Mock Server in Postman

- Use a mock server to simulate your API without having to configure a real API server.
- You can use any HTTP collection to set up a mock server.
- When you send a request to the mock server, Postman returns a real-world response using data from your collection.

Reference link:

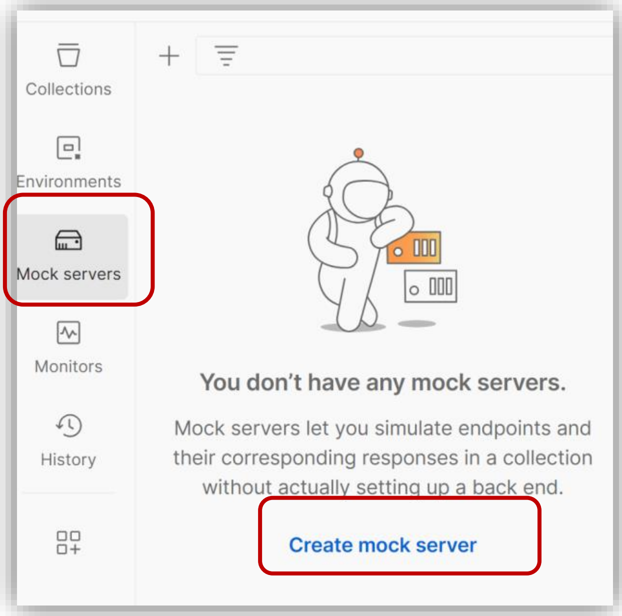
https://learning.postman.com/docs/designing-and-developing-your-api/mocking-data/setting-up-mock/

Steps to create Mock Server in Postman:

1. Open **Postman**.
2. Go to **New** → **Mock Server**.
3. Select **Create a new mock server** and choose **Add a request**.
4. Define the above APIs (one by one) with their endpoints, request methods, and example responses.
5. Save the mock server.
6. Use the generated mock server URL to test these APIs.

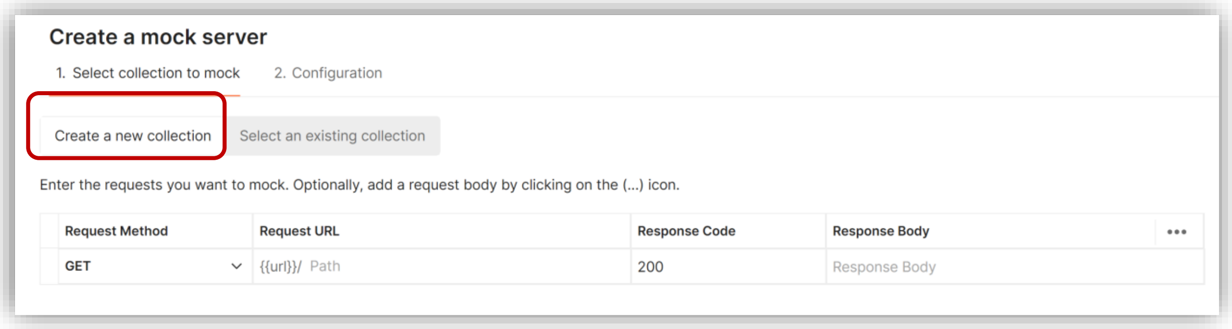
We can create Mock Server for **new Collection** (or) for **Exiting collection**.

Go to Mock servers→ Create mock server (For new collection)



You don't have any mock servers.

Mock servers let you simulate endpoints and their corresponding responses in a collection without actually setting up a back end.



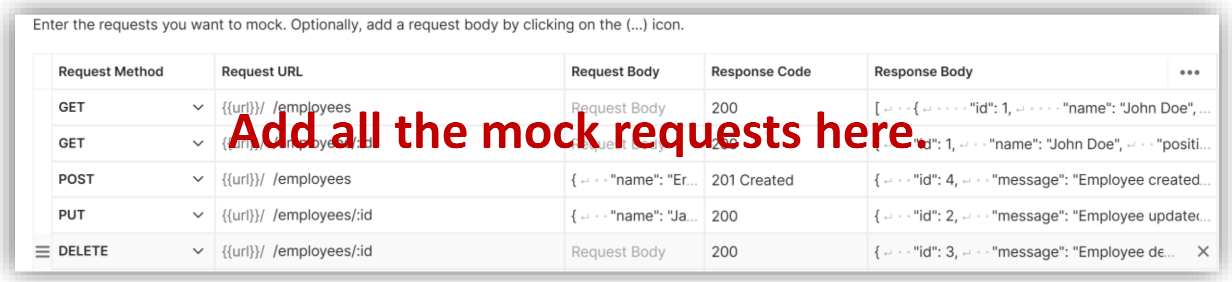
Create a mock server

1. Select collection to mock   2. Configuration

Create a new collection   Select an existing collection

Enter the requests you want to mock. Optionally, add a request body by clicking on the (...) icon.

