How to Use Github Copilot to Generate a Postman Collection File and Newman Tests Using Visual Studio Code

A tutorial for beginners who want to create and test APIs with Postman and Newman

# Introduction

Postman is a popular tool for designing, testing, and documenting APIs. It allows you to create requests, organize them into collections, and run them with different environments and variables. Postman also provides a command-line tool called Newman that can execute your collections and generate reports. Newman is useful for integrating your API tests with continuous integration and delivery pipelines, or for running them in different environments and platforms.

However, writing tests for your Postman requests can be tedious and time-consuming, especially if you have a large number of requests and scenarios to cover. Fortunately, there is a way to automate the process of generating tests for your Postman collections using Github Copilot, an AI-powered code assistant that helps you write code faster and better. Github Copilot can suggest tests for your Postman requests based on the request name, URL, method, parameters, headers, body, and response. You can then edit, customize, or accept the suggested tests and save them to your Postman collection file.

In this tutorial, you will learn how to use Github Copilot to generate a Postman Collection and Newman tests in Visual Studio Code based on a Postman collection file. You will need the following tools and prerequisites to follow along:

# Prerequisites

* A Github account and a Github Copilot subscription. You can sign up for a free trial here: https://copilot.github.com/
* A Visual Studio Code editor with the Github Copilot extension installed. You can download it here: https://code.visualstudio.com/
* A Postman desktop app or web browser. You can download it here: <https://www.postman.com/downloads/>
* A Node.js environment with Newman installed. You can install Node.js from here: https://nodejs.org/en/download/ and then run the following command in your terminal to install Newman: npm install -g newman
* Your ReportsAPI from the Github Copilot for Full Stack Development lab.

# Steps

1. Make sure you have the ReportsAPI from the previous lab open in VS Code.
2. On the left pane of VS Code, open Copilot Chat.
3. In the chat box, prompt Copilot to create a Postman collection based on the ReportsAPI. For example, you can write: **Generate a JSON schema that I can import into Postman for my ReportsAPI. The JSON schema should include methods for my getSales and getInvoices methods.**  
     
   A black background with white text

   Description automatically generated
4. Press **Submit.** You should see something like this:  
     
   A screen shot of a computer program

   Description automatically generated  
   A screen shot of a computer

   Description automatically generated
5. Hover over the suggested file and click “**Insert into New File**” to accept the code suggestion.
6. Click **Ctrl+S** to save the file and name it **postman-collection.json**.
7. Open Postman and click on the **Import** button. Choose the JSON file you just created and click on **Import**.
8. You should see the ReportsAPI collection in the left sidebar. You can expand it and see the two requests: *getSales* and *getInvoices*  
     
   A screenshot of a computer

   Description automatically generated
9. Expand **getSales**. We should see something like this:  
     
   A screenshot of a computer

   Description automatically generated
10. Notice the {{url}} in the path. Let’s revisit the suggestion Copilot gave after the postman collection file itself:  
      
    A screenshot of a computer screen

    Description automatically generated
11. To test the API, we will need to provide a value for the base URL variable. We can do this by clicking on the **Variables** tab in the collection and adding some values. For example, you can add**: url: http://localhost:5026**  
      
    A screenshot of a computer

    Description automatically generated
12. Make sure your dotnet application is running and listening on your localhost.  
      
    A computer screen with text on it

    Description automatically generated
13. Now you can select any request and click on the **Send** button. You should see the response in the right panel. You can also check the status code, headers, and body of the response.  
      
    **getSales:**  
      
    A screenshot of a computer

    Description automatically generated  
      
    **getInvoices:**  
    A screenshot of a computer

    Description automatically generated
14. You can also run the entire collection as a test suite by clicking on the **Run** button. You can configure the settings, such as the number of iterations, the delay, and the environment. You can also see the results and the logs of the test run.
15. Now that we have successfully tested our API in Postman, let’s use the Newman CLI to test it from the command line.
16. Navigate back to VS Code.
17. Open a new terminal. If you do not have newman installed, run: **npm install newman**
18. Once you have newman installed, run **newman run postman-collection.json.** This command will execute all the requests and tests in the Postman collection file and generate a report in the terminal. You can also specify different options and formats for the report, such as HTML, JSON, or JUnit. You can learn more about the Newman command-line options here: https://learning.postman.com/docs/running-collections/using-newman-cli/command-line-integration-with-newman/

A screenshot of a computer program

Description automatically generated

1. We can see from the output that our getSales and getInvoices requests failed because {{url}} was not found. Let’s use Copilot to figure out how we can fix this. Prompt Copilot with something like: **How do I pass variables into newman? I want to pass a url=“http://localhost:5026” variable into my postman-collection.json file**  
     
   A screenshot of a computer program

   Description automatically generated
2. Looking at Copilot’s response, let’s try this again. Lets run **newman run postman-collection.json –env-var “url=http://localhost:5026”**.
3. It looks like Copilot’s suggestion worked and the requests are responding with 200s as expected. We should see something like this:  
     
   A screenshot of a computer

   Description automatically generated
4. The next step is to use Github Copilot to generate tests for our Postman requests. To do so, we need to add a test property for each request object in the JSON file.
5. Open Copilot Chat and prompt it to add tests to your postman-collection.json file. You can say something like: **Add test properties to my getSales and getInvoices requests in my postman-collection.json for my ReportsAPI.**  
     
   A black background with white text

   Description automatically generated
6. Copilot will suggest something like this:  
     
   A screen shot of a computer program

   Description automatically generated A screen shot of a computer program

   Description automatically generated
7. This test script checks that the status code of the response is 200, which means OK. You can see the test result in the Postman app or in the Newman report when you run the request. **Copy** the contents of the “**event**” blocks and paste them into your **postman-collection.json** file. Click **Save**.
8. After taking into account Copilot’s response, let’s try this again. Lets run **newman run postman-collection.json –env-var “url=http://localhost:5026”**. The following newman report should look something like this:  
   A screenshot of a computer

   Description automatically generated
9. We can see that in addition to two requests being executed, we now have two additional test-scripts and assertions that have been executed.
10. Let’s shut down our localhost application in the other terminal by entering **Ctrl+C  
      
    A black background with white text

    Description automatically generated**
11. Since our application is no longer running, we will expect our tests to fail. Running the command one more time executes the test scripts, but gives us failing requests and assertions as expected:  
      
    A screenshot of a computer program

    Description automatically generated

# Conclusion

In this tutorial, you learned how to use Github Copilot to generate a Postman collection file and Newman tests using Visual Studio Code. You learned how to open the Postman collection file in VS Code, how to use Github Copilot to suggest tests for your requests, and how to save the file and run it with Newman. You can use this technique to automate the process of generating tests for your Postman collections and improve the quality and coverage of your API tests.