Module 4: The Modern API - Operations

Overview

Understanding the state of your infrastructure and systems is important for ensuring the reliability and stability of your services. In this module we are going to see how we can make smart operational decisions based on topology graphs and runtime performance metrics.

- Lab 1: Anypoint Visualizer
- Lab 2: Anypoint monitoring
- Lab 3: Functional Monitoring

Please proceed to Lab 1

Take me to the TOP

Lab 1: Anypoint Visualizer

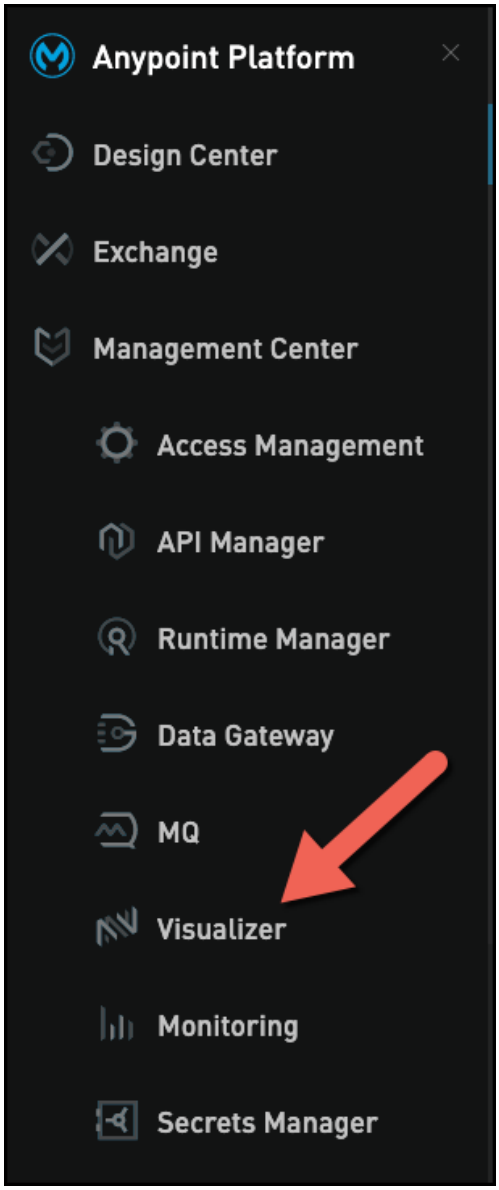
Overview

In this lab we are going to present Anypoint Visualizer and how this tool can help us reduce meantime to identification and mean time to resolution, ensuring resilient operations.

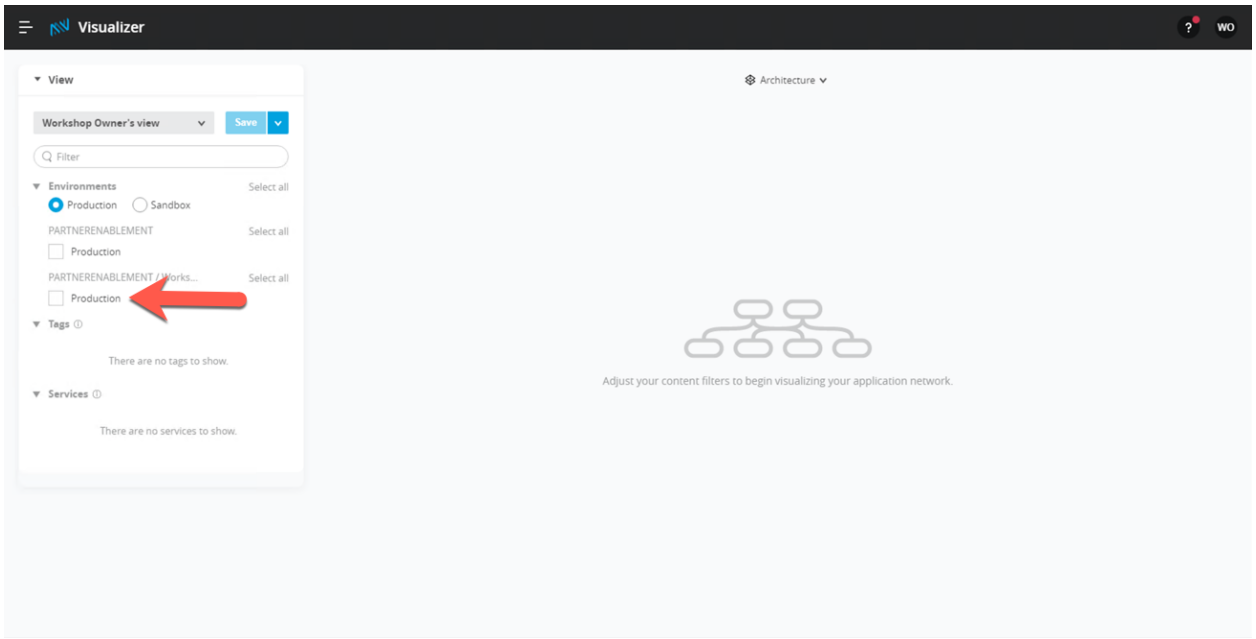
Step 1: Anypoint Visualizer View

While you develop new APIs and deploy them, they are going to appear in Visualizer and you are going to be able to see the relationship between them. For this workshop, we have already deployed APIs in production.

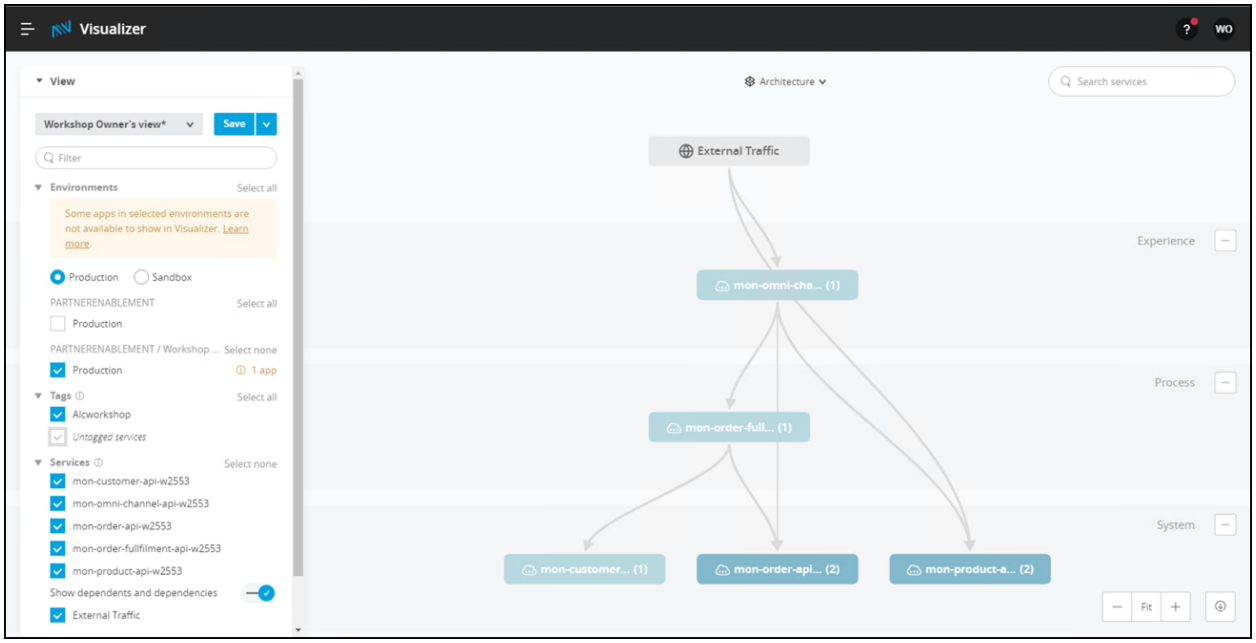
1. Go to the Anypoint Platform. Navigate to **Visualizer** on the left-side panel or from the main page of the **Anypoint Platform**


Anypoint Platform
Design Center
Exchange
Management Center
Access Management
API Manager
Runtime Manager
Data Gateway
MQ
Visualizer
Monitoring
Secrets Manager

You will find a graphic that shows the Architecture.

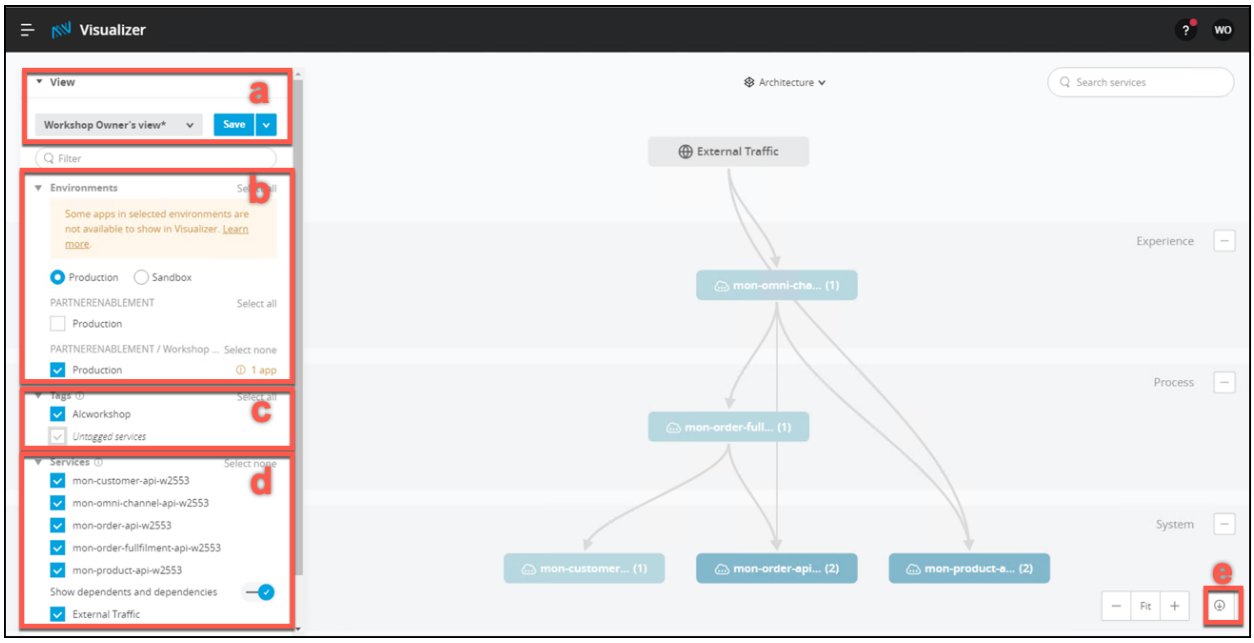


2. On the left panel, choose the production environment that corresponds to your Business Group.



3. Anypoint Visualizer provides a real-time, graphical representation of the APIs, and Mule applications that are running and discoverable. You can use the graph to explore your application network and view multiple metrics, including throughput, response time, and errors for each node in your application network. You will see how easy it is to view the APIs affected in case you need to update an API, or there is one that is failing.

On the left side of the screen you can see different filters to customize your view.



a. **Views** - This is a combo box where you can find pre saved views. You can also save the current one by pressing the save button.

b. **Environments** - You can see Production or Sandbox APIs. You can choose one or many business groups and eventually you can select all of them.

c. **Tags** - You can tag the deployed APIs so then you can filter by those tags. In this case the default deployed assets has the tag Alcworkshop. In the next section you are going to deploy an API with your own tag and you will see how the filter changes.

d. **Services** - You can filter the deployed components.

e. **Export** - Once yo have configured the View. You can export the image to a png file.

It has three use cases, or visualizations: Architecture, Troubleshooting, and Policies. By identifying your use case, you can select the visualization that suits your needs.

We are going to describe the first two.

Architecture visualization

The Architecture visualization shows the topology of your application network. In this visualization, you can change the display names of nodes, add tags, and assign nodes to layers. The Architecture visualization is most useful for documentation and presentation purposes since you can share your network's architecture without visible metrics.

1. Click on the **mon-omni-channel-api** node.

   You will see information about the service. This card is divided into the following tabs:

   a. **Details**: Provides information about the node

   b. **Customization**: Provides information about the layer to which the node is assigned and the tags assigned to the node. You can edit the node's display name or change the layer and tags from this section.

   c.

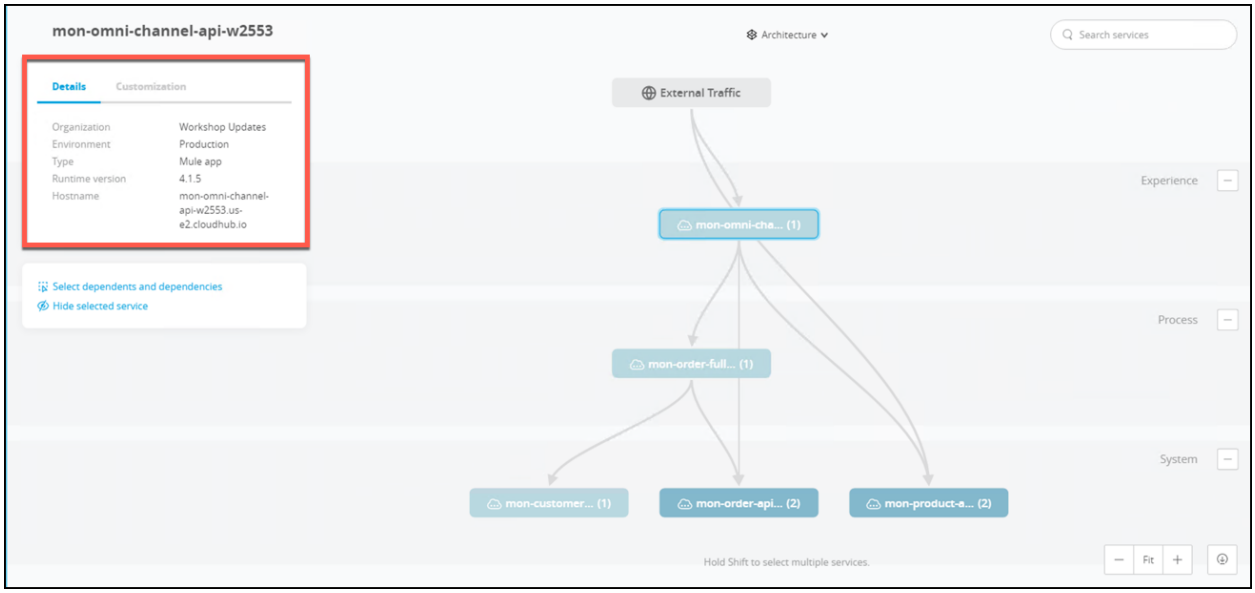


Figure 1. Selected Node

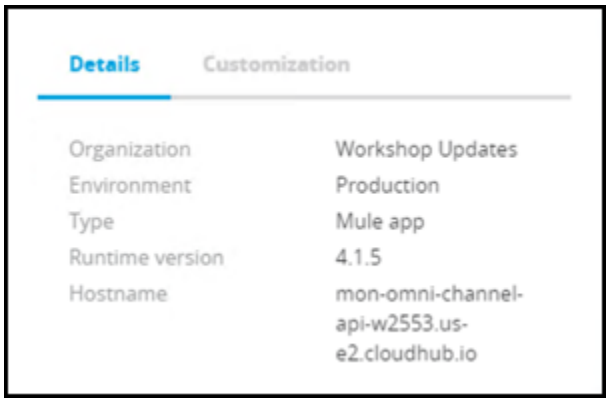


Figure 2. Detail panel

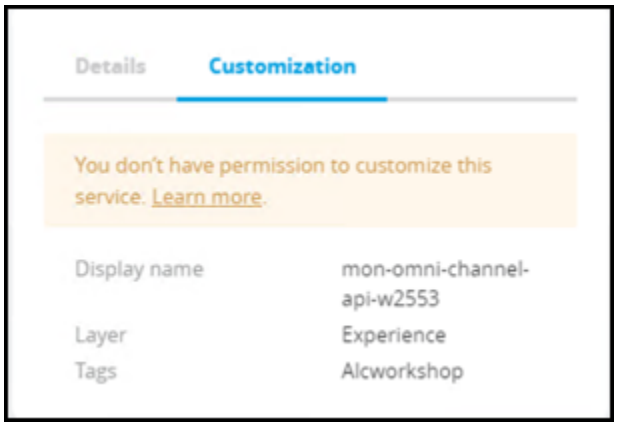


Figure 3. Customize panel

Troubleshooting Visualization

The Troubleshooting visualization enables you to view the health of the system and evaluate performance-based metrics. This visualization helps you identify issues or unusual behavior across your apps and APIs. You can see different Metrics filtered by Time Range.

The available metrics are:

- Avg Response Time
- Avg Throughput
- Failures
- Avg CPU Utilization
- Avg memory utilization

Feel free to see the different views and see how the graphic changes showing different information.

