

Installation

Checkout Payment Gateway

Use Command Line

1. Navigate to the containing folder:

```
cd {Path}\CheckoutPaymentGatewayApi\Server\CheckoutPaymentGateway
```

2. Build the solution:

```
docker build -f Dockerfile -t checkoutpaymentgateway ..
```

3. Check the image has been created:

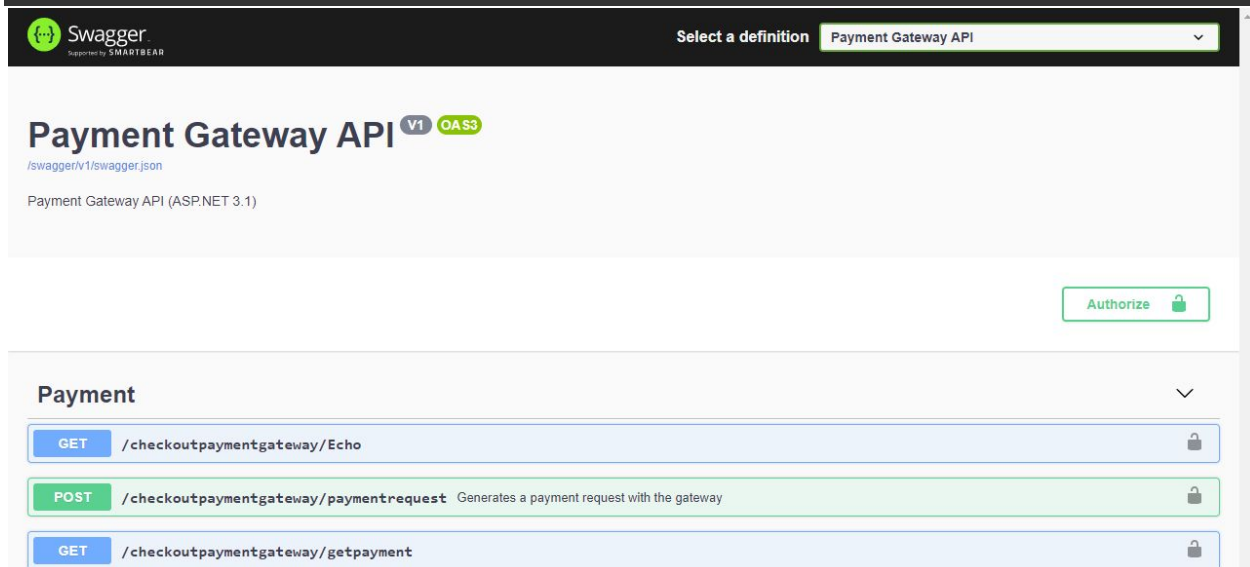
```
docker images | more
```

4. Create the docker container:

```
docker-compose up
```

5. Navigate to the Api:

<https://localhost:6001/Swagger>



The image shows the Swagger UI for the Payment Gateway API. At the top, there's a Swagger logo and a dropdown menu labeled "Select a definition" with "Payment Gateway API" selected. Below this, the title "Payment Gateway API" is displayed with "V1" and "OAS3" tags. Underneath, it says "swagger/v1/swagger.json" and "Payment Gateway API (ASP.NET 3.1)". On the right, there's an "Authorize" button with a lock icon. The main section is titled "Payment" and lists three endpoints:

- GET** `/checkoutpaymentgateway/Echo` (locked)
- POST** `/checkoutpaymentgateway/paymentrequest` Generates a payment request with the gateway (locked)
- GET** `/checkoutpaymentgateway/getpayment` (locked)

Prometheus

Use Command Line

1. Navigate to Prometheus for metrics:

```
http://localhost:9090/graph
```

2. Select the metrics to show:

```
Paymentgatewayapi_counter
```

Prometheus Alerts Graph Status ▾ Help Classic UI

☐ Enable query history ☐ Use local time ☒ Enable autocomplete

🔍 Paymentgatewayapi_counter  [Execute](#)

Table [Graph](#)

