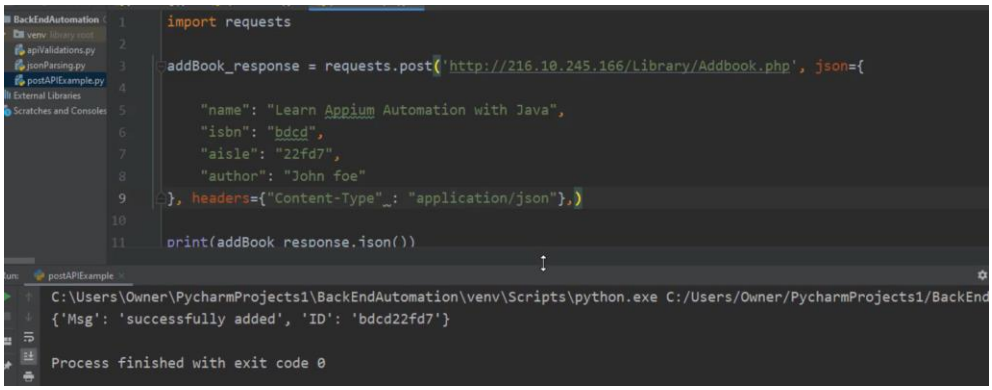
```
{
"name":"Learn Appium Automation with Java",
"isbn":"bcd",
"aisle":"227",
"author":"John foe"
}
```

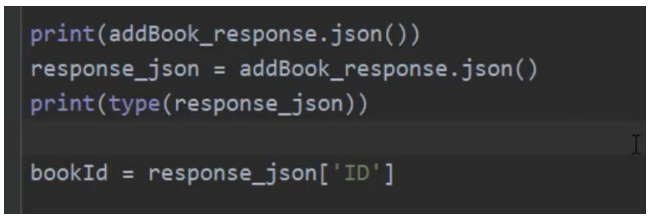
Output Json

```
{
  "Msg": "successfully added",
  "ID": "bcd227"
}
```

```python
import requests

requests.post
    post(url, data, json, kwar...  requests.api
Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip
```







1. **Resource** :/Library/DeleteBook.php **Method** : POST

Input Payload : Json:

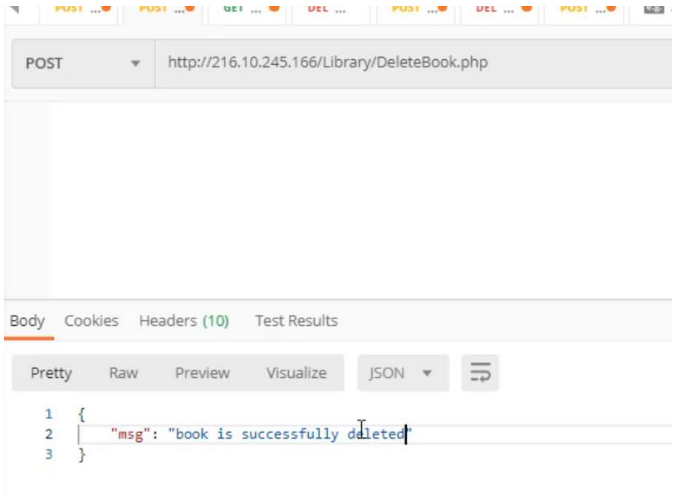
{

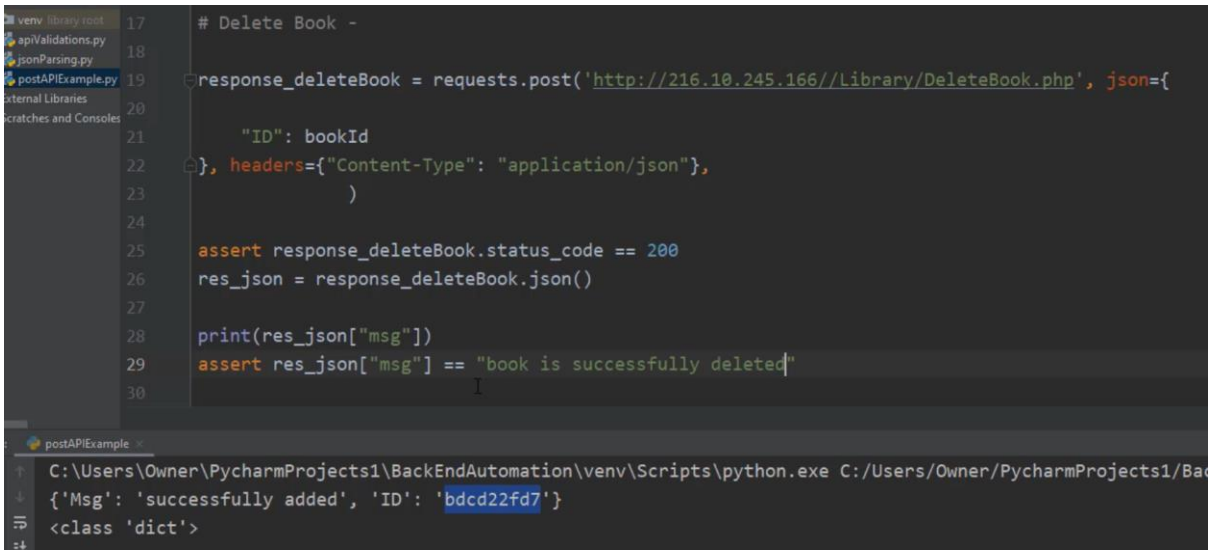
"ID" : "a23h345122332"

}

Output Response :

Using the POST method for deleting data is not a recommended practice according to the HTTP specification. The DELETE method is specifically designed for deletion operations in RESTful APIs. However, there might be cases where the DELETE method is not allowed due to various reasons such as restrictions imposed by some web servers or frameworks, or limitations in certain environments. In such cases, developers might resort to using the POST method with a designated endpoint or parameter to perform deletion operations. However, it's important to note that this deviates from standard RESTful API conventions and may not be the best practice.

