Table of Content

[**1.** **POSTMAN BASICS** 1](#_Toc513818699)

[**1.1** **Introduction** 1](#_Toc513818700)

[**1.2** **Installation of the Postman app** 1](#_Toc513818701)

[**1.3** **Sign Up using Postman App** 1](#_Toc513818702)

[**1.4** **Postman Collection and Folders** 2](#_Toc513818703)

[**1.5** **Postman Requests and Responses** 4](#_Toc513818704)

[**1.6** **POSTMAN VARIABLES** 9](#_Toc513818705)

[**1.7** **Importing and Exporting** 12](#_Toc513818706)

[**2.** **TEST SCRIPTS** 15](#_Toc513818707)

[**2.1** **Basic Introduction to Test and Test Scripts** 15](#_Toc513818708)

[**2.3** **Snippets** 18](#_Toc513818709)

[**2.4** **Relating JS Codes** 18](#_Toc513818710)

[**2.6** **Passing/Using variables** 21](#_Toc513818711)

[**2.7** **Using Multiple Request in One Test** 23](#_Toc513818712)

[**2.8** **Checking Test Response and Error Debugging** 24](#_Toc513818713)

[**3.** **Collection Runner:** 25](#_Toc513818714)

[**3.1** **Starting the Collection Runner** 26](#_Toc513818715)

[**3.2** **Choose a Collection or Folder:** 27](#_Toc513818716)

[**3.3** **Environment:** 27](#_Toc513818717)

[**3.4** **Iterations:** 27](#_Toc513818718)

[**3.5** **Delay:** 27](#_Toc513818719)

[**3.6** **Log Responses:** 27](#_Toc513818720)

[**3.7** **Data:** 28](#_Toc513818721)

[**3.8** **Persist variables:** 28](#_Toc513818722)

[**3.9** **Checking and Debugging the Collection Runner Results:** 28](#_Toc513818723)

[**4.** **Data File** 29](#_Toc513818724)

[**4.1** **Creating a Data File** 30](#_Toc513818725)

[**4.2** **Using Data files in Collection Runner:** 32](#_Toc513818726)

[**5.** **Running the Collection in Command prompt and Report Generation** 33](#_Toc513818727)

[**5.1** **Introduction to Newman** 33](#_Toc513818728)

[**5.2** **Newman Setup:** 33](#_Toc513818729)

[**5.3** **Running Collection in Newman** 34](#_Toc513818730)

[**5.4** **Viewing Newman Results** 35](#_Toc513818731)

# **POSTMAN BASICS**

## **Introduction**

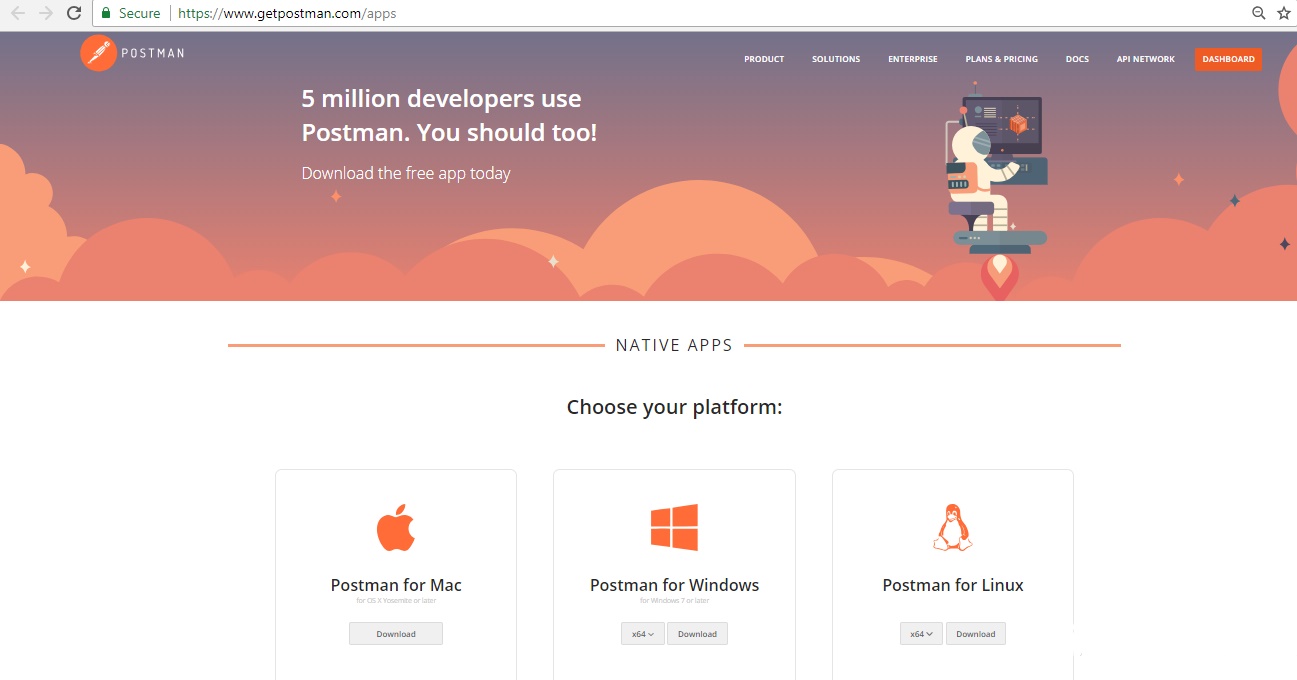
Postman is a powerful HTTP client for testing web services. Created by Abhinav Asthana, a programmer and designer based in Bangalore, India, Postman makes it easy to test, develop and document APIs by allowing users to quickly put together both simple and complex HTTP requests. Postman is a great tool when trying to dissect RESTful APIs made by others or test ones you have made yourself. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.

Speaking through the personal experience, Postman can be called as a very useful tool for effectively and efficiently testing the web API in no time. Testers/programmer can totally rely on this tool for the regression testing if used and maintained in a correct manner, every time the API gets deployed. It saves so much of time and creates a great platform to test a multiple case on a single API.

## **Installation of the Postman app**

Postman is available as a native app for Mac, Windows, and Linux operating systems.

To install Postman, go to the apps page and click Download for Mac / Windows / Linux depending on your platform.

****

We recommend using the Postman native apps, but Postman is also available as a Chrome app which can only run on the Chrome browser.

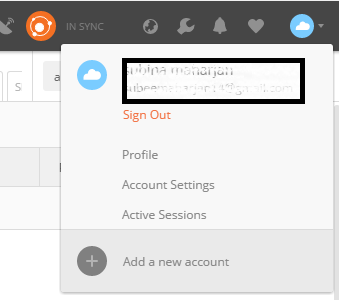
## **Sign Up using Postman App**

After you have downloaded the Postman app, launch the app and you can see a prompt to log in or sign up.

Sign up with your email address or your Google account.

When you first open the Postman app, you can sign in as an existing user or create an account. If you bypass the initial sign-in option, you can click the Sign In button on the top right corner of your screen at any time.

You can also use multiple login through **Add a new account** button at the top right corner of the screen.



Once you have successfully signed up with a Gmail account, now all you have to understand is about the Postman Components. Before that, you need to know about the Syncing feature of the Postman app. **Syncing** is the feature that makes your all the data and components available on any device when you are signed in with the same postman account.

## **Postman Collection and Folders**

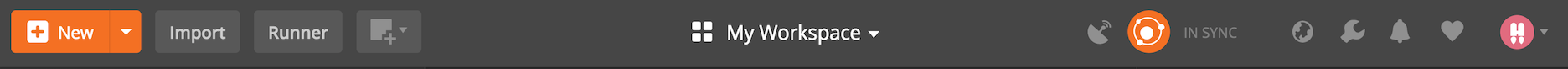
Postman Collections are a group of saved requests you can organize into folders.

Every request you send in Postman appears under the History tab of the sidebar. On a small scale, reusing requests through the history section is convenient. However, as your Postman usage scales, it can be time consuming to find a particular request in your history. Instead of combing through your history section, you can save all your requests as a group for easier access.

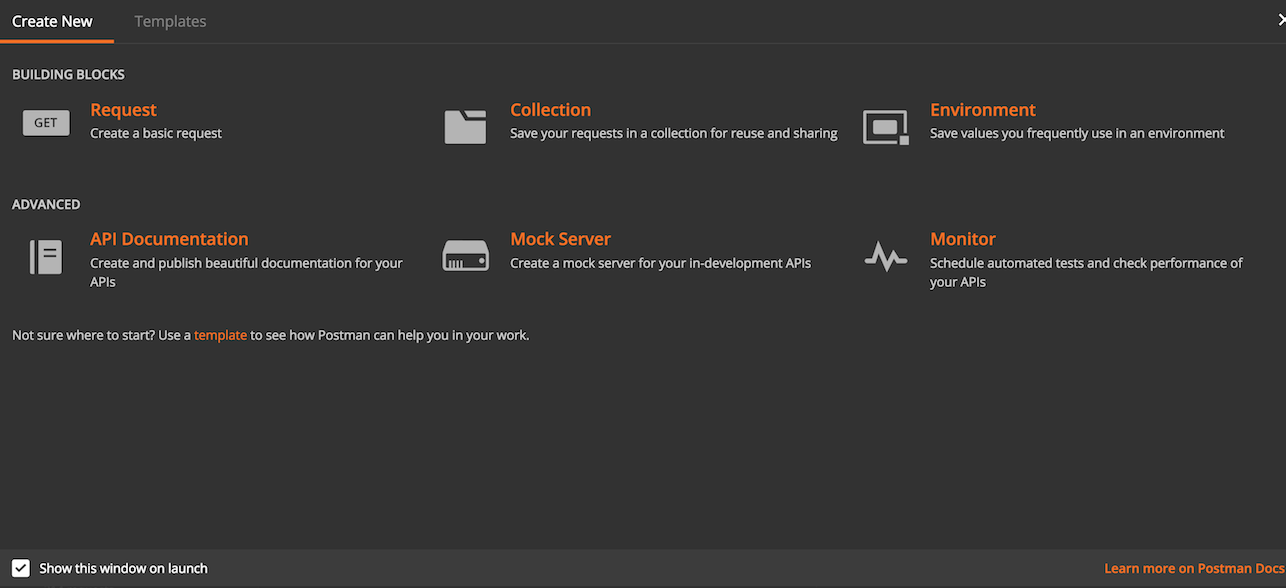
Creating workflows is important for a good organization, documentation, test suites and conditional workflows

**How to Create a Collection:**

In the header toolbar, click the New button.

****

The Create New tab appears.

****

In the Create New tab, click “Collection” to create new Collection and click “Request” to create new Request.

In the **CREATE A NEW COLLECTION** modal:

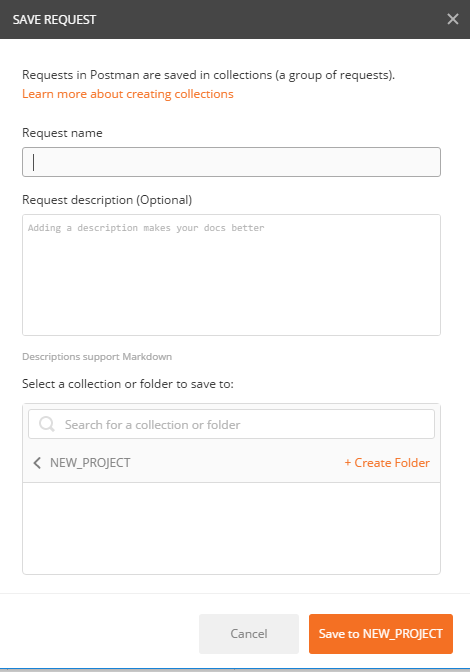
* Enter a name and optional description.
* Select an authorization type.

Click the **Create** button.

After creating the collection, you can save requests to the collection and add folders for better organization.

## **Postman Requests and Responses**

After your collection has been created, It will give you an option to add request right below the collection you have just created. So just Click **Add requests** link to create a new request.



Add a request name as you would like and select a collection or folder to save to and click **Save to Project Name** button. Description field is optional.

Before going inside the how to send a request, it is important to know about its few components.

* URL FIELD
* REQUEST METHODS
* AUTHORIZATION
* HEADER
* BODY
* PRE-REQUEST SCRIPT
* TESTS

**However, the four parts of an HTTP request are the URL, method, headers, and the body.**

**URL**: You need to enter the URL of the API you want to send request to, inside the URL field.

After the URL, you need to specify the **Request methods** for that URL. There could be multiple Request methods but one API request needs only one request method. Let’s know some more about request methods.

* **GET**

GET is one of the most common HTTP methods.

GET is used to request data from a specified resource.

* **POST**

The data sent to the server with POST is stored in the request body of the HTTP request.

