

# API Testing using Postman V1

## TABLE OF CONTENTS

**01** Postman Setup

---

**02** Create Workspace

---

**03** Create Collection

---

**04** Add Request

---

**05** Create Request,  
Analyse, Write Tests

---

**06** Create Environment  
Variables

---

**07** Export Collection

---

**08** Export Environment

---

**09** Import Collection

---

**10** Import Environment

---

**11** Run Collection

---

**12** Add Authorization

---

**13** Newman Guide

**14** Run from Jenkins

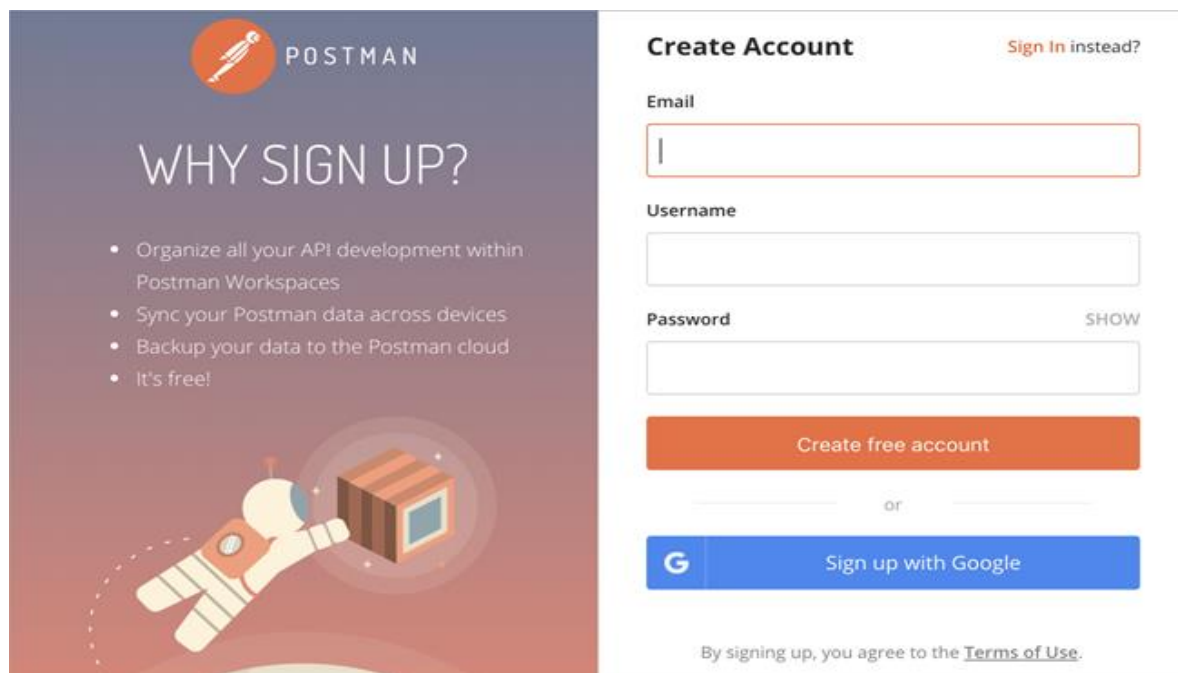
Bonus: FAQs

## 1. Postman Setup:

**Step 1** --- Download and Install postman (<https://www.postman.com/downloads/>)

**Step 2** --- Signup with required information.

**Step 2** --- Login to Postman.



The image shows the Postman 'Create Account' page. On the left, there is a dark blue section with the Postman logo and the heading 'WHY SIGN UP?'. Below this, a list of benefits is provided: 'Organize all your API development within Postman Workspaces', 'Sync your Postman data across devices', 'Backup your data to the Postman cloud', and 'It's free!'. An illustration of a rocket ship is at the bottom of this section. On the right, the 'Create Account' form is displayed. It includes fields for 'Email', 'Username', and 'Password'. A 'SHOW' link is next to the password field. Below the fields is an orange 'Create free account' button. Underneath this button is a horizontal line with 'or' in the center. Below that is a blue button with the Google 'G' logo and the text 'Sign up with Google'. At the bottom of the form, a small text line states: 'By signing up, you agree to the [Terms of Use](#)'.

**POSTMAN**

### WHY SIGN UP?

- Organize all your API development within Postman Workspaces
- Sync your Postman data across devices
- Backup your data to the Postman cloud
- It's free!

**Create Account** [Sign In instead?](#)

Email

Username

Password  [SHOW](#)

[Create free account](#)

or

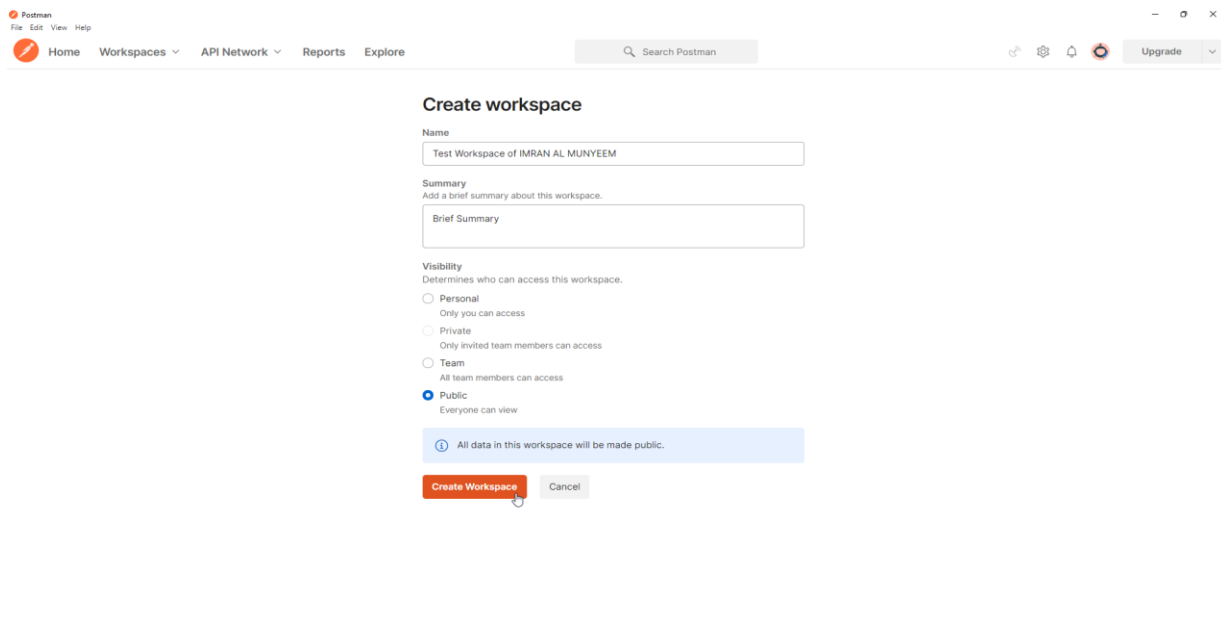
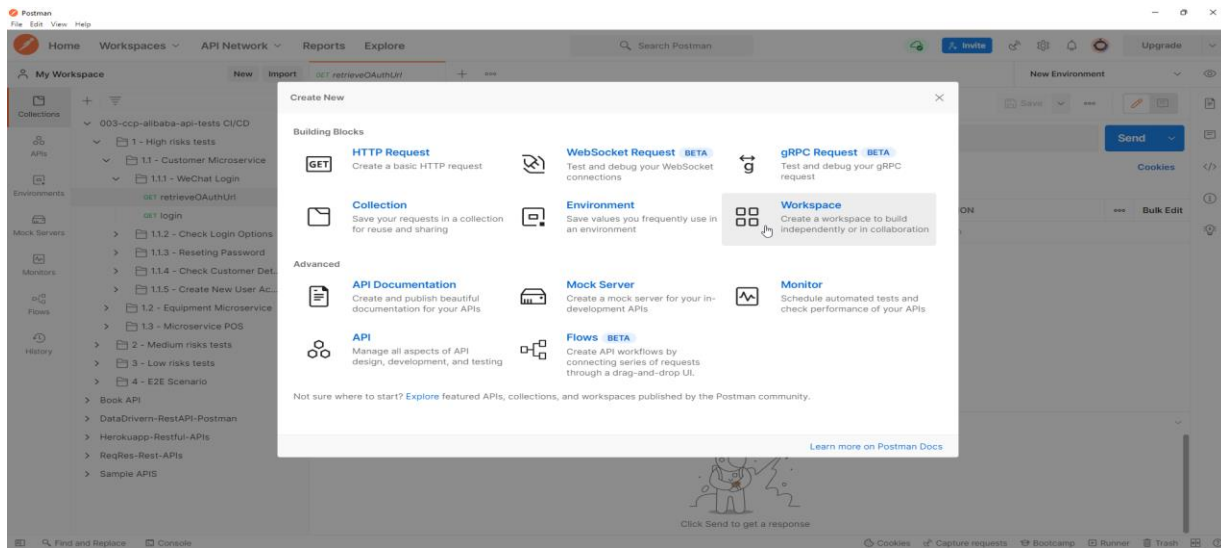
[Sign up with Google](#)

By signing up, you agree to the [Terms of Use](#).

## 2. Create Workspace:

**Step 1** --- From workspace menu, click on “New Workspace” and give workspace name.

**Step 2** --- Click on Create workspace button.



### 3. Create Collection:

**Step 1** --- Select workspace from list.

**Step 2** --- Click on the “Create Collection” button.

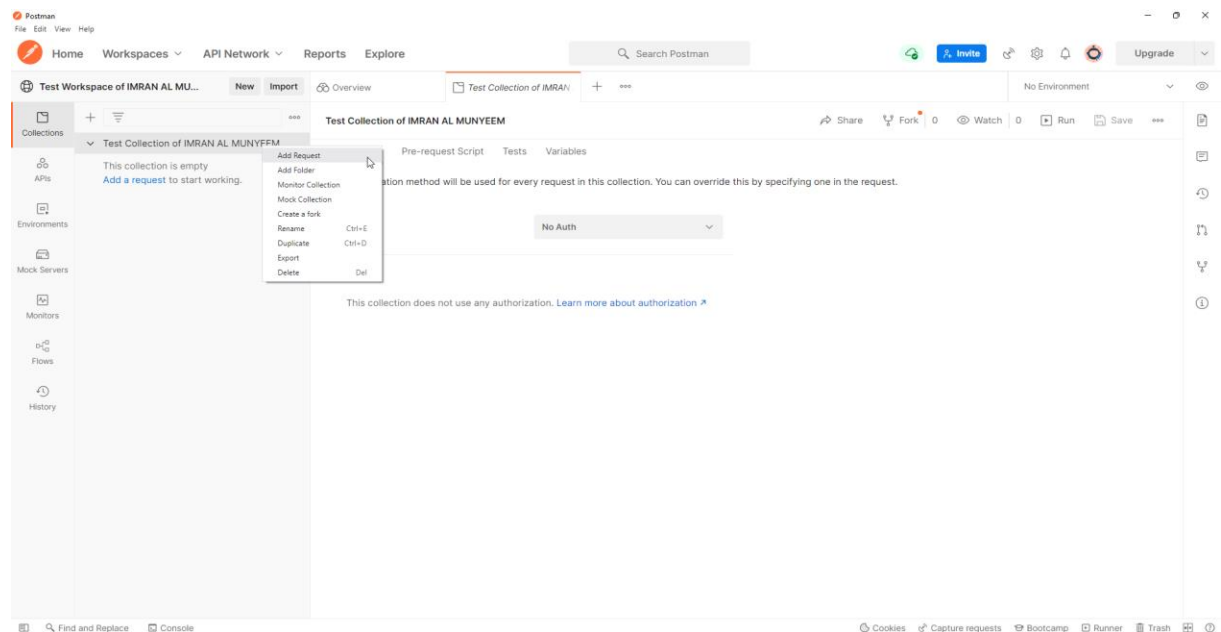
**Step 3** --- Give the collection name and press enter.



### 4. Add Request

**Step 1** --- Click on “Add a request” link or right click on the collection or Click on Add request.

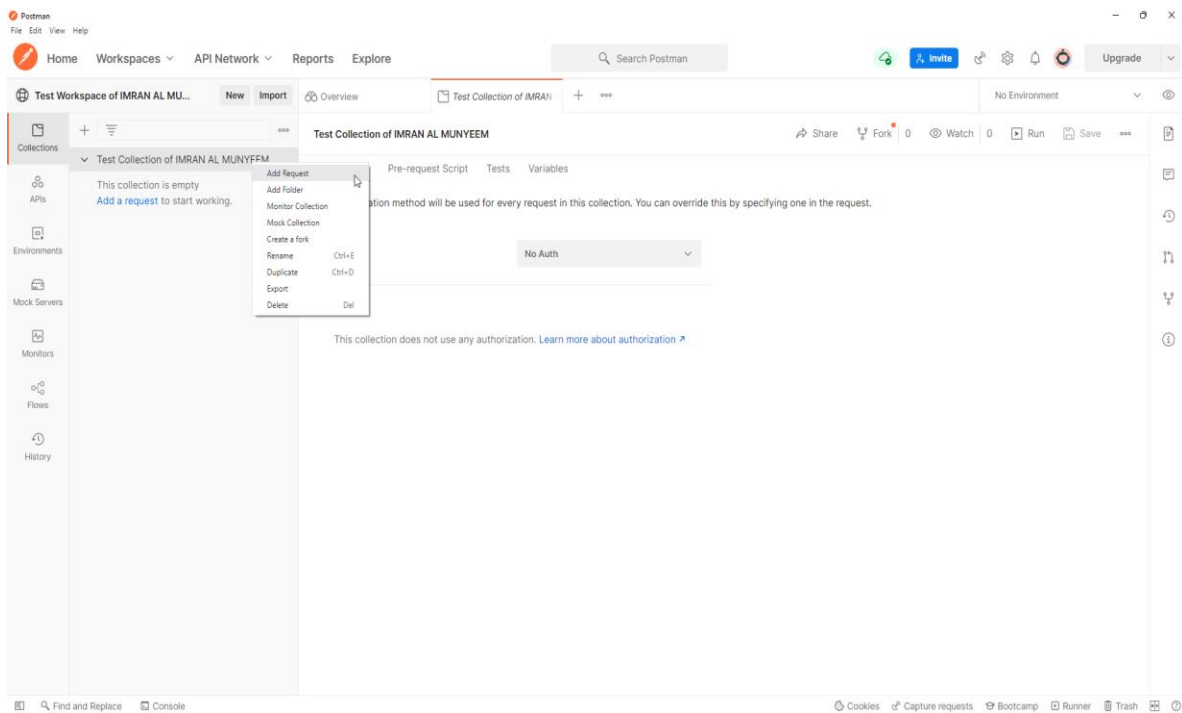
**Step 2**--- Give name for the request.