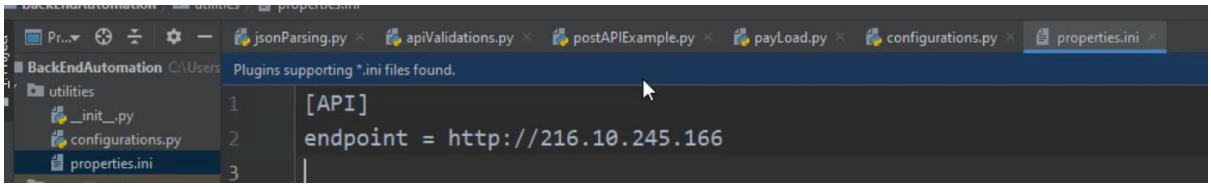


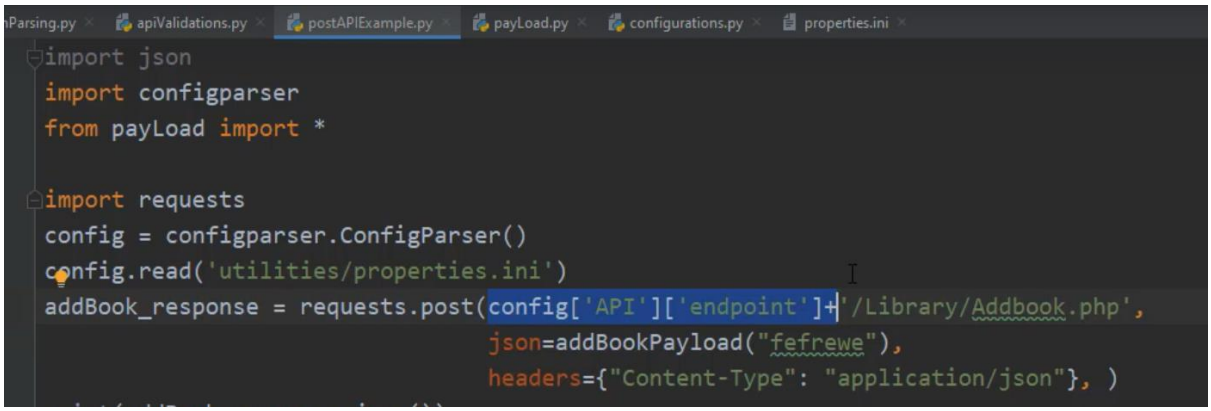


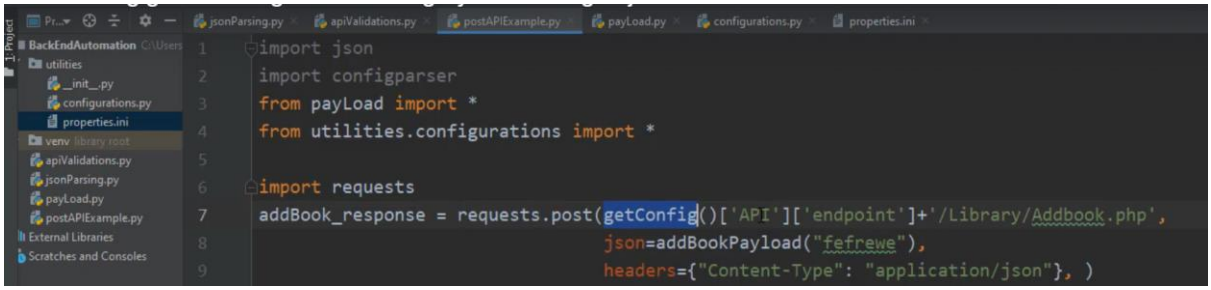

```
from flask import Flask, request,
jsonify

app = Flask(__name__)

@app.route('/data', methods=['POST'])
def receive_data():
    # Ensure that the request has a
JSON content type
    if request.headers['Content-
Type'] != 'application/json':
        return jsonify({'error':
'Unsupported Media Type'}), 415

    # Get the JSON data from the
request body
    data = request.json

    # Process the data
    # For demonstration purposes,
simply echo back the received data
    return jsonify(data)

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

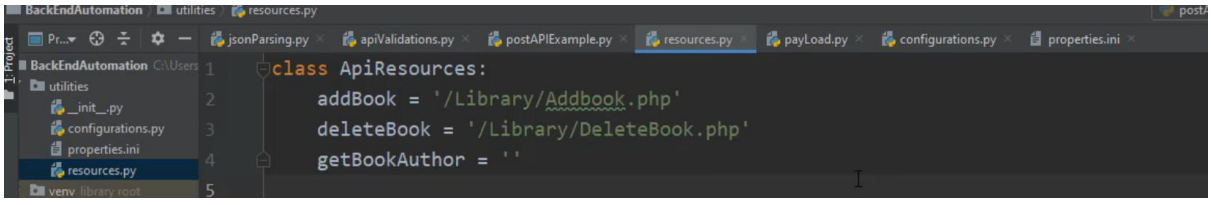
In headers we are sending content type based on content type which received from api backend code where designed so for integration process we used content type

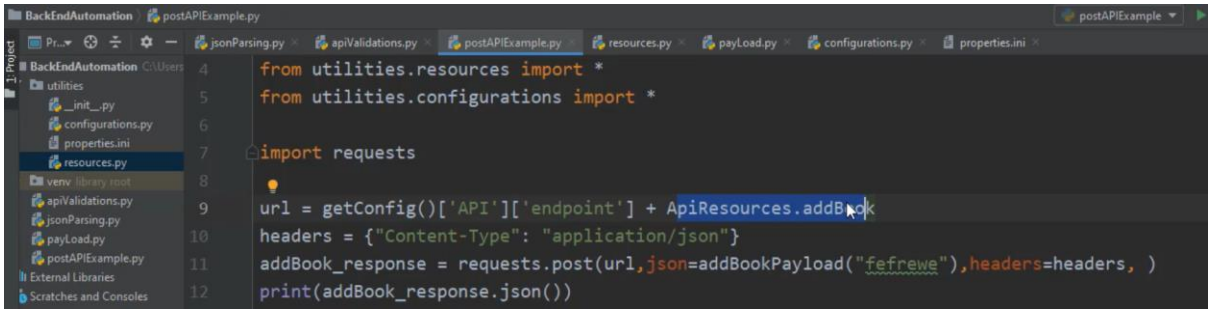
Creating Global Variables:

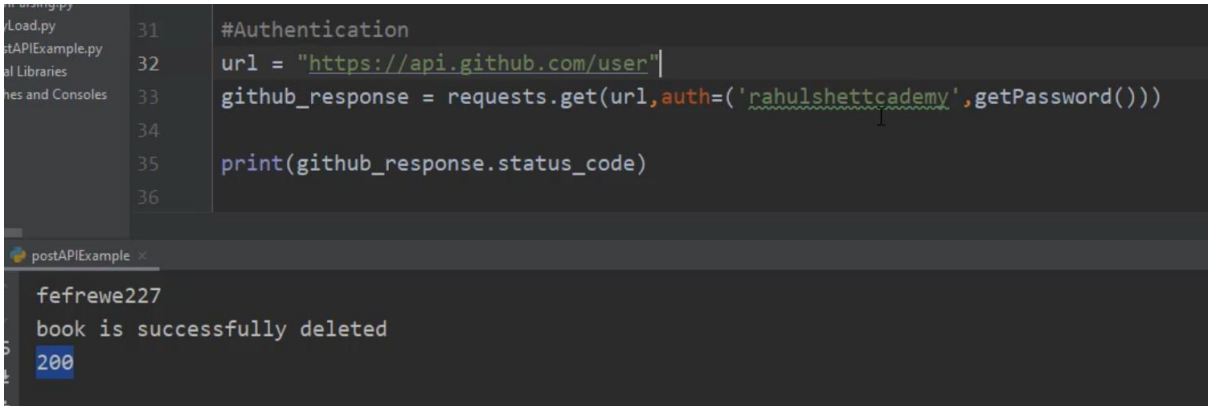


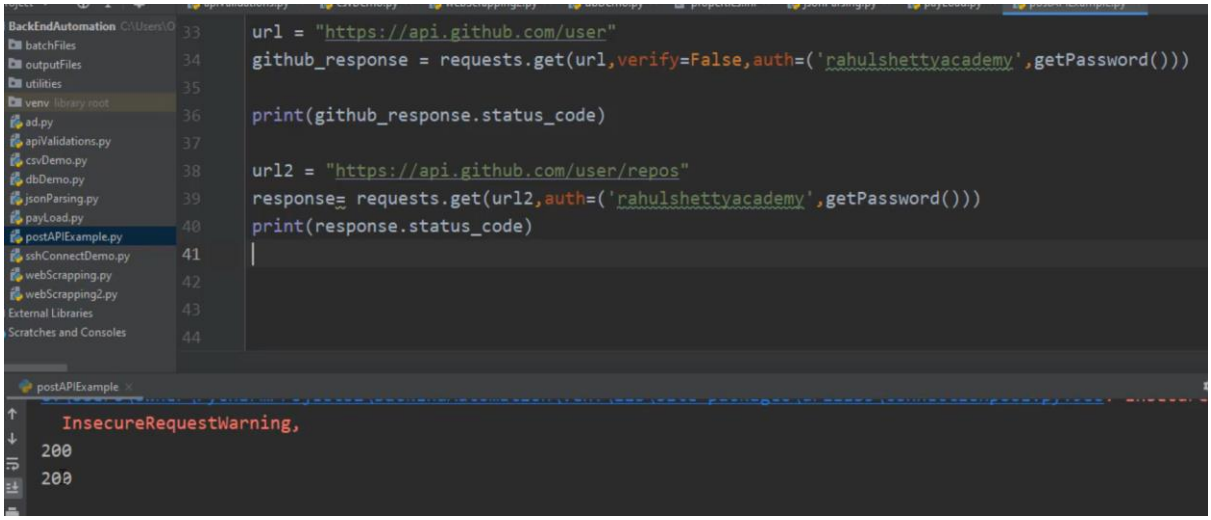




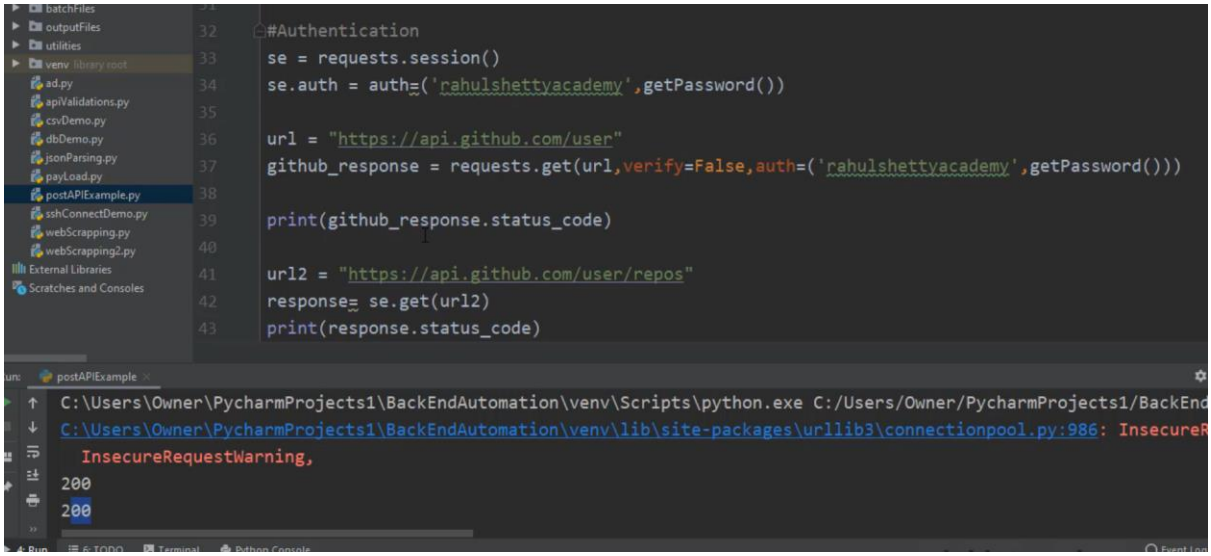




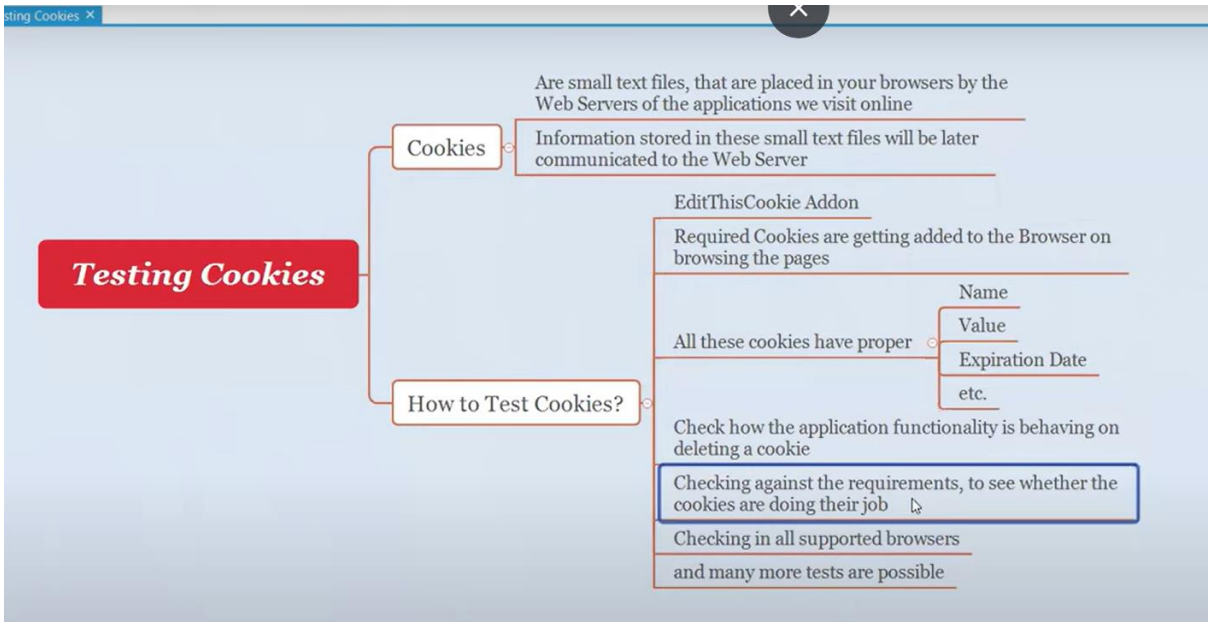


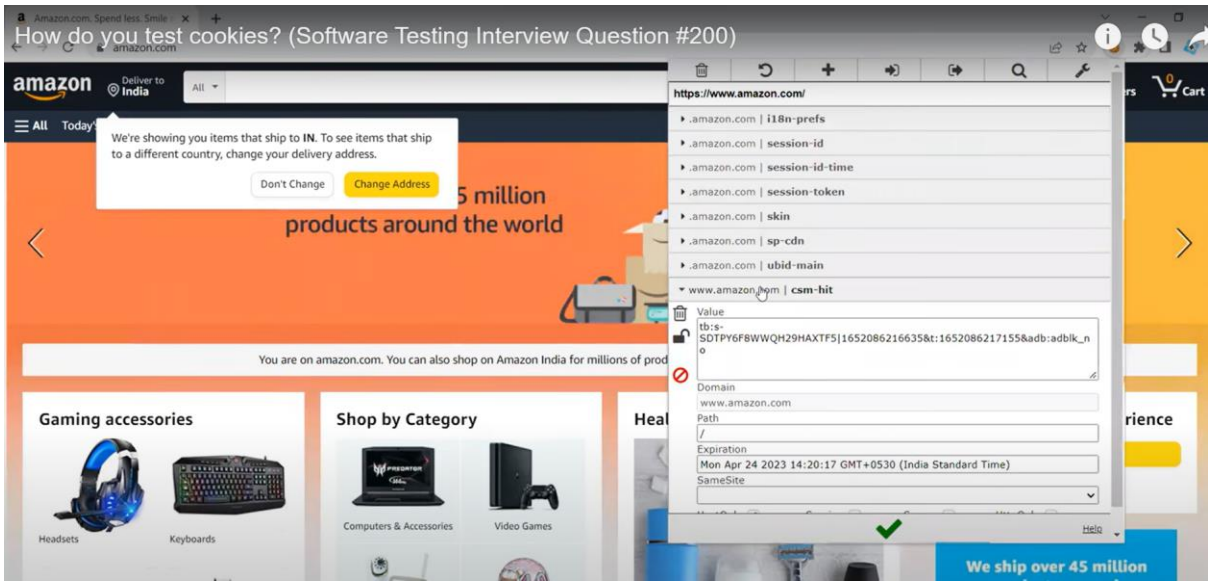


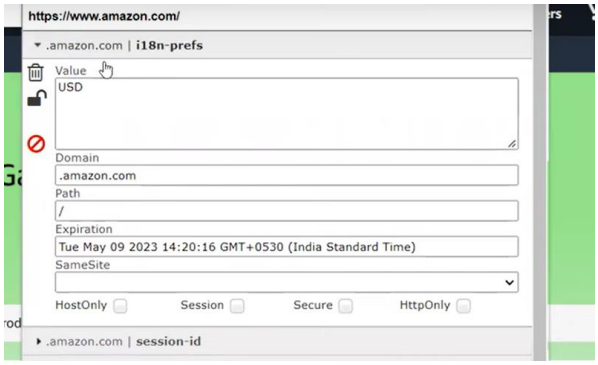
if we want to access multiple apis having auth we have to pass auth in every request or else we can create session


#Authentication
se = requests.session()
se.auth = auth=('rahulshettyacademy',getPassword())
url = "https://api.github.com/user"
github_response = requests.get(url,verify=False,auth=('rahulshettyacademy',getPassword()))
print(github_response.status_code)
url2 = "https://api.github.com/user/repos"
response= se.get(url2)
print(response.status_code)
C:\Users\Owner\PycharmProjects1\BackEndAutomation\venv\Scripts\python.exe C:/Users/Owner/PycharmProjects1/BackEnd
C:\Users\Owner\PycharmProjects1\BackEndAutomation\venv\lib\site-packages\urllib3\connectionpool.py:986: InsecureR
InsecureRequestWarning,
200
200

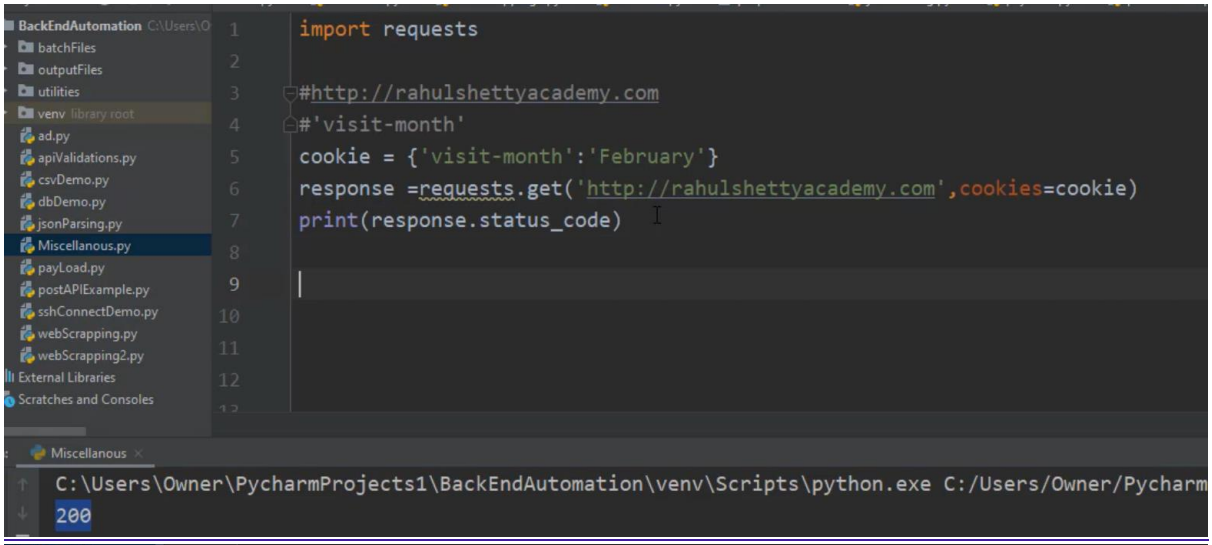
Cookies

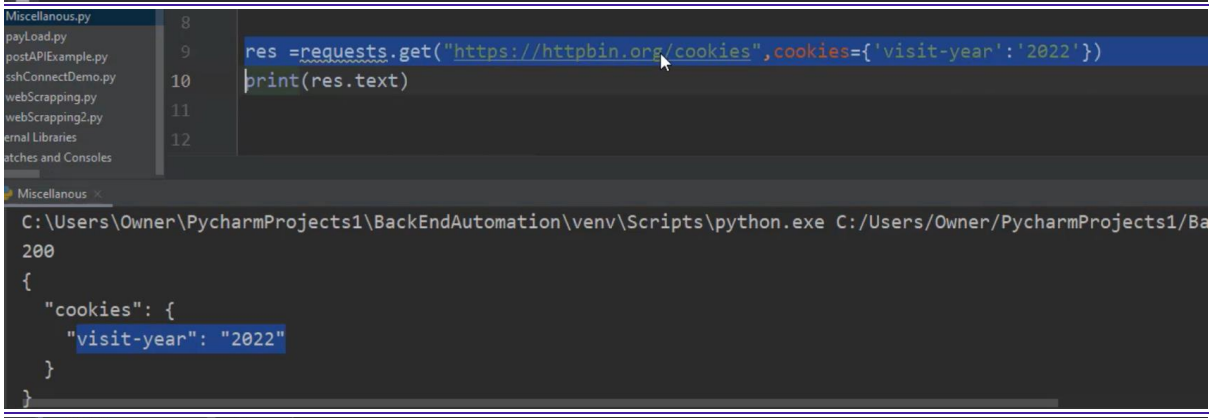

Testing Cookies
Cookies
Are small text files, that are placed in your browsers by the Web Servers of the applications we visit online
Information stored in these small text files will be later communicated to the Web Server
How to Test Cookies?
EditThisCookie Addon
Required Cookies are getting added to the Browser on browsing the pages
All these cookies have proper
Name
Value
Expiration Date
etc.
Check how the application functionality is behaving on deleting a cookie
Checking against the requirements, to see whether the cookies are doing their job
Checking in all supported browsers
and many more tests are possible

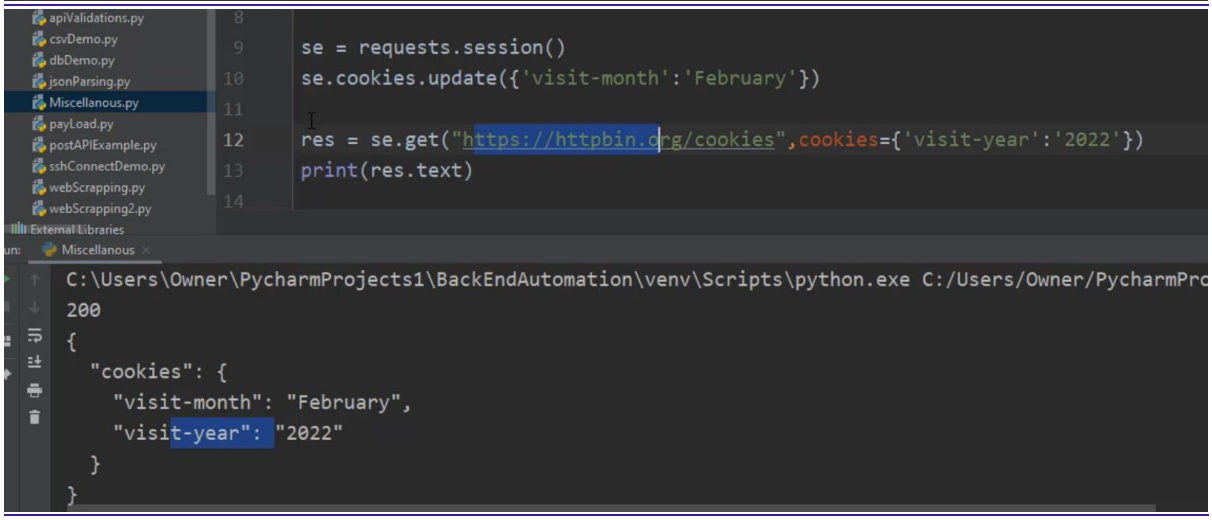

How do you test cookies? (Software Testing Interview Question #200)
amazon
Deliver to India
We're showing you items that ship to IN. To see items that ship to a different country, change your delivery address.
Don't Change
Change Address
products around the world
Gaming accessories
Headsets
Keyboards
Shop by Category
Computers & Accessories
Video Games
https://www.amazon.com/
.amazon.com | i18n-prefs
.amazon.com | session-id
.amazon.com | session-id-time
.amazon.com | session-token
.amazon.com | skin
.amazon.com | sp-cdn
.amazon.com | ubid-main
www.amazon.com | csm-hit
Value
Domain
www.amazon.com
Path
/
Expiration
Mon Apr 24 2023 14:20:17 GMT+0530 (India Standard Time)
SameSite
We ship over 45 million



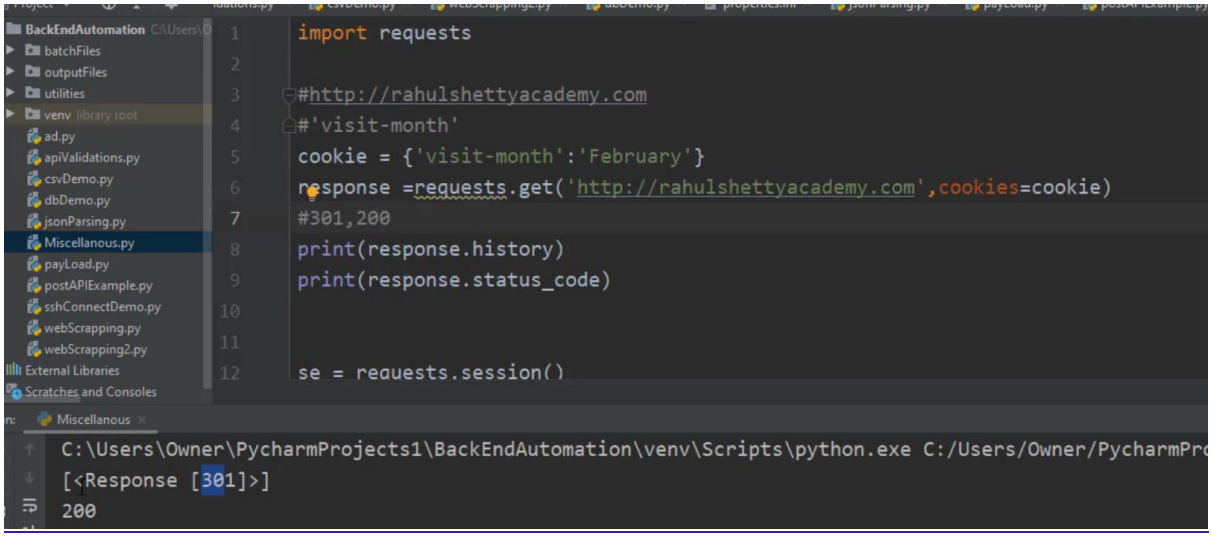
There is one my website with Http.
Rahul Shetty.
Academy.com.
Right.
So my website have one cookie.
So basically when people land, we will see what is the visit month that they visited last.
Okay.
So if their visit month is, let's say this is August and if they have visited this in June, so if
there are no site changes after June, we will load the website from the cache, what they already have
so that response time will be fast.
Okay.
And if you have updated our website after June and if the visit month is June, so then we will bring
this website from the real server without caching.
So that's where we use cache in our website and.
To track what is the visit month?
We use cookies, so whenever people land on web page, we use a cookie called Visit month.
So this is a common like every website have a number of cookies to give best experience to the users.
So whenever they land, we use a cookie called visit month.
So we will read this cookie.
And based upon that, as I told, we will load our web page.



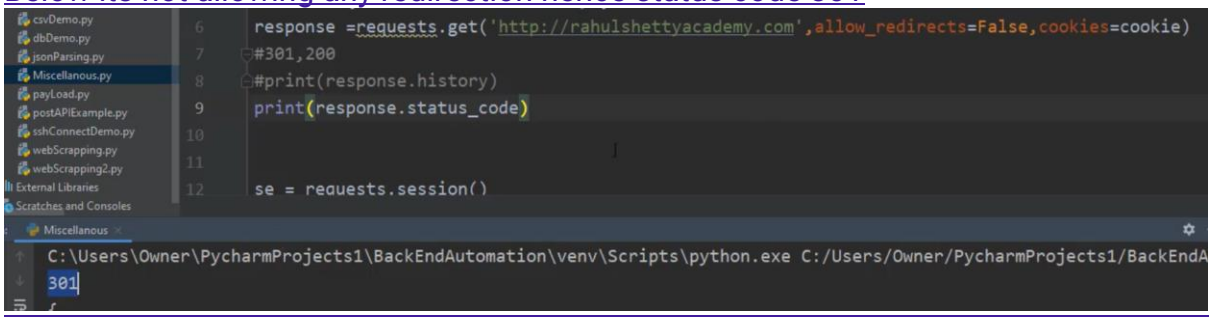




If any redirection present in url it gives 301 response code for security reasons redirects present , like all the security things mentioned in one url first it will hit that n then it redirects to other url, but if developers added all security to other also we need to make sure that url is not redirecting by checking response history or one parameter allow_redirects its bool



Below its not allowing any redirection hence status code 301



Timeouts

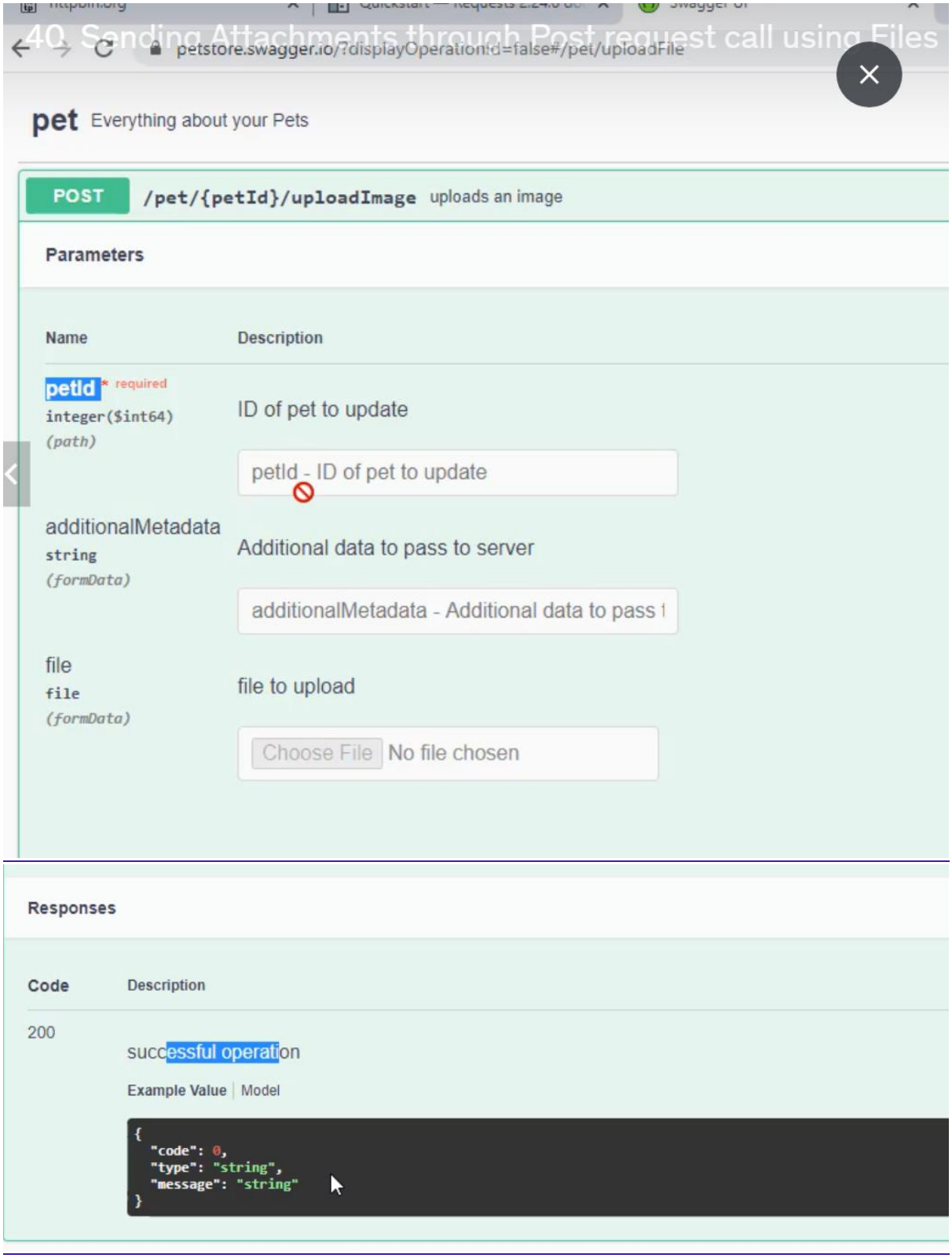
POST a Multipart-Encoded File

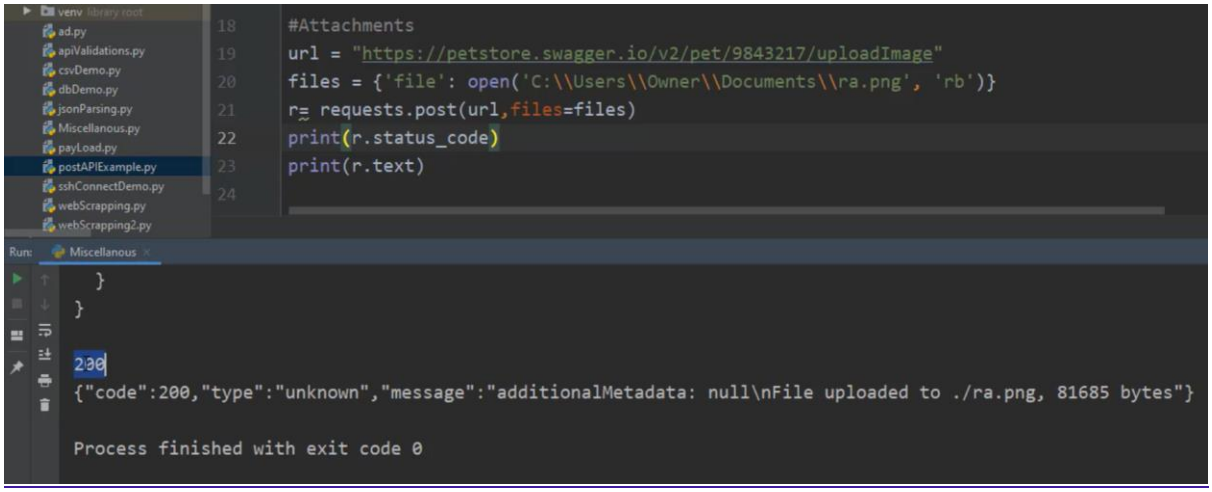
Requests makes it simple to upload Multipart-encoded files:

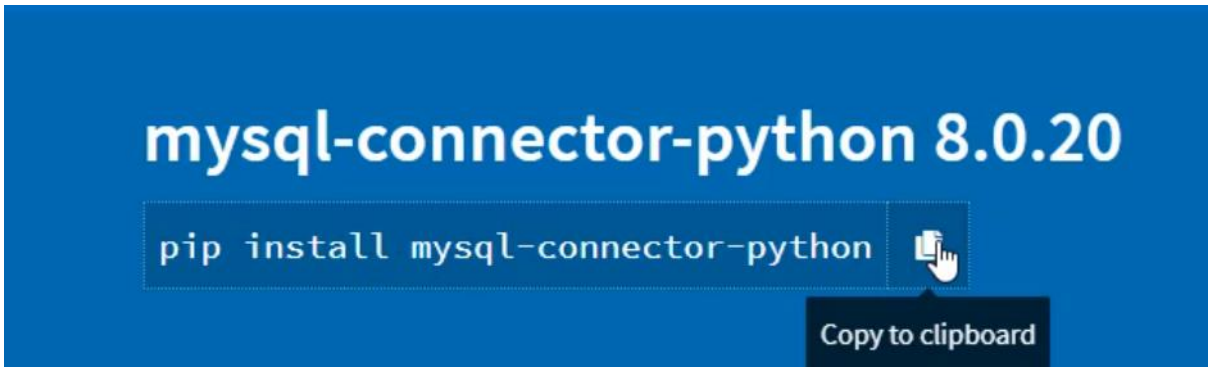
```
>>> url = 'https://httpbin.org/post'
>>> files = {'file': open('report.xls', 'rb')}