* **PUT**

The difference between POST and PUT is that PUT requests are idempotent. That is, calling the same PUT request multiple times will always produce the same result. In contrast, calling a POST request repeatedly make have side effects of creating the same resource multiple times.

* **DELETE**

The DELETE method deletes the specified resource.

So these shall be enough for the Request method concepts for you.

**HEADERS**

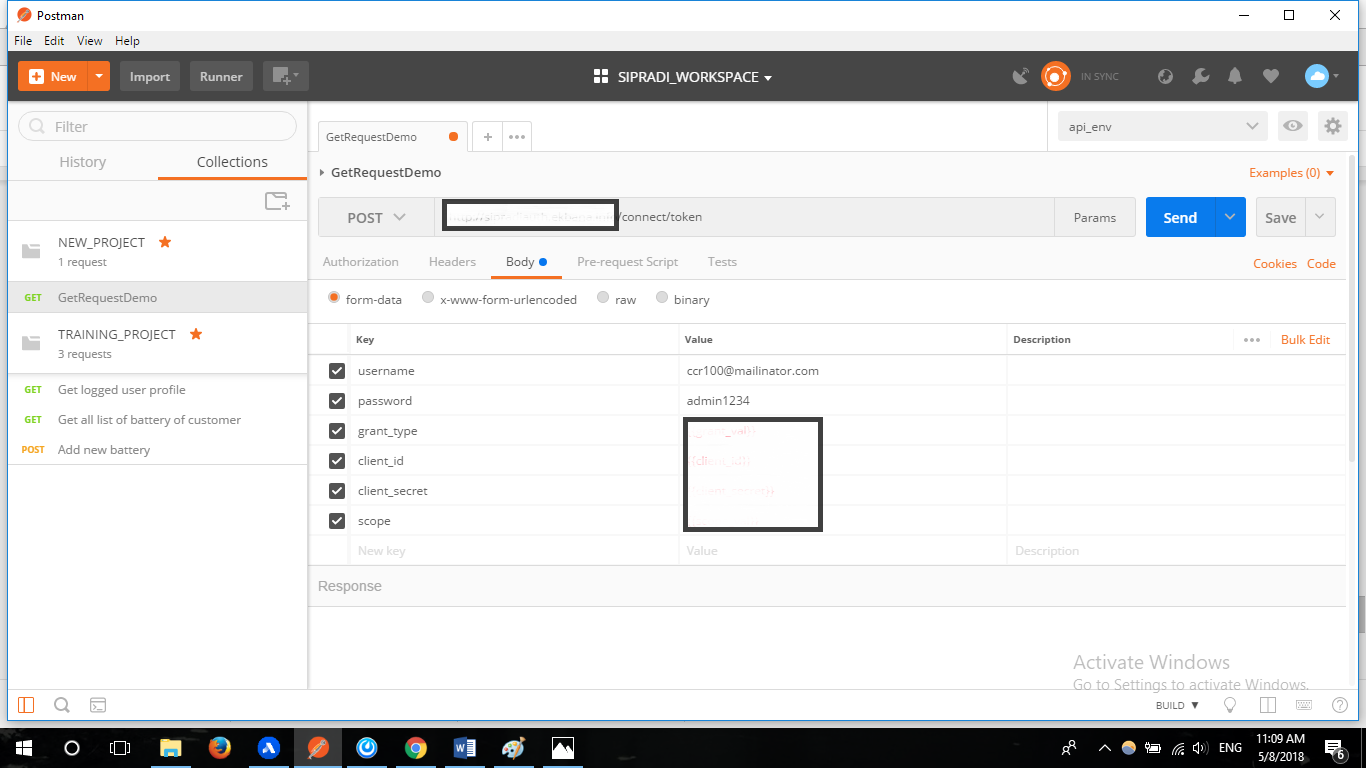
You can add the Key to send to your API request under the **Headers** Presets. Commonly used Presets is the Authorization. Other could be **Grant-type**, **Password** and **Username** as per the API’s requirement.

**BODY**

While constructing requests, you’ll work frequently with the request body editor. Postman lets you send almost any kind of HTTP request. The body editor is divided into 4 areas and has different controls, depending on the body type which are form-data, URL encoded, raw and binary. But going to focus more on Form data and Raw panel as We will be using them more often.

**Form data**

Form-data is the default encoding a web form uses to transfer data. This simulates filling a form on a website, and submitting it. The form-data editor lets you set key-value pairs.



This is the example of key values entered inside the form data. These data could also be inserted through the Bulk Edit option or one by one from the displayed field.

**RAW**

A raw request can contain anything. Postman doesn’t touch the string entered in the raw editor except replacing [environment variables](https://www.getpostman.com/docs/v6/postman/environments_and_globals/variables). Whatever you put in the text area gets sent with the request. The raw editor lets you set the formatting type along with the correct header that you should send with the raw body. You can set the Content-Type header manually too and this will override the Postman defined setting. Selecting XML/JSON in the editor type enables syntax highlighting for your request body and also sets the Content-Type header.



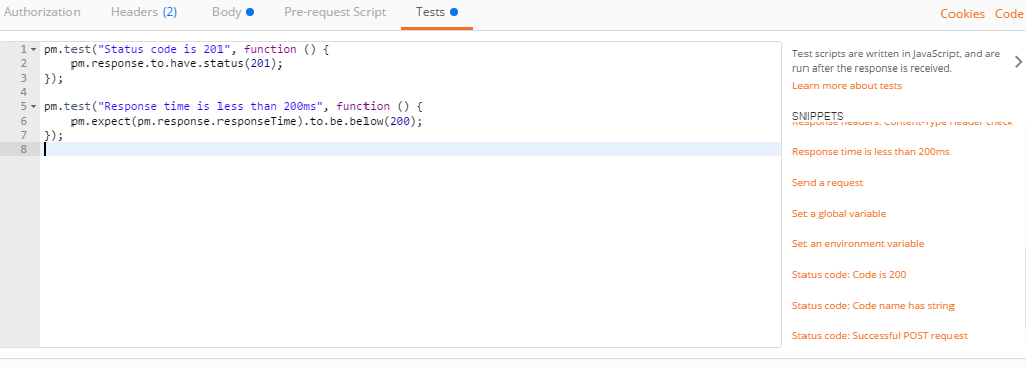
**Pre-request Script**

Pre-request scripts are snippets of code associated with a collection request that are executed before the request is sent.

**Test Scripts**

A Postman test is essentially JavaScript code executed after the request is sent, allowing access to the pm.response object. You can add the readymade js codes of the snippets available at the right side of the **test** board or add your own js codes to make various tests.

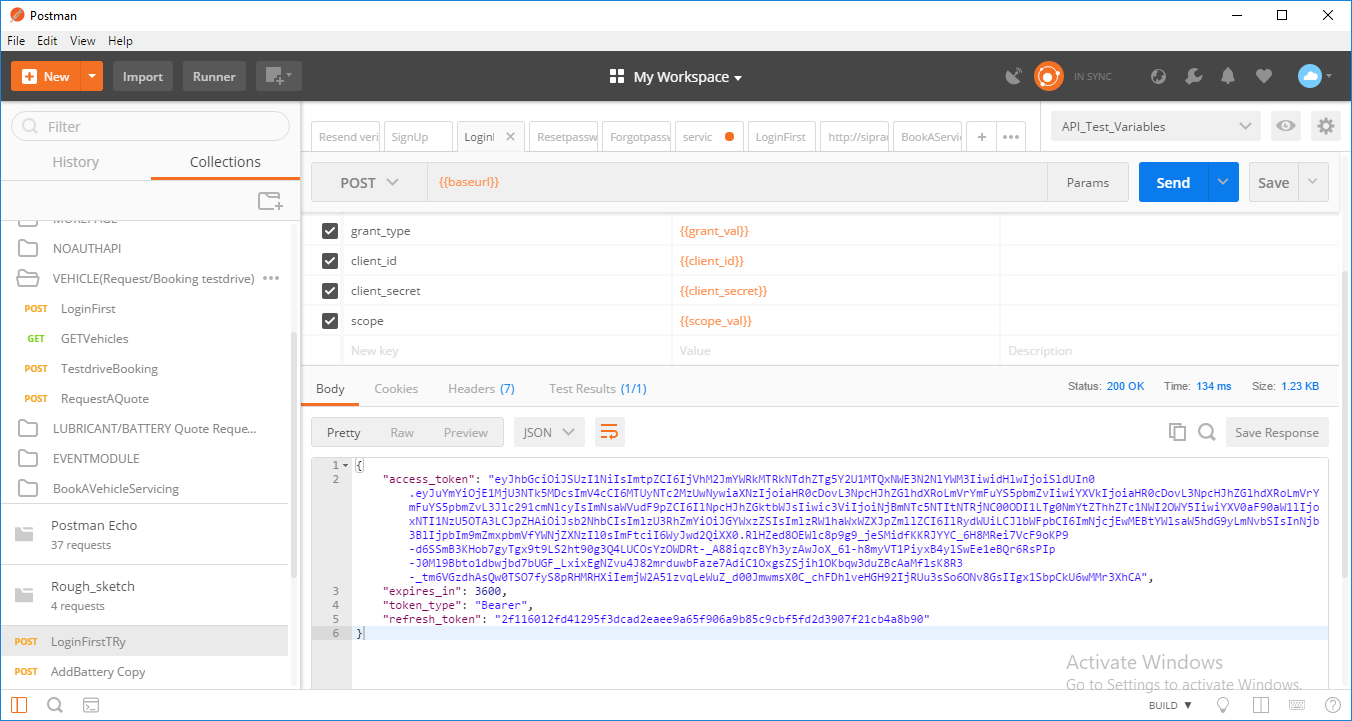
A simple example of Test Scripts is shown below.



**TEST RESPONSES:**

Test response also has few fields/components for you to understand about:

* Body
* Headers
* Test Results
* Status
* Time
* Size

****

The **body** field contains all the content provided by the API in response.The specified Request in the image has requested for the access token in the result, hence has been provided with it.

**Headers** are displayed as key-value pairs under the **Headers** Tab. Hovering over the header name can give you a description of the header according to the HTTP spec. If you are sending a HEAD request, Postman will show the headers tab by default.

**Test Result** tab shows the number of passed or failed results as per the test scripts (js scripts) you have used before while sending the Request inside the Test box.

**Status code** Status code in the response menu informs about the status of the response body. Different type of response code has a different number like that 20. You just need to know the meaning of those codes.

Few of those response codes are listed below.

200(OK)

201(CREATED)

202(ACCEPTED)

204(NO CONTENT)

302(FOUND)

400(BAD REQUEST)

401(UNAUTHORIZED)

403(FORBIDDEN)

404(NOT FOUND)

405(METHOD NOT ALLOWED)

412(PRECONDITION FAILED)

415(UNSUPPORTED MEDIA TYPE)

500(INTERNAL SERVER ERROR)

Through these status codes of these response, you can know the condition of the response body.

**Response time** Postman automatically calculates the time it took for the response to arrive from the server. This is useful for some preliminary testing for performance.

**Response size** Postman breaks down the response size into body and headers. The response sizes are approximate.

## **POSTMAN VARIABLES**

Variables allow you to reuse values in multiple places so you can keep your code DRY (Don’t Repeat Yourself). Also, if you want to change the value, you can change the variable once with the impact cascading through the rest of your code.

The following scopes are available to you:

1. Global
2. Collection
3. Environment

1. **Global Variable**

Global Variables are the one which can be accessed by every collection available in the [Workspaces](https://www.getpostman.com/docs/v6/postman/workspaces/intro_to_workspaces). You can set global variables for such kind of data which is required in almost every collection and is not frequently changing.

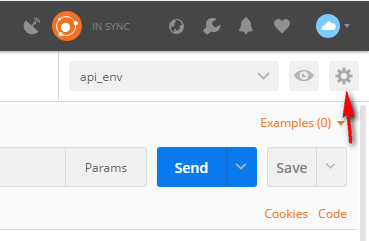
**2. Environment Variable**

An **environment** is a set of key-value pairs. The key represents the name of the **variable**. ... Environments let you customize requests using **variables** so you can easily switch between different setups without changing your requests. You won't have to remember all those values once they are in **Postman**.Environment Variable are the variables saved inside one Environment folder.

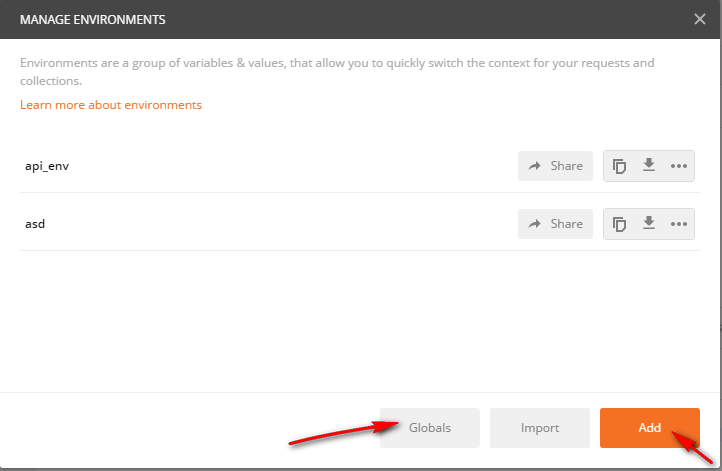
One Environment can be used for multiple collections and Multiple Environment can also be used for A single collection. You just need to make sure that you have selected the Environment which has values you require.