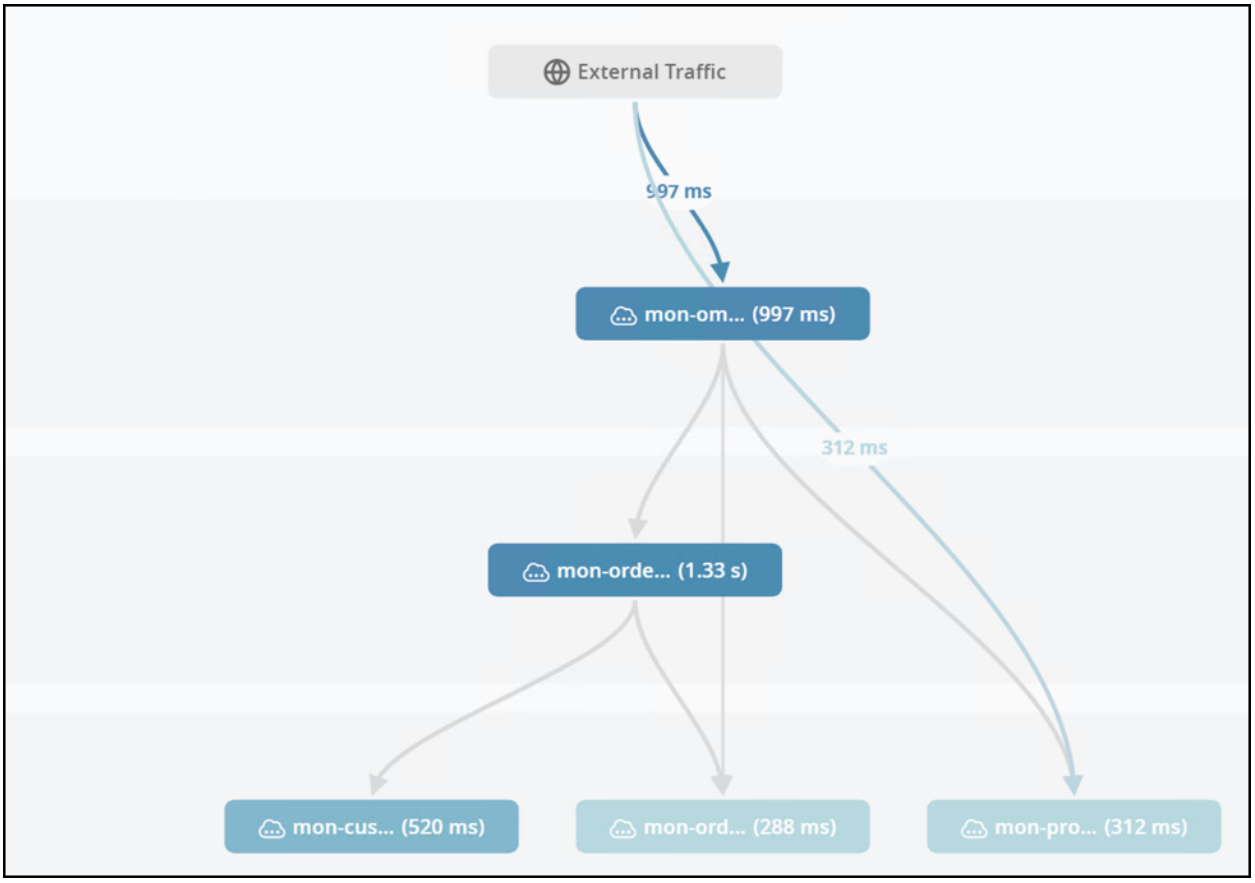


Figure 4. Avg response view

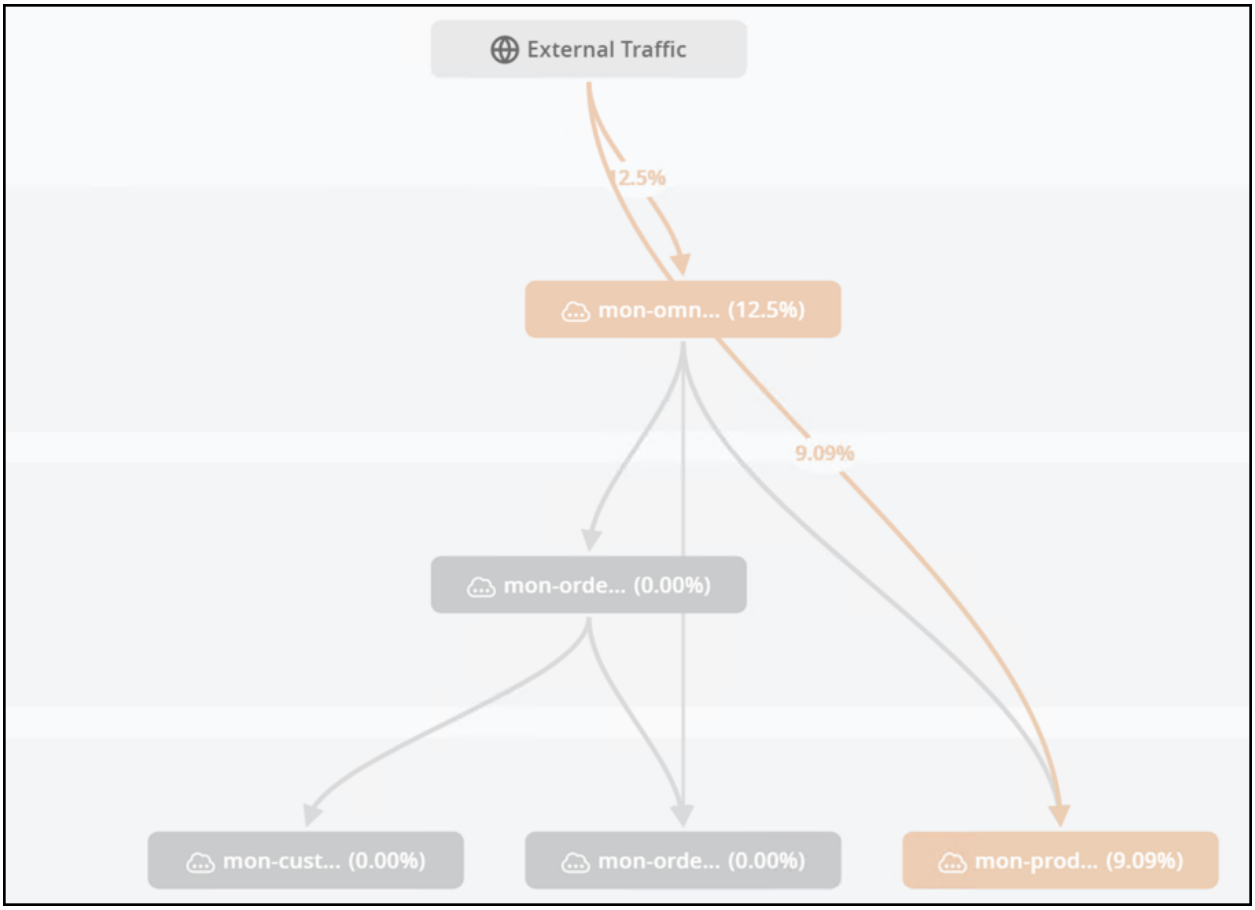


Figure 5. Failures view

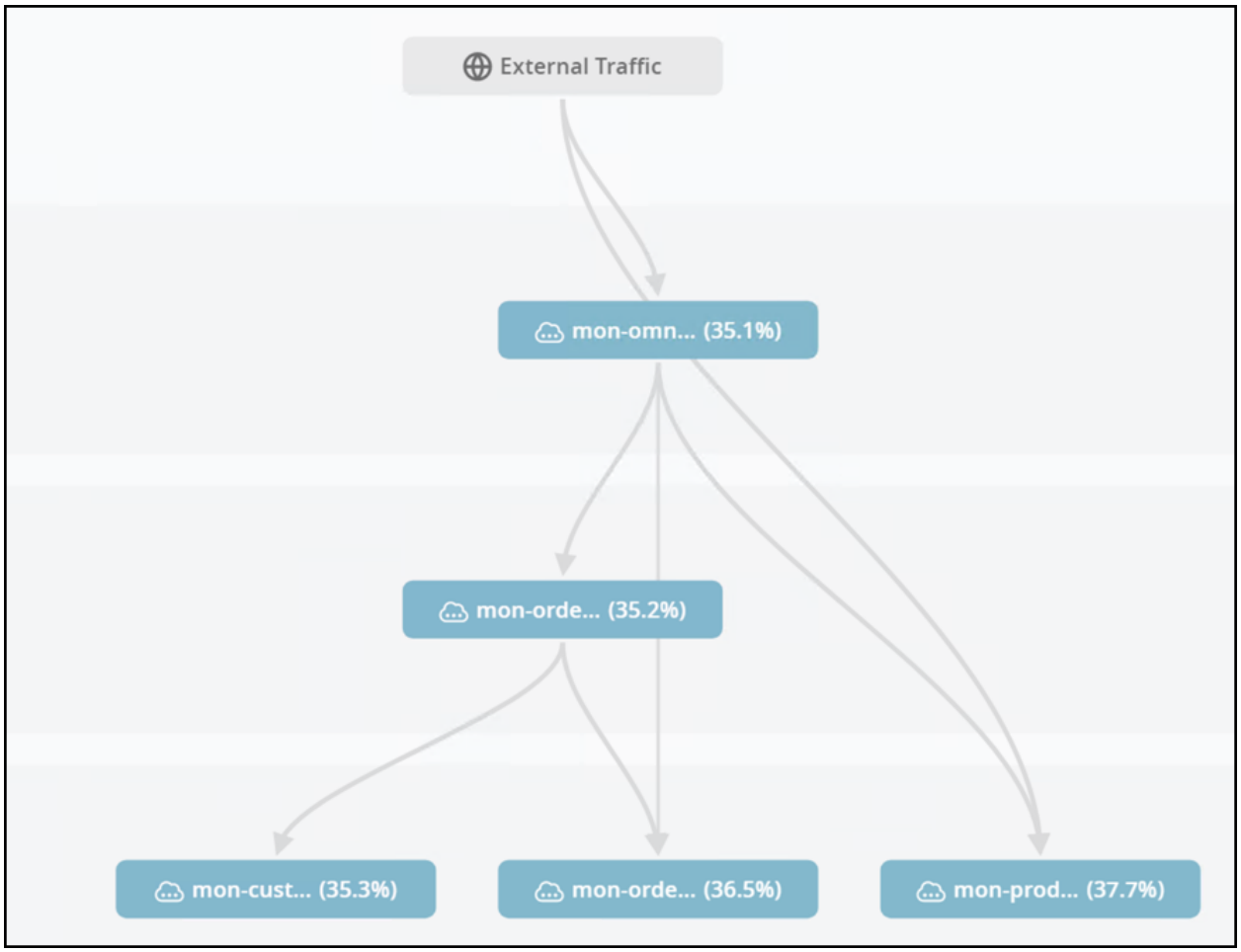
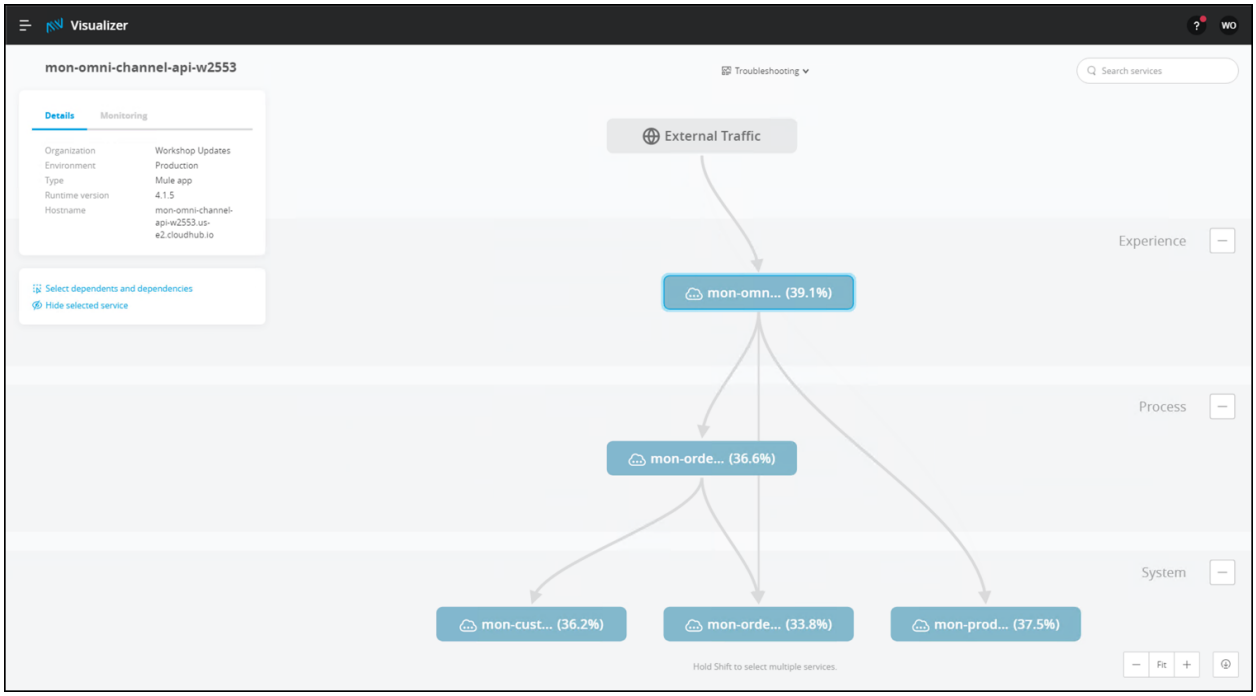


Figure 6. Avg memory utilization view

1. Click on the **mon-omni-channel-api-<#>** node.

   On the left side, you can see the details of the Omni Channel API.



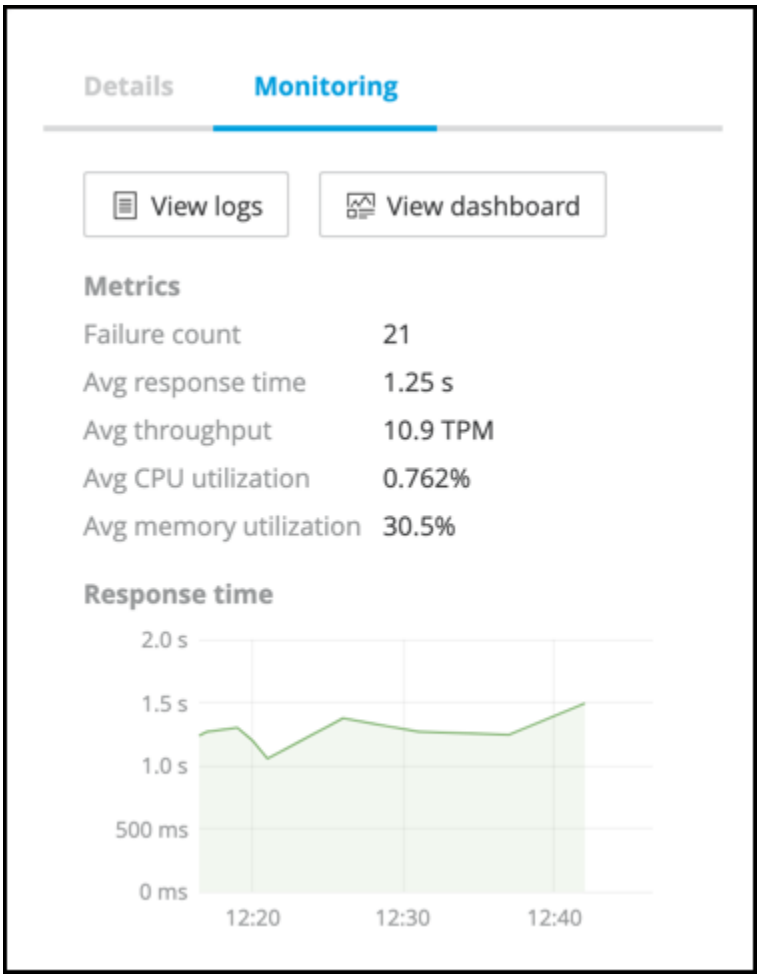
You will find two panels

- **Details**: In this section you can see general information.
- **Monitoring**: You can see specific metrics and dashboards.
  1. Click on any side of the graph outside of the Details view.
  2. In Metrics, select to **Avg response time** and click on the **mon-omni-channel-api-<id>** again.
  3. Click on the **Monitoring** tab.

     You will see two sections

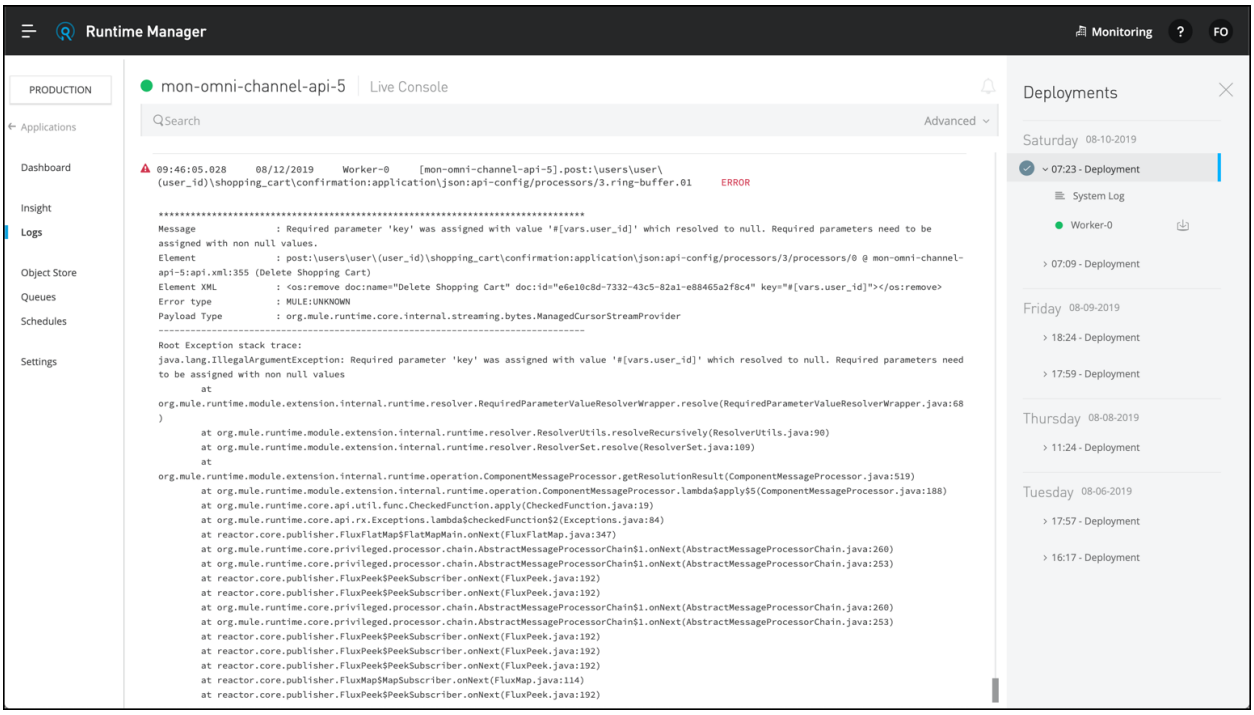
     a. **Metrics**

b. **Response time**

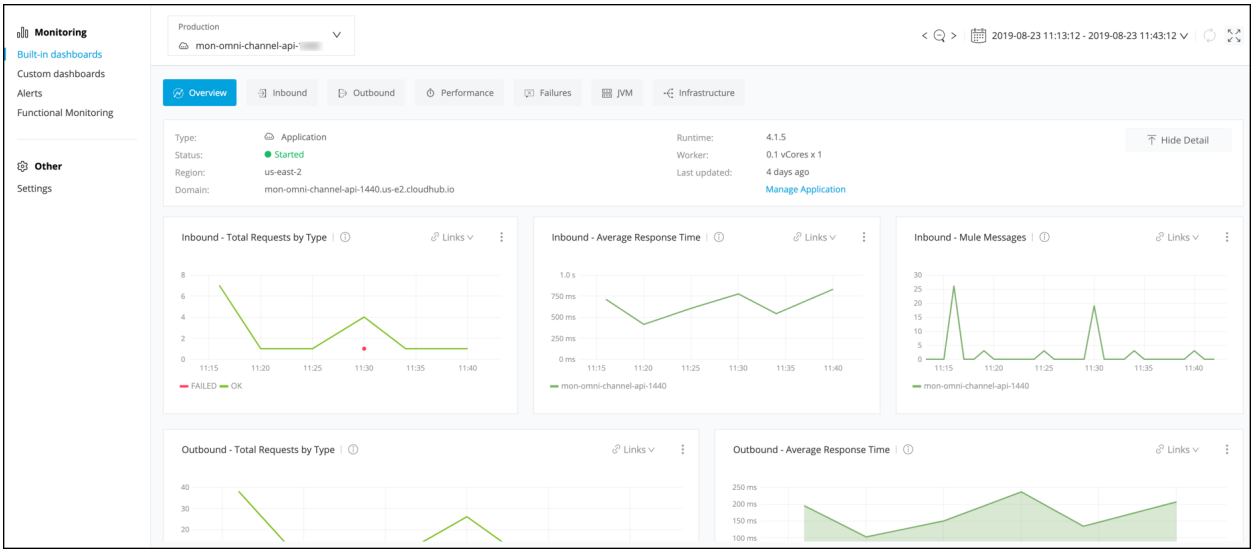


You will see two buttons:

- View Logs: this is a link to the application logs in the Runtime Manager.



- View Dashboard: this is a link to the Monitoring Dashboard.



We will cover the Monitoring Dashboard in Lab 2.

Policy visualization

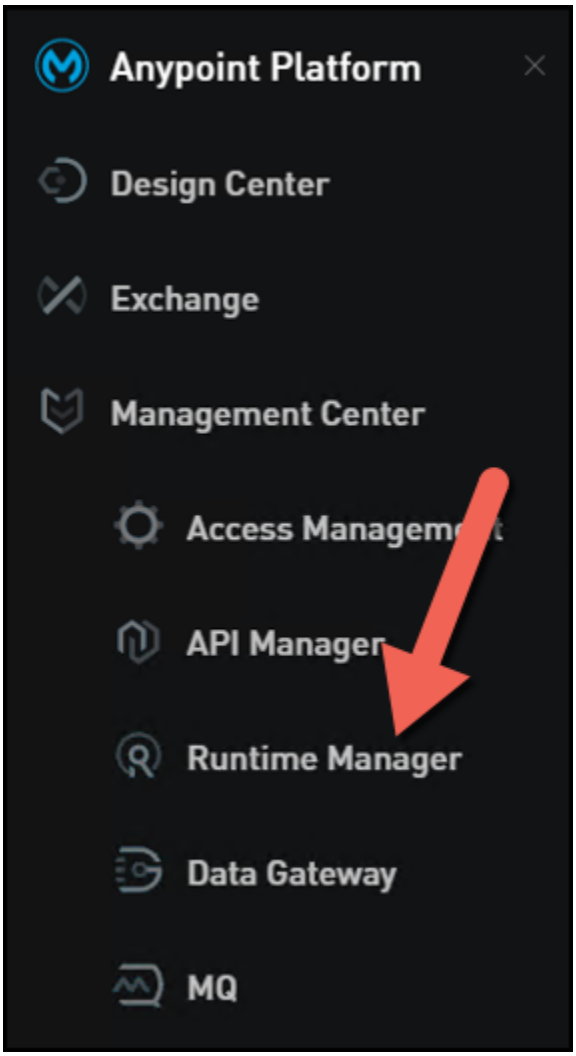
The **Policy visualization** enables you to see the policy landscape of your application network at a glance. You can view which components are compliant with policies you select, and you can determine where lapses in security or policy application may have occurred.

+ The Mule Apps deployed, don’t have Managed APIs.

Step 2: Deploy Omni Channel API

We are going to deploy another Omni Channel API to cloudhub and we are going to stress the API.

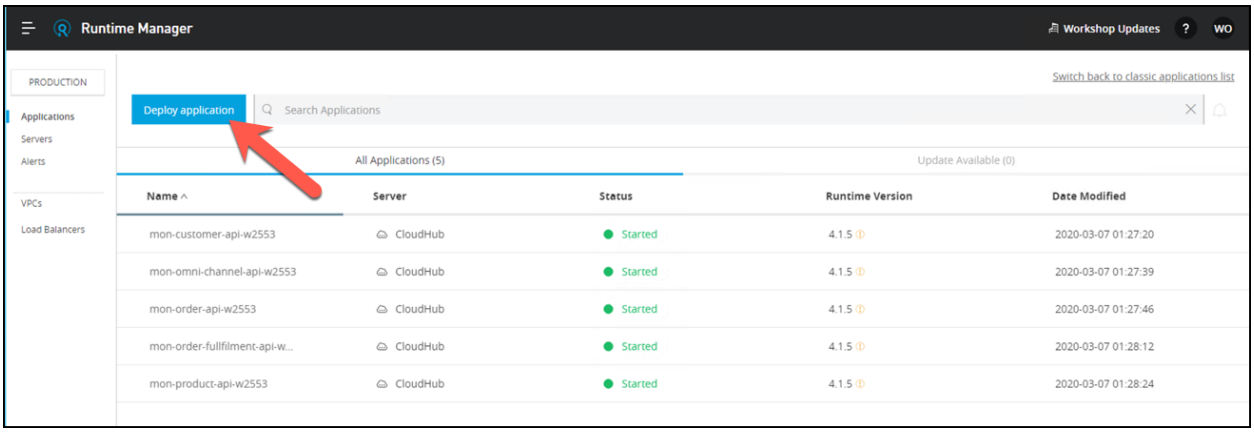
1. Go to Runtime Manager.



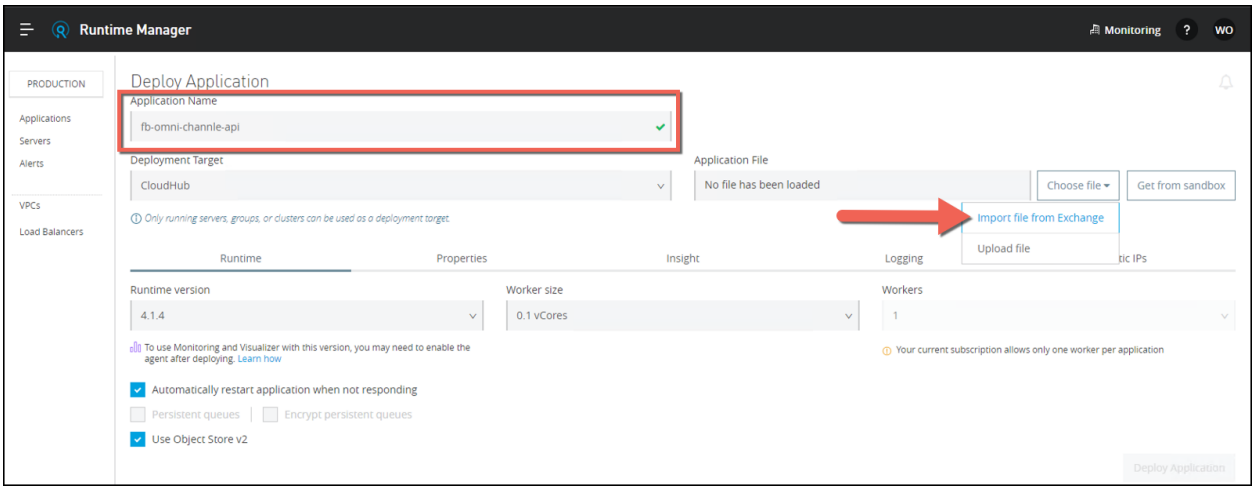
2. Select **Production** environment.
3. Take a look at the Mule applications deployed.

   All of them end with a number. Take note of that number, which is going to be needed in the next steps.

4. Press **Deploy application** Button

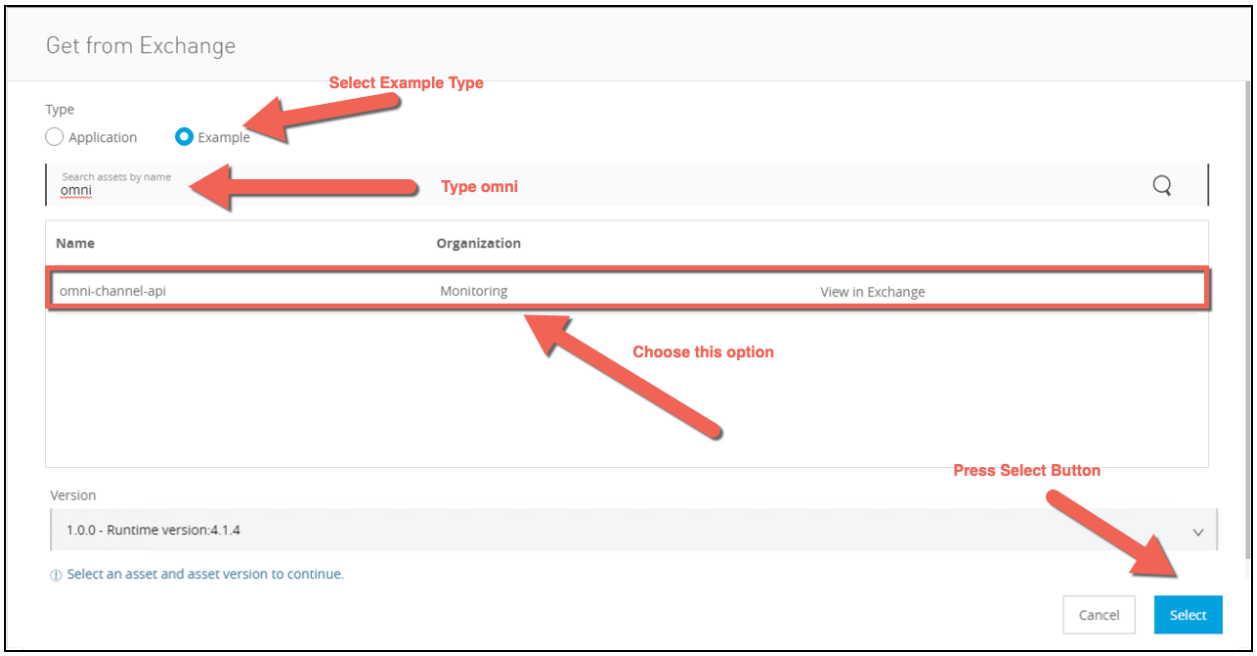


5. On the next screen, press the **Choose file** button and select **Import file from Exchange**



6. On the next page select the **Example** Type.
7. In the search asset name type `omni`
8. Select the `omni-channel-api` option.

9. Finally press **Select**



10. Complete the application name with `<name>-omni-channel-api`. Replace <name> with your initials

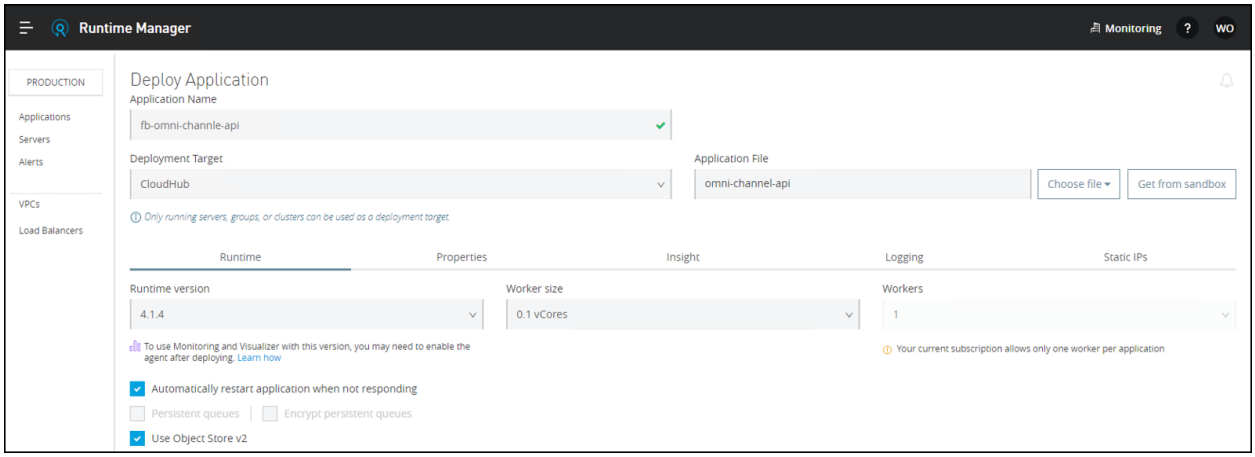
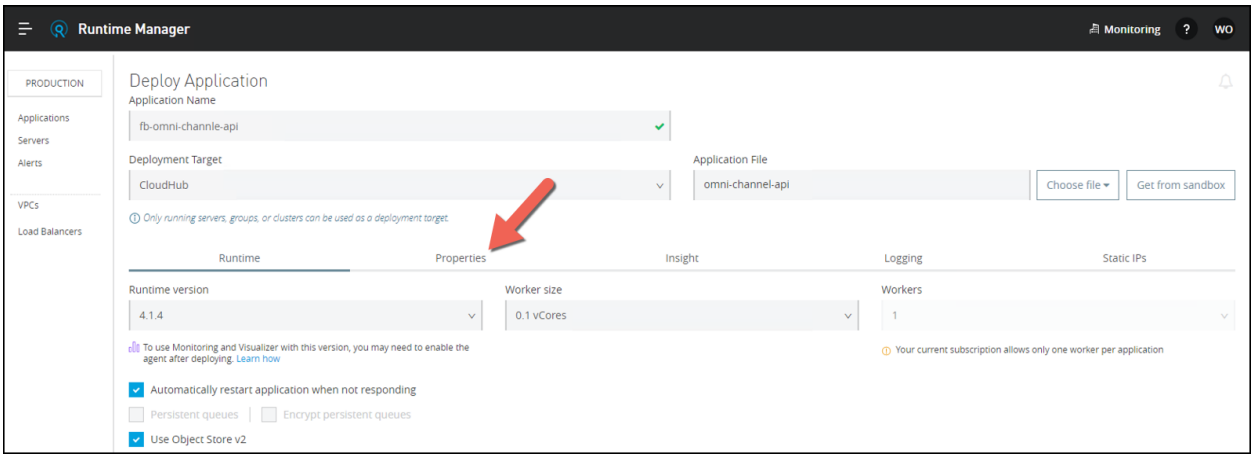


Figure 7. You should see something similar to this.

