

Tutorial for Postman

In this tutorial, you'll learn how to call a web API using Postman.

Step One: Start the API server

First, make sure you are in the same directory as `package.json`. Then, run the command `npm install` from the command line to install any dependencies. You won't need to do this on any subsequent run.

Next, start the project in VS Code by running `npm start`. This starts a local web API server that you'll use with Postman. The server is an API that simulates an Instagram-like API with users, images, and comments.

Once the server starts, you'll see a message that looks like this:

```
\{^_^}/ hi!  
  
Loading ./db/instaounce.json  
Done  
  
Resources  
http://localhost:3000/users  
http://localhost:3000/images  
http://localhost:3000/comments  
  
Home  
http://localhost:3000  
  
Type s + enter at any time to create a snapshot of the database
```

Note: You can stop the server, or any other process that has been started from the console, by using the keyboard shortcut `ctrl + c`.

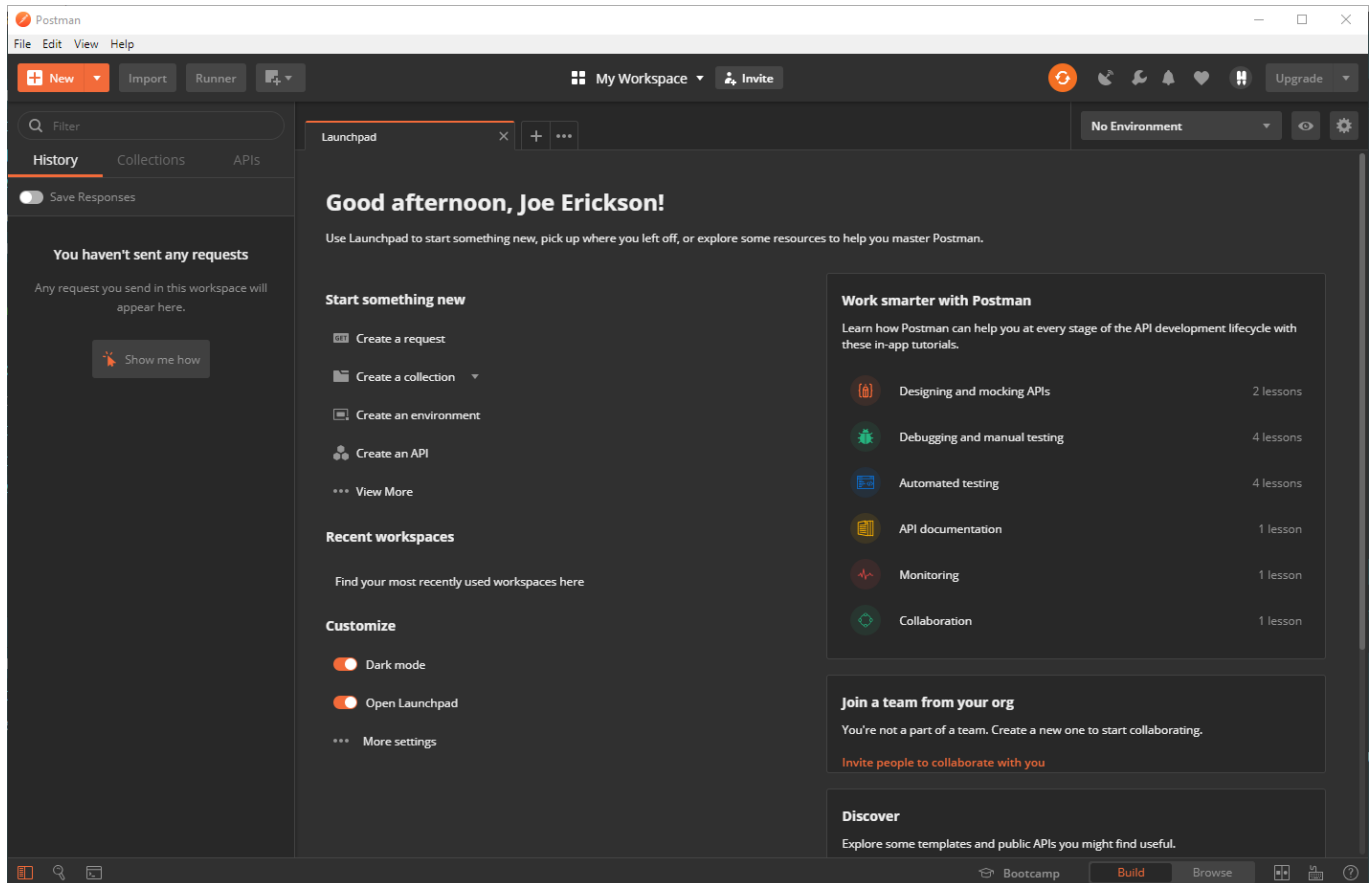
If you see this message, it means that the web API server has started successfully and is now waiting for requests. To send it HTTP requests, you'll use Postman.