**How to create Global and Environment variables:**

Click on Manage Environment Icon on the top right of the screen.

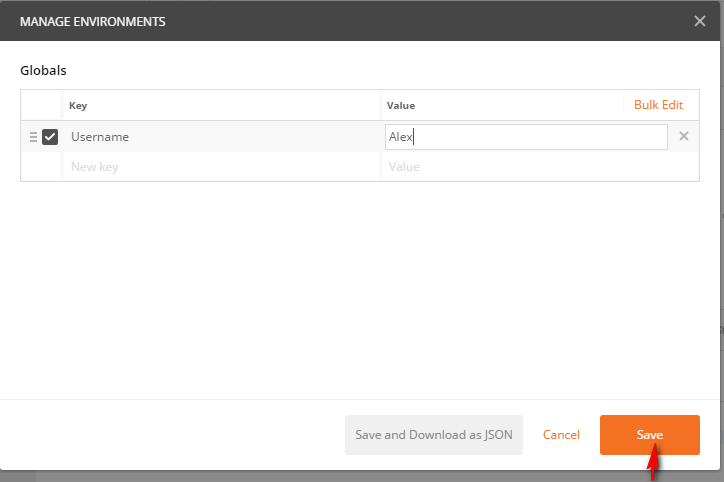


Once you Click this icon, you will see a form to add environment or global variables like below.



In this form, the right pointed **Add** button allows you to add environment variables and Left pointed **Global** Button allows you to add global variables.

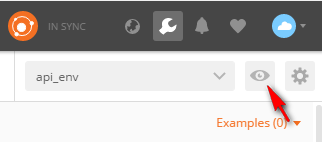
Any two button will give you a doubled columned box to add variables and its values.The box to fill up the key and values will look like below.



So in the Box like this, add a meaningful key to input the values into, and also insert the value on the second column too.

After adding all the required number of variables, click **Save**. Now you can access those variables in the body field, URL field, Authorization field and the Test field either.

You can visualize the saved Global and environment variable from the **Environment quick look** option available at the right side of the manage environment option at the top right corner of the page.



This quick look icon will show you all the global and environment variables that you have defined.

After this, we have one more variable, Collection variable which is the variable defined inside a specific collection, only for that collection to use. The collection variable can be defined at the time of creation of the collection. If not, then it can be done through the right arrow icon available on every collection available.

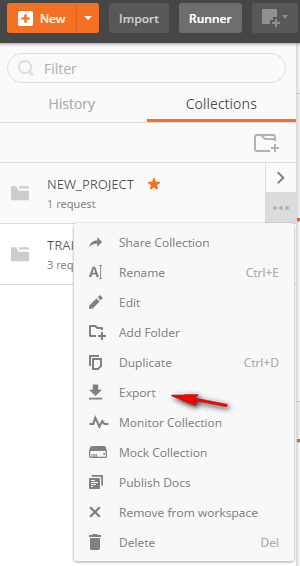
For detail information about the [variables](https://www.getpostman.com/docs/v6/postman/environments_and_globals/variables) , you can also go through the postman documentation.

## **Importing and Exporting**

As we already know that the collection and all the tests we have created in the postman could be synced into different devices by using the same Email. And those collections also could be shared to the team by using the premium version of postman. But what we we can do in this version to share the work we have created is to export and import the collection file or environment variable files . This also gives the way of sharing in some other way. Once the collection has been exported as a file, it could be used for desired number of times by desired number of user by using any email id simply by importing those files and folders into their postman app.

**Exporting the collection:**

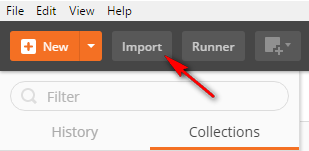
To export a collection, click the export option that appears when you click the right icon over the collection as shown in the image below.



You can then save the generated file wherever you want.

**Importing the collection:**

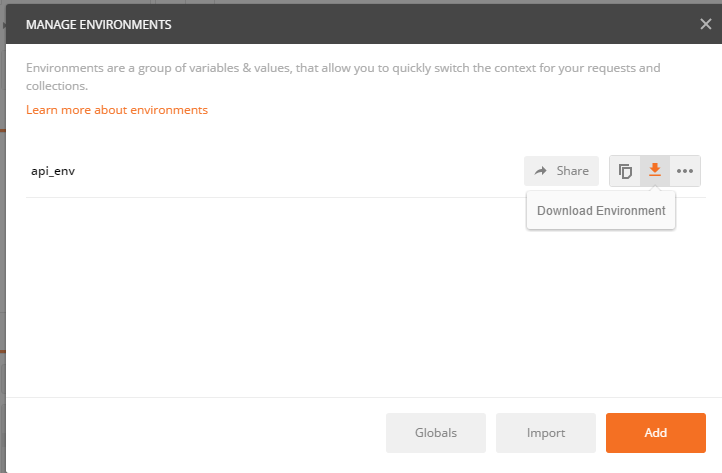
To import a collection, click the import option that is available on the task bar as shown in the image below.



You can Import through the folder, file or even the link.

**Exporting the Environment:**

To import the Environment variable, just click the download option available right beside the environment inside the manage environment dialog as shown below.



**Importing the Environment:**

The import option is available on the same dialog which is used for Exporting the environment variable as “**Import**” button.

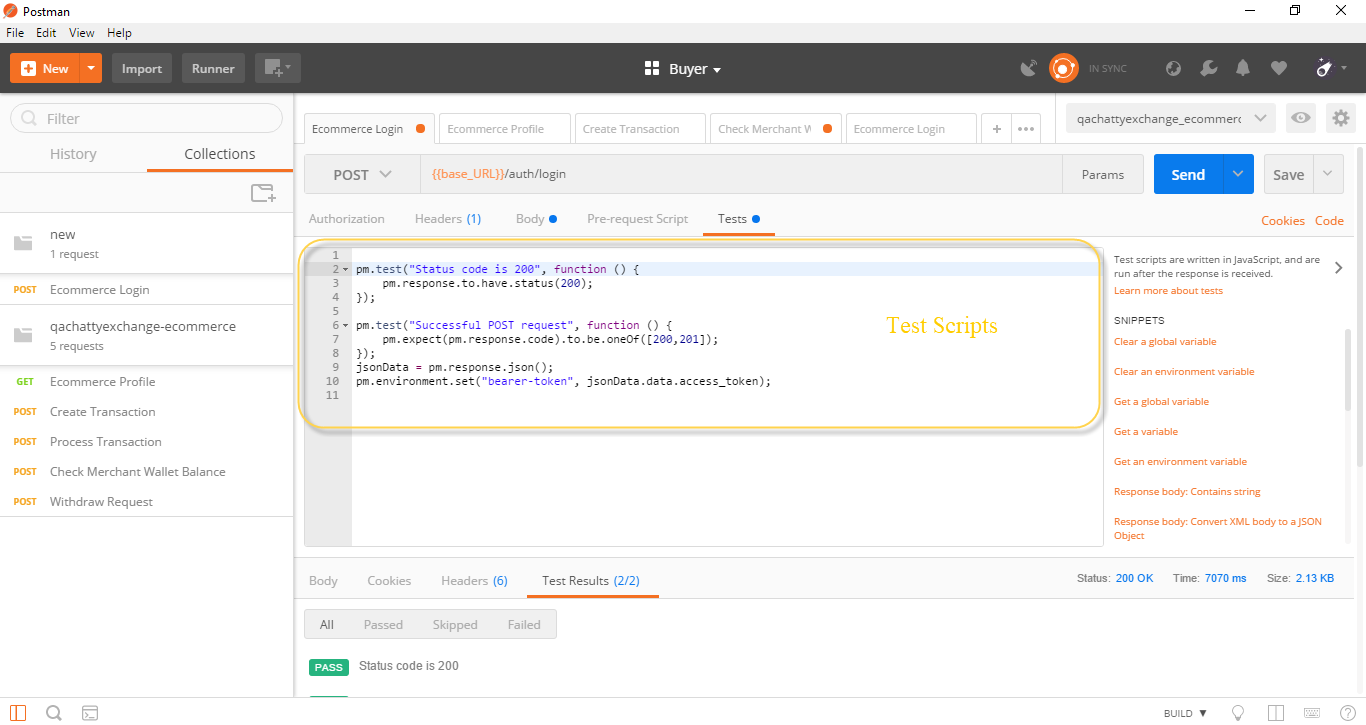
# **TEST SCRIPTS**

## **Basic Introduction to Test and Test Scripts**

A postman test is essentially JavaScript code executed after the request is sent, allowing to the pm.response object.

A postman test scripts in a block of JavaScript code which allows you to write and run the tests for each request.

For more details <https://www.getpostman.com/docs/v5/postman/scripts/test_scripts>



* 1. **Accessing Data from Response Body**

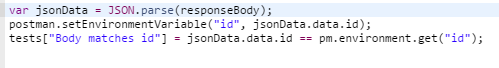
In the postman, we can access data from the response body to ensure that the test output is valid data or not and also to set and get variables to ease the further test.

The data in the response body may be in normal JSON or in array format. The JavaScript code written to access for normal format will not work to access the data in the array format. So here is the code how to access the both data format.

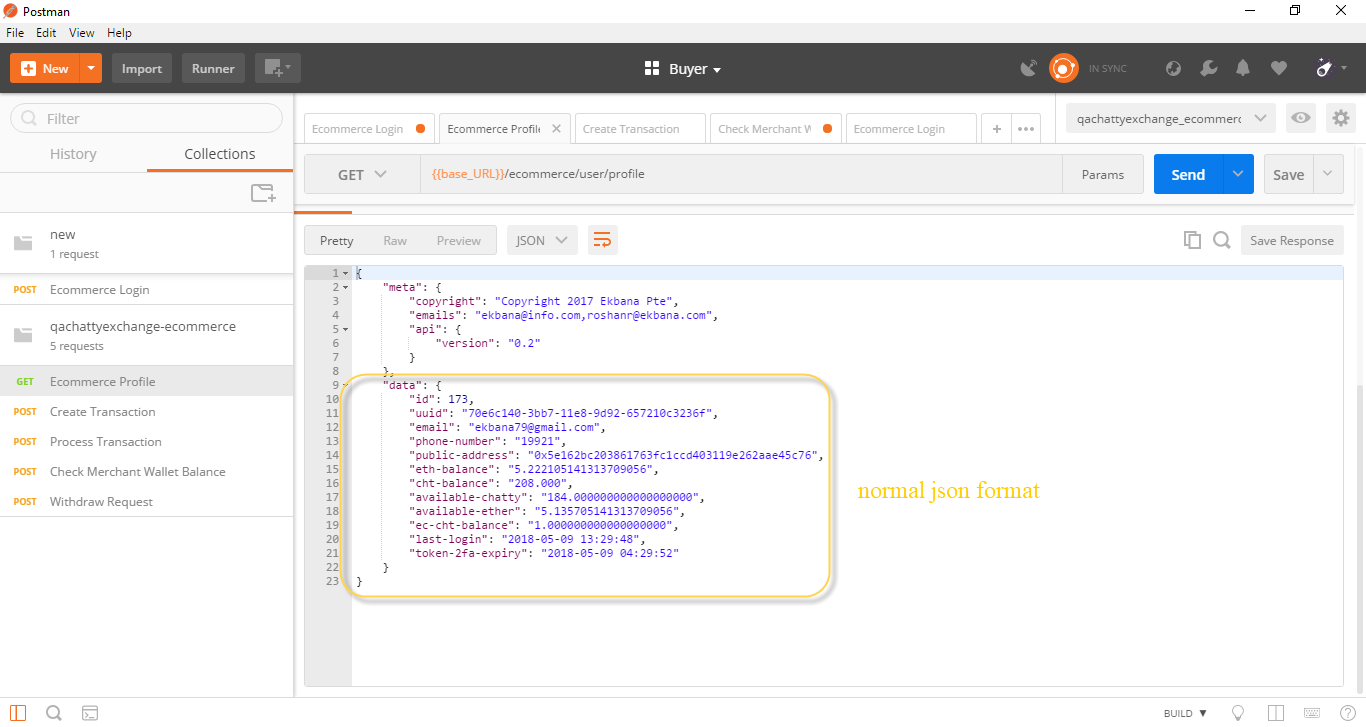
1. **Normal JSON Format Data**

Normal JSON format data can be seen enclosed in the {} braces.