< Evaluation time >

No data queried yet

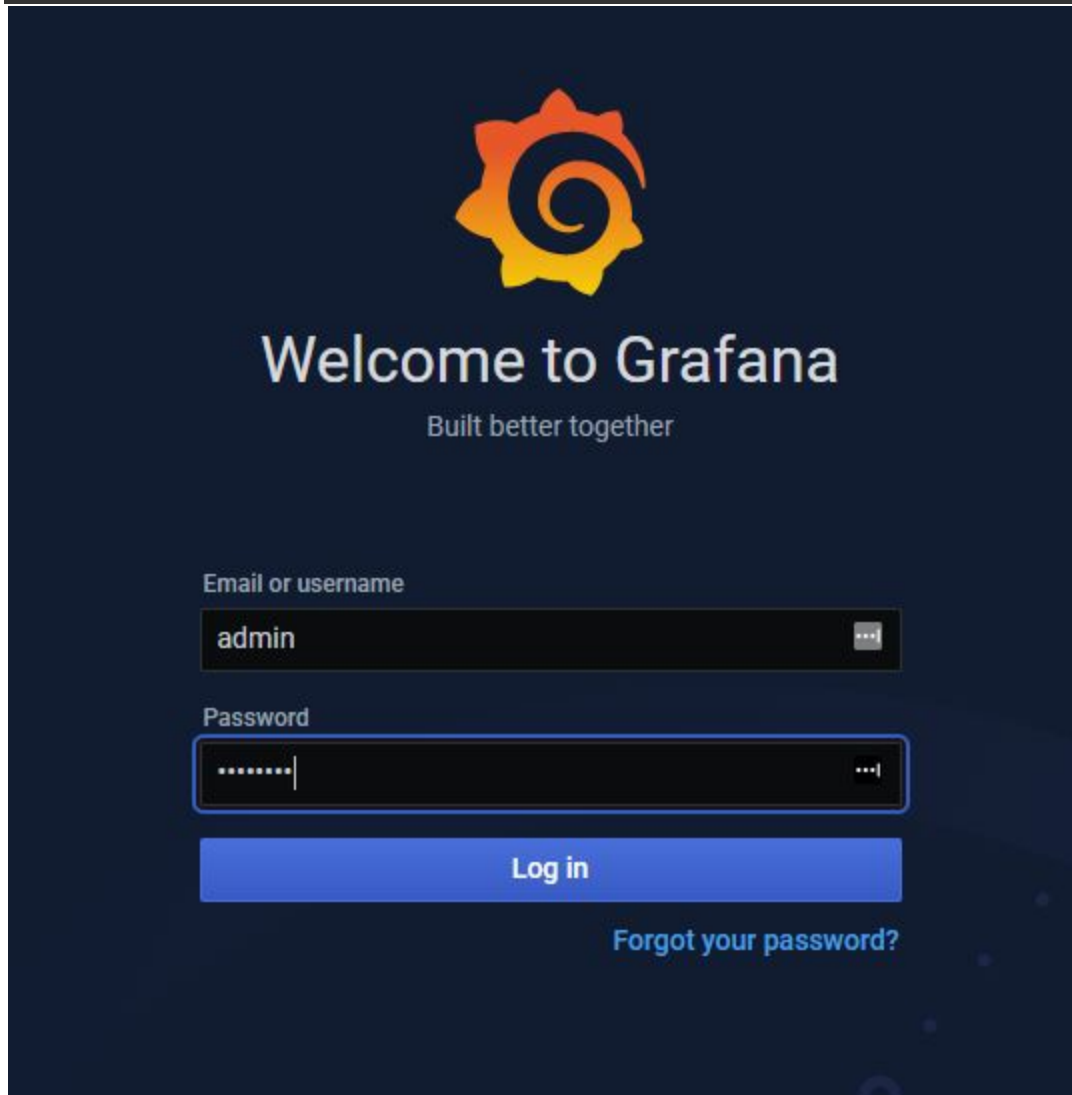
3. Click Execute

Grafana

Use Command Line

1. Navigate to Grafana:

```
http://localhost:3000/login
```



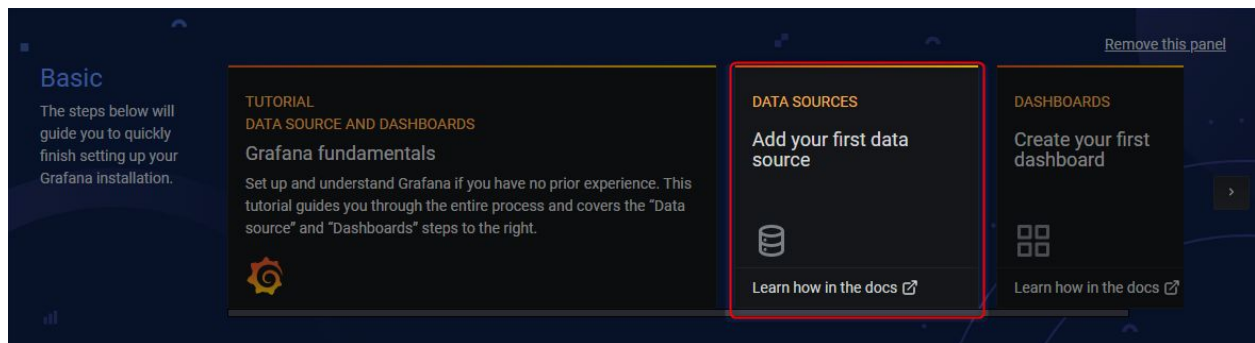
2. Log in:

```
username: admin
```

```
password: P@ssw0rd
```

3. Add Datasource

```
Click Add your first data source
```

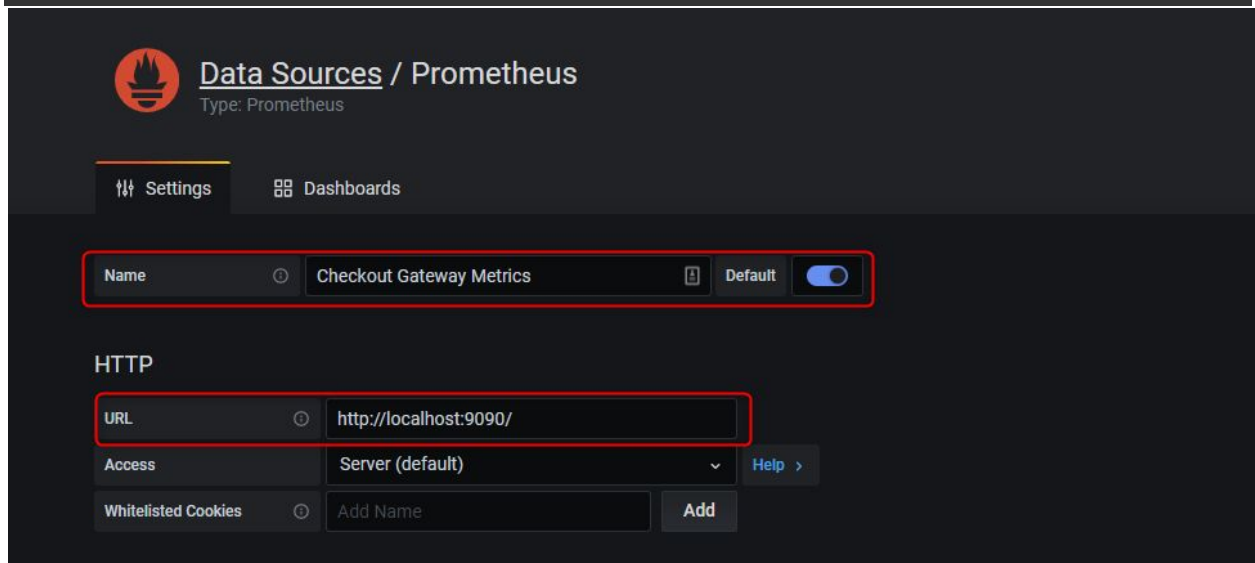


Click Prometheus



Create a useful name i.e. checkout gateway metrics

Add the URL: `http://localhost:9090/`



Click Save & Test

Misc

Disable metrics lookup ☐

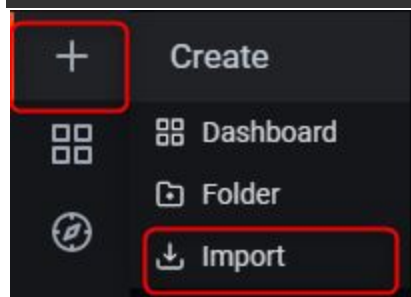
Custom query parameters

Save & Test Delete Back


4. Add Dashboard

a. Example dashboard imports:

Click the + on the left of the screen



Type the id: 10427 or the id 10915 to load preset dashboards

 **Import**
Import dashboard from file or Grafana.com

Upload JSON file


Import via grafana.com

Load

Click Load

Select the data source you created from the drop-down (above import)