Step Two: Start Postman

Postman should be installed on your laptop already. The icon looks like this in either your start menu or in your Applications folder:



Double-click that icon. You should see a new screen like this:

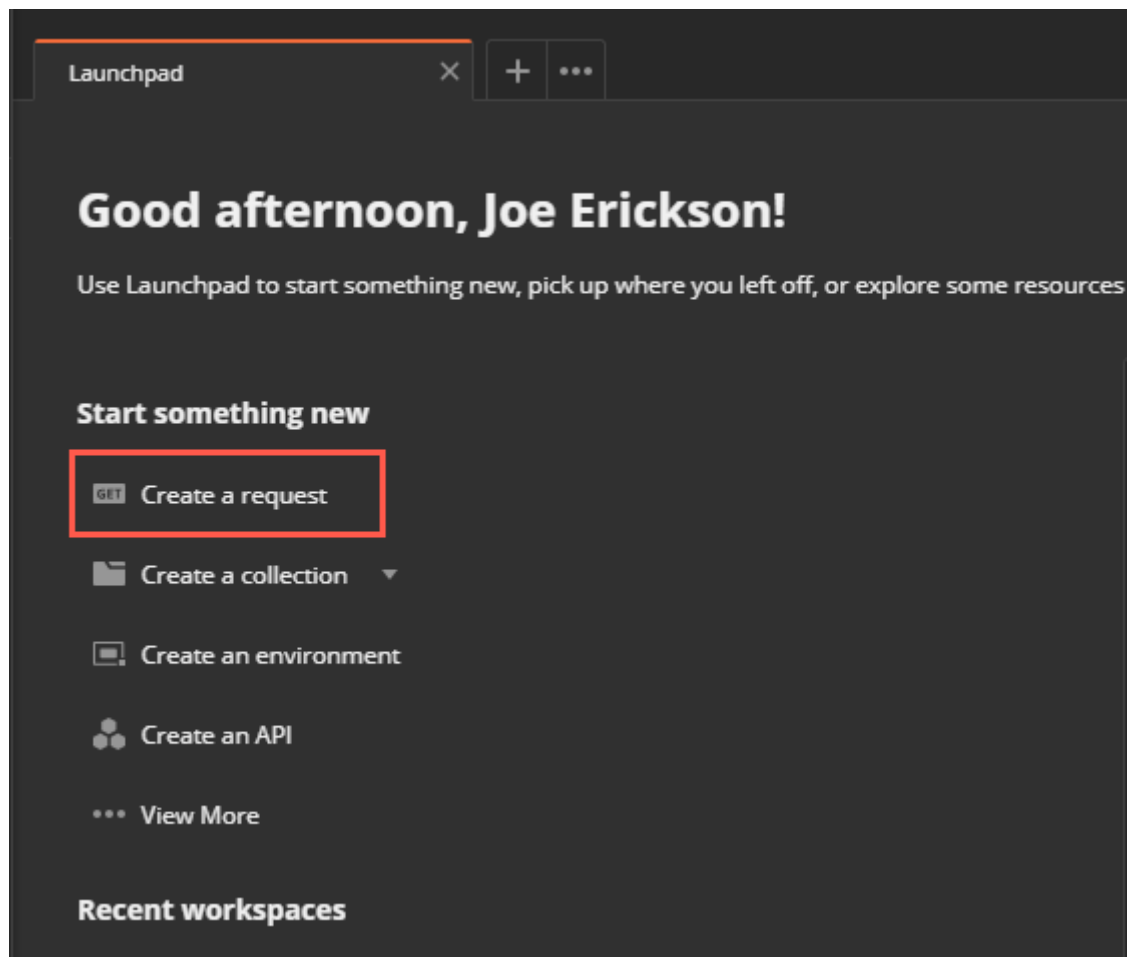


Postman is an application that lets you make HTTP requests like a browser but instead shows the raw information returned from the server. When you're interacting with a web server, that information is typically HTML. But when interacting with a web API, that information