## 5. Create Requests, Analyse & Write Tests to Validate Responses:

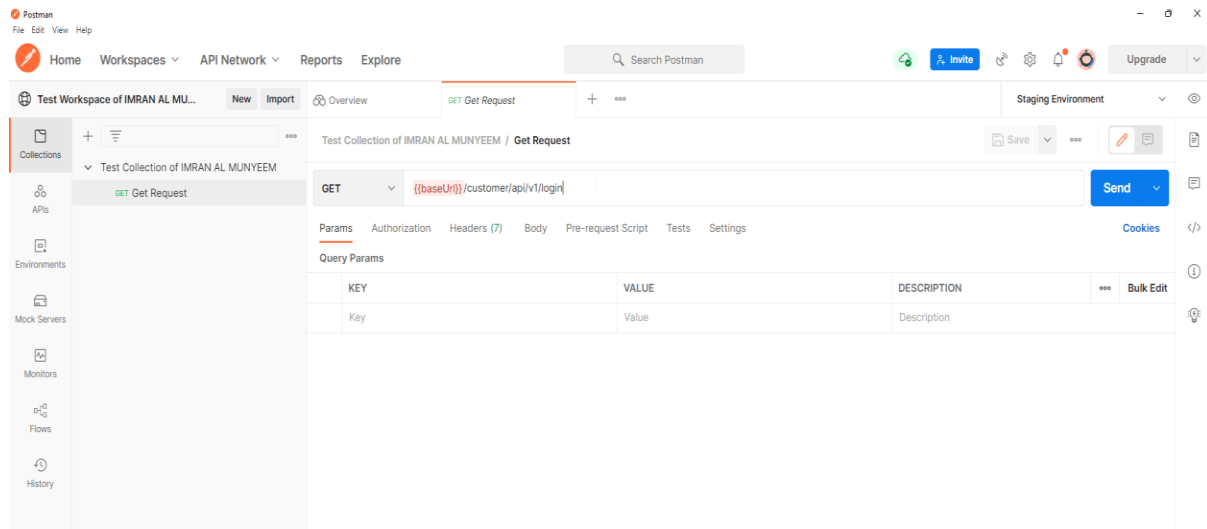
### Send Get Requests:

**Step 1:** Click a **new** tab to add a new request.



## Step 2: Creating a GET request for a REST API end point

- Set your HTTP request to **GET**
- Input the link in request **URL**.
- Click on **SEND** to execute.

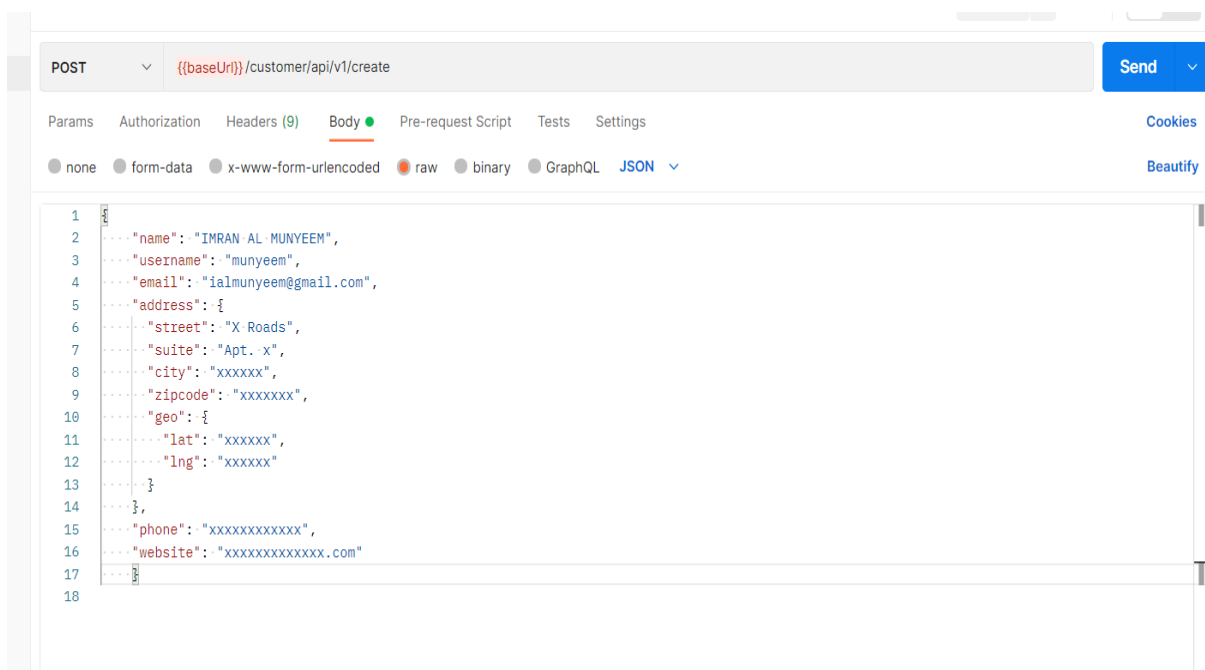


## Send Post Requests:

Post requests are used to do data manipulation by adding data to the endpoint. Now, let's add a user into the application. To do this, we need to send data to the application. We use POST request to send data. In POST request we send data in the body of the request and API returns some data in response to the POST request to us which validates the user has been created. We use the same data which we used in GET request to add a new user.

### Steps:

- Set your HTTP request to **POST**
- Input the link in request **URL**
- Click on **Body** Tab and select **"Raw"** radio button – Select **JSON** – Copy and paste just one user result from the previous get request as shown in the below screenshot.



Likewise, we will test other requests PUT, PATCH, DELETE, and others.

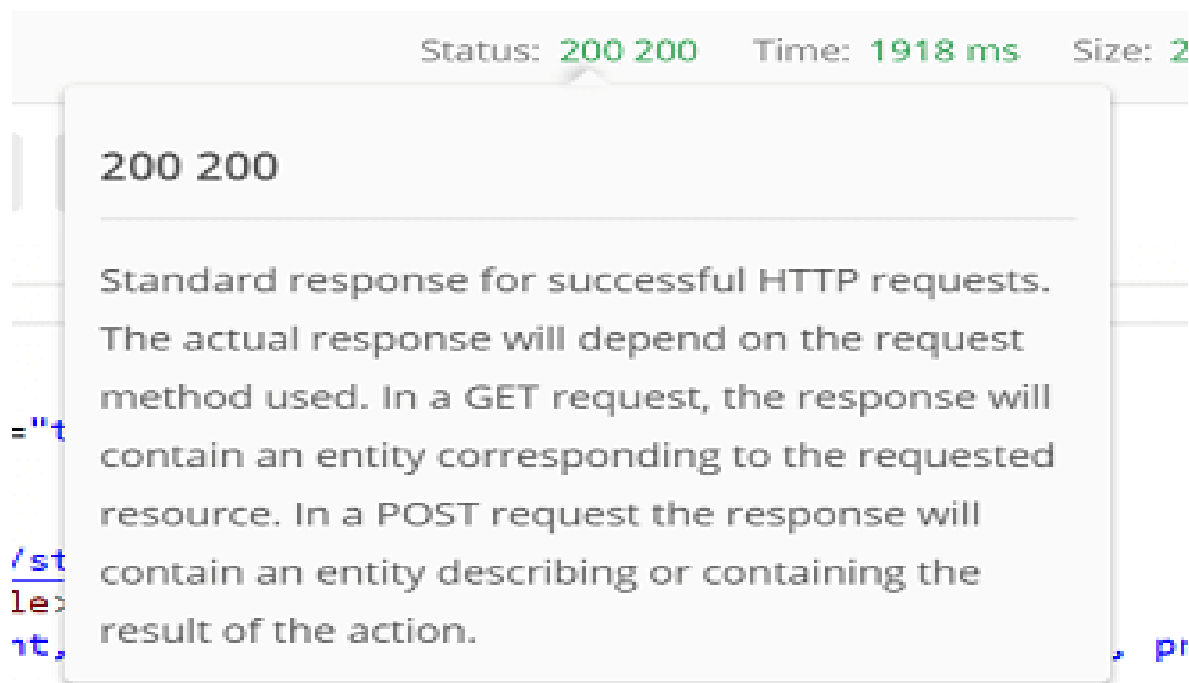
## Analyse Responses:

You should be able to see various data around the response after the server responds in the Body section.

In the above screenshot under the request headers, we can see response status code, time taken for the request to complete, the size of the payload

We can find the details about the response time and response size by hover over them.

**Response code for GET:** You can see 200 OK message in the screenshot below because our request is successful. In some cases, GET requests may be unsuccessful due to an invalid request URL or incorrect authentication.



**Response code for POST:** You will see 201 OK message for successful POST request. In some cases, POST requests may be unsuccessful due to an invalid request URL or incorrect data



**Response time:** We can see individual components like Connect time, Socket time, DNS lookup, etc.



**Response size:** We can see individual components like actual response size, how much size the headers are constituted etc.



**Cookies:** We can find session related information in the cookies that were returned from the server.

GEThttps://jsonplaceholder.typicode.com/users

SendSave

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookiesCode

▼ Headers (1)

KEY	VALUE	DESCRIPTION	***	Bulk Edit	Presets ▼
<input checked="" type="checkbox"/> Content-Type	application/json				
Key	Value	Description			

▶ Temporary Headers (8) ⓘ

BodyCookies (1)Headers (19)Test Results

Status: 200 OKTime: 1075msSize: 6.11 KBSave Response ▼

Name	Value	Domain	Path	Expires	HttpOnly	Secure
_cfduid	d27e2f460f649a7000038d0196eaf1e161583369076	typicode.com	/	Sat, 04 Apr 2020 00:44:36 GMT	true	false

## Response header:

Here we can find information about the request that got processed.

The screenshot shows the Headers tab in a web browser's developer tools. The request is a GET to `https://jsonplaceholder.typicode.com/users`. The Headers tab is active, showing a list of headers. The status is 200 OK, time is 1075ms, and size is 6.11 KB.

KEY	VALUE	DESCRIPTION
Content-Type	application/json	
Date	Thu, 05 Mar 2020 04:48:43 GMT	
Transfer-Encoding	chunked	
Connection	keep-alive	
X-Powered-By	Express	
Vary	Origin, Accept-Encoding	
Access-Control-Allow-Credentials	true	

Once you click on header you can see different information such as below. Although, every entry in the Headers tab is a header item we will just take a look at the most important ones.

- **Content-Type:** The content type is given as **text/HTML** because the response is being sent in the HTML which is one of the options.
- **Date:** This option shows the date, day and time of the response along with the time zone.
- **Server:** This option tells the name of the server which has responded to the request.
- **Cookie expires time:** As the name suggests, this option tells the expire time of the cookie that has been sent along with the response.

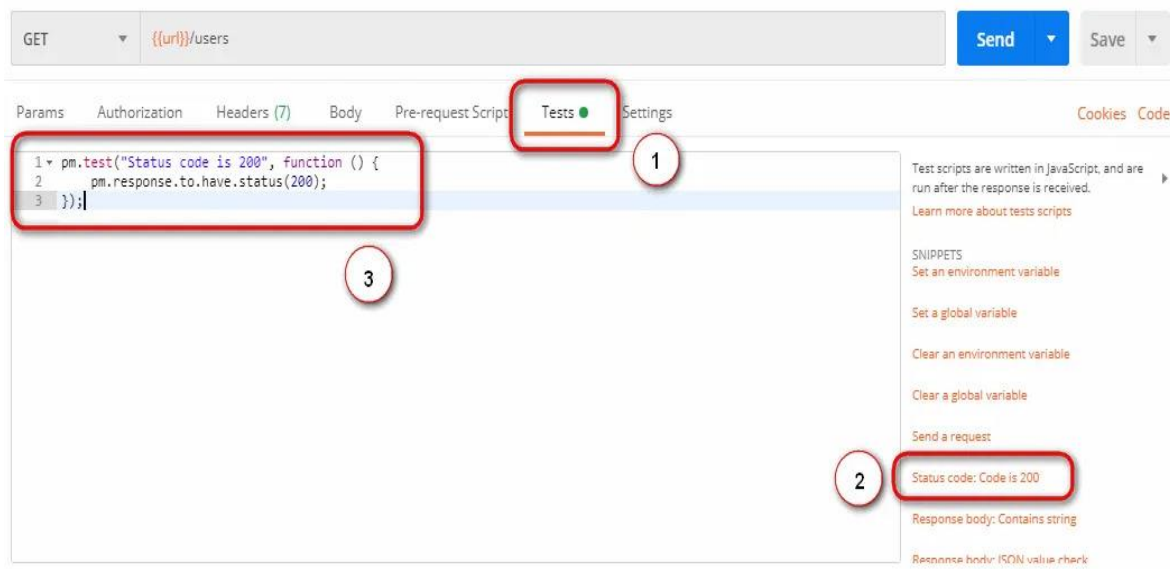
## Write Tests in Postman:

Postman Tests allow you to ensure that your API is working as expected. It is to establish integrations between services are functioning reliably, and to verify that new developments haven't broken any existing functionality. It helps you verify results such as successful or failed status, comparison of expected results etc.

**Let's start with some basic tests.**

**Step 1:**

- Go to the **GET request** which we created earlier.
- Switch to the **tests** tab.
- From the **snippets** section, click on "Status code: Code is 200". Script will be auto-populated.
- Click on **Send**.



Let's add another test. In this test, we do compare the expected result to the actual result. To do this,

### Step 1:

- Click on "Response body:JSON value check" from the snippets section.
- Let's check if Leanne Graham has the userid 1.