Click Import


 **Import**
Import dashboard from file or Grafana.com

Importing Dashboard from [Grafana.com](https://grafana.com)


Published by	sandersaares
Updated on	2019-09-27 08:02:10

Options

Name

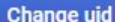
ASP.NET Core - controller summary (Prometheus) 

Folder



General 

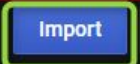

Unique identifier (uid)

The unique identifier (uid) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The uid allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

h1FE3PpWk 

prometheus-medium

 Checkout Gateway Metrics 

A dashboard can also be built from scratch

SQL Database

Use PowerShell

1. Download latest SQL server 2019:

```
docker pull mcr.microsoft.com/mssql/server:2019-latest
```

2. Run SQL Server linux docker container. **NOTE: Password: \$!DH7q94**

```
docker run -e "ACCEPT_EULA=Y" -e "SA_PASSWORD=$!DH7q94" \
-p 1633:1433 --name PaymentTechTest -h PaymentTechTest \
-d mcr.microsoft.com/mssql/server:2019-latest
```

Note this is a developer version of SQL Server.

3. Change the SA PASSWORD. New Password: \$!DH7q94

```
docker exec -it PaymentTechTest /opt/mssql-tools/bin/sqlcmd `
-S localhost -U SA -P "$!DH7q94" `
-Q "ALTER LOGIN SA WITH PASSWORD='%zYG614&' "
```

4. Connect to the SQL Server

```
docker exec -it PaymentTechTest "bash"
```

5. Connect locally to sql command:

```
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P "%zYG614&"
```

6. Publish the PaymentsDb to the container

Running

Bearer Token:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJQZXRLciI6IlB1bXB5IiwiaXhwIjoxNjk4NTc3NTI0LCJpc3MiOiJodHRwczovL2xvY2FsaG9zdDo2MDAxIiwiaXVkiJoiaHR0cHM6Ly9sb2NhbGhvc3Q6NjAwMSJ9.8_qyzWbfrw8CfAymD7gaZJk2bw83zNc_PxkC_7-8qUE
```

Swagger UI

1. Navigate to:

```
https://localhost:6001/swagger/index.html
```

2. Click Authorize

Payment Gateway API V1 OAS3

/swagger/v1/swagger.json

Payment Gateway API (ASP.NET 3.1)

Authorize

3. Add the bearer token in to the box and click Authorize then close the popup

Available authorizations ✕

Bearer (http, Bearer)
JWT Authorization header using the Bearer scheme. Example: "bearer {token}"
Value:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5In0.

AuthorizeClose

Available authorizations ✕

Bearer (http, Bearer)
Authorized
JWT Authorization header using the Bearer scheme. Example: "bearer {token}"
Value: *****

LogoutClose

NOTE: the padlocks on the API endpoints are now closed

Payment			▼
GET	/checkoutpaymentgateway/Echo		
POST	/checkoutpaymentgateway/paymentrequest	Generates a payment request with the gateway	
GET	/checkoutpaymentgateway/getpayment		

4. Click the API Endpoint you wish to test

5. Click Try it out

Payment ▼

GET /checkoutpaymentgateway/Echo

Parameters

Try it out

Name	Description
------	-------------

7. Click Execute

Payment

GET

/checkoutpaymentgateway/Echo

Parameters

Cancel

Name	Description
echo string <i>(query)</i>	<div>Testing the API</div>

Execute

Responses

8. Check the response

ExecuteClear

Responses

Curl

```
curl -X GET "https://localhost:6001/checkoutpaymentgateway/Echo?echo=Testing%20the%20API" -H "accept: text/plain" -H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJQZXNlcmVkaWkiOiwiZWxzZmJjYWNjaTNTOTBkCjc3Wi0iOjodHRwczovL2xvY2Fsag69zdDo2MDAxIiwiaWF0IjoxMjE0MTA0NDg5LnB5sh2Nhbgvc3Q6NjAwMSJ9.8_qyznbfcy8CfAynd7gaZ3K2bw83znCc_PxdC_7-8qUE"
```

Request URL