11. Go to the **Properties** Tab

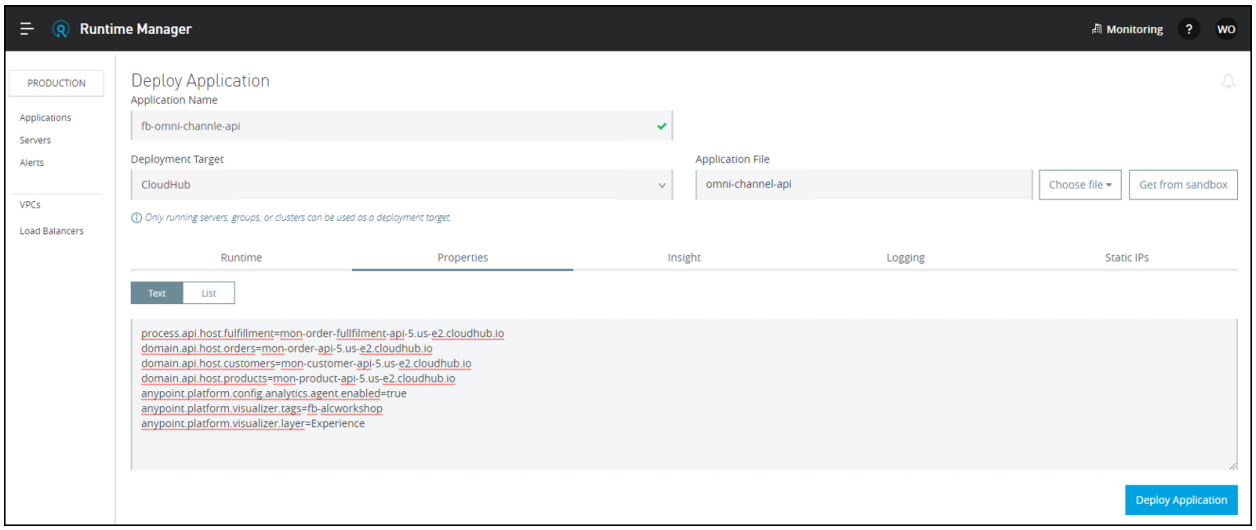


Copy and paste the following Properties

```
process.api.host.fulfillment=mon-order-fullfilment-api-<id>.us-e2.cloudhub.io
domain.api.host.orders=mon-order-api-<id>.us-e2.cloudhub.io
domain.api.host.customers=mon-customer-api-<id>.us-e2.cloudhub.io
domain.api.host.products=mon-product-api-<id>.us-e2.cloudhub.io
anypoint.platform.config.analytics.agent.enabled=true
anypoint.platform.visualizer.tags=<name>-alcworkshop
anypoint.platform.visualizer.layer=Experience
```

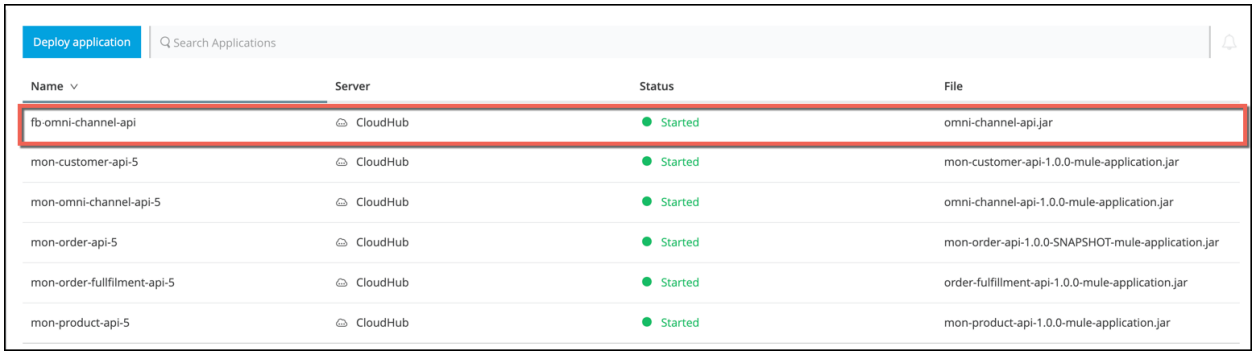
Replace <id> with the number you took note of at the beginning
Replace `<name>` with your initials in the **anypoint.platform.visualizer.tags** property.

12.

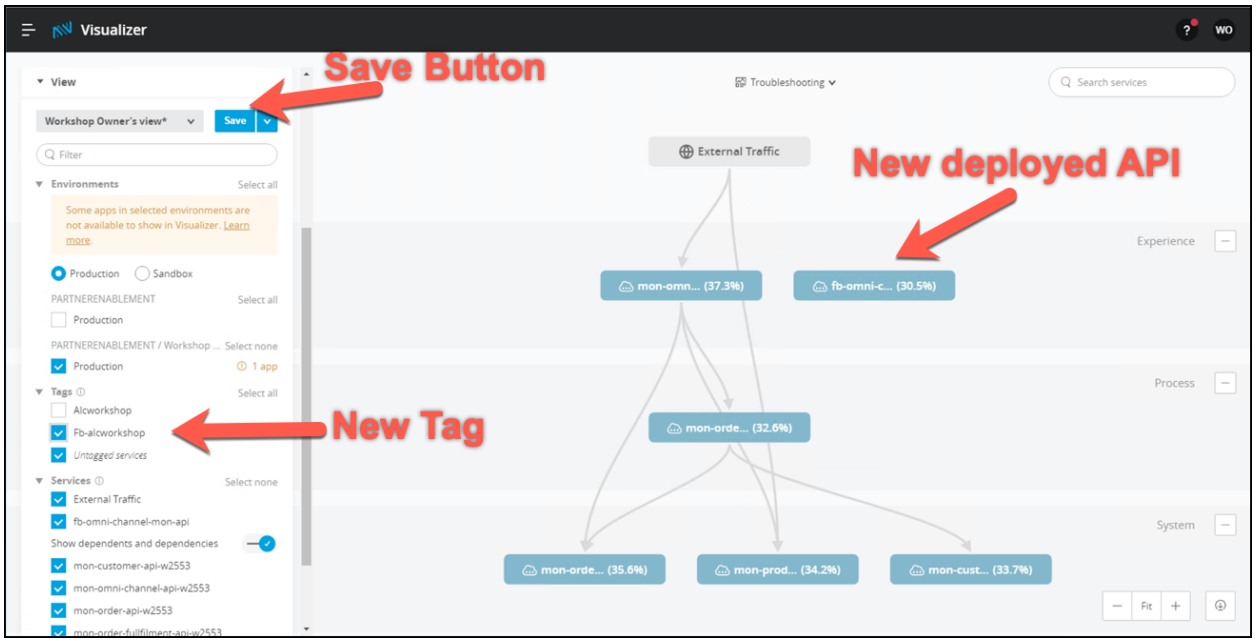


13. Press **Deploy Application** button.

    Once the application is deployed and started.



14. Go back to Anypoint Visualizer
    You should see the new deployed application.



15. To save the view with the applied filters, Press the save button.

In the next step we are going to stress this API and see the metrics online.

Step 3: Stress the API

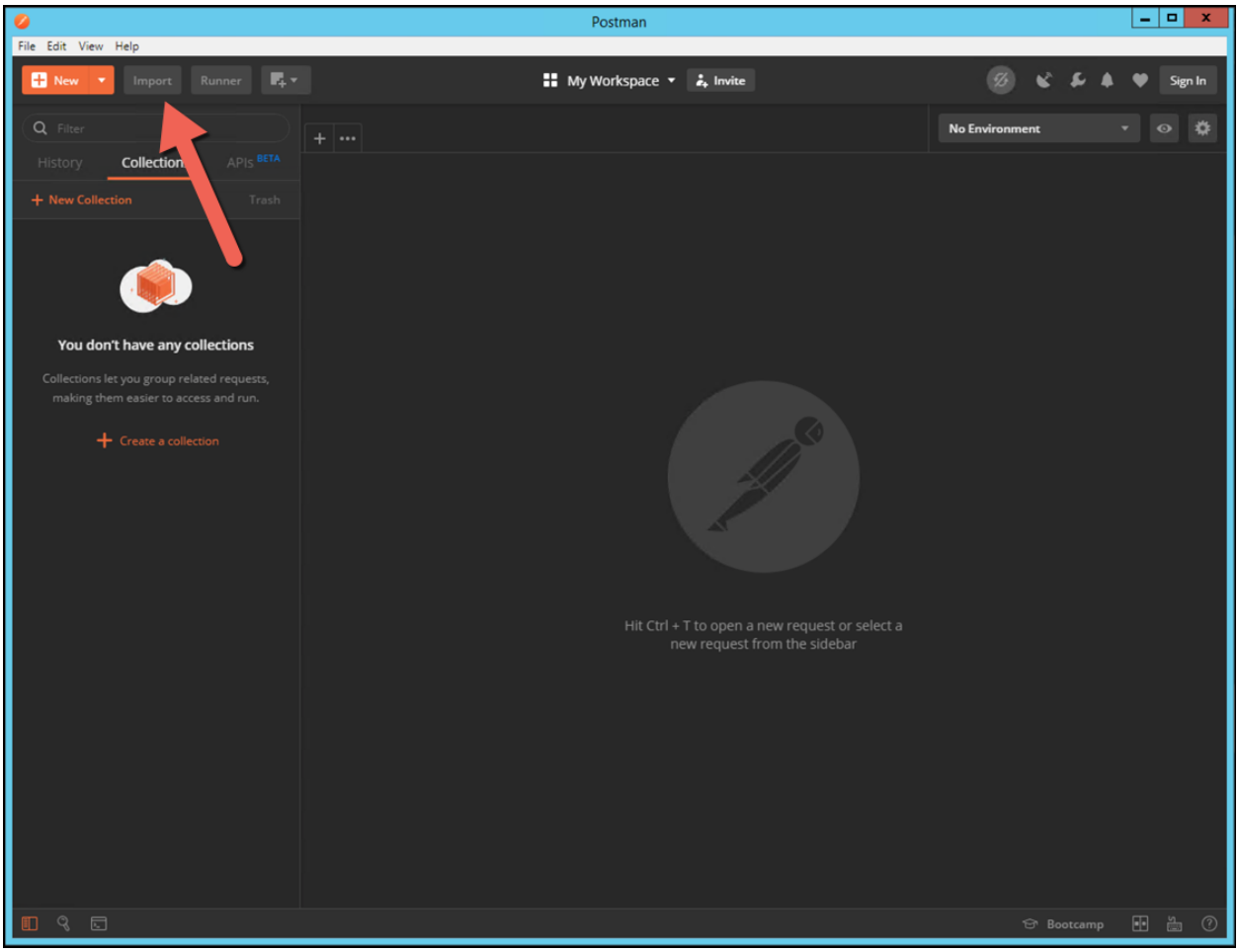
In this step you are going to stress the API already deployed and see how the metrics change over time.

We are going to import a collection of requests in Postman and run it several times.

If you don't have Postman installed, you can get it from here

1. Download the postman scripts from here
2. Double-click to open the zip. Inside, you will find two files:
    - omni-channel.postman_collection.json
    - alc-workshop.postman_environment.json
3. Open Postman

4. Press the **Import** button on the top.



5. A new screen will open. Import the file `omni-channel.postman_collection.json`.

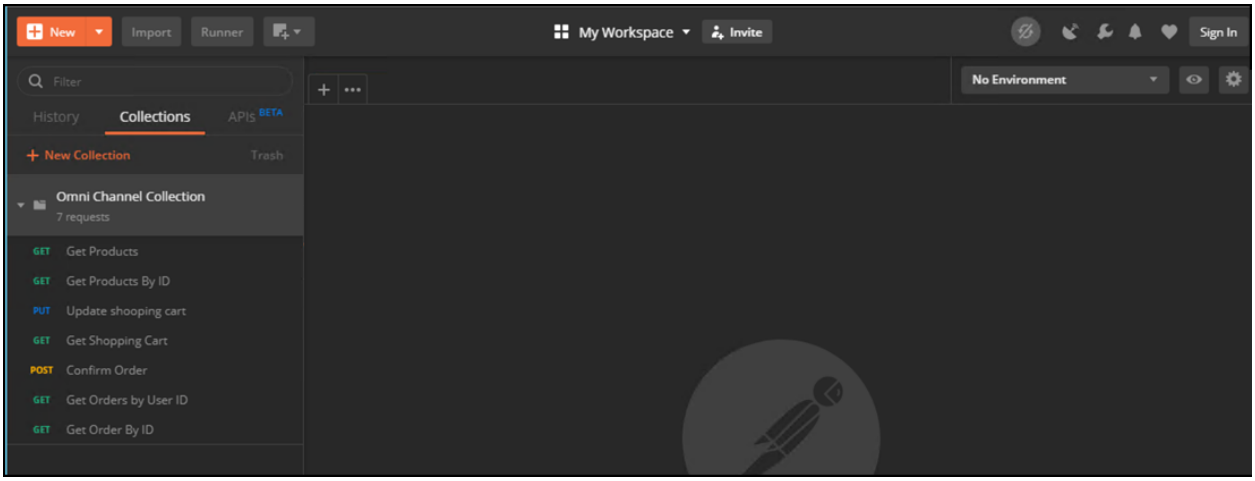
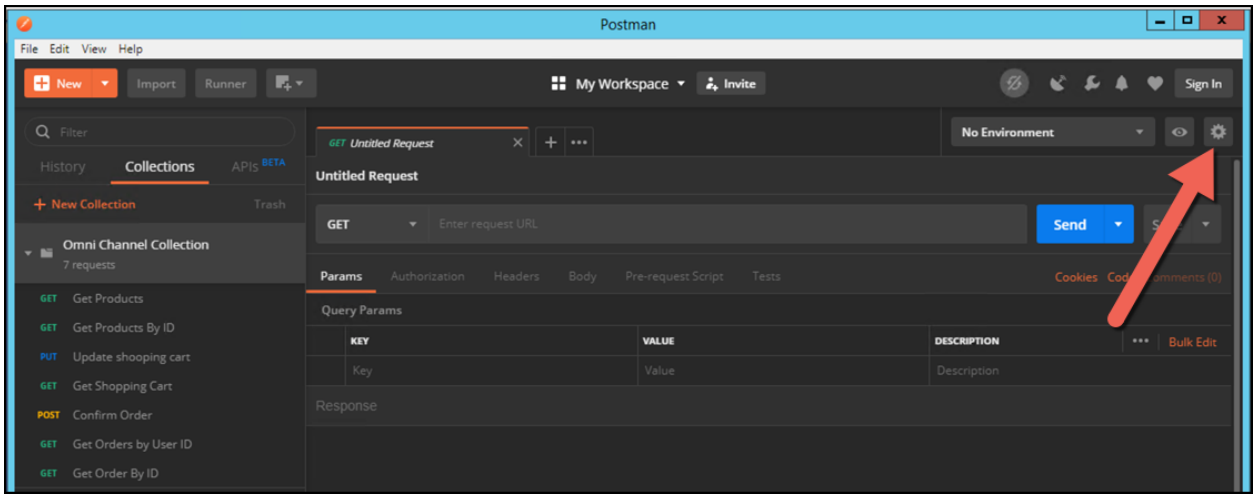
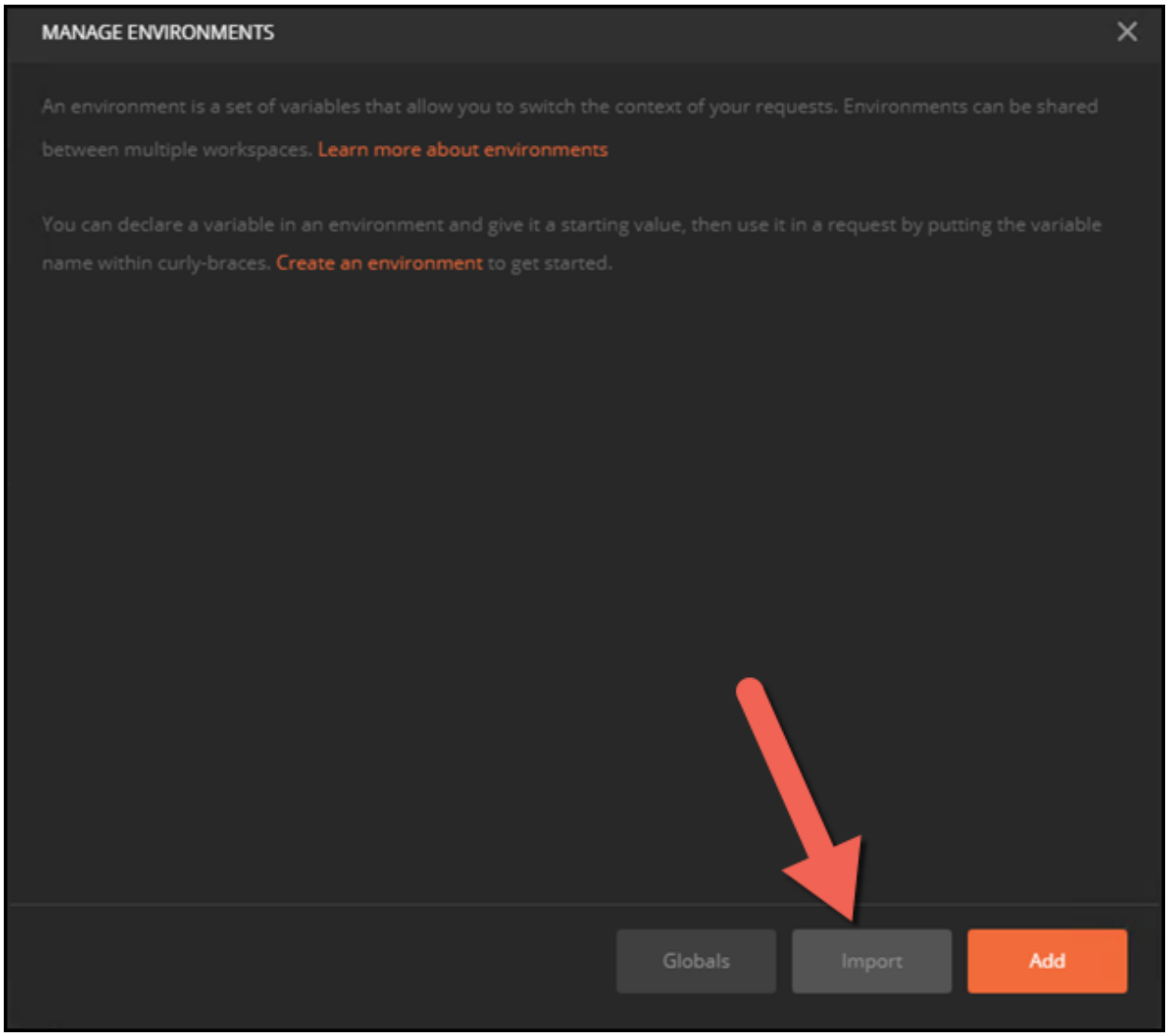


Figure 8. You should see 7 Requests imported on the left.

6. Now we are going to import an environment variable. Press the wheel on the top right.

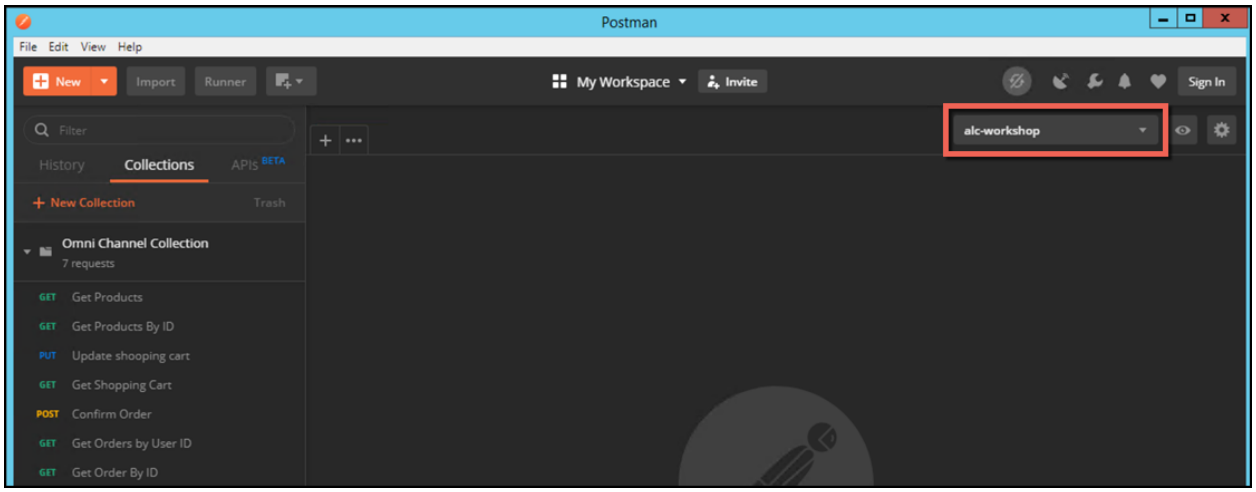


7. Click on the **Import** button.

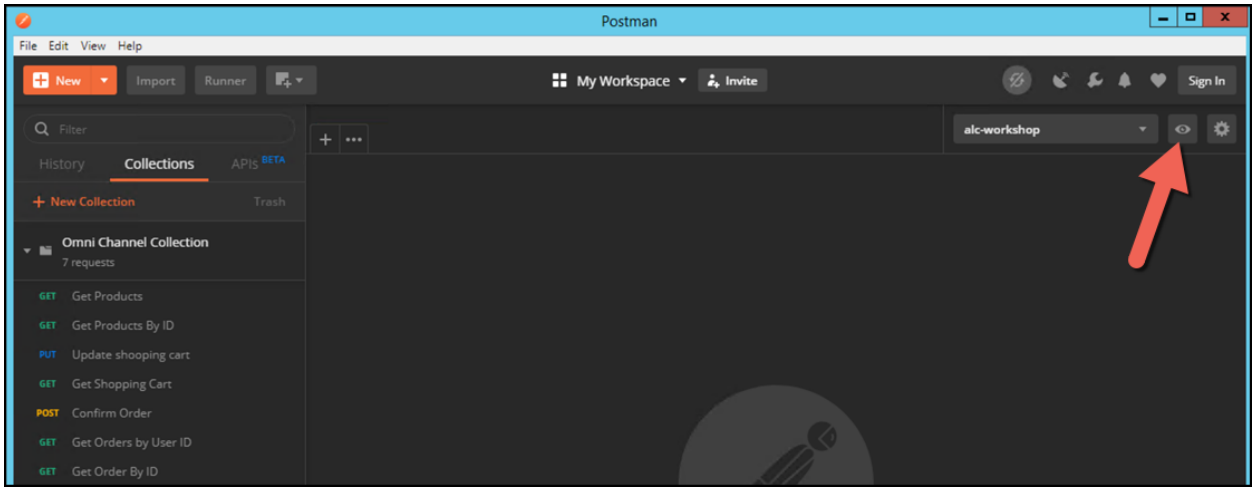


8. Choose the `alc-workshop.postman_environment.json` file.
9. Click **Add**.

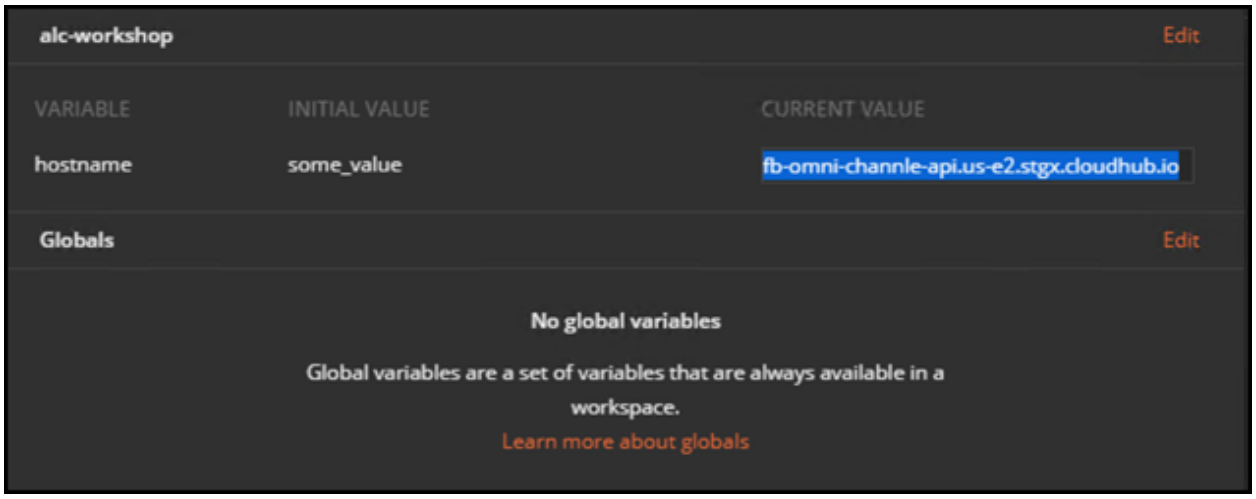
10. Now choose the alc-workshop variable from the drop down list to use the environment variable included there.