is usually JSON data, and a browser isn't designed to display that information very well.

Web APIs also typically expect special HTTP headers and status codes that are more complex to set in a browser, so Postman is used in those circumstances. Think of it as your first interface to a Web API.

Step Three: Create a new request in Postman

In the Postman interface, click the "Create a request" link to start a new API session:

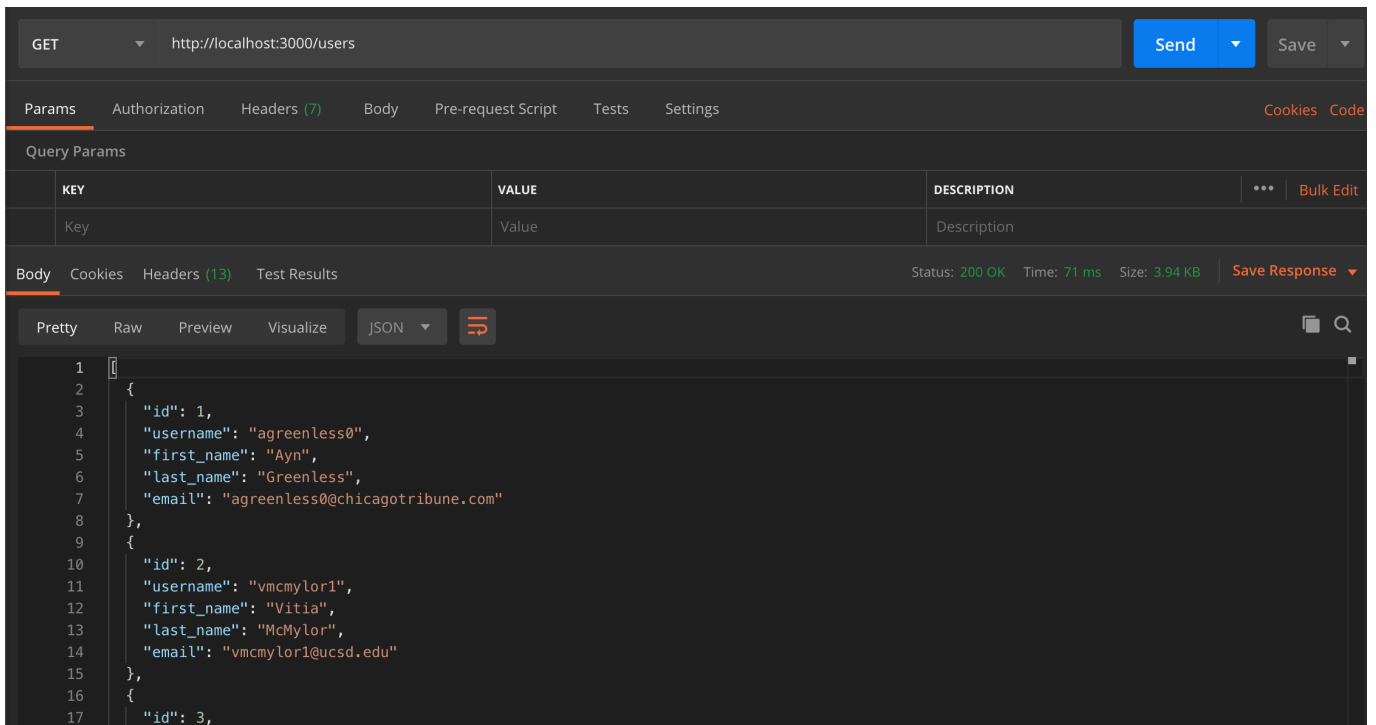


This session records and lets you play back API requests.