How to access?



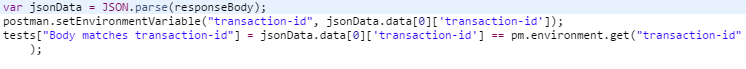
Here, we are trying to set the id into environment variable accessing data from response body. So the id is inside the data in response body in {} braces. So the code to access the id is jsonData.data.id (reference above figure).



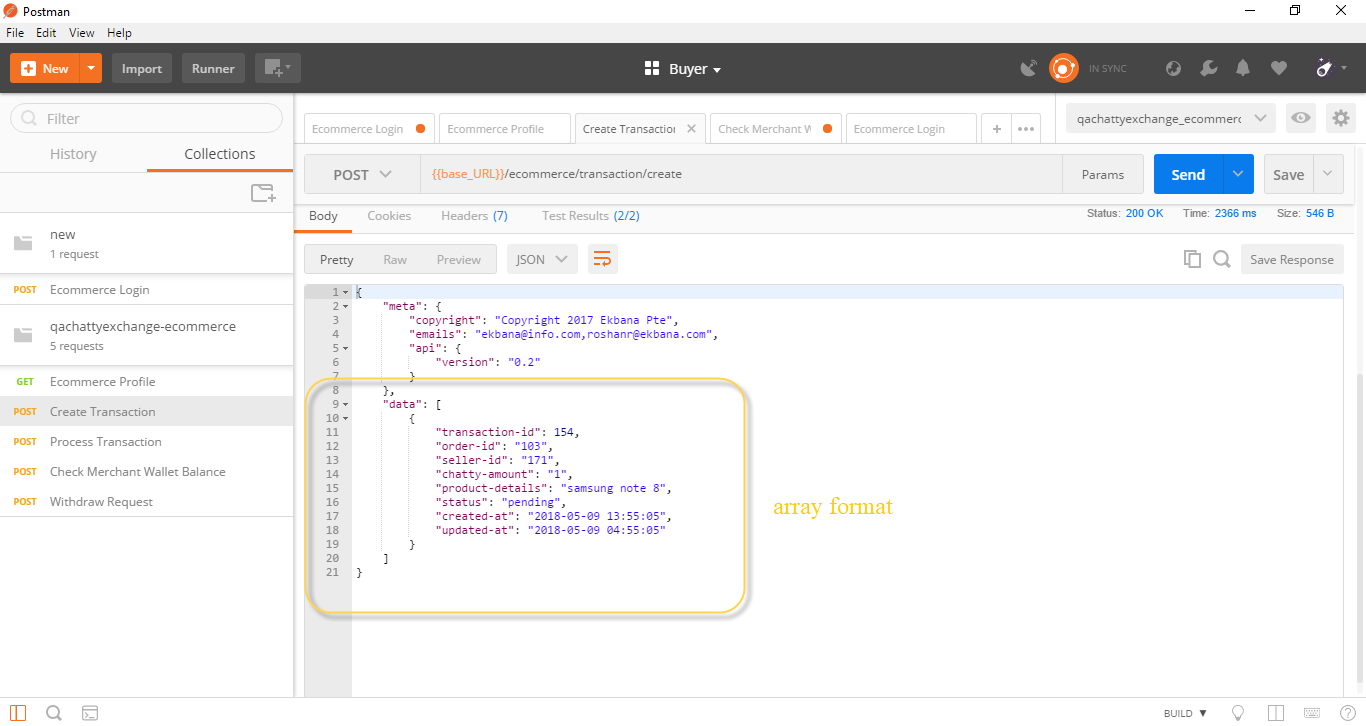
1. **Array Format Data**

Array format data can be seen enclosed in [ ] bracket.

How to access?



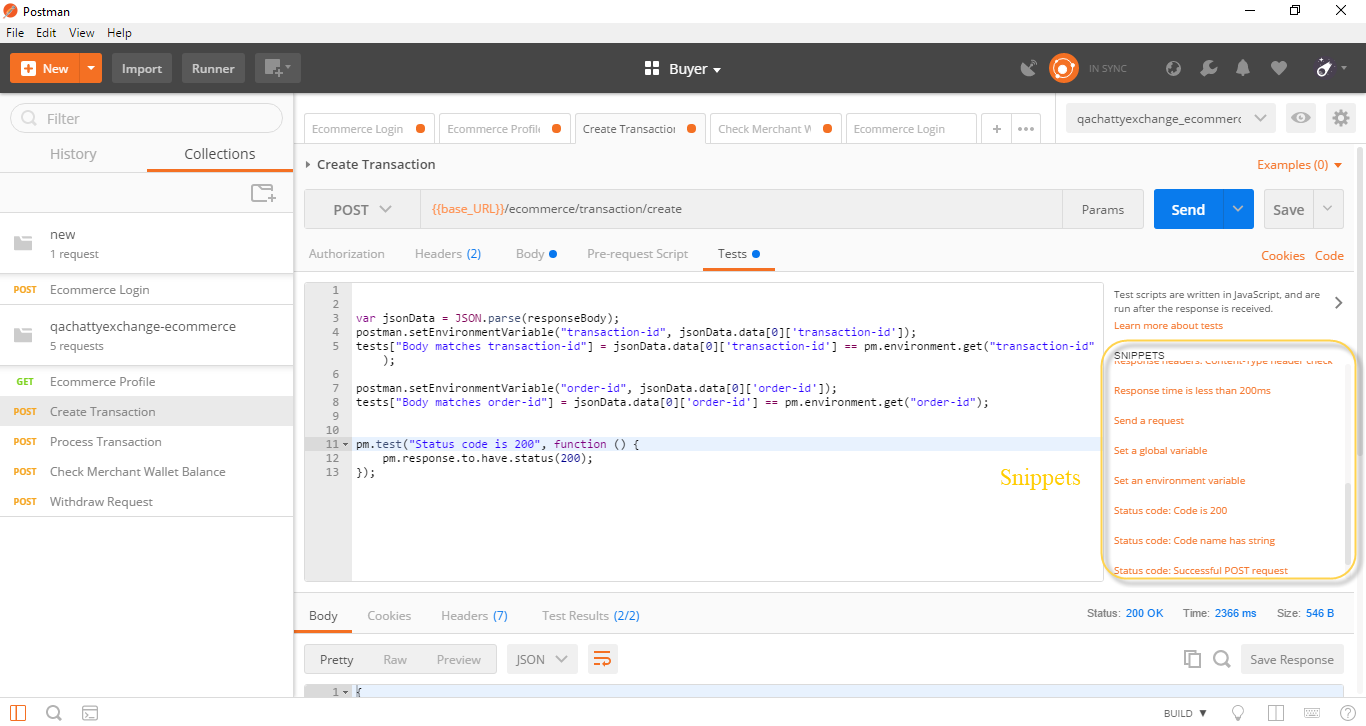
Here we are trying to set the transaction-id into environment variable accessing data from response body. So the transaction-id is inside the data in response body in [ ] bracket. So the code to access the transaction-id is jsonData.data[0][‘transaction-id’] (reference above figure). We here added [0] to access data due to the data here present is in array format.



## **Snippets**

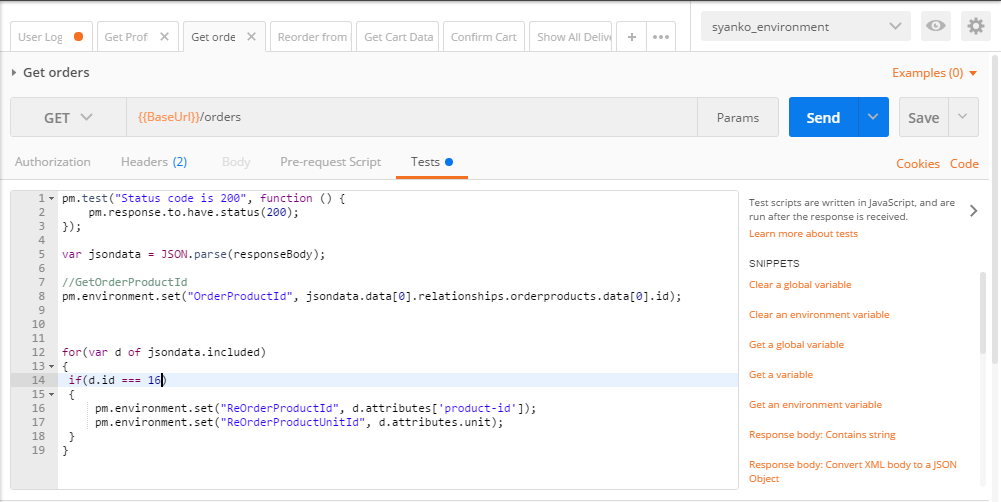
Snippets are the block of JS code which makes basic test easy and fast.

We can access the snippets block to check status code, set environment variable and so on.

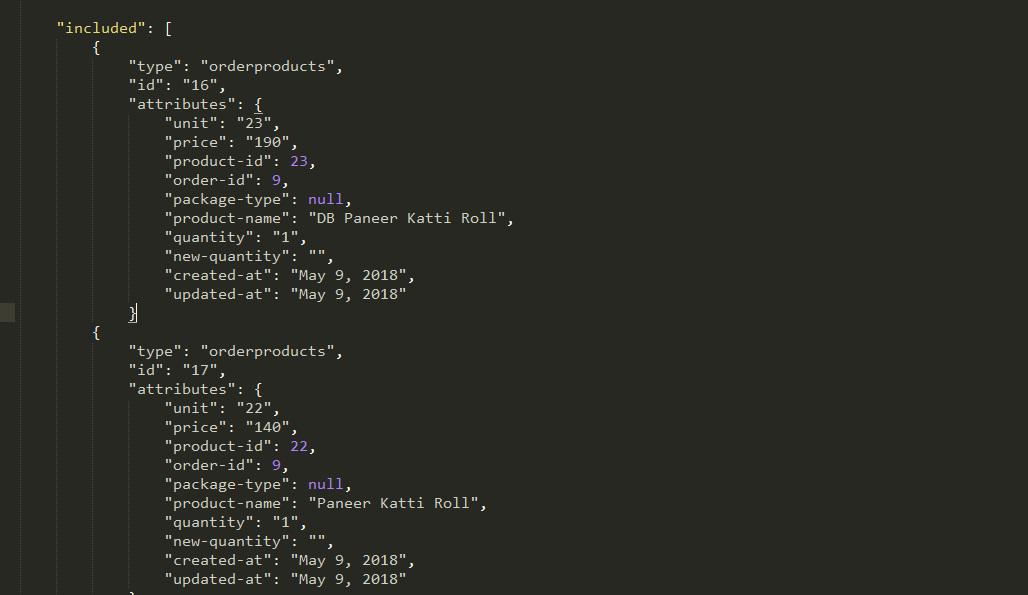


## **Relating JS Codes**

As per the API test in some request response might have multiple data in an array. To access this kind of data we must use loop and conditions in our test.



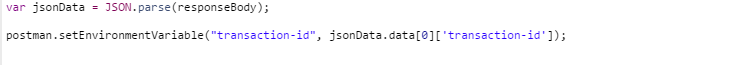
In the above figure, we have defined variable d which have certain value. The value here contains the attributes of the products. If the defined id matches the order number here it will set the attributes of that product in the environment variable.