The screenshot shows the Postman interface. At the top, the request method is 'GET' and the URL is '{{url}}/users'. Below the request bar, there are tabs for 'Params', 'Authorization', 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Tests' tab is selected. In the 'Tests' tab, a JavaScript test script is written in a text area, which is highlighted with a red box. The script is as follows:

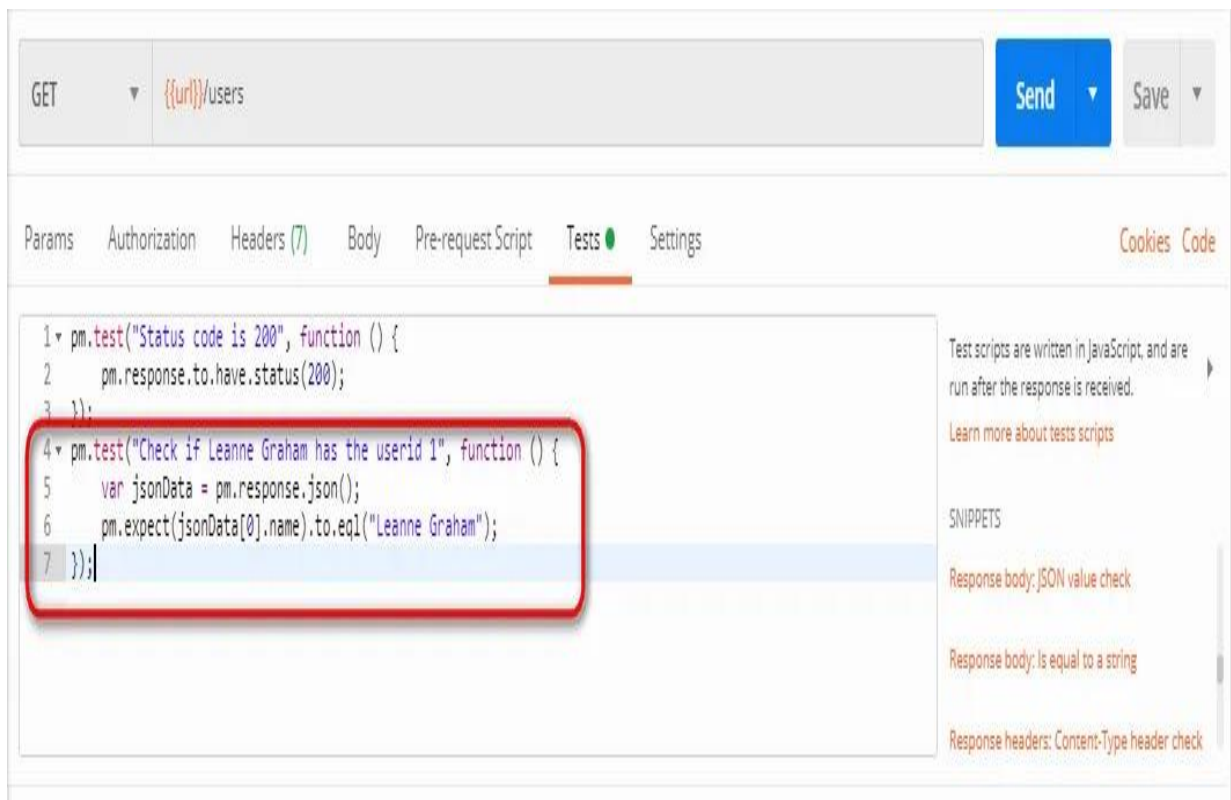
```
1 pm.test("Status code is 200", function () {  
2   pm.response.to.have.status(200);  
3 });  
4 pm.test("Your test name", function () {  
5   var jsonData = pm.response.json();  
6   pm.expect(jsonData.value).to.eql(100);  
7 });
```

To the right of the test script, there is a 'SNIPPETS' section, also highlighted with a red box. It contains three options: 'Response body:JSON value check', 'Response body: Is equal to a string', and 'Response headers: Content-Type header check'. The first option, 'Response body:JSON value check', is selected.

Below the 'Tests' tab, there is a 'Cookies' tab and a 'Code' tab. The 'Code' tab is currently active, showing a text area for the test script.

## Step 2:

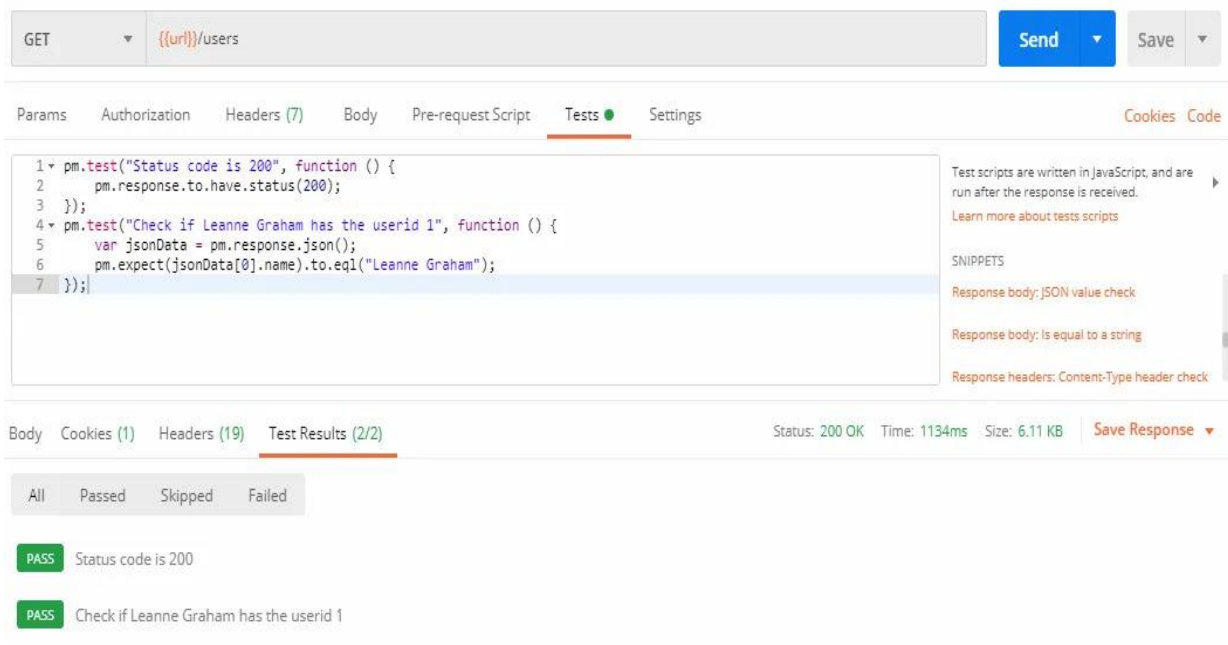
- To specify the test name exactly what we want to test, simply replace “Your Test Name” from the code with “Check if Leanne Graham has the userid 1”.
- Also replace jsonData.value with jsonData[0].name. To get the path (It is there in the body of earlier GET result).
- . Since “Leanne Graham” is userid 1, jsonData is in the first result which should start with 0. To get the second result, use jsonData[1] and so on for succeeding results.



**Step 3:** Click send.

You can see the test result below in the screenshot.

**N.B:** They are passed. Can be failed if the test scripts are not correct, or network issues, or for changing or moving anything.



**We can create more tests depending on our requirement. Explore the tools by trying different tests.**

To get more ideas about Snippets you can visit the following link below:

<https://learning.postman.com/docs/sending-requests/generate-code-snippets/#generating-code-snippets-in-postman>

## 6. Create environment variable:

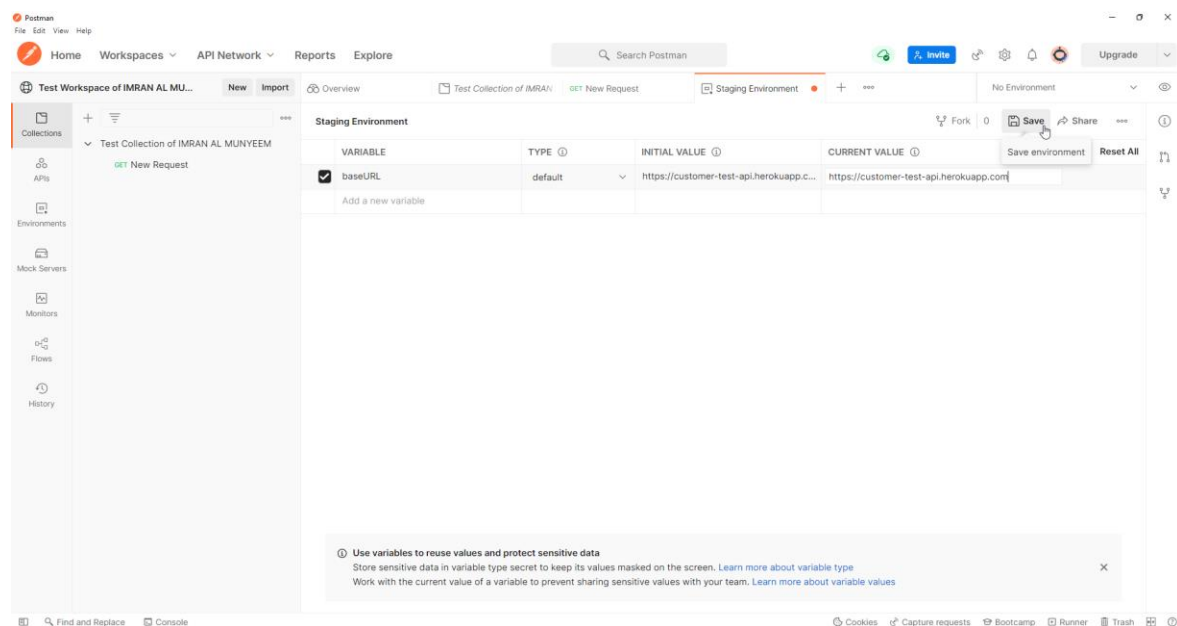
**Step 1** --- Click on the eye icon

**Step --- 2:** Click on “Add” button and give Environment name

**Step --- 3:** Now set VARIABLE as baseUrl and INITIAL VALUE for example

<https://customer-test-api.herokuapp.com>

**N.B:** You will see that CURRENT VALUE will be set automatically after you set INITIAL VALUE



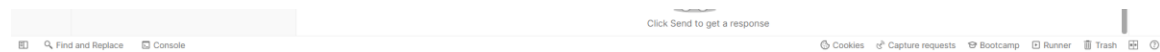
**Step 4 ---** Now go to the collection again and replace the base url “<https://customer-test-api.herokuapp.com>” With {{baseUrl}}.



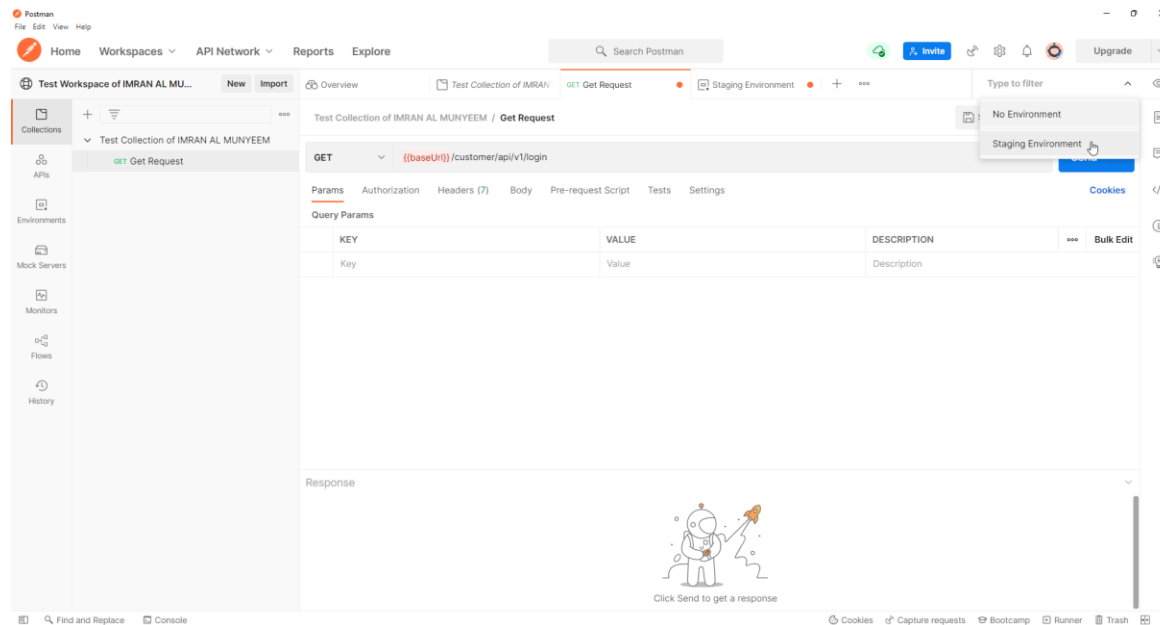
**Step 5** --- Don't forget to click on Save button after adding url and variables.

**Important notes:** *When you save something in an environment variable, then while using it, you have to put the variable name within a curly bracket.*

Then, you will see the `{{baseUrl}}` color gets orange. If the color is not orange, there some spelling mistake or you don't set the environment from the environment list.



