Anypoint Platform Operations: Universal API Management

Student Manual

Feb 8, 2023

Table of Contents

Introducing the Course

In this module, you will:

- Learn about the course format.
- Download the course files.
- Make sure your computer is set up for class.
- Review the course outline.

Walkthrough: Set up your computer for class

In this walkthrough, you make sure your computer is set up correctly so you can complete the class exercises. You will:

- Download the student files.
- Install Advanced REST client (if you did not already).

Create a folder to store the student files for this course

1. Create a new folder named APOpsUAPIM on your computer;

   *Note: Use an easy-to-find location such as C:\APOpsUAPIM (on Windows) or ~/APOpsUAPIM (on UN*X)*

Download the student files

1. Locate your course enrollment email and follow the instructions to download the student files ZIP.
2. Download the student files zip file and save it to your APOpsUAPIM folder.

   Note: The course materials include: the student files zip file, which includes the applications you will deploy in class and other configuration files; the student manual PDF file and the student slides zip file with PDFs of the instructor slides.

Expand the student files

3. On your computer, locate the student files ZIP archive and expand it.
4. Open the course snippets.txt file.

   Note: Keep this file open. You will copy and paste text from it during class This file will be referred to as simply the snippets file or the snippets.txt file.

Open Advanced REST Client

5. Open Advanced REST Client.

   Note: If you do not have Advanced REST Client (or another REST API client) installed, download it now from https://intall.advancedrestclient.com/ and install it.

6. Leave Advanced REST client open; you will use this REST client throughout the class.

Module 1: Introducing Universal API Management on Anypoint Platform

At the end of this module, you should be able to:

- List the functionality included in universal API management.
- Describe the gateway options for managing APIs on Anypoint Platform.
- Navigate Anypoint Platform and Anypoint API Manager.

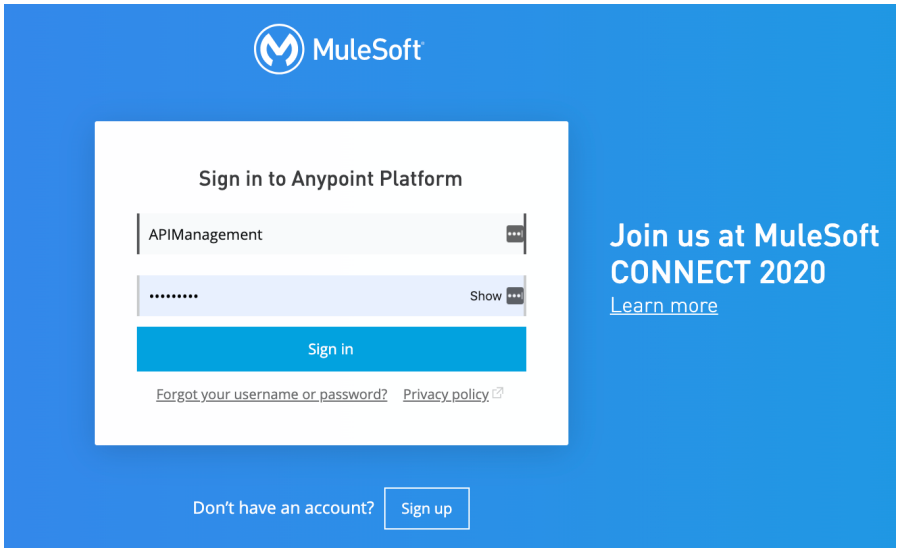
Walkthrough 1-1: Explore Anypoint Platform

In this walkthrough, you familiarize yourself with the Anypoint Platform web application. You will:

- Log in to Anypoint Platform.
- Navigate Anypoint Platform.
- Explore support, documentation, and resources.

Log in to Anypoint Platform

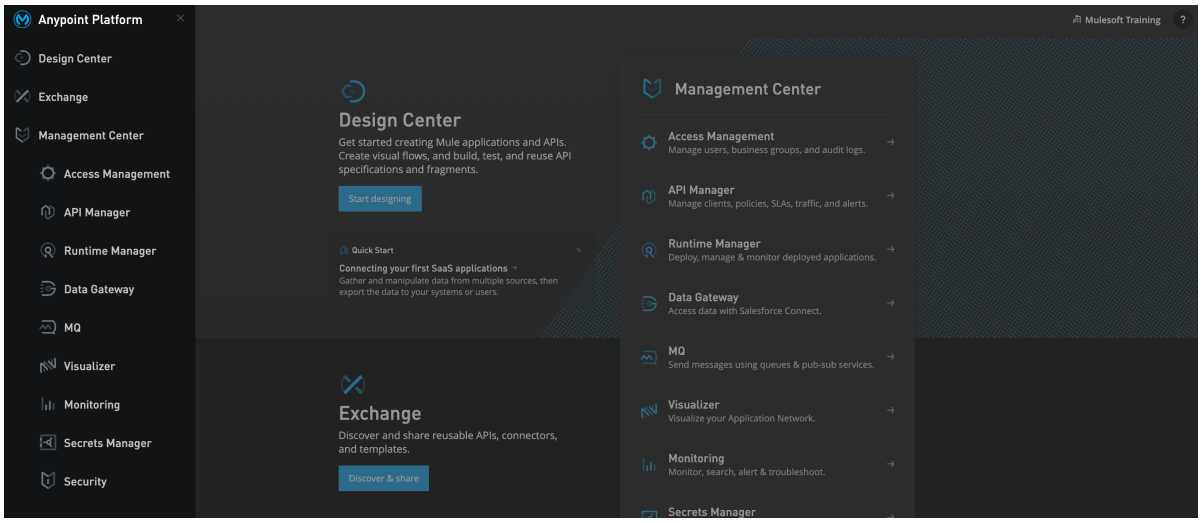
1. In a web browser, navigate to https://anypoint.mulesoft.com.
2. Log in to Anypoint Platform.



Note: Use a trial account (if you already have one). If you do not have an account, sign up for a free, 30-day trial account now.

Explore a few Anypoint Platform entitlements

3. In the upper-left corner, click the menu button.

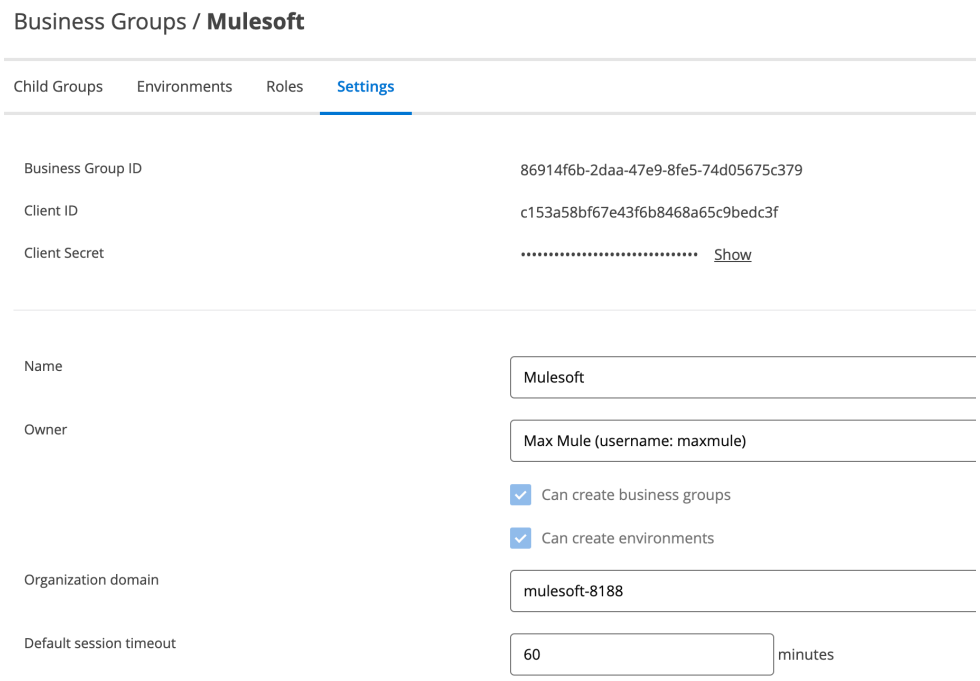


Note: A menu should appear that includes links to your Anypoint Platform entitlements. This will be called the main menu from now on.

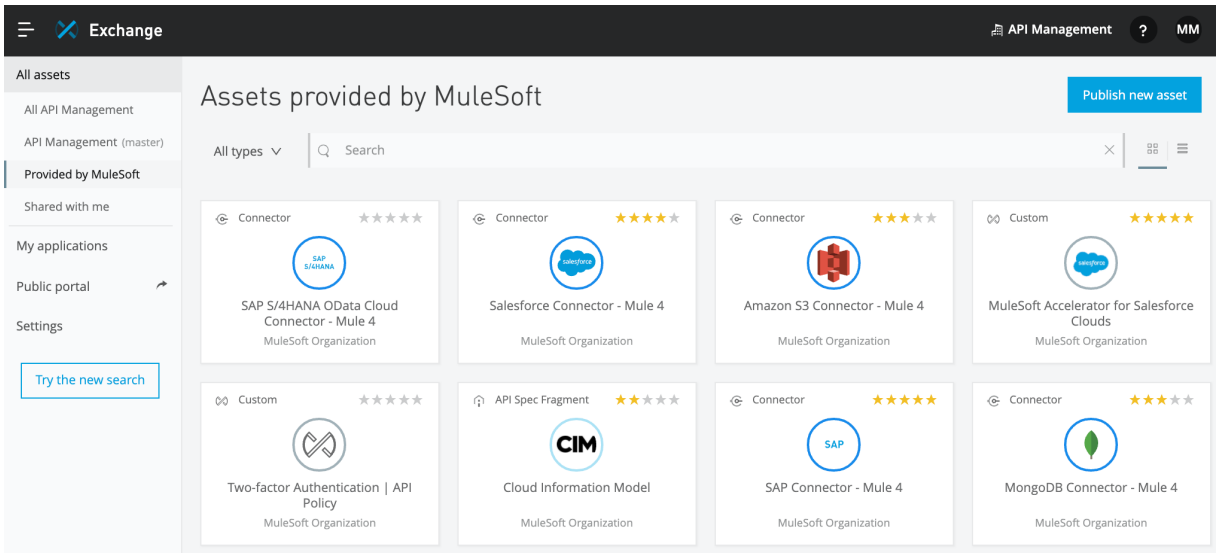
4. In the main menu, select **Access Management.**
5. Click the button labeled **Show New Features**, to enable the new Teams features.

   Note: If you did this previously, the button will display as Hide New Features. In this case don't click it.

6. In Access Management, click on **Business Groups**.
7. Select the name of your business group.
8. Select the **Settings** tab.
9. Verify that you can view the details of your business organization (like organization id, domain name, default session timeout period).



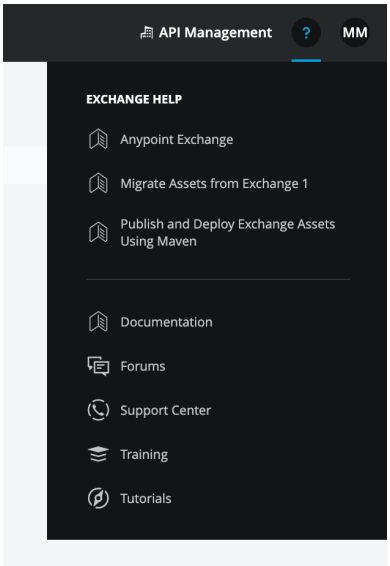
10. In the main menu, select Exchange.



Note: Exchange contains public MuleSoft assets and assets that are private to your business organization. You can customize documentation of an asset you publish to Exchange, make them public or share them with specific users in your organization.

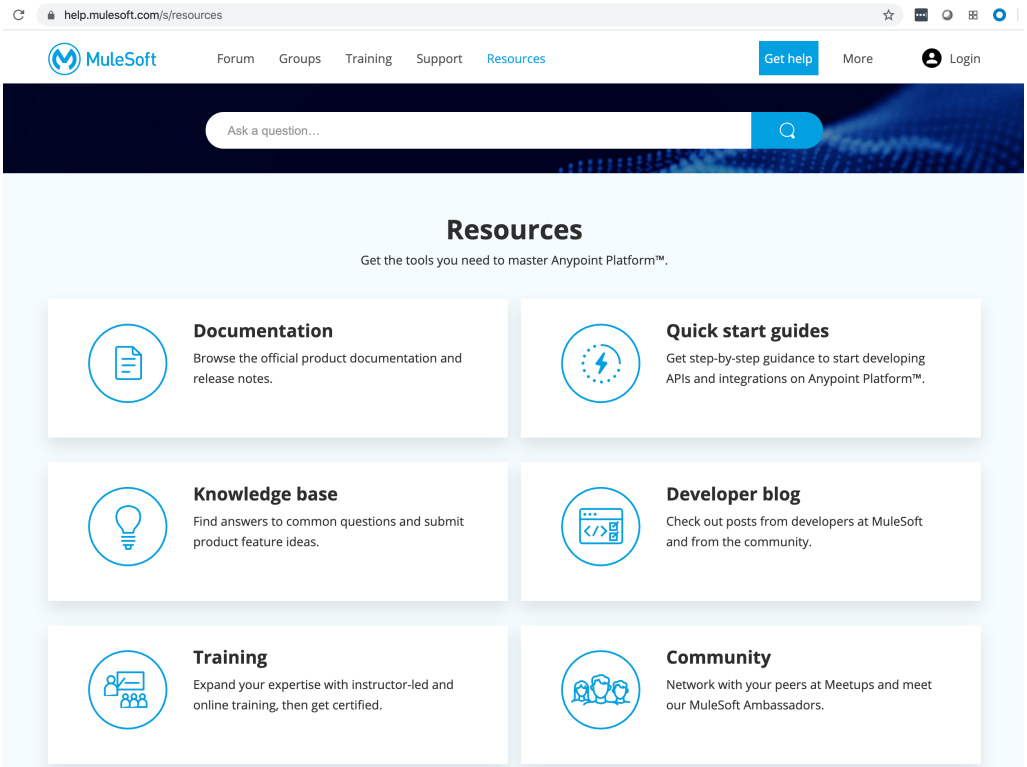
Explore Anypoint Platform support and documentation resources

11. Click the ? (Support) button located in the upper-right corner of the window.



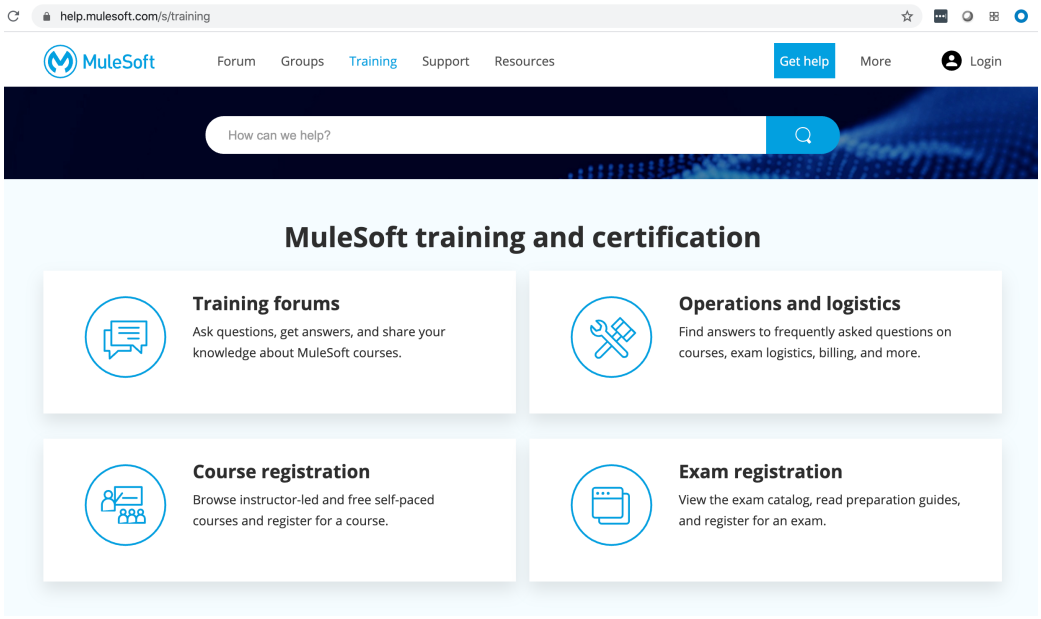
Note: The shortcuts inside the ? menu are also available in the homepage of Anypoint Platform.

12. In the Support menu, select Documentation; the MuleSoft documentation should open in a new browser tab.

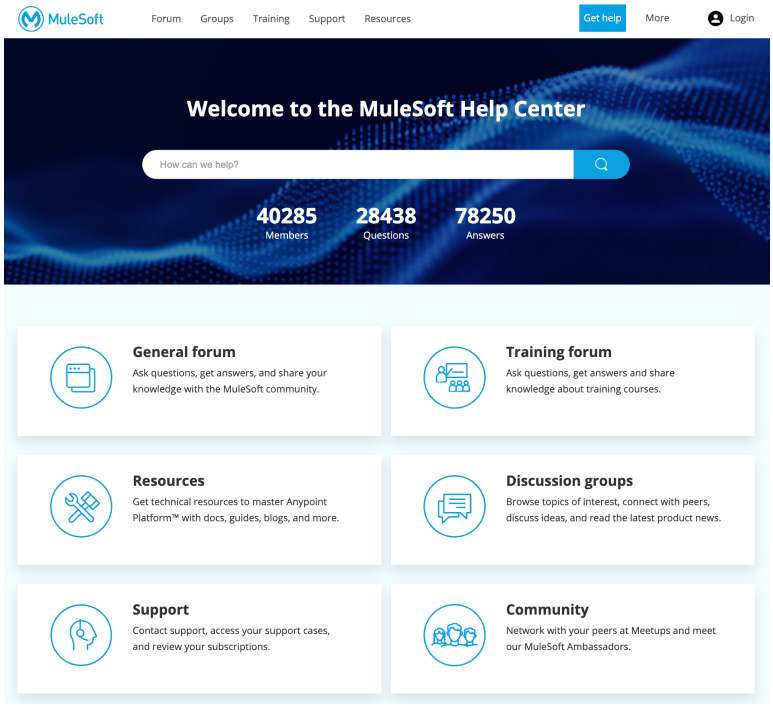


Note: The documentation site is a great resource while learning to use the Anypoint Platform.

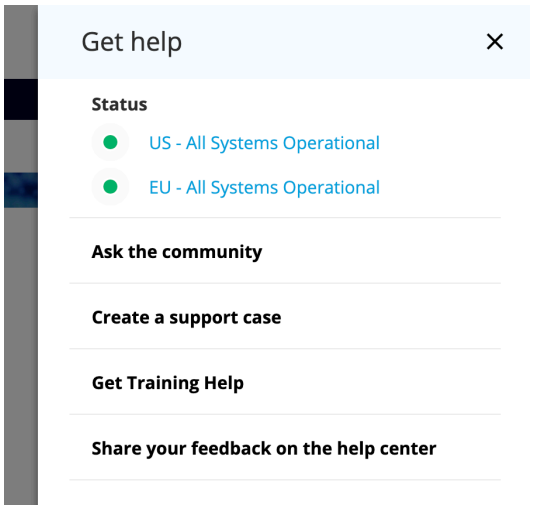
13. Return to the browser tab with Anypoint Platform open, and click ? -> Training. The MuleSoft Training and Certification page should open in a new browser tab.



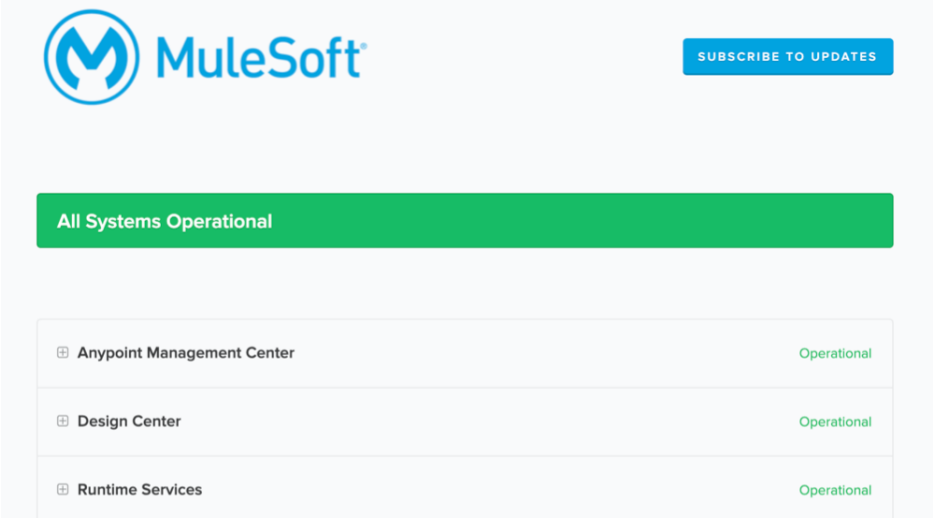
14. Similarly, return to the browser tab with Anypoint Platform open, and click the Support Center link. A new tab should open with the MuleSoft Help Center.



15. In the MuleSoft Help Center page, click the Get Help button on the top right corner to open Get Help Menu.

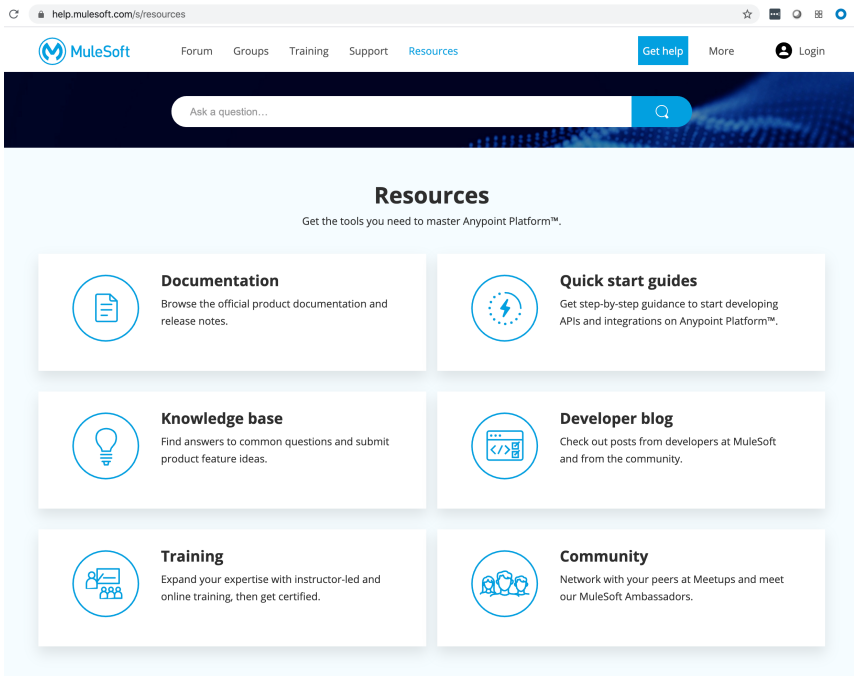


16. From the Get Help Menu, click the US – All Systems Operational link under Status. The status page should open in a new tab.

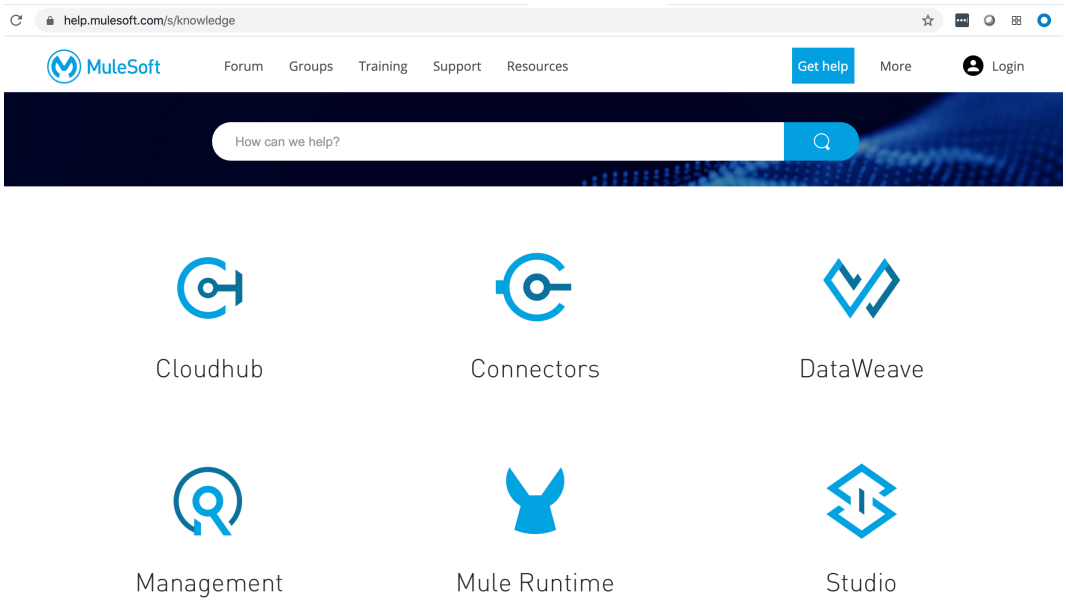


Note: In the status page, you can click the entitlements and services listed to check latest status updates. You can receive emails about the status updates by clicking the subscribe to updates button.

17. In the MuleSoft Help Center page, click the Resources card.


help.mulesoft.com/s/resources
MuleSoft
Forum
Groups
Training
Support
Resources
Get help
More
Login
Ask a question…
Resources
Get the tools you need to master Anypoint Platform™.
Documentation
Browse the official product documentation and release notes.
Quick start guides
Get step-by-step guidance to start developing APIs and integrations on Anypoint Platform™.
Knowledge base
Find answers to common questions and submit product feature ideas.
Developer blog
Check out posts from developers at MuleSoft and from the community.
Training
Expand your expertise with instructor-led and online training, then get certified.
Community
Network with your peers at Meetups and meet our MuleSoft Ambassadors.

18. In the Resources Page, click the Knowledge Base card.



Note: The Knowledge page links to trending and related articles organized by topics, and also contains FAQ docs related to each topic.

19. Return to the browser tab that contains Anypoint Exchange.
20. In the main menu, select Anypoint Platform to return to the home page.

Module 2: Discovering and Consuming APIs

At the end of this module, you should be able to:

- Create APIs in Anypoint Design Center using RAML specifications.
- Publish, document, and test API specifications in Anypoint Exchange.
- Control API access using Anypoint Exchange portals.
- Discover APIs through public portals.
- Catalog APIs with Anypoint CLI.
- Describe how to extend the capabilities of Anypoint Exchange with API Community Manager.