11. Click on the eye on the right of the drop down list.

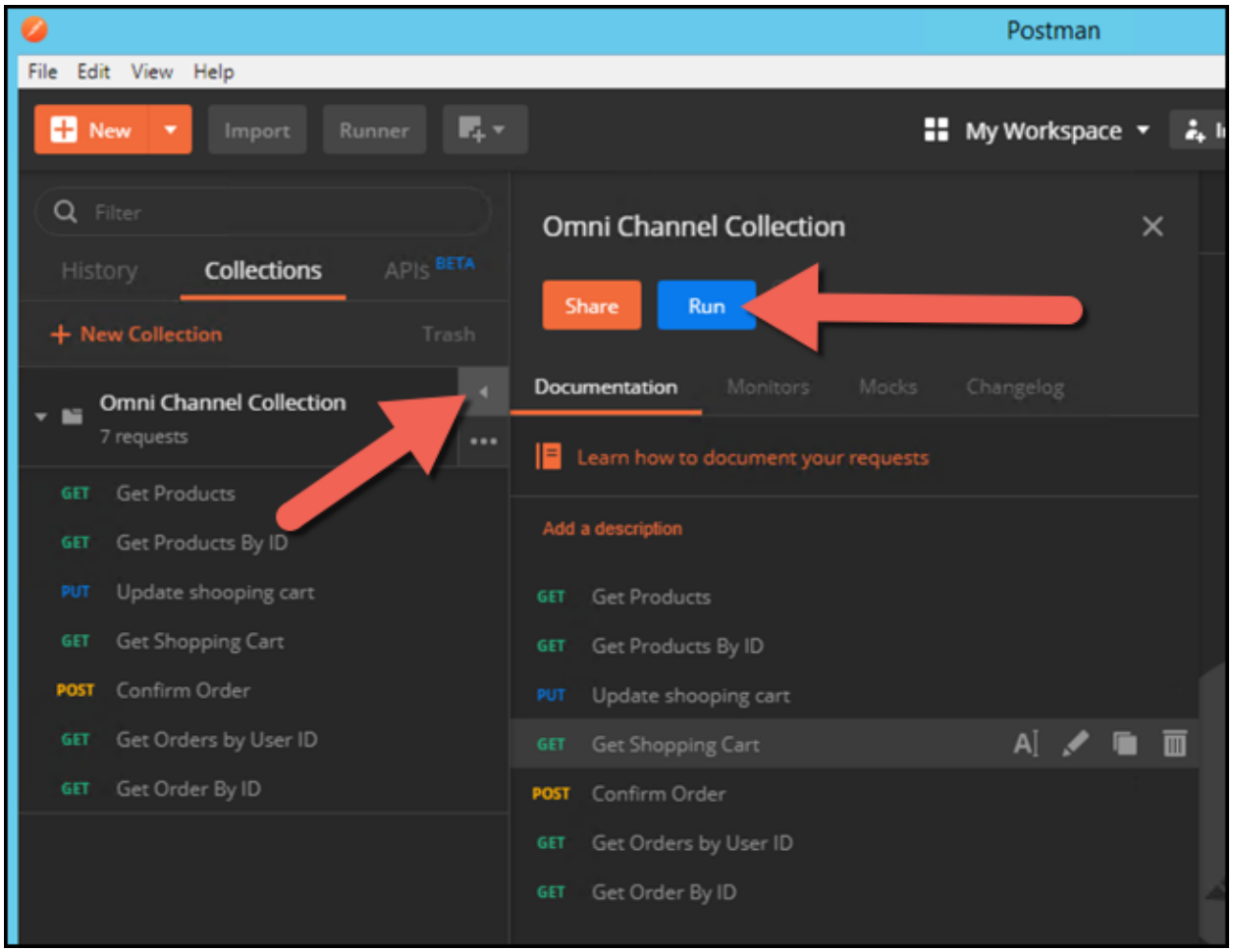


12. Configure the current value.



13. In the Collection Name, press the right arrow.

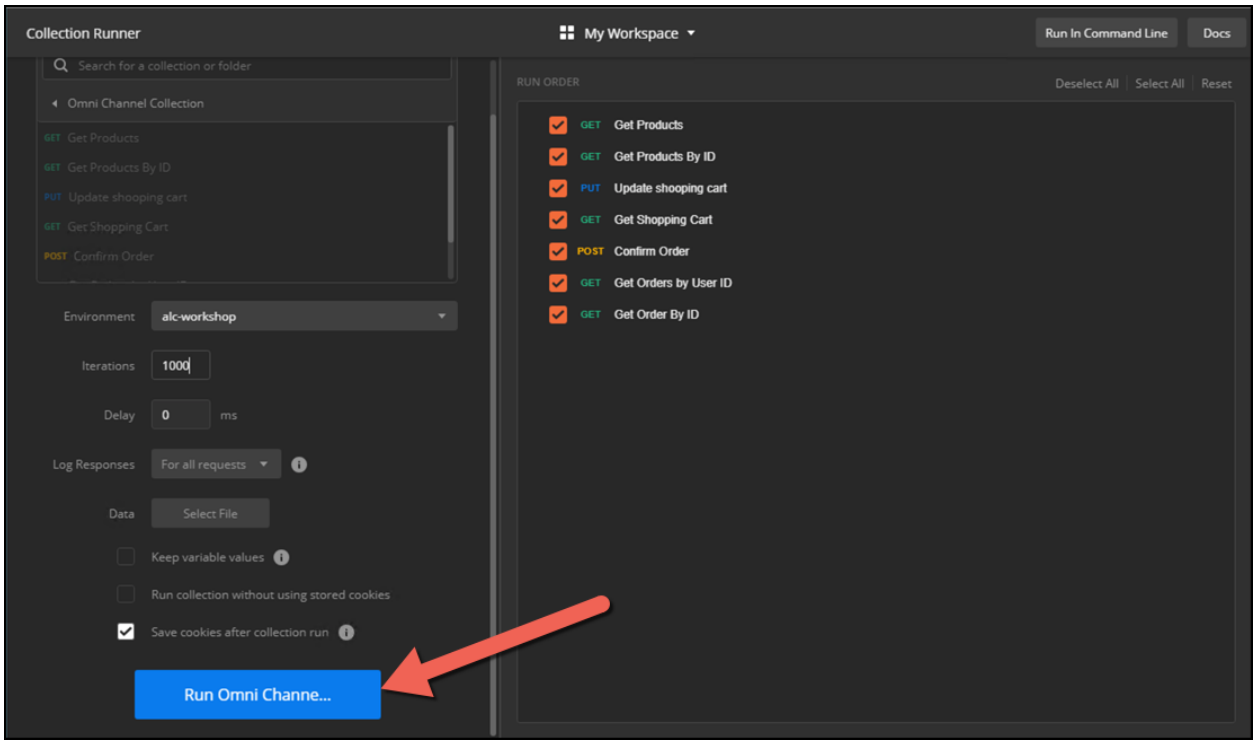
14. Press the **Run** button.



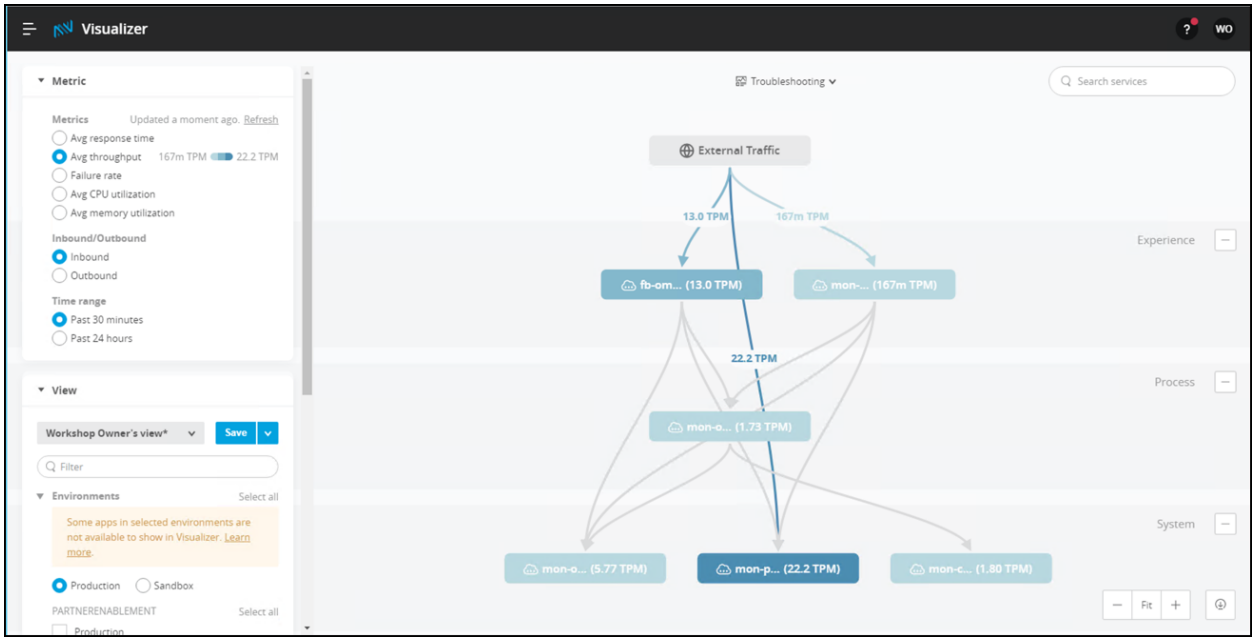
15. A new window will open. Complete with the following options.

Name	Value
Environment	alc-workshop
Iterations	1000
Delay	0

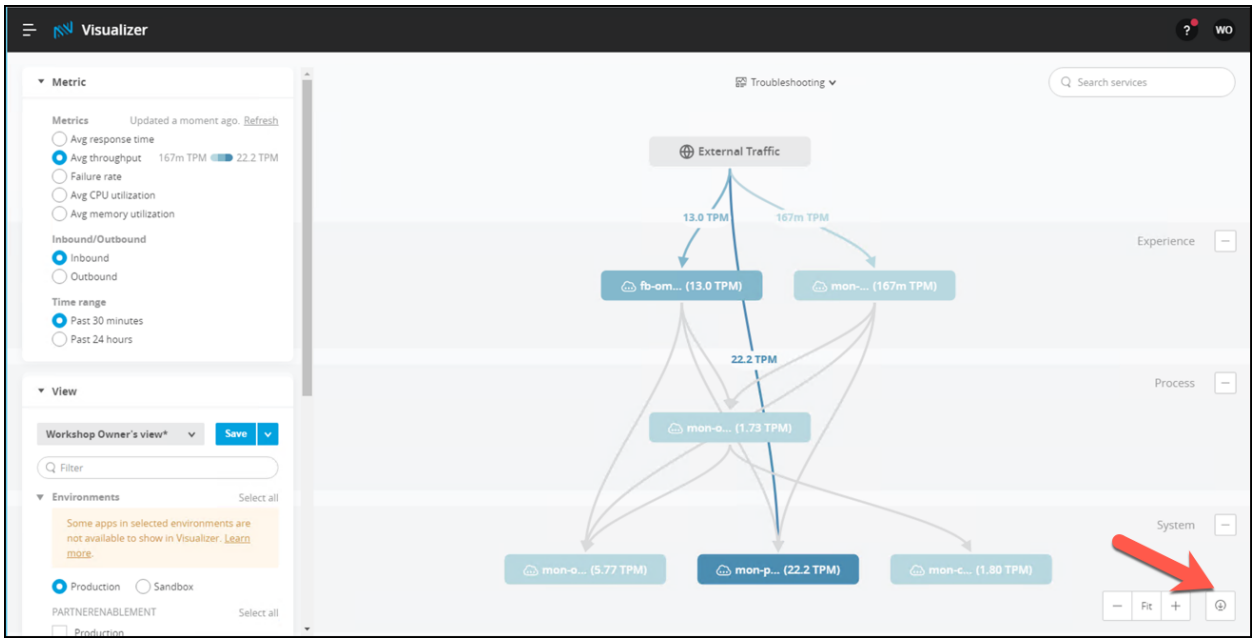
16. Leave the rest of the values as default. Press the **Run Omni Channel** button.



17. Wait 1 minute and go back to the Visualizer graph. Check how the different metrics changes in real time.



18. Finally you can export the graph as an image.



Summary

In this lab, you completed the following steps:

- Introduced to Anypoint Visualizer
- Deployed a new API and verified Visualizer topology updates
- Invoked the new API and verified Visualizer performance metric updates

Go Further:

- See the Visualizer documentation for more information.

Congratulations! You have completed Lab 1.

Please proceed to Lab 2

Take me to the TOP

Lab 2: Anypoint Monitoring

Overview

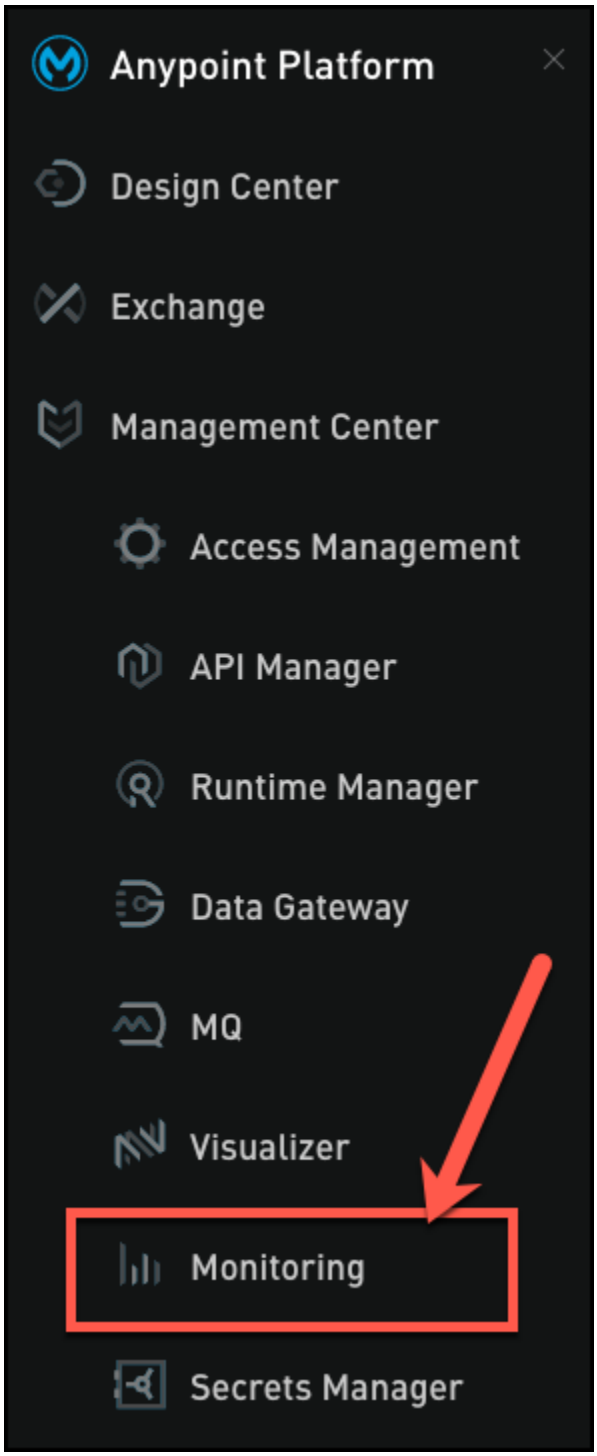
Understanding the state of your infrastructure and systems is important for ensuring the reliability and stability of your services. As part of Anypoint Platform, Anypoint Monitoring provides visibility into integrations across your app network. The monitoring tools provide feedback from Mule flows and components in your app network.

In this lab we are going to review Builtin Dashboards and Custom Dashboards. This functionality can be found in the base subscription. Advanced monitoring capabilities available through the Titanium licensing package are not covered in this lab.

Step 1: Browse Built-in Dashboards

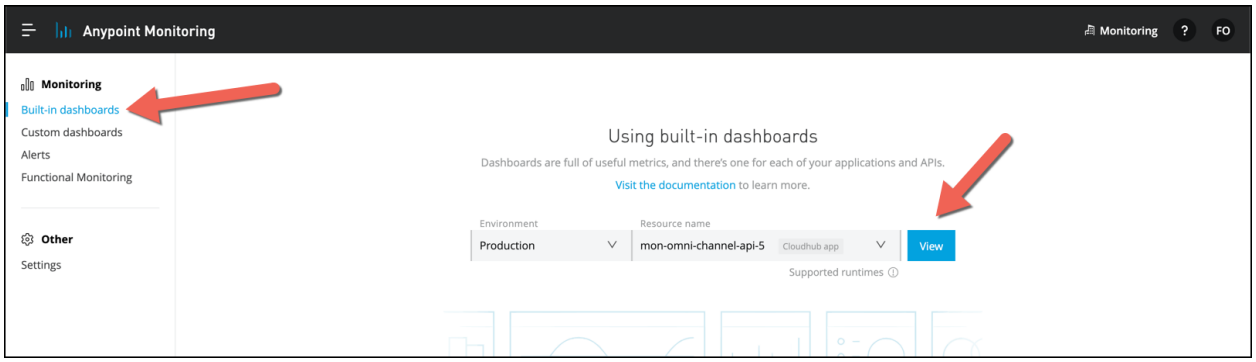
Dashboards in Anypoint Monitoring provide visibility into Mule apps and APIs that are deployed to your environments. In this Step we are going to review Built-in Dashboards.

1. Go to Monitoring

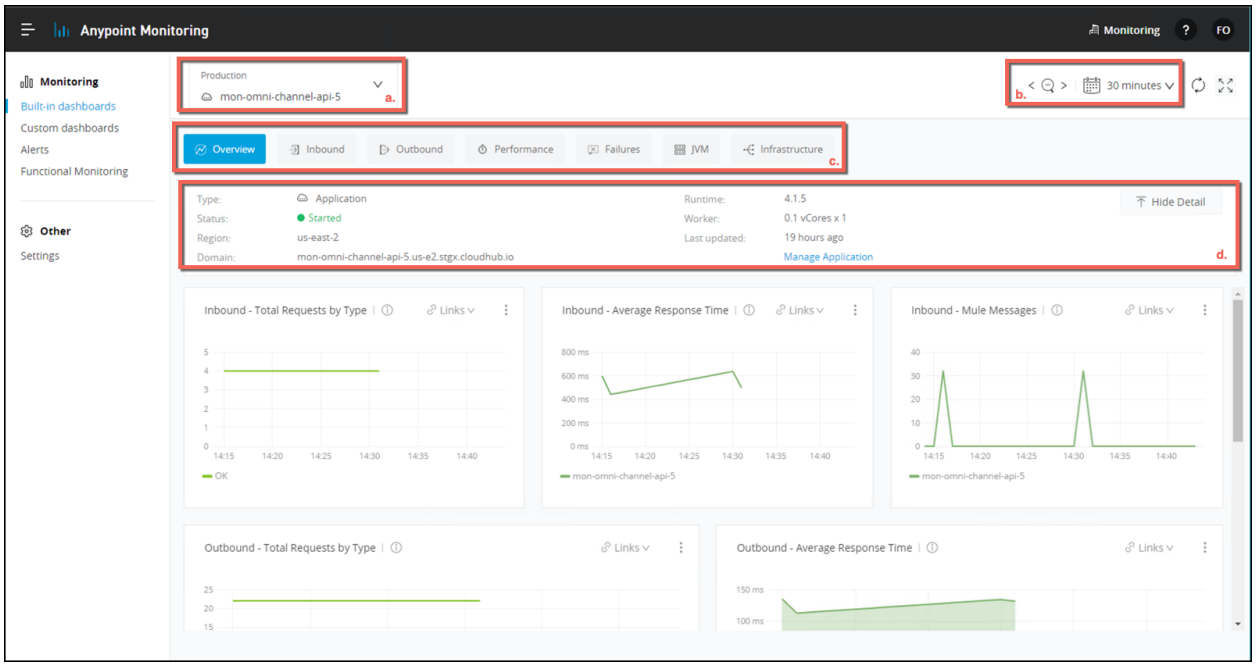


2. Select **Built-in dashboards** on the left side.
3. Choose:
   a. Environment: `Production`
   b. Resource name: `mon-omni-channel-api-<id>`

4. Press **View**



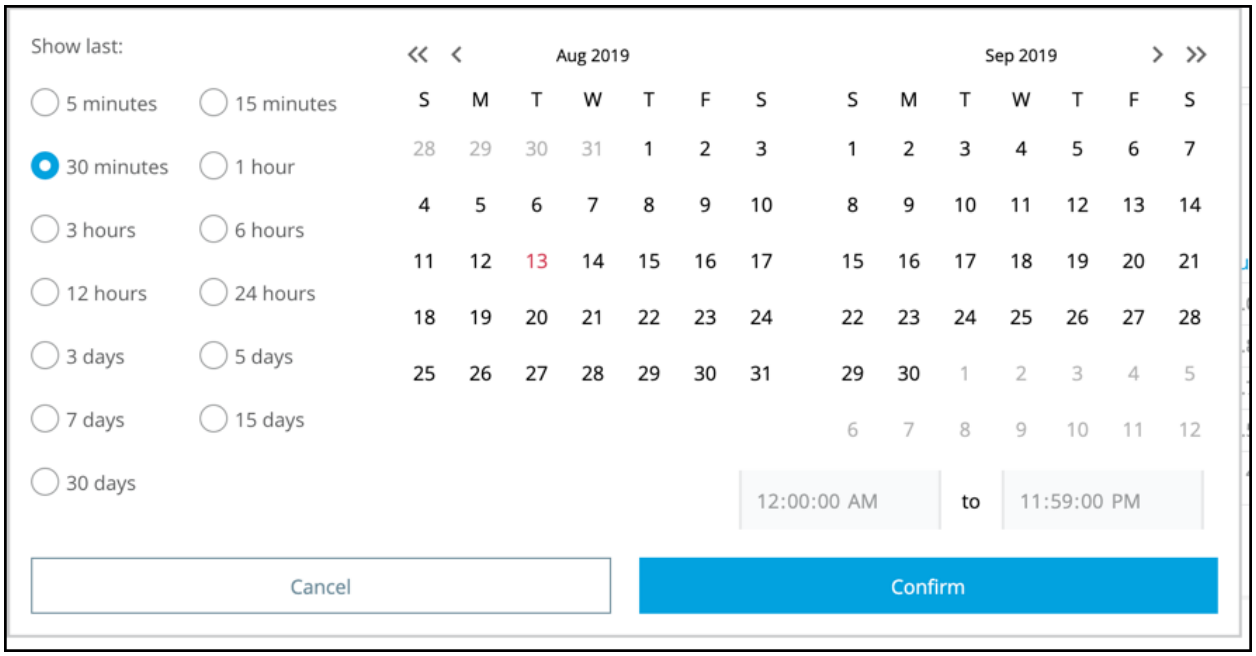
5. You will see an Overview panel with numerous out-of-the-box graphs generated from standard metrics collected by Anypoint Monitoring.