>>> r = requests.post(url, files=files)
>>> r.text
{
  ...
  "files": {
    "file": "<censored...binary...data>"
  },
  ...
}
```

You can set the filename, content_type and headers explicitly:

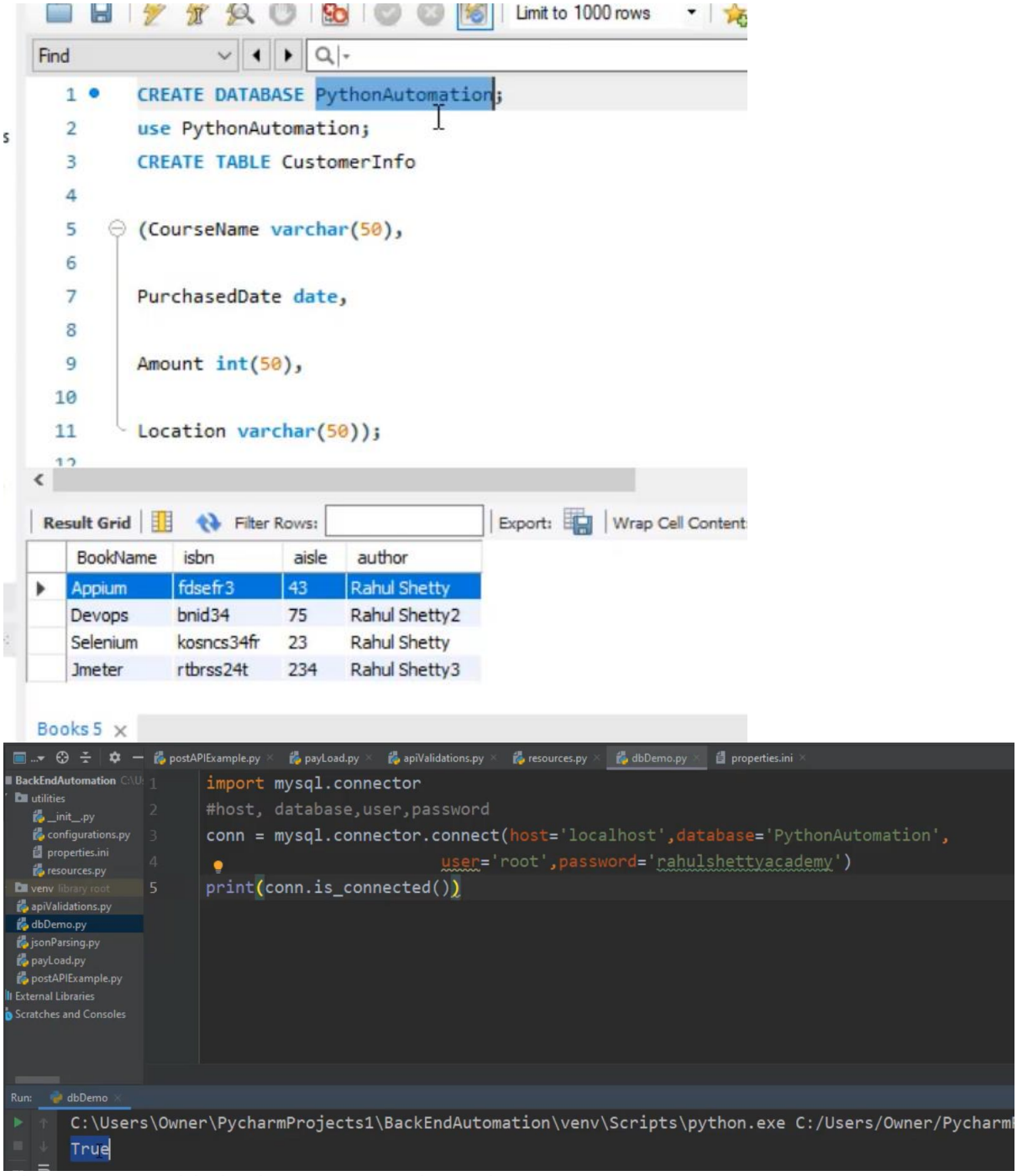
```
>>> url = 'https://httpbin.org/post'
>>> files = {'file': ('report.xls', open('report.xls', 'rb'), 'application/vnd.ms-exc

>>> r = requests.post(url, files=files)
>>> r.text
{
```


40. Sending Attachments through Post request call using Files
petstore.swagger.io/?displayOperationId=false#/pet/uploadFile
pet Everything about your Pets
POST /pet/{petId}/uploadImage uploads an image
Parameters
Name
Description
petId * required
integer($int64)
(path)
ID of pet to update
petId - ID of pet to update
additionalMetadata
string
(formData)
Additional data to pass to server
additionalMetadata - Additional data to pass t
file
file
(formData)
file to upload
Choose File No file chosen
Responses
Code
Description
200
successful operation
Example Value | Model
{
"code": 0,
"type": "string",
"message": "string"
}



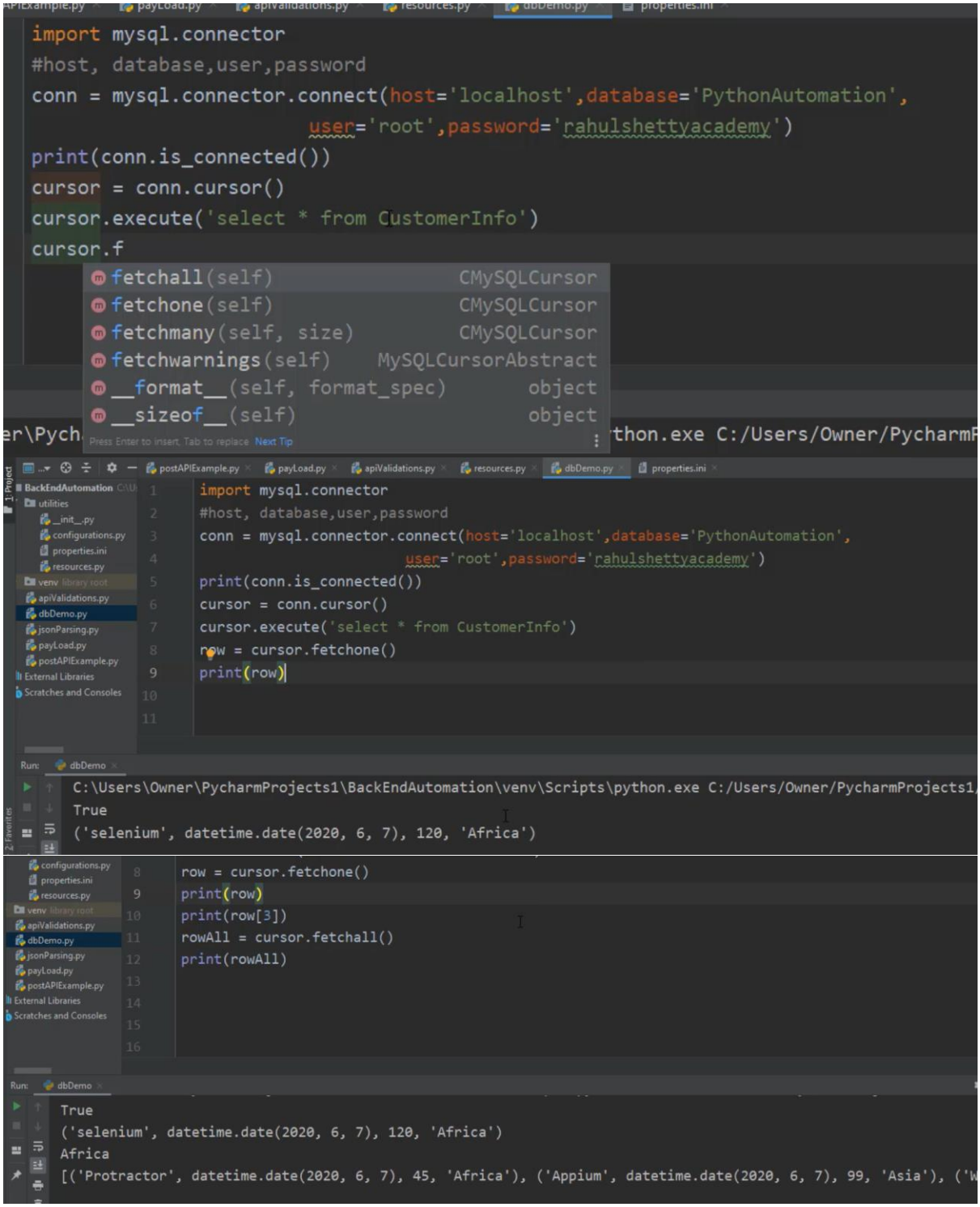


To connect mysql server

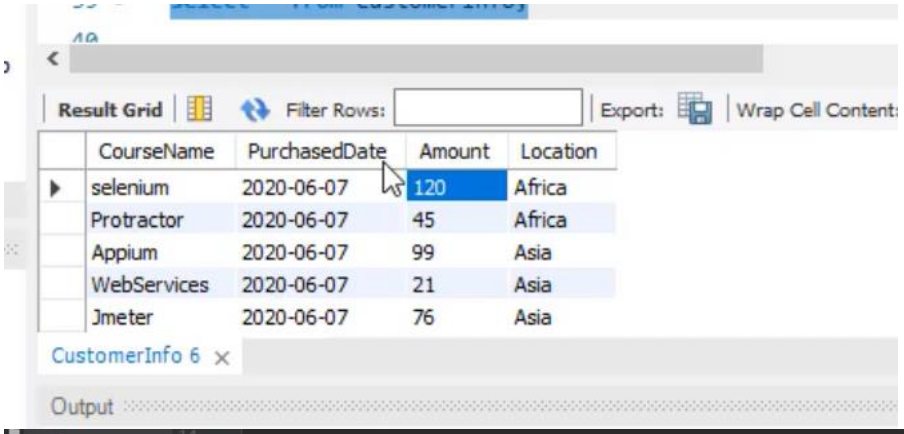


Above we established connection through conn

Below is cursor to send query and get result back



fetchall returns list of tuples and it started where cursor was there earlier



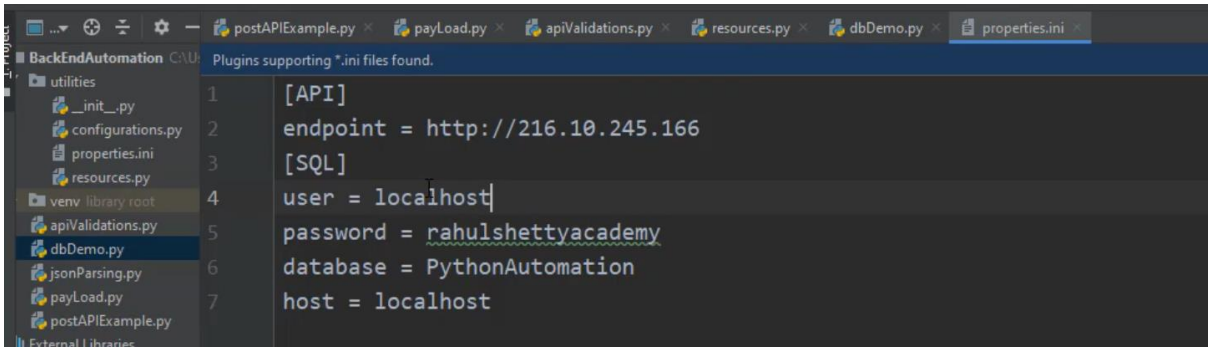