```
https://localhost:6001/checkoutpaymentgateway/Echo?echo=Testing%20the%20API
```

Server response

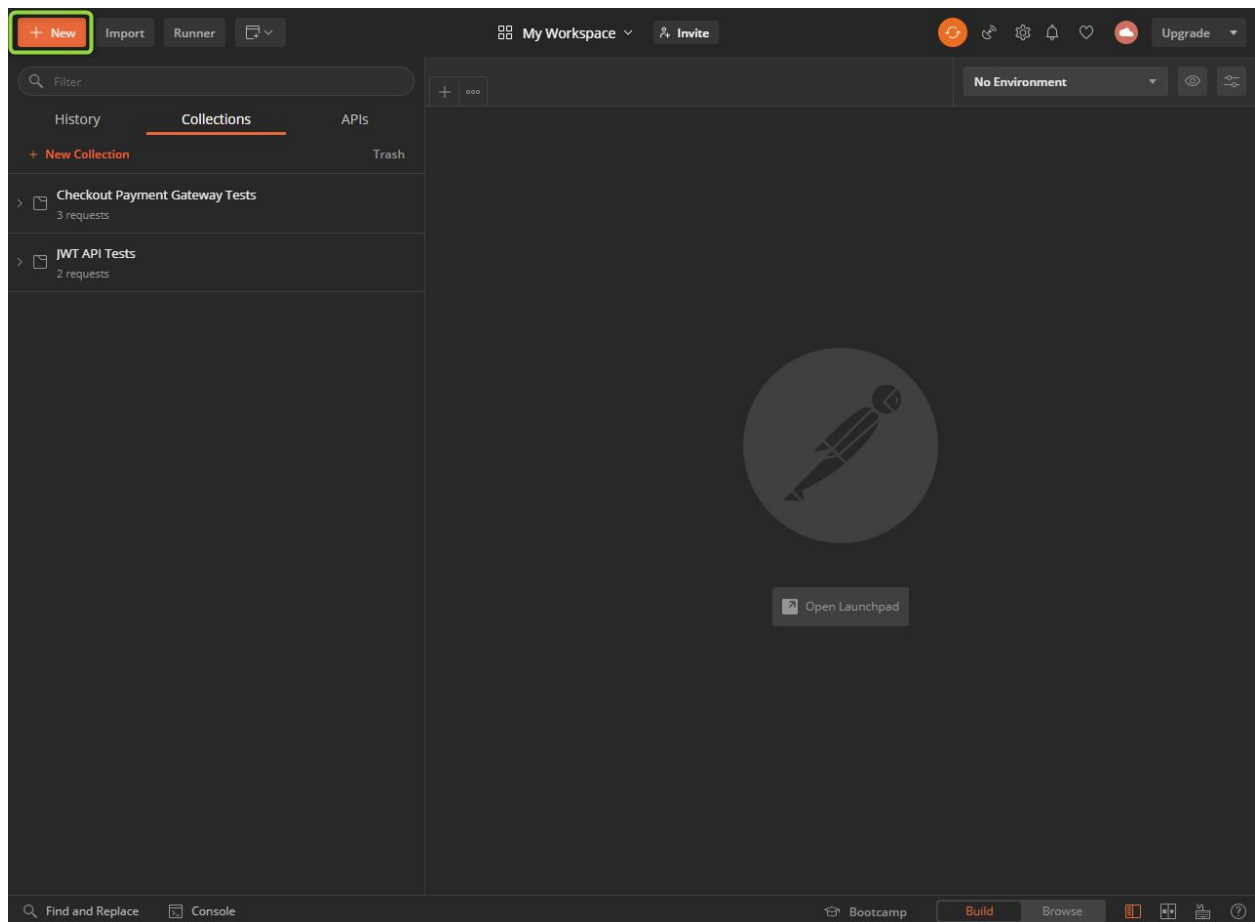
Code	Details
200	<div>Response body<p>Testing the API</p><div> Download</div></div> <div>Response headers<pre>content-type: text/plain; charset=utf-8 date: Mon30 Nov 2020 16:26:38 GMT server: Kestrel status: 200</pre></div>

Postman

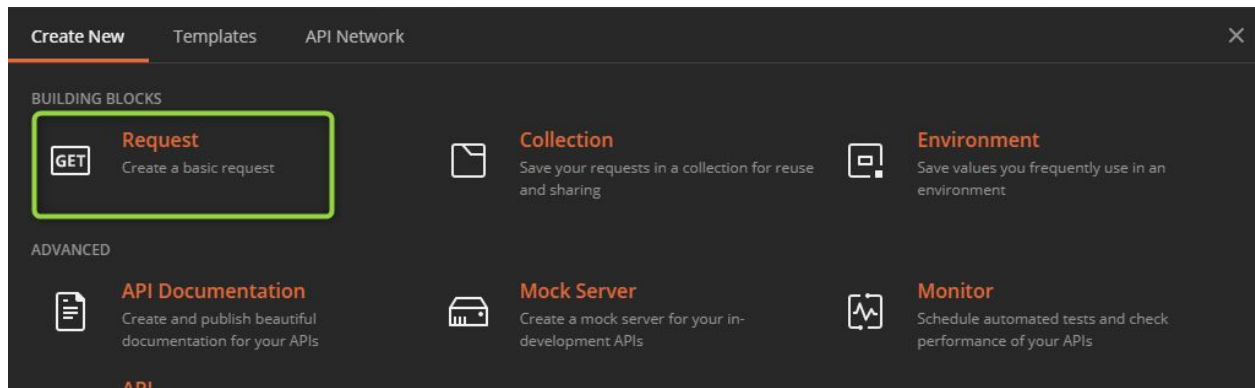
Note 1: There is a test project called **PaymentGatewayAPIClientTests** that can be run to get the status of the **Server Endpoint**