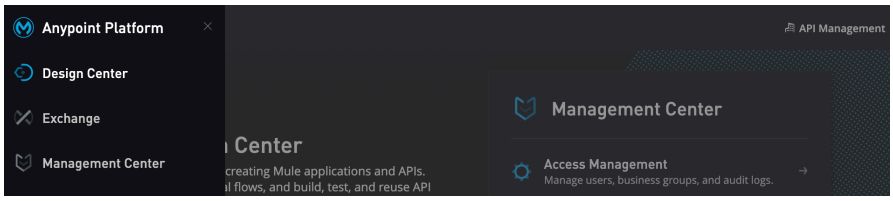
Walkthrough 2-1: Create API specifications in Design Center

In this walkthrough, you create APIs by importing in to Design Center. You will:

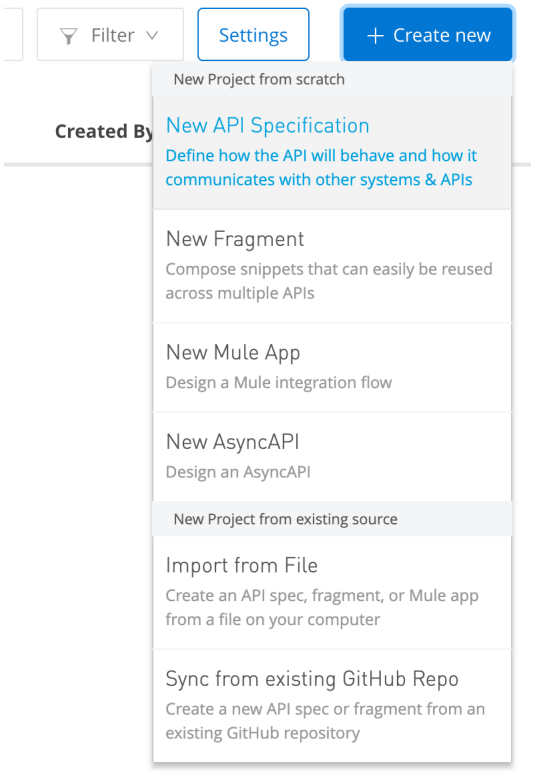
- Create new API Specification projects.
- Import API Specifications into Design Center.

Create an API specification in Design Center

1. Return to Anypoint Platform.
2. In the main menu, select Design Center.

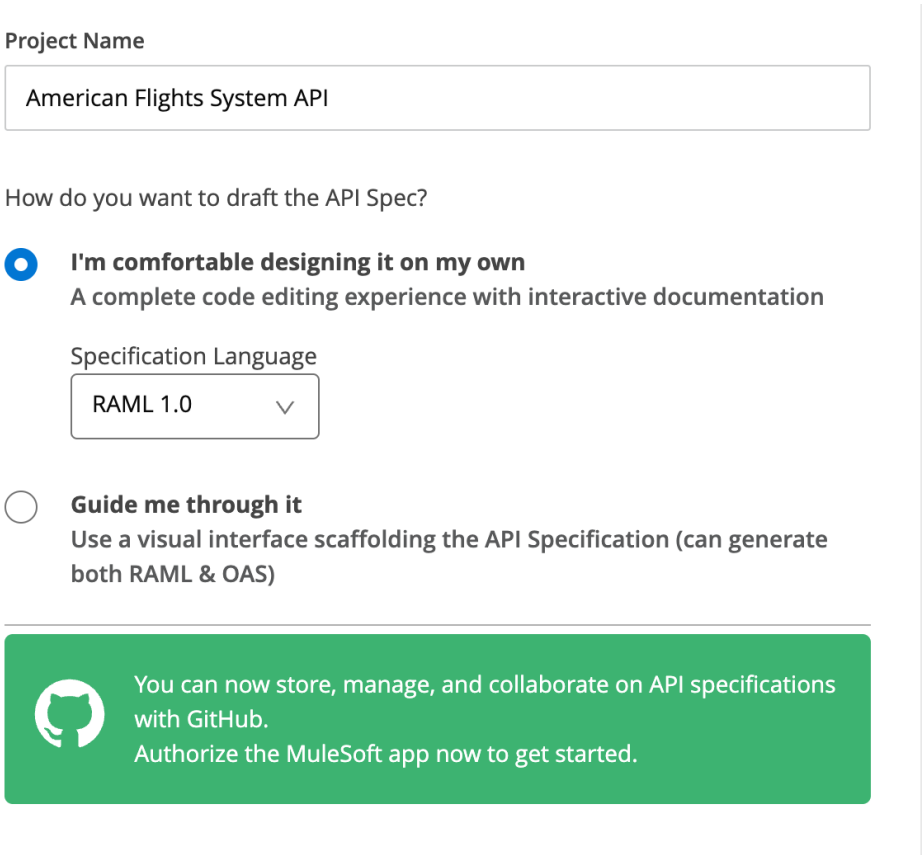


3. In the Design Center Projects page, click +Create new and select New API Specification from the options.



4. In the New API Spec dialog box, select **I'm comfortable designing it on my own** and enter the following details:

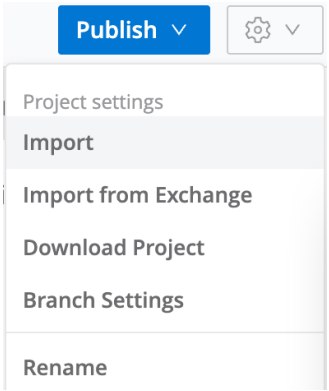
   - **API Title**: American Flights System API
   - **Specification Language:** RAML 1.0



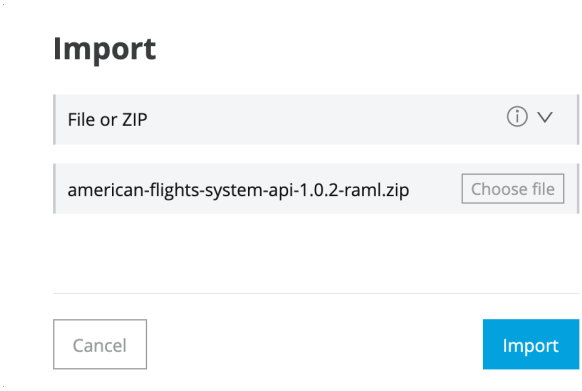
5. Click Create API.

Import the American Flights System API specification in to Design Center

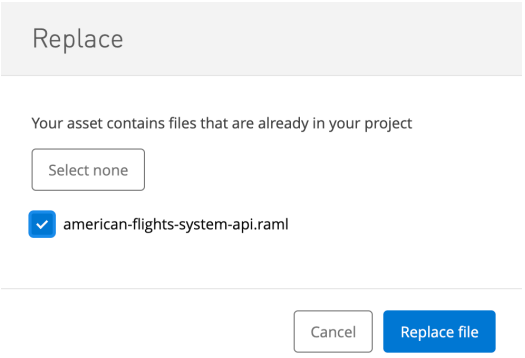
6. Open the project settings menu and select Import.



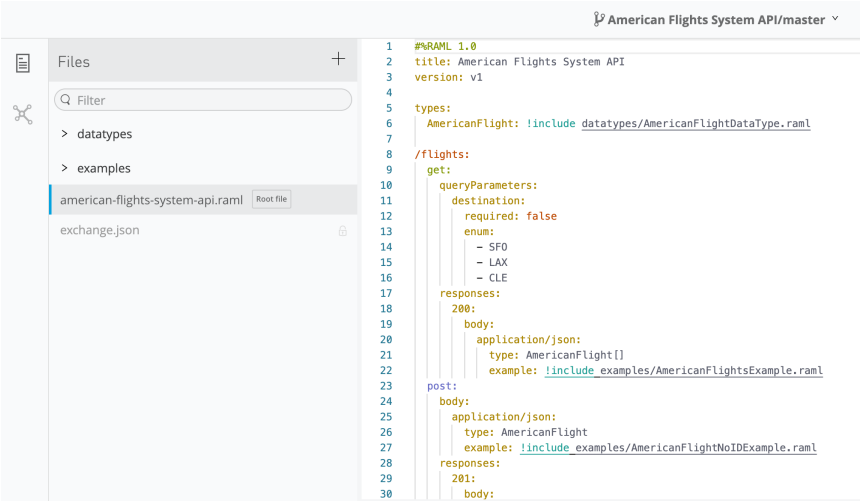
7. In the Import dialog box, select **File or Zip** if it is not already selected.
8. Click **Choose file** and browse to the location of the student files folder.
9. Locate and select american-flights-system-api-1.0.2-raml.zip.
10. Click Open, then Import.



11. In the Replace files dialog box, select american-flights-system-api.raml and click Replace file.

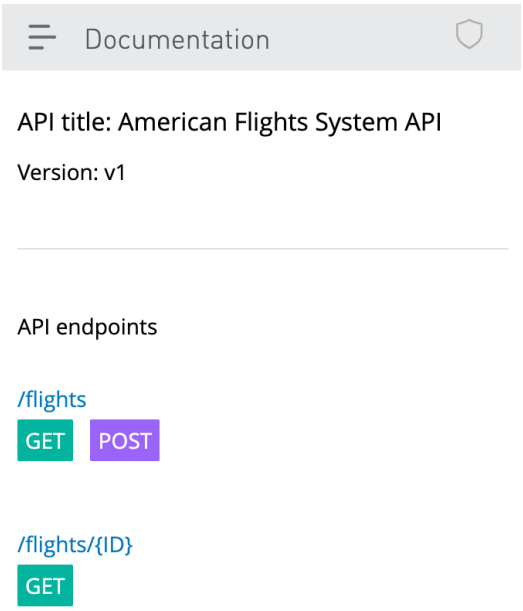


12. In the file browser section, click american-flights-system-api.raml.



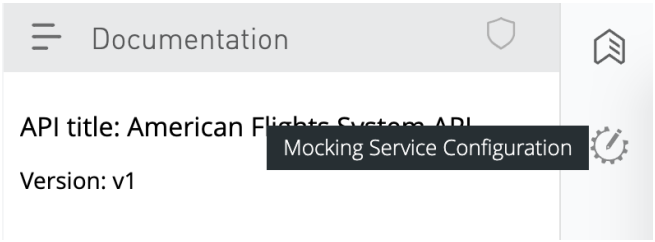
13. In the right-side panel, click on the Documentation Icon.

14. view the API Documentation with information such as types, resources, and methods.

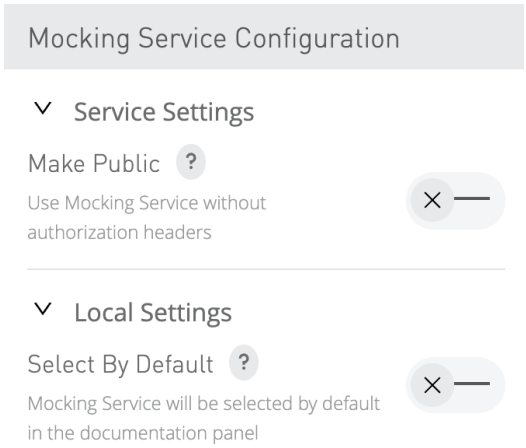


Note: In the rest of the walkthroughs, a panel similar to this one will be called the API Console. You will use the API Console to test endpoints of the API.

15. Click the Mocking Service Configuration icon below the documentation icon.



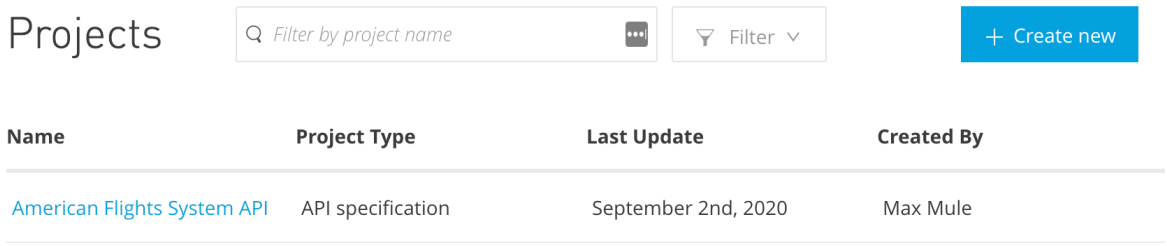
16. Inspect the Mocking Service Configuration panel.



Note: From this panel you can enable the mocking service.

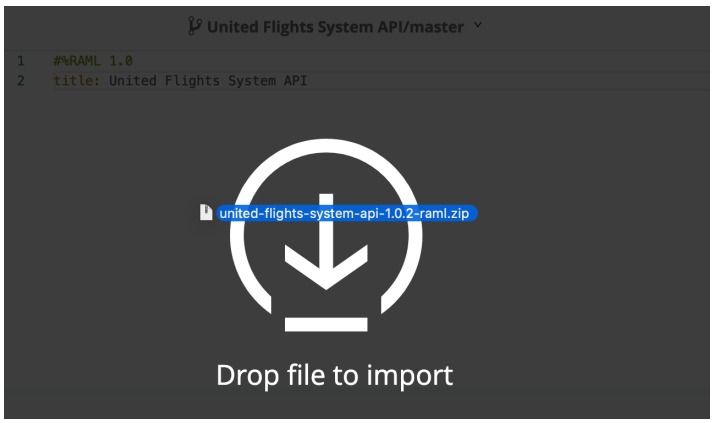
Add United Airlines API Specification project to Design Center

17. Return to Design Center Projects page by clicking on the header.
18. Click the + Create new button.



19. From the menu select New API Specification.
20. Set the API Title as United Flights System API for the new specification.
21. Leave the rest of the default values and click **Create API**.
22. Locate united-flights-system-api-1.0.2-raml.zip in your student files folder.

23. Drag and drop the zip file in the design center area.



24. Click Import.
25. Click to Select all to replace all files.
26. Click Replace file.

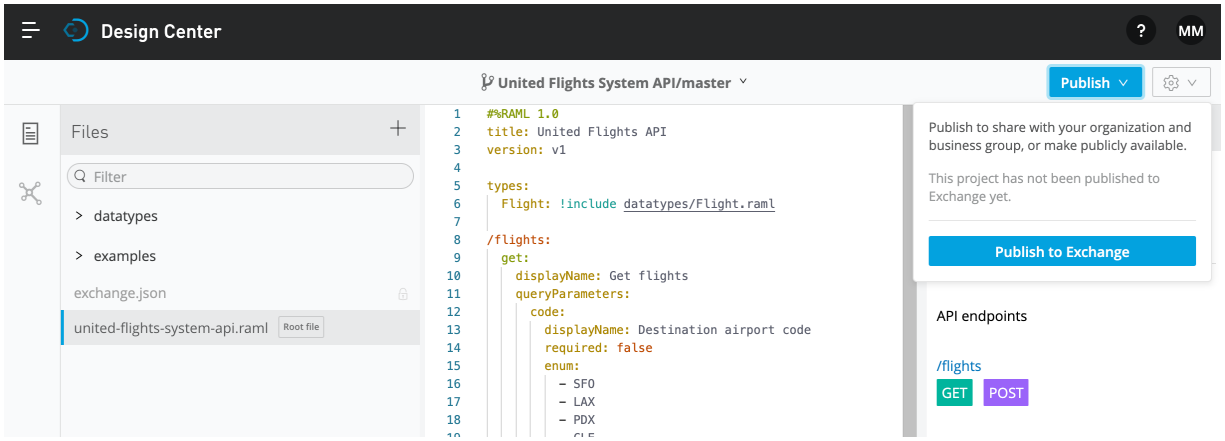
Walkthrough 2-2: Document, test, and share an Anypoint Exchange asset

In this walkthrough, you create and modify API documentation to better engage with API consumers. You will:

- Publish an API to Anypoint Exchange.
- Use Markdown language to add content to an Exchange portal.
- Publish and view an updated Exchange portal.
- Test an API using the API Console in the API's Exchange portal.
- Add a private Anypoint Exchange asset into a public portal.

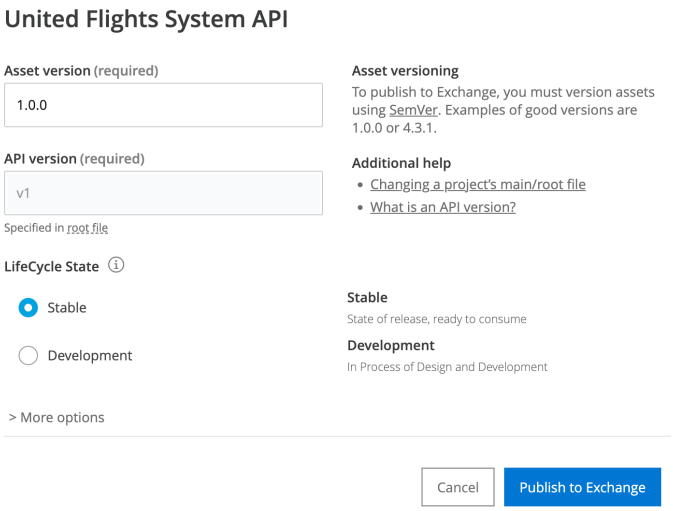
Publish United API specification to Anypoint Exchange

1. In the top-right corner of API designer, click the Publish button.



2. From the dialog box click Publish to Exchange.

3. In the Publish to exchange dialog box, set the following values:
    - **Asset version**: 1.0.0
    - **API version**: v1 (pre-filled)



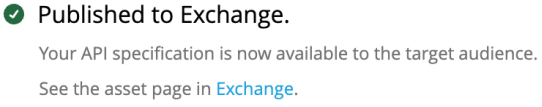
4. Click the Publish to Exchange button.
5. Click Close in the Published to Exchange confirmation dialog.

Publish American API specification to Anypoint Exchange

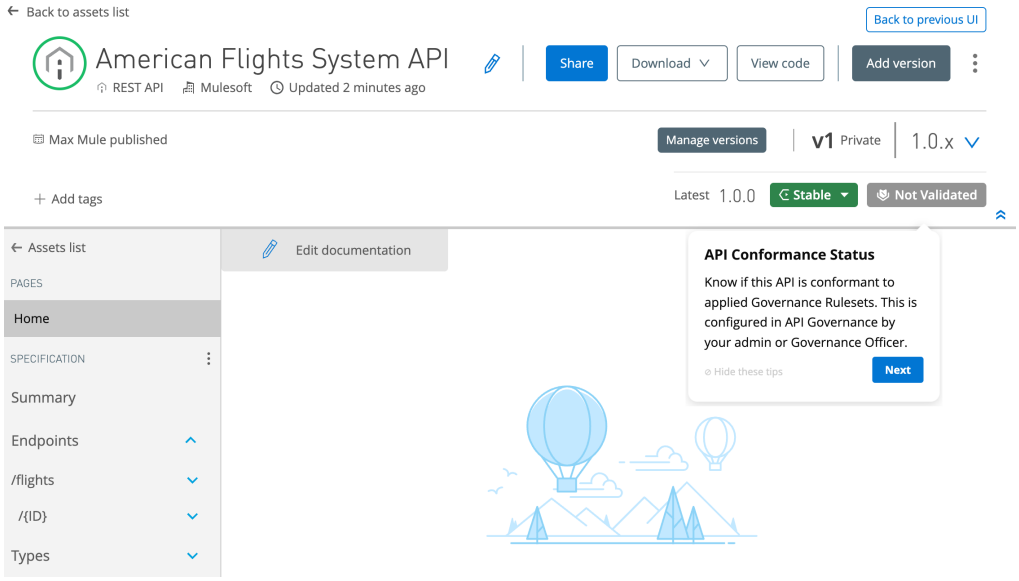
6. Return to the Projects page in design center by clicking on the header.
7. Select and open the American Flights System API project.
8. In the top-right corner of API designer, click the Publish button.
9. From the dialog box click Publish to Exchange.
10. In the Publish to exchange dialog box, set the following values:
    - **Asset version**: 1.0.0
    - **API version**: v1 (pre-filled)
11. Click the Publish to Exchange button.

Use Markdown language to add content to an Exchange portal

12. In the Successfully published to Exchange dialog box, click the Exchange link.



13. In the American Flights System API Exchange portal page, click Edit documentation.



14. View the course snippets file.

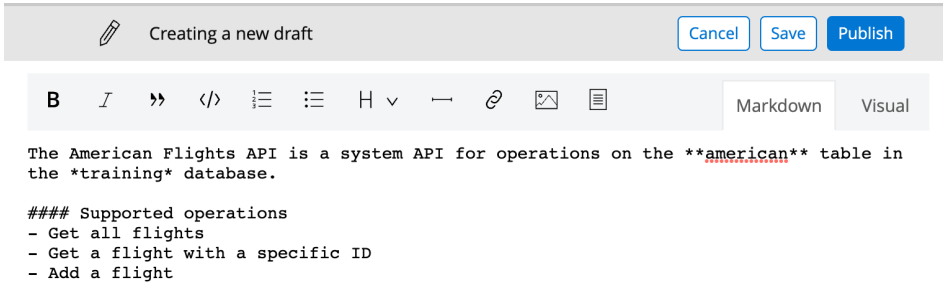
    Note: If it is not already open, navigate to the student files directory and open the course snippets file.

15. Below the Module 2 section, copy all the markdown formatted text.

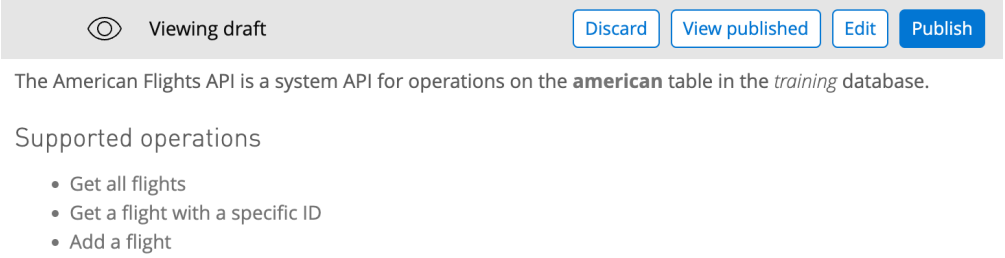
```
The American Flights API is a system API for operations on the **american**
table in the *training* database.

#### Supported operations
- Get all flights
- Get a flight with a specific ID
- Add a flight
```

16. Return to the Exchange portal editor and paste the Module 2 Markdown text.



17. Click Save; you should see the Markdown formatted text as it will be displayed to viewers.

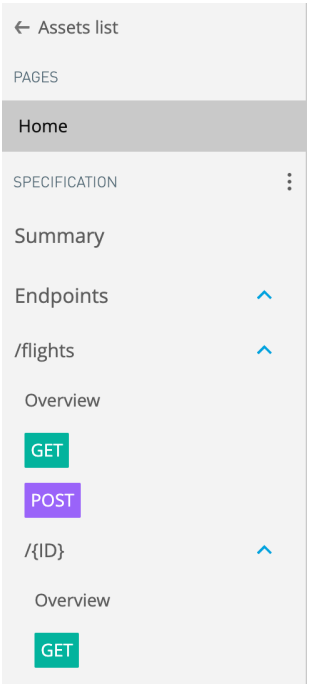


Publish the edited API page and view the updated Exchange portal page

18. In the upper right, click Publish.
19. Verify you can view the API overview in the American Flights System API Exchange portal and the description is updated with the Markdown text.

Test the API asset using the API Console

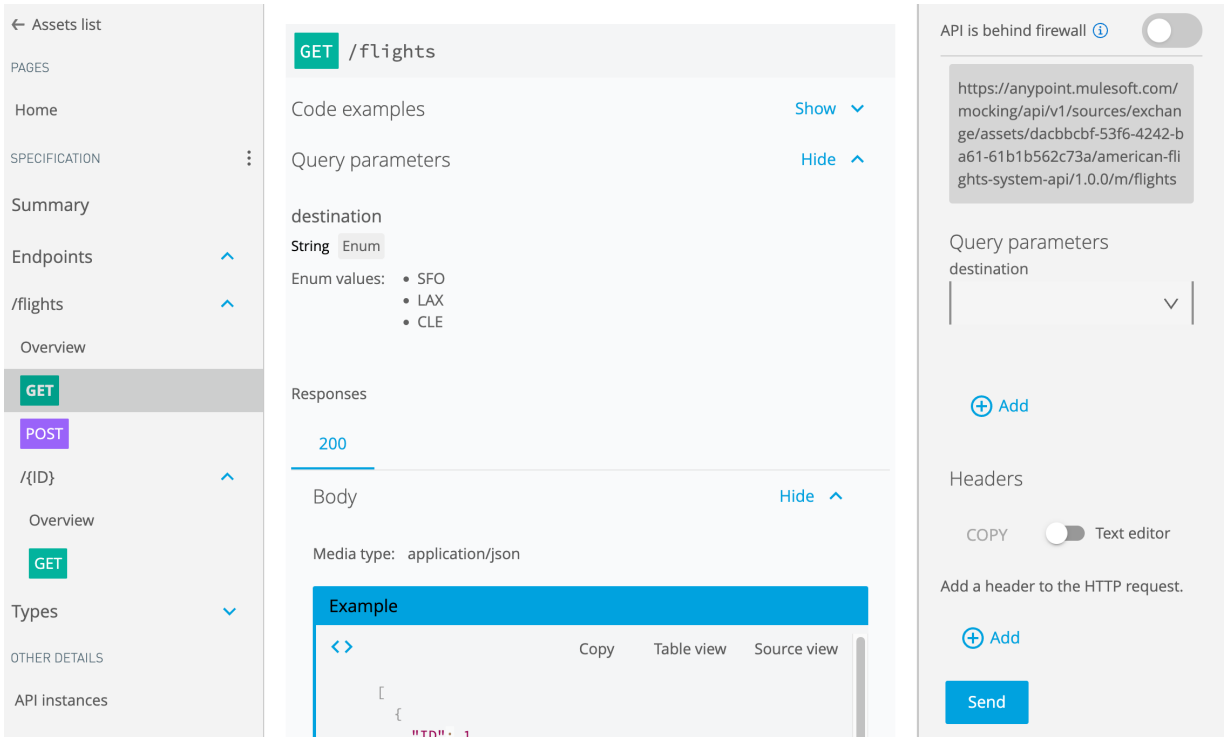
20. In the left-side API Console, expand the /flights and /{ID} resources and notice the supported HTTP methods.



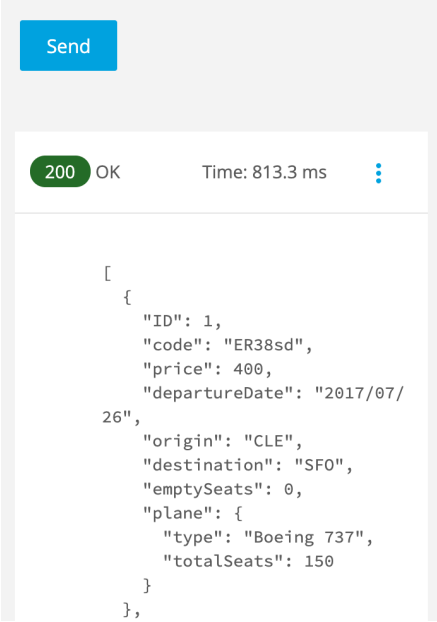
Note: This is the same API Console you saw in the API Designer, but here it is on the left side, not the right side.