```
rows = cursor.fetchall()
print(type(rows))
print(rows)
for row in rows:  #('selenium', datetime.date(2020, 6, 7), 120, 'Africa')
    print(row[2])


conn.close()
```

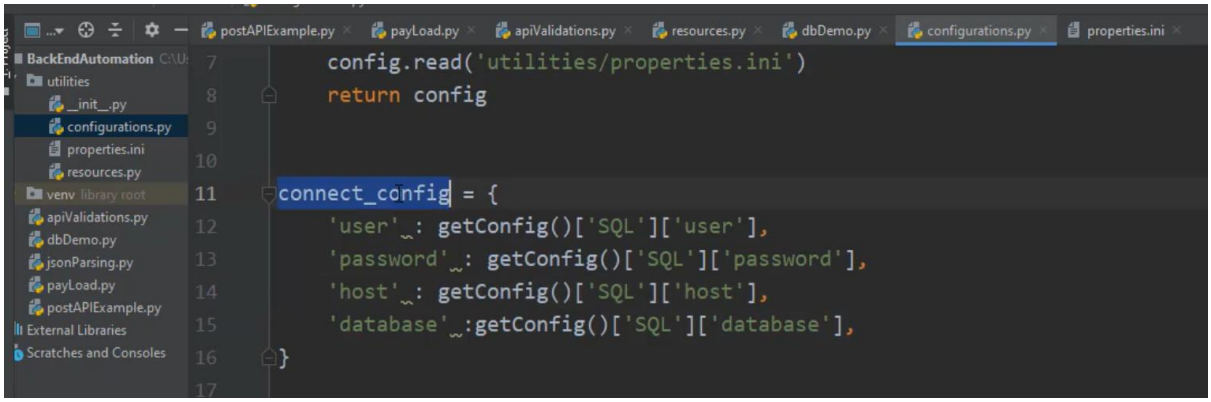
```
120
45
99
21
76
```

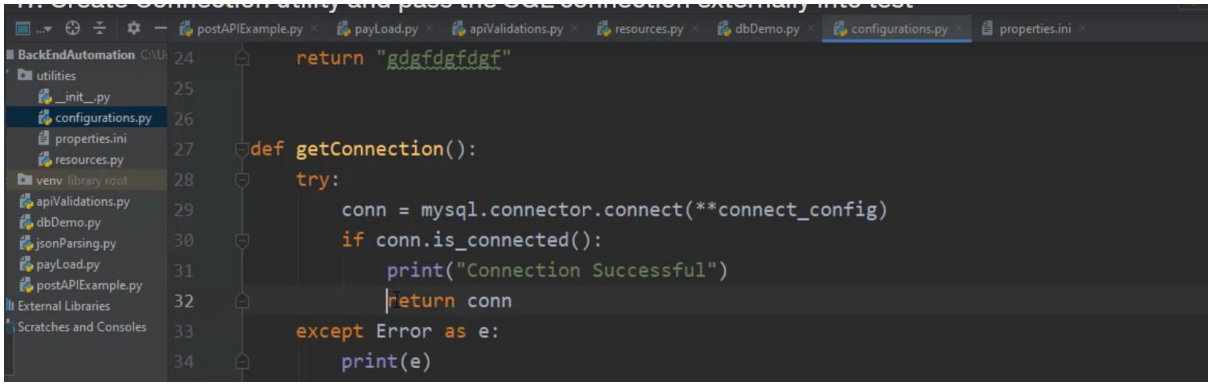
```
rows = cursor.fetchall()
print(type(rows))
print(rows)
sum = 0
for row in rows:  #('selenium', datetime.date(2020, 6, 7), 120, 'Africa')
    sum = sum + row[2]