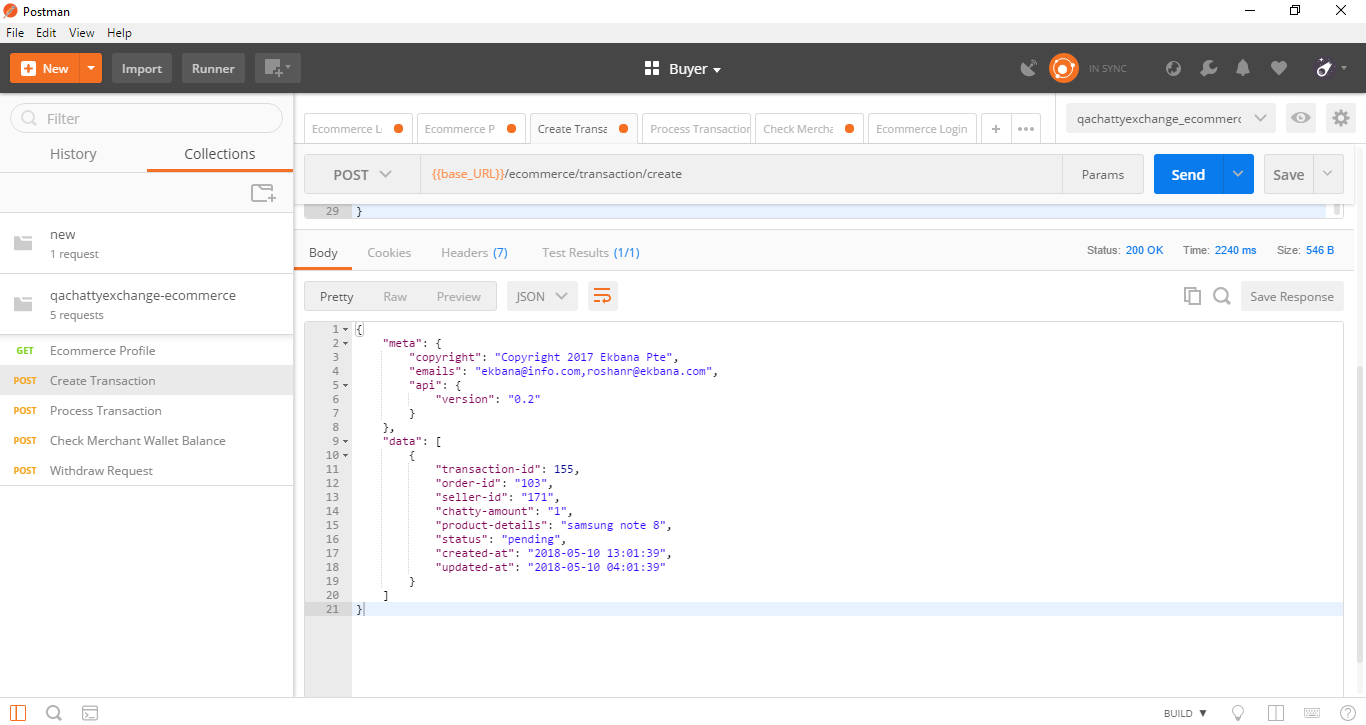
* 1. **SET and GET Variables**

Variables are symbols that can take different values. Most of the times variables are defined to reuse the values in multiple places so you can keep your code DRY (Don’t Repeat Yourself).

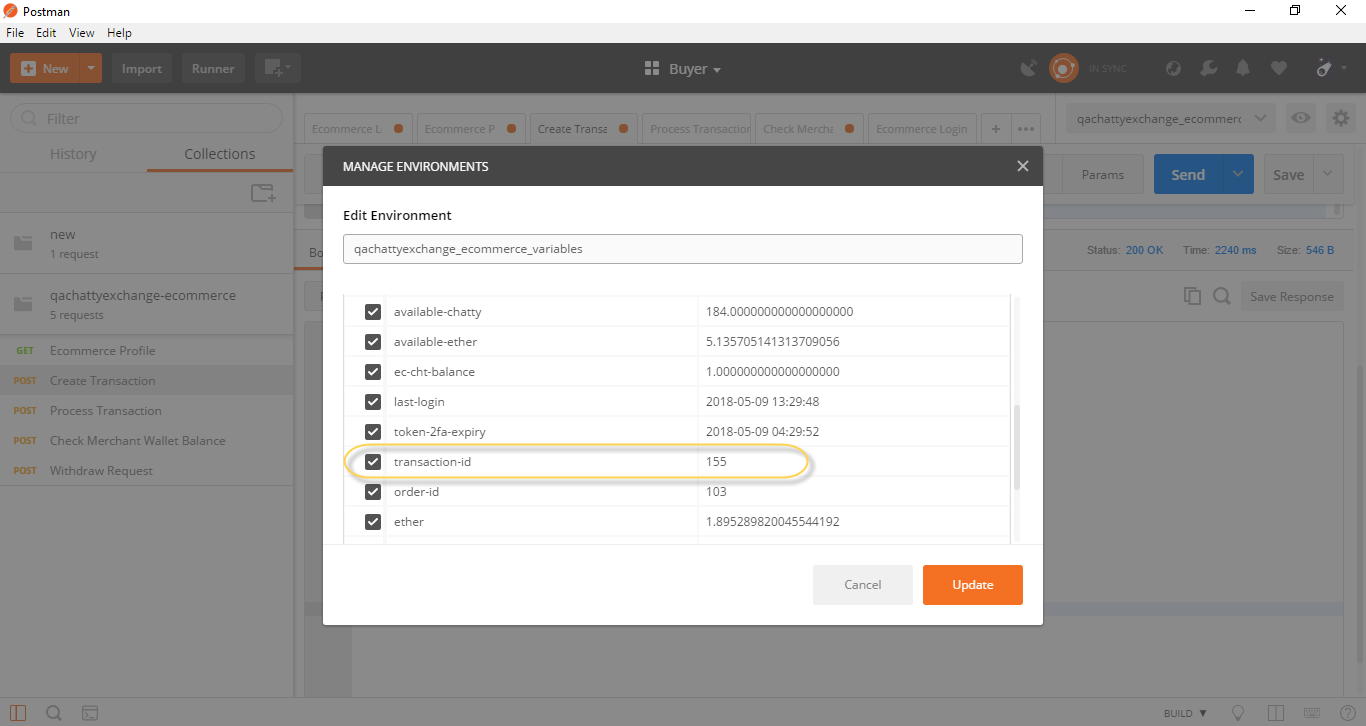
How to set?



The above code is use to set the environment variable.

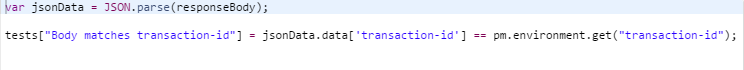


Here what we are trying to do is set transaction-id as an environment variable. So to set the transaction-id as environment variable we use the above code in the test scripts.



The transaction-id has been set as an environment variable. You can go to manage environments setting icon to view the variables that have been saved.

How to get?



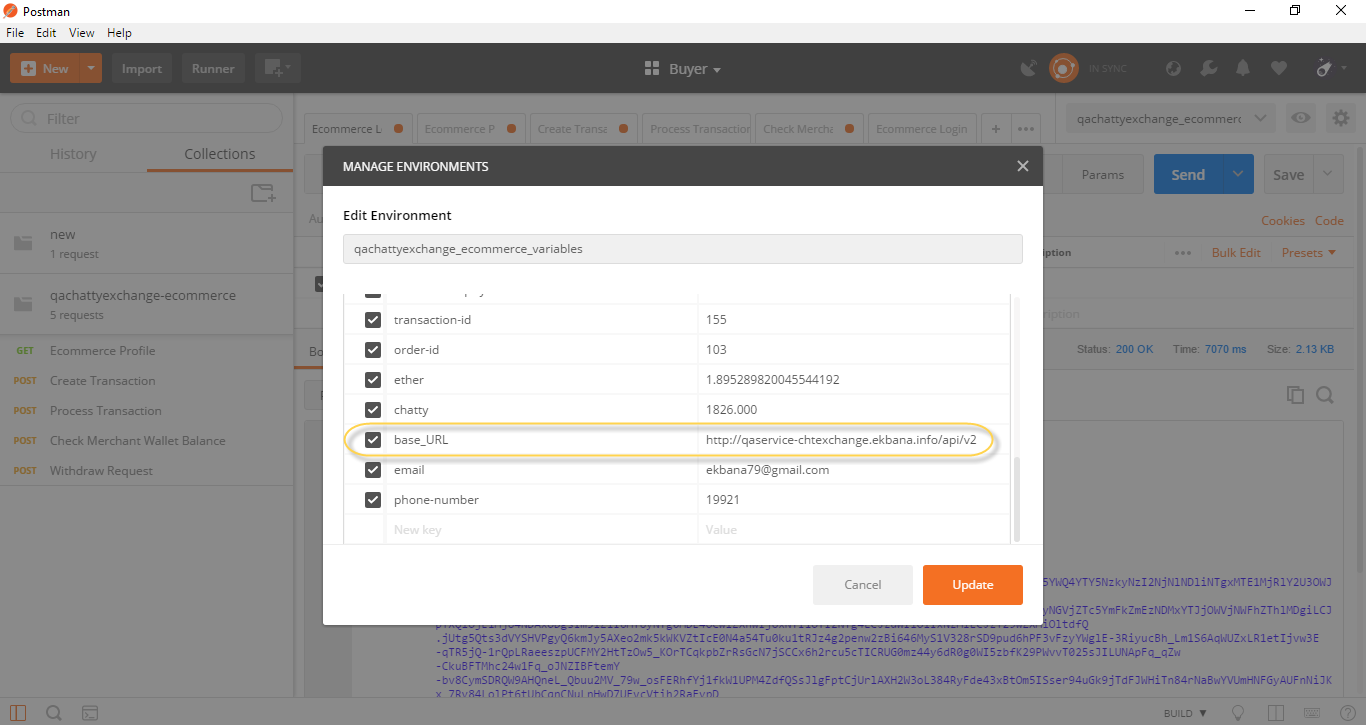
The above code is use to get the environment variable. The variable set is access through the get command. As per our project scenario we have to use transaction-id in multiple request so we set the transaction-id in environment variable and get the same transaction-id in other request which we have done.

## **Passing/Using variables**

Variables can be used in the following form in the postman user interface- {{variableName}}.

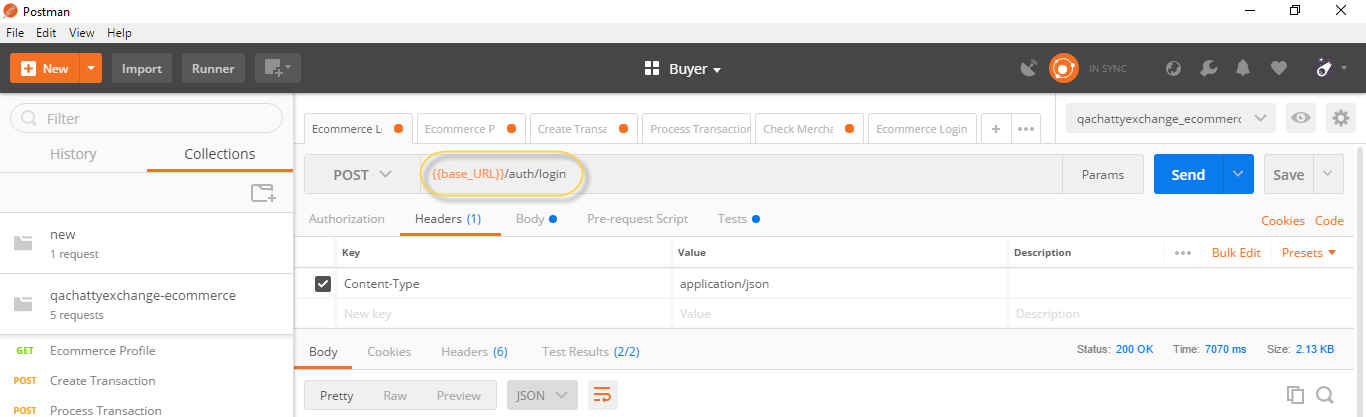
The string {{variableName}} will be replaced with its corresponding value when postman resolves the variable.

For more details <https://www.getpostman.com/docs/v5/postman/environments_and_globals/variables>

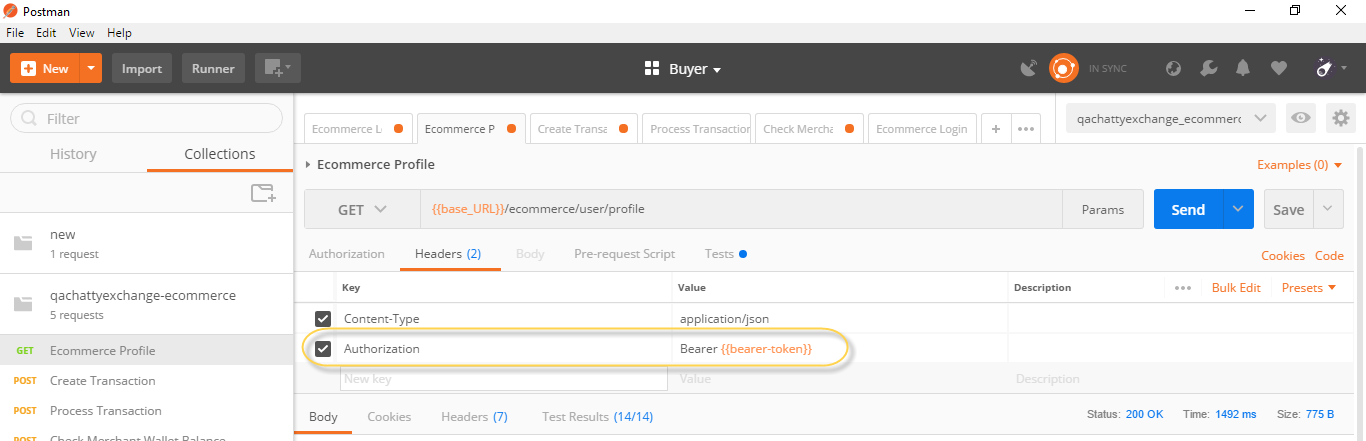


In the above figure, we have defined the base\_URL in the environment variable. Now we can access this variable in URL.