Request Method	Request URL	Response Code	Response Body
GET	{{url}}/ Path	200	Response Body

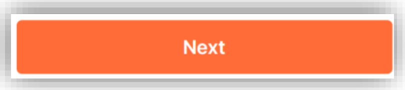


Enter the requests you want to mock. Optionally, add a request body by clicking on the (...) icon.

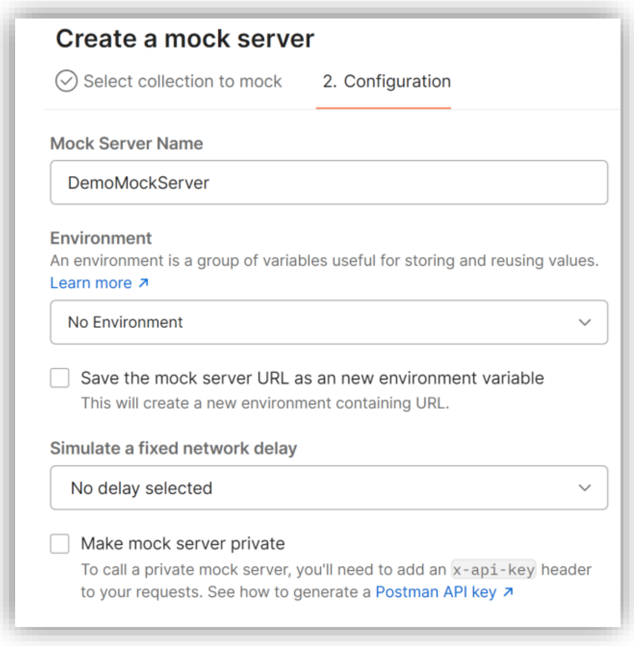
Request Method	Request URL	Request Body	Response Code	Response Body
GET	{{url}}/ /employees	Request Body	200	[{ "id": 1, "name": "John Doe", ...
GET	{{url}}/ /employees/:id	Request Body	200	{ "id": 1, "name": "John Doe", "positi...
POST	{{url}}/ /employees	{ "name": "Er...	201 Created	{ "id": 4, "message": "Employee created...
PUT	{{url}}/ /employees/:id	{ "name": "Ja...	200	{ "id": 2, "message": "Employee updatec...
DELETE	{{url}}/ /employees/:id	Request Body	200	{ "id": 3, "message": "Employee de...

Add all the mock requests here.

After that Click on the **Next Button.**



Provide mock server name.



Create a mock server

Select collection to mock   2. Configuration

Mock Server Name

DemoMockServer

Environment

An environment is a group of variables useful for storing and reusing values. Learn more ↗

No Environment

□ Save the mock server URL as an new environment variable

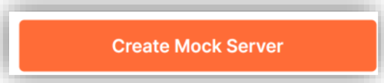
This will create a new environment containing URL.

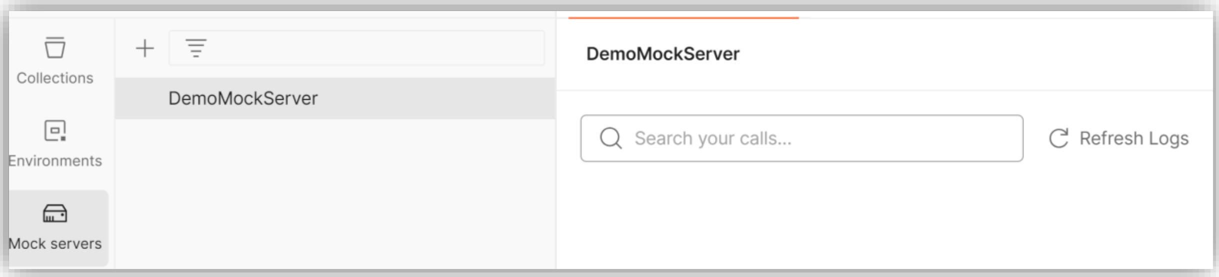
Simulate a fixed network delay

No delay selected

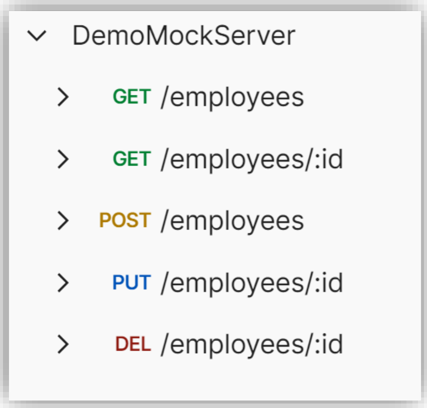
□ Make mock server private

To call a private mock server, you'll need to add an `x-api-key` header to your requests. See how to generate a Postman API key ↗