print(sum)
```

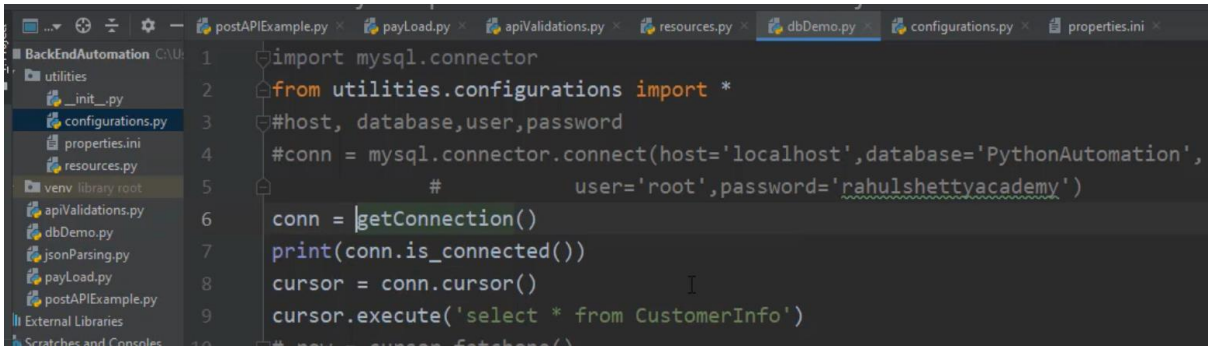
```
[('selenium', datetime.date(2020, 6, 7), 120, 'Africa'), ('Protractor', datetime.date(2020, 6, 7), 45
361
```

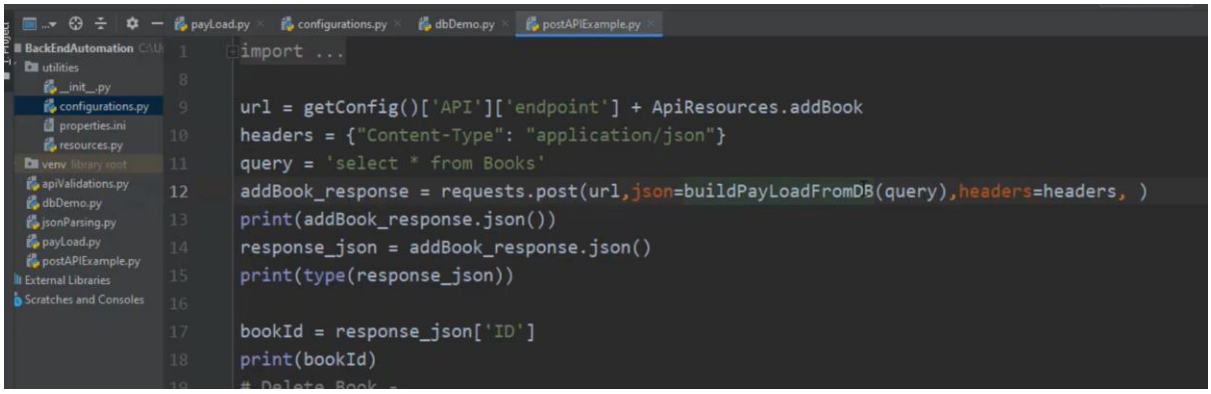
```
query = "update customerInfo set Location = %s where CourseName = %s"
data = ("UK","Jmeter")
cursor.execute(query,data)
conn.commit()
```

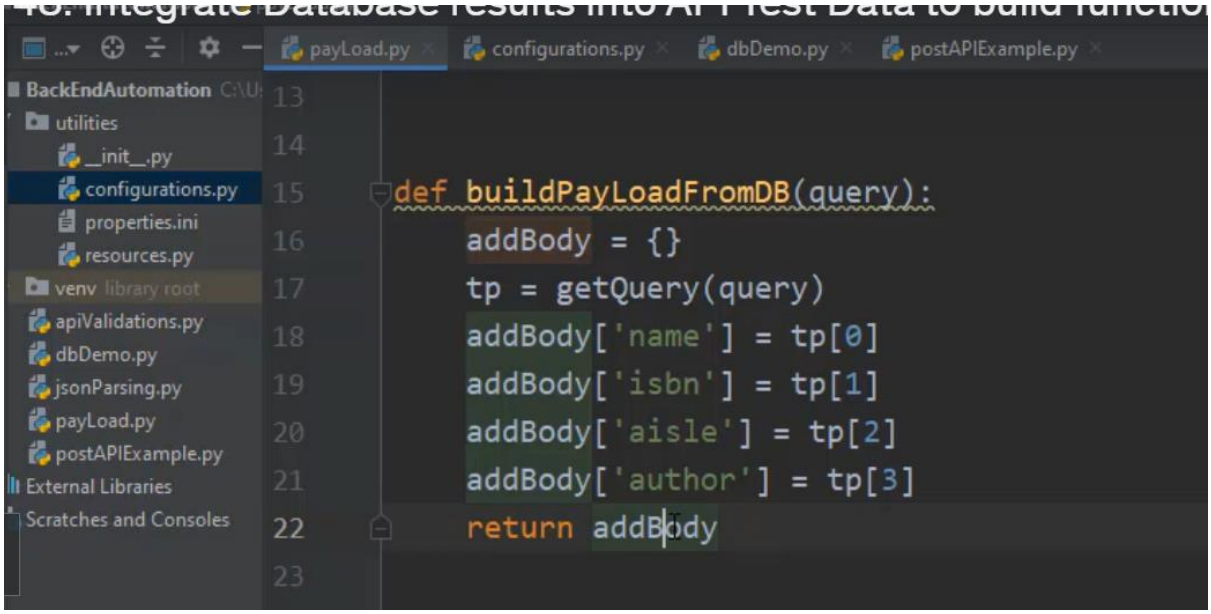

postAPIExample.py
payLoad.py
apiValidations.py
resources.py
dbDemo.py
properties.ini
Plugins supporting *.ini files found.
[API]
endpoint = http://216.10.245.166
[SQL]
user = localhost
password = rahulshettyacademy
database = PythonAutomation
host = localhost

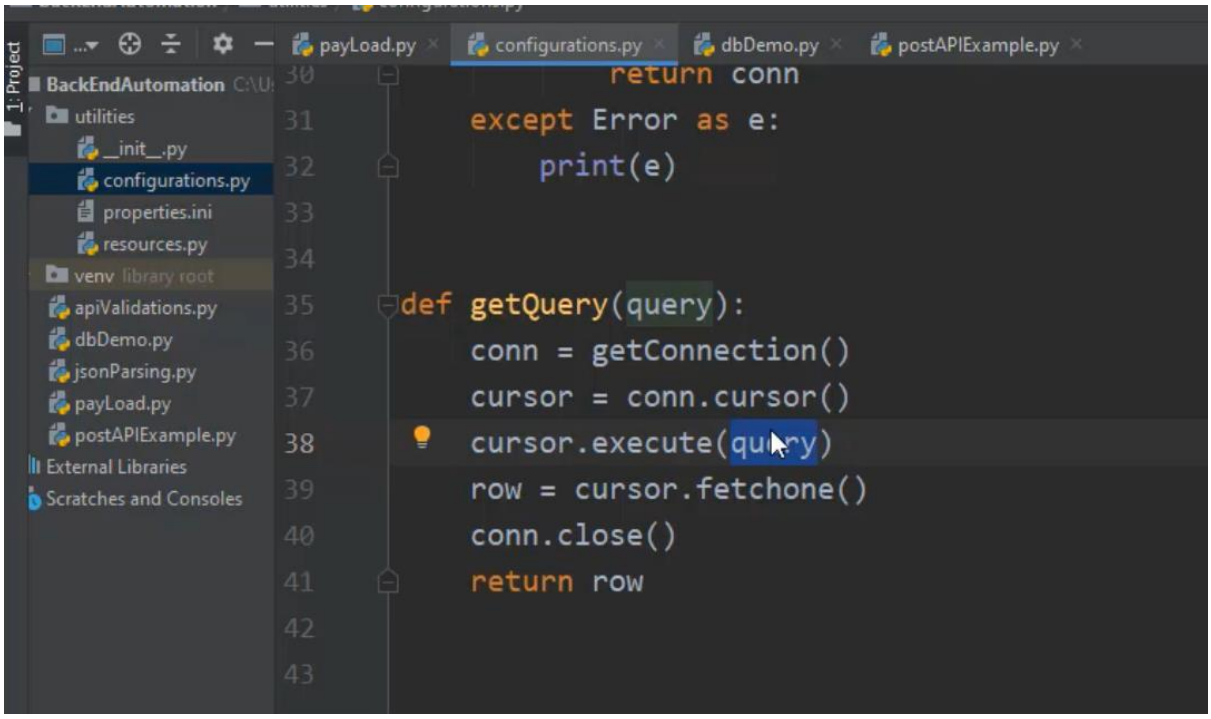

config.read('utilities/properties.ini')
return config
connect_config = {
'user': getConfig()['SQL']['user'],
'password': getConfig()['SQL']['password'],
'host': getConfig()['SQL']['host'],
'database':getConfig()['SQL']['database'],
}


return "gdgfdgfdgf"
def getConnection():
try:
conn = mysql.connector.connect(**connect_config)
if conn.is_connected():
print("Connection Successful")
return conn
except Error as e:
print(e)


import mysql.connector
from utilities.configurations import *
#host, database,user,password
#conn = mysql.connector.connect(host='localhost',database='PythonAutomation',
# user='root',password='rahulshettyacademy')
conn = getConnection()
print(conn.is_connected())
cursor = conn.cursor()
cursor.execute('select * from CustomerInfo')

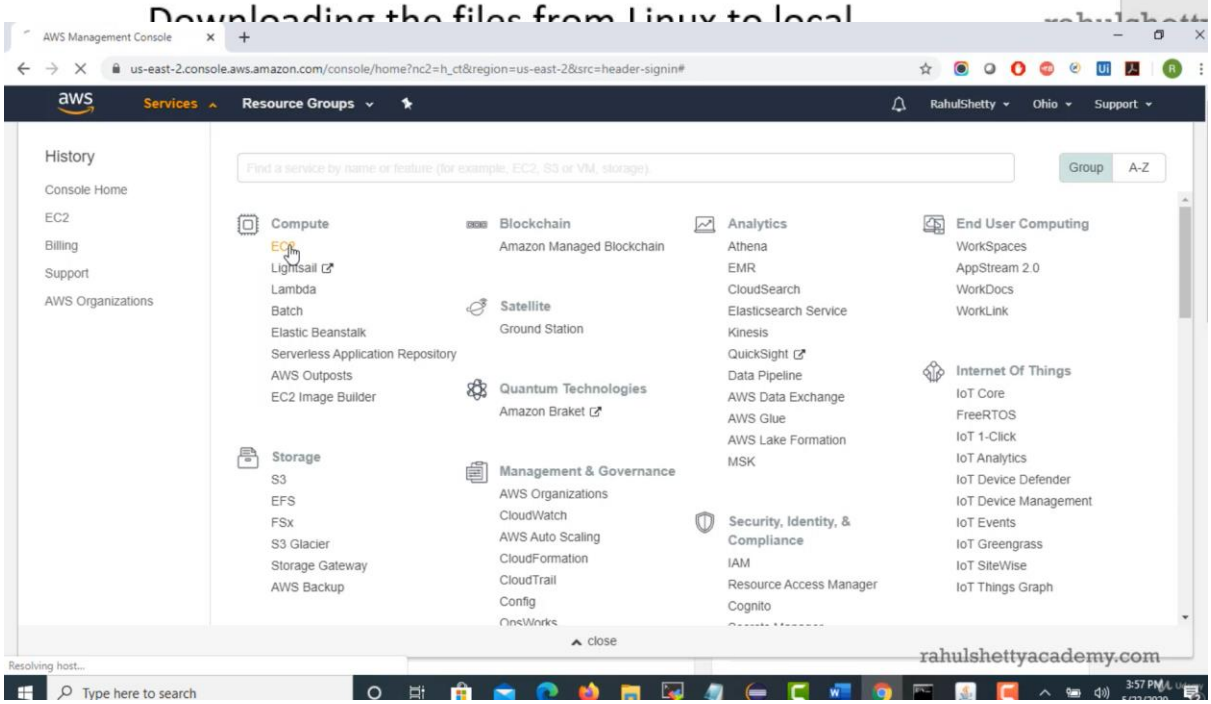

payLoad.py
configurations.py
dbDemo.py
postAPIExample.py
BackEndAutomation
utilities
__init__.py
configurations.py
properties.ini
resources.py
venv library root
apiValidations.py
dbDemo.py
jsonParsing.py
payLoad.py
postAPIExample.py
External Libraries
Scratches and Consoles
import ...
url = getConfig()['API']['endpoint'] + ApiResources.addBook
headers = {"Content-Type": "application/json"}
query = 'select * from Books'
addBook_response = requests.post(url,json=buildPayLoadFromDB(query),headers=headers,)
print(addBook_response.json())
response_json = addBook_response.json()
print(type(response_json))
bookId = response_json['ID']
print(bookId)

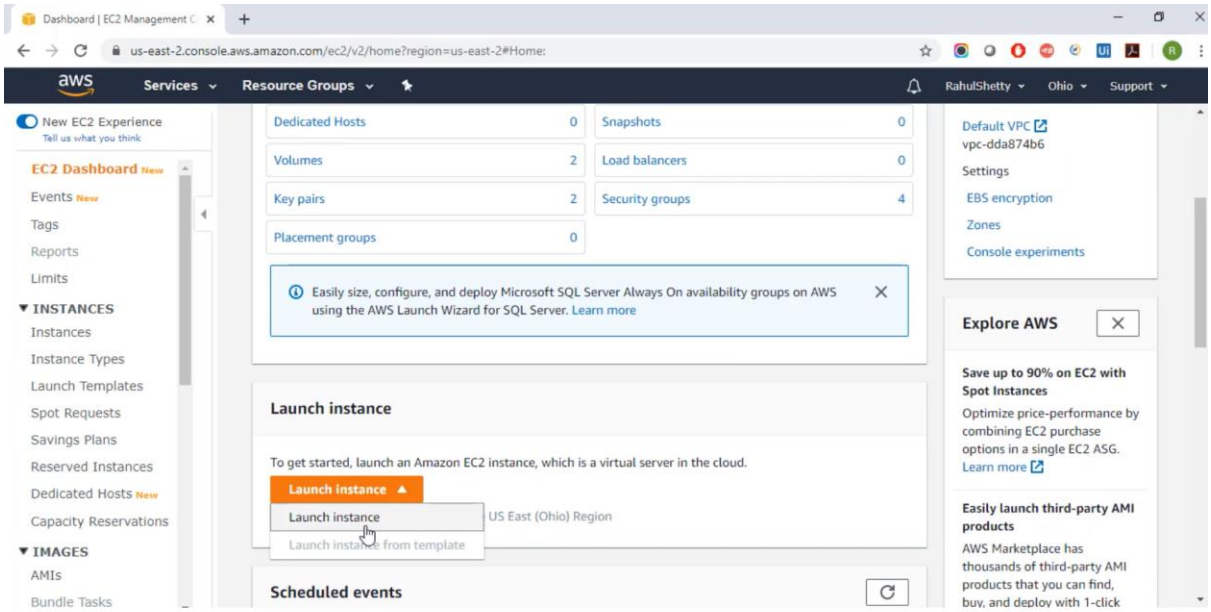

payLoad.py
configurations.py
dbDemo.py
postAPIExample.py
BackEndAutomation
utilities
__init__.py
configurations.py
properties.ini
resources.py
venv library root
apiValidations.py
dbDemo.py
jsonParsing.py
payLoad.py
postAPIExample.py
External Libraries
Scratches and Consoles
def buildPayLoadFromDB(query):
addBody = {}
tp = getQuery(query)
addBody['name'] = tp[0]
addBody['isbn'] = tp[1]
addBody['aisle'] = tp[2]
addBody['author'] = tp[3]
return addBody

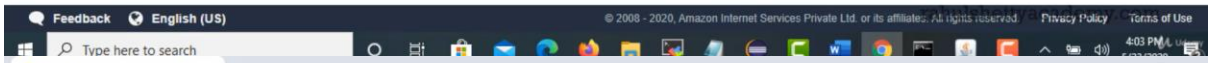

payLoad.py
configurations.py
dbDemo.py
postAPIExample.py
BackEndAutomation
utilities
__init__.py
configurations.py
properties.ini
resources.py
venv library root
apiValidations.py
dbDemo.py
jsonParsing.py
payLoad.py
postAPIExample.py
External Libraries
Scratches and Consoles
return conn
except Error as e:
print(e)
def getQuery(query):
conn = getConnection()
cursor = conn.cursor()
cursor.execute(query)
row = cursor.fetchone()
conn.close()
return row

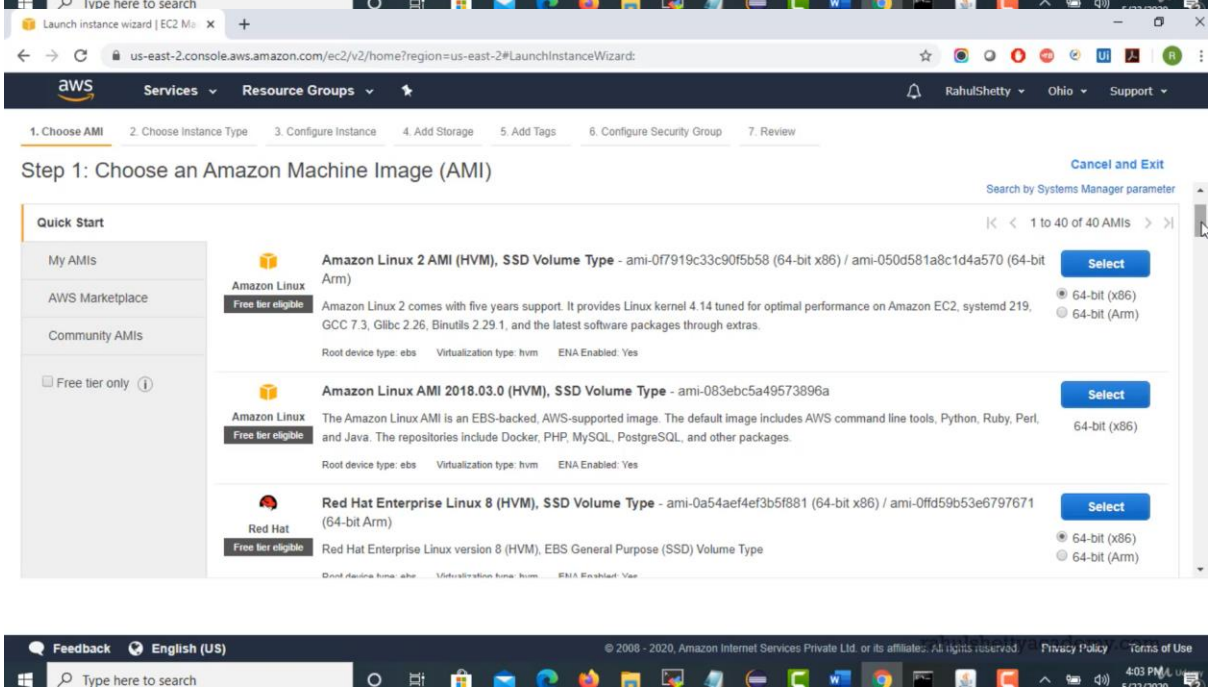
Python SSH Connection to Linux:

Launching AWS Linux Instance
Update Configurations to accept SSH Connection
Build Paramiko SSH Connection
Run the commands from local Python Script on Linux via SSH
How to read the output from Linux machine
How to transfer files from local server to Linux Server
Running Python batch files in Linux server via SSH
Downloading the files from Linux to local

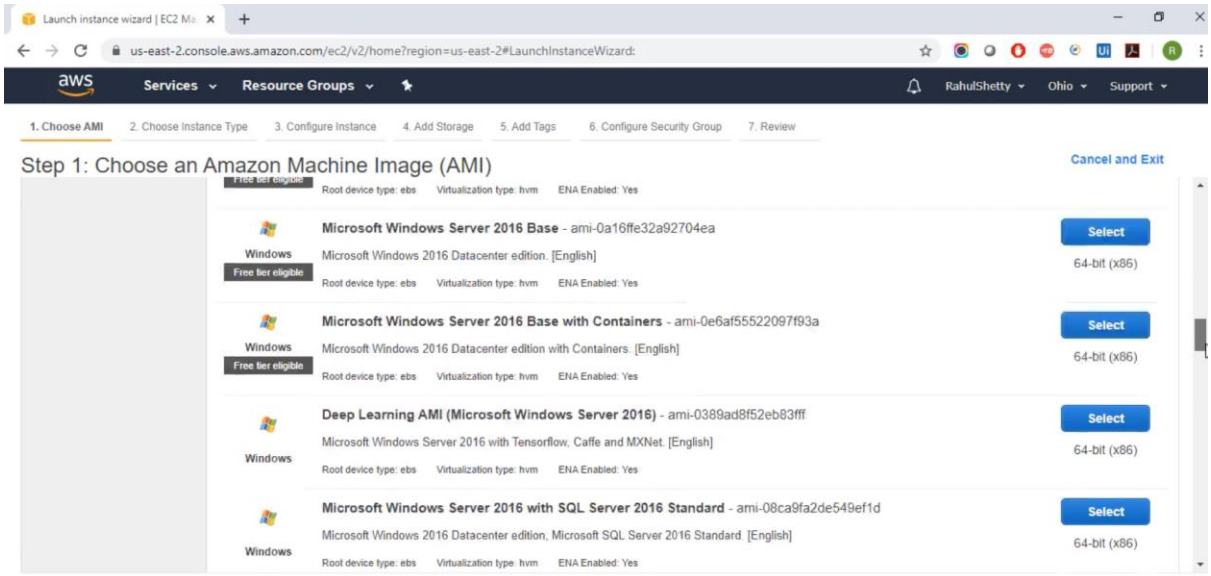


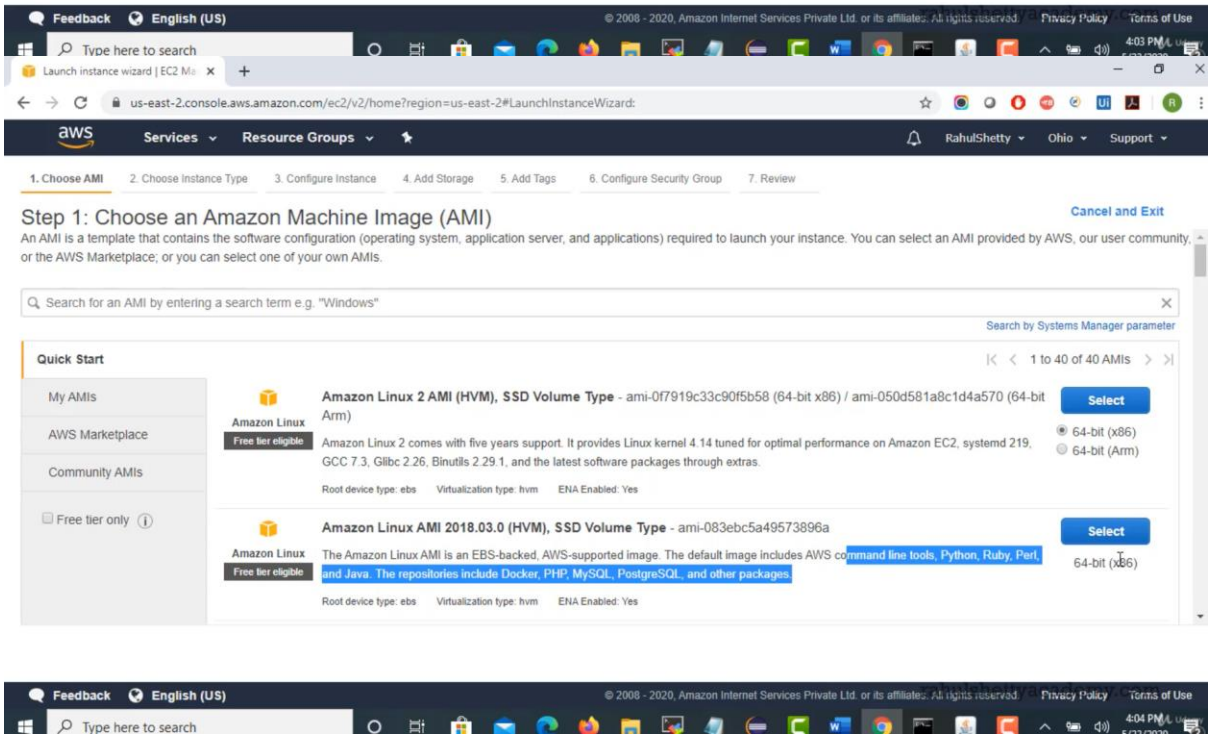

Dashboard | EC2 Management C
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Home:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
New EC2 Experience
Tell us what you think
EC2 Dashboard New
Events New
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts New
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
Dedicated Hosts 0
Snapshots 0
Volumes 2
Load balancers 0
Key pairs 2
Security groups 4
Placement groups 0
Easily size, configure, and deploy Microsoft SQL Server Always On availability groups on AWS using the AWS Launch Wizard for SQL Server. Learn more
Launch instance
To get started, launch an Amazon EC2 instance, which is a virtual server in the cloud.
Launch instance
Launch instance
Launch instance from template
US East (Ohio) Region
Scheduled events
Default VPC
vpc-dda874b6
Settings
EBS encryption
Zones
Console experiments
Explore AWS
Save up to 90% on EC2 with Spot Instances
Optimize price-performance by combining EC2 purchase options in a single EC2 ASG.
Learn more
Easily launch third-party AMI products
AWS Marketplace has thousands of third-party AMI products that you can find, buy, and deploy with 1-click


Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
4:03 PM

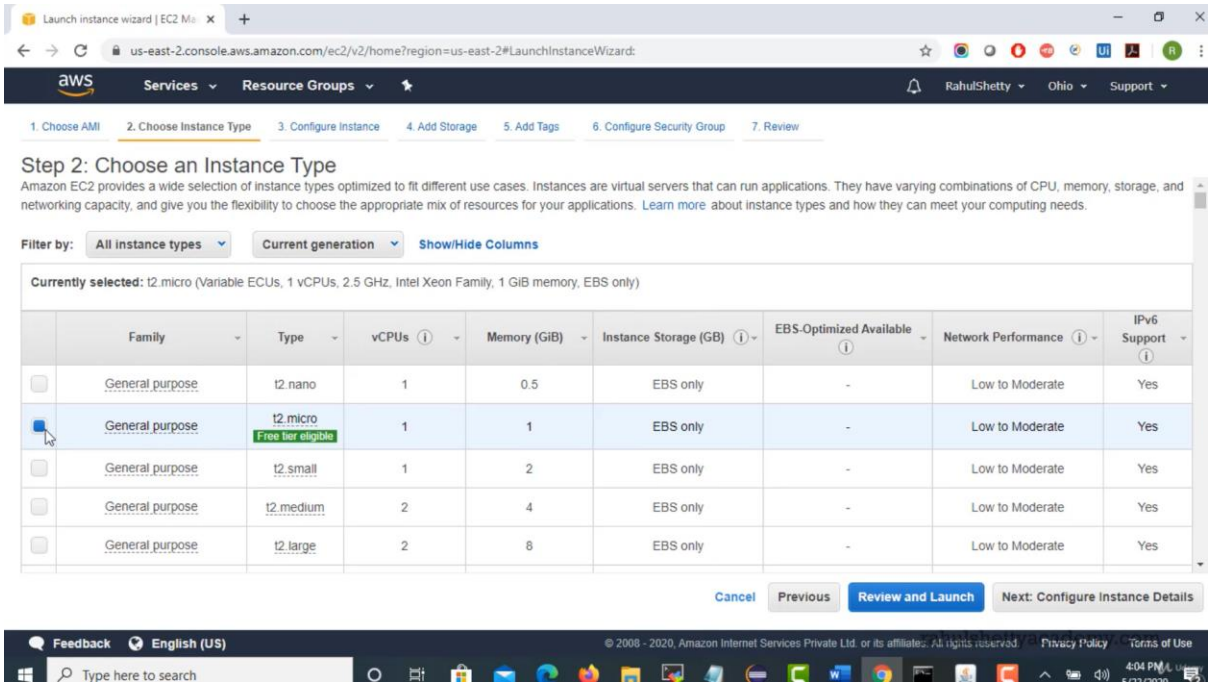

Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
Search by Systems Manager parameter
Quick Start
1 to 40 of 40 AMIs
My AMIs
AWS Marketplace
Community AMIs
Free tier only
Amazon Linux
Free tier eligible
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0f7919c33c90f5b58 (64-bit x86) / ami-050d581a8c1d4a570 (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.
Root device type: ebs
Virtualization type: hvm
ENA Enabled: Yes
Select
64-bit (x86)
64-bit (Arm)
Amazon Linux
Free tier eligible
Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-083ebc5a49573896a
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs
Virtualization type: hvm
ENA Enabled: Yes
Select
64-bit (x86)
Red Hat
Free tier eligible
Red Hat Enterprise Linux 8 (HVM), SSD Volume Type - ami-0a54aef4ef3b5f881 (64-bit x86) / ami-0ffd59b53e6797671 (64-bit Arm)
Red Hat Enterprise Linux version 8 (HVM), EBS General Purpose (SSD) Volume Type
Select
64-bit (x86)
64-bit (Arm)

Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
4:03 PM

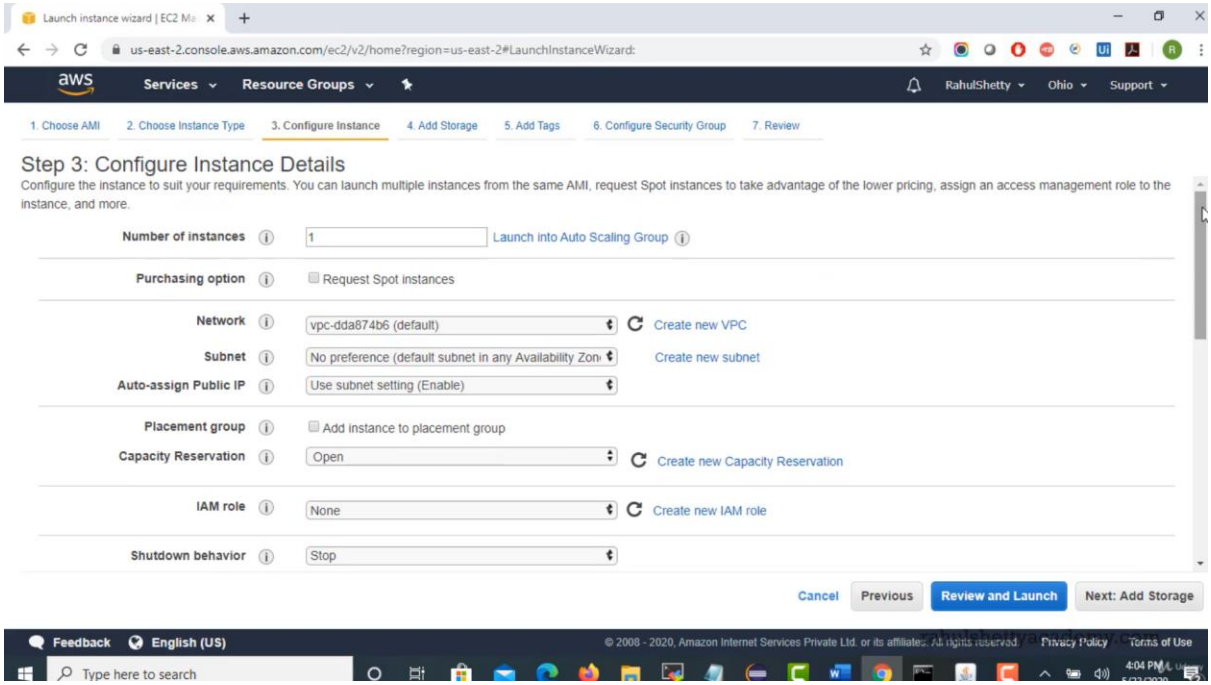

Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Windows
Free tier eligible
Microsoft Windows Server 2016 Base - ami-0a16ffe32a92704ea
Microsoft Windows 2016 Datacenter edition. [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)
Microsoft Windows Server 2016 Base with Containers - ami-0e6af55522097f93a
Microsoft Windows 2016 Datacenter edition with Containers. [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)
Deep Learning AMI (Microsoft Windows Server 2016) - ami-0389ad8f52eb83fff
Microsoft Windows Server 2016 with Tensorflow, Caffe and MXNet. [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)
Microsoft Windows Server 2016 with SQL Server 2016 Standard - ami-08ca9fa2de549ef1d
Microsoft Windows 2016 Datacenter edition, Microsoft SQL Server 2016 Standard. [English]
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)

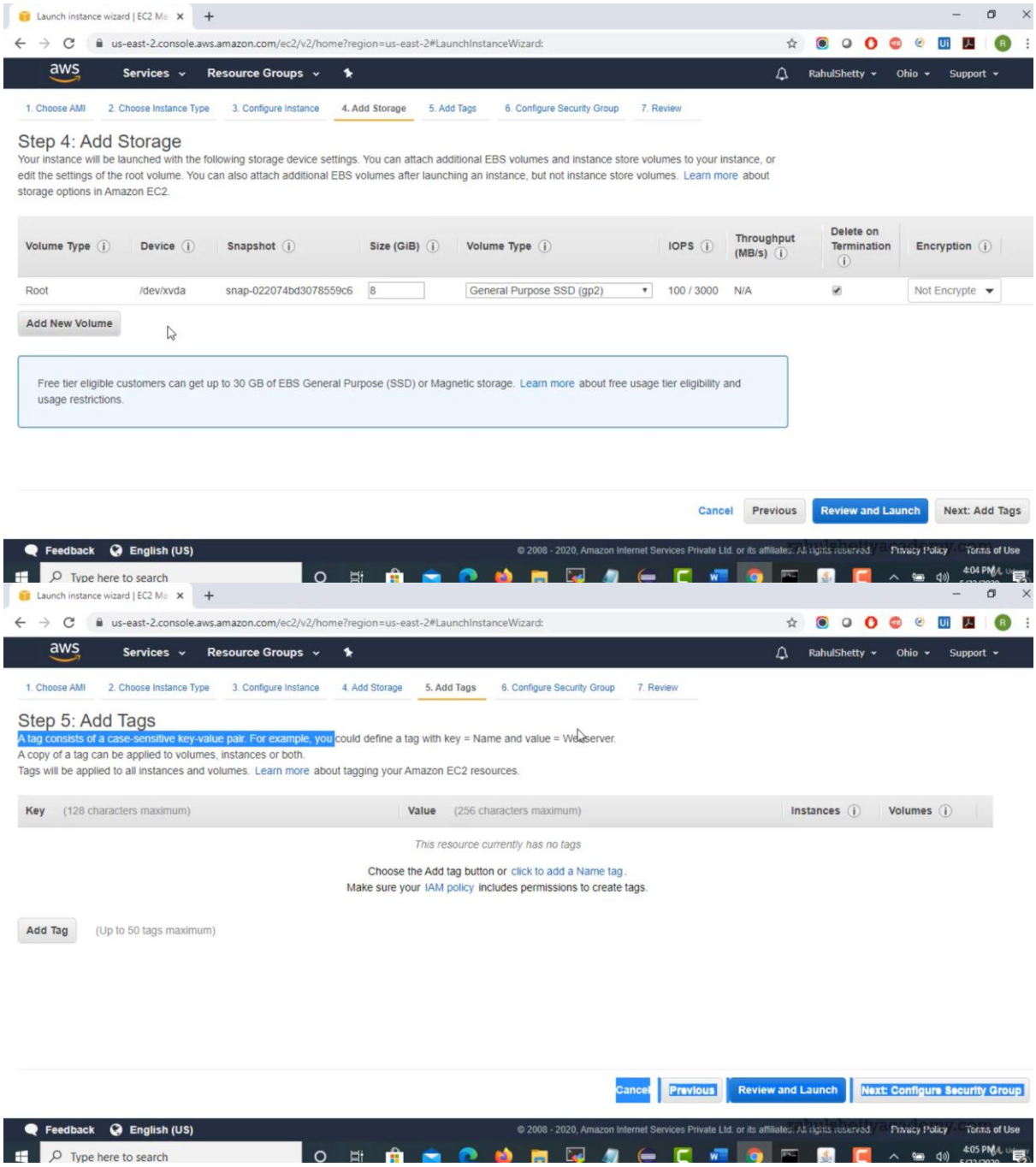

Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
4:03 PM
Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 1: Choose an Amazon Machine Image (AMI)
Cancel and Exit
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.
Search for an AMI by entering a search term e.g. "Windows"
Search by Systems Manager parameter
Quick Start
1 to 40 of 40 AMIs
My AMIs
AWS Marketplace
Community AMIs
Free tier only
Amazon Linux
Free tier eligible
Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-0f7919c33c90f5b58 (64-bit x86) / ami-050d581a8c1d4a570 (64-bit Arm)
Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)
64-bit (Arm)
Amazon Linux
Free tier eligible
Amazon Linux AMI 2018.03.0 (HVM), SSD Volume Type - ami-083ebc5a49573896a
The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.
Root device type: ebs Virtualization type: hvm ENA Enabled: Yes
Select
64-bit (x86)

Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
4:04 PM

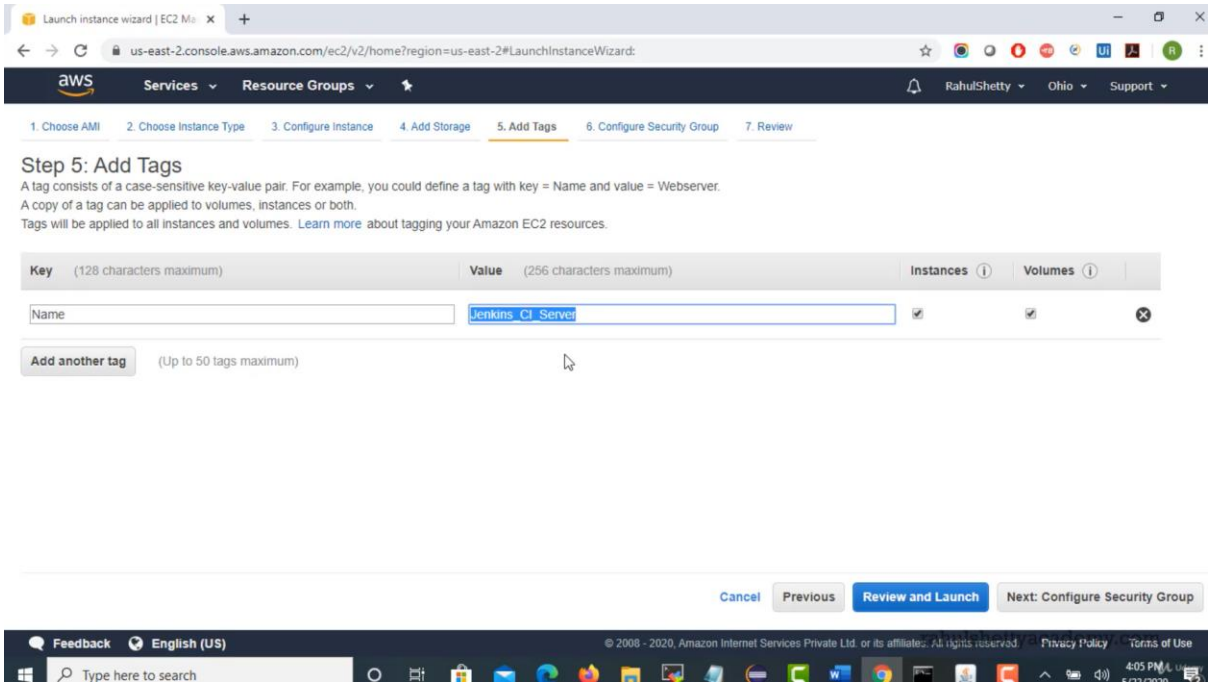


Below are network related settings