For your first request, make a request that returns all the users as an array. The URL for that request is <http://localhost:3000/users>. Enter the URL in the text box and then click send:



In the body tab, you'll see the JSON data for 25 users:



GET <http://localhost:3000/users> Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (13) Test Results Status: 200 OK Time: 71 ms Size: 3.94 KB Save Response

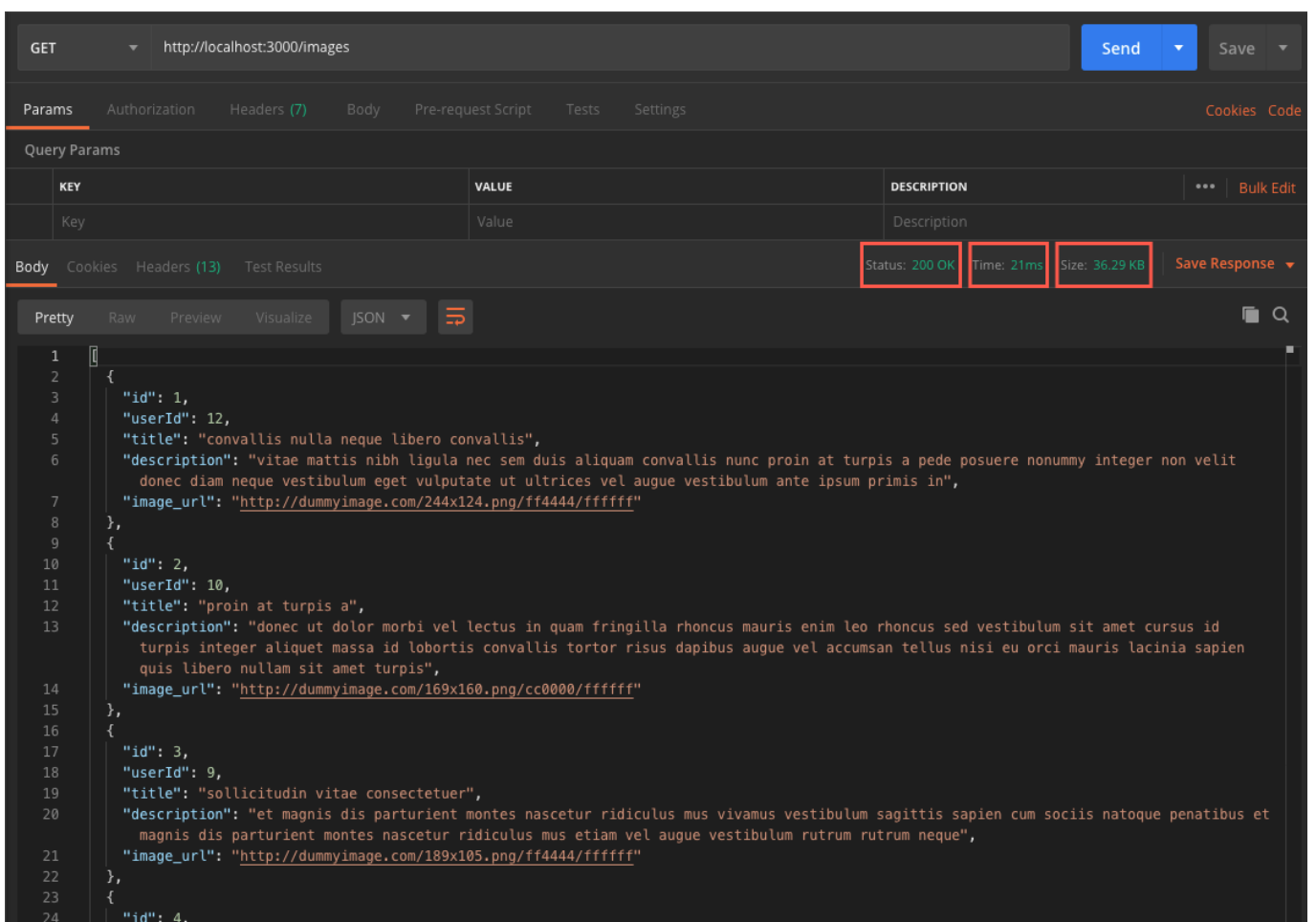
Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "username": "agreenless0",
5     "first_name": "Ayn",
6     "last_name": "Greenless",
7     "email": "agreenless0@chicagotribune.com"
8   },
9   {
10    "id": 2,
11    "username": "vmcmylor1",
12    "first_name": "Vitia",
13    "last_name": "McMylor",
14    "email": "vmcmylor1@ucsd.edu"
15  },
16  {
17    "id": 3,
```

1. Modify GET request

If you change the request to <http://localhost:3000/images>, you'll see the image information load in when you click Send.

The Postman interface also shows you three important pieces of information about the request: the returned HTTP Status Code, how long the response took, and the size of the response data:



GET <http://localhost:3000/images> Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (13) Test Results Status: 200 OK Time: 21ms Size: 36.29 KB Save Response

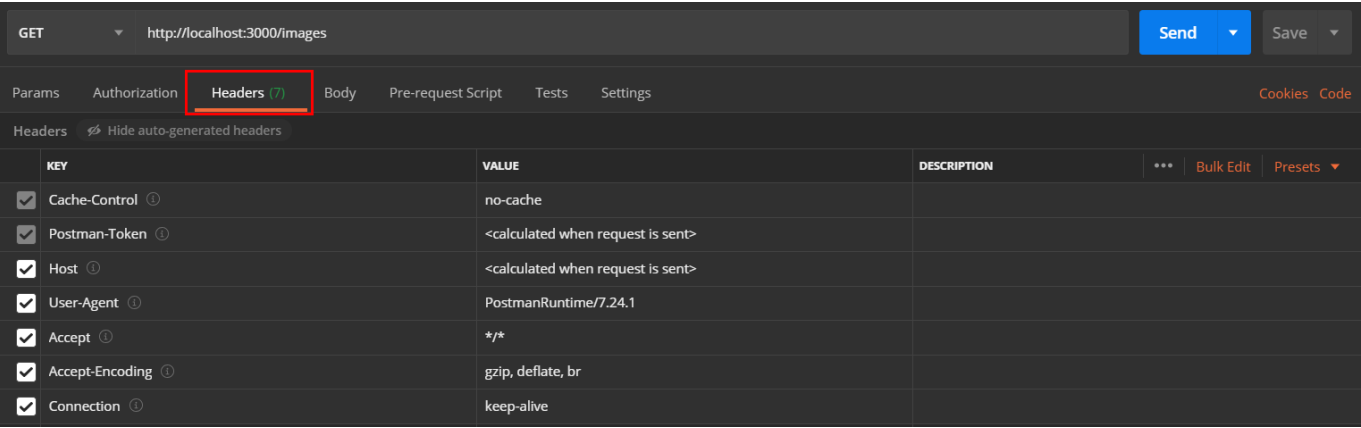
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "userId": 12,