21. Click the /flights GET method.

22. In the right-side API method details panel, click Send.



23. Verify that you see a 200 OK response with an array of AmericanFlight JSON objects.

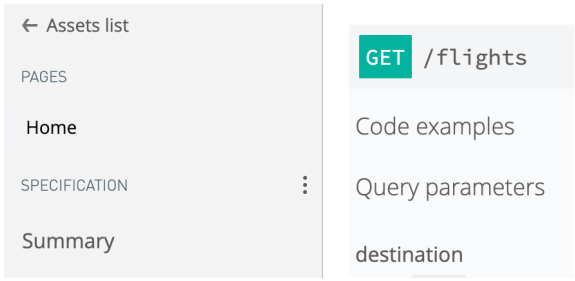


24. Notice the URL the GET was sent starts with https://anypoint.mulesoft.com/mocking.

Note: This endpoint is automatically generated for every Exchange portal hosted API. It returns response data hardcoded in the API specification file. Notice the response payload is an array

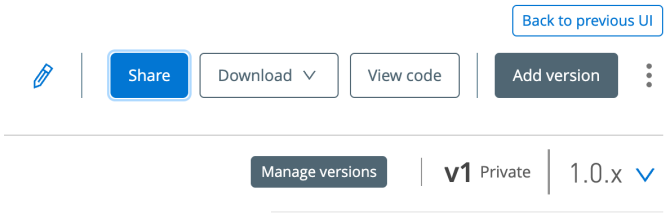
of flight objects, with a plane key whose value is a child object nested two levels deep in the JSON hierarchy. You'll use this structure later in the class to test some API policies.

25. In the left-side navigation, click Home.

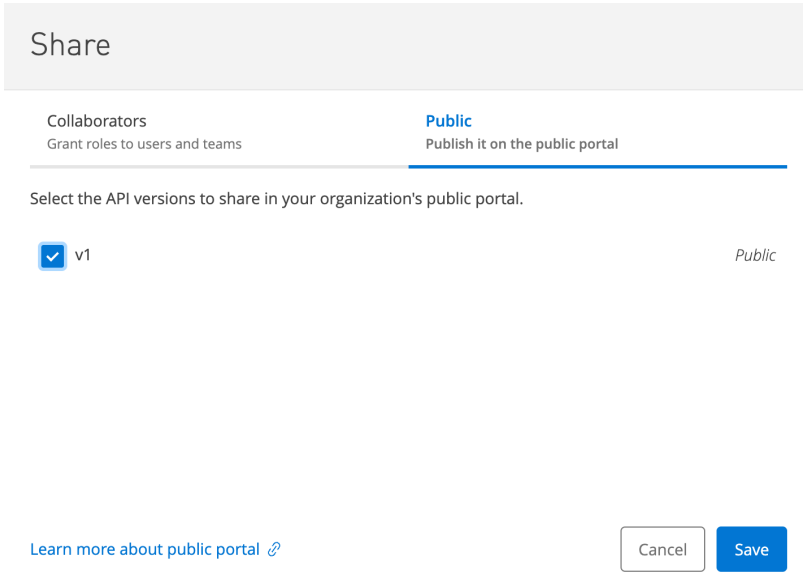


Return to the exchange browser window

26. In the upper-right, click the Share.

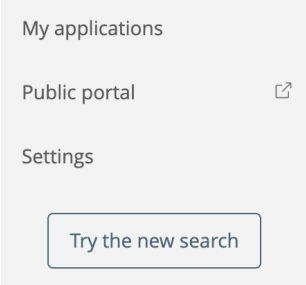


27. In the Share dialog box select the Public tab.

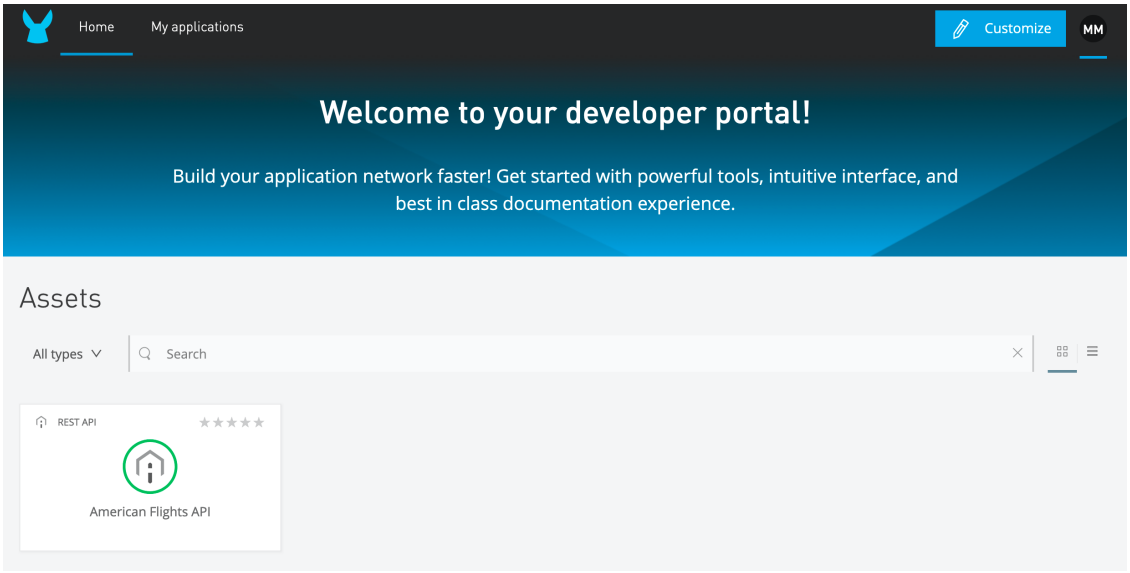


28. Verify that the version v1 of the API is checked.
29. Click Save.
30. In the Exchange portal left-side navigation, click Assets list.

31. In the left-side navigation, click Public portal.



32. In the developer portal webpage, verify you see a card for the American Flights System API.



Note: This is the public portal that is accessible to all users outside of your organization.

Module 3: Managing APIs using Anypoint Flex Gateway

At the end of this module, you should be able to:

- Add and register a Flex Gateway
- Expose an API through Flex Gateway
- Apply policies to APIs managed by Flex Gateway
- Add external log forwarding to Flex Gateway
- Scale Flex Gateway using replicas
- Secure Flex Gateway using TLS

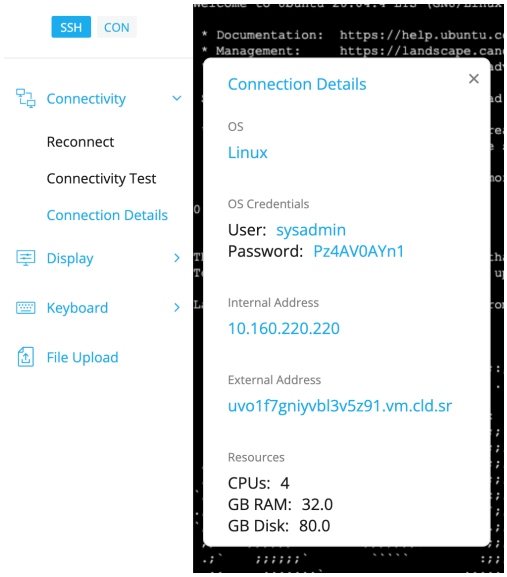
Walkthrough 3-1: Installing Flex Gateway using docker

In this walkthrough you register and run an instance of Flex Gateway in connected mode. You will:

- Create a connected app with the required scopes to install Flex Gateway
- Register Flex Gateway in connected mode using docker
- Start a Flex Gateway replica using docker

Set up your environment

1. In Cloudshare open the Linux Server environment.
2. From the left menu select Connectivity – Connection Details



3. Copy the sysadmin password and external address in your snippets file.
4. SSH into the Linux VM with Ubuntu Focal installed, replace <<external-address>> with the right value from your snippets file and provide the sysadmin password when prompted.

```
ssh sysadmin@<<external-address>>
```

5. Create a folder called flex and move into it.

```
mkdir flex; cd flex
```

6. Create another folder to store the credentials and custom configurations.

```
mkdir conf; cd conf
```

Create a connected app to be used by Flex Gateway commands

7. Return to the browser tab with Anypoint Platform open.
8. In the main menu under Management Center, select **Access Management**.
9. From the left menu select **Connected Apps**.
10. Click **Create app**.
11. Enter the following details:

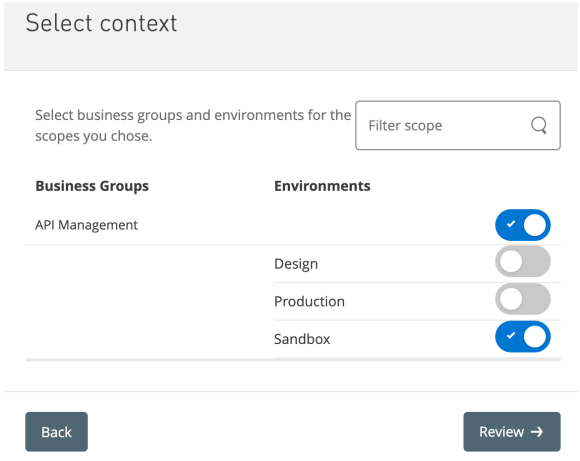
    - **Name**: flex-gw
    - **Type**: App acts on its own behalf (client credentials).

12. Click Add Scopes.
13. On the Select scopes dialog add the following scopes:

    - Read Servers
    - Manage Servers
    - View Organization

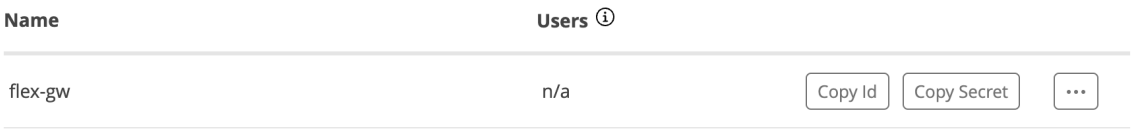
    Note: If you are using a child business group to install Flex Gateway the View Organization scope should include the master organization as well as your child business group.

14. Click Next.
15. On the Select context dialog select the name of the business group that contains the target environment.
16. Select the target environment e.g. Sandbox.



17. Click Review and confirm that the required scopes are listed.
18. Click Add scopes.
19. Click Save.

20. Use the Copy Id and Copy Secret buttons to save the credentials for the connected app into the snippets file under section Connected App.



Register a Flex Gateway in connected mode instance using docker

21. Navigate to API Manager.
22. Switch to the environment that will contain the Flex Gateway instance.
23. Click the environment information button.
24. Copy the Environment ID and Business Group ID and save it in your snippets file.
25. Return to your SSH console.

    Note: make sure your current location is /home/sysadmin/flex/conf

26. Download the docker image for Flex Gateway.

```
docker pull mulesoft/flex-gateway
```

    Note: by default it will download the image tagged as latest.

27. List all the images available to your docker instance.

```
docker images
```

28. Confirm that the results contain the mulesoft flex-gateway image.

```
REPOSITORY              TAG       IMAGE ID       CREATED       SIZE
mulesoft/flex-gateway   latest    4ba042860c8f   4 weeks ago   273MB
```

29. In your snippets file locate the command to register Flex Gateway using Docker.

```
docker run --entrypoint flexctl -w /registration -v $(pwd):/registration
mulesoft/flex-gateway \
register \
--client-id=<<client-id>> \
--client-secret=<<client-secret>> \
--environment=<<environment-id>> \
--organization=<<business-group-id>> \
--connected=true \
flex-gw
```

Note: The gateway name is the last parameter of the command, in this case flex-gw.

30. Replace the placeholders for client-id, client-secret, environment-id and business-group-id with the real values found in the previous steps.

    *Note: the value **flex-gw** is the name that will be given to the Flex Gateway instance. It could be any value but must be unique per environment.*

31. Execute the command for registration with Anypoint Platform using the real values located in your snippets files.
32. Confirm the command executed successfully by checking that the output matches bellow.

```
Starting registration, please be patient.
Registration completed, the configuration files were written in directory "."
```

33. List the files in the current directory and confirm there is one file called registration.yaml.

    Note: If the optional split flag is set to true another file called certificate.yaml will be present.

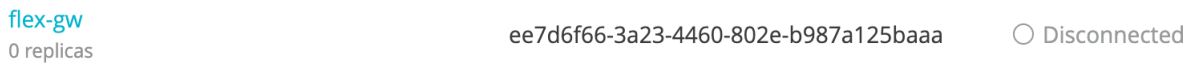
34. Open the file with the vi editor.

```
vi registration.yaml
```

35. Note the agentID under spec.platformConnection.
36. Scroll to the bottom of the file, note the tls.cert and tls.key.

    Note: This are embedded in the registration file because the split flag was not set; otherwise it will be contained on the certificate.yaml file.

37. Switch to the browser tab with Anypoint Platform and navigate to Runtime Manager.
38. Select Flex Gateways from the left menu.
39. Confirm that you can see a Flex Gateway instance called flex-gw with status disconnected and the agentID is the same as the targetID.



Start a Flex Gateway using Docker

40. Locate the command to start with docker in your snippets file.

```
docker run -d --rm \
-v $(pwd):/usr/local/share/mulesoft/flex-gateway/conf.d\
-p 8081:8081 \
-e FLEX_NAME=replica-1 mulesoft/flex-gateway
```

Note: The FLEX_NAME could be any value and will be the name given to the first replica.

41. Execute the command.

    Note: Since the docker command ran with the detached mode (-d) the only ouput would be the container id instead of any logs and the container will be running in the background.

42. List all the running docker containers.

```
docker container ls
```

43. Confirm that you see a container ID matching the first values of the one printing when the command executed. The status should be Up and the image will be mulesoft/flex-gateway.

```
CONTAINER ID   IMAGE                  COMMAND   CREATED         STATUS         PORTS                                        NAMES
39c69f199b34   mulesoft/flex-gateway  "/init"   4 minutes ago   Up 4 minutes   0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   serene_allen
```

44. Return to Runtime Manager – Flex Gateways in Anypoint Platform.
45. Confirm the status of your Flex Gateway instance changed to Connected.

Walkthrough 3-2: Forward Flex Gateway access and runtime logs to an external location

In this walkthrough you will aggregate access and runtime logs generated by the Flex Gateway. The output format will be a file located in an external directory. You will:

- Deploy an API to Flex Gateway
- Access the runtime logs available within the docker containers
- Create a configuration file to add a fluentbit output
- Examine the fluentbit configuration within Flex Gateway
- Apply a message logging policy

Deploy an API to Flex Gateway

Note: The JSONPlaceholder API is free fake API available in online. We will be creating a proxy to access it for teaching purposes.

2. Navigate to API Manager.
3. Click Add API and select Add New API.
4. Select Flex Gateway as the Runtime if not already selected.
5. From the list of gateways select flex-gw, or the name given in the previous walkthrough.
6. Click Next.
7. Under API select Create new API.
8. Enter the following values:
    - Name: json-placeholder
    - Asset types: HTTP API
9. Click Next.
10. Configure the Endpoint with the following values:
    - Implementation URI: https://jsonplaceholder.typicode.com
    - Base path: /
11. Expand Advanced options.

    Note: Notice port 8081 that this API will be listening to matches the port exposed by the Flex Gateway running in docker.
12. Click Next.
13. Click Save & Deploy.

14. From your terminal call the endpoint for users of the json-placeholder API.

```
curl http://localhost:8081/users
```

15. Confirm the JSON output looks contains several users entries like the one bellow.


```
{
   "id": 1,
   "name": "Leanne Graham",
   "username": "Bret",
   "email": "Sincere@april.biz",
   "address": {
     "street": "Kulas Light",
     "suite": "Apt. 556",
     "city": "Gwenborough",
     "zipcode": "92998-3874",
     "geo": {
       "lat": "-37.3159",
       "lng": "81.1496"
     }
   },
   "phone": "1-770-736-8031 x56442",
   "website": "hildegard.org",
   "company": {
     "name": "Romaguera-Crona",
     "catchPhrase": "Multi-layered client-server neural-net",
     "bs": "harness real-time e-markets"
   }
```


Access the logs within the docker container

16. Find the container ID running the Flex Gateway instance.

```
docker container ls
```

17. Copy the container ID running the Flex Gateway instance and save it in your snippets file.

18. Retrieve the logs for within the Flex Gateway container.

```
docker logs <containerID>
```

Note: Replace <containerID> with the container ID found in the previous step.

19. Confirm the output contains flex gateway envoy and flex gateway agent runtime logs.

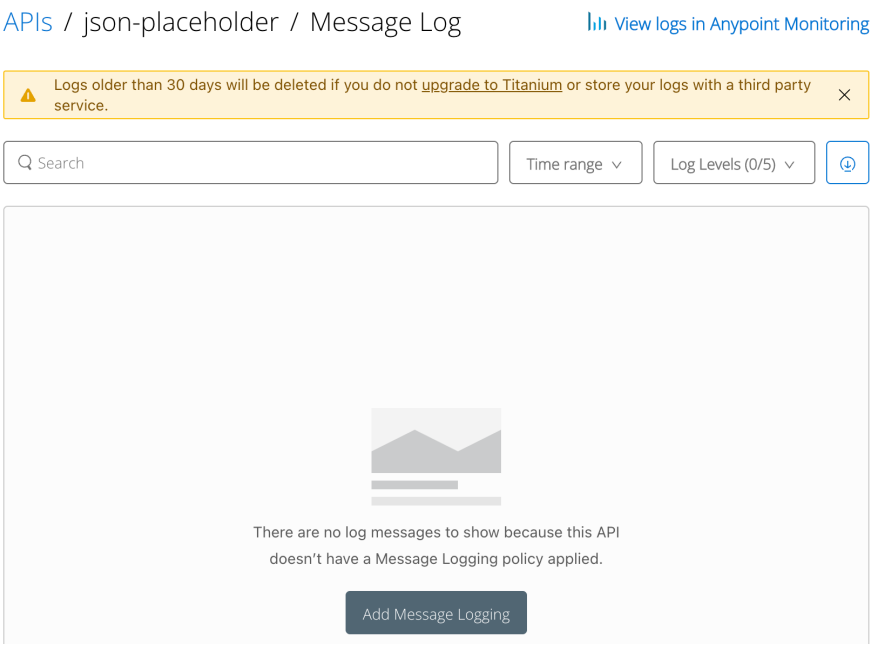
```
…
[flex-gateway-envoy][debug] Thread-Local Wasm created 5 now active
[flex-gateway-envoy][debug] Thread-Local Wasm created 6 now active
[flex-gateway-envoy][info] lds: add/update listener 'json-placeholder-
17888769.ee875caf-de64-44f4-a848-2bf3ce5246eb.svc'
[flex-gateway-agent][info] Creating gateway
[flex-gateway-agent][info] Creating commands
[flex-gateway-agent][info] Processing commands
[flex-gateway-agent][info] Validating gateway
[flex-gateway-agent][info] Generating config
[flex-gateway-agent][info] Gateway default/replica-1: Adding ApiInstance
ee875caf-de64-44f4-a848-2bf3ce5246eb/json-placeholder-17888769
http://0.0.0.0:8081/
…
```

Note: To include the access logs will require additional configuration and to apply a Logging policy.

Check message logs in API Manager

20. From the left menu select Message Log.

21. Notice the message stating that there are no logs to show because there is no Message Logging policy applied.



Note: The logs can also be viewed in Anypoint Monitoring for Titanium subscription users.

Add a custom log forwarding configuration

22. Return to your console and make sure current directory is flex/conf.

```
cd /home/sysadmin/flex/conf
```

Note: The conf folder was mapped to the conf.d folder in docker that will contain the custom configurations files.

23. Create and open in an editor a new file called log-forwarding.yaml.

```
vi log-forwarding.yaml
```

24. Locate, open and inspect a file called log-forwading.yaml in your student folder Module 4.

```
apiVersion: config.mulesoft.com/v1alpha1
kind: Configuration
metadata:
  name: logging
spec:
  logging:
    outputs:
```

```
    - name: default
      type: file
      parameters:
        path: /usr/local/share/mulesoft/flex-gateway/conf.d
        file: logs.txt
        format: plain
    runtimeLogs:
      logLevel: info
      outputs:
      - default
    accessLogs:
      outputs:
      - default
```

25. Copy all the contents from the file log-forwarding.yaml in your local computer to the editor open in your virtual machine.

    Note: Be extra careful to not modify the indentation.

26. Save and close the editor.
27. List the files in the current folder.

```
ls -al
```

28. Confirm there is a new file created called logs.txt.
29. Display the contents of the logs file.

```
tail logs.txt
```

    Note: The runtime logs are now forwarded using fluentbit.

Examine the fluentbit configuration

30. Locate in your snippets files the command to display the fluentbit configuration file contents inside the docker container.

```
docker exec <container-id> cat /tmp/fluent-bit.conf
```

    Note: This file shouldn't be modified directly, instead it should be configured by adding more configuration type yaml file to the conf.d folder.

31. Within the output of the command locate the output configuration just added after the previous steps.

```
[OUTPUT]
    Name file
    Match output.default
    file logs.txt
    format plain
    path /usr/local/share/mulesoft/flex-gateway/conf.d
```

32. Locate the output to forward metering metrics to Anypoint Platform.

```
[OUTPUT]
    Name http
    Match input.metering
    format json
    host metering.ingestion.us-east-1.prod.cloudhub.io
    port 443
    retry_limit false
    storage.total_limit_size 5M
    tls on
    tls.crt_file /tmp/cert.pem
    tls.key_file /tmp/cert.key
    uri /ingestion/api/v1/metering
```

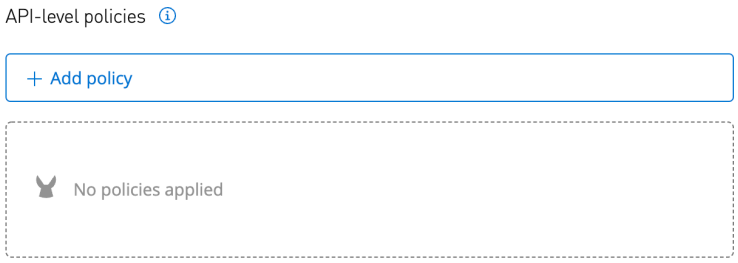
33. Locate the output to forward the logs to Anypoint Monitoring.

```
[OUTPUT]
    Name http
    Match input.monitoring
    format json
    host monitoring.ingestion.us-east-1.prod.cloudhub.io
    port 443
    retry_limit false
    storage.total_limit_size 5M
    tls on
    tls.crt_file /tmp/cert.pem
    tls.key_file /tmp/cert.key
    uri /ingestion/api/v1/monitoring
```

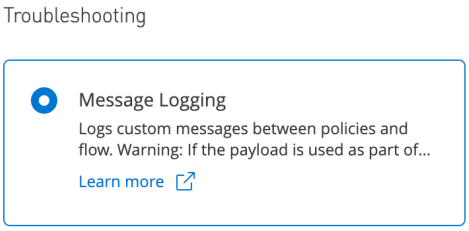
Add a Logging policy to add access logs to the fluentbit output

34. Navigate to API Manager in your browser.

35. Click on the json-placeholder API.
36. On the left menu select Policies.
37. Under API-level policies click Add policy.



38. Select the Message Logging policy under within the Troubleshooting category.



39. Click Next.
40. On the #1 Logging Configuration screen set the following configuration data:

- Configuration Name: Request Logs
- Message:

```
#['API called with ' ++ attributes.method ++ ' method on ' ++
attributes.requestPath]
```

- Before Calling API: Selected
- After Calling API: Unselected

41. Click the +Add button.
42. On the #2 Logging Configuration set the following configuration data:

- Configuration Name: Response Logs
- Message:

```
#['statusCode: ' ++ attributes.statusCode]
```

- Before Calling API: Unselected
- After Calling API: Selected

43. Click Apply and notice the message stating the policy was applied correctly.
44. Call the json-placeholder API once more using curl.

```
curl http://localhost:8081/users
```

45. Check the end of the logs file for the access logs for request and response.

```
tail logs.txt
```

46. Confirm the log entries contain similar output as below:

```
{"kind":"accessLog","category":"17888769-message-logging-flex-1.default.json-placeholder-17888769.ee875caf-de64-44f4-a848-2bf3ce5246eb.svc","message":"API called with GET method on /users","level":"INFO","date":"08/06/2022-12:30:15AM","logger":"flex-gateway-envoy","instanceID":"replica-1.default","request_id":"0e6e6320-ebce-4b3b-a104-2e2f8ce1baf9"}
{"kind":"accessLog","category":"17888769-message-logging-flex-1.default.json-placeholder-17888769.ee875caf-de64-44f4-a848-2bf3ce5246eb.svc","message":"statusCode: 200","level":"INFO","date":"08/06/2022-12:30:15AM","logger":"flex-gateway-envoy","instanceID":"replica-1.default","request_id":"0e6e6320-ebce-4b3b-a104-2e2f8ce1baf9"}
```

Walkthrough 3-3: Scale a Flex Gateway using replicas

In this walkthrough you will add and remove replicas to a Flex Gateway. You will:

- Add a replica to a Flex Gateway instance
- Apply a rate limiting policy to a gateway with multiple replicas
- Check the status of a replica
- Deactivate a gateway replica

Add a replica to a running Flex Gateway instance

1. Return to your terminal connected to the VM.
2. Change the current directory to flex/conf.
3. Locate in your snippets file the command to add a replica.

```
docker run -d --rm \
-v $(pwd):/usr/local/share/mulesoft/flex-gateway/conf.d \
-p 8083:8081 \
-e FLEX_NAME=replica-2 mulesoft/flex-gateway
```

Note: Notice the command is similar to the command needed to start the Flex Gateway, the differences are the FLEX_NAME parameter will contain a new name for the replica and a different port 8083 will be used. It will still route requests to port 8081 where the API is listening.

4. Copy and run the command in your terminal.
5. Confirm that you received the container id as the output of the command.