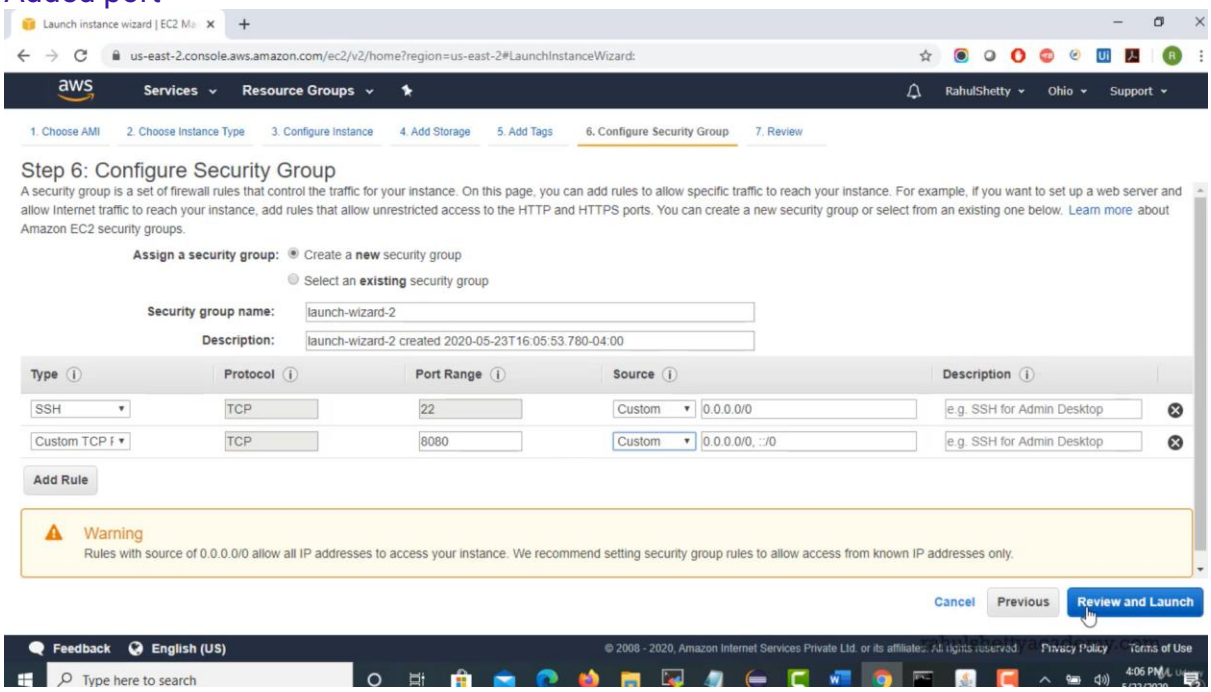


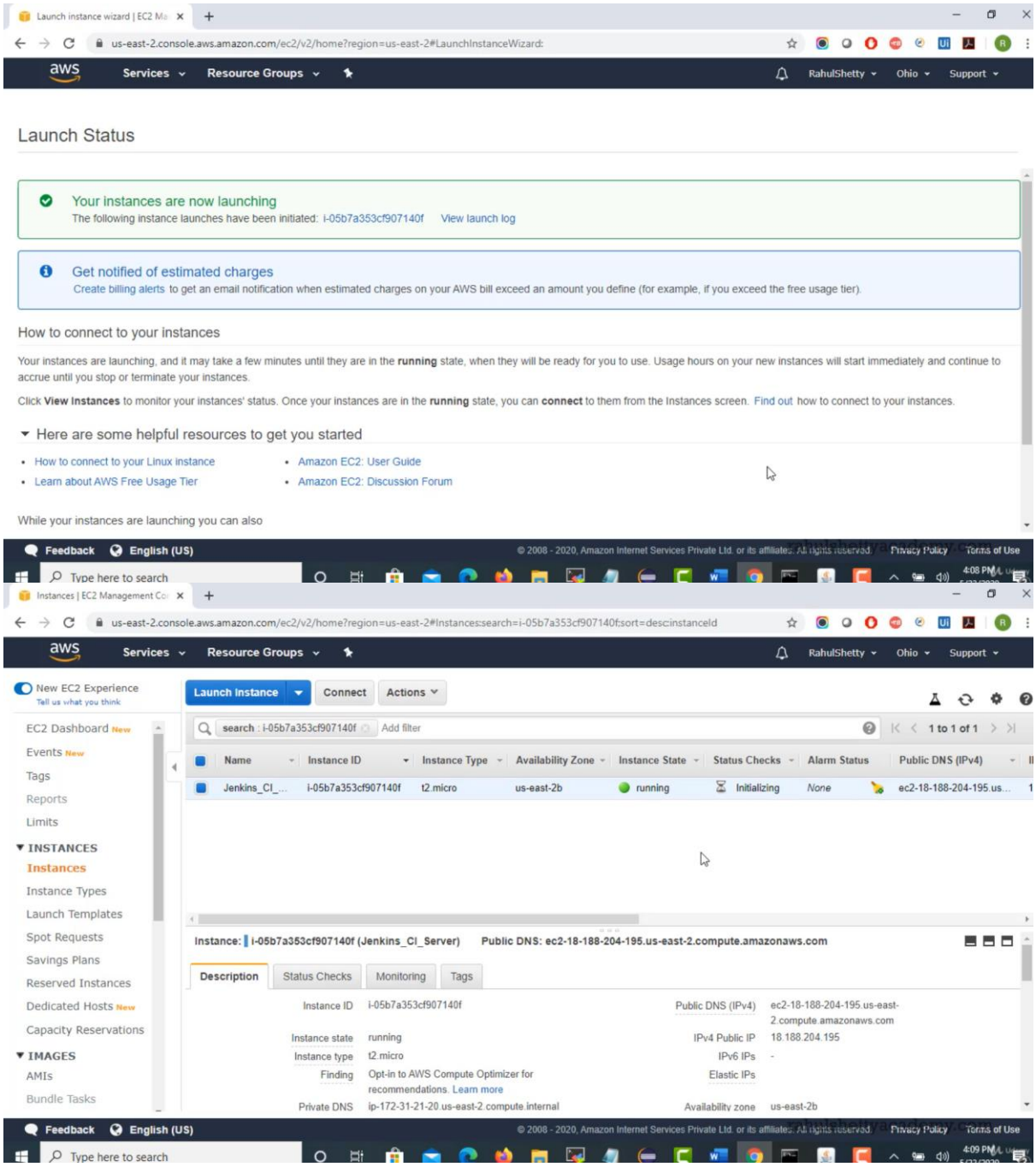


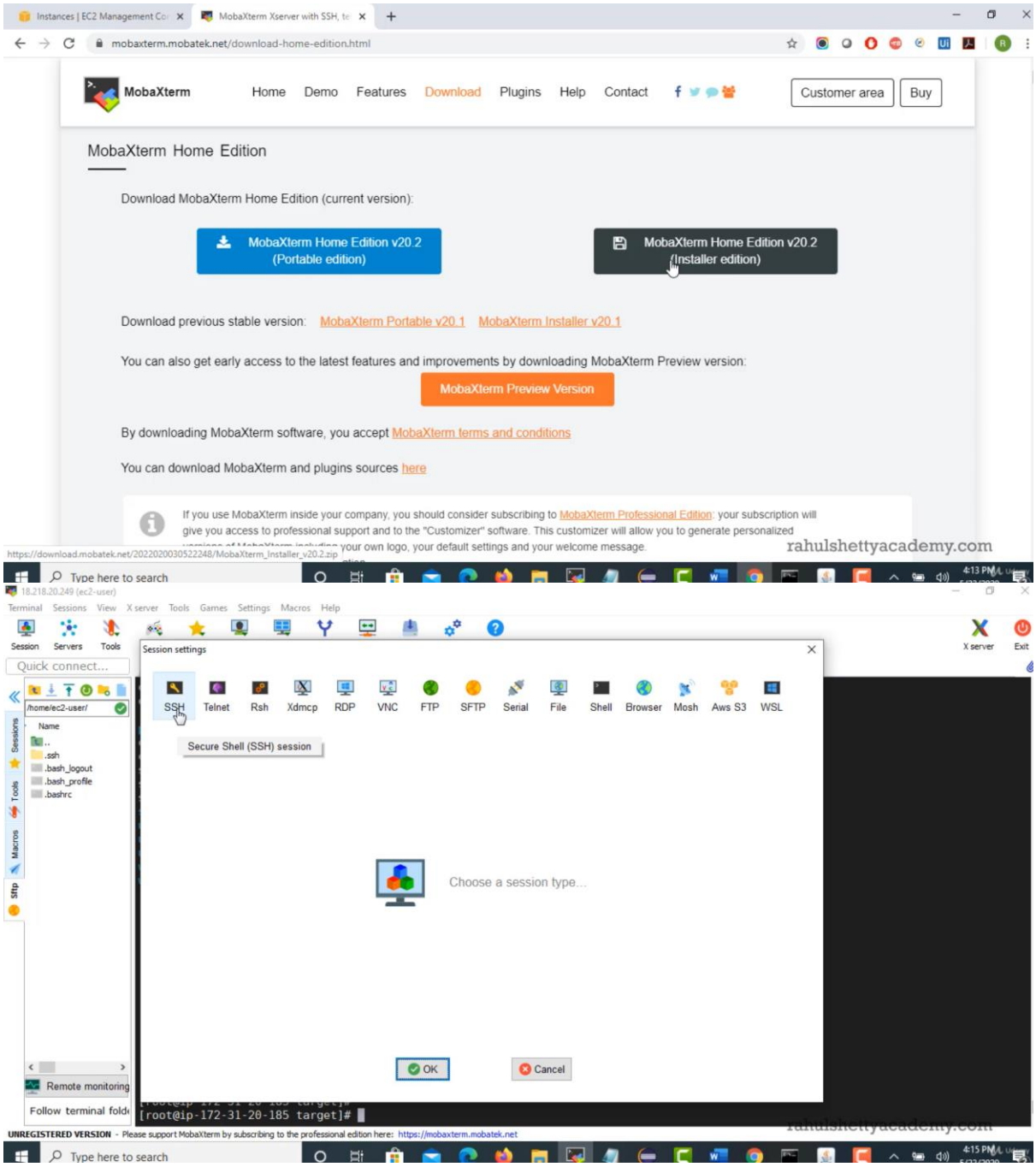
To identify server name


Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 5: Add Tags
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.
A copy of a tag can be applied to volumes, instances or both.
Tags will be applied to all instances and volumes. Learn more about tagging your Amazon EC2 resources.
Key (128 characters maximum)
Value (256 characters maximum)
Instances
Volumes
Name
Jenkins_CI_Server
Add another tag
(Up to 50 tags maximum)
Cancel
Previous
Review and Launch
Next: Configure Security Group
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search

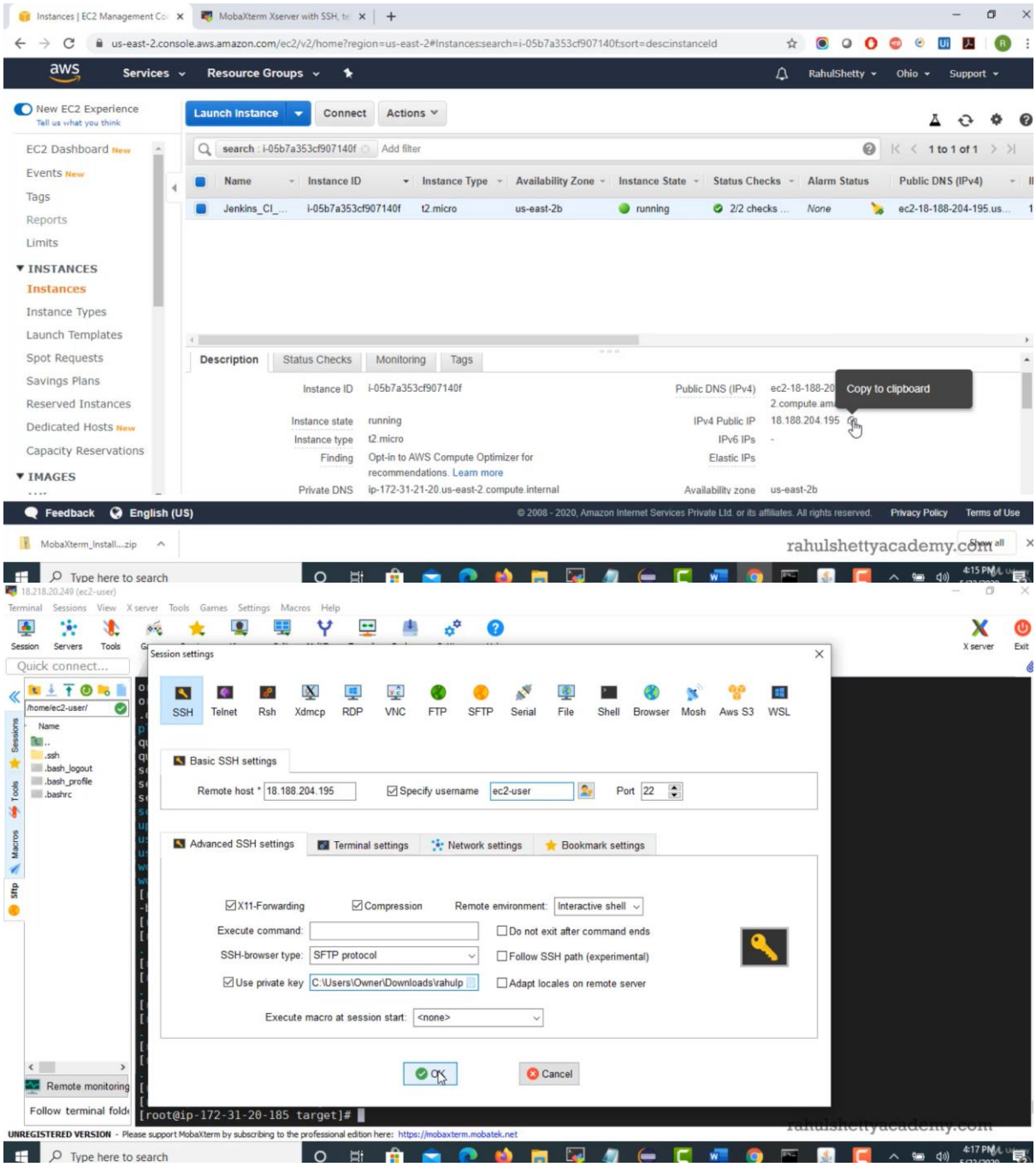
Added port

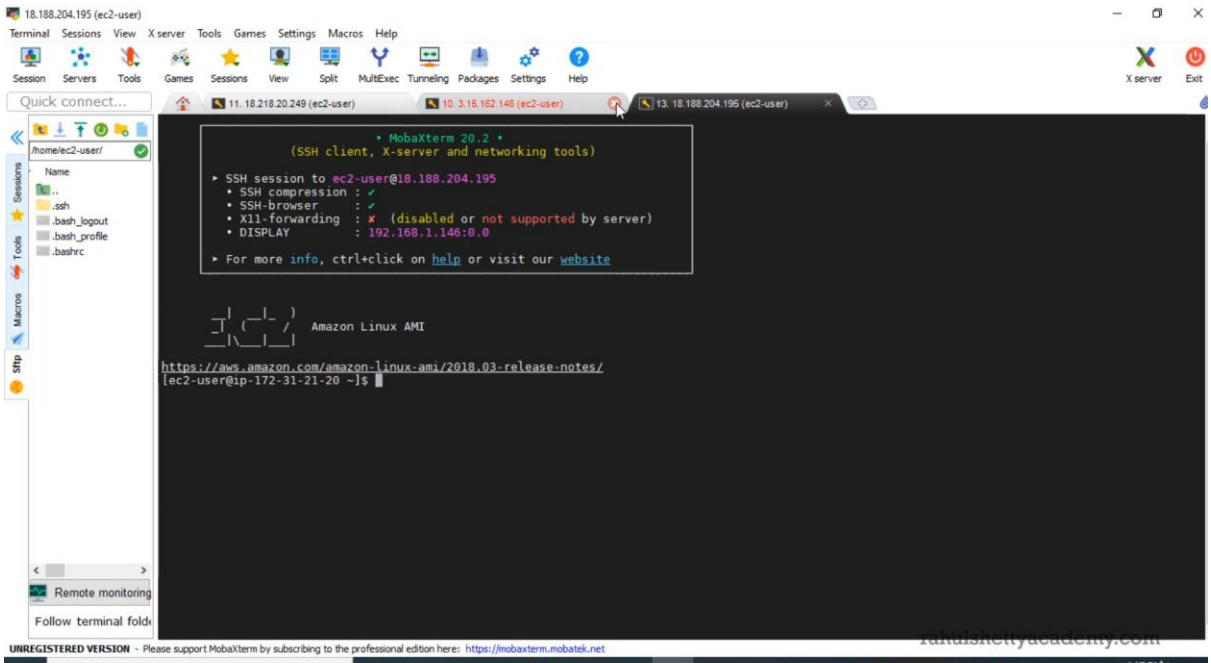

Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
1. Choose AMI
2. Choose Instance Type
3. Configure Instance
4. Add Storage
5. Add Tags
6. Configure Security Group
7. Review
Step 6: Configure Security Group
A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.
Assign a security group:
Create a new security group
Select an existing security group
Security group name:
launch-wizard-2
Description:
launch-wizard-2 created 2020-05-23T16:05:53.780-04:00
Type
Protocol
Port Range
Source
Description
SSH
TCP
22
Custom
0.0.0.0/0
e.g. SSH for Admin Desktop
Custom TCP F
TCP
8080
Custom
0.0.0.0/0, ::/0
e.g. SSH for Admin Desktop
Add Rule
Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.
Cancel
Previous
Review and Launch
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search

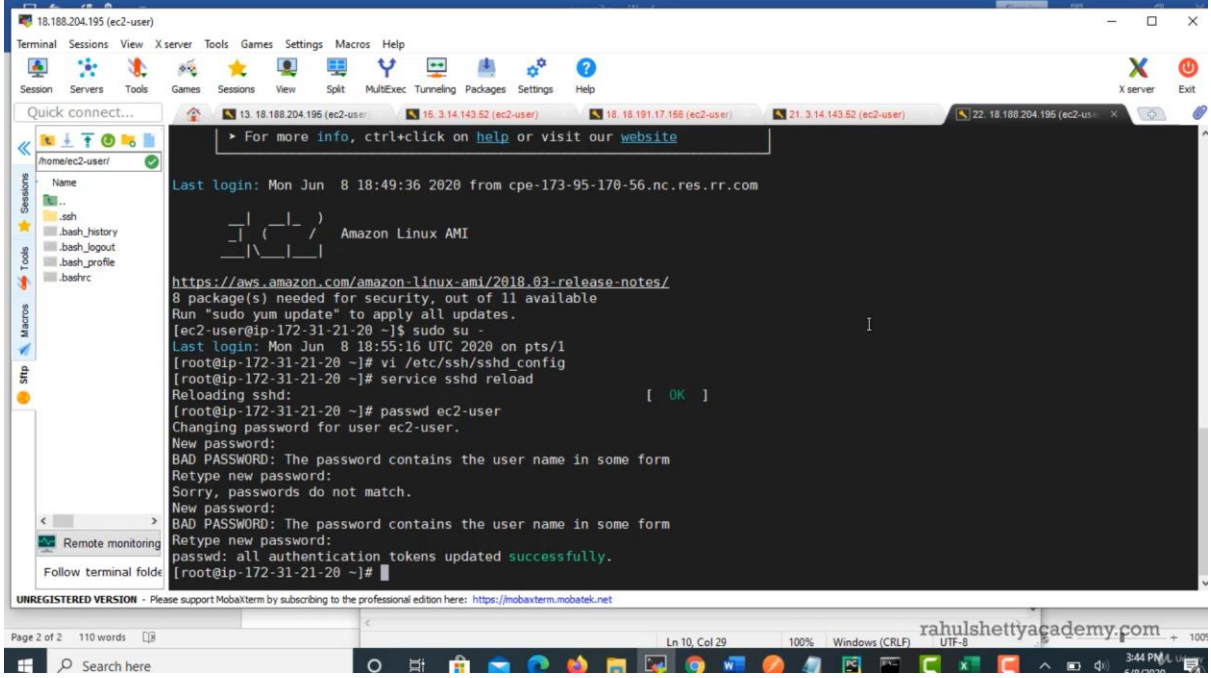

Launch instance wizard | EC2 Ma
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#LaunchInstanceWizard:
aws
Services
Resource Groups
RahulShetty
Ohio
Support
Launch Status
Your instances are now launching
The following instance launches have been initiated: i-05b7a353cf907140f
View launch log
Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).
How to connect to your instances
Your instances are launching, and it may take a few minutes until they are in the running state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.
Click View Instances to monitor your instances' status. Once your instances are in the running state, you can connect to them from the Instances screen. Find out how to connect to your instances.
Here are some helpful resources to get you started
How to connect to your Linux instance
Learn about AWS Free Usage Tier
Amazon EC2: User Guide
Amazon EC2: Discussion Forum
While your instances are launching you can also
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search
Instances | EC2 Management Co
us-east-2.console.aws.amazon.com/ec2/v2/home?region=us-east-2#Instances:search=i-05b7a353cf907140f;sort=desc:instanceId
New EC2 Experience
Tell us what you think
Launch Instance
Connect
Actions
EC2 Dashboard
Events
Tags
Reports
Limits
INSTANCES
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
IMAGES
AMIs
Bundle Tasks
search : i-05b7a353cf907140f
Add filter
1 to 1 of 1
Name
Instance ID
Instance Type
Availability Zone
Instance State
Status Checks
Alarm Status
Public DNS (IPv4)
Jenkins_CI_...
i-05b7a353cf907140f
t2.micro
us-east-2b
running
Initializing
None
ec2-18-188-204-195.us...
Instance: i-05b7a353cf907140f (Jenkins_CI_Server)
Public DNS: ec2-18-188-204-195.us-east-2.compute.amazonaws.com
Description
Status Checks
Monitoring
Tags
Instance ID
i-05b7a353cf907140f
Instance state
running
Instance type
t2.micro
Finding
Opt-in to AWS Compute Optimizer for recommendations. Learn more
Private DNS
ip-172-31-21-20.us-east-2.compute.internal
Public DNS (IPv4)
ec2-18-188-204-195.us-east-2.compute.amazonaws.com
IPv4 Public IP
18.188.204.195
IPv6 IPs
-
Elastic IPs
Availability zone
us-east-2b
Feedback
English (US)
© 2008 - 2020, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Type here to search

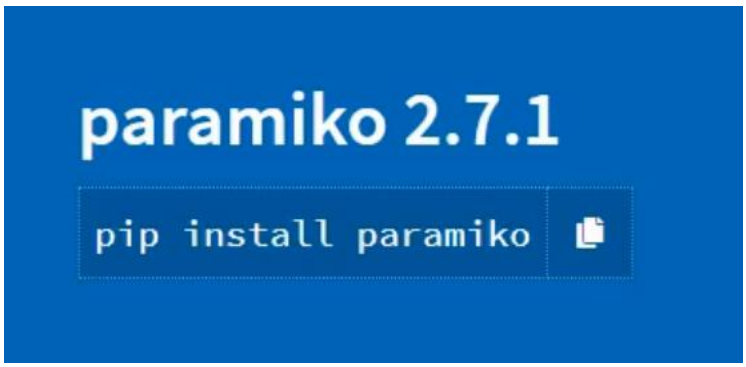

mobaxterm.mobatek.net/download-home-edition.html
MobaXterm
Home
Demo
Features
Download
Plugins
Help
Contact
Customer area
Buy
MobaXterm Home Edition
Download MobaXterm Home Edition (current version):
MobaXterm Home Edition v20.2 (Portable edition)
MobaXterm Home Edition v20.2 (Installer edition)
Download previous stable version: MobaXterm Portable v20.1 MobaXterm Installer v20.1
You can also get early access to the latest features and improvements by downloading MobaXterm Preview version:
MobaXterm Preview Version
By downloading MobaXterm software, you accept MobaXterm terms and conditions
You can download MobaXterm and plugins sources here
If you use MobaXterm inside your company, you should consider subscribing to MobaXterm Professional Edition: your subscription will give you access to professional support and to the "Customizer" software. This customizer will allow you to generate personalized
your own logo, your default settings and your welcome message.
https://download.mobatek.net/2022020030522248/MobaXterm_Installer_v20.2.zip
rahulshettyacademy.com

Session settings
SSH
Telnet
Rsh
Xdmcp
RDP
VNC
FTP
SFTP
Serial
File
Shell
Browser
Mosh
Aws S3
WSL
Secure Shell (SSH) session
Choose a session type...
OK
Cancel
[root@ip-172-31-20-185 target]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

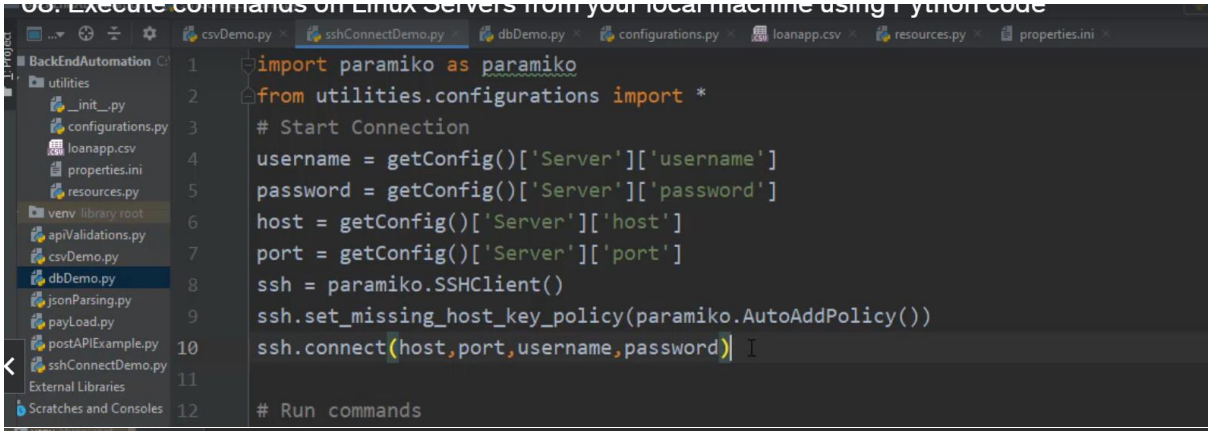

Launch instance
Connect
Actions
Description
Status Checks
Monitoring
Tags
Copy to clipboard
rahulshettyacademy.com


18.188.204.195 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
11. 18.218.20.249 (ec2-user)
10. 3.15.162.148 (ec2-user)
13. 18.188.204.195 (ec2-user)
/home/ec2-user/
Name
.ssh
.bash_logout
.bash_profile
.bashrc
Sessions Tools Macros Sftp
Remote monitoring
Follow terminal folde
• MobaXterm 20.2 •
(SSH client, X-server and networking tools)
▸ SSH session to ec2-user@18.188.204.195
• SSH compression : ✓
• SSH-browser : ✓
• X11-forwarding : ✗ (disabled or not supported by server)
• DISPLAY : 192.168.1.146:0.0
▸ For more info, ctrl+click on help or visit our website
Amazon Linux AMI
https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
[ec2-user@ip-172-31-21-20 ~]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
rahulshettyacademy.com
Type here to search
4:17 PM

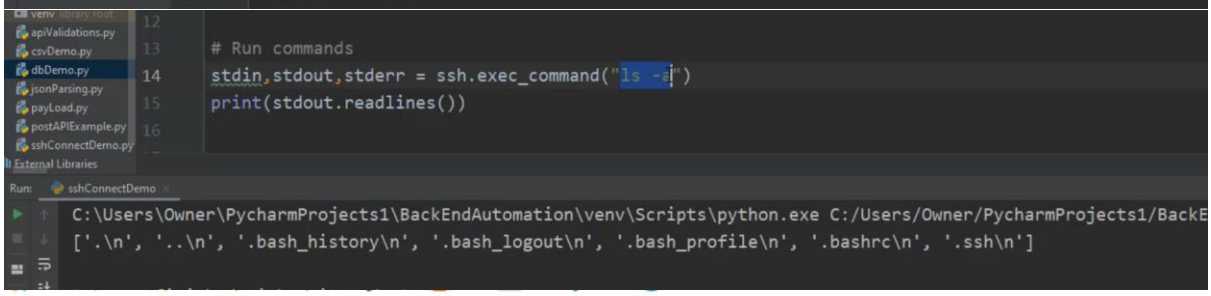

18.188.204.195 (ec2-user)
Terminal Sessions View X server Tools Games Settings Macros Help
Session Servers Tools Games Sessions View Split MultiExec Tunneling Packages Settings Help
X server Exit
Quick connect...
13. 18.188.204.195 (ec2-user)
15. 3.14.143.52 (ec2-user)
18. 18.191.17.158 (ec2-user)
21. 3.14.143.52 (ec2-user)
22. 18.188.204.195 (ec2-user)
/home/ec2-user/
Name
.ssh
.bash_history
.bash_logout
.bash_profile
.bashrc
Sessions Tools Macros Sftp
Remote monitoring
Follow terminal folde
▸ For more info, ctrl+click on help or visit our website
Last login: Mon Jun 8 18:49:36 2020 from cpe-173-95-170-56.nc.res.rr.com
Amazon Linux AMI
https://aws.amazon.com/amazon-linux-ami/2018.03-release-notes/
8 package(s) needed for security, out of 11 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-21-20 ~]$ sudo su -
Last login: Mon Jun 8 18:55:16 UTC 2020 on pts/1
[root@ip-172-31-21-20 ~]# vi /etc/ssh/sshd_config
[root@ip-172-31-21-20 ~]# service sshd reload
Reloading sshd: [OK]
[root@ip-172-31-21-20 ~]# passwd ec2-user
Changing password for user ec2-user.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
Sorry, passwords do not match.
New password:
BAD PASSWORD: The password contains the user name in some form
Retype new password:
passwd: all authentication tokens updated successfully.
[root@ip-172-31-21-20 ~]#
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net
Page 2 of 2 110 words
Ln 10, Col 29 100% Windows (CRLF) UTF-8
rahulshettyacademy.com
Search here
3:44 PM


paramiko 2.7.1
pip install paramiko

08. Execute commands on Linux Servers from your local machine using Python code

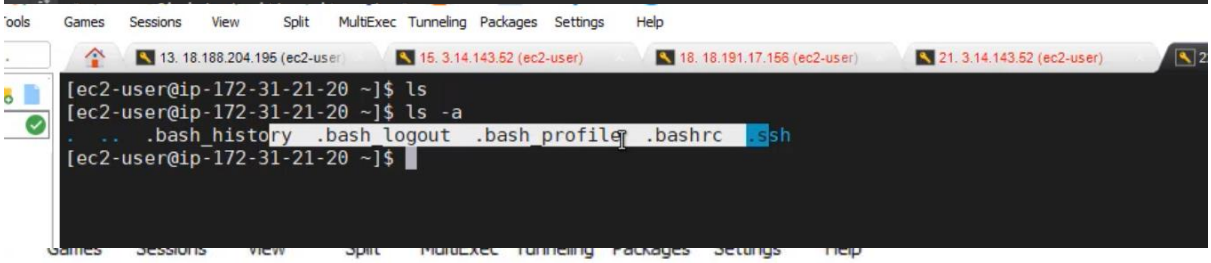