4   "title": "convallis nulla neque libero convallis",
5   "description": "vitae mattis nibh ligula nec sem duis aliquam convallis nunc proin at turpis a pede posuere nonummy integer non velit
6     donec diam neque vestibulum eget vulputate ut ultrices vel augue vestibulum ante ipsum primis in",
7   "image_url": "http://dummyimage.com/244x124.png/ff4444/ffffff"
8 },
9 {
10  "id": 2,
11  "userId": 10,
12  "title": "proin at turpis a",
13  "description": "donec ut dolor morbi vel lectus in quam fringilla rhoncus mauris enim leo rhoncus sed vestibulum sit amet cursus id
14    turpis integer aliquet massa id lobortis convallis tortor risus dapibus augue vel accumsan tellus nisi eu orci mauris lacinia sapien
15    quis libero nullam sit amet turpis",
16  "image_url": "http://dummyimage.com/169x160.png/cc0000/ffffff"
17 },
18 {
19  "id": 3,
20  "userId": 9,
21  "title": "sollicitudin vitae consectetur",
22  "description": "et magnis dis parturient montes nascetur ridiculus mus vivamus vestibulum sagittis sapien cum sociis natoque penatibus et
23    magnis dis parturient montes nascetur ridiculus mus etiam vel augue vestibulum rutrum rutrum neque",
24  "image_url": "http://dummyimage.com/189x105.png/ff4444/ffffff"
25 },
26 {
27   "id": 4,
```

You can use this information for debugging purposes—for example, how slow the API response is in case you need to compensate for it.