**i. Accessing the base\_URL variable in URL.**

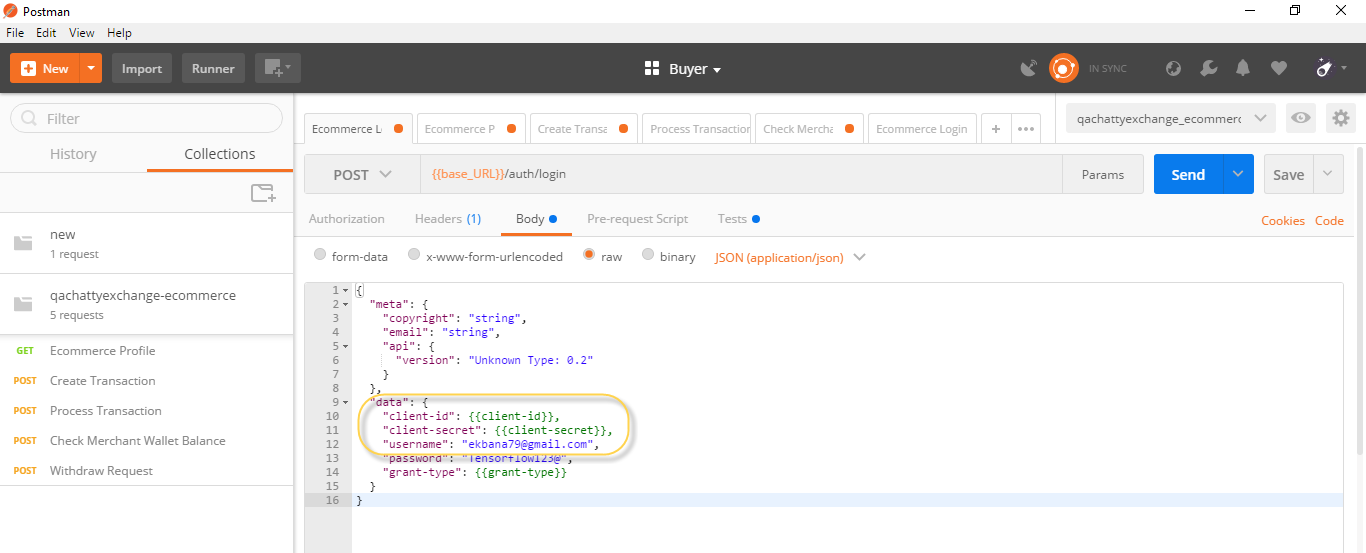


**ii. Accessing the Bearer token in Header**

****

As the base URL is defined in environment variable in the same way we defined Bearer token and pass the variable in the header as seen in the figure.

**iii. Accessing the variable in Request Body**

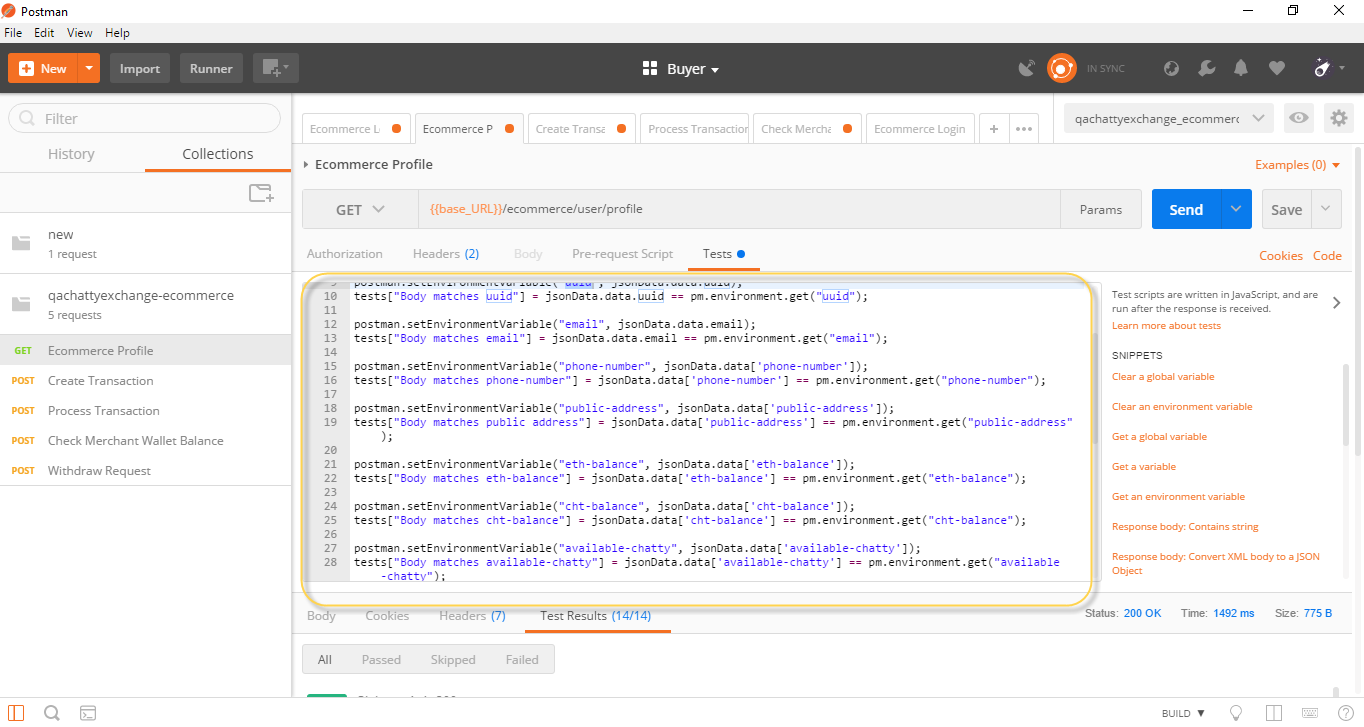
****

Here the client-id and client-secret id defined in the environment variable and pass the variable in the request body as {{client-id}} and {{client-secret}}.

In this way we can pass and use the variable in different areas.

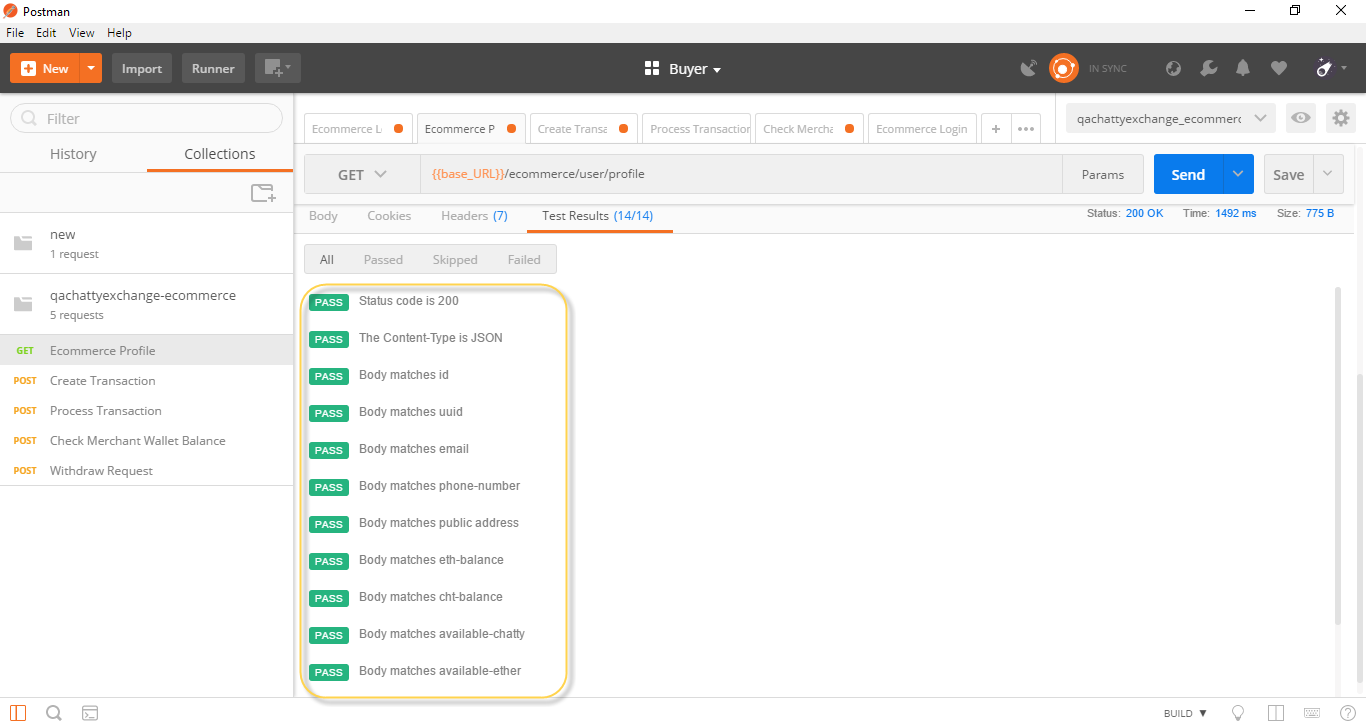
## **Using Multiple Request in One Test**

The response body contains lots of data which you want to validate and check whether the response body data matches as per the requirement or not. So we have to test the response body is providing actual data or not. In this case we do multiple test to check the actual value are reflecting or not.



In the above figure we have used multiple test in one request to validate id, email, etc.

## **Checking Test Response and Error Debugging**



The above figure shows the test results that we have tested and all the test have pass.

Not all the time test will pass, sometimes we mistype the name, we may be taking data in normal format but the data in response body may be in array format. So in that case we have to be careful and debug the errors.



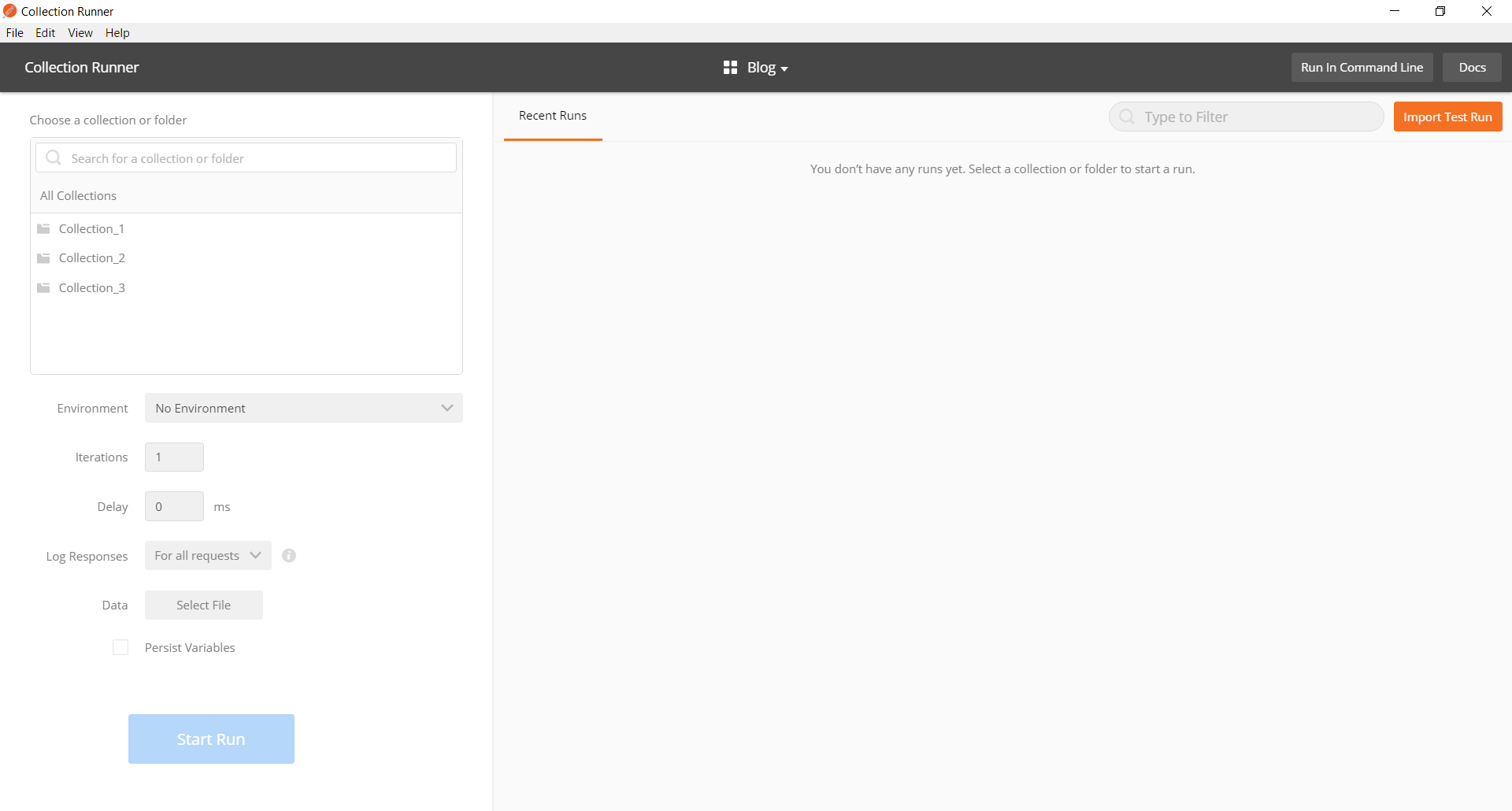


In the above case, we have mistype the variable name. So the test has been failed due to assertion error.