We will describe each section

a. You can see the Environment and the API being monitored.

b. You can filter the report by hours, days, months, period of times.



c. Here you can see different dashboards grouped by

- **Overview**
- **Inbound**
- **Outbound**
- **Performance**
- **Failures**
- **JVM**
- **Infrastructure**

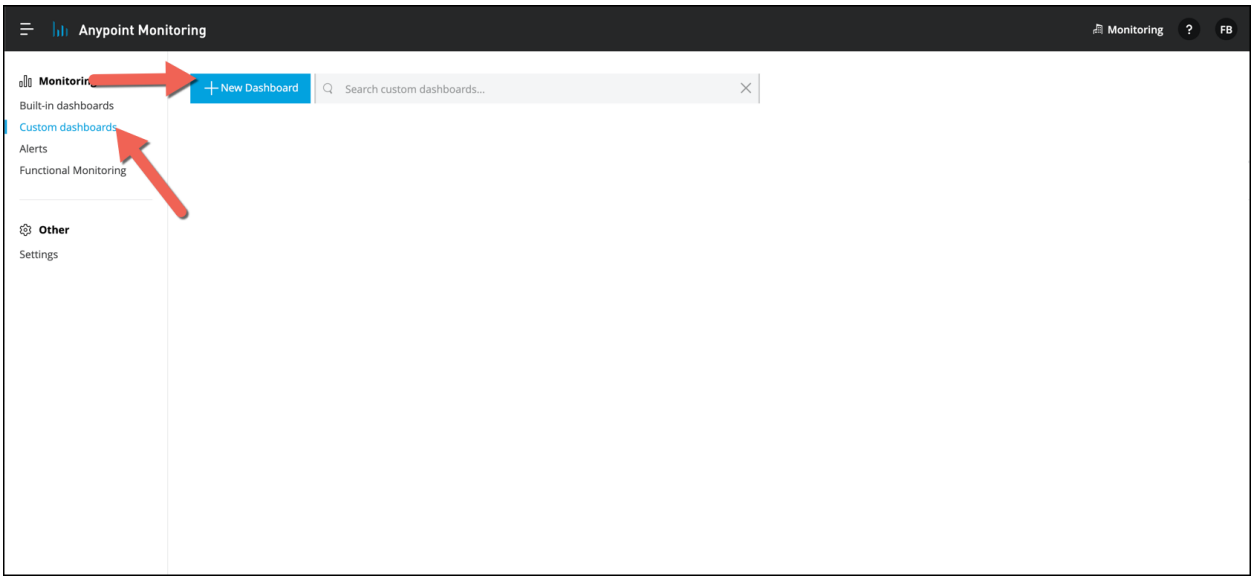
6. Navigate over those items to see the different dashboards.

Step 2: Create Custom Dashboard

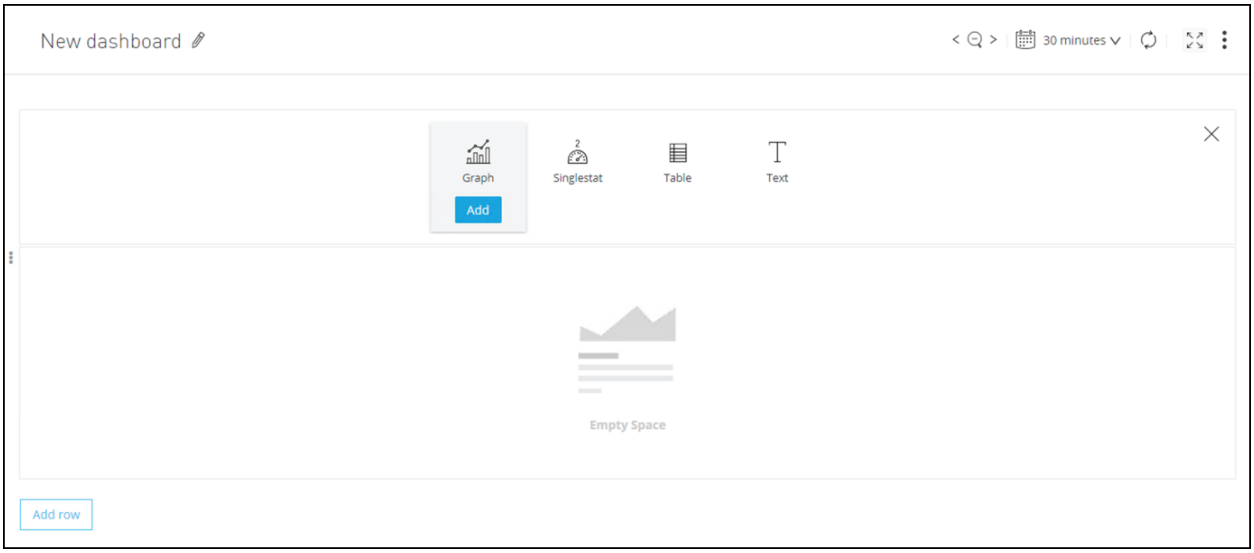
We've already seen the different graphics that come out of the box. You can see them filtered by application. In this section we will create and customize dashboards with different APIs and Metrics.

You will see that you can retrieve metrics from both the Runtime Manager and the API Manager. In this lab we are going to concentrate on Runtime Manager metrics.

1. Go to the Monitoring section and then select Custom dashboards.
2. Press the **New Dashboard** button.



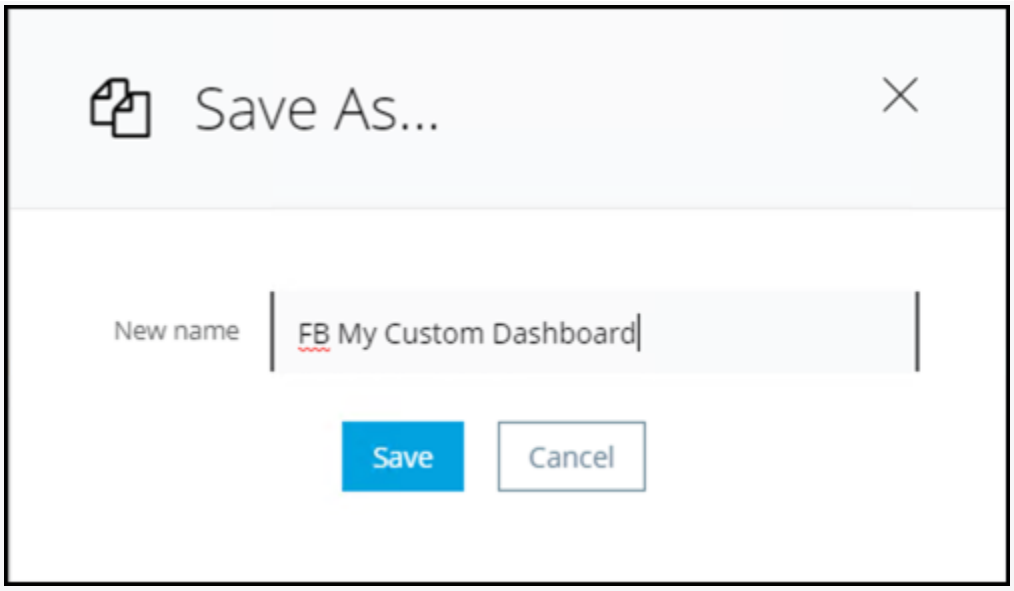
After that you will see a page like this



3. Click on the Pencil icon to edit the dashboard properties.
4. Complete with the following values

Name	Value	Description
Name	<initials> My Custom Dashboard	Name of the Custom Dashboard
Description	Custom Dashboard for ALC Worskop	Brief description
Tags	<initials>-alc-workshop	Add a tag to facilitate finding
Timezone	Local Browser Time	Prefered Time Zone
Graph Tooltip	Shared Tooltip	This option will show you metrics inside the graphics

5. Press the Save **Changes** button.
6. In the **Save As** Pannel choose a representative name like `<initials> My Custom Dashboard`

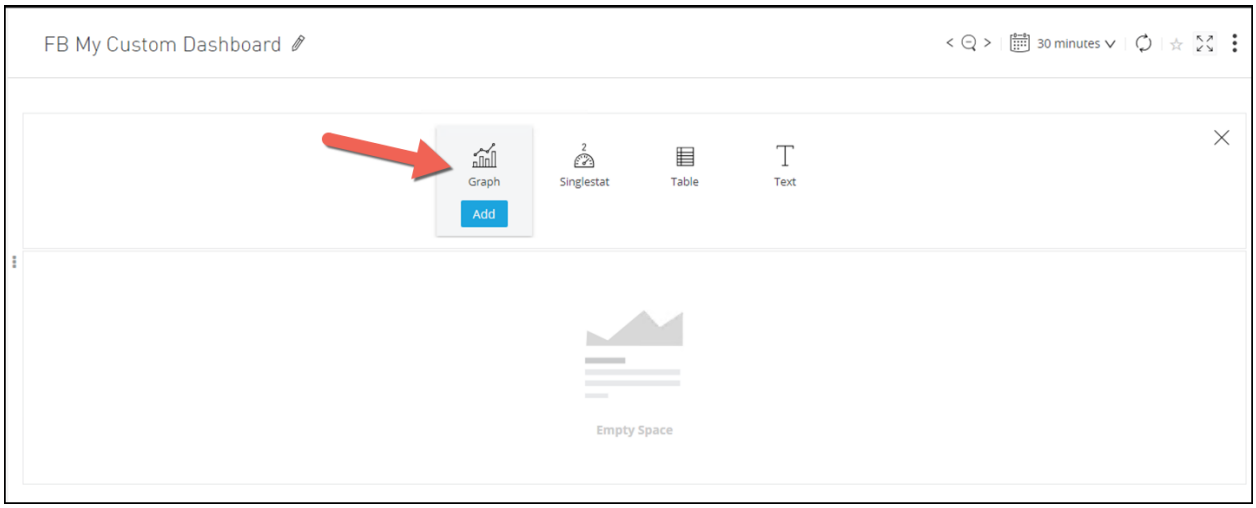


Great, now you can continue to add Graphics and Metrics.

Step 3: Create a Request by Endpoint Graphic

In this step we are going to create a Bar Chart with the number of requests grouped by Endpoint.

1. Press **Add row** button and then the **Add Graph** button.



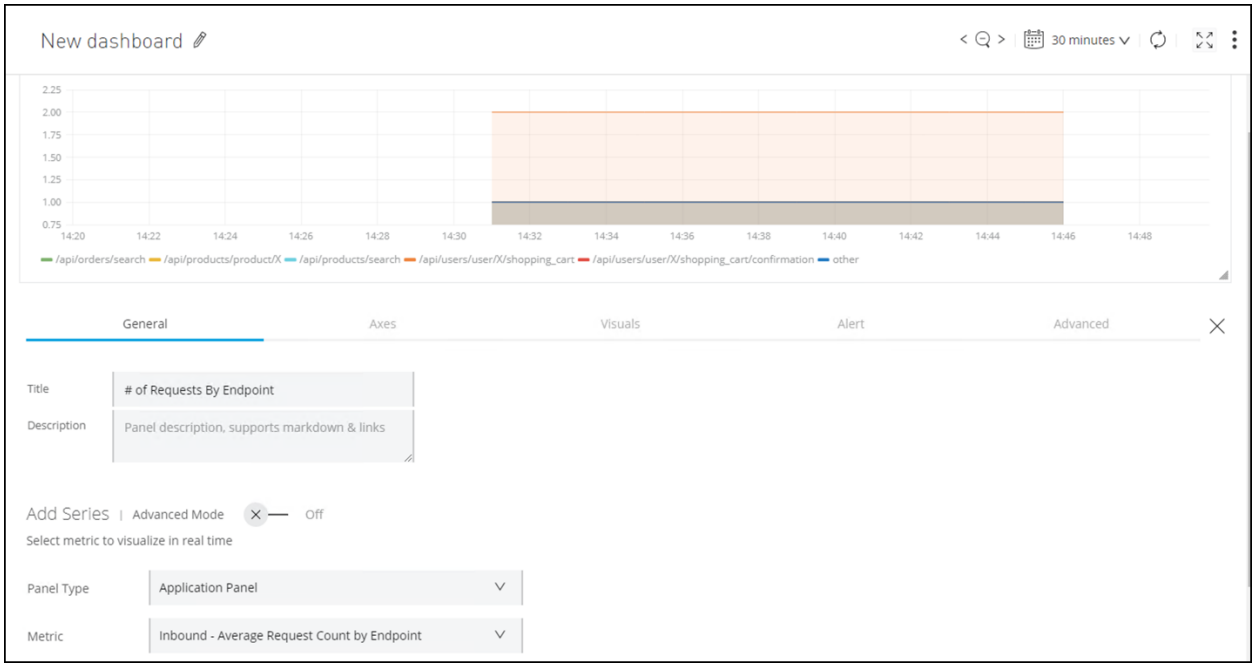
2. Press the three dots on the right of the graphic and select **Configure**.

You will see a configuration panel.

3. In the **General** tab complete with the following values:

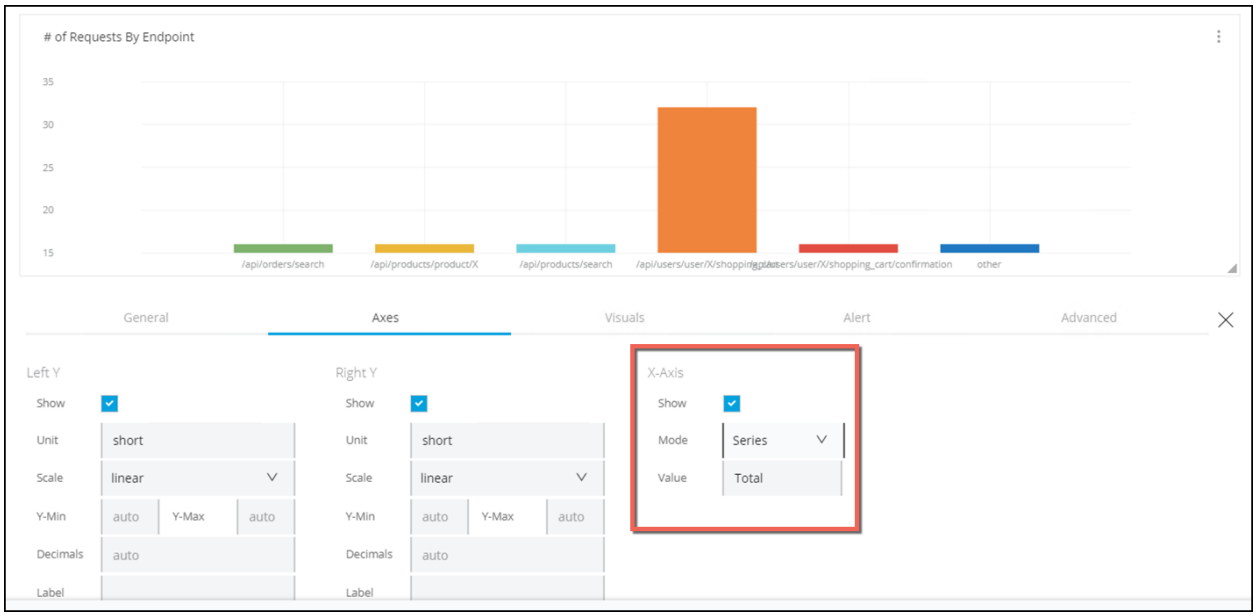
Name	Value	Description
Title	# of Requests by Endpoint	Title of the dashboard
Panel Type	Application Panel	You can choose metrics between Runtime Manager and API Manager
Metric	Inbound - Average Request Count by Endpoint	You can choose between different metrics here
Environment	Production	Environment where the Mule Application is deployed
Resource	mon-omni-channel-api-<id>	Mule Application that will provide the metrics

4. After you enter those values, you will see the graphic display some metrics.



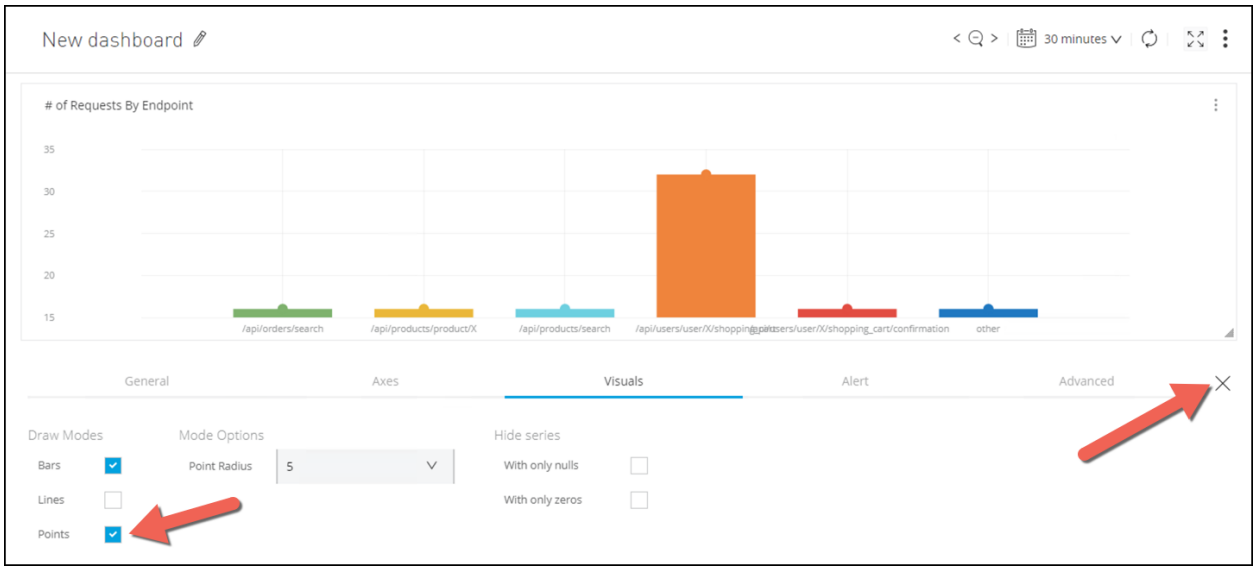
5. Go to the **Axes** tab
6. In **the X-Axis** Change the **Mode** to Series.

   See how the graph changes.



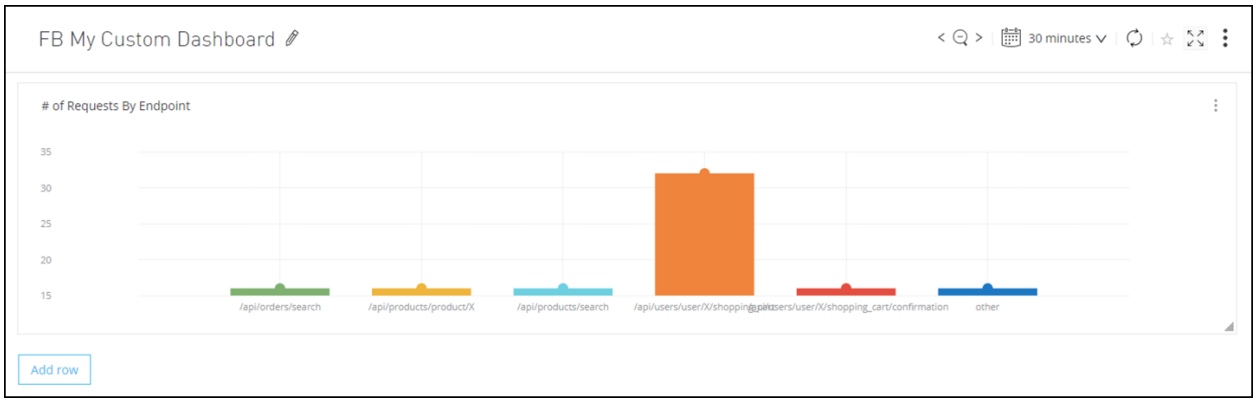
7. Go to the **Visuals** tab.

8. Select the Points Checkbox



9. Finally press the **X** that is on the right side of the panel.
10. Press the Save **Changes** button.
11. In the **Save As** Panel choose a representative name like `<initials> My Custom Dashboard`

    At the end you should see something like this:



Step 4: Create Response Time Dashboard

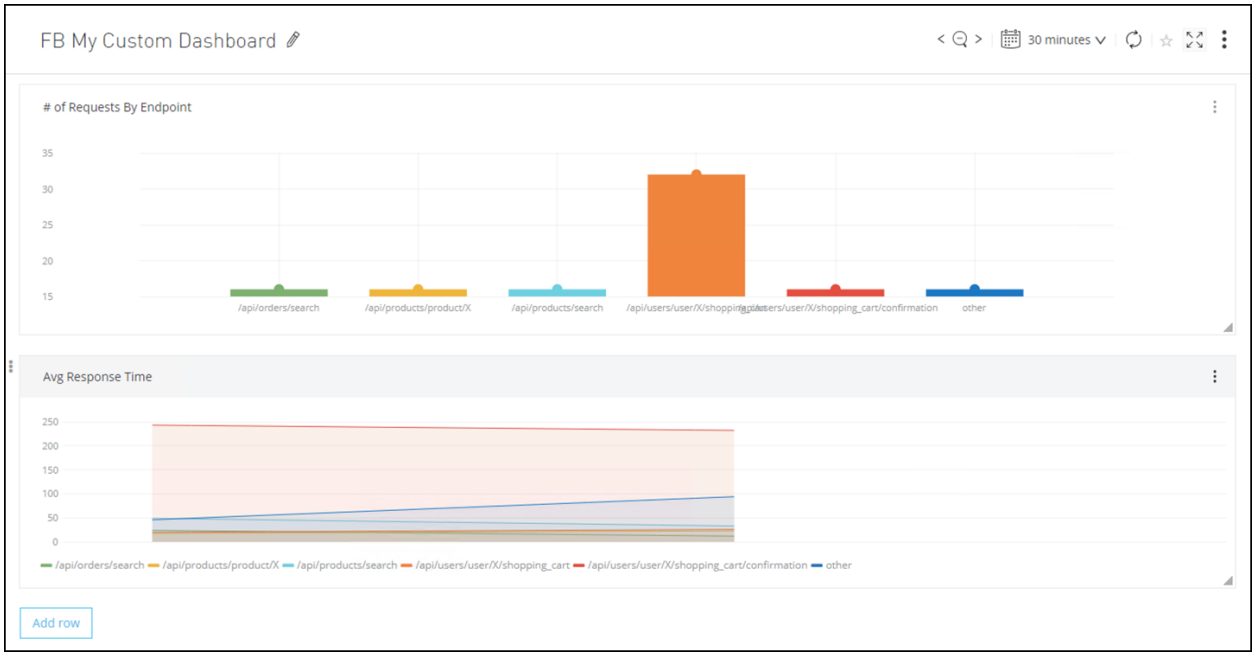
Now we are going to add a new panel with information on the Response Time

1. Press the **Add Row** button.
2. Select the Graph icon.
3. Press the three dots on the right of the graphic and select **Configure**
4. Complete with the following values:

Name	Value
Title	Avg Response Time
Panel Type	Application Panel
Metric	Inbound - Average Response Time by Endpoint
Environment	Production
Resource	mon-omni-channel-api-<id>

5. Press the X to close the configuration panel.
6. Press the Save **Changes** button.

   You should see something like this:

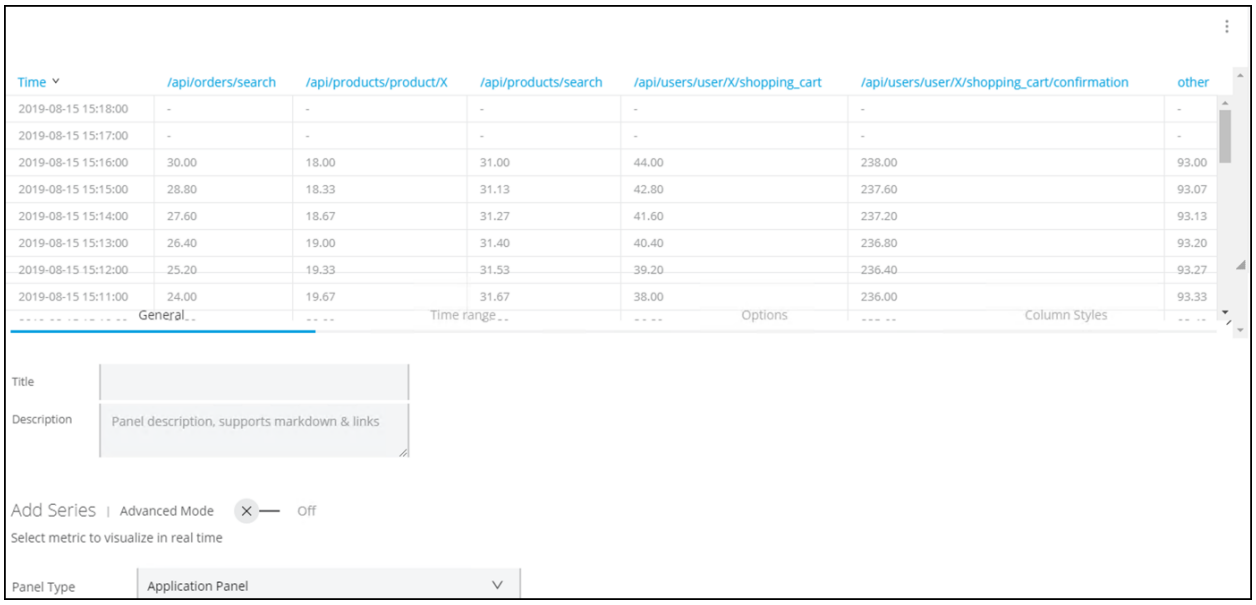


We want to see the response time numbers. For that we are going to add a table.