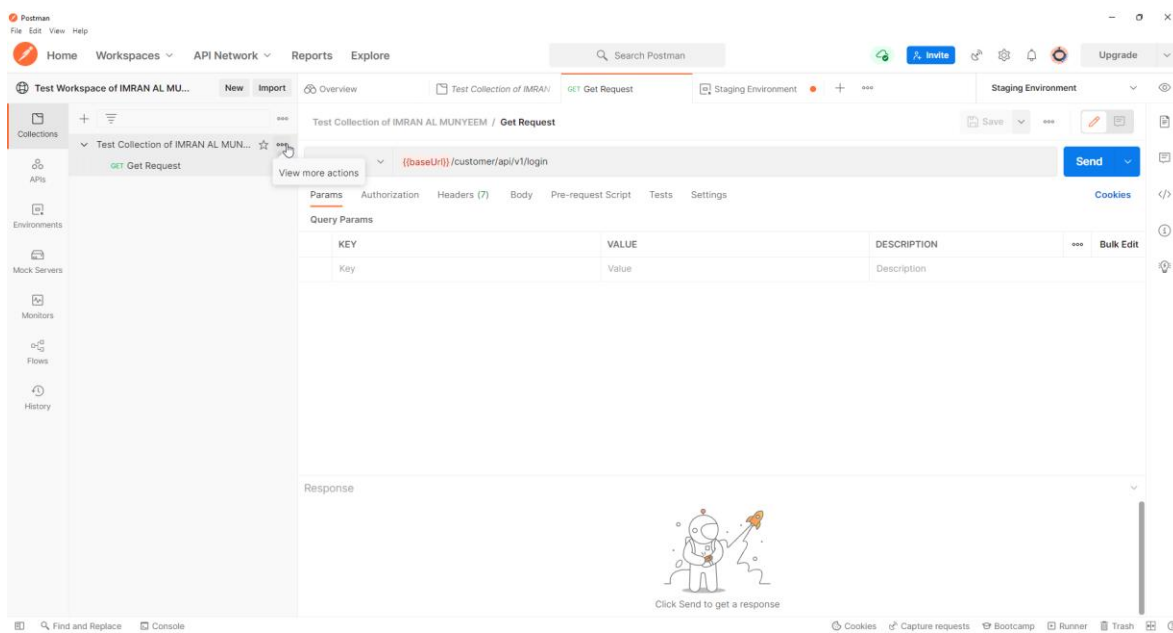
**Step 6** --- Don't forget to set the environment from the environment list.



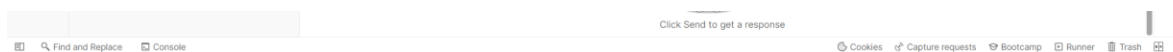
## 7. Export Collection: Export in 2 ways

### -----Way 1-----

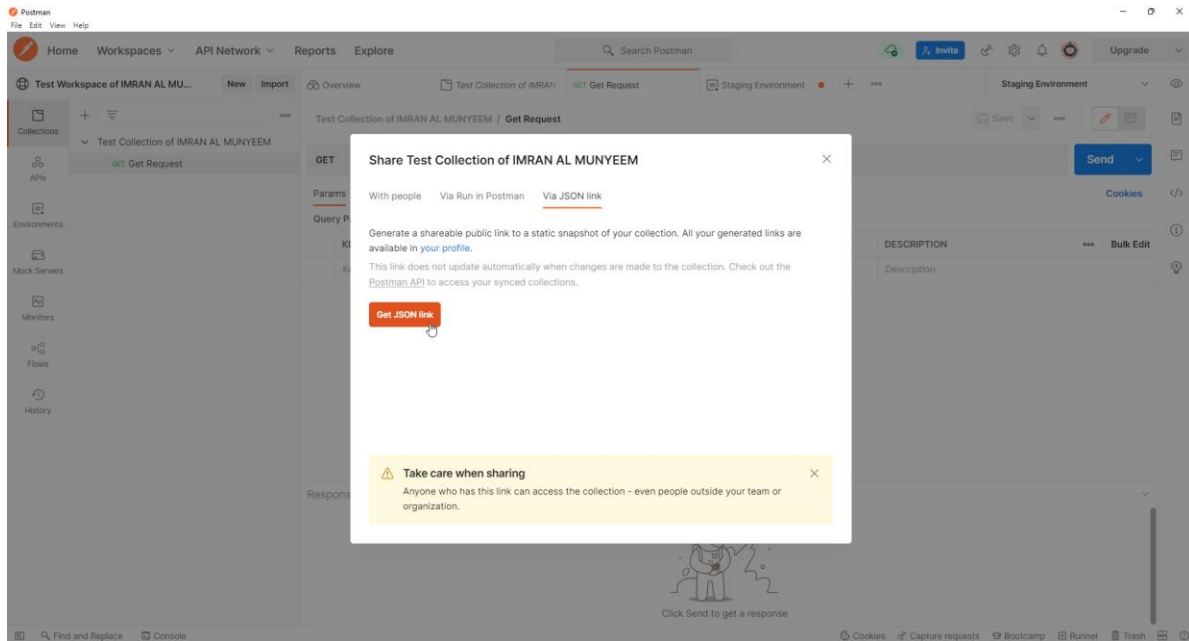
**Step 1** --- click on the [...] button besides your collection



**Step 2** --- Click on Share collection



### Step 3 --- Click on “Get public link” button



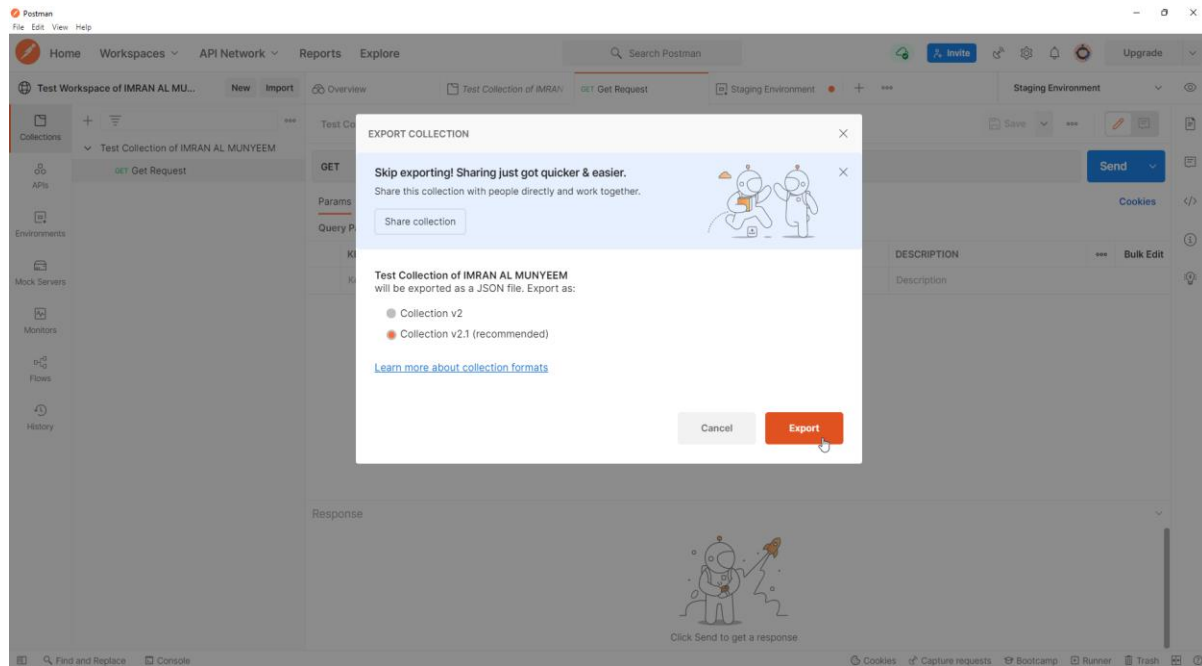
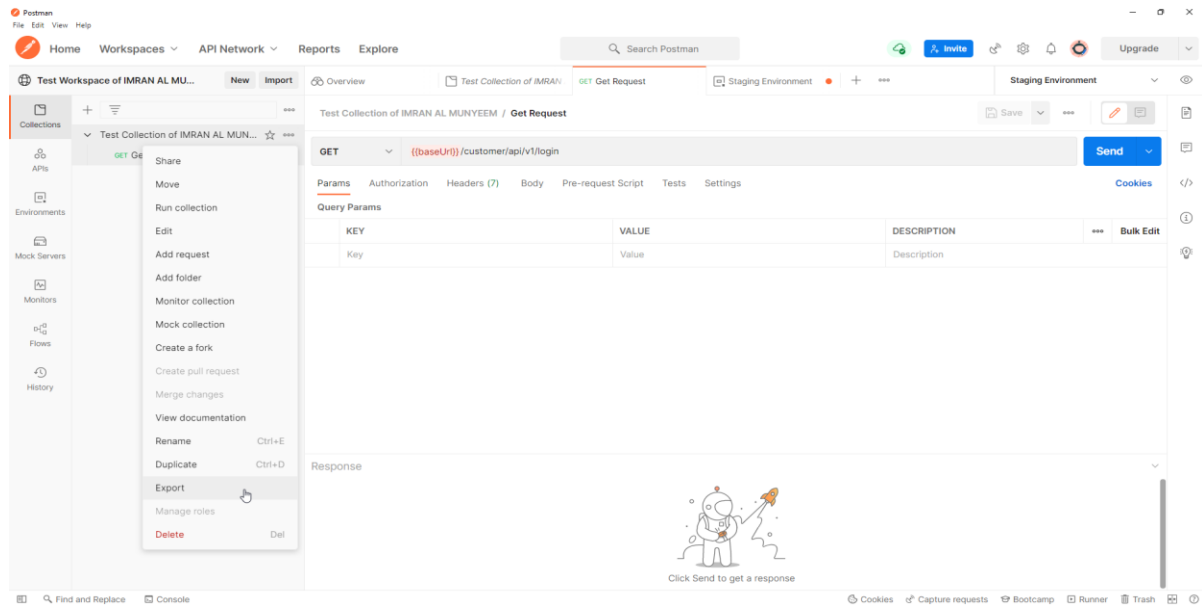
**Step 4** --- Then you will get a link. This link you can share with anybody and h/she can import the collection

### Export Collection:

#### -----Way 2-----

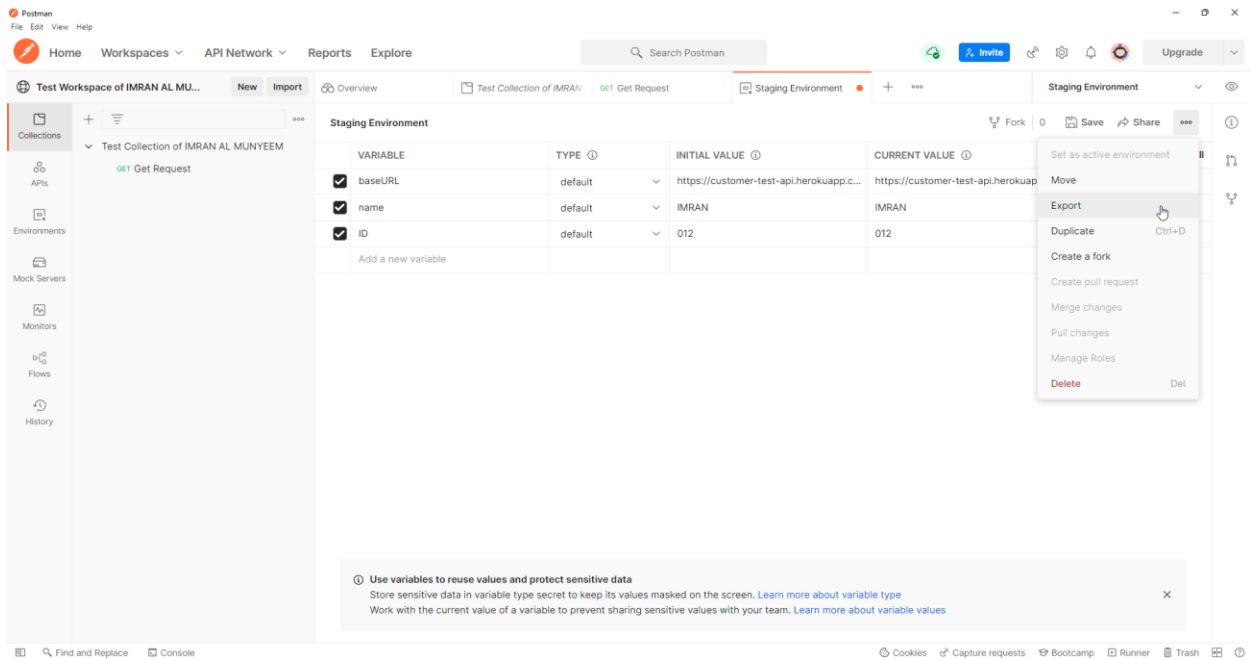
**Step 1** --- Click on the [...] button besides your collection like before.

**Step 2** --- Click on Export button and save to your local disk. Now anybody can import the collection



## 8. Export Environment

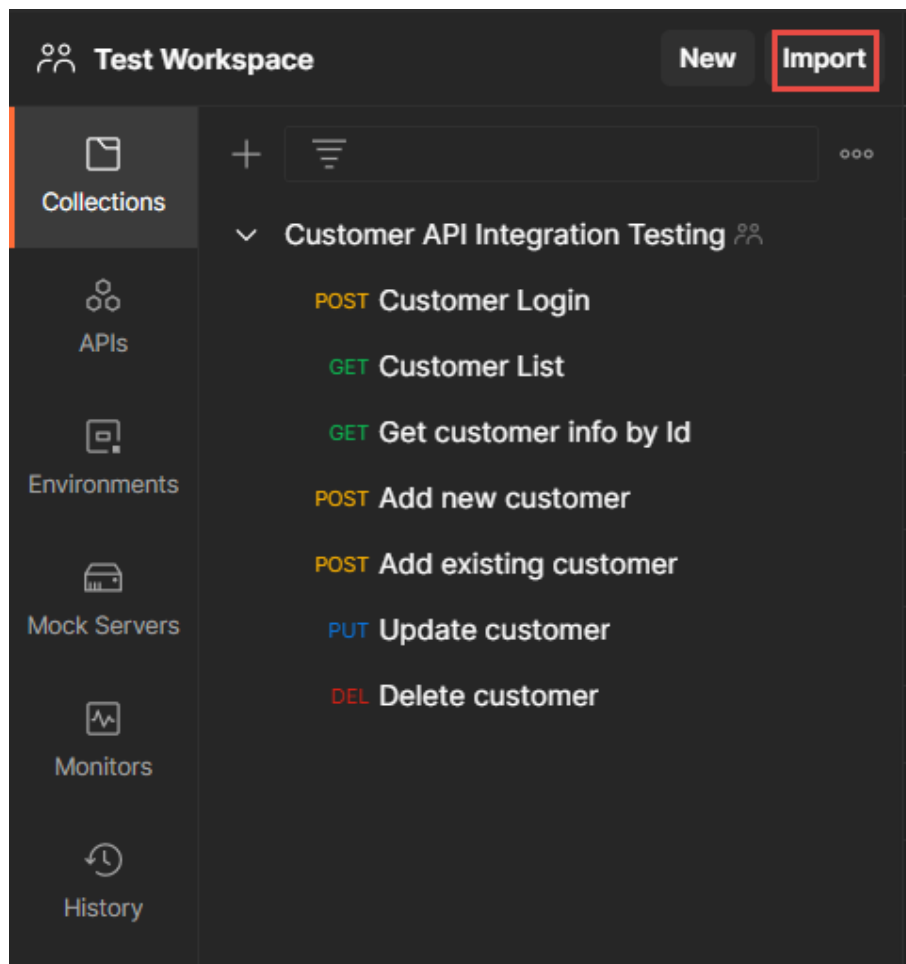
**Step:** Same way, you can export the environment and then save to your local disk where you saved the collection.