NOTE 2: The console outputs all failed payments to a JSON file to the **MyDocuments** folder. To change the location go to the **PaymentsSetup** Class and edit the **Line 79**.

1. Click New in the top left



2. Click Request



3. Create a useful name
4. Add it to a collection
5. Save to collection

SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
[Learn more about creating collections](#)

Request name

EchoTest

Request description (Optional)

Make things easier for your teammates with a complete request description.

Descriptions support [Markdown](#)

Select a collection or folder to save to:

Search for a collection or folder

◀ Checkout Payment Gateway Tests + Create Folder

GET NotComing In

GET AuthorisedName

POST PaymentSuccess

Cancel

Save to Checkout Payment Gatew...

6. Enter the endpoint URL

GET EchoTest

No Environment

▶ EchoTest

Examples 0 BUILD

GET

https://localhost:6001/checkoutpaymentgateway/Echo

Send

Params

Authorization

Headers (6)

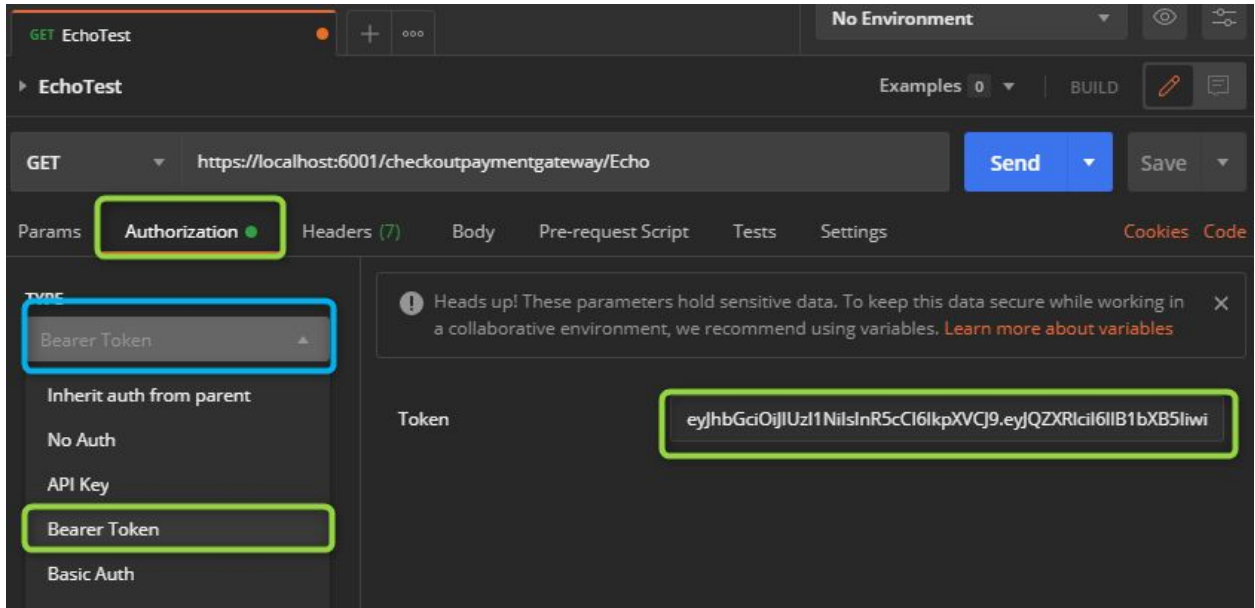
Body

Pre-request Script

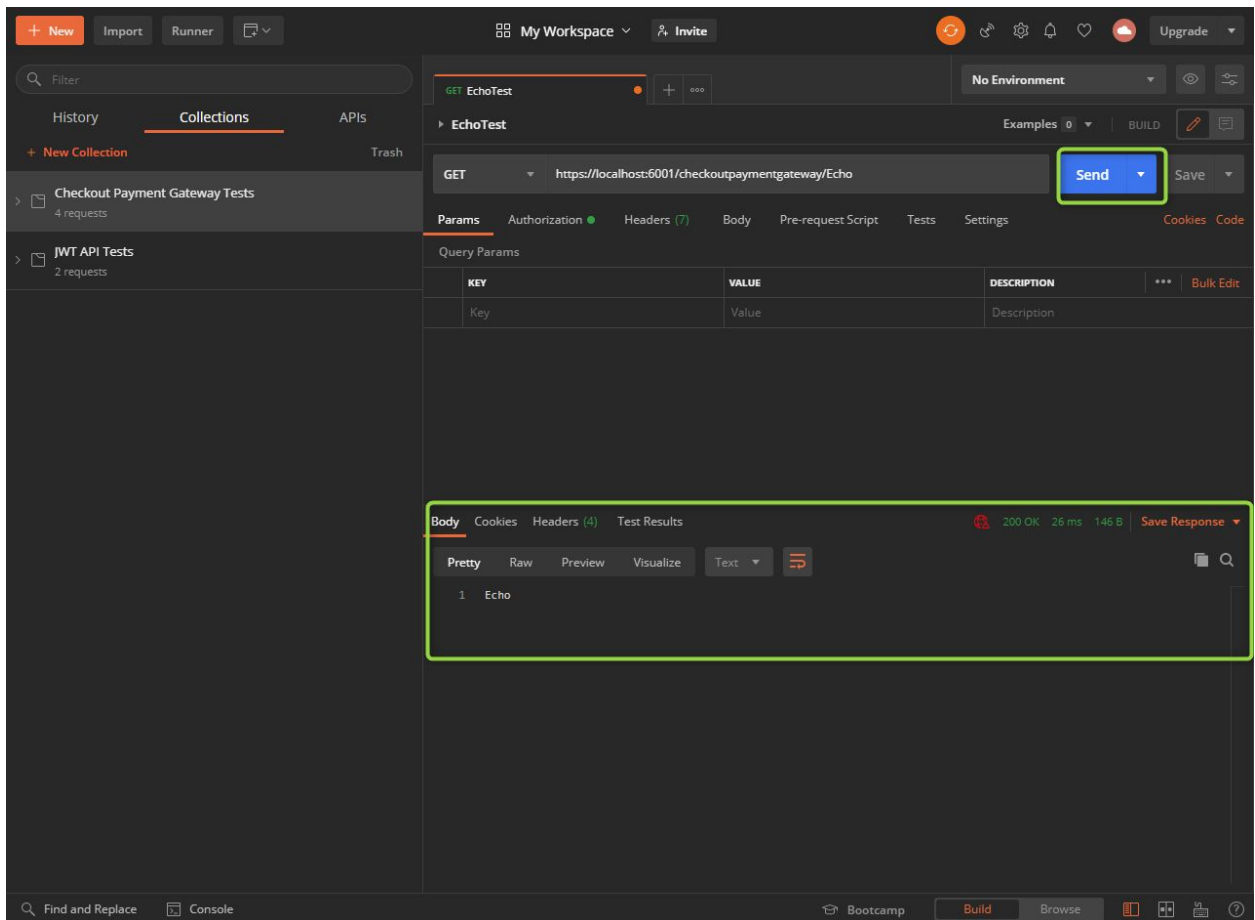
Tests

Settings

7. Click the Authorization tab
8. Select Bearer Token from the drop down
9. Enter the Bearer Token



- ### 10. Click Send to receive the string Echo



Edit the URL to:

```
https://localhost:6001/checkoutpaymentgateway/Echo?echo=TestingTheApi
```

Pass your own string value to the echo endpoint

Console App in the Client Api

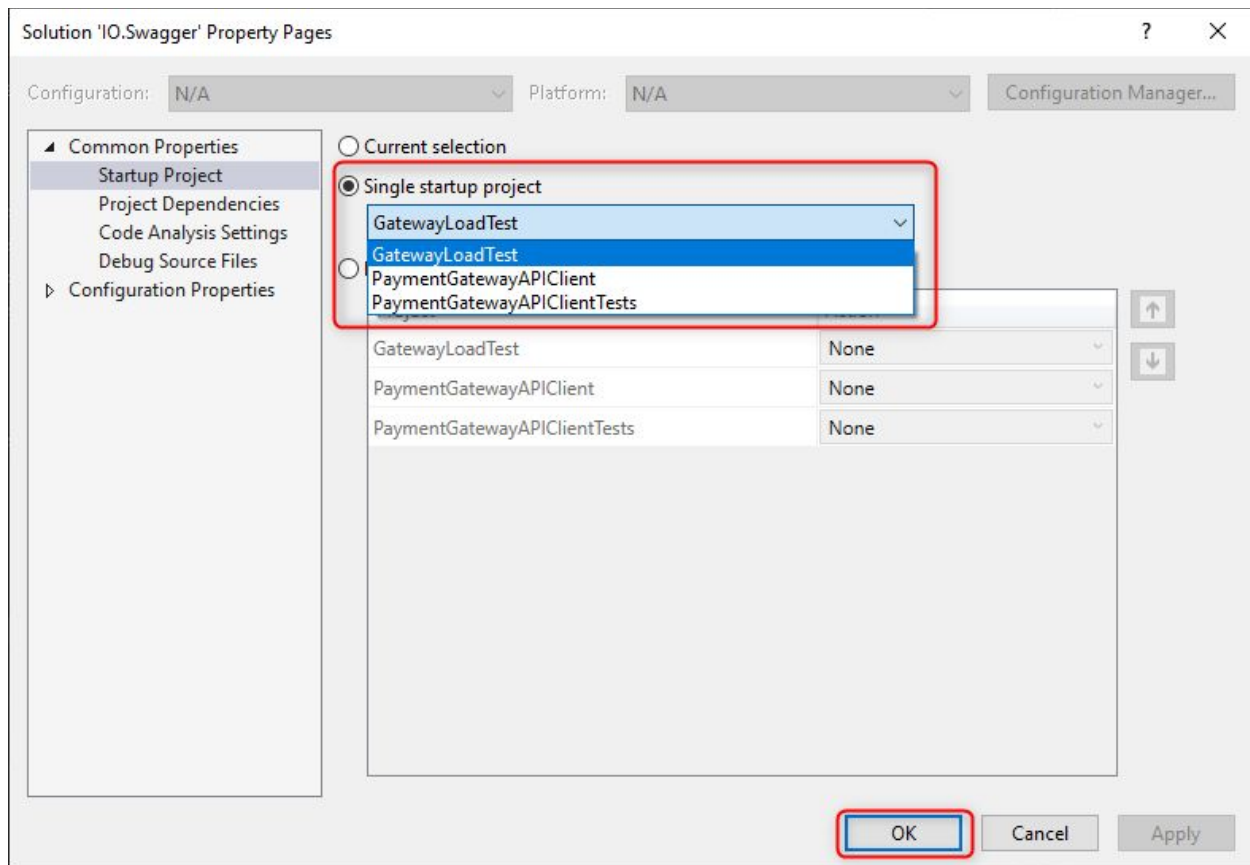
1. Navigate to the Client folder

```
{Path}\CheckoutPaymentGatewayApi\Server\CheckoutPaymentGateway
```

2. Open the solution

3. Right click on the solution and open properties

4. Set GatewayLoadTest as the start-up project



5. Press F5 to run the project

6. Type the number of payments you wish to send to the API



```
G:\Programming\CheckoutPaymentGatewayApi\Client\GatewayLoadTest\bin\Debug\netcoreapp3.1\GatewayLoadTest.exe
Enter the number of payments to generate, then press enter
10
```

7. The console will show the basic breakdown of:
- a. Total Requests
 - b. Total passing requests
 - c. Total failing requests
 - d. A file path to a JSON output file of all the failed requests