7. Press the **Add row** button.
8. Choose Table
9. Press the three dots on the right of the graphic and select **Configure**
10. Complete with the following values:

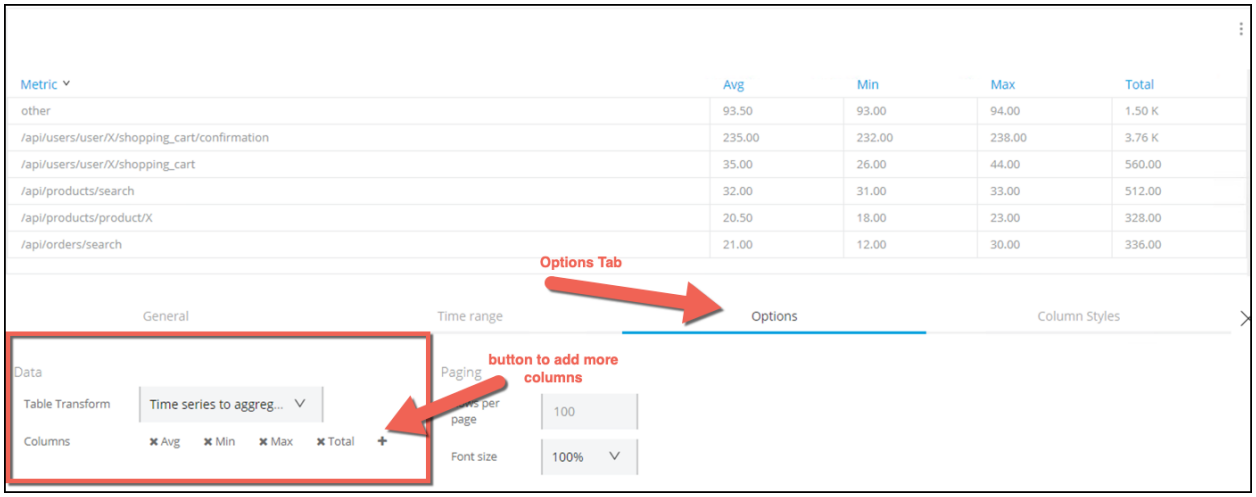
Name	Value
Title	Leave it Blank
Panel Type	Application Panel
Metric	Inbound - Average Response Time by Endpoint
Environment	Production
Resource	mon-omni-channel-api-<id>

11. Look at the table:



Time	/api/orders/search	/api/products/product/X	/api/products/search	/api/users/user/X/shopping_cart	/api/users/user/X/shopping_cart/confirmation	other
2019-08-15 15:18:00	-	-	-	-	-	-
2019-08-15 15:17:00	-	-	-	-	-	-
2019-08-15 15:16:00	30.00	18.00	31.00	44.00	238.00	93.00
2019-08-15 15:15:00	28.80	18.33	31.13	42.80	237.60	93.07
2019-08-15 15:14:00	27.60	18.67	31.27	41.60	237.20	93.13
2019-08-15 15:13:00	26.40	19.00	31.40	40.40	236.80	93.20
2019-08-15 15:12:00	25.20	19.33	31.53	39.20	236.40	93.27
2019-08-15 15:11:00	24.00	19.67	31.67	38.00	236.00	93.33

12. Go to the **Options Panel**
13. In the **Data** area.
    a. Choose **Time series to aggregate** from the combo box.
    b. In Columns, you will see Avg. Click the + button to add `Min`, `Max` and `Total`.



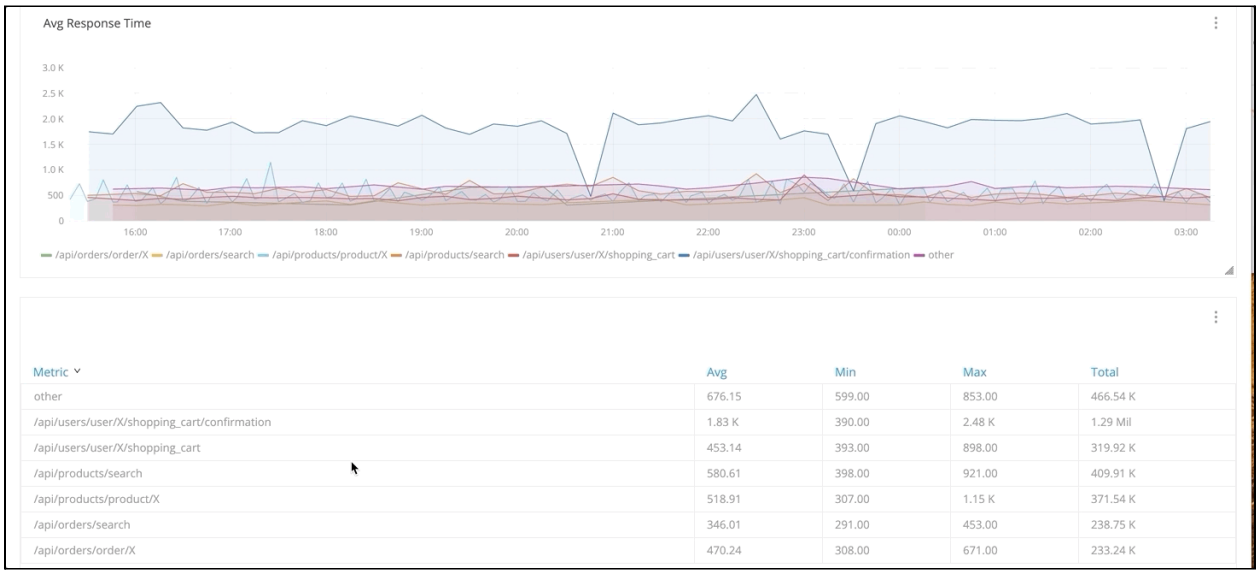
Metric	Avg	Min	Max	Total
other	93.50	93.00	94.00	1.50 K
/api/users/user/X/shopping_cart/confirmation	235.00	232.00	238.00	3.76 K
/api/users/user/X/shopping_cart	35.00	26.00	44.00	560.00
/api/products/search	32.00	31.00	33.00	512.00
/api/products/product/X	20.50	18.00	23.00	328.00
/api/orders/search	21.00	12.00	30.00	336.00

14. Press the **X** that is on the right side of the panel.
15. Press the Save **Changes** button.

    We are going to move the Table panel and place it in-line with the Avg Response Time panel.

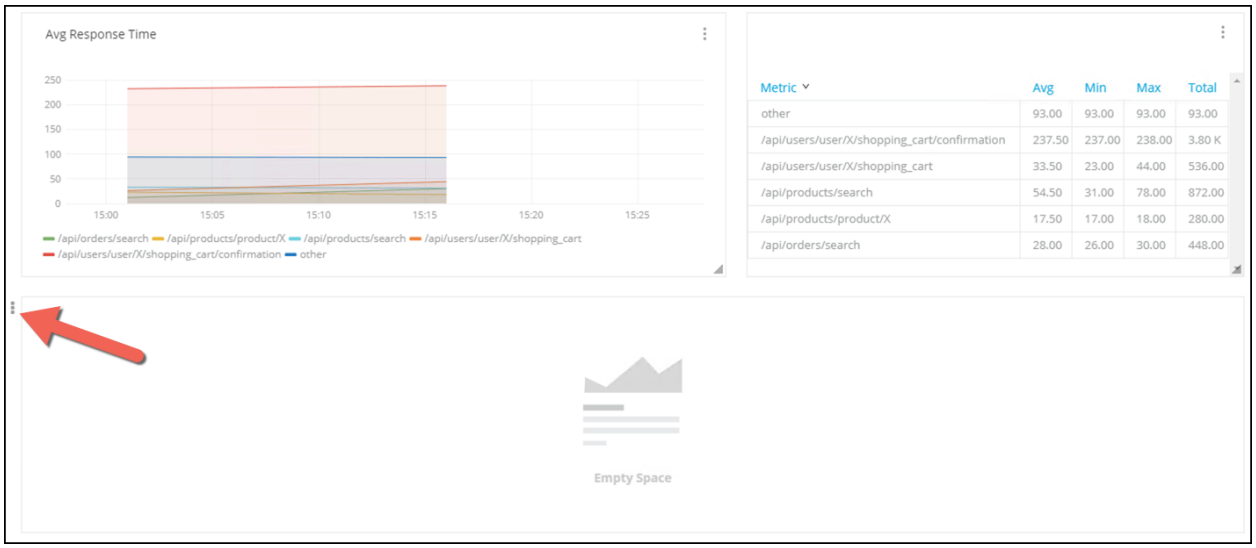
    The UI gives the ability to drag and drop panels and graphics.

16. Click to the right of the three dots associated with the row on the left side of the panel.



Metric	Avg	Min	Max	Total
other	676.15	599.00	853.00	466.54 K
/api/users/user/X/shopping_cart/confirmation	1.83 K	390.00	2.48 K	1.29 Mil
/api/users/user/X/shopping_cart	453.14	393.00	898.00	319.92 K
/api/products/search	580.61	398.00	921.00	409.91 K
/api/products/product/X	518.91	307.00	1.15 K	371.54 K
/api/orders/search	346.01	291.00	453.00	238.75 K
/api/orders/order/X	470.24	308.00	671.00	233.24 K

17. Drag and Drop the Table panel into the Graph panel.
18. Since we moved the Table, you will see that there is an Empty Space section. Let's remove that.
19. Move the pointer to the Empty Space panel. You will see three dots on the left side of the panel.

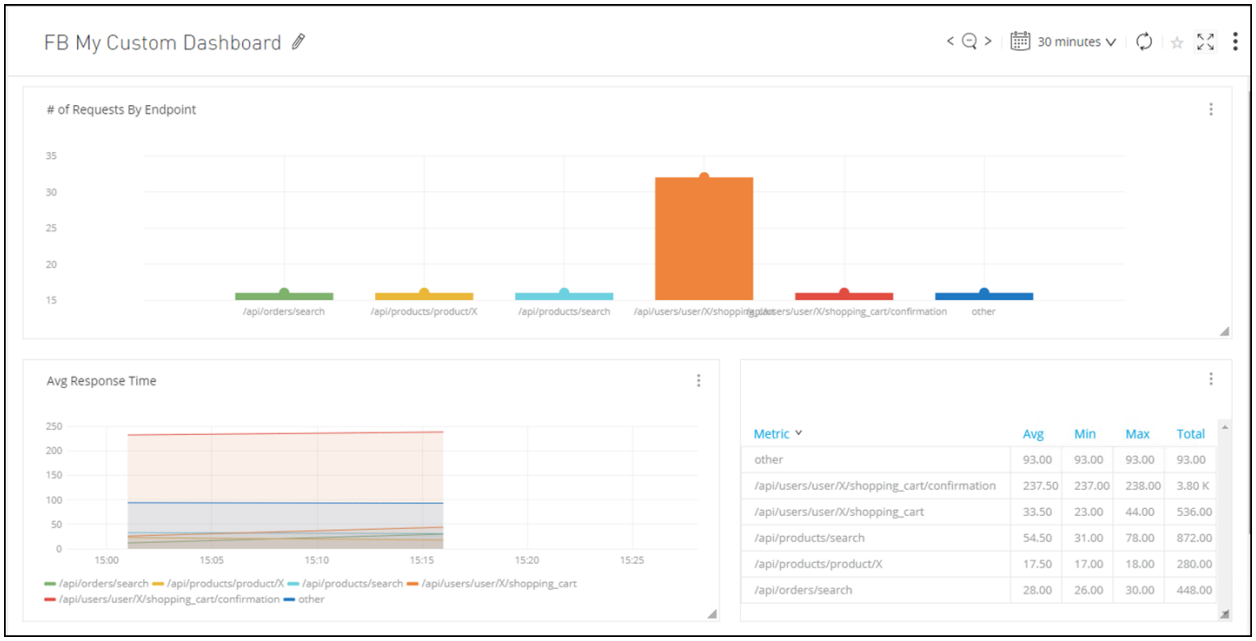


Metric	Avg	Min	Max	Total
other	93.00	93.00	93.00	93.00
/api/users/user/X/shopping_cart/confirmation	237.50	237.00	238.00	3.80 K
/api/users/user/X/shopping_cart	33.50	23.00	44.00	536.00
/api/products/search	54.50	31.00	78.00	872.00
/api/products/product/X	17.50	17.00	18.00	280.00
/api/orders/search	28.00	26.00	30.00	448.00

20. Point with the pointer to those dots. A new panel will appear.

21. Select **Delete**

At the end you should see something like this:



Step 5: Create Single Stats Graphics

In this section we are going to add single stat graphic for the System APIs. We would like to know if the avg response times of the system APIs are less than 10 seconds.

1. Press the **Add row** button.
2. Select **Singlestat** icon
3. Press the three dots on the right of the graphic and select **Configure**
4. Complete with the following values:

Name	Value
Title	Product API Avg Response Time
Panel Type	Application Pane
Metric	Inbound - Average Response Time by Flow
Environment	Production
Resource	mon-product-api-<id>

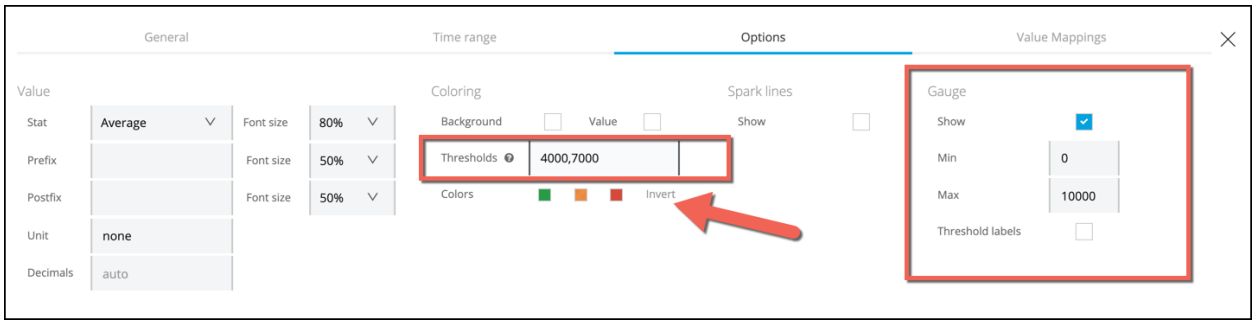
5. Go to the **Options** panel
6. In the **Coloring** column complete with:

a. Threshold: `4000,7000`. This are the threshold values where the colors in the graphic will change.

b. Colors: Press the invert button. The order of the colors should be green, orange, red.

7. In the **Gauge** check the **Show** box.

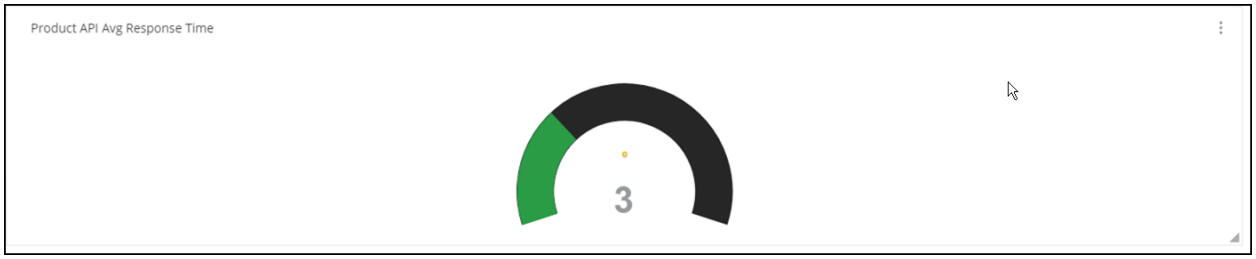
a. Change the Max value to `10000`. Max Value means that a response time of 10000 ms or more will be unacceptable.



This graph will be green if the response time less than 4s, orange if it is between 4s and 7s and red if its over 7s.

8. Click the 'X' button to close the panel.

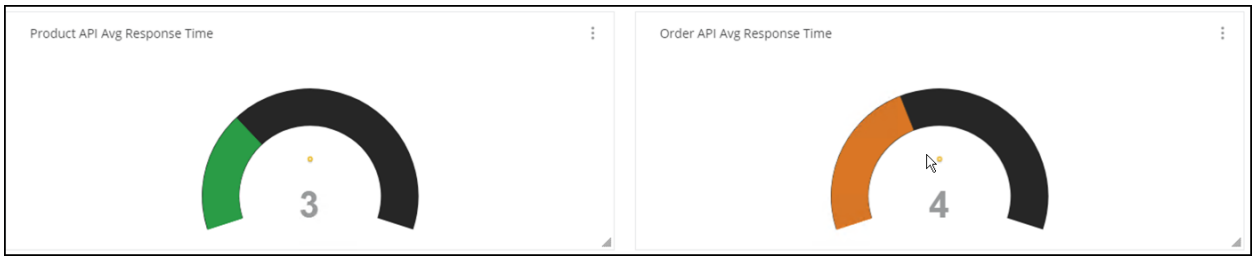
   This is the final result



   We will replicate the same graphic for the Order API and the Customer API.

9. Click on the three dots section for the Singlestat panel and choose **Duplicate**.
10. Go to the second panel and select **Configure**.

11. Change the Title to `Order API Avg Response Time`
12. Change the Resource field and select `mon-order-api-<id>`
13. Click the 'X' button to close the panel.
14. Press the **Save Changes** button.

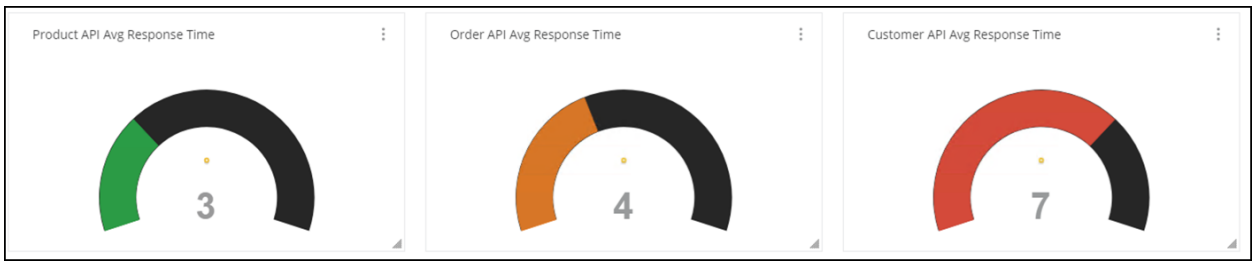


   We are going to repeat the last steps to Create the Customer API Panel.

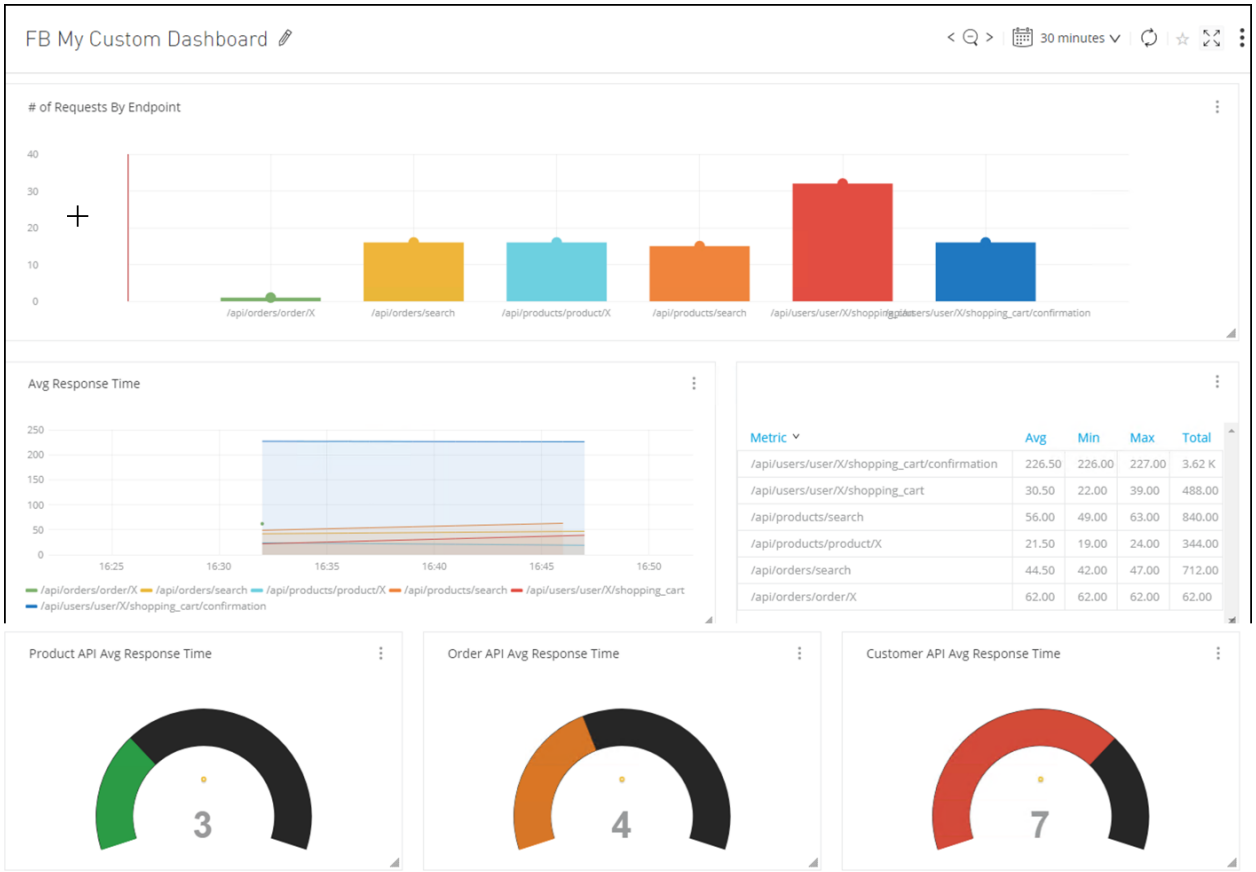
15. Click on the three dots section for the Singlestat panel and choose **Duplicate**.

16. Go to the third panel and select **Configure**.

17. Change the Title to `Customer API Avg Response Time`
18. Change the Resource field and select `mon-customer-api-<id>`
19. Click the 'X' button to close the panel.
20. Press the **Save Changes** button.