## 9. Import Collection

**Step 1** --- Go to collection tab.

**Step 2** --- Click on Import button.



You'll be able to import as link and files.

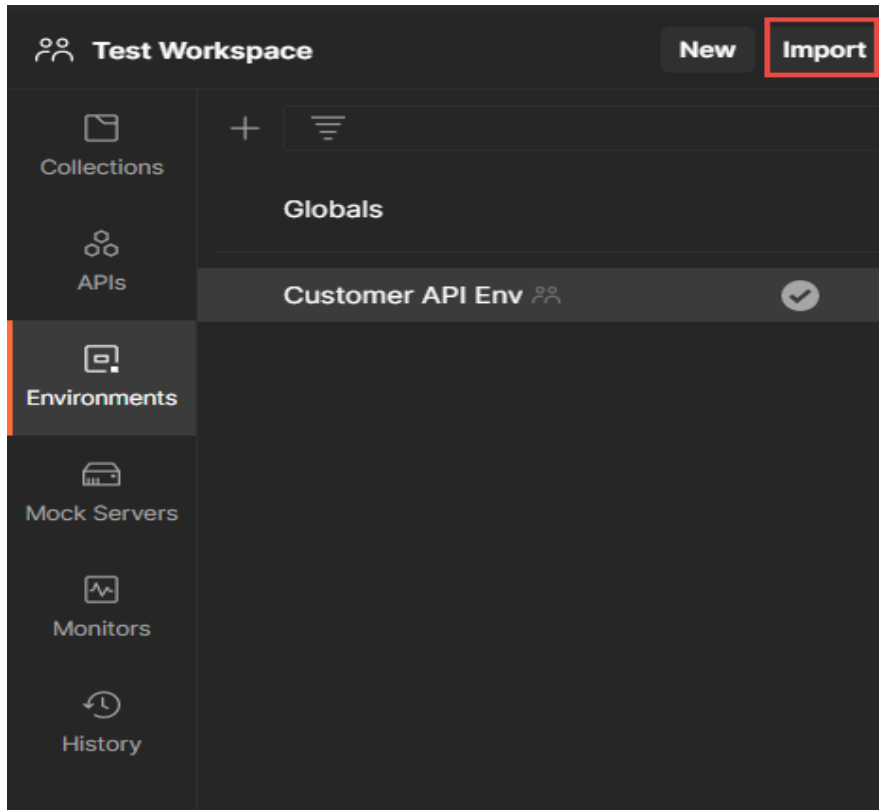
You will see the whole collection has been imported. Anybody can import your collection this way.

## 10. Import Environment

**Step 1** — Go to Environment tab.

**Step 2** — Click on Import button.

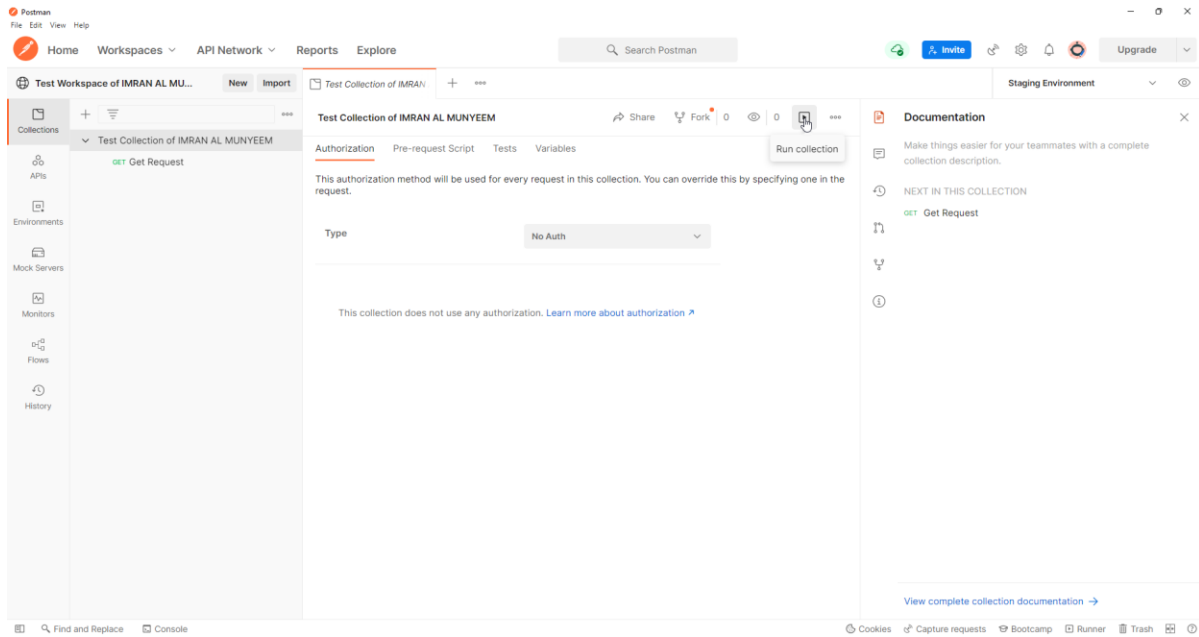
**Step 3** — Locate the environment file you just exported. Now the importer can call any API.



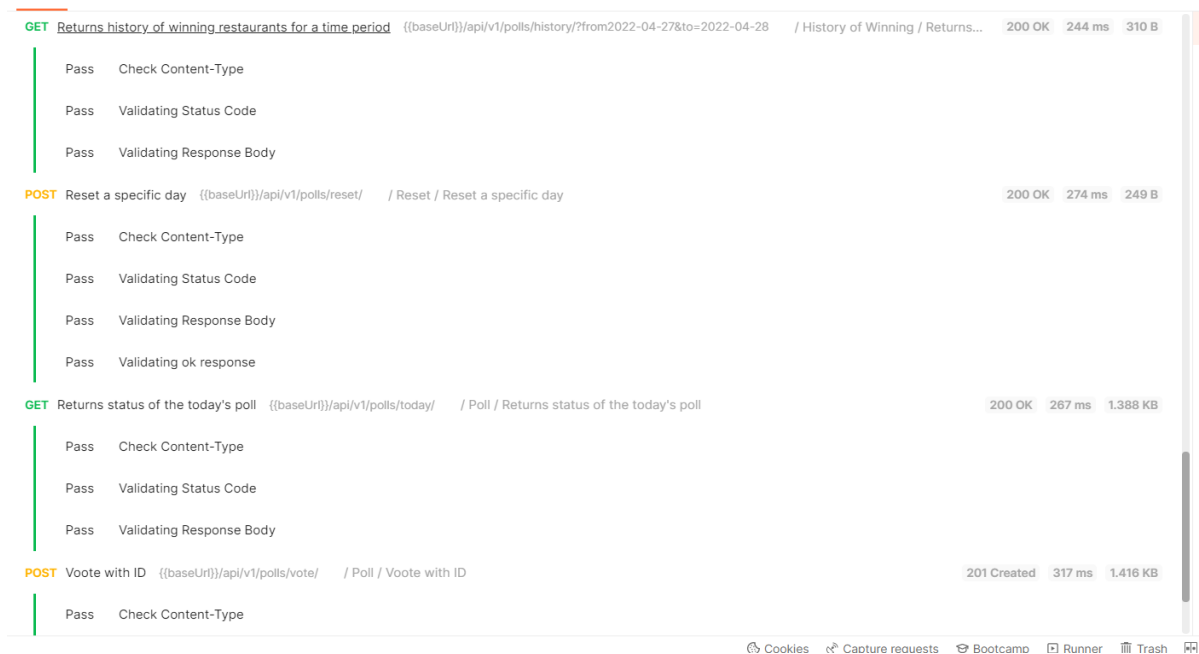
## 11. Run Collection: Run in 2 ways

-----One way-----

**Step 1** — Select your collection and you will see a tab named Run. Click on there.



**Step 2** --- You will see Iteration is set by default 1 which means the whole test will run for 1 time.

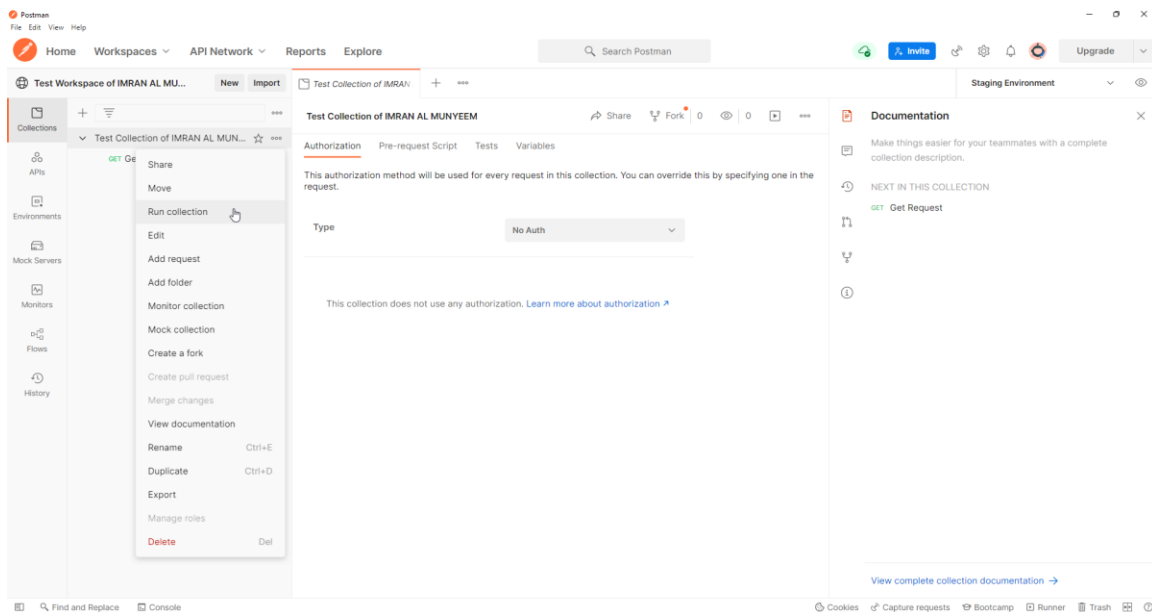




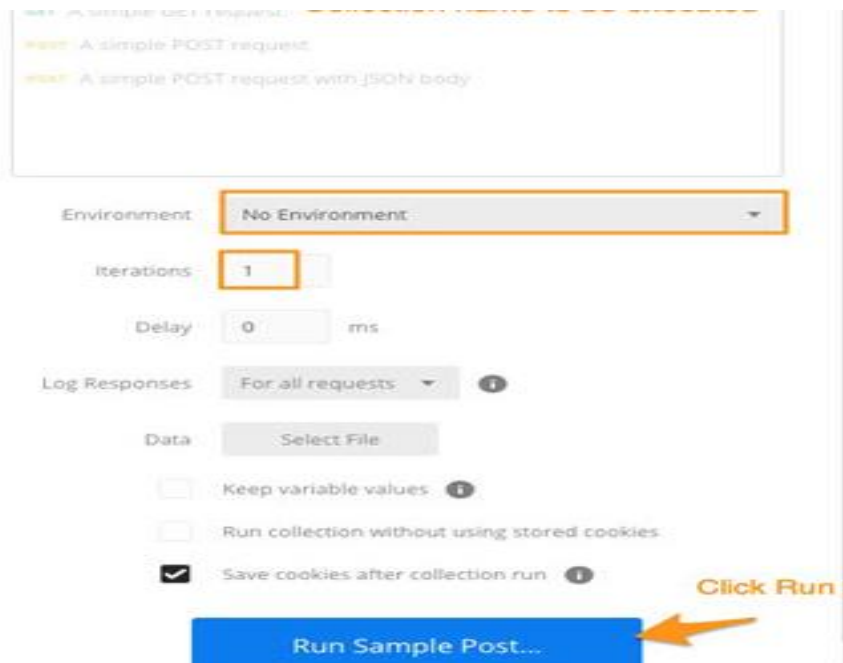
## Run Collection

-----Another way-----

Step 1 --- Click on Run from [...]



Step 2 --- Set Environment, iteration, data if needed.



## 12. Add authorization if available:

### Steps:

- Click on Headers
- put an authorization name
- Put authorization key in the 'Value' field if you have.