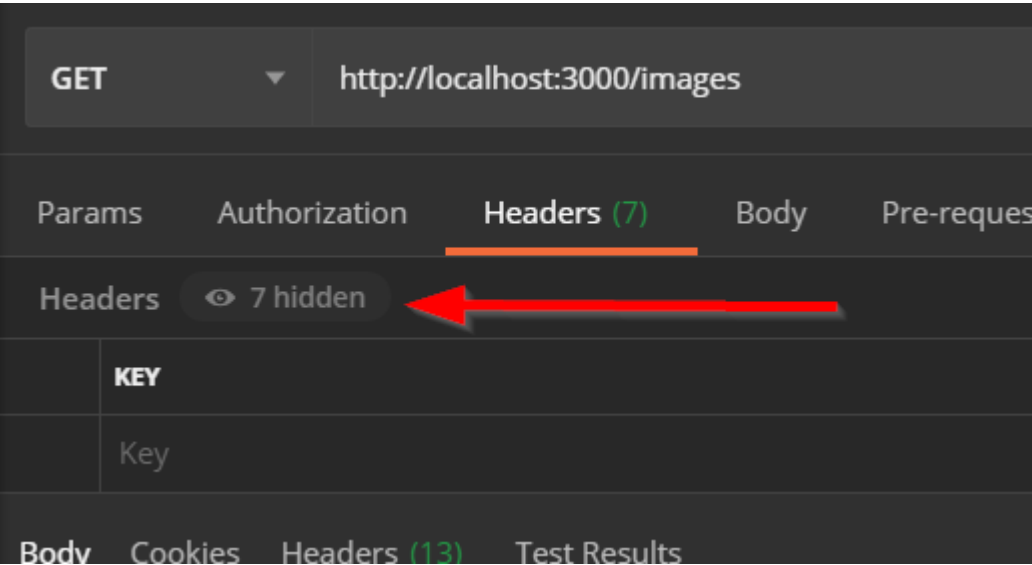
Step Four: Review request and response headers

Clicking on Headers in the Request section lets you set HTTP headers for the request and see what HTTP headers Postman sets for the request:



Postman sets and auto-generates some of the headers, like the User-Agent and Host. These can be overwritten by your own headers for testing or exercising the back-end API.

Note: You may need to click the "X hidden" button to show the hidden headers.



You can also see the Response Headers that the server sends back in the Response view:

Body	Cookies	Headers (13)	Test Results	Status: 200 OK	Time: 21ms	Size: 36.29 KB	Save Response ▼
		KEY	VALUE				
		X-Powered-By ⓘ	Express				
		Vary ⓘ	Origin, Accept-Encoding				
		Access-Control-Allow-Credentials ⓘ	true				
		Cache-Control ⓘ	no-cache				
		Pragma ⓘ	no-cache				
		Expires ⓘ	-1				
		X-Content-Type-Options ⓘ	nosniff				
		Content-Type ⓘ	application/json; charset=utf-8				
		ETag ⓘ	W/"8f97+gtJgZQR1GwvMgANtiGs6Y80e1s"				
		Content-Encoding ⓘ	gzip				
		Date ⓘ	Tue, 25 Feb 2020 18:20:51 GMT				
		Connection ⓘ	keep-alive				
		Transfer-Encoding ⓘ	chunked				

These are the headers that the server sends back. These can be useful to look at if you need to debug the API.

Summary

You can now:

- Open Postman
- Make requests in Postman to web APIs
- View and set the HTTP Headers in the request
- View the Response Status Code, response time, size, and headers