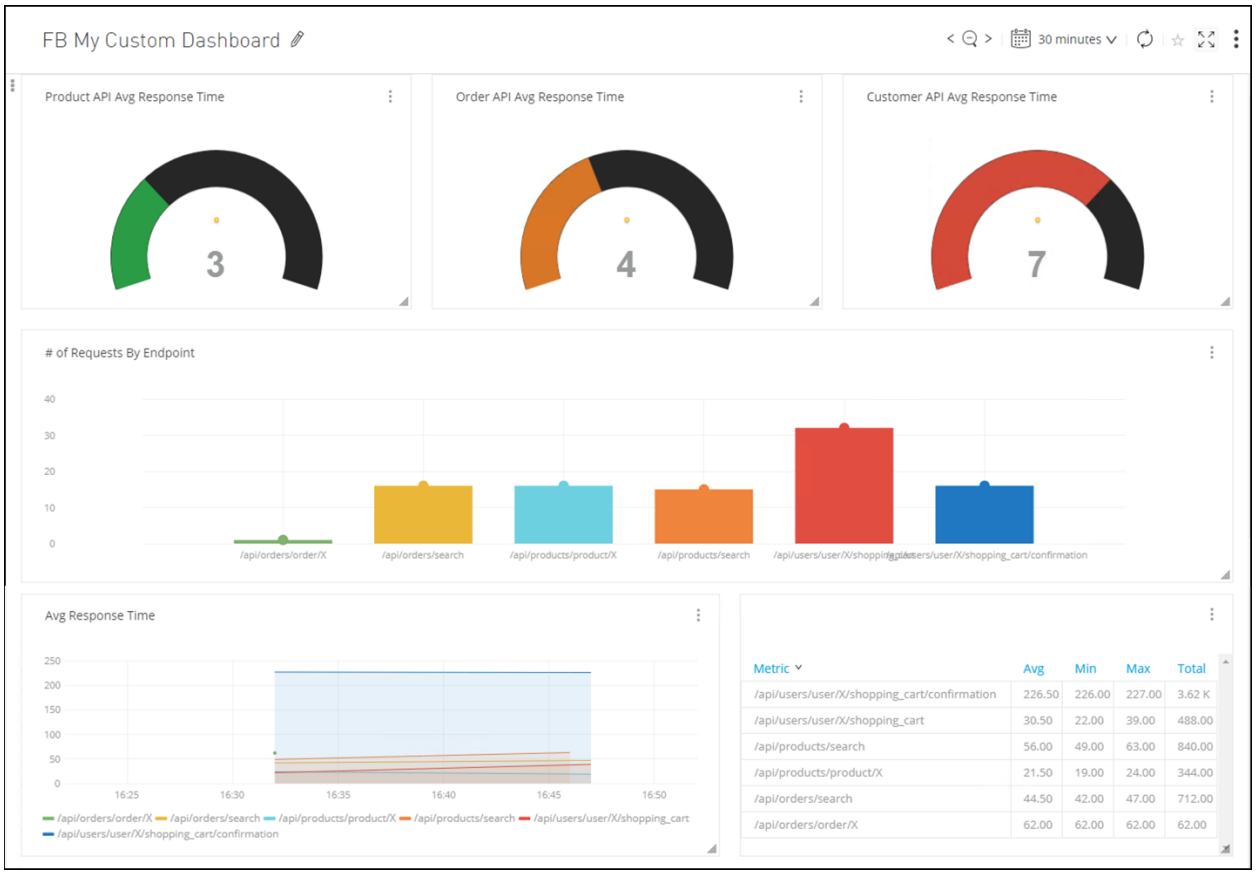
The complete dashboard should look like this:



We would like to have the single stat at the beginning, so let's move the panel to the top.

21. Point to the left three dots of the last panel.

22. Select **Move Up** twice.



Metric	Avg	Min	Max	Total
/api/users/user/X/shopping_cart/confirmation	226.50	226.00	227.00	3.62 K
/api/users/user/X/shopping_cart	30.50	22.00	39.00	488.00
/api/products/search	56.00	49.00	63.00	840.00
/api/products/product/X	21.50	19.00	24.00	344.00
/api/orders/search	44.50	42.00	47.00	712.00
/api/orders/order/X	62.00	62.00	62.00	62.00

Now it looks perfect. You can save the changes.

Summary

In this lab, you completed the following steps:

- Step 1: Browse Built-in Dashboards
- Step 2: Create Custom Dashboard
- Step 3: Create a Request by Endpoint Graphic
- Step 4: Create Response Time Dashboard
- Step 5: Create Single Stats Graphics

Go Further:

- See the Functional Monitor doc for more information.
- See the Built-In Dashboards doc for more information.
- See the Configure Custom Dashboards and Charts doc for more information.
- See the Custom Metrics doc for more information.

Congratulations! You have completed Lab 2 of Module 6

Please proceed to Lab3

Take me to the TOP

Lab 3: Go further - Functional Monitoring

Overview

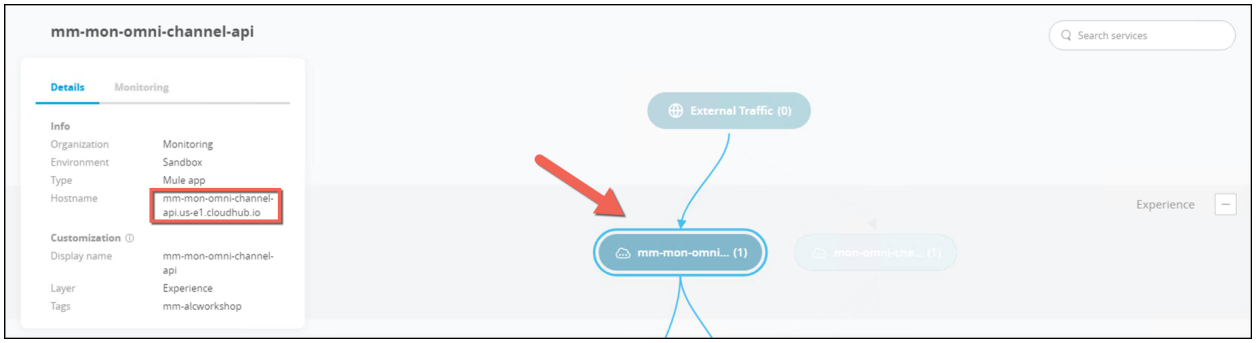
The API Functional Monitoring solution enables developers and operators to perform consistent and constant testing of the functional behavior and performance of their APIs, throughout the entire API lifecycle, in both non-production and production environments.

API Functional Monitoring drives up the quality and reliability of publicly exposed APIs as they are being developed, tested and delivered into production. The monitor is not only specific to APIs that you might develop on Anypoint Platform, it can also be used to perform quality and reliability tests on any publicly exposed API.

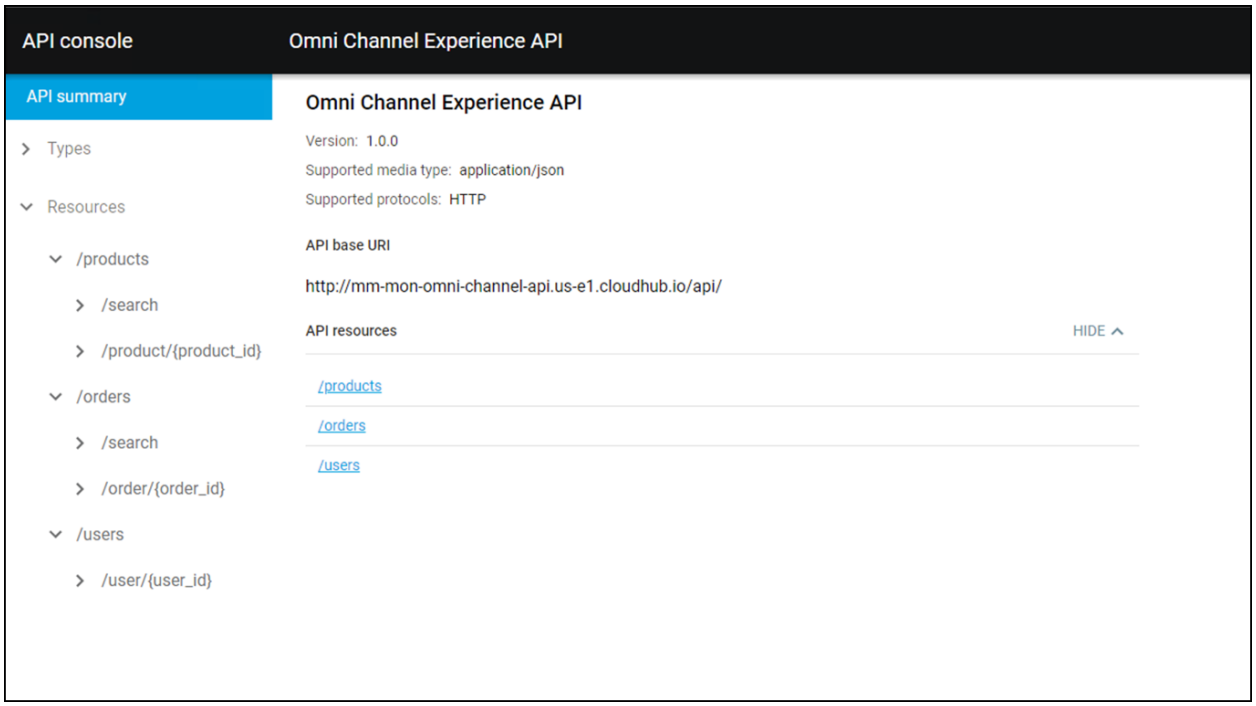
In this Lab, we are going to create an API Functional Monitoring test that will call APIs on a regular interval and perform tests to ensure reliability and quality of a publicly exposed API.

Step 1: Test Endpoint in API Console

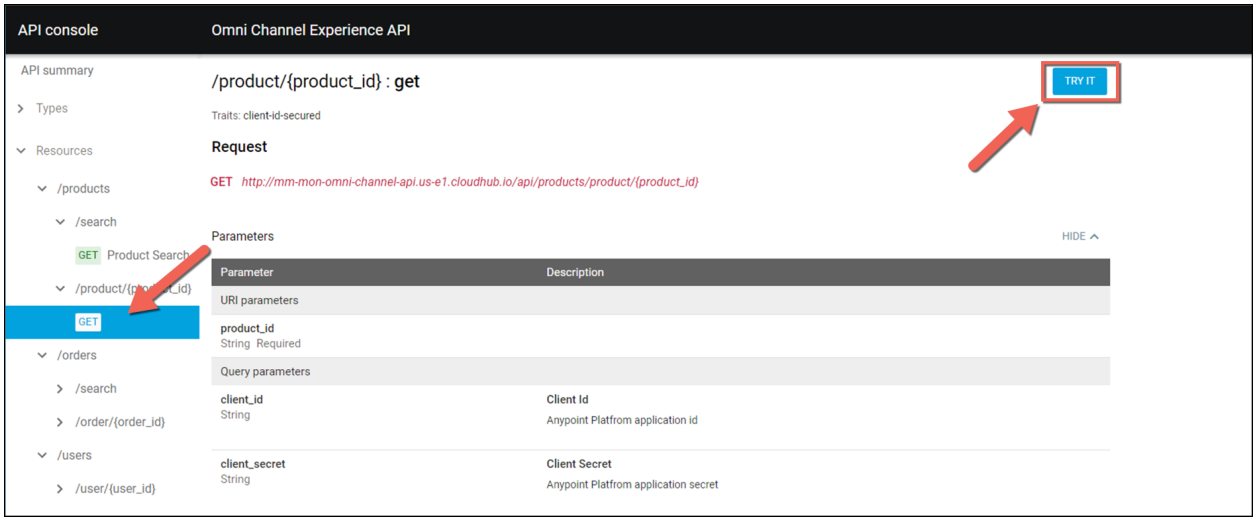
1. In the previous lab we reviewed Visualizer and the functionality available. Please return to Visualizer and select the node that you deployed in the previous lab.



2. Copy and paste the hostname of the application and paste it into a browser adding `/console` to the end of the url. This will open up the console for the API you deployed.

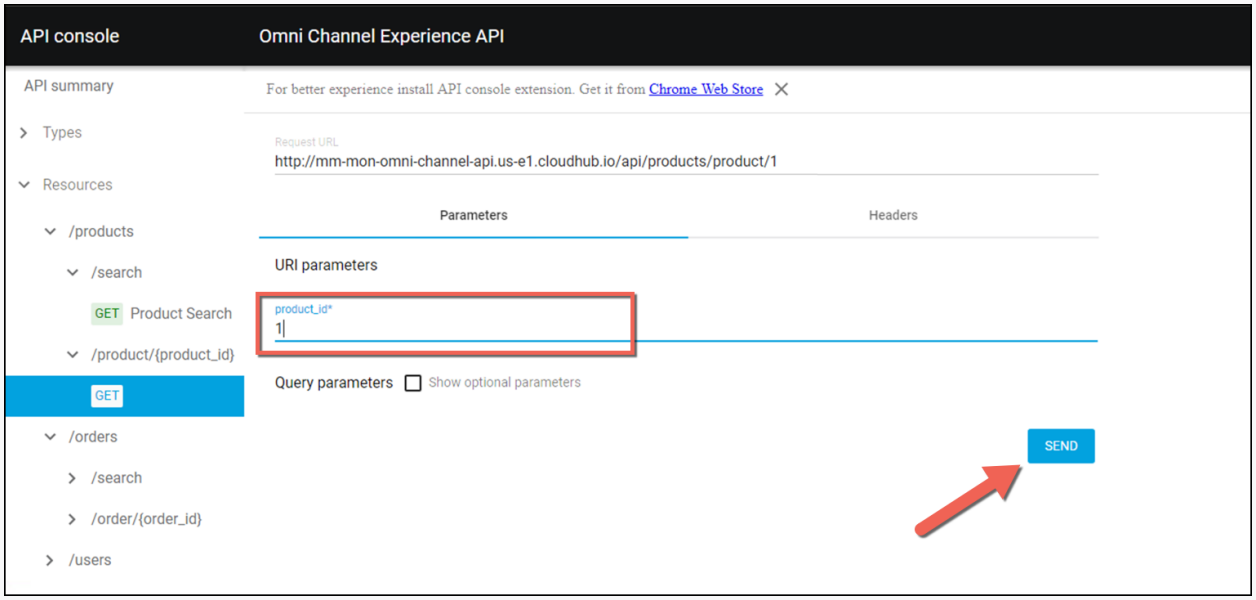


3. We want to select an endpoint to use in our functional monitor. To keep this simple let's select the `GET` Product by id endpoint.

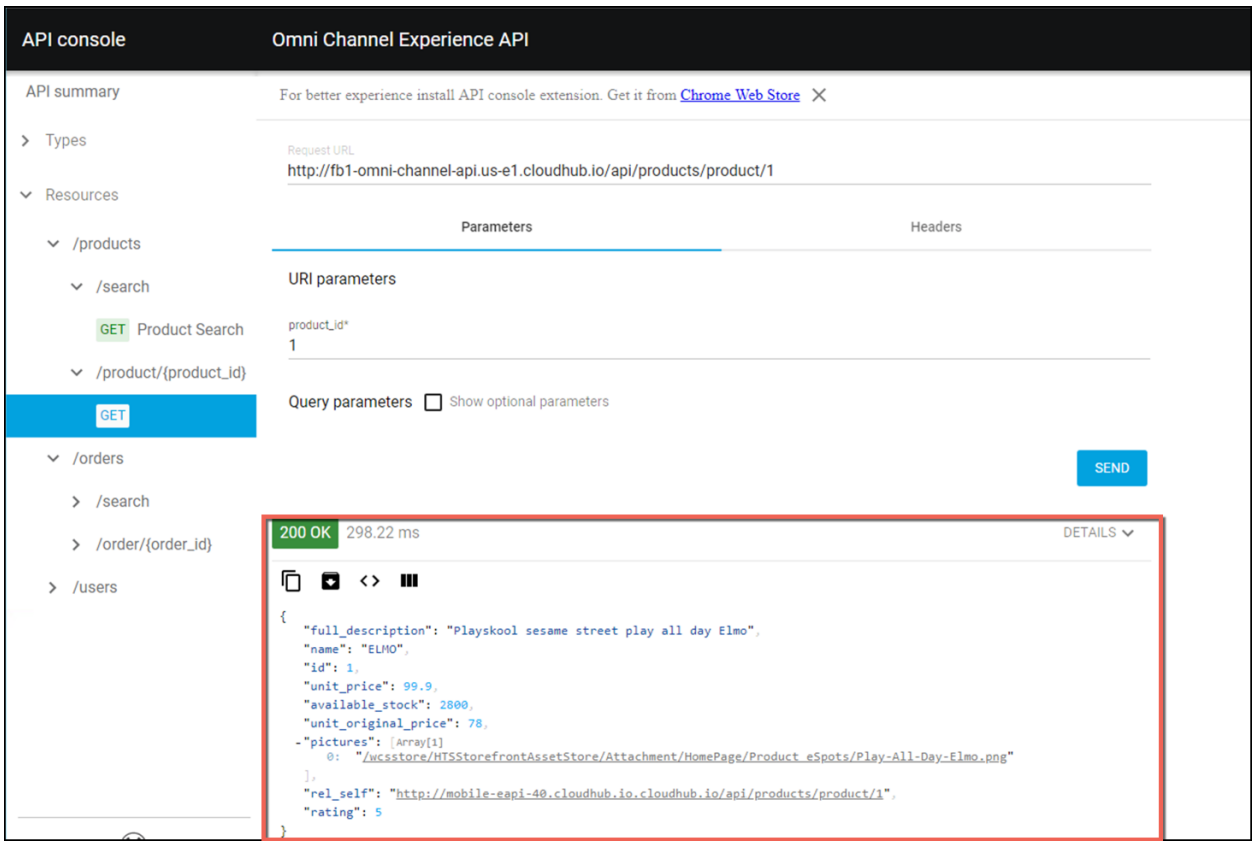


Click `Try It` in the top right corner to test out the API

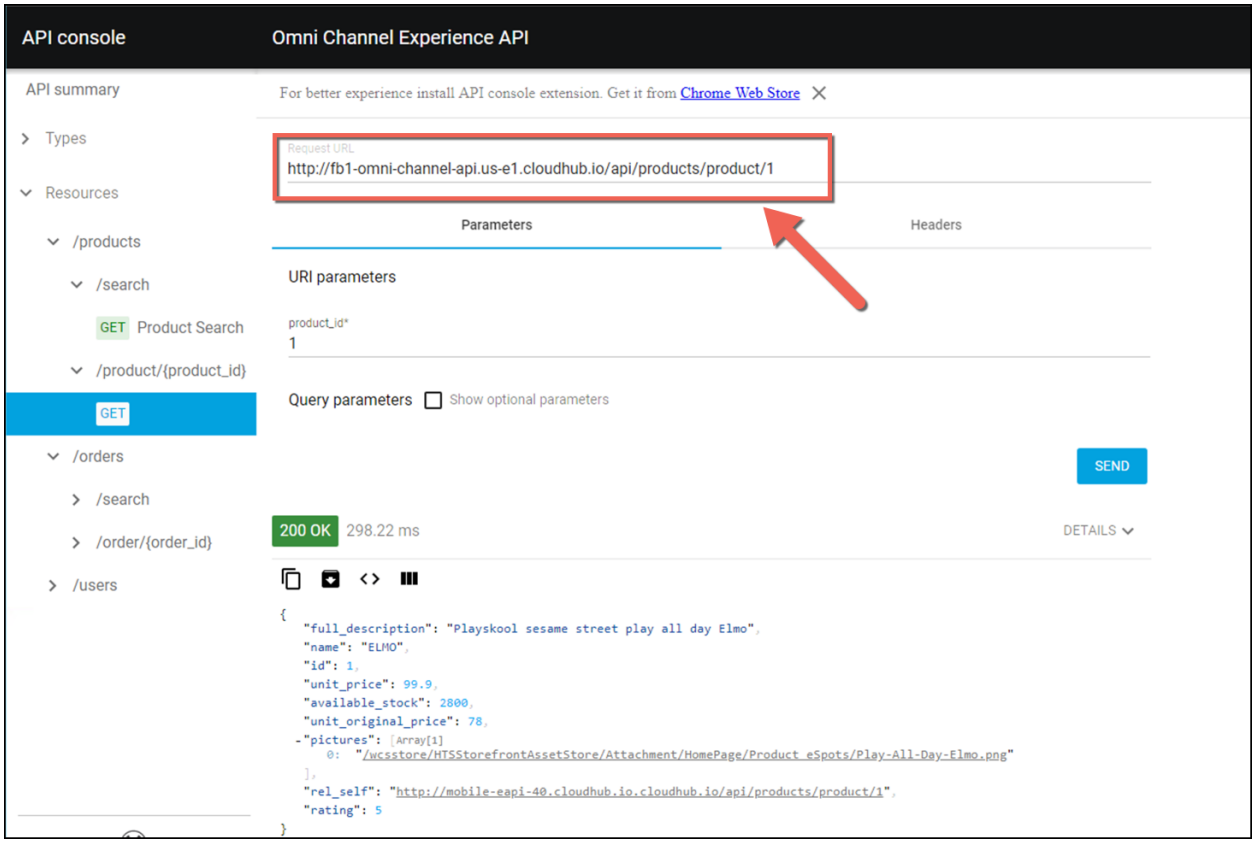
4. We need to give the endpoint an `id` to get a resource. We will use the value `1` for the id. In the **product_id** field on the console enter the value `1` and then click `SEND`



You should see the output for a product, possibly an ELMO toy.

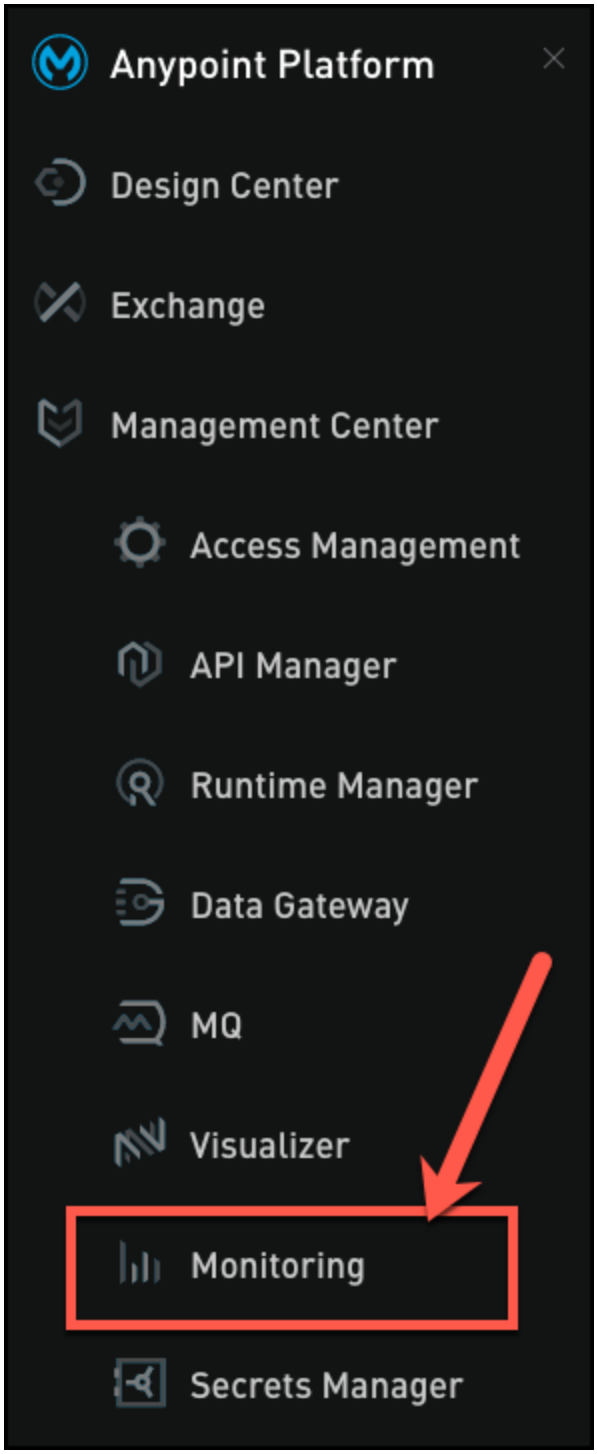


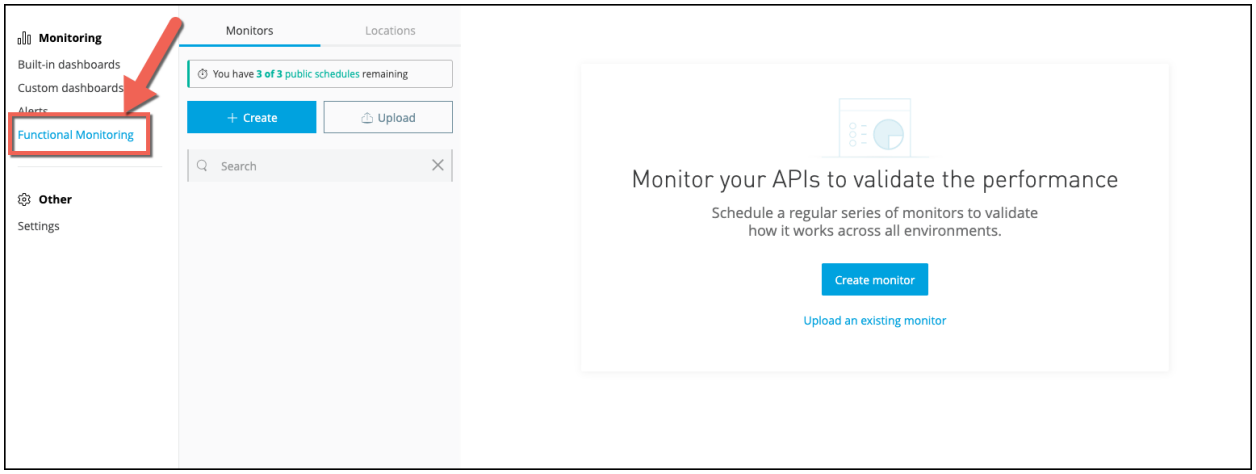
5. Now the endpoint is working you should copy the url from the **request url** field and make a note of this for use in the functional monitor.