Herokuapp-Restful-APIs / History of Winning / Returns history of winning restaurants for a time period

GET `{{baseUrl}}/api/v1/polls/history/?from2022-04-27&to=2022-04-28` Send

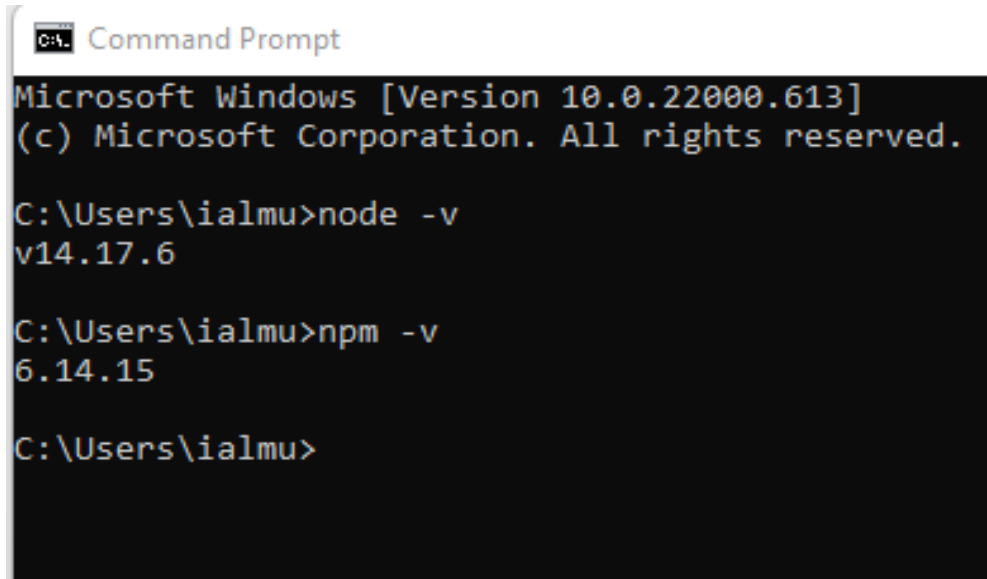
Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Cache-Control	no-cache				Go to settings
<input checked="" type="checkbox"/>	Postman-Token	<calculated when request is sent>				
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>				
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.0				
<input checked="" type="checkbox"/>	Accept	/*/*				
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br				
<input checked="" type="checkbox"/>	Connection	keep-alive				
<input checked="" type="checkbox"/>	Authorization	Token 96c3fe710340701c311d625bd3e9142a37cda098				
	Key	Value	Description			

Response

### 13. Newman Guide – Report Generation

**Step 1** — check if node and npm is already installed: Open cmd and write

A screenshot of a Windows Command Prompt window. The title bar says "C:\> Command Prompt". The window content shows the Microsoft Windows version (10.0.22000.613) and copyright information. The user has entered two commands: "node -v" which returned "v14.17.6", and "npm -v" which returned "6.14.15". The prompt is currently at "C:\Users\ialmu>".

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ialmu>node -v
v14.17.6

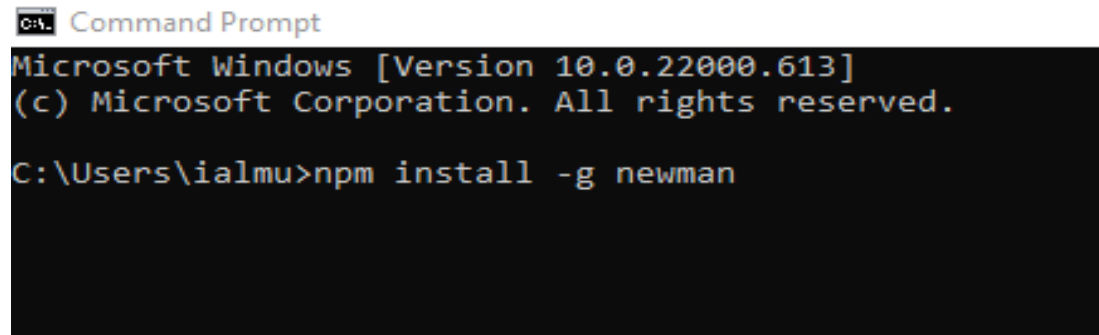
C:\Users\ialmu>npm -v
6.14.15

C:\Users\ialmu>
```

**Step 2** — Install nodejs from (<https://nodejs.org/en/download/>)

**Step 3** — Install npm from (<https://www.npmjs.com/package/download>)

**Step 4** — Run this command on windows cmd to install newman (npm install -g newman)

A screenshot of a Windows Command Prompt window. The title bar says "C:\> Command Prompt". The window content shows the Microsoft Windows version (10.0.22000.613) and copyright information. The user has entered the command "npm install -g newman".

```
C:\> Command Prompt
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ialmu>npm install -g newman
```

**Step 5** — Run this command on windows cmd to install newman htmlextra reporter (npm install -g newman-reporter-htmlextra)

 Command Prompt

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ialmu>npm install -g newman-reporter-htmlextra
```

**Step 6** — Run this command on windows cmd to run the collection and make html report (newman run "collectionname" -r htmlextra)

 Command Prompt

```
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ialmu>newman run "C:\Imran\Fiverr\HerokuApp-API-Test-Postman\Collection\Postman Collection.json" -r htmlextra
```

**Step 7** --- Then a folder named "Newman" will be created automatically where the report will be stored.

**Step 8** — To get more help on newman, visit here

(<https://www.npmjs.com/package/newman-reporter-html>)

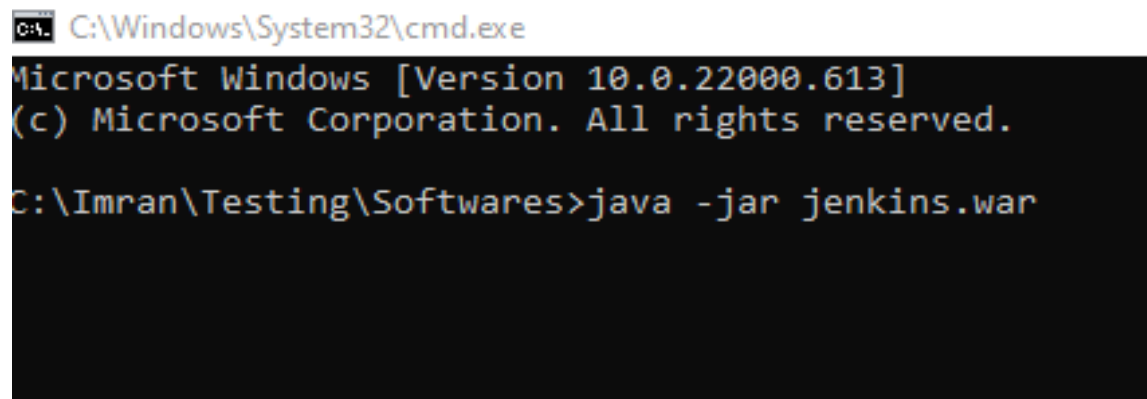
## 14. Run from Jenkins

**Step 1** --- Download and install Java and set the environment variable for it ([https://www.java.com/download/ie\\_manual.jsp](https://www.java.com/download/ie_manual.jsp))

**Step 2** --- Download and install JDK 8 and set the environment variable for it (<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>)

**Step 3** --- Download Jenkins war file (<https://www.jenkins.io/download/>)

**Step 4** --- Go to the Jenkin's location, open cmd and write "java -jar jenkins.war" and it will run jenkins server on your local port 8080 (N.B: Don't close the cmd)

A screenshot of a Windows command prompt window. The title bar shows 'C:\Windows\System32\cmd.exe'. The window content displays the Microsoft Windows version (10.0.22000.613) and copyright information. The current directory is 'C:\Imran\Testing\Softwares'. The command 'java -jar jenkins.war' has been entered at the prompt.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