Check the status of the replicas

6. Check the status of the container started for the replica.

```
docker ps
```

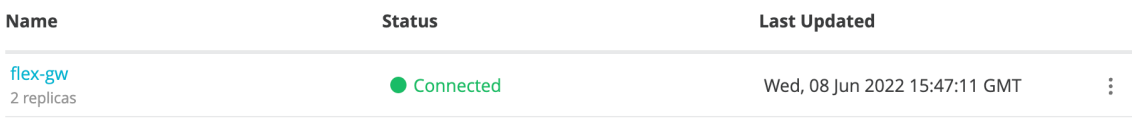
7. Confirm that there are two containers with status up.

```
CONTAINER ID   IMAGE                  COMMAND   CREATED          STATUS          PORTS                                       NAMES
0bc1bff59efe   mulesoft/flex-gateway   "/init"   35 seconds ago   Up 32 seconds   0.0.0.0:8083->8081/tcp, :::8083->8081/tcp   affectionate_rubin
39c69f199b34   mulesoft/flex-gateway   "/init"   6 days ago       Up 6 days       0.0.0.0:8081->8081/tcp, :::8081->8081/tcp   serene_allen
```

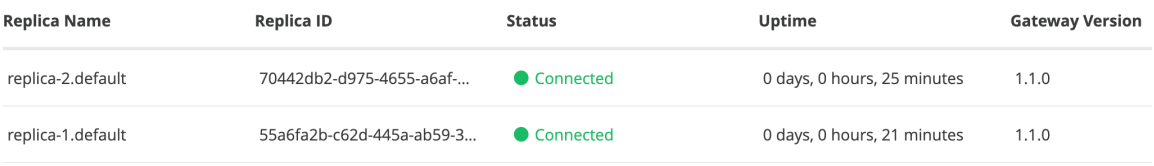
Note: Notice there container names assigned, these are randomly generated by docker.

8. Navigate to Runtime Manager – Flex Gateways.

9. Notice the instance flex-gw now reflects two replicas.

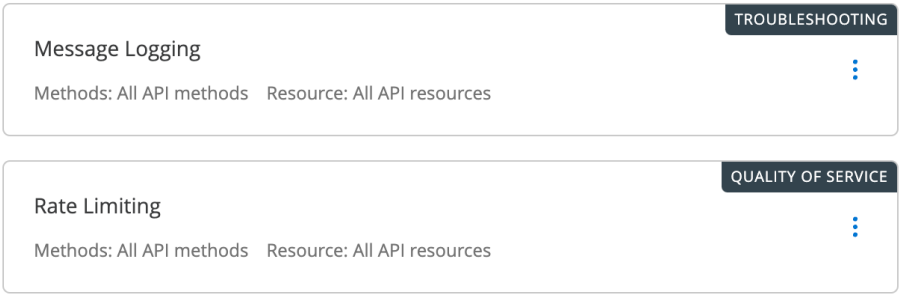


10. Click on the gateway name to open the details.
11. Notice there are two replicas now, replica-1 and replica-2 with status connected.



Apply a rate limiting policy to an API

12. Navigate to API Manager -> API Administration.
13. Select the json-placeholder API.
14. Select Policies from the left menu.
15. Click +Add Policy.
16. Select Rate Limiting under the Quality of Service section.
17. Click Next.
18. Configure the Rate Limiting policy with the following limits:
    - Number of Requests: 10
    - Time Period: 5
    - Time unit: Minute
19. Select Expose Headers.
20. Click Apply.
21. Verify that you see the Message Logging and Rate Limiting API-level policies applied.



Test the rate limiting policy applied to the API

22. Perform one more request using curl to see the headers exposed after applying the policy.

```
curl -v http://localhost:8081/users
```

23. Scroll up and locate the x-ratelimit values following x-envoy-upstream-service-time.

```
x-ratelimit-remaining: 9
x-ratelimit-limit: 10
x-ratelimit-reset: 106278
```

Note: This limit is for replica-1 exposed in port 8081.

24. Perform another request this time to replica-2 exposed in port 8082.

```
curl -v http://localhost:8083/users
```

25. Confirm the rate limit remaining is still 9 since the limits are applied per replica.

```
x-ratelimit-remaining: 9
x-ratelimit-limit: 10
x-ratelimit-reset: 110956
```

26. Call the users endpoints 11 times.

```
for i in `seq 1 11`; do curl -i http://localhost:8081/users; done
```

27. Locate the error returned by the Flex Gateway once the rate limit is reached.

```
HTTP/1.1 429 Too Many Requests
content-type: application/json; charset=UTF-8
content-length: 29
x-ratelimit-remaining: 0
x-ratelimit-limit: 10
x-ratelimit-reset: 17301
date: Thu, 23 Jun 2022 16:21:02 GMT
server: Anypoint Flex Gateway

{"error":"Too Many Requests"}
```

Deactivate a gateway replica

28. List the running docker containers.

```
docker container ls
```

29. Copy the containerID running in port 8083.

    Note: The container running in port 8083 belongs to the replica named replica-2 which was the second one started.

30. Run the command to stop the docker container with the containerID found in the previous step.

```
docker container stop <containerID>
```

31. List the containers one more time and verify that the stopped container is no longer listed.

```
docker container ls
```

32. Navigate to Runtime Manager - Flex Gateways.
33. Click on the name your Flex Gateway instance e.g. flex-gw.
34. Confirm that replica-2 is in disconnected status.

Replica Name	Replica ID	Status
replica-1.default	7862a28c-b93b-4e28-ad23-e6cae...	Connected
replica-2.default	0f2cb9aa-c076-48d0-8879-edd6d...	Disconnected

Note: The replica cannot be deleted from runtime manager will be removed on its own after 30 days.

Walkthrough 3-4: Configure TLS/SSL in Flex Gateway

In this walkthrough you will configure TLS to enable HTTPS traffic for all the APIs running in Flex Gateway. You will:

- Create a certificate and key pair
- Add a TLS policy configuration
- Call an API via HTTPS and check the logs
- Inspect Flex Gateway configuration files

Create a certificate/key pair (optional)

1. Open a terminal and SSH into your VM.
2. Generate a new certificate and a private key.

```
openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 3650
```

3. Enter the following values when prompted:
   - PEM pass phrase: mulesoft
   - Verifying – Pem pass phrase: mulesoft
   - Country Name: US
   - State: CA
   - Locality Name: San Francisco
   - Organization Name: Mulesoft
   - Organizational Unit Name: Training
   - Common Name: Max Mule
   - Email Address: max.mule@mulesoft.com
4. List the files in your current directory and note the newly created cert.pem and key.pem files.
5. Decrypt the contents of the encrypted key.

```
openssl pkcs8 -in key.pem -out decrypted-key.pem
```

   Note: The private key content cannot be encrypted before being added to the TLS configuration in Flex Gateway.

6. When prompted enter mulesoft as the value.
7. Note the file created decrypted-key.pem, this will contain the decrypted key contents that will be used in later steps.

Create and apply TLS yaml configuration file and check the logs

8. Create a second ssh connection to your VM in another terminal.
9. Change to directory flex/conf.
10. Tail the logs to watch the latest entries as they get appended.

```
tail -f logs.txt
```

11. In your local, locate inside the student files folders the file tls-config.yaml.
12. Open the file in an editor and examine its content.

```
apiVersion: gateway.mulesoft.com/v1alpha1
kind: PolicyBinding
metadata:
  name: http-tls
spec:
  targetRef:
    kind: Selector
    selector:
      kind: ApiInstance
  policyRef:
    name: tls
  config:
    certificate:
      key: |
        -----BEGIN PRIVATE KEY-----
        MIIEv…
        -----END PRIVATE KEY-----
      crt: |
        -----BEGIN CERTIFICATE-----
        MIIEBTCC…
        -----END CERTIFICATE-----
```

Note: The configuration kind is PolicyBinding and policyRef.name is tls. This combination can be applied even in connected mode.

13. Switch to the Terminal with your VM.
14. Navigate to the folder flex/conf.
15. Create a new configuration file called tls-config.yaml.

```
vi tls-config.yaml
```

16. Copy all the contents from your local tls-config.yaml to the file with the same name just created in your VM.
17. Save the file and exit the editor in your VM.

```
:wq
```

18. Switch to the Terminal with the logs and note how the configuration is applied as soon as the file is saved.

Call the API using the HTTPS protocol and check the logs

19. Call the API using https.

```
curl -k https://localhost:8081/users
```

Note: We are passing the insecure (-k) flag for simplicity to skip the certificate validation. Further steps are required to pass the certificate in the request.

20. Verify that the same results of 10 users are retrieved.
21. Call the API again using http.

```
curl http://localhost:8081/users
```

22. Verify that the application doesn't return any values as now it is listening for traffic only on HTTPS.

```
curl: (52) Empty reply from servers
```

Perform a dump of Flex configurations applied

23. Run the flexctl dump command from inside the docker container.

```
docker exec -t -i <container-id> flexctl dump
```

24. Notice the output says the writing the output(json files) to current directory, however there will be no new files in your current directory. The files will be inside the container.
25. List the json files in the current directory inside the docker container.

```
docker exec -t -i <container-id> ls -l | grep json
```

26. Notice the files created because of the flexctl dump command.

```
-rw-------   1 root root    13544 Jun 28 16:16 api-instances.json
-rw-------   1 root root     2139 Jun 28 16:16 configuration.json
```

```
-rw-------   1 root root 20399953 Jun 28 16:16 extensions.json
-rw-------   1 root root      870 Jun 28 16:16 services.json
```

27. Display the contents of api-instances.json.

```
docker exec -t -i <containerID> cat api-instances.json
```

28. Locate the API instance that belongs to the JSON Placeholder API. It has the following label.

```
flex.mulesoft.com/apiinstance-id": "17923159
```

Note: The API ID is unique to every instance created.

29. Locate the policies applied including TLS and logging.
30. Display the contents of configurations.json.

```
docker exec -t -i <containerID> cat configuration.json
```

31. Locate the outputs configuration, which include the default created by the logging configuration.

Module 4: Managing APIs using Anypoint Mule Gateway

At the end of this module, you should be able to:

- Distinguish between API implementations and API specifications.
- Describe the two types of API proxy configurations.
- Manage APIs using basic endpoint configurations.
- Manage APIs using proxy endpoint configurations.
- Deploy API implementations to CloudHub.
- Apply automated policies to all API instances.

Walkthrough 4-1: Connect an API to a Mule application via autodiscovery

In this walkthrough, in API Manager you create an API instance using a basic endpoint. Next, in Runtime Manager, you configure the deployment of a Mule application with API Autodiscovery. You will:

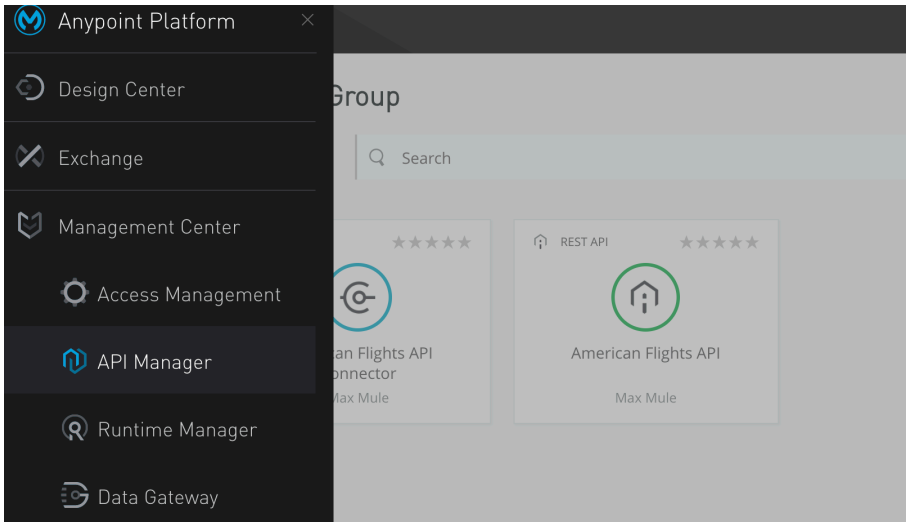
- Manage an API by importing it from Anypoint Exchange into API Manager.
- Define a basic endpoint API instance in API Manager to apply policies to a Mule application.
- Configure a Mule application's API Autodiscovery parameter with the value from API Manager.
- Deploy a Mule application as an API proxy using Autodiscovery.

Log in to API Manager and select the Sandbox environment

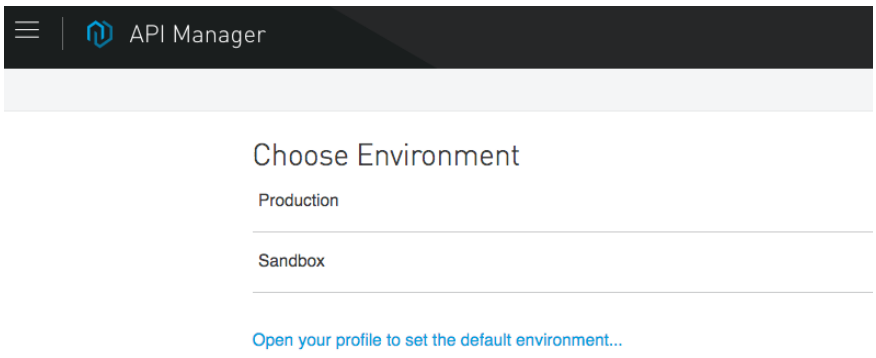
1. Return to the Anypoint exchange browser window.

   Note: If you are logged out, navigate to http://anypoint.mulesoft.com and log in.

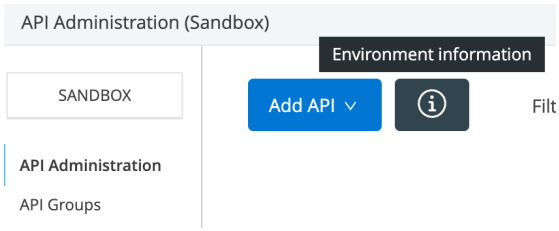
2. In the main menu, select API Manager.



3. If you are prompted to select an environment, select the Sandbox environment.



4. In the upper-left, look at the environment selected; if it is not Sandbox, click on the current environment name and select SANDBOX from the Switch Environment dialog box.

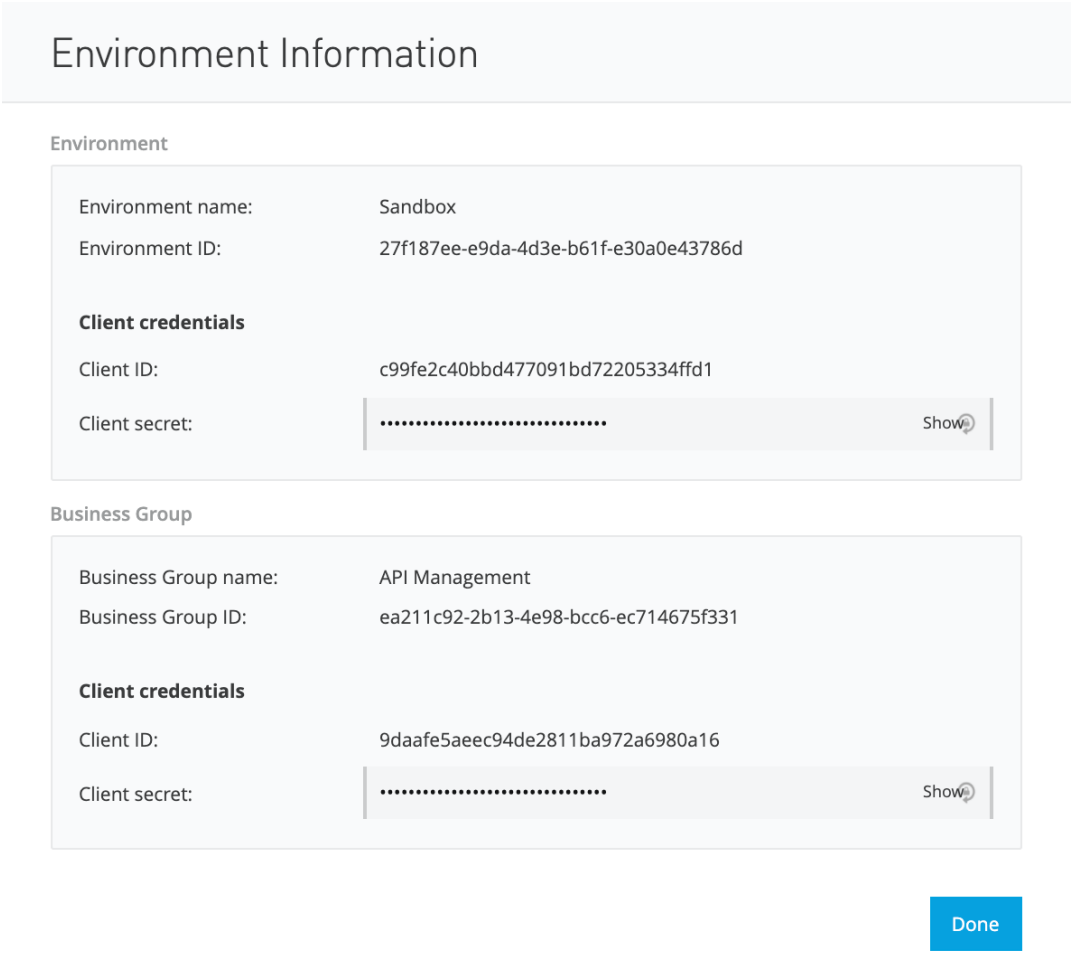


Copy the environment client credentials

5. Click the information icon to show the Environment Information.
6. Return to the course snippets file.

   Note: If the course snippets is not still open, open a file browser and navigate to the student files directory and open the course snippets file.

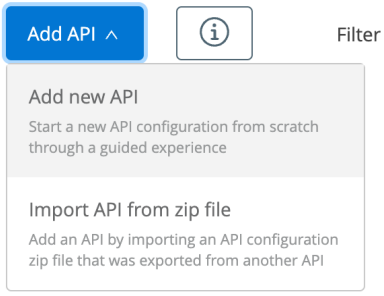
7. In the Environment Information dialog box, copy the client id and paste it into the course snippets file under Module 4, replacing the placeholder
<< INSERT ENVIRONMENT CLIENT ID>>.



8. Return to API Manager.
9. In the Client secret text field, click Show to display the client secret.
10. Copy and paste the client secret value into the course snippets file in Module 3, replacing the placeholder << INSERT ENVIRONMENT CLIENT SECRET>>.
11. Return to the Environment Information dialog box and click Done.

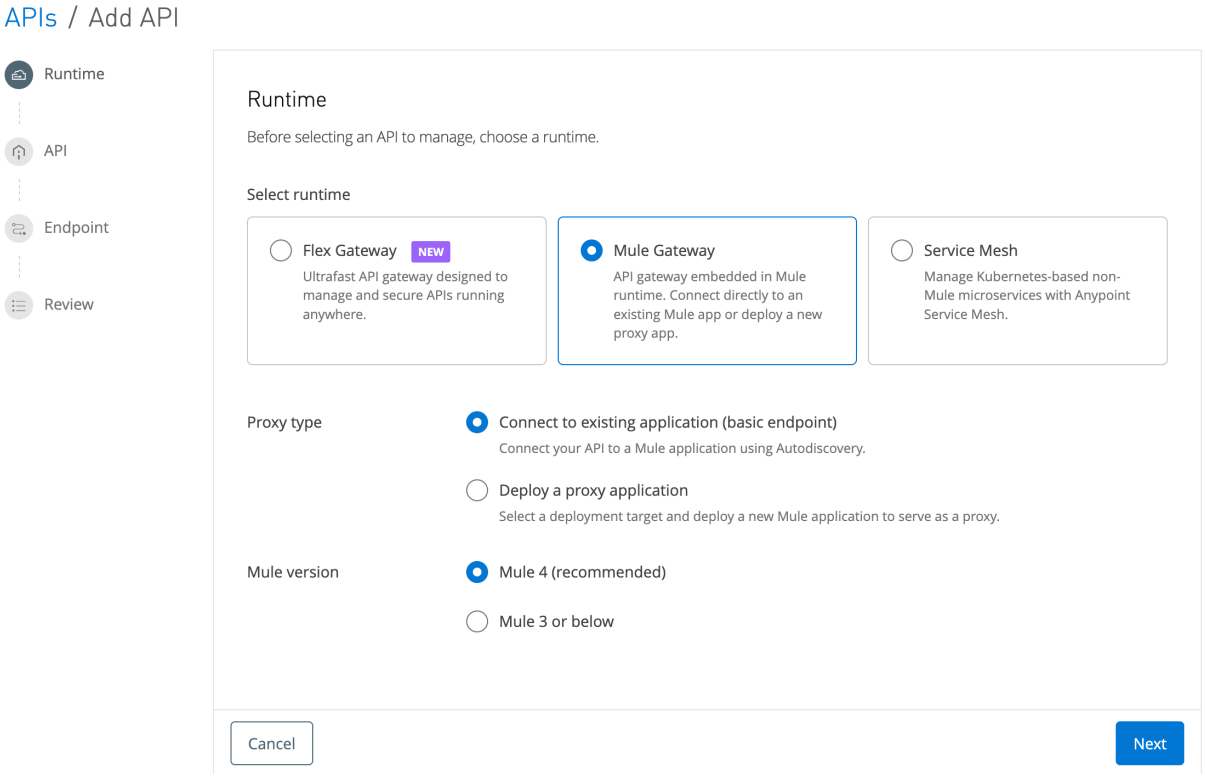
Import an API asset to API Manager

12. In the Add API drop-down list, select Add new API.



13. In the Runtime selection screen select:

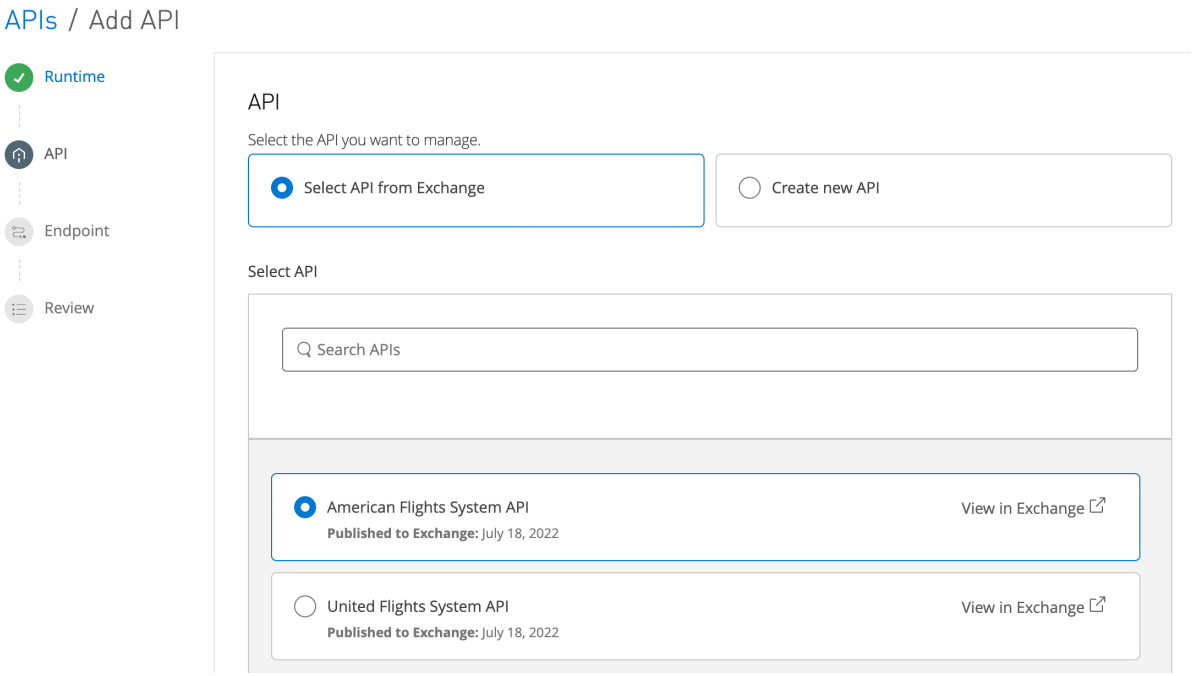
- **Runtime**: Mule Gateway
- **Proxy type**: Connect to existing application (basic endpoint)
- **Mule version**: Mule 4



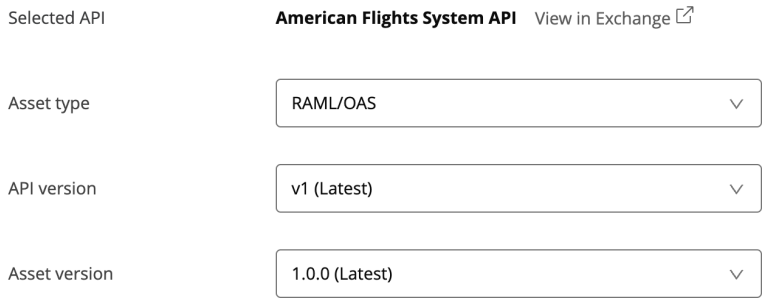
14. Click Next.

15. In the API selection screen select:

- **API you want to manage**: Select API from Exchange
- **Select API**: American Flights Systems API
- Leave the rest of the values unchanged.



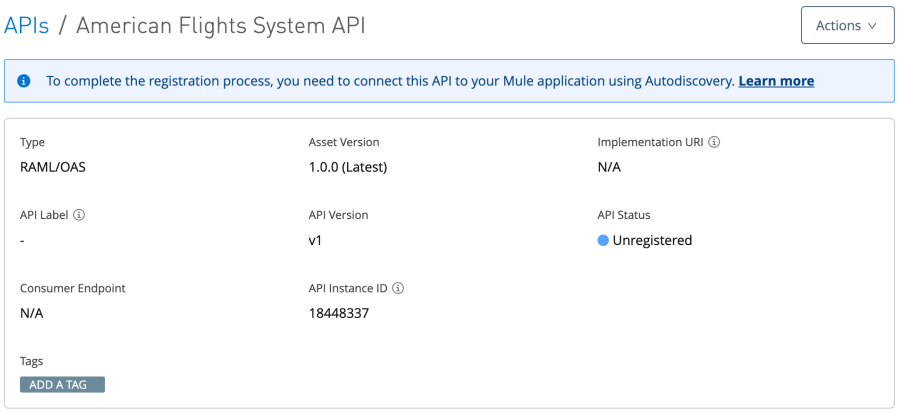
16. Scroll down and verify that the following values are preselected.



17. Click Next.
18. For the Endpoint configuration screen leave all settings blank as they are optional.
19. Click Next.
20. In the Review screen, verify your selections and click Save.

Save the API instance and view the Autodiscovery ID

21. Click Save; the American Flights System API v1 instance Settings tab should appear.
22. Under Autodiscovery, copy the API Instance ID from the API settings page.



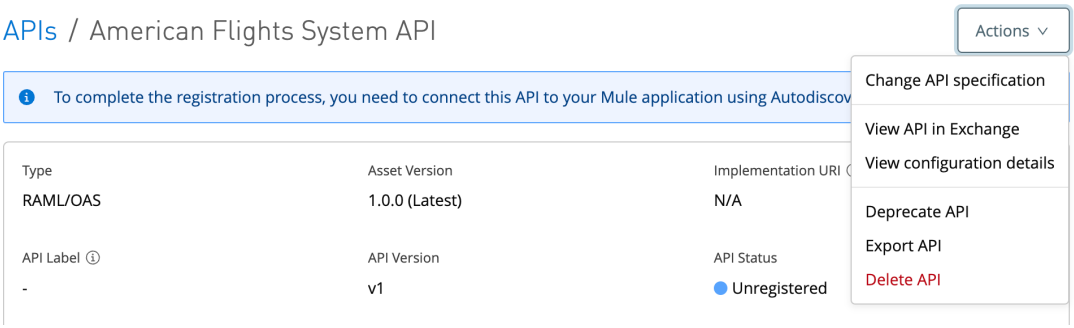
23. View the API Status; it should say Unregistered.