# **Collection Runner:**

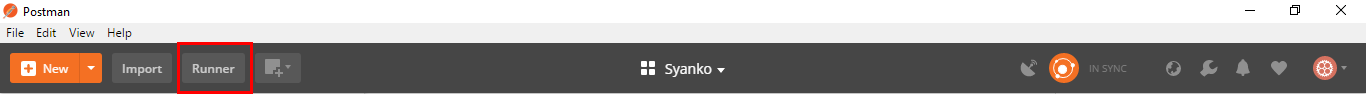
Creating a collection and the use of variables in postman are describes in above. Now we shall look into how to run the created collection containing different requests in a sequence.

Basically collections are a group of API requests that can be run together as a series of requests (one after the other), against a corresponding environment. We can perform the collection run within the postman app using the Collection runner and also from the command line using Postman’s Newman tool (which is discussed later on this blog).

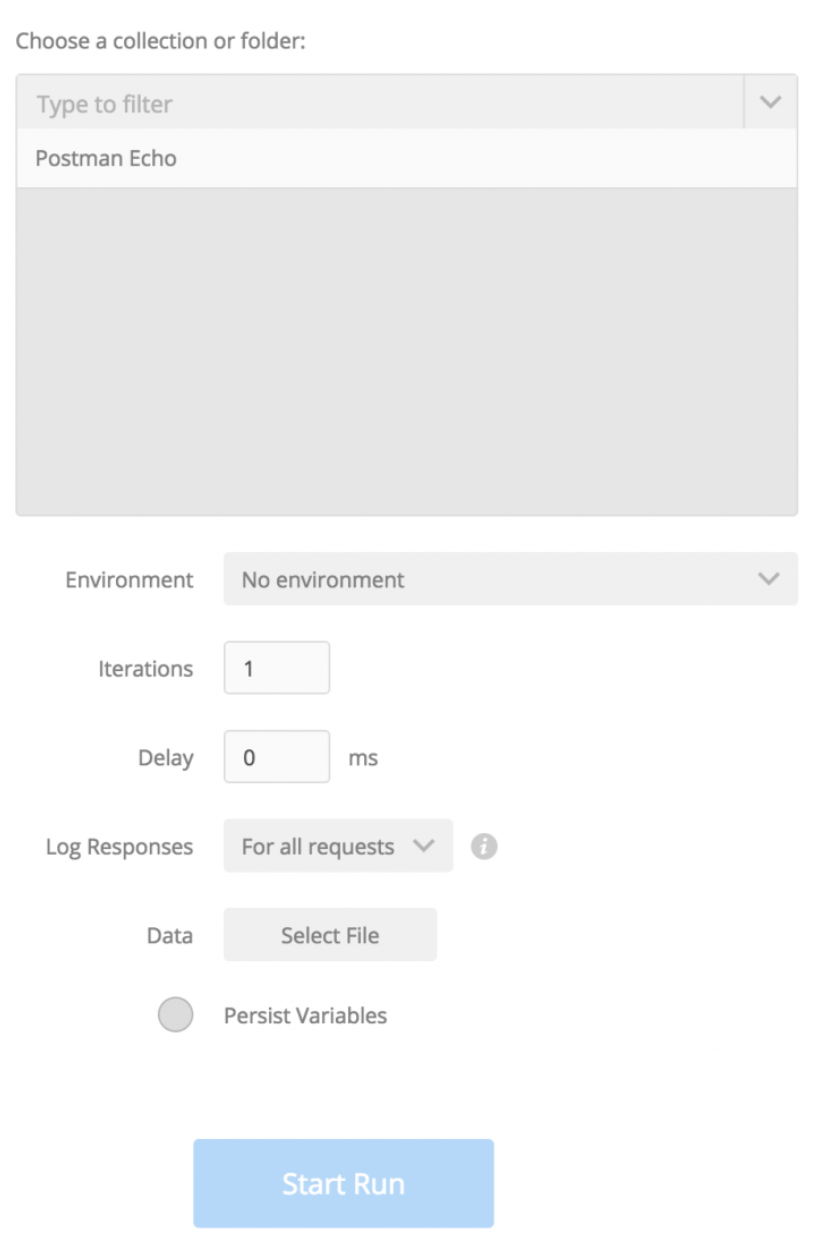


## **Starting the Collection Runner**

The Collection Runner can be started by clicking on the Runner button in the Tool bar.



Now let’s look over several parameters that you can configure for a collection run.



## **Choose a Collection or Folder:**

In this field we choose the collection or the folder that we want to run. If you choose a collection containing folders, then all the requests in the collection are sent in sequential order in which they are arranged inside the folder.

But, when you select a folder here, only that folder is executed, which means only requests inside the folder are sent.

## **Environment:**

In this field we choose the environment against which we want to run our collection.

## **Iterations:**

This is the number of times the whole collection will run. It is used when we have to run the same collection a number of times.

## **Delay:**

In this field we can set the interval between each request in the collection run if needed.

## **Log Responses:**

This field is used to limit the response logging when the collection is run. By default all the responses are logged. The options that can be chosen for logging are.

* For all requests: responses for all requests will be logged.
* For failed requests: only responses for requests with at least one failing test will be logged.
* For no requests: no responses will be logged.

## **Data:**

In this field we supply the data file to be used for the collection run.

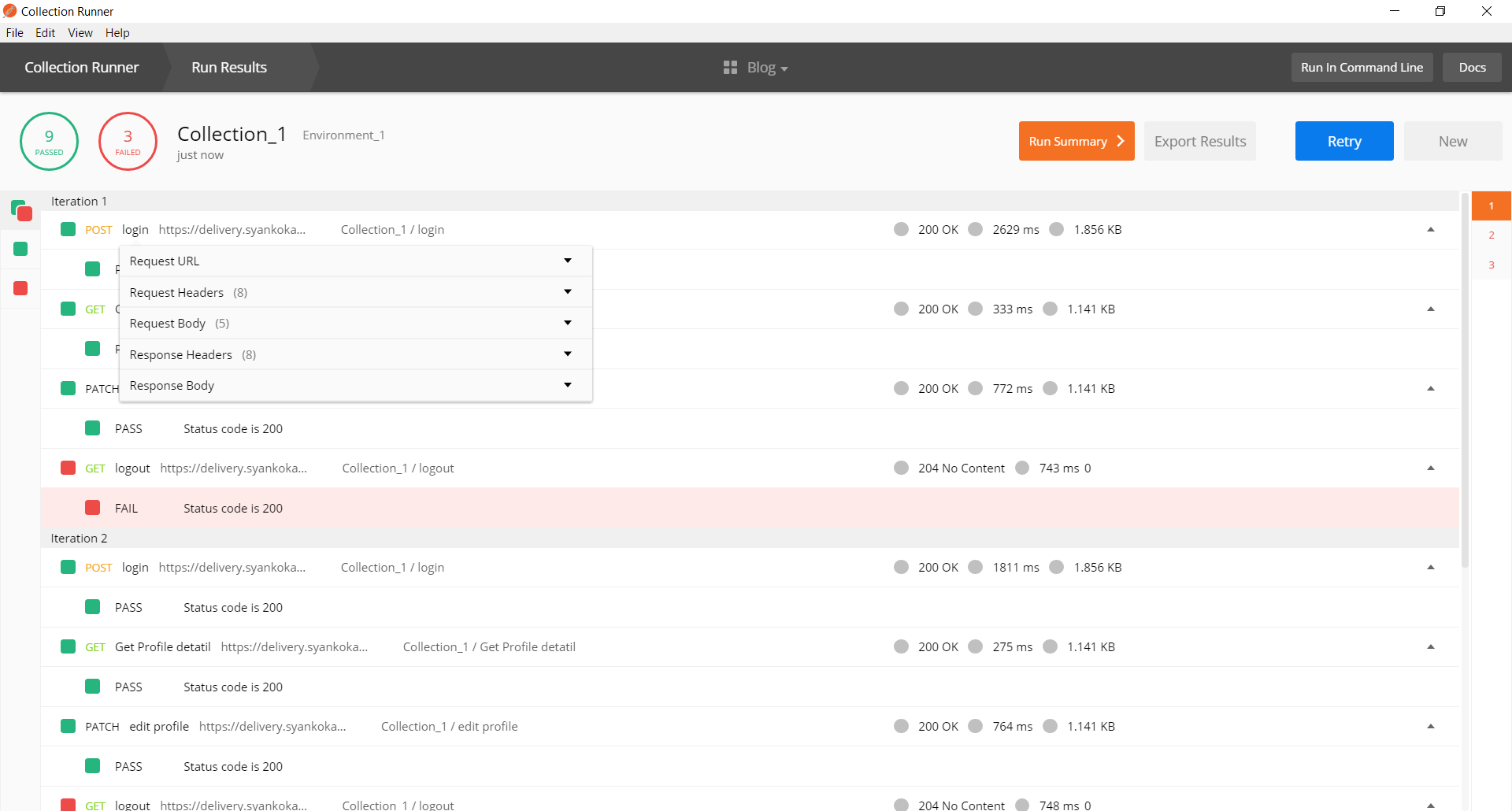
## **Persist variables:**

It is the feature which allows us weather to save or not to save the changes made in the environment variables when the collection is executed in the collection runner. By default, Persist Variables is checked the first time you open the Collection Runner. If you do not want variables to be updated during the run, deselect the Persist Variables checkbox.

Now, once we have filled all the fields according to our need then we can start the Collection Runner by clicking on the Start Run Button.

## **Checking and Debugging the Collection Runner Results:**

The collection runner also displays the result of the collection run. In this result page we can see the number of passed and failed requests in the total number of iteration.



Now things don’t always go as planned and sometimes your collection tests will fail even though you expect them all to pass. When this happens we can debug using the Request and Response Body.

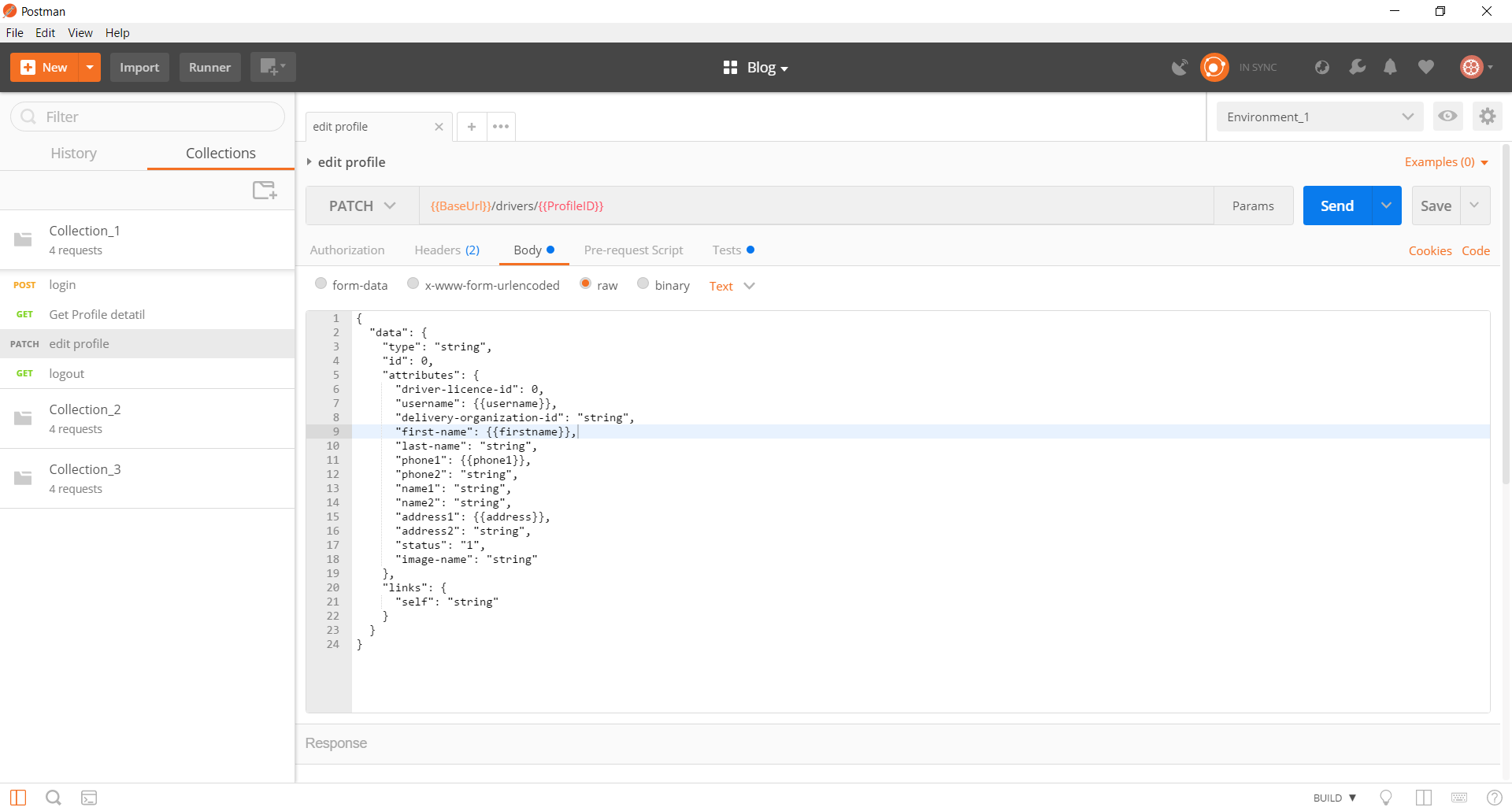
If you click on the request name in your collection run, a dropdown will appear which has the useful information about the request. Information you might need when figuring out what went wrong or why the test failed when it was supposed to pass. By expanding the different fields like Request URL, Request Header, Request Body, Response Headers and Response Body we can see what parameters were sent in the request and what response were generated in the request. Using this information, we solve the problem in the requests in our collection.