C:\Imran\Testing\Softwares>java -jar jenkins.war
```

**Step 5** --- Go there to use Jenkins (<http://localhost:8080/credentials/>)

**Step 6** --- Setup and login to your Jenkins server.



Welcome to Jenkins!

imranalmunyeem

\*\*\*\*\*

Sign in

☐ Keep me signed in

**Step 7** --- Click on 'Manage Jenkins', find and click on "Plugin Manager" , then select and install all the recommended plugins.

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Open Blue Ocean

Lockable Resources

New View

Build Executor Status

1 Idle

2 Idle

All					
S	W	Name ↓	Last Success	Last Failure	Last Duration
⊗	⚙	ReqRes-Restful-APIs-Postman	N/A	2 mo 12 days - #10	0.92 sec
Icon: S M L					

Icon legend

Atom feed for all

Atom feed for failure

← → ↻ localhost:8080/manage

Dashboard ▸

■ New View

Build Queue ^

No builds in the queue.

Build Executor Status ^

1 Idle

2 Idle

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#). [Set up agent](#)

The word "master" is being retired as the term for the main Jenkins process and the built-in node. The main process is now called "controller" and the built-in node is called just "built-in node". The UI has been updated with these changes. The following features are also affected:

- The implicit label of the built-in node changes from master to built-in.
- The built-in node's NODE\_NAME environment variable also changes from master to built-in.

These changes could affect build behavior, so are not applied automatically. Before you apply these changes, you should do the following:

- Review label assignments in job configurations and tool installers for uses of master label. Any such label assignments will not match the built-in node after migration. Besides updating these assignments, add the master label to the built-in node.
- Review use of the NODE\_NAME environment variable in build scripts.

[Learn more](#).

Java 11 is the recommended version to run Jenkins on; please consider upgrading.

Warnings have been published for the following currently installed components: [Go to plugin manager](#) [Configure](#)

Jenkins 2.319.2 core and libraries:  
[DoS vulnerability in bundled XStream library](#)

Favorite 2.3.3:  
[Stored XSS vulnerability](#)

← → ↻ localhost:8080/pluginManager/

Jenkins

Search

Dashboard ▸ Plugin Manager ▸

Back to Dashboard

Manage Jenkins

filter

Updates Available Installed Advanced

Install	Name	Version	Released
<input type="checkbox"/>	<a href="#">Ant</a> <a href="#">Build Tools</a> Adds Apache Ant support to Jenkins	475.vf34069fe73c	10 days a
<input type="checkbox"/>	<a href="#">Autofavorite for Blue Ocean</a> Automatically favorites multibranch pipeline jobs when user is the author	1.2.5	2 mo 23 c
<input type="checkbox"/>	<a href="#">Bitbucket Branch Source</a> <a href="#">bitbucket</a> <a href="#">Source Code Management</a> Allows to use Bitbucket Cloud and Bitbucket Server as sources for multi-branch projects. It also provides the required connectors for Bitbucket Cloud Team and Bitbucket Server Project folder (also known as repositories auto-discovering).	765.v5a_2d6a_23c01d	17 days a
<input type="checkbox"/>	<a href="#">Bitbucket Pipeline for Blue Ocean</a> BlueOcean Bitbucket pipeline creator	1.25.3	2 mo 3 d


**Step 8** --- Click on "New Item" from the left side, enter project name, select project type as your choice (Recommended: Freestyle) and save.

← → ↻

localhost:8080

🔍 Search

🔔 4 🔒 2 👤

 **Jenkins**

Dashboard ▾

New Item

People

Build History

Manage Jenkins

My Views

Open Blue Ocean

Lockable Resources

New View

Build Queue

No builds in the queue.

Build Executor Status

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
⊗	☁	ReqRes-Restful-APIs-Postman	N/A	2 mo 12 days - #10	0.92 sec

Icon: S M L

Icon legend ☰ Atom feed for all ☰ Atom feed for failure

← → ↻

localhost:8080/view/all/newJob


🔍 Search


🔔 4 🔒 2 👤


Dashboard ▾ All ▾


Enter an item name


» Required field

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Maven project**  
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

 **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

 **Multi-configuration project**



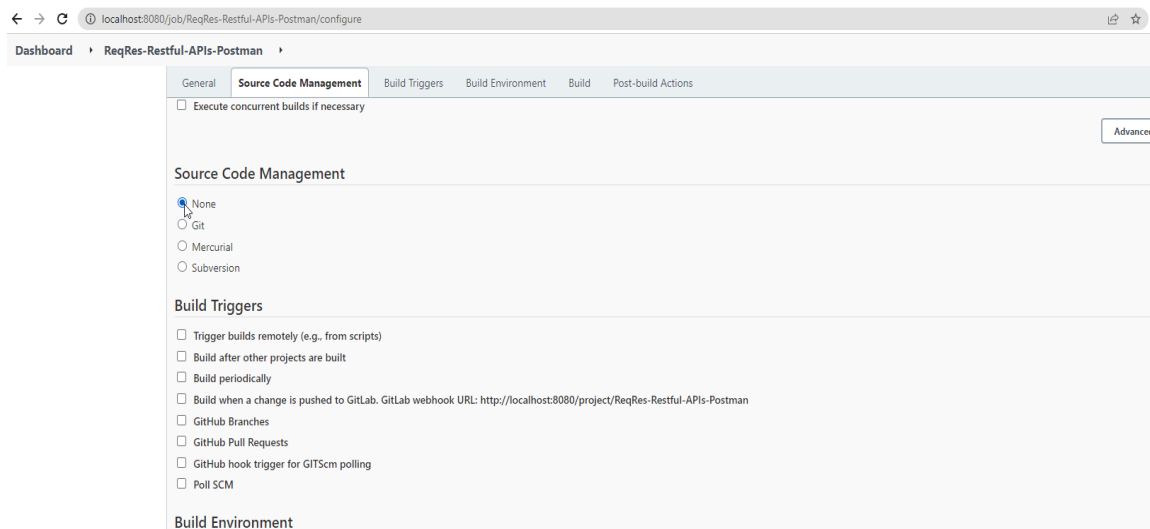
**Step 9** --- Go to the project and click on "Configure" from the left.



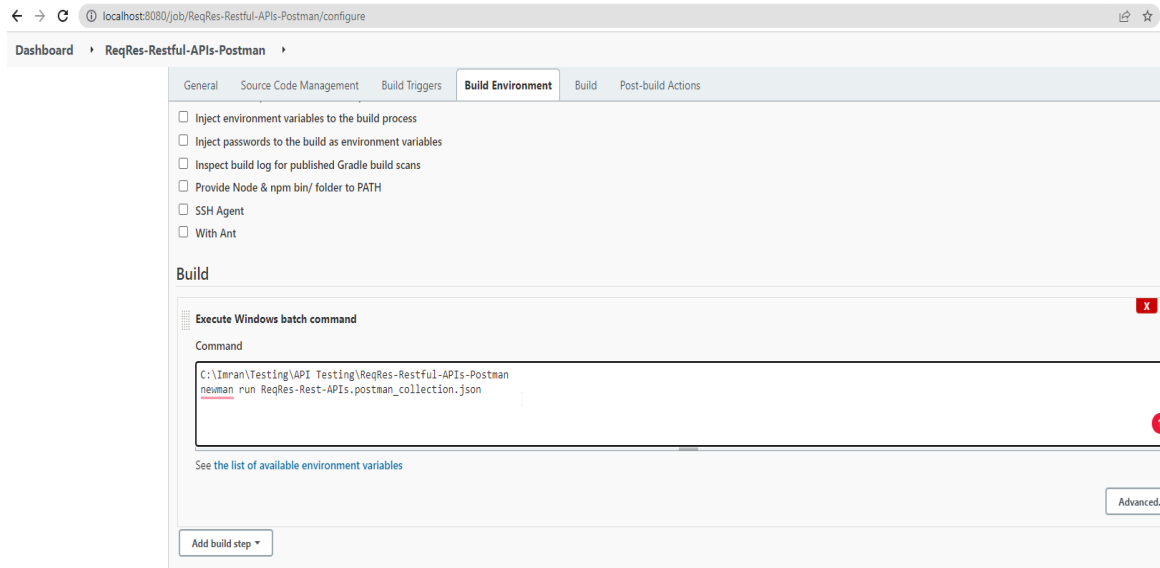
**Step 10** --- Running from Different sources

## Run from local device:

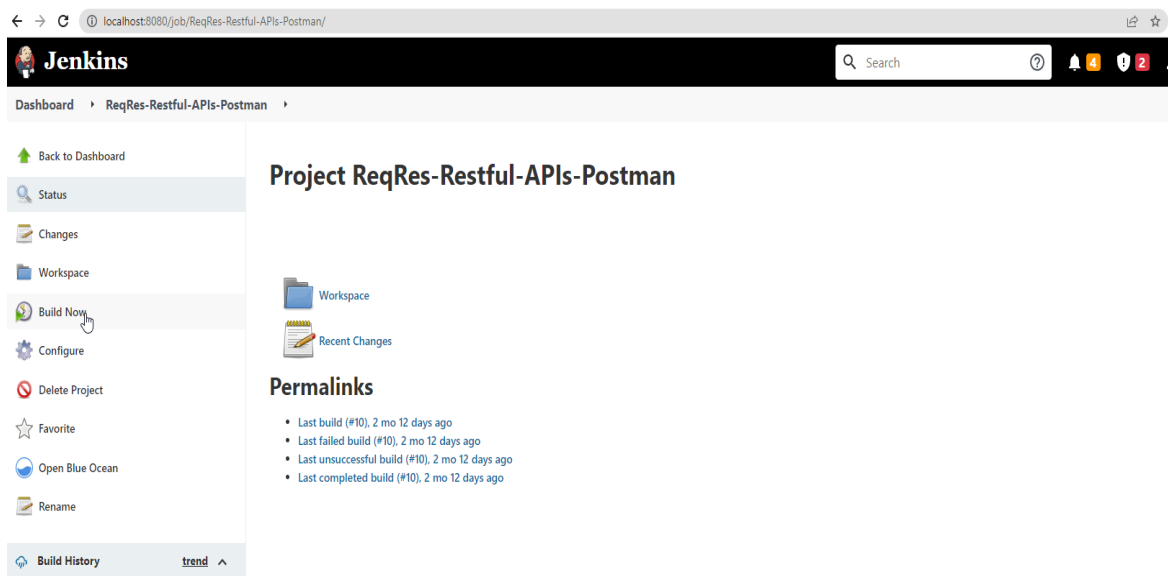
----> Select source code management to NO



----> Select 'Execute Windows Batch Command' from 'Build' and write the Current project location and collection name and then save it.



---> Click on Build now from the left. It will start building.



---> Click on console output to see the output -> click on view as plain text to see the plain view.

← → ↻ localhost:8080/job/ReqRes-Restful-APIs-Postman/lastBuild/console

# Jenkins

Search ? 🔔 4 📢 2 👤

Dashboard > ReqRes-Restful-APIs-Postman > #14

- Back to Project
- Status
- Changes
- Console Output**
- View as plain text
- Edit Build Information
- Delete build #14
- Environment Variables
- Open Blue Ocean
- Previous Build

## Console Output

Started by user **Imran Al Munyeen**  
Running as SYSTEM  
[EnvInject] - Loading node environment variables.  
Building in workspace C:\Users\ialmu\.jenkins\workspace\ReqRes-Restful-APIs-Postman  
The recommended git tool is: NONE  
No credentials specified  
> git.exe rev-parse --resolve-git-dir C:\Users\ialmu\.jenkins\workspace\ReqRes-Restful-APIs-Postman\.git # timeout=10  
Fetching changes from the remote Git repository  
> git.exe config remote.origin.url https://github.com/imranalmunyeen/ReqRes-Restful-APIs-Postman # timeout=10  
Fetching upstream changes from https://github.com/imranalmunyeen/ReqRes-Restful-APIs-Postman  
> git.exe --version # timeout=10  
> git --version # 'git version 2.36.0.windows.1'  
> git.exe fetch --tags --force --progress -- https://github.com/imranalmunyeen/ReqRes-Restful-APIs-Postman +refs/heads/\*:refs/remotes/origin/\* # timeout=10  
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10  
> git.exe rev-parse "origin/master^{commit}" # timeout=10  
ERROR: Couldn't find any revision to build. Verify the repository and branch configuration for this job.  
Finished: FAILURE

## Running from Github:

---> Select 'GitHub Project' from 'General' and put GitHub link

← → ↻ localhost:8080/job/ReqRes-Restful-APIs-Postman/configure

# Jenkins

Search ? 🔔 4 📢 2 👤

Dashboard > ReqRes-Restful-APIs-Postman >

- General**
- Source Code Management
- Build Triggers
- Build Environment
- Build
- Post-build Actions

Description

[Plain text] Preview

☐ Discard old builds

☒ GitHub project

Project url

https://github.com/imranalmunyeen/ReqRes-Restful-APIs-Postman

☐ Use custom workspace

GitLab Connection

Advanced..

---> Select source code management to Git and add repository link.

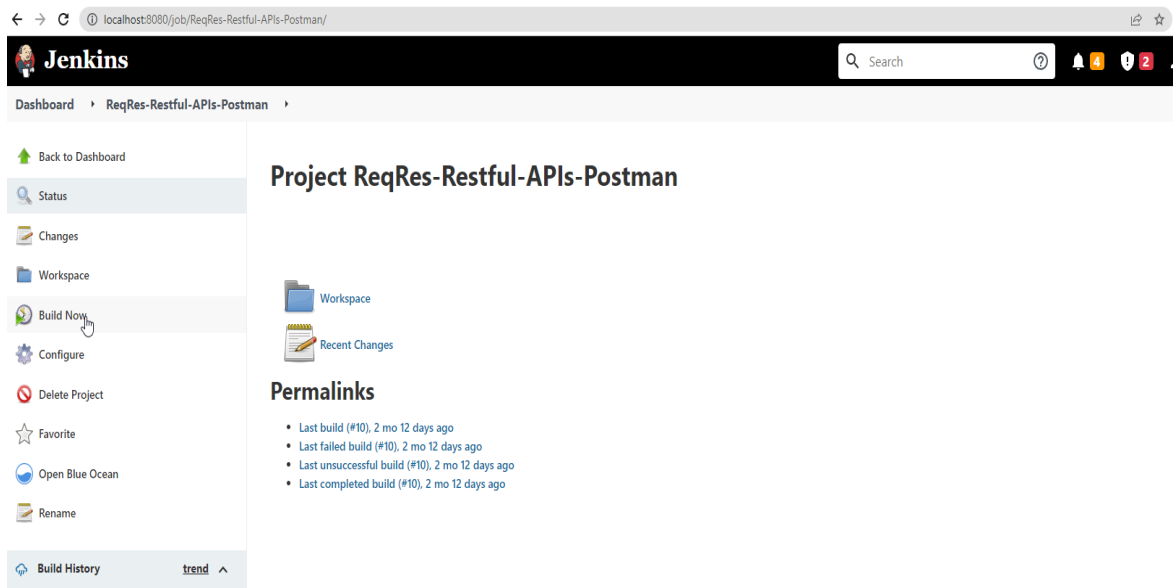
**N.B:** Add credentials if needed.

The screenshot shows the Jenkins configuration page for a job named 'ReqRes-Restful-APIs-Postman'. The 'Source Code Management' tab is selected. Under 'Source Code Management', 'Git' is chosen. The 'Repository URL' is set to 'https://github.com/imranalimuneem/ReqRes-Restful-APIs-Postman'. A red error message 'Please enter Git repository.' is displayed below the URL. The 'Credentials' section shows a dropdown menu with 'none' selected and an 'Add' button. An 'Add Repository' button is at the bottom. The 'Branches to build' section has a 'Branch Specifier (blank for \'any\')' field with '\*/master' entered.

---> Select 'Execute Windows Batch Command' from 'Build'. Write the Current project location and collection name, save.

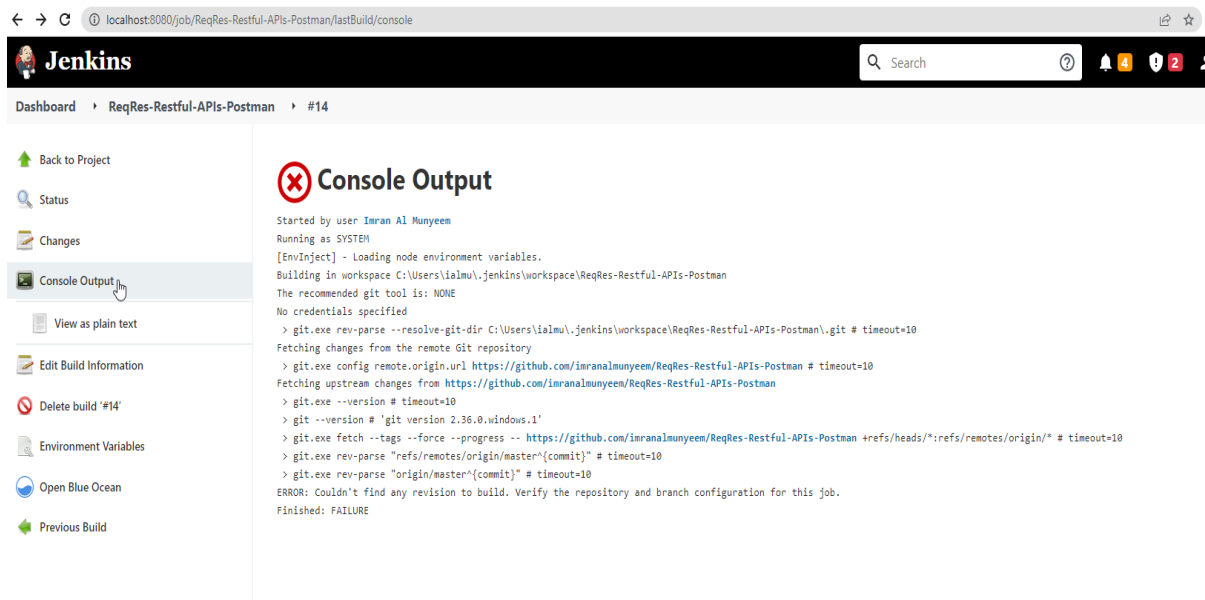
The screenshot shows the Jenkins configuration page for the same job, with the 'Build Environment' and 'Build' tabs visible. In the 'Build Environment' section, several checkboxes are present: 'Inject environment variables to the build process', 'Inject passwords to the build as environment variables', 'Inspect build log for published Gradle build scans', 'Provide Node & npm bin/ folder to PATH', 'SSH Agent', and 'With Ant'. In the 'Build' section, 'Execute Windows batch command' is selected. The 'Command' field contains the text: 'C:\Imran\Testing\API Testing\ReqRes-Restful-APIs-Postman\n Newman run ReqRes-Rest-APIs.postman\_collection.json'. A red error message '1' is shown at the bottom right of the command field. An 'Add build step' button is at the bottom.

---> Click on Build now from the left. It will start building.



The screenshot shows the Jenkins dashboard for the project 'ReqRes-Restful-APIs-Postman'. The left sidebar contains a list of actions: 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now' (highlighted with a mouse cursor), 'Configure', 'Delete Project', 'Favorite', 'Open Blue Ocean', and 'Rename'. The main area displays the project name, a 'Workspace' icon, 'Recent Changes', and a 'Permalinks' section with four links: 'Last build (#10), 2 mo 12 days ago', 'Last failed build (#10), 2 mo 12 days ago', 'Last unsuccessful build (#10), 2 mo 12 days ago', and 'Last completed build (#10), 2 mo 12 days ago'.

---> Click on console output to see the output -> click on view as plain text to see the plain view.



The screenshot shows the Jenkins console output for build #14 of the project 'ReqRes-Restful-APIs-Postman'. The left sidebar contains a list of actions: 'Back to Project', 'Status', 'Changes', 'Console Output' (highlighted with a mouse cursor), 'View as plain text', 'Edit Build Information', 'Delete build '#14'', 'Environment Variables', 'Open Blue Ocean', and 'Previous Build'. The main area displays the title 'Console Output' with a red 'X' icon. The output text is as follows:

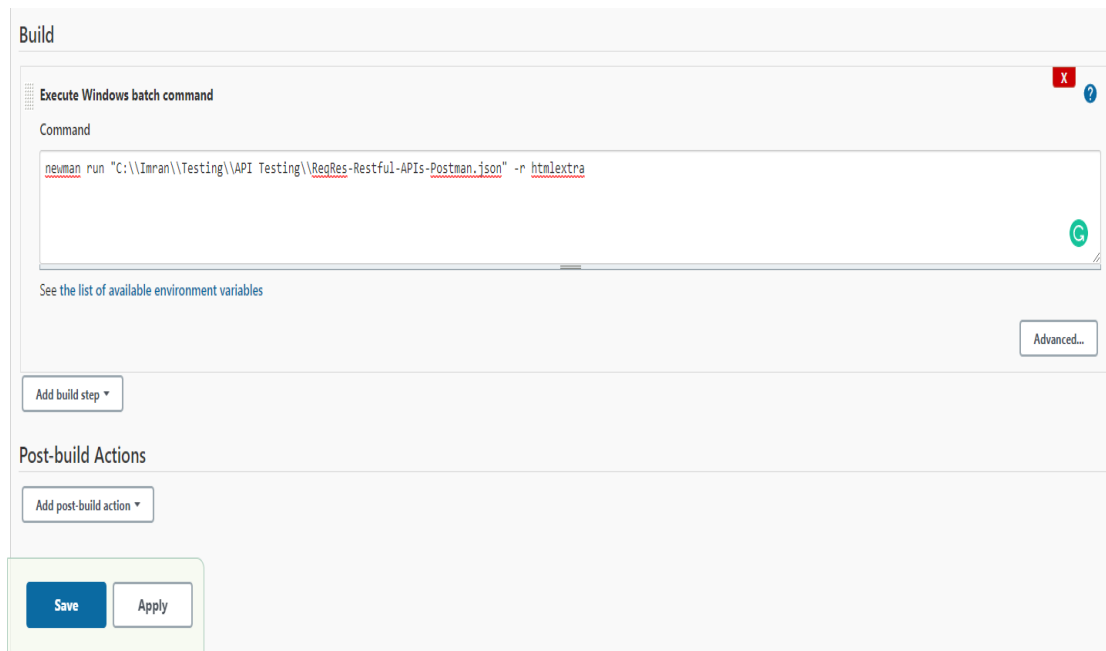
```
Started by user Imran Al Munyeeen
Running as SYSTEM
[EnvInject] - Loading node environment variables.
Building in workspace C:\Users\ialmu\.jenkins\workspace\ReqRes-Restful-APIs-Postman
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\Users\ialmu\.jenkins\workspace\ReqRes-Restful-APIs-Postman\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/imranalmunyeeen/ReqRes-Restful-APIs-Postman # timeout=10
Fetching upstream changes from https://github.com/imranalmunyeeen/ReqRes-Restful-APIs-Postman
> git.exe --version # timeout=10
> git --version # 'git version 2.36.0.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/imranalmunyeeen/ReqRes-Restful-APIs-Postman +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse 'refs/remotes/origin/master^{commit}' # timeout=10
> git.exe rev-parse 'origin/master^{commit}' # timeout=10
ERROR: Couldn't find any revision to build. Verify the repository and branch configuration for this job.
Finished: FAILURE
```

## Automate Newman from Jenkins:

**Steps** --- Just put newman command in windows batch command field. Rest of the steps will remain same like previous.

**N.B:** You can either put your collection file from device with location or you can also put collection link from the postman.

Now Jenkins will automatically run newman and also generate report after building the job. You can see the result just like you did before in previous steps.



The screenshot shows the Jenkins 'Build' configuration page. The main section is 'Execute Windows batch command'. The 'Command' field contains the following text: `newman run "C:\\Imran\\Testing\\API Testing\\ReqRes-Restful-APIs-Postman.json" -r htmlextra`. Below the command field is a link that says 'See the list of available environment variables'. To the right of the command field is an 'Advanced...' button. Below the 'Execute Windows batch command' section is a section for 'Post-build Actions' with an 'Add post-build action' button. At the bottom of the page are 'Save' and 'Apply' buttons.

## FAQs:

### 1. What is Postman?

Postman is an API platform for developers to design, build, test and iterate their APIs.

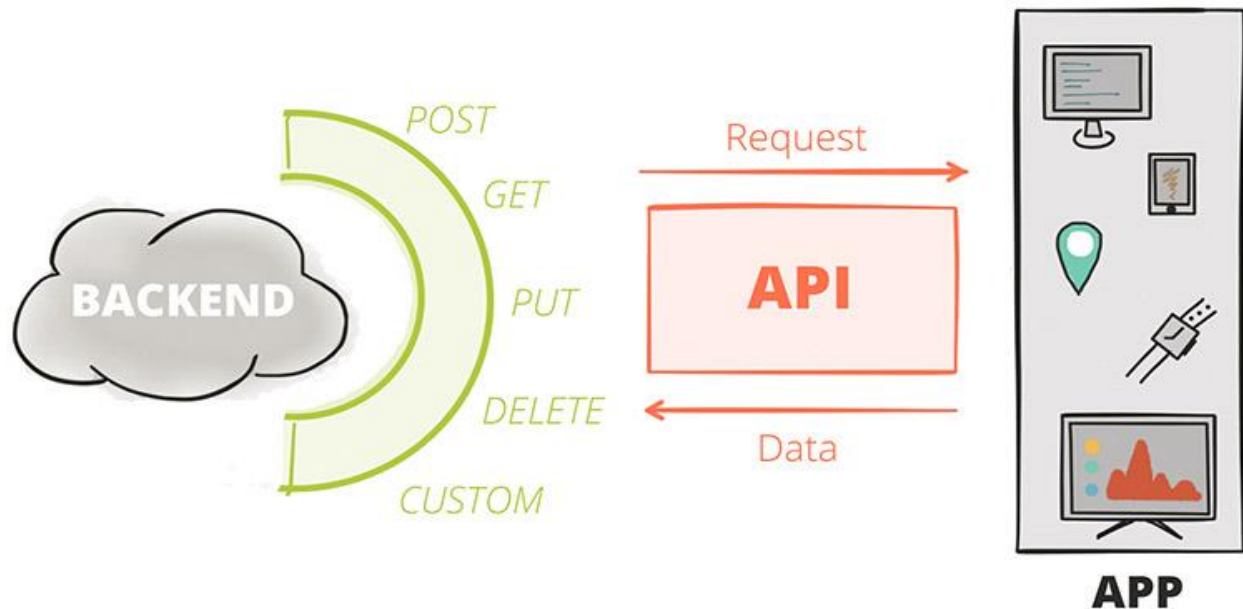
Using the Postman tool, we can send HTTP/s requests to a service, as well as get their responses. By doing this we can make sure that the service is up and running.

### 2. Why Postman?

- **Free:** It is free to download and use for teams of any size.
- **APIs Support:** You can make any kind of API call (REST, SOAP, or plain HTTP) and easily inspect even the largest responses.
- **Extensible:** You can customize it for your needs with the Postman API.
- **Integration:** You can easily integrate test suites into your preferred CI/CD service with Newman (command line collection runner)

### 3. What is an API?

Application Programming Interface (API) is a connector between user and database. When a user sends a request, API receives it, try to fetch the responses from database and provide responses.



### 4. What are the core components of an HTTP request?

An HTTP request includes five key elements:

- HTTP methods – Set of request methods to perform desired action for a given resource (GET, PUT, PATCH, POST, DELETE)
- Uniform Resource Identifier (URI) – Describes the resource
- HTTP Version, (example- HTTP v1.1)
- Request Headers, (example- Content-type : application/json, Content-Length : 511)
- Payload – It is basically a Request Body which includes message content.



## **5. State the Core Components of an HTTP Response?**

Every HTTP response contains four key elements.

- Status/Response Code – These are response codes issued by a server to a client's request. For example, 404 means Page Not Found, and 200 means Response is OK.
- HTTP Version – describes HTTP version, for example-HTTP v1.1.
- Response Header – Includes information for the HTTP response message. For example, Content-type, Content-length, date, status and server type.
- Response Body – It contains the data that was requested by a client to see

## **6. What are the HTTP Response codes?**

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

- Informational responses (100–199)
- Successful responses (200–299)
- Redirection messages (300–399)
- Client error responses (400–499)
- Server error responses (500–599)

## 7. Explain the HTTP Response codes with their types?

<b>1XX Informational</b>		<b>4XX Client Error Continued</b>	
<b>100</b>	Continue	<b>409</b>	Conflict
<b>101</b>	Switching Protocols	<b>410</b>	Gone
<b>102</b>	Processing	<b>411</b>	Length Required
<b>2XX Success</b>		<b>412</b>	Precondition Failed
<b>200</b>	OK	<b>413</b>	Payload Too Large
<b>201</b>	Created	<b>414</b>	Request-URI Too Long
<b>202</b>	Accepted	<b>415</b>	Unsupported Media Type
<b>203</b>	Non-authoritative Information	<b>416</b>	Requested Range Not Satisfiable
<b>204</b>	No Content	<b>417</b>	Expectation Failed
<b>205</b>	Reset Content	<b>418</b>	I'm a teapot
<b>206</b>	Partial Content	<b>421</b>	Misdirected Request
<b>207</b>	Multi-Status	<b>422</b>	Unprocessable Entity
<b>208</b>	Already Reported	<b>423</b>	Locked
<b>226</b>	IM Used	<b>424</b>	Failed Dependency
<b>3XX Redirection</b>		<b>426</b>	Upgrade Required
<b>300</b>	Multiple Choices	<b>428</b>	Precondition Required
<b>301</b>	Moved Permanently	<b>429</b>	Too Many Requests
<b>302</b>	Found	<b>431</b>	Request Header Fields Too Large
<b>303</b>	See Other	<b>444</b>	Connection Closed Without Response
<b>304</b>	Not Modified	<b>451</b>	Unavailable For Legal Reasons
<b>305</b>	Use Proxy	<b>499</b>	Client Closed Request
<b>307</b>	Temporary Redirect	<b>5XX Server Error</b>	
<b>308</b>	Permanent Redirect	<b>500</b>	Internal Server Error
<b>4XX Client Error</b>		<b>501</b>	Not Implemented
<b>400</b>	Bad Request	<b>502</b>	Bad Gateway
<b>401</b>	Unauthorized	<b>503</b>	Service Unavailable
<b>402</b>	Payment Required	<b>504</b>	Gateway Timeout
<b>403</b>	Forbidden	<b>505</b>	HTTP Version Not Supported
<b>404</b>	Not Found	<b>506</b>	Variant Also Negotiates
<b>405</b>	Method Not Allowed	<b>507</b>	Insufficient Storage
<b>406</b>	Not Acceptable	<b>508</b>	Loop Detected
<b>407</b>	Proxy Authentication Required	<b>510</b>	Not Extended
<b>408</b>	Request Timeout	<b>511</b>	Network Authentication Required
		<b>599</b>	Network Connect Timeout Error
<b>HTTP STATUS CODES</b>			
When a browser requests a service from a web server, an error may occur.			
This is a list of HTTP status messages that might be returned.			

**8. What API information is exposed in Web Developer tools?**

Request headers, Response body, Response cookies

**9. What can we use to get API information from web developer tools into Postman?**

Copy as cURL can get API information from web developer tools into Postman.

**10. In which type of encoding does postman accept authorization credentials?**

Postman accepts Base64 encoding only. This is provided inbuilt in postman or else you can also refer 3rd party websites to convert the credentials in base64.

**11. Why does Postman accept Base64 encoding only?**

We use base64 particularly because it transmits the data into the textual form and sends it in easier form such as HTML form data. Also, we can rely on the same 64 characters in any encoding language that we use.

**12. What is meant by the term environment in postman?**

An environment in postman is a set of key value pairs. You can create multiple environments in postman which can be switched quickly with a press of a button. There are 2 types of environments, global and local.

**13. Can global scope variables have duplicate names in postman?**

Since global variables are global i.e., without any environment, global variables cannot have duplicate names. Local variables can have the same name but in different environments.

**14. Which one will be preferred in postman- a global variable or a local variable?**

In postman, if 2 variables have the same name (one being local, other global) then the higher priority is of the local variable. it will overwrite the global variable.

**15. What is a Postman Collection?**

A Postman Collection lets us group individual requests together. Simply it allows us to organize the requests into folders.

**16. What do you mean by postman monitors?**

The postman monitor is used for running collections. Collections are run till specified time defined by the user. Postman Monitor requires the user to be logged in. Monitor reports are shared by users over email on a daily/monthly basis.

**17. What do you understand by the term Postman Collection runners?**

A postman collection runner is used to perform Data-driven testing. The group of API requests are run in a collection for the multiple iterations with different sets of data.

**18. Can local variables be imported in Postman Monitors?**

Yes. Postman monitors allow to import local variables but it does not allow to import global variables.

**19. What is the purpose of Postman cloud if we are working in a company? Why?**

A Postman cloud is a common repository of companies to access Postman collections. In Postman cloud, work can be saved instantly after logging in. Anyone from the team

can access data/collections from anywhere.

## **20. Why is it not preferred to save work in Postman cloud?**

It is not preferred to save your work in Postman cloud as company's work is not allowed to be leaked and remain confidential. Security breaches can be experienced if Postman cloud is used as Postman cloud requires sign in. Therefore, Postman Cloud is discouraged for saving work and team workspace is highly encouraged.

## **21. When do we use global variables, collection variables, and local variables?**

**Global variables** are general purpose variables, ideal for quick results, and prototyping. They are used while passing data to other requests.

**Collection variables** can be mostly used for storing some constants that do not change during the execution of the collection. They are used for constants that do not change during the execution and also for URLs / authentication credentials if only one environment exists.

**Local variables** are only available within the request that has set them or when using Newman/Collection runner during the entire execution. They are used whenever you would like to override all other variable scopes.

## **22. How do you remove local variables?**

Local variables are automatically removed once the tests have been executed.

## **23. How can we stop executing requests or stop the collection run?**