    Note: The API Status will change to Activated when a Mule application connects to API Manager with this Autodiscovery ID.

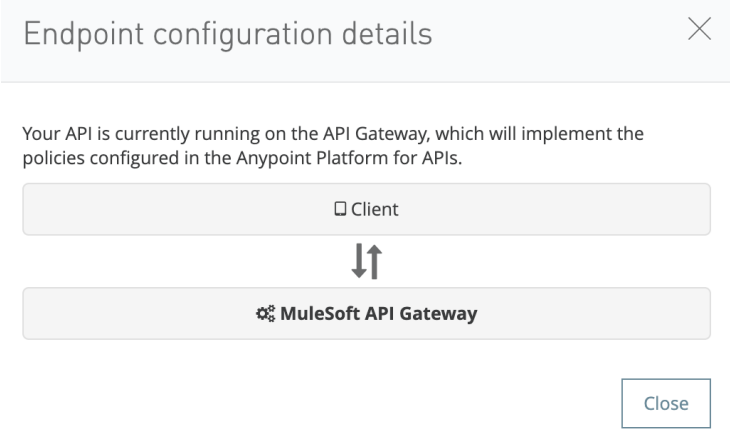
24. Return to the course snippets file and in the Autodiscovery Properties section paste the API Instance ID value to replace the <<INSERT API ID>> value of the api.id.

View the configuration details for this type of API instance

25. In the upper-right, click the View configuration details link.



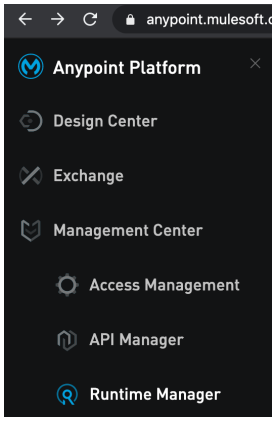
26. View the Endpoint configuration details dialog box; you should see a basic endpoint configuration showing API clients connect directly to the API implementation that is running inside the MuleSoft API Gateway.



27. Click Dismiss.

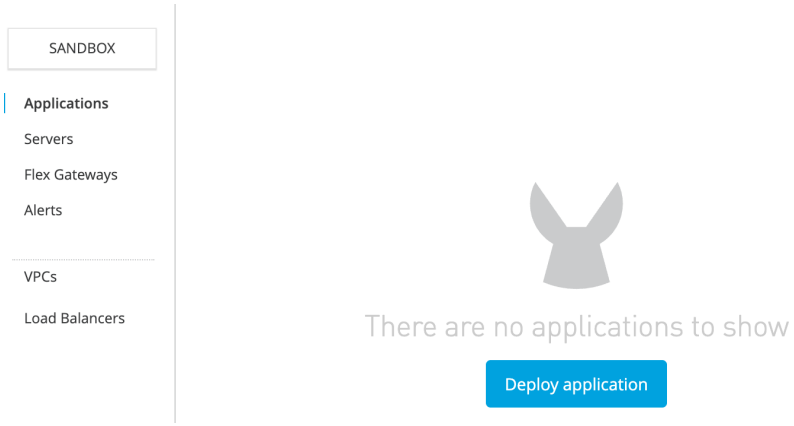
Deploy a Mule application to CloudHub and configure the API Autodiscovery ID

28. From the main menu, right-click on Runtime Manager and select Open link in new tab.



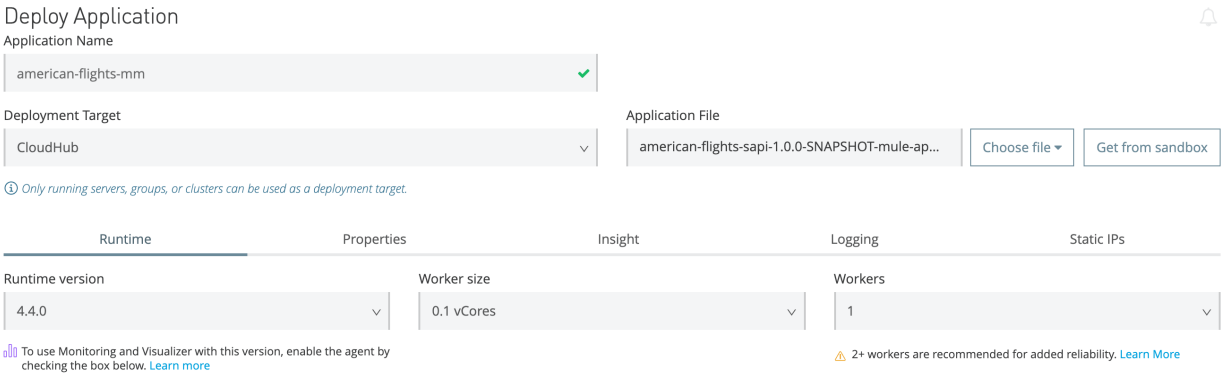
29. If you are prompted to select an environment, select the Sandbox environment.

30. Click Deploy application; a Deploy Application page should appear.



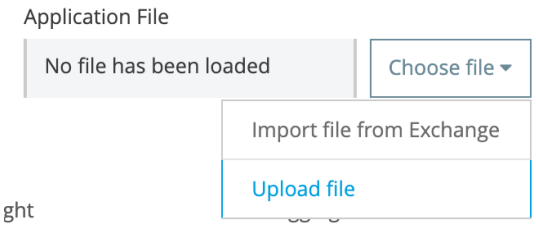
31. Configure the following deployment information:

- **Application Name**: american-flights-{initials}
- **Deployment Target**: CloudHub
- **Runtime version**: Latest (e.g. 4.4.0)
- **Worker size**: 0.1 vCores
- **Workers**: 1

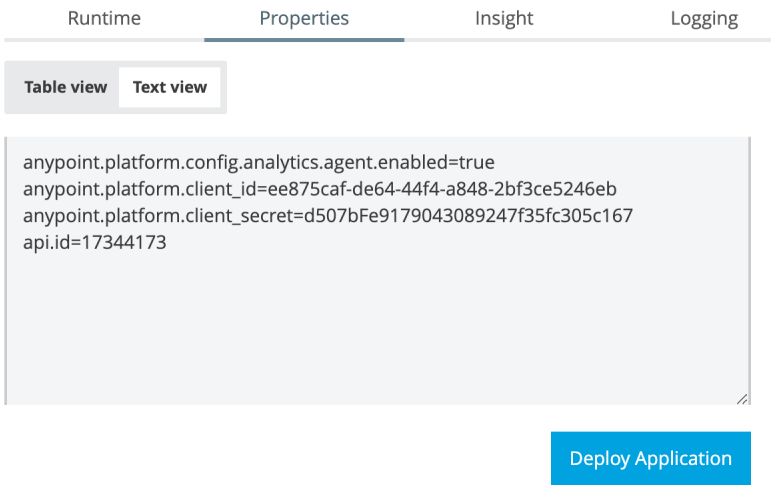


Note: If american-flights-{initials} already exists, use your full name. For example, american-flights-maxmule for Max Mule.

32. Click Choose file.
33. Select Upload file form the drop-down list.



34. Browse your studentFiles folder for american-flights-sapi-1.0.0-SNAPSHOT-mule-application.jar.
35. Select the Properties tab and change to Text view.
36. Copy the environment credentials and app id properties under the Autodiscovery Properties section in the snippets file.

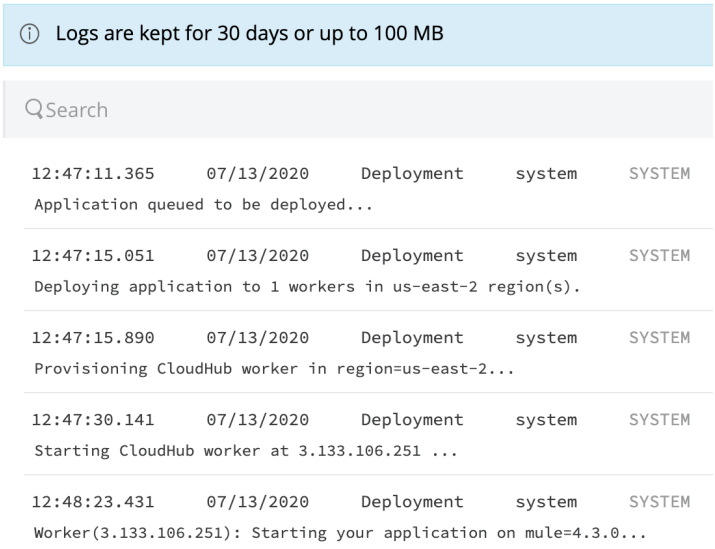


Note: Make sure to not remove the analytics agent enabled variable and value.

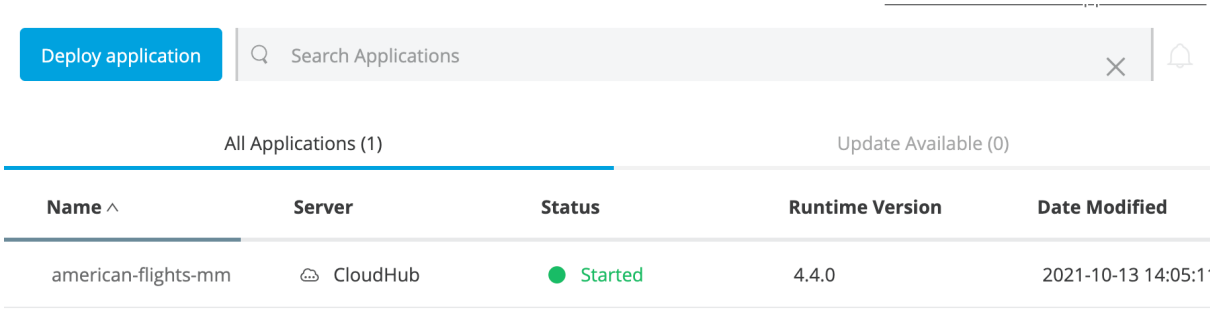
37. Click Deploy Application; the Live Console with the logs should appear.

View the application logs to verify the Mule application is deploying to a CloudHub worker

38. Verify you see log messages that the application is deploying to 1 worker.



39. Select the Applications from the left menu.
40. Wait for the Mule application to deploy and start.



Note: It can take over 10 minutes for the Mule application to finish deploying. You can move ahead and define policies in API Manager while you wait for the Mule application to finish deploying.

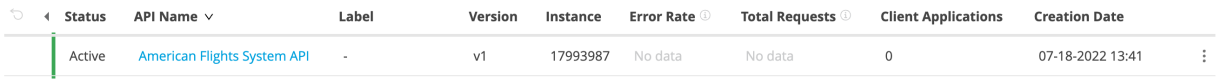
Walkthrough 4-2: Apply policies to a single API instance or to all API instances in an environment

In this walkthrough, you apply a Message Logging and Rate limiting policy to every API instance managed by an environment in API Manager. You will:

- Define a JSON threat protection policy to be applied to just one particular API instance.
- Define an automated message logging policy to be applied to every API instance in an API Manager environment.
- Test how automated policies are applied to every API instance of any API Manager environment.

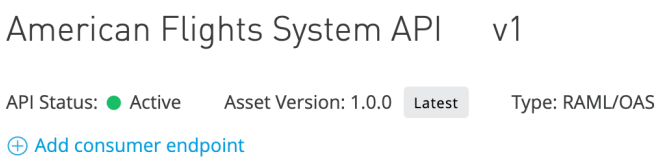
Define a JSON threat protection policy for an API instance

1. Return to API Manager.
2. From the list of APIs click on the API name field of American Flights System API to navigate to the API settings page.



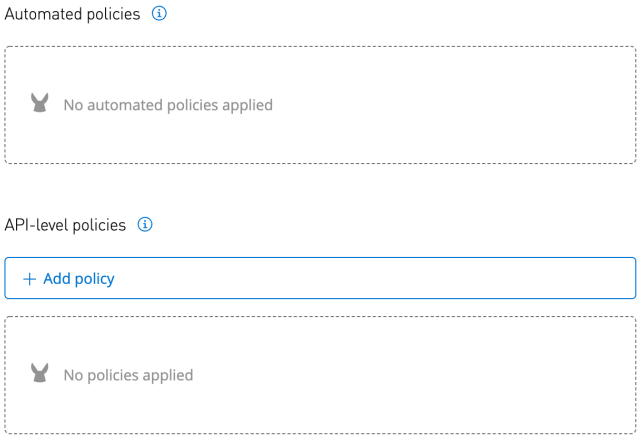
   Note: The error rate and total requests fields show no data because we haven't made any request to the newly deployed API.

3. Look at the API Status; it will say Unregistered until the Mule application deployment registers with API Manager, and then the status will change to Active.

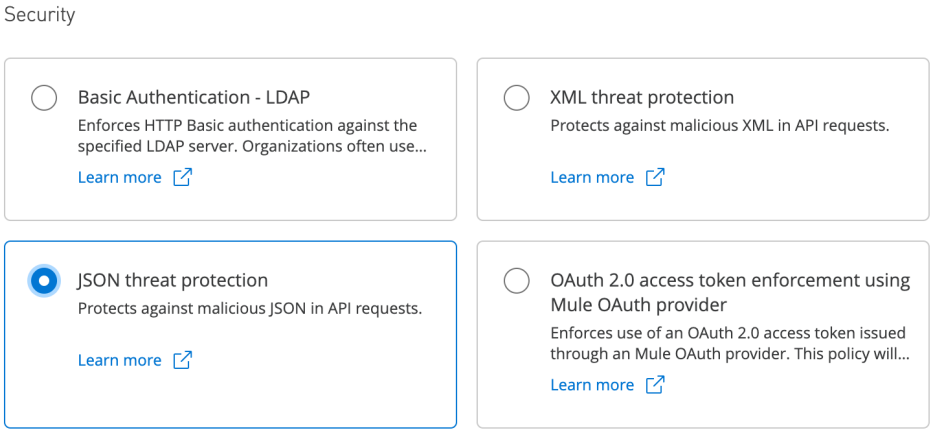


4. In the left-side navigation, select the Policies tab.

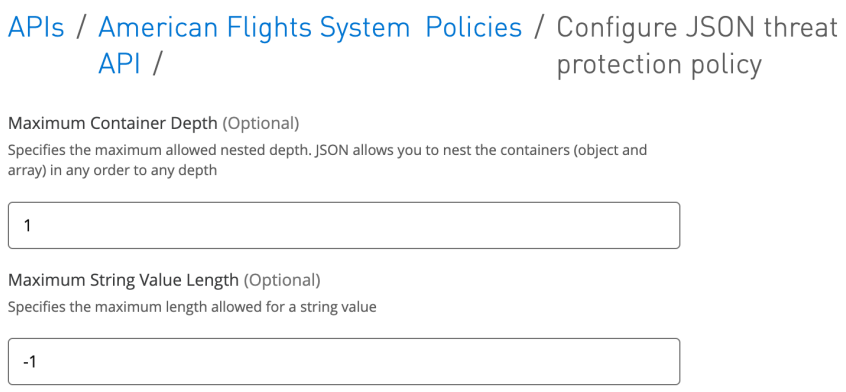
5. Under API-level policies, click +Add policy.



6. In the Add a policy screen, scroll down to see all policies.
7. Under Security, select the JSON threat protection policy and click Next.



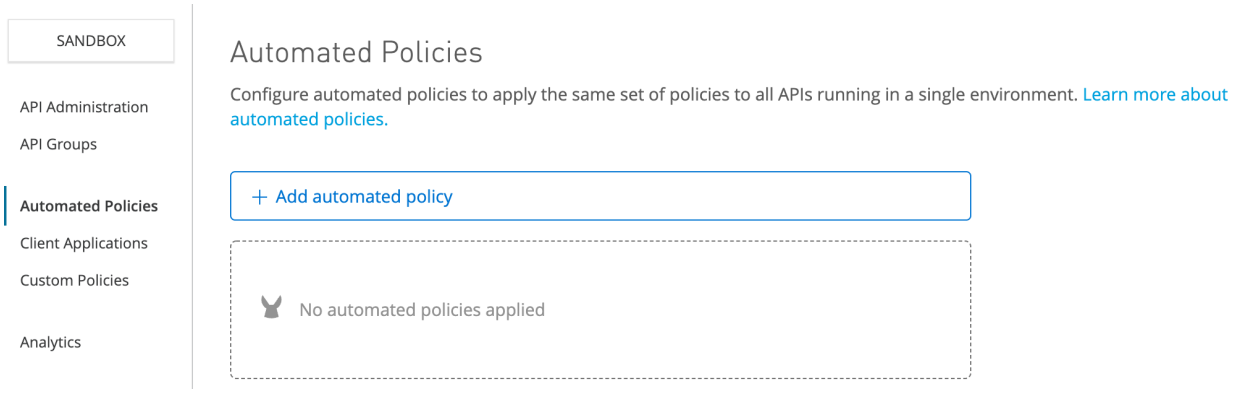
8. In the Configure policy screen, change the first Maximum Container Depth value to 1.



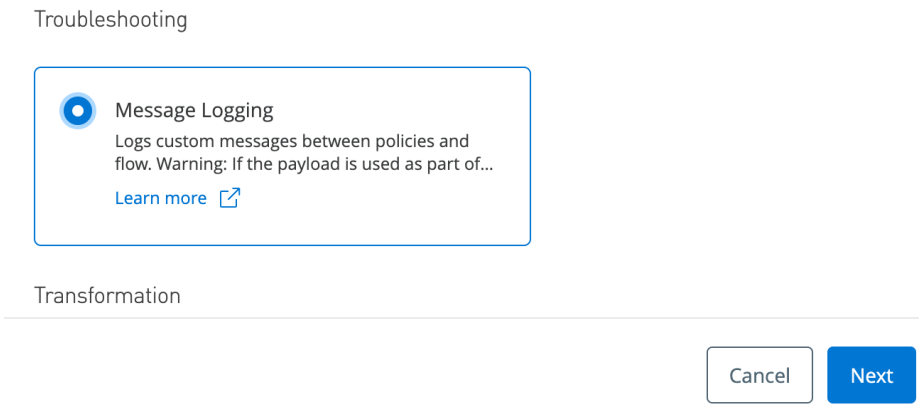
9. Click Apply.

Define an automated Message Logging policy to apply to every API instance in the Sandbox environment

10. In the upper-left, click API Administration.
11. In the left-side navigation, select **Automated Policies**.
12. Click +Add automated policy.



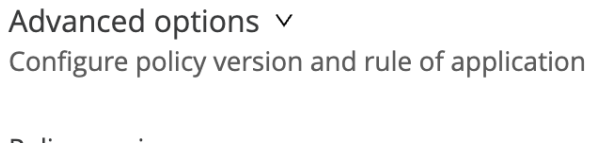
13. Under troubleshooting, select Message Logging and click Next.



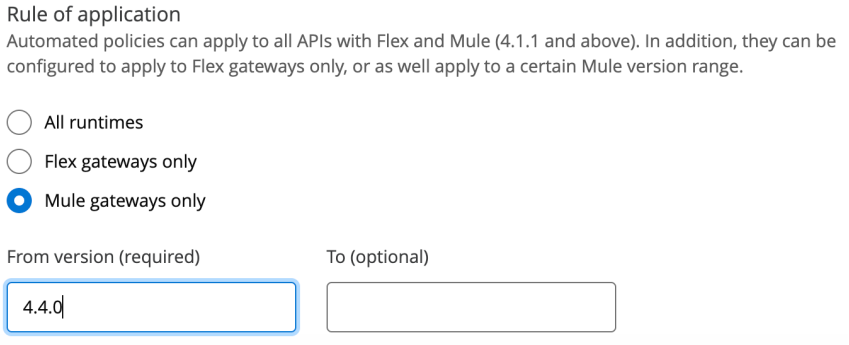
14. In the Configure Message Logging screen, enter the following values:

- **Configuration Name**: <Leave default>
- **Message**: #[attributes]
- **Conditional**: <leave blank>
- **Category**: <leave blank>
- **Level**: INFO
- **Before Calling API:** Checked
- **After Calling API**: Checked

15. Scroll down to Advanced Options and expand it.



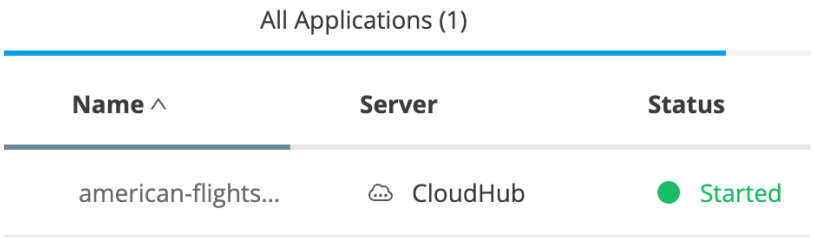
16. Select Mule gateways only under Rule of application. Enter the From version as 4.4.0.



17. Click Apply.

Wait for the Mule application to start up

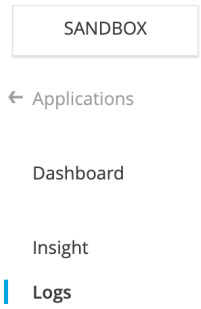
18. Return to Runtime Manager.
19. Verify the status is Started.



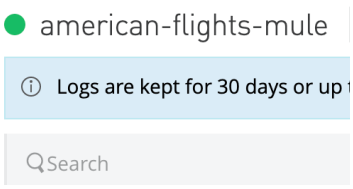
Note: If the status is Stopped, there is an error in your deployment. Review the log files to find the issue, such as an incorrect value for one of the properties in the Properties tab, or uploading the wrong deployable archive JAR file.

Search for log messages that show the Mule application started correctly

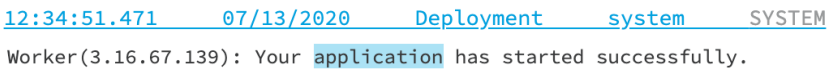
20. Click on the API name, american-flights-<XX>.
21. In the left-side navigation, select Logs.



22. In the Search bar, type application and press the Return key to enter the search.

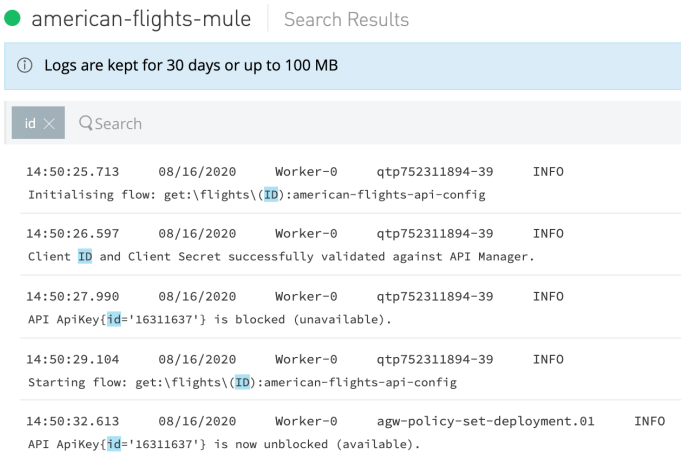


23. View matching log messages and verify there is a message that the application has started successfully.

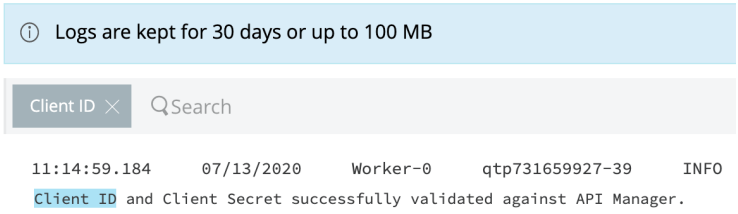


Search for log messages related to API Manager and policy enforcement

24. In the Search bar, type id.



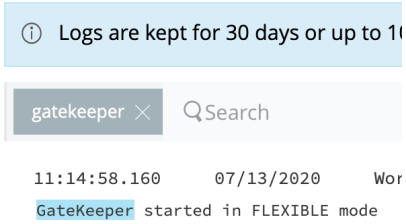
25. In the Search bar, type client id and press the Return key.
26. Verify see a log message that the Client ID and Client Secret were successfully validated against API Manager; this shows you that the basic endpoint successfully connected to API Manager.



27. In the Search bar, type apikey and press the Return key.
28. Verify you see a log message that the API ApiKey with the id value you copied from API Manager reports it is now unblocked (available).

Note: this shows you that API ID you copied into this Mule application's properties file is the correct API instance ID to configure autodiscovery with API Manager to have policies applied.

29. In the Search bar, search for gatekeeper.
30. Verify you see a log message that the GateKeeper started in FLEXIBLE mode.



31. In the Search bar, search for policy.
32. Verify a proxy-policy bean started.

```
Starting Bean: proxy-policy-57176ed0-c1e0-11ea-8b1d-0a54c5af029a
```

33. In the Search bar, search for analytics.
34. Verify there is log message showing an analytics-policy was applied to this API instance; this shows you that policies are being applied from API Manager to this Mule application.

```
**********************************************************************
* Policy: analytics-policy-16247831-american-flights-api-main        *
* OS encoding: UTF-8, Mule encoding: UTF-8                           *
*                                                                    *
**********************************************************************
```

Search for log messages related to the Mule application and its implementation

35. In the Search bar, search for apikit.
36. Verify you see log messages from the APIKit Router in the Mule application; this is the Mule component in the Mule application that receives and processes HTTP requests to the API implementation endpoint.

```
14:50:25.118     08/16/2020     Worker-0     qtp752311894-39     INFO

**************************************************************************
* APIKit Router 'american-flights-api-config' started using Parser: AMF  *
**************************************************************************
```

```
14:50:25.138     08/16/2020     Worker-0     qtp752311894-39     INFO

**************************************************************************
* APIKit Console URL: http://american-flights-mule.us-e2.cloudhub.io/console/  *
**************************************************************************
```

37. In the Search bar, search for flow.
38. Verify you see the american-flights-api-main flow; this is the part of the Mule application that uses APIkit to receive requests on the API endpoints.
39. Verify you see a flow named api-ready and api-alive; these are flows injected into the Mule application to allow API Manager to monitor the status of API endpoints.