```
import paramiko as paramiko
from utilities.configurations import *
# Start Connection
username = getConfig()['Server']['username']
password = getConfig()['Server']['password']
host = getConfig()['Server']['host']
port = getConfig()['Server']['port']
ssh = paramiko.SSHClient()
ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh.connect(host,port,username,password)

# Run commands
```

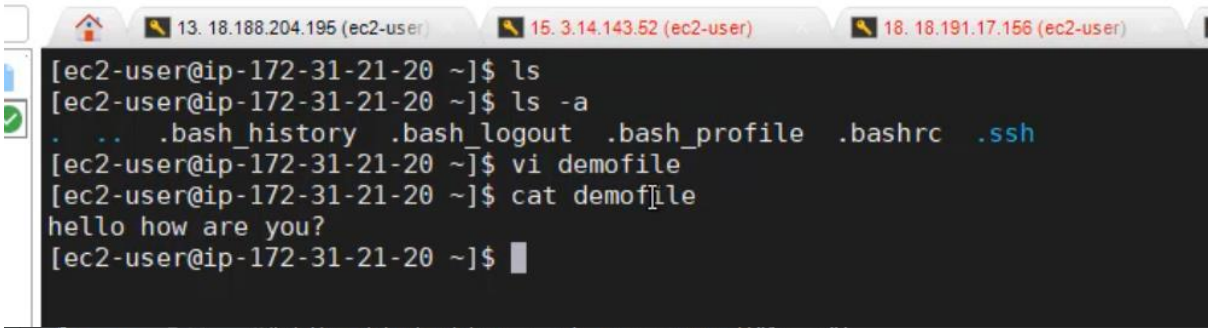


```
# Run commands
stdin,stdout,stderr = ssh.exec_command("ls -a")
print(stdout.readlines())
```

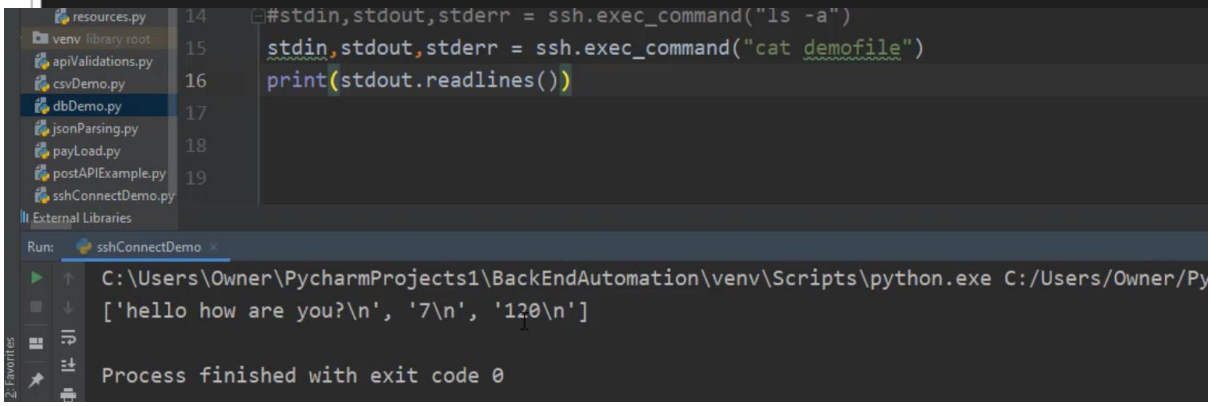
```
C:\Users\Owner\PycharmProjects1\BackEndAutomation\venv\Scripts\python.exe C:/Users/Owner/PycharmProjects1/BackE
['.\n', '..\n', '.bash_history\n', '.bash_logout\n', '.bash_profile\n', '.bashrc\n', '.ssh\n']
```