```
postman.setNextRequest(null);
```

**24. What is the difference between form data and x-www-form-urlencoded ?**

The difference between the form data and x-www-form-urlencoded is that the url will be encoded when sent through x-www-form-urlencoded.

**25. Where are query parameters stored in a GET request?**

Query parameters are stored in the URL in a GET request.

**26. How can we access a Postman variable?**

We can access a Postman variable by entering the variable name as {{var}}

**27. What is the HTTP response code for a POST request with incorrect parameters?**

400 Bad Request is an ideal response code for request with incorrect parameters.

**28. How can you iterate a request 100 times in Postman?**

By using Collection Runner.

**31. How can we organize requests in Postman?**

We can organize requests in Postman with the Collections.

**32. Which programming language is used for Postman tests?**

JavaScript.

**33. What will execute first in a Collection Run?**

Pre-request scripts at the Collection level are executed first in a Collection run.

**34. What are some of the JS libraries available in Postman?**

Lodash, Moment, GUID

**35. Which tool can be used to run Postman Collections in Jenkins?**

Newman can be used.

**36. How can we log requests and responses in Postman?**

We can view requests logs and response logs through the Postman Console window.

**37. What is GUID?**

GUID stands for Global Unique Identifier. It is basically hexadecimal digits separated by hyphens. GUID solves the purpose of uniqueness.

In Postman, we use this to generate and send a random value to APIs.