View log messages for the two API policies

40. In the Search bar, search for applied.
41. Verify you see log messages that the JSON threat protection and message logging policies were applied to the Mule application.

```
15:32:31.404     08/16/2020     Worker-0     agw-policy-set-deployment.01     INFO
Applied policy json-threat-protection-1184621 version 1.1.4 to API American Flights API-v1-
v1:16311637 (16311637) in application american-flights-mule
```

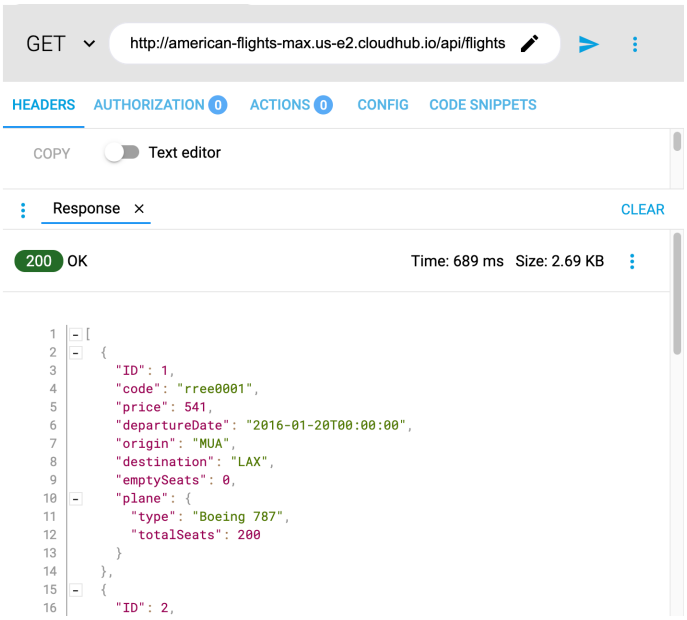
```
15:32:34.101     08/16/2020     Worker-0     agw-policy-set-deployment.01     INFO
Applied policy message-logging-11724-16311637 version 1.0.0 to API American Flights API-v1-
v1:16311637 (16311637) in application american-flights-mule
```

Confirm the American Flights System API instance deployment is working correctly by testing one of its endpoints

42. In the left-side navigation, click Settings.
43. Copy the App url link address.
44. Paste in Advanced Rest Client in the Request URL.

    Note: Select method as GET if not selected already.

45. Append /api/flights to the URL and click Send.
46. Verify a JSON array of flights is returned from the Mule application.

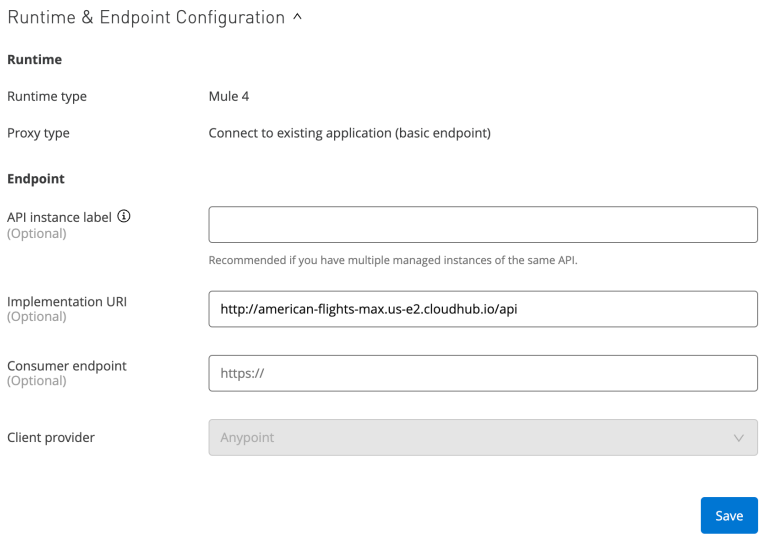


Note: This is the public URL to access the Mule application. Notice the response is an array of JSON objects, where each flight object is nested two-levels deep: the plane key has a second-level child object as its value. Later in this walkthrough you will use this structure to test the JSON threat detection policy.

Configure the API instance in API Manager with the deployed Mule application's public URL

47. Return to API Manager.
48. Navigate to the American Flights System API v1 settings page.
49. Scroll to the bottom and expand Runtime & Endpoint Configuration.
50. Copy the App URL into the Implementation URI text field.

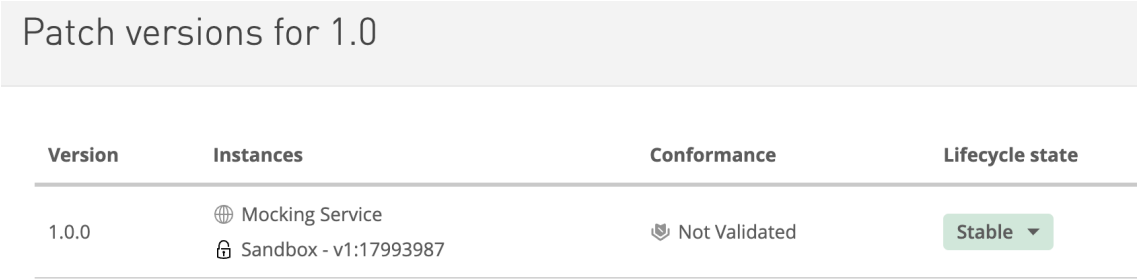
51. Append /api to the App URI then click Save.



52. Copy and paste the value of Implementation URI to the Consumer endpoint.

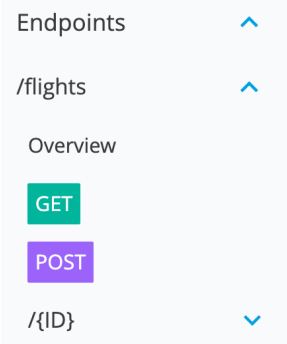
Test that the JSON threat protection policy is applied correctly

53. In the right side, click View API in Exchange; this should open a new browser tab and open the American Flights System API portal in Anypoint Exchange.
54. Click Manage versions.
55. Verify that it shows the Mocking Service and Sanbox instances.

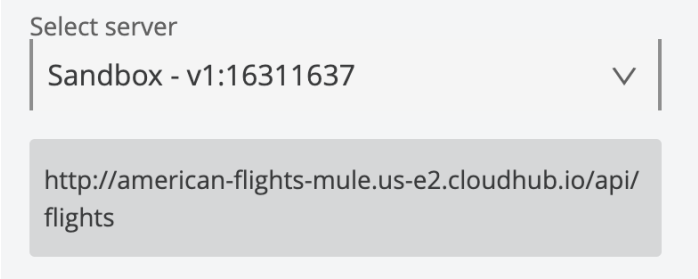


56. In the left-side Endpoints list click /flights.

57. Click the POST method.



58. In the Select server drop-down list, change the server to Sandbox - v1.
59. Verify the correct App URL appears for the deployed Mule application.

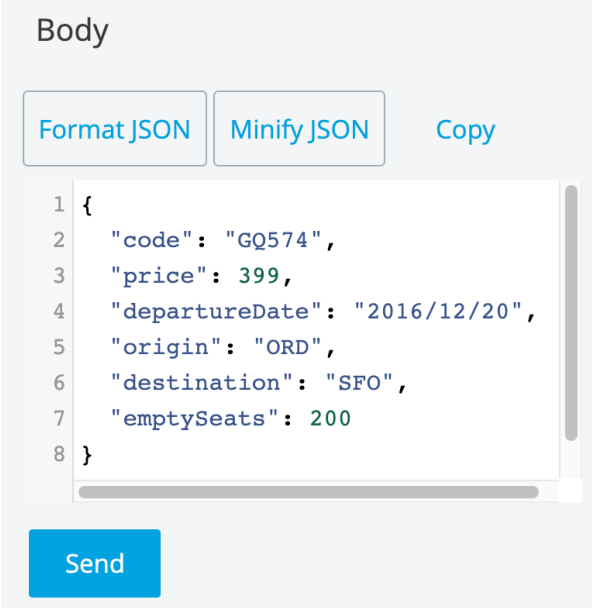


60. Click Send.
61. Verify a 400 Bad Request error status is returned.

```
{
  "error": "Container depth has been exceeded.
Maximum allowed is: 1"
}
```

Note: The error message indicates the max allowed JSON container depth of 1 was exceeded.

62. Replace the body text with the one level JSON payload from the snippets file.



63. Click Send.
64. Verify the request returns a success 201 Created status code.

    Note: This shows you the JSON threat protection policy is working correctly. It blocked a request with a depth of 2, but allowed a request with a depth of 1.

Walkthrough 4-3: Deploy a Mule application as a proxy

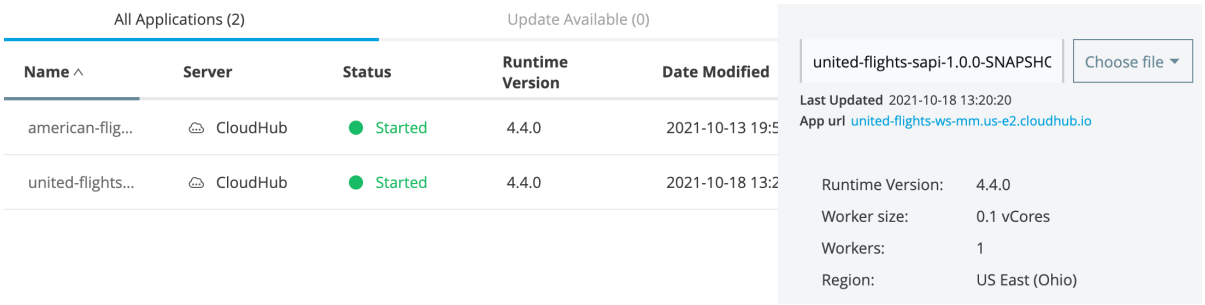
In this walkthrough, in API Manager, you configure an API instance as an API Proxy. Then the automatically created API proxy Mule application is deployed it to a CloudHub worker. You will:

- Manage an API by importing it from Anypoint Exchange into API Manager.
- Configure a proxy API instance in API Manager to enforce policies and forward requests to a separate API implementation endpoint.
- Test that an API proxy properly forwards requests to the API implementation.

Deploy a Mule application that implements the United Airlines API

1. Return to Runtime Manager.
2. Click Deploy Application.
3. Configure the following deployment information:
   - **Application Name**: united-flights-ws-{initials}
   - **Deployment Target**: CloudHub
   - **Runtime version**: Latest (e.g. 4.4.0)
   - **Worker size**: 0.1 vCores
   - **Workers**: 1

   Note: If united-flights-ws-{initials} is already taken then use your name instead.
4. For Application file choose the united-flights-sapi-1.0.0-SNAPSHOT-mule-application.jar located under your student files folder.
5. Click Deploy Application.
6. Click Applications from the left menu.
7. Make sure the united-flights-ws-{initials} application is in status started.
8. Select the united-flights-ws-{initials} row.
9. Copy the App url from the right hand side menu and paste it in your snippets file.



Configure the API proxy endpoint for the United Flights System API

10. Return to the API Administration page in API Manager.
11. Click Add API and from the drop-down list, select Add new API.
12. In the Runtime selection screen select:

    - **Runtime**: Mule Gateway
    - **Proxy type**: Deploy a proxy application
    - **Target type**: Cloudhub
    - **Runtime version**: 4.4.0 (or latest)
    - **Proxy app name**: united-flights-api-{initials}

    Note: If united-flights-api-{initials} already exists, use your fullname. For example, If united-flights-api-maxmule.

13. Click Next.
41. In the API selection screen select:

    - **API you want to manage**: Select API from Exchange
    - **Select API**: United Flights Systems API
    - Leave the rest of the values unchanged.

14. Click Next.
15. For the Endpoint configuration set the implementation URI to the United Flights System API application URI saved previously in the snippets file.

    ```
    http://united-flights-ws-{initials}.{region}.cloudhub.io/api
    ```

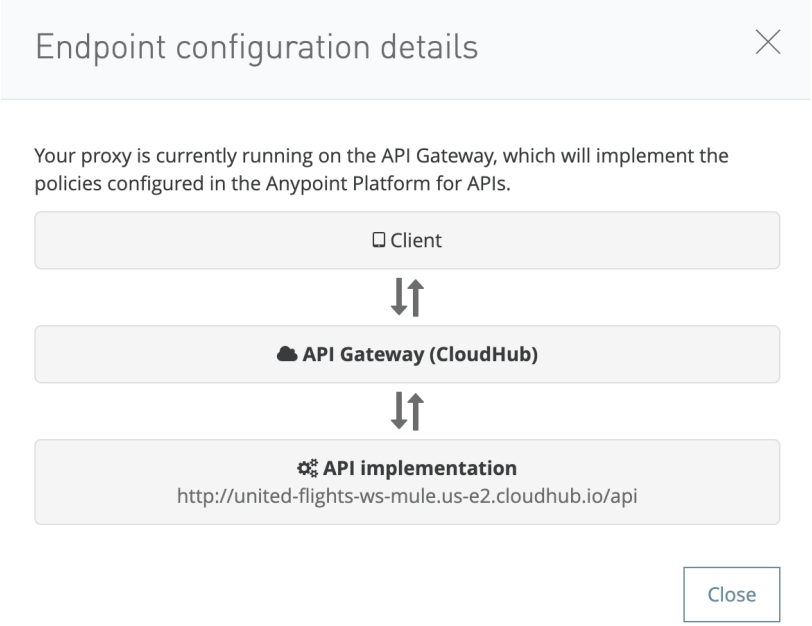
    Note: This is the API implementation endpoint URI that is masked by the API proxy URI. This shows you that an API proxy can point to an API implementation endpoint that you do not manage or control.

16. Click Next.
17. Review your selections and click Save & Deploy.
18. Wait for the API proxy to deploy successfully.

View the configuration details for this type of API instance

19. In the upper-right, click the View configuration details link.

20. Look at the Endpoint configuration details dialog box; you should see a two-step proxy configuration diagram showing that API clients connect to the API Gateway that forwards requests to the API implementation endpoint URL.



21. Click Close.

Setup the consumer endpoint for the United Flights System API instance

22. Below the label field, locate the Proxy URL link and copy it to the clipboard.

    Proxy

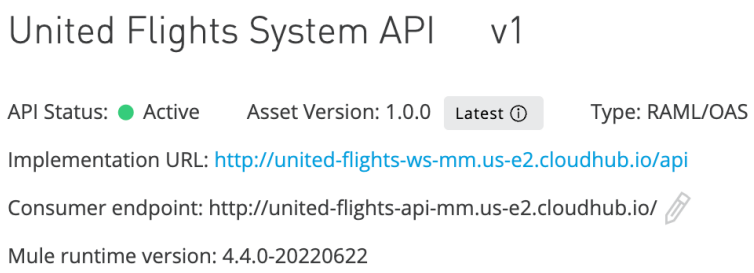
    Proxy Application: united-flights-api-mule

    Proxy URL: united-flights-api-mule.us-e2.cloudhub.io

    Note: Right click the link and select Copy link address to copy the entire link including the http:// protocol prefix.

23. In the top of the United Flights System API instance settings page, click Add consumer endpoint.

24. Paste the copied Proxy URL link and press Enter.



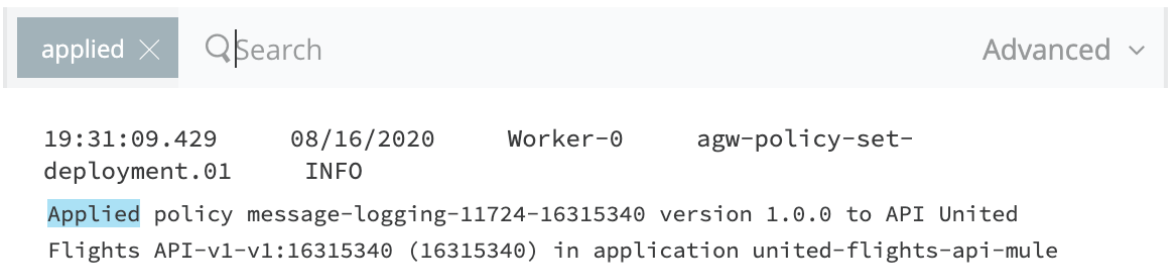
Note: Unlike the API implementation URL, this consumer endpoint of the API proxy URI does not have to include /api/ in the base path. This shows you how API proxies can simplify API endpoint access for API clients.

Test the United API instance by using the managed API's Exchange portal

25. Click View API in Exchange.
26. In the Endpoints list, click /flights.
27. Click POST.
28. In the API Console, click the Select server drop down and select the Sandbox – v1.
29. Click Send.
30. Verify that you get a 201 Created response.

View the logs for the API proxy Mule application in Runtime Manager

31. Return to Runtime Manager.
32. In the left-side navigation, click Applications.
33. Click the united-flights-api-{initials} application.
34. In the left-side navigation, click Logs.
35. Search for the word applied.
36. Verify the message logging automated policy was applied.



Walkthrough 4-4: Override API instance policies with an automated policy

In this walkthrough, you create a JSON threat protection automated policy and verify it applies to the American Flights System API instance and overrides the JSON threat protection policy already defined and deployed to the API instance.

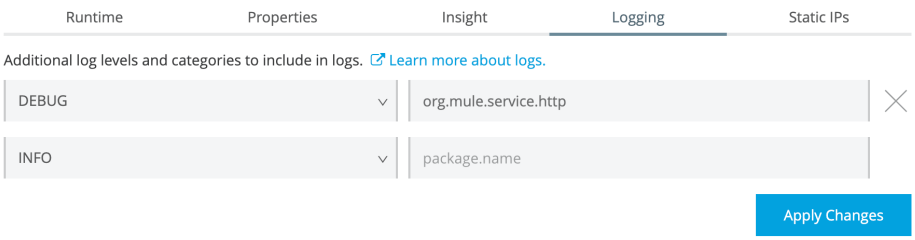
- Enable wire logging in an API proxy Mule application.
- Create an automated JSON threat detection policy.
- Verify the policy applies to both the American Flights System API instance and the United Flights System API instance.

Enable wire logging in the API proxy

1. In Runtime Manager, click the name of the united-flights-api-{initials} instance.
2. In the left-side navigation, click Settings.
3. Switch to the Logging tab.
4. Change the log level to DEBUG and type in the package.name.

```
org.mule.service.http
```

5. Hit enter in your keyboard.

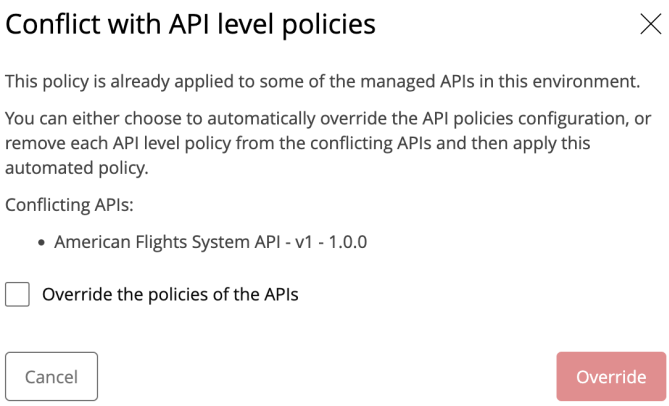


6. Click Apply Changes.

Override the American API JSON threat protection policy with an automated policy

7. Return to API Manager and navigate to the API Administration page.
8. In the left-side navigation, click Automated Policies.
9. Click +Add automated policy.
10. Under Security category select JSON threat protection.
11. Click Next.

12. In the configuration screen change the Maximum Container Depth value to 2.
13. Expand Advanced options and for the Rule of application select the following values.
    - Mule gateways only
    - **From version**: 4.4.0
14. Click Apply.
15. Read the warning in the Conflicts with API Level Policies dialog box.



16. Check the Override the policies of the APIs checkbox and click Overrride.

Verify the JSON threat protection automated policy is applied to all the managed API instances

17. Return to Runtime Manager; the Logging tab of the united-flights-api-{initials} application should still be selected, otherwise navigate to that page.
18. In the search bar, type json-threat.
19. Wait for the JSON threat protection automated policy to be applied and print a log in the API proxy's log.
20. Scroll down and search for log messages created by the message logging policy.

```
20:12:07.184     08/16/2020     Worker-0     agw-policy-set-deployment.01     INFO

************************************************************************
* Policy: json-threat-protection-11725-16315340-proxy                  *
* OS encoding: UTF-8, Mule encoding: UTF-8                             *
*                                                                      *
************************************************************************

20:12:07.191     08/16/2020     Worker-0     agw-policy-set-deployment.01     INFO
Applied policy json-threat-protection-11725-16315340 version 1.1.4 to API United Flights API-v1-v1:16315340 (16315340) in
application united-flights-api-mule
```

Note: The policy is applied without restarting the Mule application.

21. Click Applications from the left menu.

22. Click the american-flights-{initials} application name.
23. From the left menu select Logs.
24. In the Search bar, type json-threat.
25. Confirm that the JSON Threat Policy was applied.

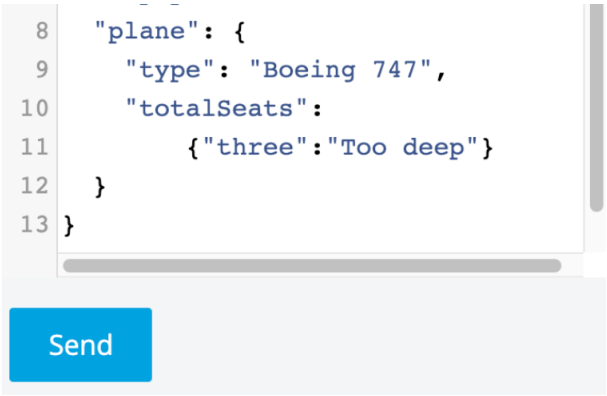
```
************************************************************************
* Policy:                                                              *
* json-threat-protection-11725-16311637-american-flights-api-main      *
* OS encoding: UTF-8, Mule encoding: UTF-8                             *
*                                                                      *
************************************************************************

20:11:41.757     08/16/2020     Worker-0     agw-policy-set-deployment.01     INFO
Applied policy json-threat-protection-11725-16311637 version 1.1.4 to API American Flights API-v1-v1:16311637 (16311637) in
application american-flights-yh
```

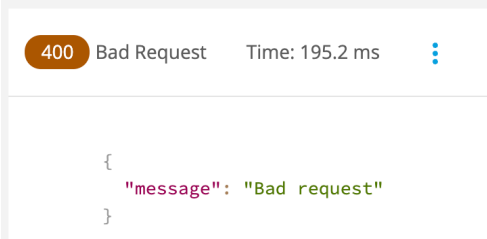
Note: Notice the logs also show the previous API Instance Json Threat Protection policy unapplied.

Send requests to the API implementation

26. Return to the API Settings page of American Flights System API v1.
27. Click View in Exchange and select /flights POST method.
28. Under Select server select Sandbox v1.
29. Send a POST request and verify you get a 201 Created response.
30. From the snippets file copy the total seats fragment under Three Levels Deep JSON Payload.
31. Replace the totalSeats line with the value copied from the snippets file to form a JSON payload that has an object nested three-levels deep.



32. POST the request and verify a 400 Bad Request error status is returned.



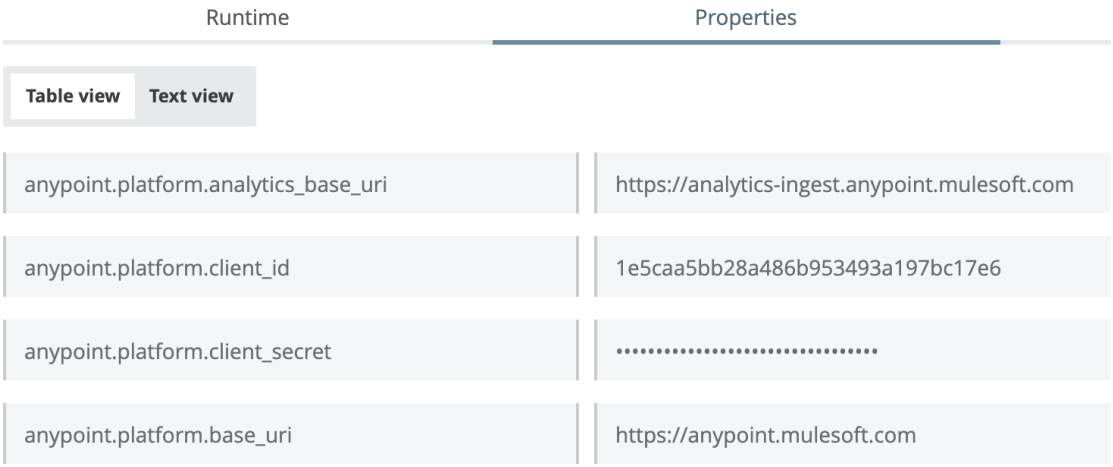
Note: This shows you the JSON threat detection policy is applied to the API proxy API instance and is rejecting the request payload because it has an object nested three-levels deep.

View the properties set to connect the API proxy application with Anypoint Platform

33. Return to Runtime Manager and select the united-flights-api-{initials} application.
34. On the left menu click Settings.
35. Select the Properties tab.
36. Notice the Anypoint Platform client_id and client_secret properties.

Note: This client_id is the same value you copied from the Sandbox environment information in API Manager to configure the american api instance.

37. Notice there are two other properties set to connect with the Anypoint Platform.



Module 5: Enabling API Governance and enhancing security

At the end of this module, you should be able to:

- Implement API Governance profiles and rulesets.
- Validate API Governance conformance from API Design and Exchange.
- Use Anypoint CLI to automate the API Governance tasks.
- Create custom governance rulesets.
- Promote managed APIs from one environment to another.
- Secure managed APIs with the OAuth2 token enforcement policy.

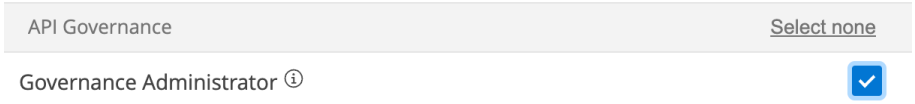
Walkthrough 5-1: Create a new API Governance profile and define rulesets to apply to APIs

In this walkthrough, you set up the tags and categories needed to apply a predefined governance ruleset to an API. You will:

- Set required permissions to use API Governance.
- Set up tags to identify which APIs will be used by API Governance.
- Apply a predefined API Governance ruleset.
- Set up alerts and conformance notifications.

Set API Governance administrator user permission

1. In the main menu select Access Management.
2. From the Access Management menu select **Users**.
3. Click on your Username for the admin user or the user that you have been using during the class.
4. On the Permissions tab, click **Add permissions**.
5. On the Select Permissions dialog, scroll down to the API Governance section and select the **Governance Administrator** permission.

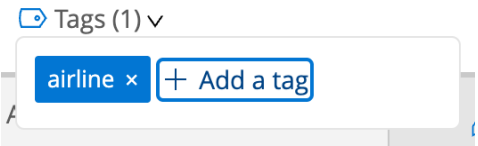


   Note: This permission is not required for users with Organization Administrator permissions.
6. Click **Next**.
7. In the Select Business Groups dialog select the name of your business group.
8. Click **Next**.
9. Hover over API Governance to verify that **Governance Administrator** is listed.