```
[ec2-user@ip-172-31-21-20 ~]$ ls
[ec2-user@ip-172-31-21-20 ~]$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .ssh
[ec2-user@ip-172-31-21-20 ~]$
```



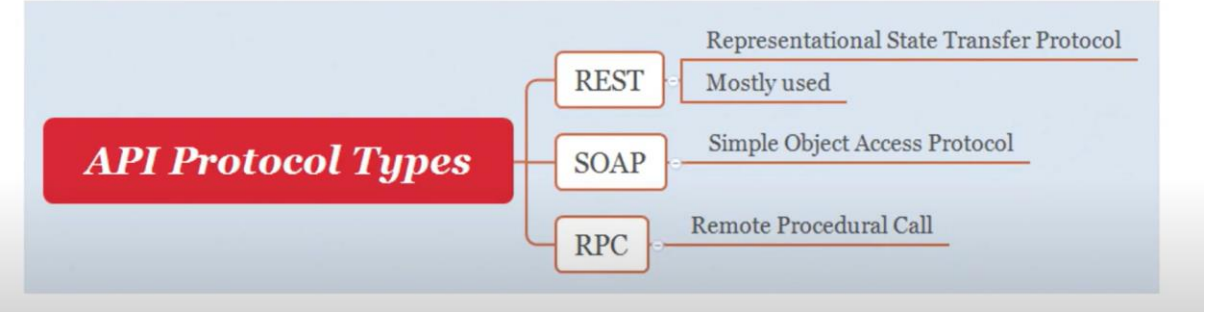
```
[ec2-user@ip-172-31-21-20 ~]$ ls
[ec2-user@ip-172-31-21-20 ~]$ ls -a
.  ..  .bash_history  .bash_logout  .bash_profile  .bashrc  .ssh
[ec2-user@ip-172-31-21-20 ~]$ vi demofile
[ec2-user@ip-172-31-21-20 ~]$ cat demofile
hello how are you?
[ec2-user@ip-172-31-21-20 ~]$
```

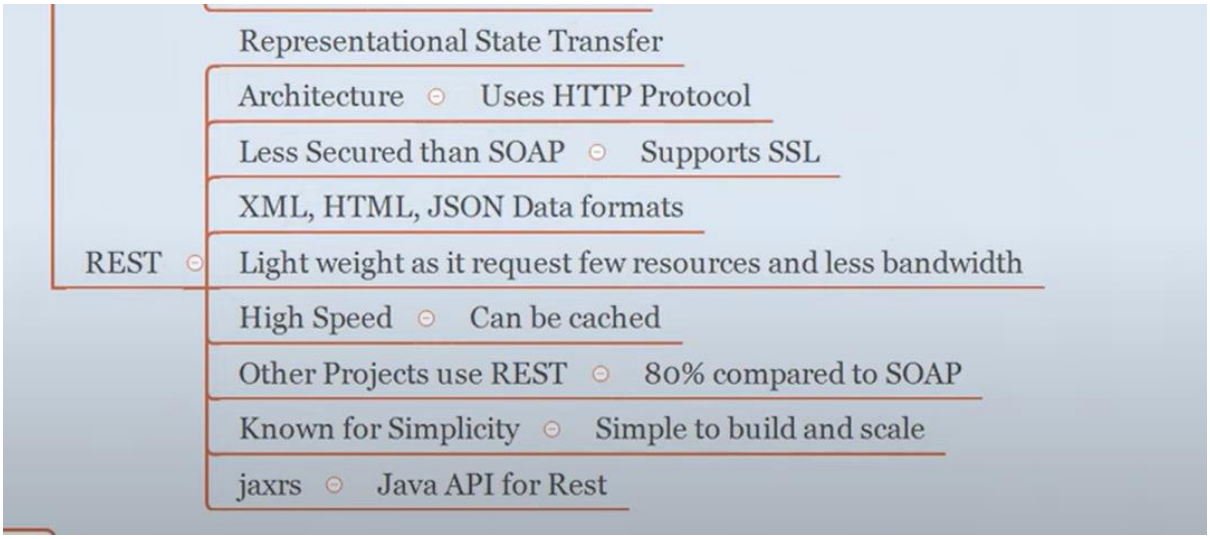


```
#stdin,stdout,stderr = ssh.exec_command("ls -a")
stdin,stdout,stderr = ssh.exec_command("cat demofile")
print(stdout.readlines())
```

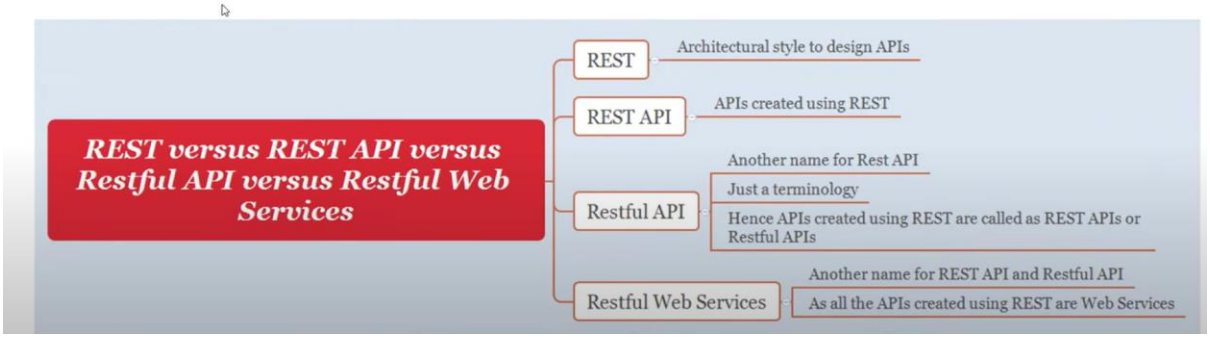
```
C:\Users\Owner\PycharmProjects1\BackEndAutomation\venv\Scripts\python.exe C:/Users/Owner/Py
['hello how are you?\n', '7\n', '120\n']

Process finished with exit code 0
```

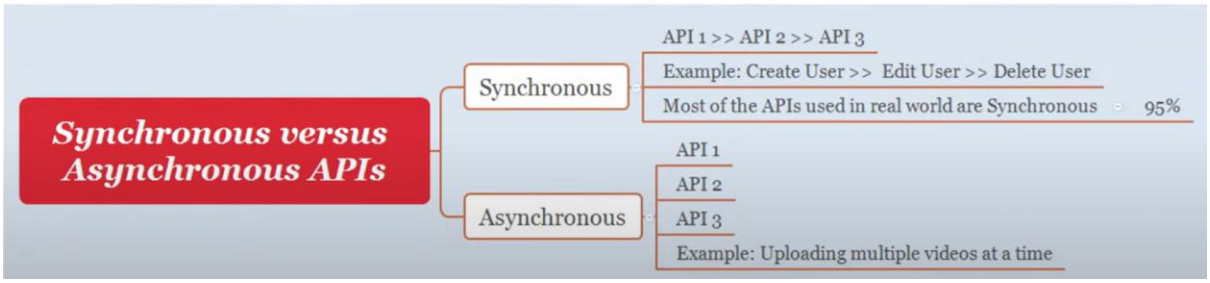
API Protocol Types (API Testing - Part 8)



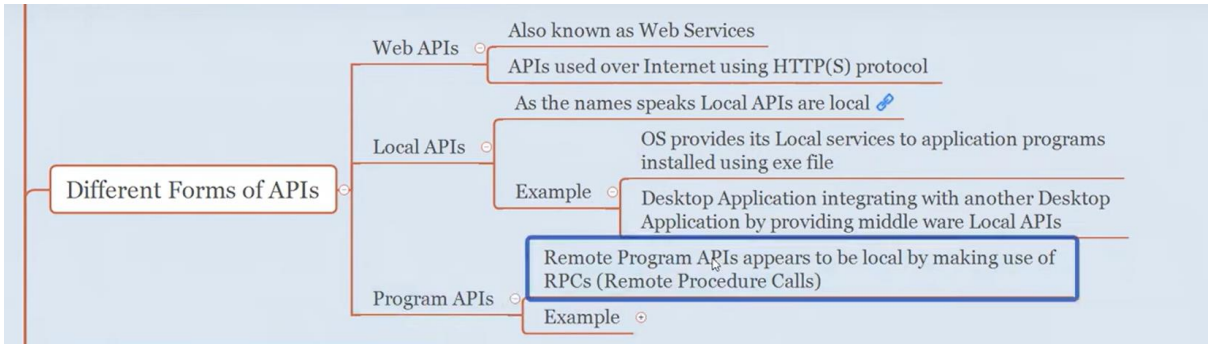


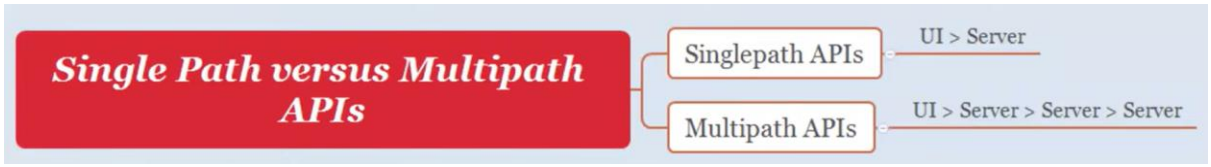
SOAP	REST
20%	80%
Simple Object Access Protocol	Representational State Transfer Protocol
Protocol	Architecture (HTTP(S))
More Secured - WS-Securitry SSL	Less Secured SSL
XML Format	XML, HTML AND JSON
Heavy Weight	Light Weight (Simple and Scable)
Low Speed Not cacheable	High Speed Can be cached

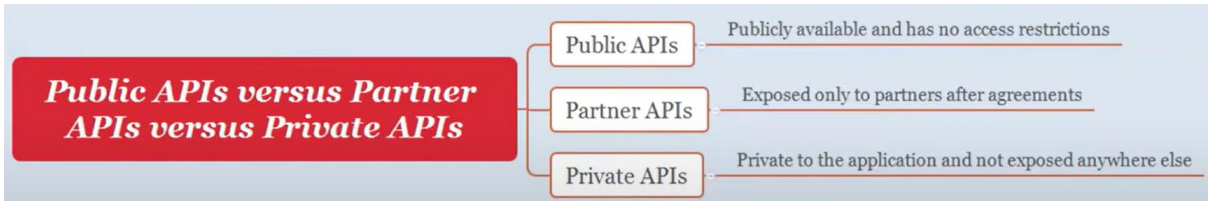


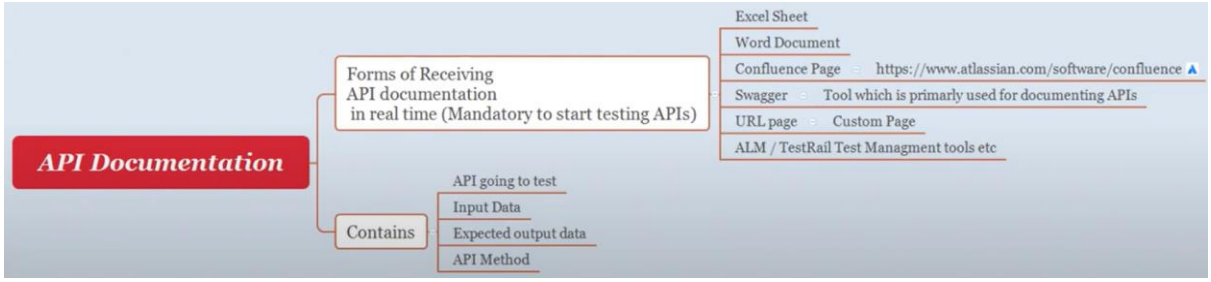


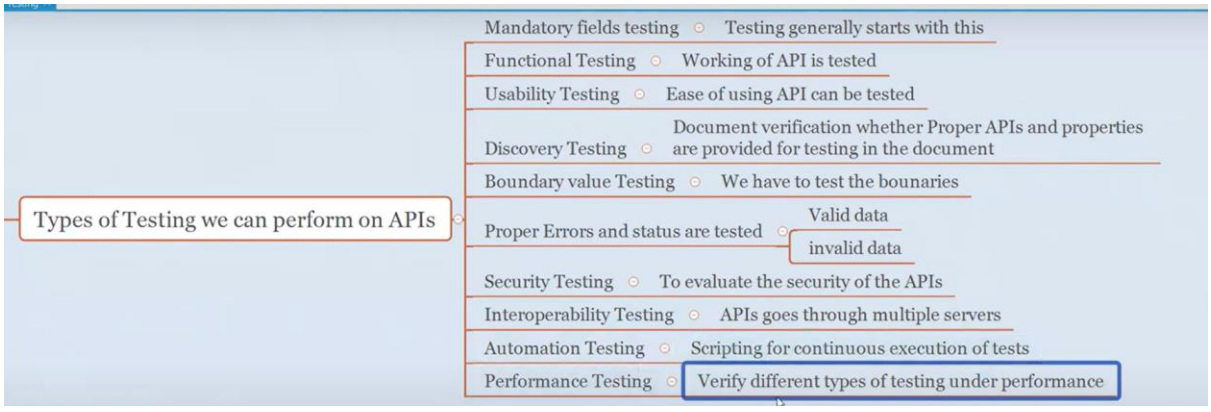
API3 is dependent on api2 and it depends on1 synchronous

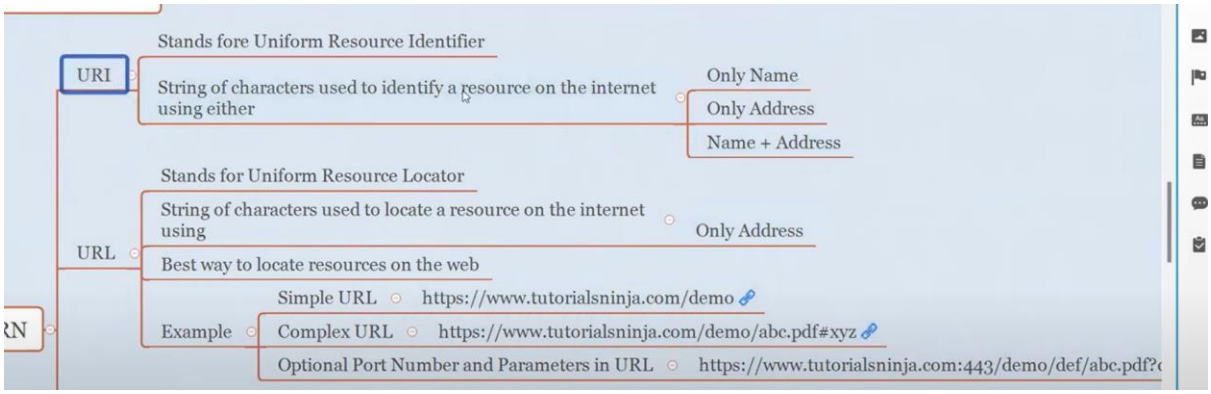

Different Forms of APIs
Web APIs
Also known as Web Services
APIs used over Internet using HTTP(S) protocol
Local APIs
As the names speaks Local APIs are local
Example
OS provides its Local services to application programs installed using exe file
Desktop Application integrating with another Desktop Application by providing middle ware Local APIs
Program APIs
Remote Program APIs appears to be local by making use of RPCs (Remote Procedure Calls)
Example

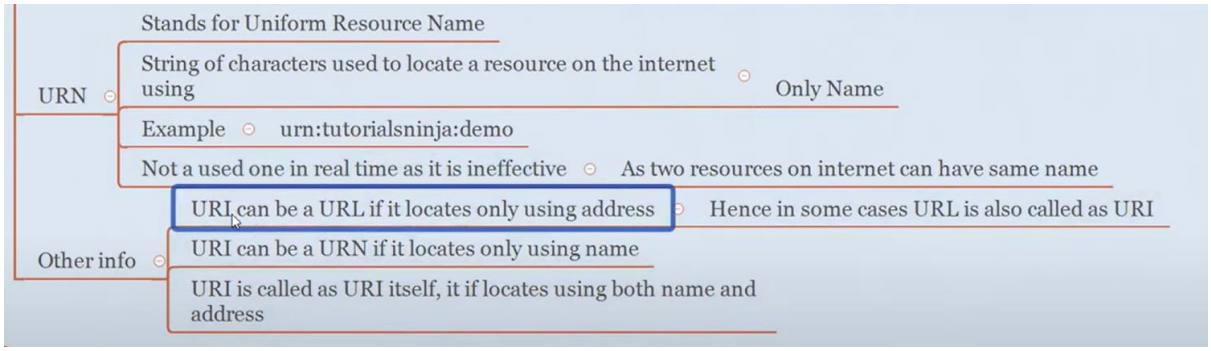

Single Path versus Multipath APIs
Singlepath APIs
UI > Server
Multipath APIs
UI > Server > Server > Server


Public APIs versus Partner APIs versus Private APIs
Public APIs
Publicly available and has no access restrictions
Partner APIs
Exposed only to partners after agreements
Private APIs
Private to the application and not exposed anywhere else


API Documentation
Forms of Receiving API documentation in real time (Mandatory to start testing APIs)
Excel Sheet
Word Document
Confluence Page
https://www.atlassian.com/software/confluence
Swagger
Tool which is primarly used for documenting APIs
URL page
Custom Page
ALM / TestRail Test Managment tools etc
Contains
API going to test
Input Data
Expected output data
API Method


Types of Testing we can perform on APIs
Mandatory fields testing
Testing generally starts with this
Functional Testing
Working of API is tested
Usability Testing
Ease of using API can be tested
Discovery Testing
Document verification whether Proper APIs and properties are provided for testing in the document
Boundary value Testing
We have to test the bounaries
Proper Errors and status are tested
Valid data
invalid data
Security Testing
To evaluate the security of the APIs
Interoperability Testing
APIs goes through multiple servers
Automation Testing
Scripting for continuous execution of tests
Performance Testing
Verify different types of testing under performance





https://www.youtube.com/watch?v=ahhGS3fyU00&list=PLsjUcU8CQXGFqvw5OIKEZ9UUehLOtZwZO&index=21