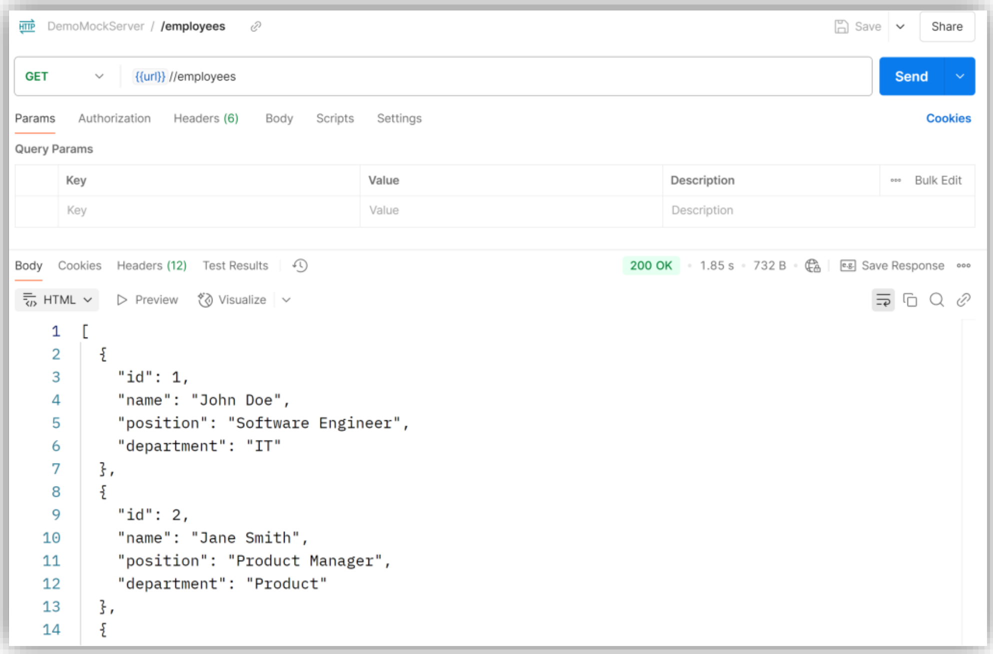




After that go to Collections list, you will see new collection is created with Mock API's.



Sending Request:



GET {{url}} //employees

200 OK

```
[
  {
    "id": 1,
    "name": "John Doe",
    "position": "Software Engineer",
    "department": "IT"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "position": "Product Manager",
    "department": "Product"
  },
  {
```

Mock API’s

1. Get List of All Employees (GET)

Endpoint: /employees
Response:

```
[
  {
    "id": 1,
    "name": "John Doe",
    "position": "Software Engineer",
    "department": "IT"
  },
  {
    "id": 2,
    "name": "Jane Smith",
    "position": "Product Manager",
    "department": "Product"
  },
  {
    "id": 3,
    "name": "Alice Johnson",
    "position": "QA Tester",
    "department": "Quality Assurance"
  }
]
```

2. Get Employee by ID (GET)

Endpoint: /employees/:id
Example URL: /employees/1
Response:

```
{
  "id": 1,
  "name": "John Doe",
  "position": "Software Engineer",
  "department": "IT"
}
```

3. Create Employee (POST)

Endpoint: /employees
Request Body:

```
{
  "name": "Emma Wilson",
  "position": "UI/UX Designer",
  "department": "Design"
}
```

Response:

```
{
  "id": 4,
  "message": "Employee created successfully"
}
```

4. Update Employee by ID (PUT)

Endpoint: /employees/:id
Example URL: /employees/2
Request Body:

```
{
  "name": "Jane Doe",
  "position": "Senior Product Manager",
  "department": "Product"
}
```

Response:

```
{
  "id": 2,
  "message": "Employee updated successfully"
}
```

5. Delete Employee by ID (DELETE)

Endpoint: /employees/:id
Example URL: /employees/3
Response:

```
{
  "id": 3,
  "message": "Employee deleted successfully"
}
```