Apply Tags to APIs being governed

10. Navigate to Anypoint Platform.
11. From the main menu, navigate to Exchange.
12. Select American Flights System API under API Management assets.
13. Click +Add Tags then +Add a tag.

14. Type **airline** and hit enter.



15. Click go back to assets list.
16. Select United Flights System API under API Management assets.
17. Click +Add Tags then +Add a tag.
18. Type **airline** and hit enter.

Create a new Governance profile

19. From the main menu select Management Center – API Governance.
20. Click +New Profile.
21. In the General Information step enter the following information:

    - **Profile Name**: Airlines Security
    - **Purpose:** Security - Protect access to all airlines APIs

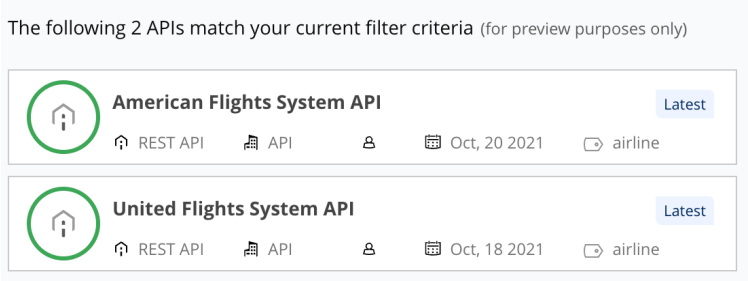
22. Click Next.
23. For Rulesets select Authentication Security Best Practices.

    Note: To access the documentation for this ruleset click View details in Exchange.

24. Click Next.
25. Under Filter Criteria select the following values:

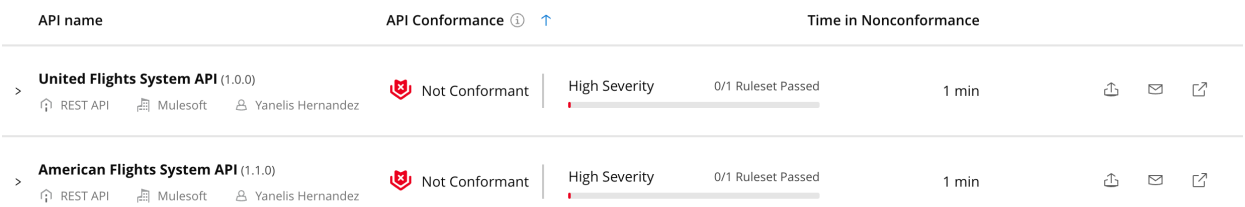
    - **API Types**: REST API (if not selected already)
    - **Tags**: airline

26. Make sure American and United Flights System APIs match your filter criteria.



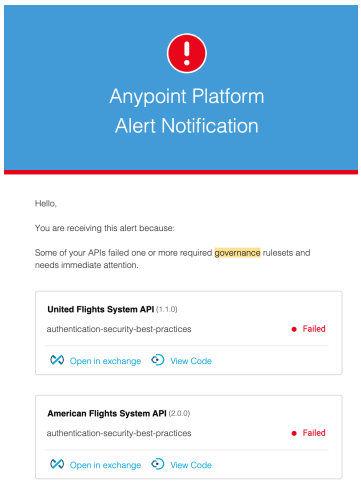
27. Click Next.
28. Leave Nonconformance notifications unchanged, click Next.

29. Review your settings and click Create.
30. Click on the Airlines Security profile name.
31. Confirm the API Governance Console created with the Airlines security profile and conformance statuses.



Check Governance API Alerts.

32. After a few minutes, check the email address specified as the API owner.
33. Locate and open an email with the subject Governance API Alert Summary.



Note: The alert will contain the APIs that are failing the governance rulesets.

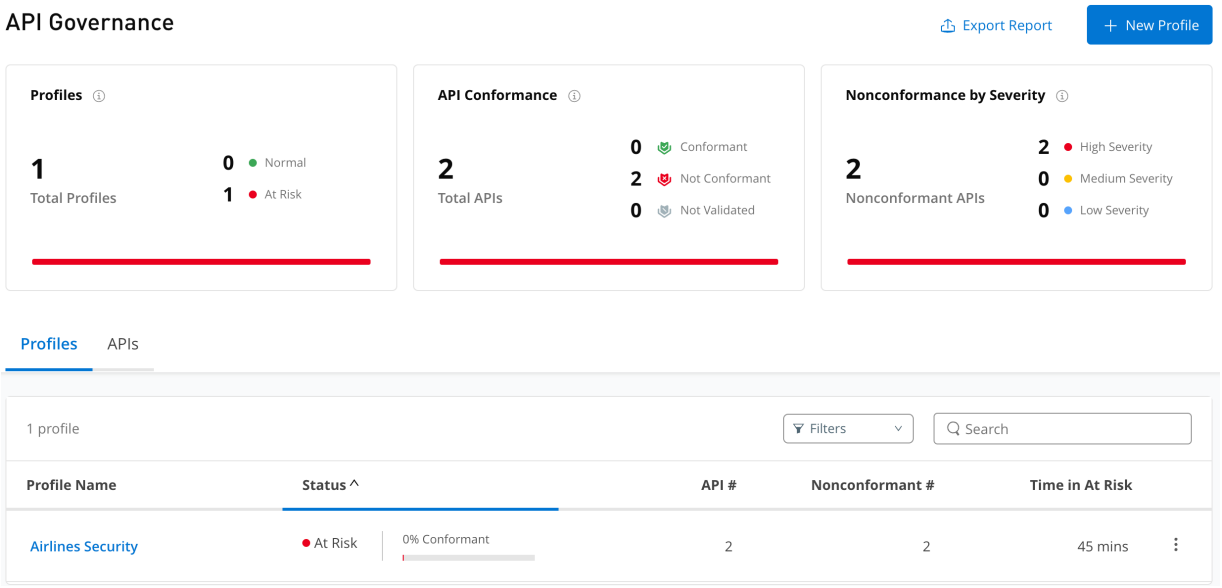
Walkthrough 5-2: Identify and monitor conformance with API Governance

In this walkthrough, you view the status of API conformance, identify and correct conformance issues. You will:

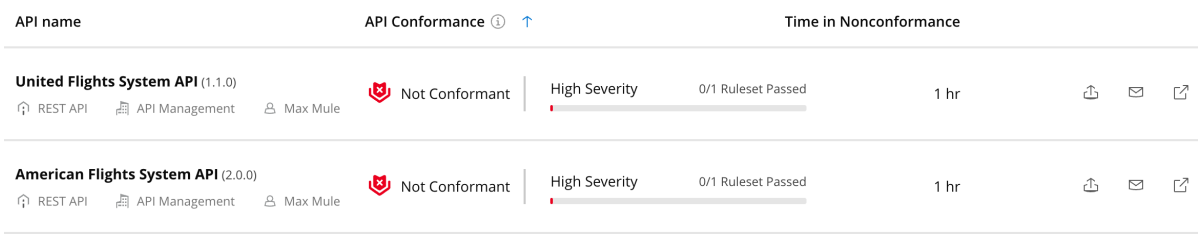
- View the status of API conformance to rulesets.
- Create a conformance report.
- View conformance status in Exchange.
- Identify conformance issues in API designer.

Examine the API conformance status to a ruleset

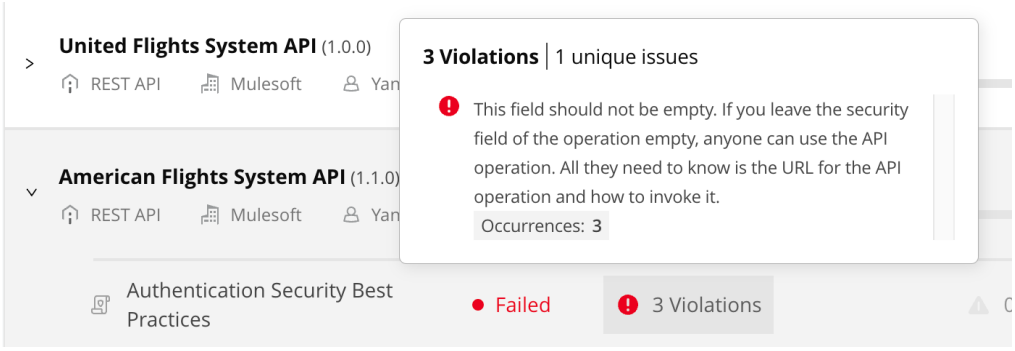
1. In Anypoint Platform navigate to API Governance.
2. Note the API Governance console showing the profile created in the previous walkthrough, with the conformance information for the APIs that matched the criteria.



3. Select the APIs tab.
4. Notice both American Flight System API and United Flight System API are in Not Conformant status.



5. Expand American Flight System API by clicking on the arrow before the name.
6. Verify that for the Authentication Security Best Practices there are violations.
7. Click on the three violations to see a brief description of the violations.



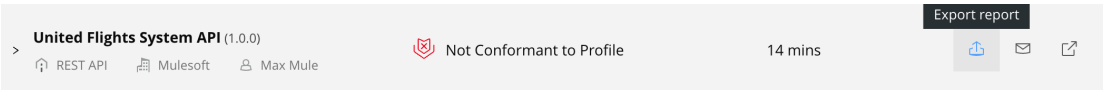
Generate reports and send emails with conformance statuses

8. In the upper right corner locate and click Export Report.

9. Open the file just created in your local with any CSV editor.

   Note: If the name is not changed before downloading it will be named report.csv. The report will contain all the APIs, version, pass, fail, and time since nonconformance.

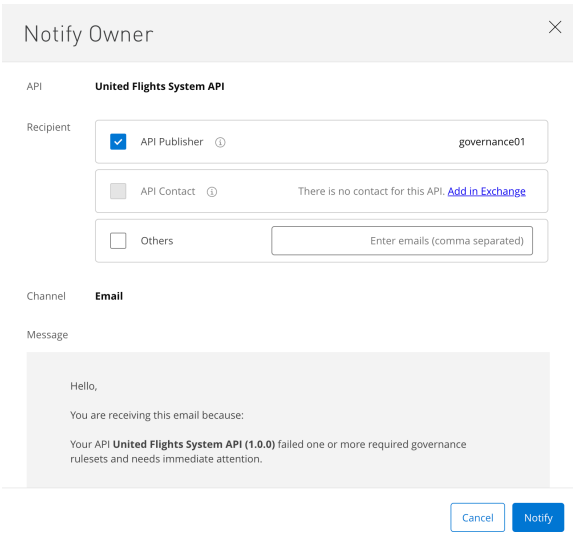
10. Return to the API Governance console – Airlines Security Dashboard and scroll down to the United Flight System API.
11. Click on export report for this API.



12. Open a new file created with a detailed report for this the United Flight System API.

    Note: This report will be specific for the United Flight System API. It will contain conformance status, duration, error and warning count.

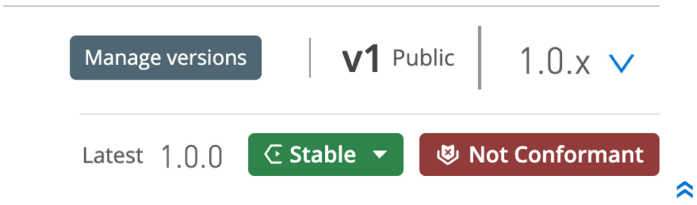
13. Click on the email icon to notify owner via email about the nonconformance status.



14. After a few minutes the API Publisher will receive an alert with the nonconformance status.

View conformance status in Anypoint Exchange

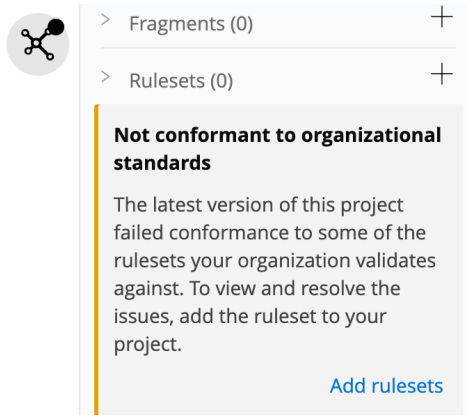
15. Navigate to Exchange.
16. From the list of assets from your organization select American Flight System API.
17. Note the Not Conformant badge in red.



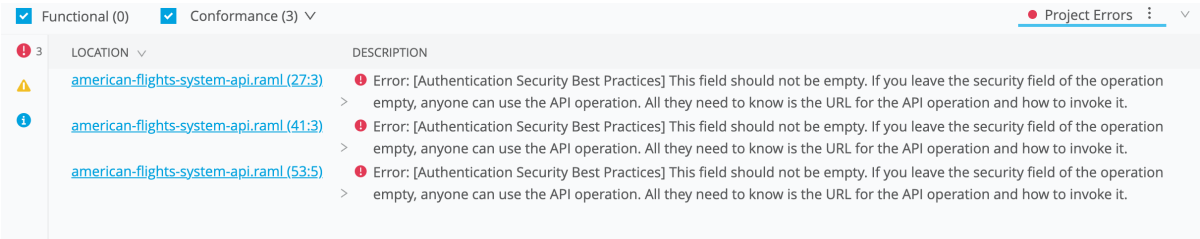
View detailed errors explanations in API Designer

18. Click View code to open the API specification in PI Designer.
19. If the API opens in read only view, click Edit spec.
20. Click on the Exchange Dependencies icon on the left menu.

21. Notice the message stating that this API is not conformant to organizational standards.



22. Click Add rulesets.
23. On the Rulesets in organizational standards click the Add rulesets button.
24. Open the project errors panel to see location and description of each of the ruleset violations.



Note: Each of the violations point to one operation without security schema.

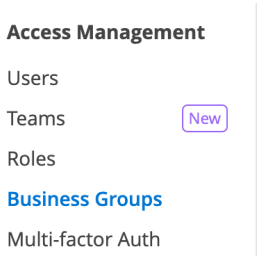
Walkthrough 5-3: Promote a managed API from a Sandbox environment to a Production environment

In this walkthrough, you promote a managed API instance from sandbox to production. You will:

- Create an environment named Production.
- Promote a managed API instance from Sandbox to Production environment.

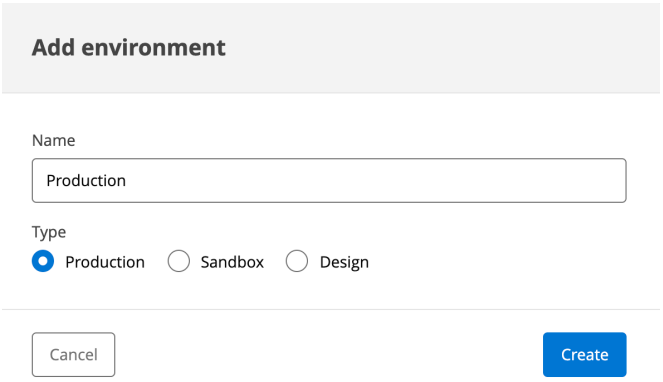
Create an environment named Production in a business group

1. In the main menu select **Access Management**.
2. In the left-side navigation, click **Business Groups**.



*Note: If your graphical user interface looks different, click **Try new feature**s button at the end of the menu.*

3. Click the name of your business group.
4. Switch to the **Environments** tab.
5. Verify that a Design and Sandbox environment is present in the list of available environments.
6. Click Create environment.
7. In the Add environment dialog box, type the Environment name as Production.
8. Select Sandbox radio button.

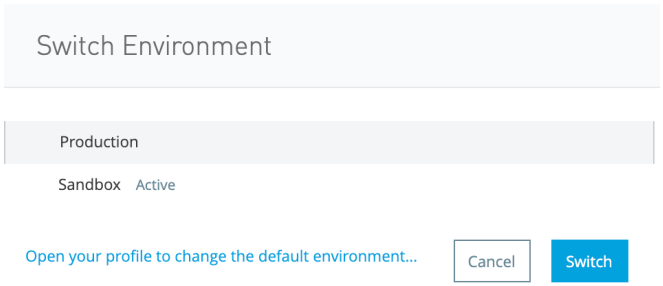


Note: Since the Production environment vCores are a paid entitlement, we will create a sandbox type environment for demonstration purposes.

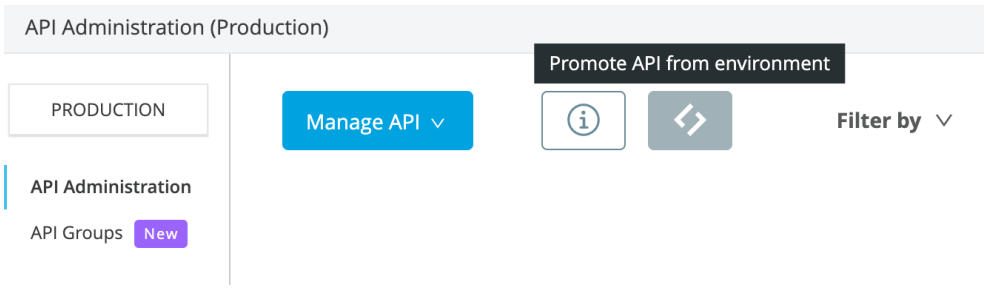
9. Click Create.

Promote a managed API instance of an API from Sandbox to Production

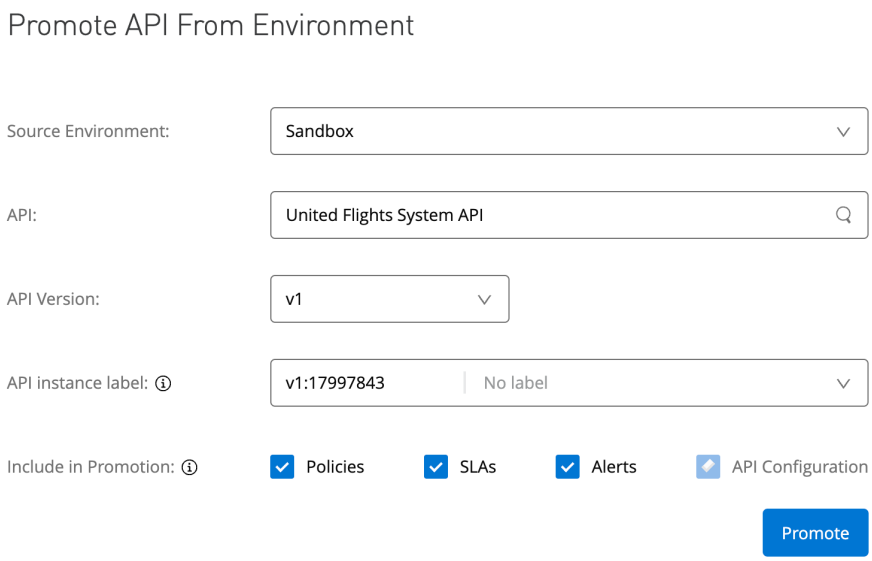
10. Navigate to the United Flights System API (v1) Settings page.
11. In the left-side navigation click Sandbox.
12. In the Switch environment dialog box, select Production and click Switch.



13. In the Production environment API Administration page, click promote API from environment icon.



14. In the Promote API from Environment page, fill in the following information:
    - Source Environment: Sandbox
    - API: United Flights System API
    - API Version: v1
    - API Instance label: <leave blank>



15. Click Promote.

Deploy the United Flights API Proxy in Production

16. Expand the Runtime and Endpoint Configuration section.
17. Set the proxy app name as united-flights-api-prod-{initials}.
18. Set the consumer endpoint as united-flights-api-prod-{initials}.
19. Click Save & Deploy.
20. Verify that the deployment was successful, and the application is registered.

Try to test the United Flights API in Production

21. Return to Exchange.
22. Change the API version to 1.1.x if it is not previously selected.
23. In the United Flights System API left-side navigation menu, click the /flights resource GET method.
24. In the API Console, click the Mocking service endpoint and select Production – v1.

25. Click Send.
26. Confirm that Client Id and Secret are required as well as the scope value. Also the response is 400 Bad Request.

Walkthrough 5-4: Add Okta as an external client provider in Anypoint Platform

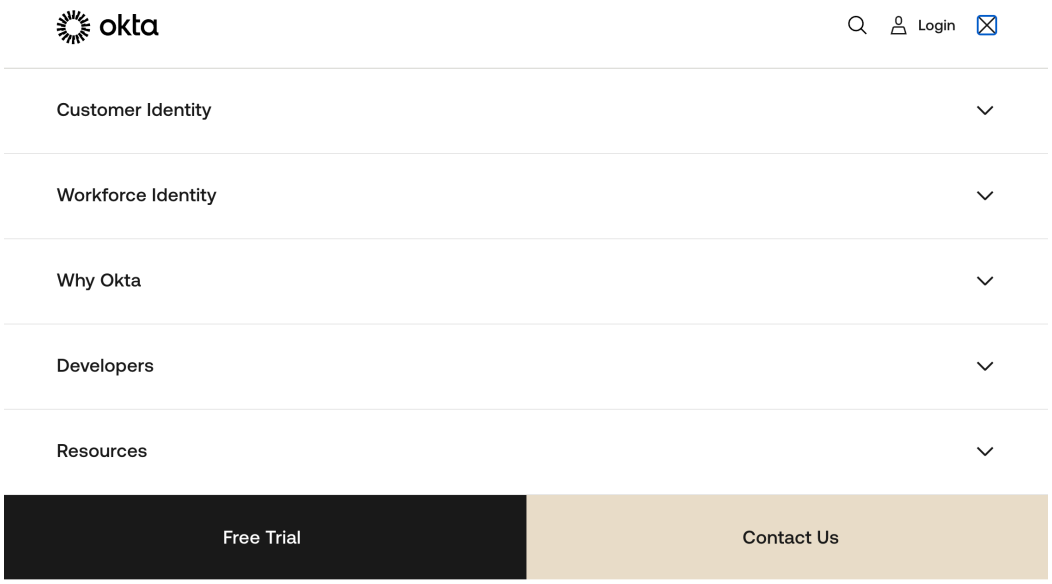
In this walkthrough, you will add Okta as an external client provider and configure it as a client identity provider. You will:

- Register for a trial account with Okta.
- Setup an Authorization Server using Okta.
- Configure an OpenID Connect Client Provider in Anypoint Platform using Okta.

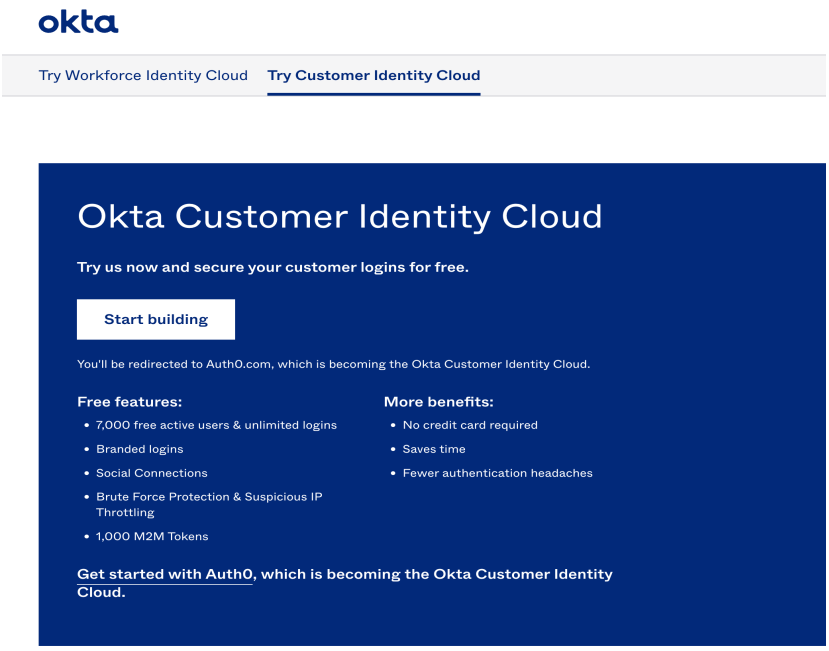
Create Okta account

1. Navigate to okta.com and click on the three bars in the top right corner.

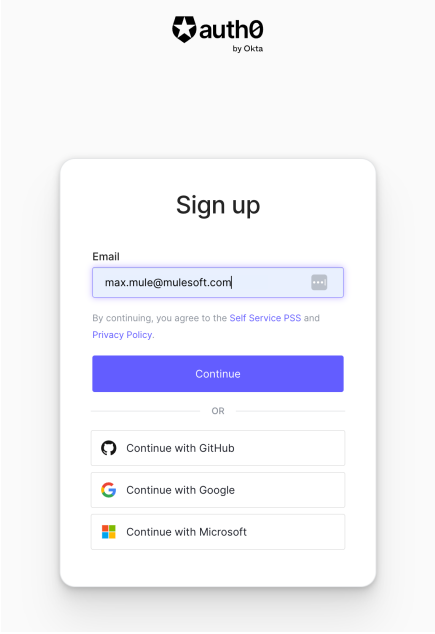
2. From the menu, select Free Trial.



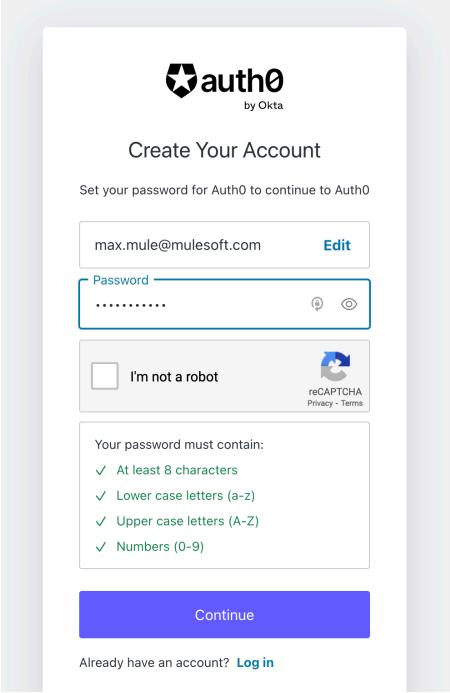
3. In the free-trial window that opens, switch to the Try Customer Identity Cloud tab.



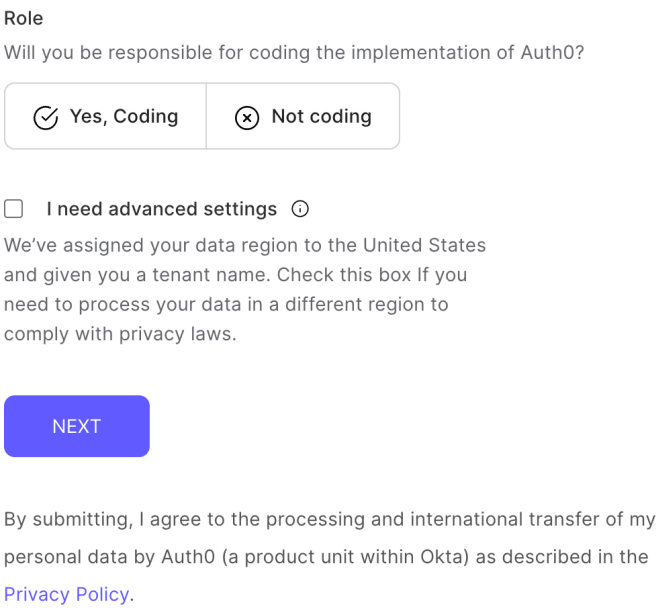
4. Click on Start Building button.
5. In the Auth0 sign up window enter your email and click on Continue.