Step 2: Create a Functional Monitor

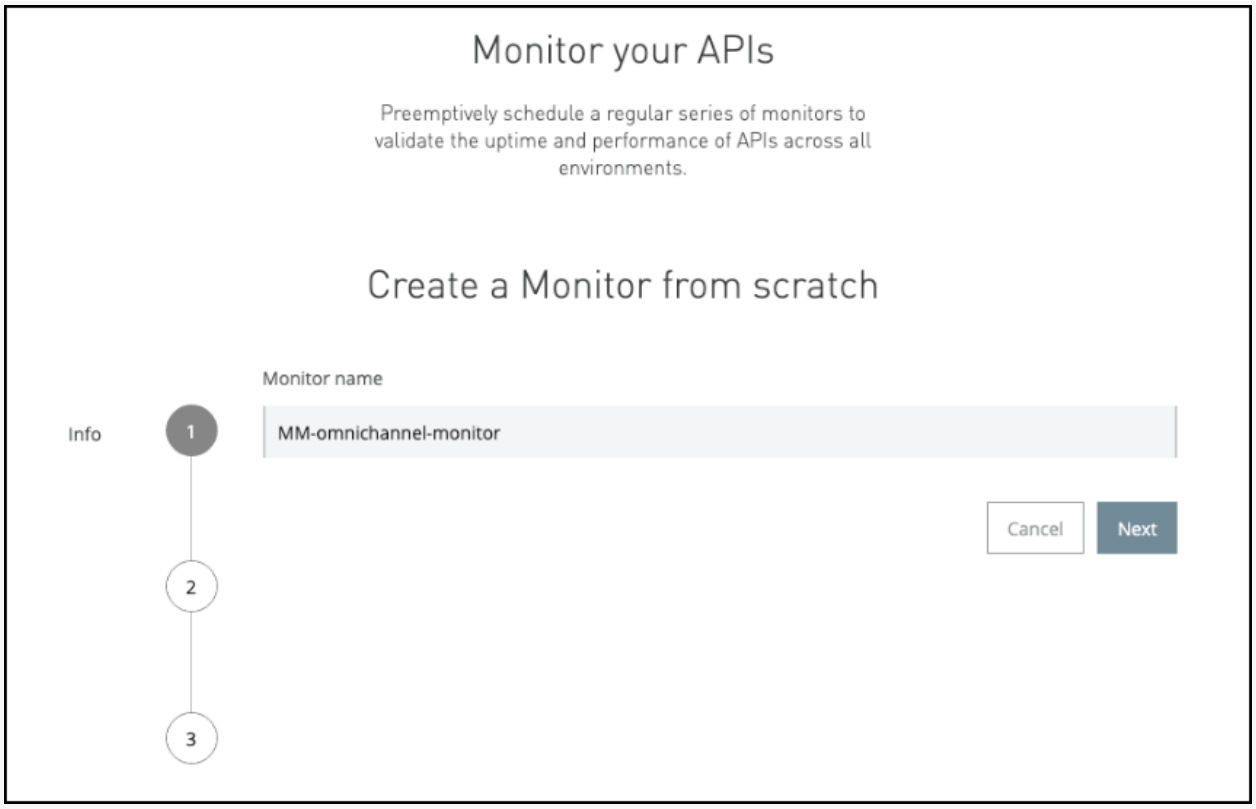
1. Return to Anypoint Platform and from the main menu in the top left, navigate to **Monitoring** and then click **Functional Monitoring** where you will be presented with a page that looks similar to this:


Anypoint Platform
Design Center
Exchange
Management Center
Access Management
API Manager
Runtime Manager
Data Gateway
MQ
Visualizer
Monitoring
Secrets Manager

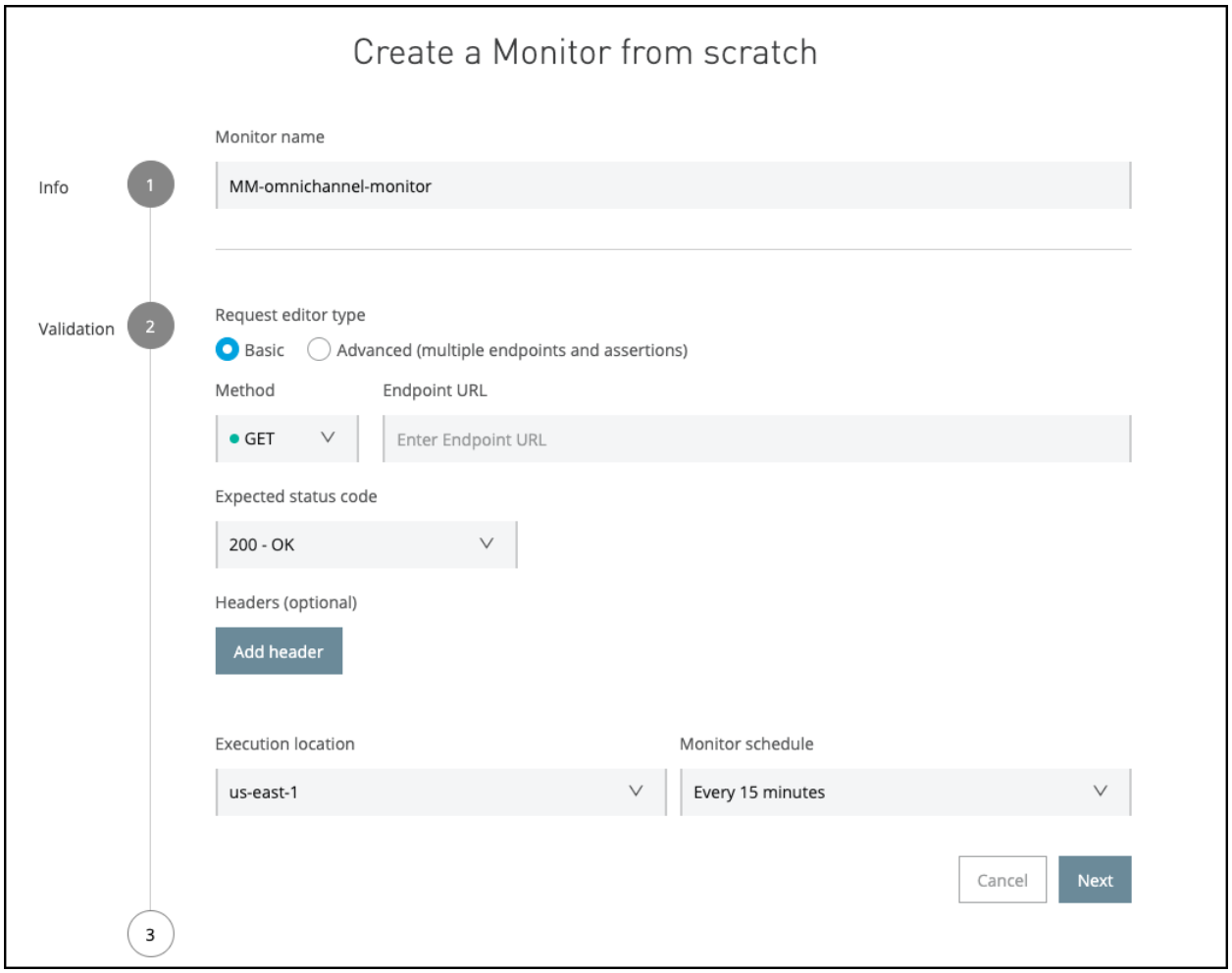


2. We want to create a monitor, click **Create** You then be presented with a set of steps. First step is to give the monitor a name. Use your initials and the following **-omnichannel-monitor** For example, below Max Mule would look like this:

`mm-omnichannel-monitor`



3. Click next and you are onto step 2 of the process:

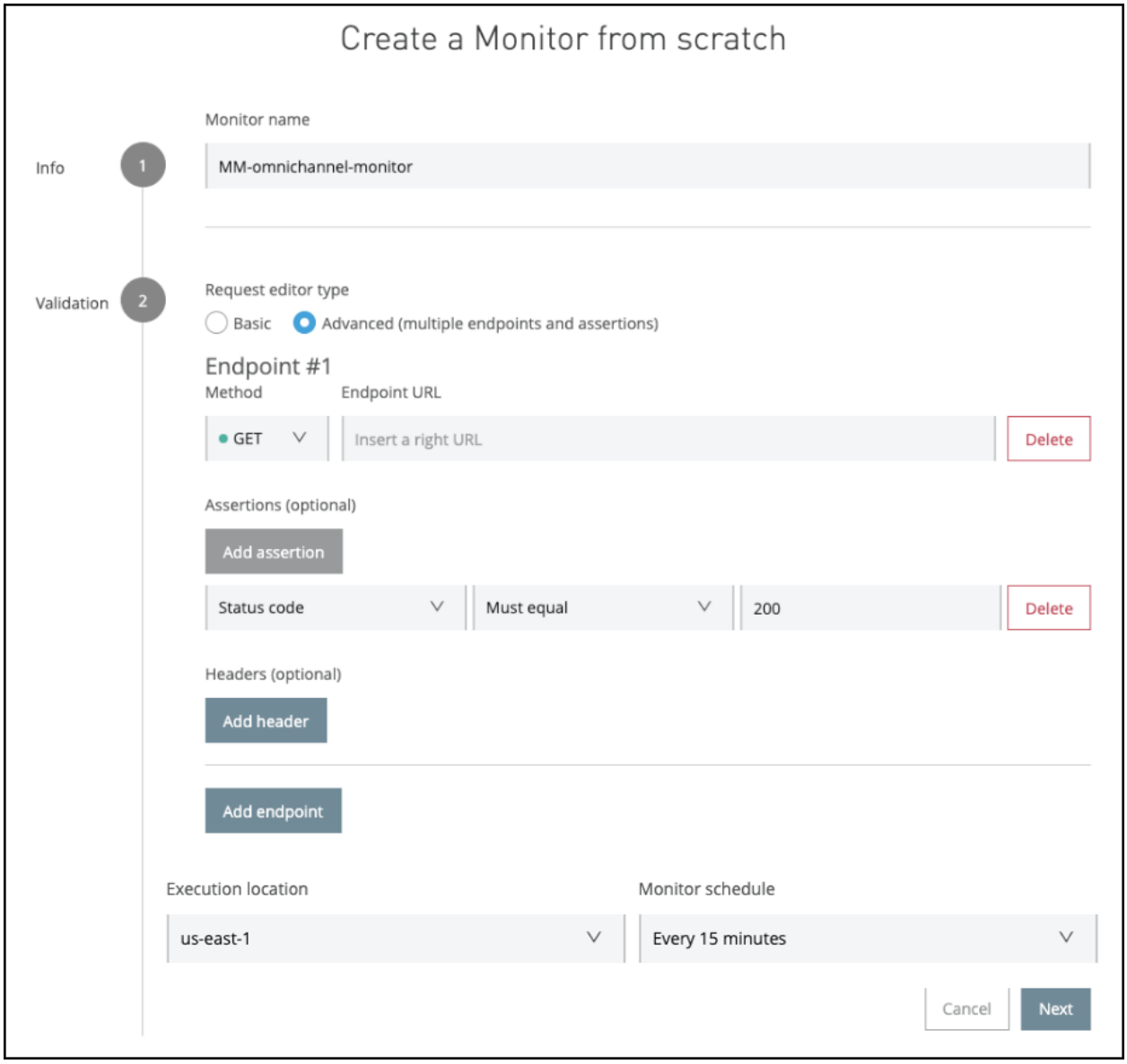


Here you are presented with a way to specify the endpoint & HTTP method that you will use to monitoring an API. There are two types of editor available, basic & advanced:

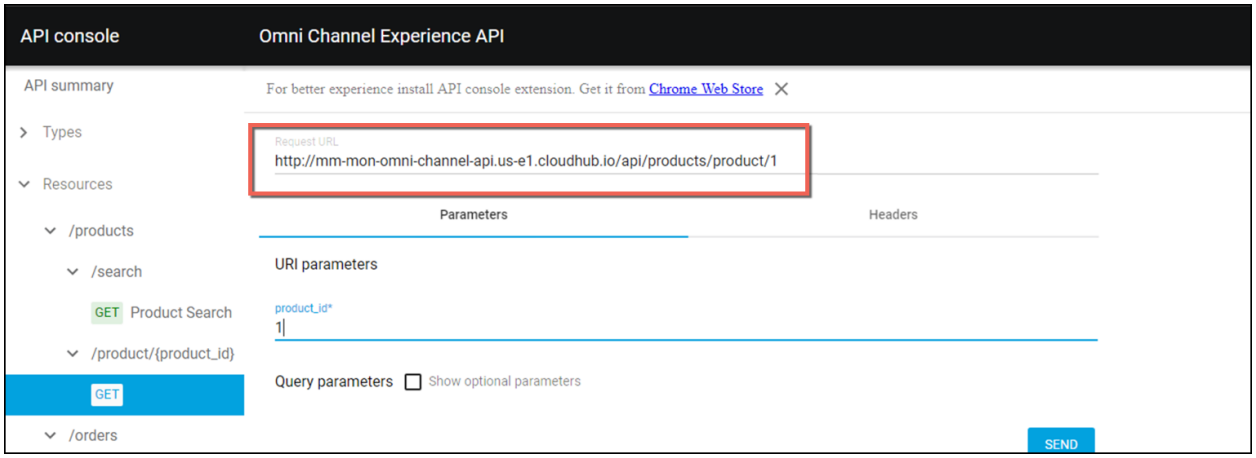
a. **Basic** provides a way to call one endpoint using one method and also lets you specify the expected response code and headers.
b. **Advanced** allows you to add multiple endpoints and create more complex assertions based on status codes, response bodies, and response headers.

4. We will build an advanced monitor with more than 1 assertion.

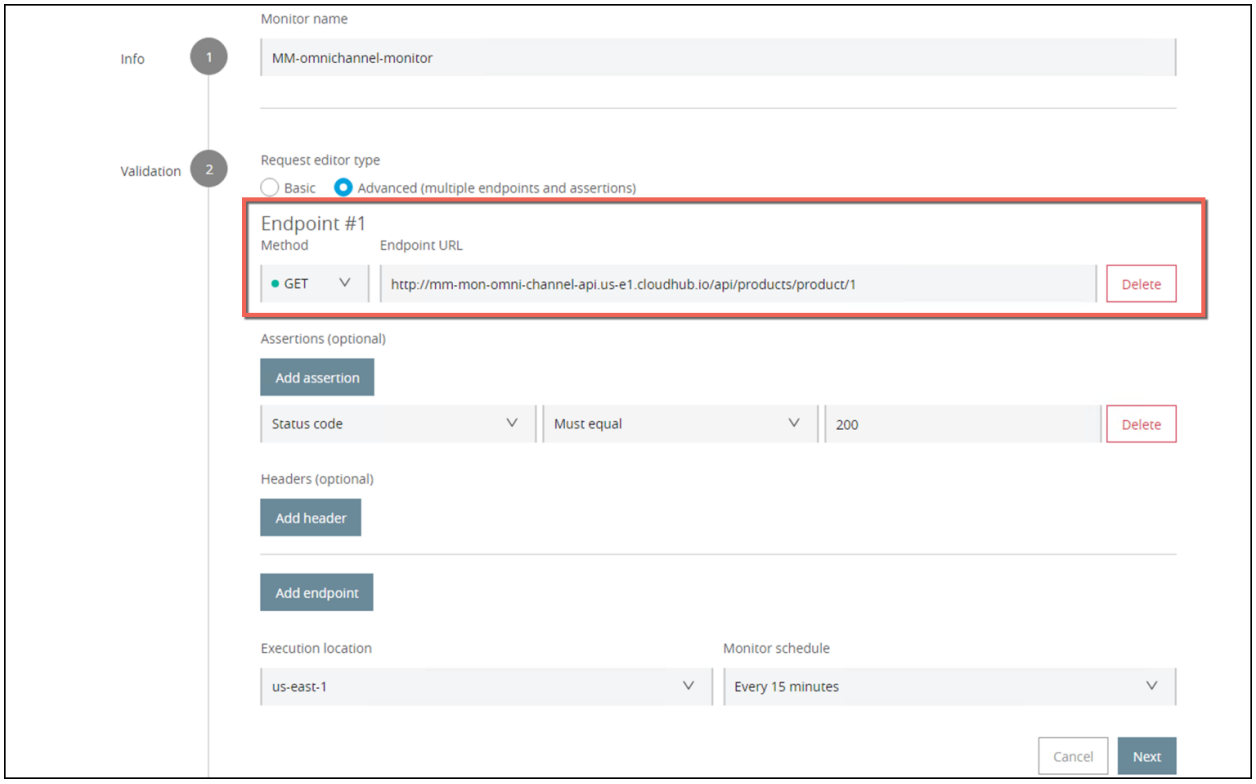
   Click the **Advanced** option and you will be presented with the following:



5. Go back to the browser tab with the API console for the omni-channel-api and copy the request url for the query that was performed



Then return to the browser tab with the functional monitor and paste in the request url



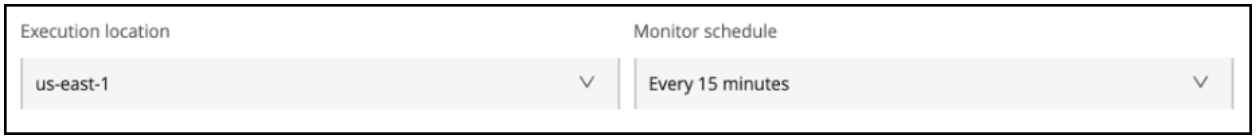
6. Now we want to add some assertions to ensure we get the functional behaviour we are expecting from the API. There are 2 assertions we are going to add:
    a. Status code must equal 200

b. Response body id must equal 1

To add an additional assertion click the **add assertion** button when needed

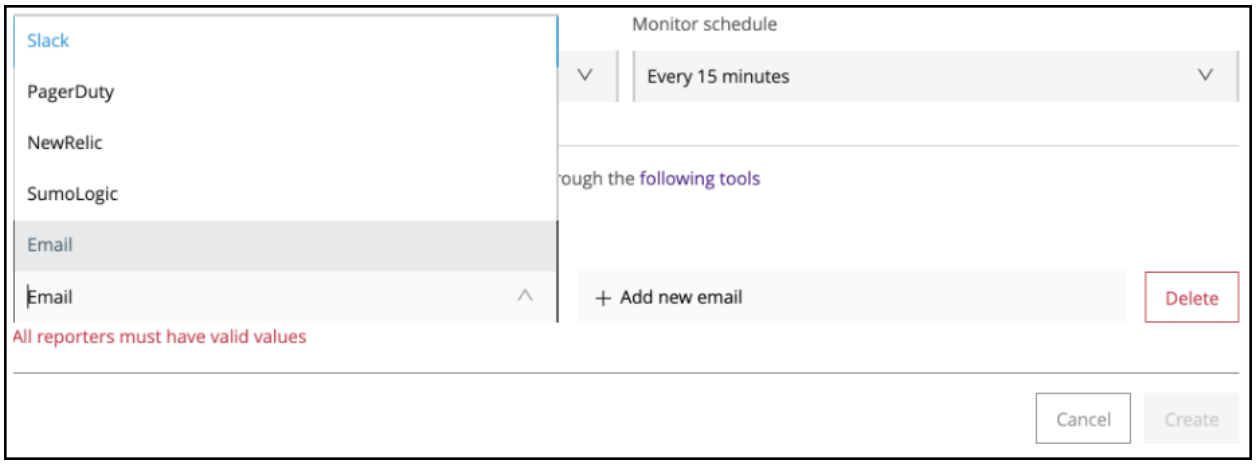
7. You then have the choices for where you want your monitor to execute from and what time schedule you want the call to occur.

   Leave the default Execution Location as us-east-1 and leave the monitor schedule at 15 minutes



   Click **Next** to continue

8. Step 3 is where we can add reporters which feed 3rd party notification tools. You will add an email reporter as part of this step.
   Click **Add Reporter** and 2 boxes will appear. On the first box open the drop down list and choose email



   Once Email is selected, move to the next box, add your email address and press **enter** on the keyboard. You can enter multiple email addresses as needed.

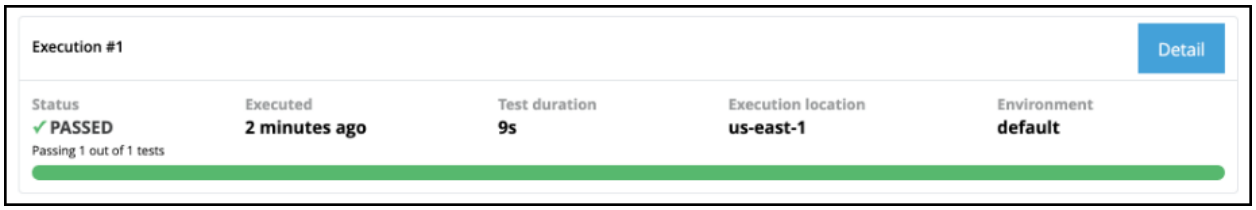
9. The configuration of the monitor is now complete. The final step is to click the create button at the bottom of the page.

   Wait a few moments for the monitor to be created and deployed. The test will execute and start to give feedback on the executions.

Step 3: Review Functional Monitor Output

1. You can view the history of the executions by clicking on the timer symbol

2. The history of the tests will show each execution at a high level



To review the test outputs click on **Detail** which shows open another view.

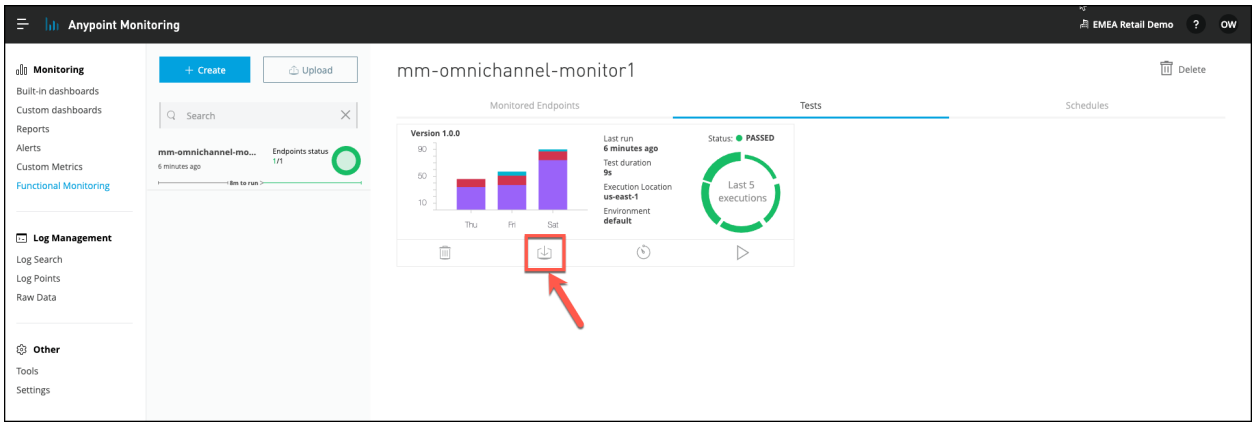
Click the down arrow on the right hand side to review further details on the execution including input HTTP and output HTTP headers and payload. If errors occur on the functional test you will get all the details on the input and

output of the call in a downloadable file.

3. Return to the main functional monitoring menu by clicking on the item in the menu on the left
4. Select your functional monitor that you created.

   You can get really advanced with functional monitors. Functional monitors can be written in Dataweave which gives you the ability to do much more with your functional monitoring activity.

5. Click the down arrow symbol to download the test file.



A zip file will be downloaded. Open the archive and then open the `main.dwl` file inside. You will see some Dataweave code that looks similar to the below:

```
import * from bat::BDD import * from bat::Assertions --- suite("mm-omnichannel-monitor1") in [
GET `http://mm-mon-omni-channel-api.us-e2.cloudhub.io/api/products/product/1` with { "headers":
{} } assert [ $.response.status mustEqual 200, $.response.body.id mustEqual 1 ] ]
```

As an extended exercise in your own time, feel free to edit these scripts and upload them back into the platform to create more advanced functional monitors. You can find all the reference material here.

Summary

In this lab, you completed the following steps:

- Tested the endpoint using the API Console
- Created and exercised a functional monitor
- Reviewed the output of the functional monitor executions and saw how the functional monitors are built up in Dataweave code.

Go Further:

- See the Functional Monitor doc for more information.

Congratulations! You have completed Lab 3, the final lab of Module 4

Take me to the TOP