# **Data File**

We learned about different types of variables used in postman. Data files are also a way of passing the variables in different requests in postman. Data files are extremely powerful ways to test your APIs with varying data to check if they behave properly under unexpected circumstances.

Also Data Files are used to control the number of iteration of the collection run. Hence, we can think of data files are parameters for each iteration of a collection run.

Now let’s look into where are how data files are used in an API request with the help of the following example



In the picture above notice that five data file variables are used. {{ProfileID}} in Request URL and {{username}}, {{firstname}}, {{phone1}} and {{address}} in the Request body section. Data files are passed in the request header same as in the request URL section. These are used just like the environment variables.

We will supply the value to these variables using a JSON / CSV file. Which brings us to our next topic.

## **Creating a Data File**

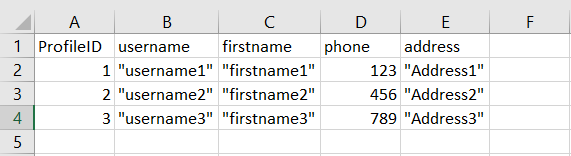
Postman currently supports JSON & CSV files.

The JSON data file for the above variables will look like



This is an array of objects. Each object represents the variable values for one iteration. Each member of this object represents a variable. In this way, in the first iteration, the variable called **ProfileID** will have the value 1 and variable **username** will have the value username1 and so on. Similarly, in the second iteration the variable **ProfileID** will have the value2 and the variable **username** will have the value username2 and so on. Note that the name of the variable in the postman app and the JSON file should exactly be the same.

Now let’s look at the CSV format. The CSV file for the above variables will look like.

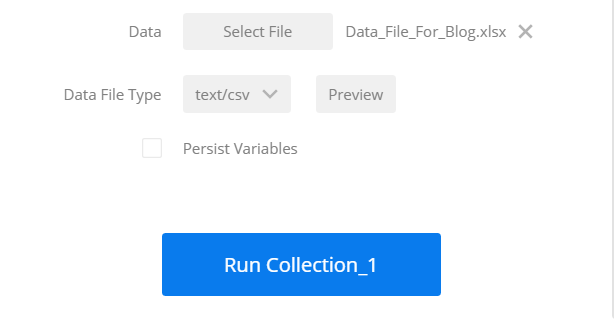


In the CSV format the first row represents the variable names and the following rows represent the values for these variables for each iteration. The number of row determine the number of iterations

Do note that you can use only one data file for one run, so put all the variables values in a single data file.

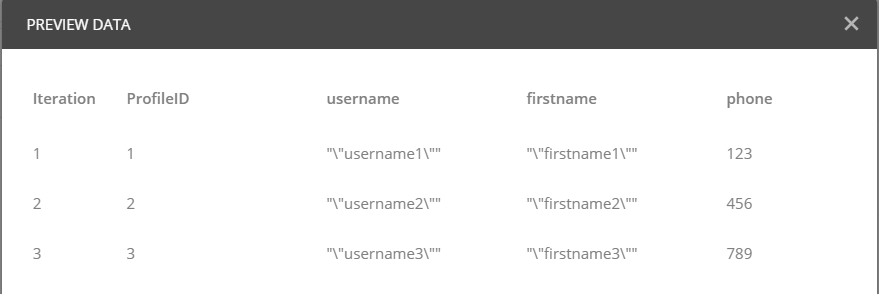
## **Using Data files in Collection Runner:**

Now that we have learned how to create a data file, let’s look into how to supply the data file in the collection runner.



After choosing the Collection and the Environment, click on the Select File button of the Data field in the Collection Runner window and select your data file. Then select the file type (CSV or JSON) in the Data File Type field.

You can also preview what values each variable has in each iteration by clicking on Preview next to the file name.



After you have chosen all the required fields, just click on the **Run** button to run the collection against the selected Data file.

# **Running the Collection in Command prompt and Report Generation**

## **Introduction to Newman**

Newman is the command line collection runner for postman. Through Newman you can run and test a postman collection directly from the command line. Due to which it can be easily integrated with your continuous integration servers and build system.

Also another feature of Newman is that we can generate different reports of the Collection run carried out in the command line. The mostly used form of reports that is generated using postman are HTML, CLI, JSON, Junit etc.

## **Newman Setup:**

Now let us look into how to setup Newman

Newman is built on node.js, so to run Newman make sure you have node.js installed. If you haven’t installed node.js and do not know how to do it don’t worry.

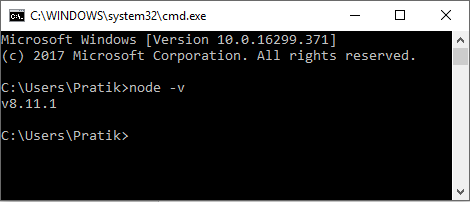
Below I have listed the steps to install node.js and install Newman.

**Steps:**

1. Install Node.js

Follow the following steps to install node.

* 1. Download Node.js from <https://nodejs.org/en/>
  2. Run the setup/ install node.js. (restart of the device may be required)
  3. Setup the system variable in the environment.
     1. Go to start
     2. Type go to Edit the system environment variable option.
     3. Click on Environment variable button.
     4. Click on the new button of system variable
     5. Set the variable name as nodejs
     6. In the variable value field set the path of the installed node.js folder. Usually, the path is “C:\Program Files\nodejs”.
     7. Click on ok.
  4. Open command and type node –v to check if node is successfully installed.



If node is installed the version of the installed node will be displayed as shown in the picture above.

1. Now to install Newman, type “npm install –g newman” in the command line and click Enter button. (It may take couple of minutes for Newman to install)

## **Running Collection in Newman**

Once Newman is installed, it’s time to run the collection in newman and view its result.

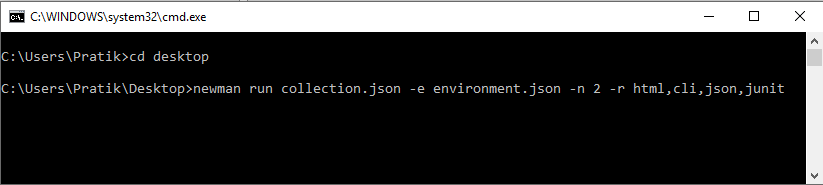
The first thing we have to do before running newman is that the Collection JSON file, Environment JSON file and the Data file all should be inside a single folder.

Keeping this in mind let’s look at the general code used in newman to run a collection

C:\Users\Pratik\Desktop> newman run collection.json -e environment.json -n 2 -r html,cli,json,junit

Description:

1. C:\Users\Pratik\Desktop> : Specifies the directory in why the exported collection, environment and data file is saved in the system. Also the location where the newman reports are saved.
2. Newman run: It is the general code written to start the newman execution which specifies the collection to be run.
3. Collection.json: name of the exported collection file. (Note: Do not name your collection with a space in between e.g.: abc collection.json because the space is not accepted in commend line while running the code.)
4. –e environmebt.json : name of the exported environment file. (Note: Do not name your environment with a space in between e.g.: abc environment.json because the space is not accepted in commend line while running the code.)
5. –n 2 : Specifies the number of times the collection has to be run when used in conjunction with iteration data file.
6. -r html,cli,json,junit : Specifies the type of reports to be generated.



There are many other options provided by newman to customize a run. If you want to look in the other features please visit : <https://github.com/postmanlabs/newman>

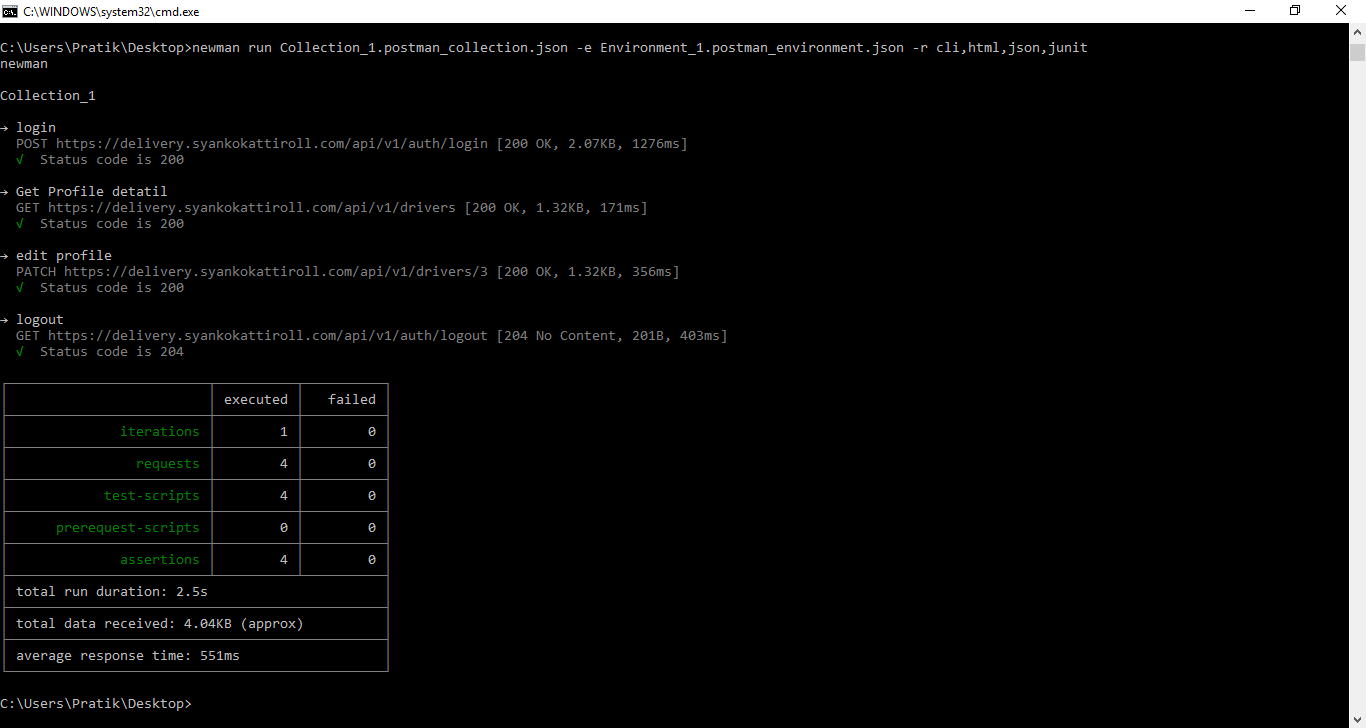
## **Viewing Newman Results**

The –r tag is used in the above newman code example is used to specify different reports to be generated in newman. Once the newman code is run it will generate different results based on the input given. The available reports are: cli, json, html and junit.

Note that spaces should not be used between report names / commas whilst specifying a comma separated list of reporters. For Example:

*-r html,cli,json,junit is correct while -r html, cli , json,junit is wrong way to write.*

Now the cli report will be generated in the Command Prompt itself after you click the enter button.



The cli report will look like this.

The html, json, junit report will be saved inside a newman folder in the location where the collection and the environment are kept. In case of above example, the newman folder containing the html, json and junit report will be saved in the desktop.