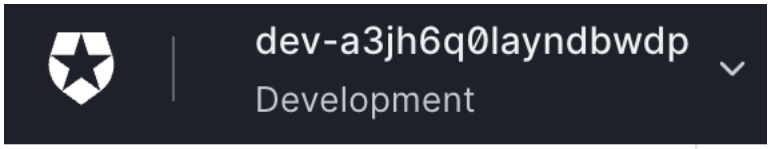
6. Create a password for your account, validate the reCAPTCHA and click on the Continue button.



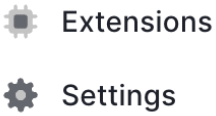
7. Select Not Coding when asked for your Role and click on Next.



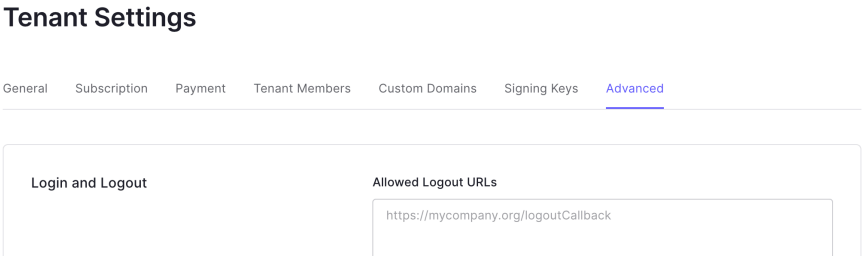
8. Copy the default tenant id assigned to you and paste it in your snippets.txt file. You can find it in the top left corner.



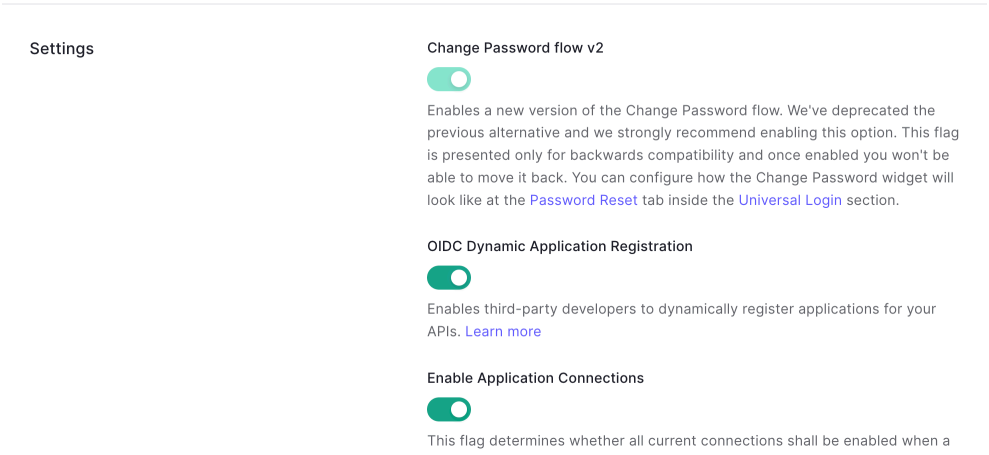
9. In the left menu, navigate to Settings.



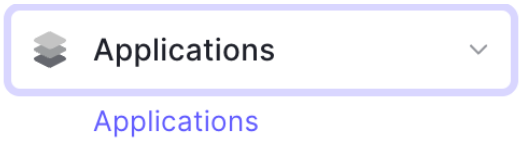
10. Navigate to Advanced tab in the Tenant Settings window.



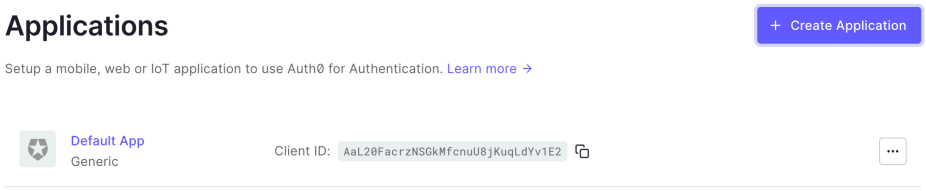
11. Scroll down to Settings and turn on the OIDC Dynamic Application Registration.



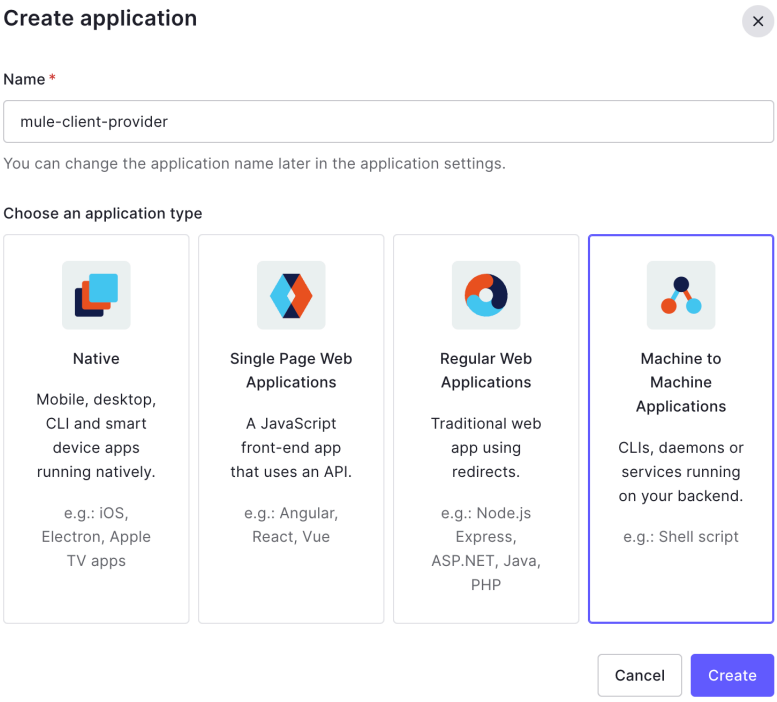
12. In the left menu, navigate to Applications->Applications.



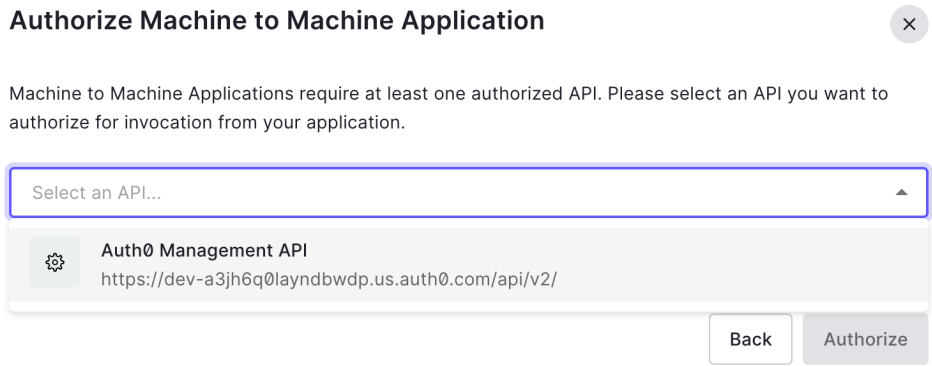
13. Click on Create Application.



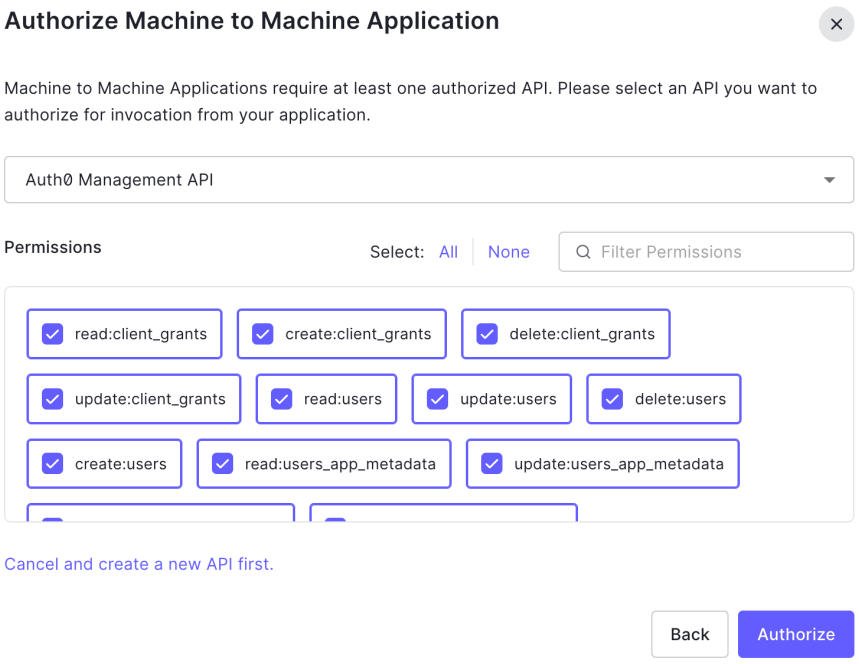
14. Name your application mule-client-provider. Select Machine to Machine Applications as the application type. Click on Create.



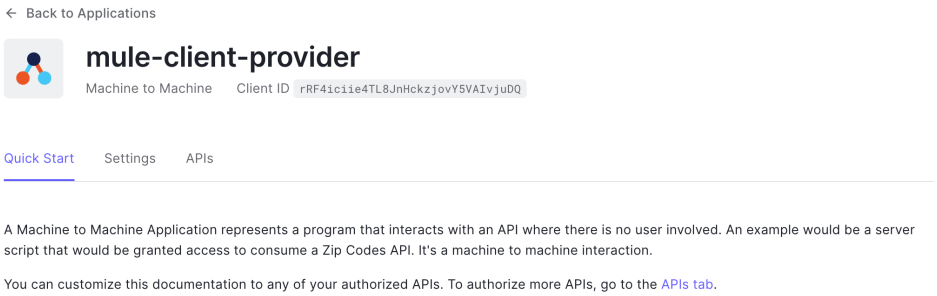
15. From the dropdown list of APIs select Auth0 Management API to authorize invocation from your application.



16. In the Permissions window click on Select All.

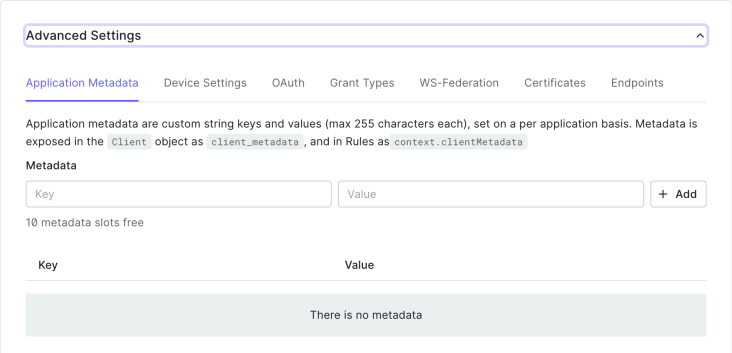


17. Click on Authorize.
18. In the application window that is created for mule-client-provider application, select the Settings tab.

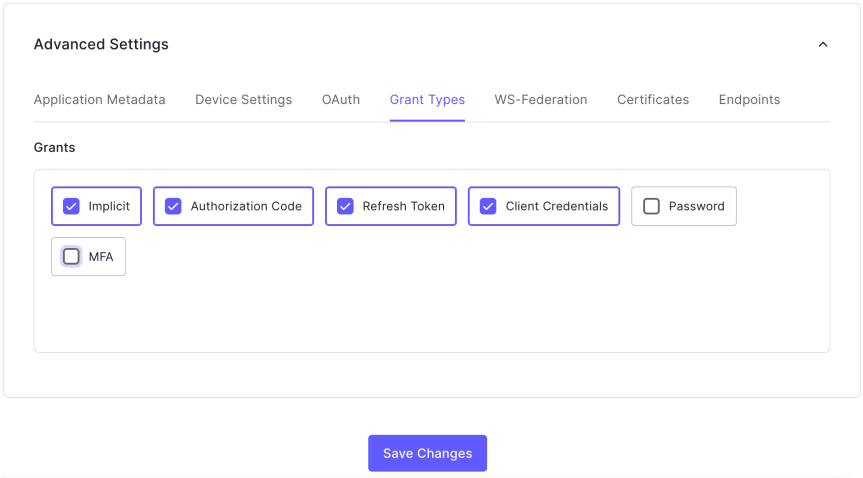


19. Copy the values of Client Id and Client Secret and paste them in your snippets.txt file.

20. Scroll down to Advanced Settings.

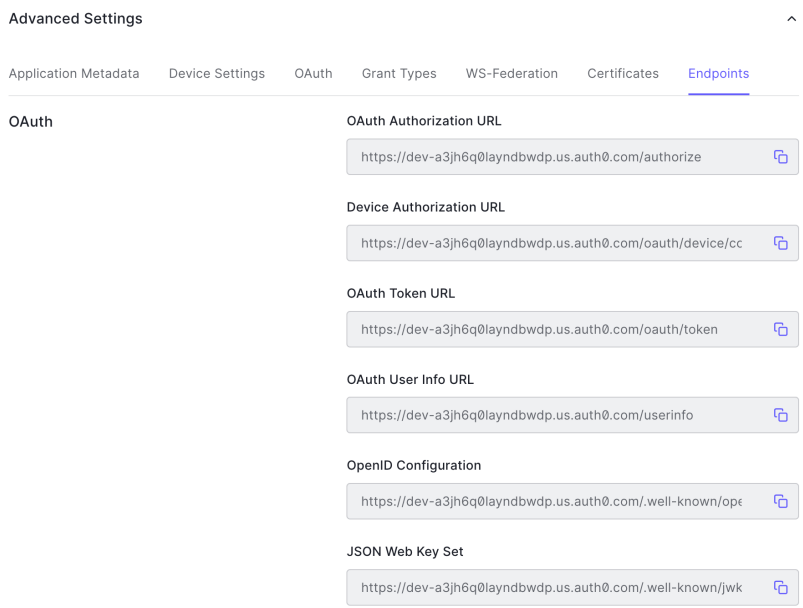


21. Select the Grant Types tab.
22. Enable the Grants – Implicit, Authorization Code, Refresh Token, Client Credentials. Click on Save Changes.

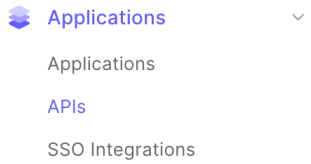


23. Navigate to Endpoints tab.

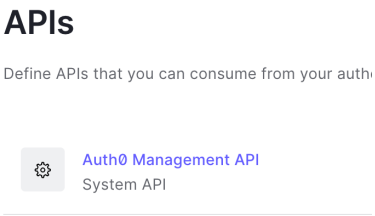
24. Copy the values of OAuth Authorization URL, OAuth Token URL and OAuth User Info URL and paste them in the snippets.txt file.



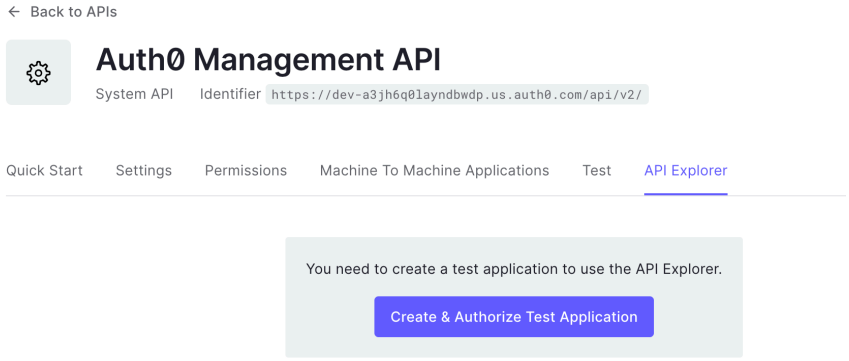
25. From the left menu, navigate to Applications->APIs.



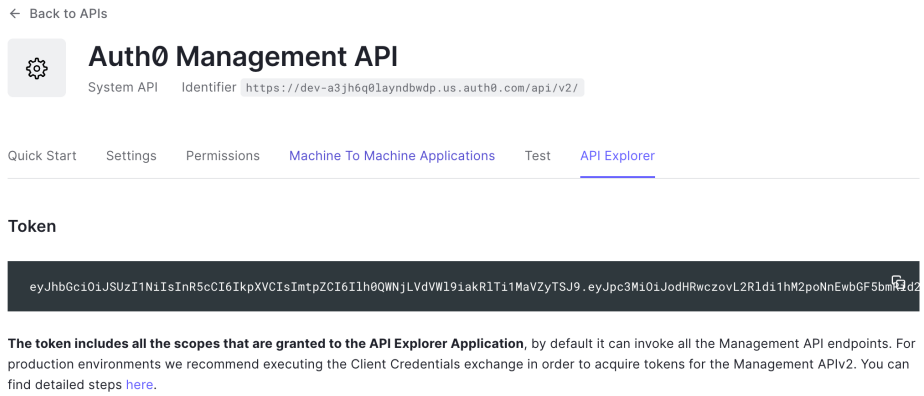
26. Click on the Auth0 Management API.



27. Navigate to API Explorer tab.



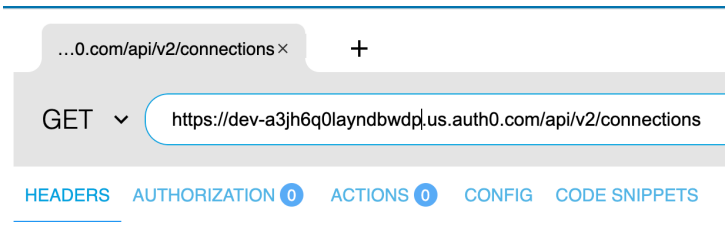
28. Click on Create & Authorize Test Application.
29. Copy the value of Token and paste it in your snippets.txt file as Test Application Token.



30. Open Advanced Rest Client.
31. Copy the following URL and paste it in a new request in Advanced Rest Client.

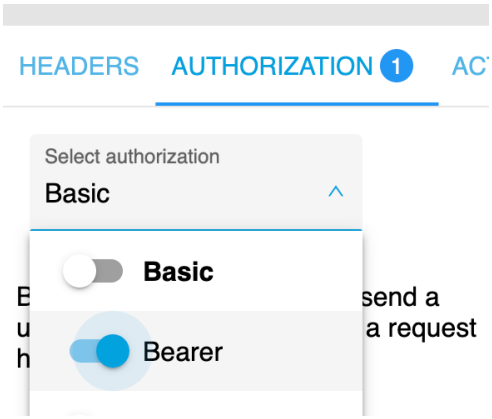
```
https://{your auth0 tenant id}.us.auth0.com/api/v2/connections
```

32. Copy the value of your tenant id from snippets.txt file and replace it in URL.

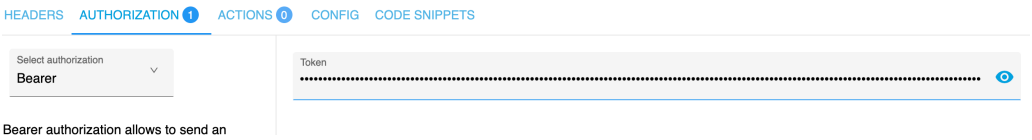


33. Select the Authorization tab.

34. From the dropdown toggle the slider next to Bearer and click on it.



35. Paste the value of token from snippets.txt file and paste it in Advanced Rest Client.



36. Send the Request. Verify you get a 200 OK response.
37. In the response section, scroll down to find Username-Password-Authentication.
38. Make note of the value of the connection id for Username-Password-Authentication from the response and paste it in snippets.txt file.

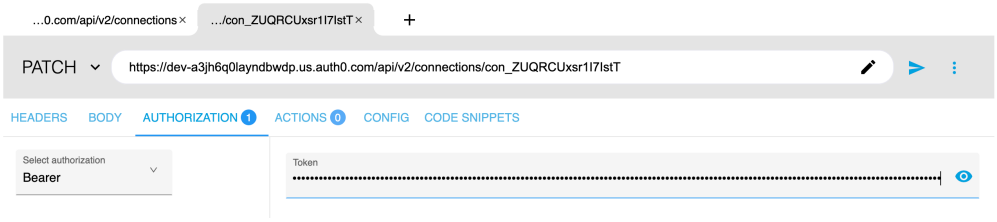

```
{
  "id": "con_ZUQRCUxsr1I7IstT",
  "options": {
    "mfa": {
      "active": true,
      "return_enroll_settings": true
    },
    "passwordPolicy": "good",
    "strategy_version": 2,
    "brute_force_protection": true
  },
  "strategy": "auth0",
  "name": "Username-Password-Authentication",
  "is_domain_connection": false
```


39. Open a new tab in Advanced Rest Client and change the request type to PATCH.
40. Copy and paste the following URL into the request URL.

```
https://{your tenant id}.us.auth0.com/api/v2/connections/{your connection id}
```

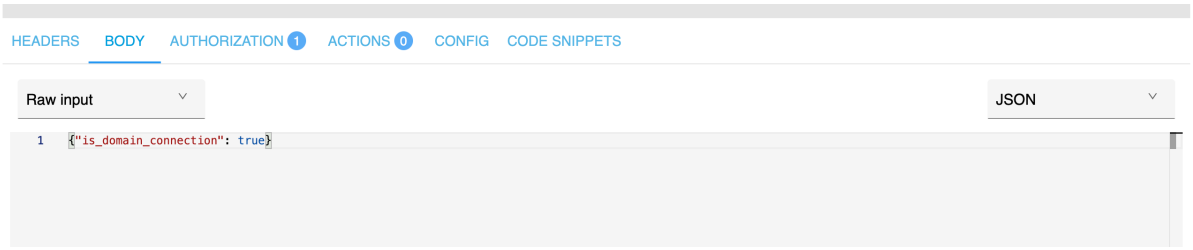
41. Copy the values of your tenant id and connection id from snippets.txt file and replace them in the URL in Advanced Rest Client.
42. Navigate to Authorization tab and change the authorization method to Bearer.

43. Paste the value of token from snippets.txt file in the request.

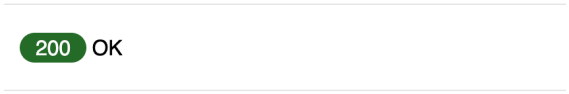


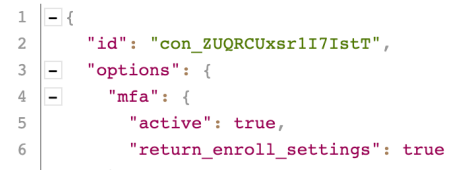
44. Navigate to Body tab.
45. Change the content type to JSON in the body.
46. Copy the following and paste it in the request body.

```
{“is_domain_connection”: true}
```



47. Send the request.
48. Verify you get a 200 OK response.

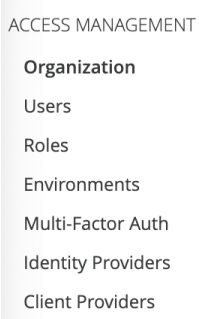




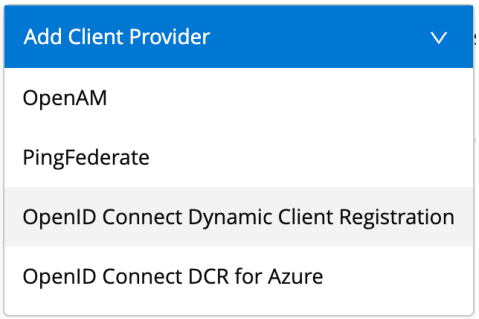
49. Scroll down in the response section and verify the value of “is_domain_connection” is set to true.

50. Navigate back to Anypoint Platform.
51. From the main menu navigate to Access Management.

52. From the left menu select Client Providers.



53. Click on Add Client Provider and from the drop down, select OpenID Connect Dynamic Client Registration.

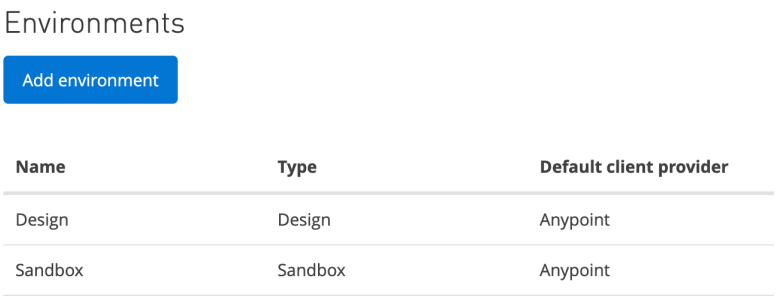


54. Provide the following details in the OIDC client provider window.

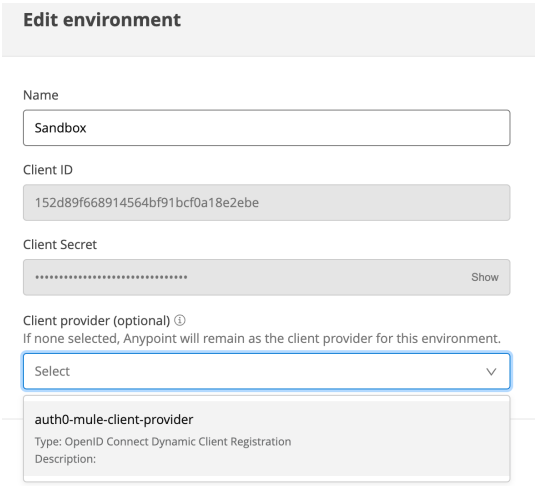
- Name: auth0-mule-client-provider
- Issuer: https://{your auth0 tenant id}.us.auth0.com
- Client Registration URL: https://{your auth0 tenant id}.us.auth0.com/oidc/register
- Client ID: <<Paste from snippets file>>
- Client Secret: >>Paste from snippets file>>
- Authorize URL: https://{your auth0 tenant id}.us.auth0.com/authorize
- Token URL: https://{your auth0 tenant id}.us.auth0.com/oauth/token
- Token Introspection URL: https://{your auth0 tenant id}.us.auth0.com/userinfo

55. Click on Create.

56. From the left menu, navigate to Environments.



57. Click on Sandbox.
58. From the Client provider drop down, select auth0-mule-client-provider.



59. Click on Update.
60. Repeat the above steps for Design environment to assign the client provider to Design environment as well.

Walkthrough 5-5: Apply an OpenId Connect access token enforcement policy

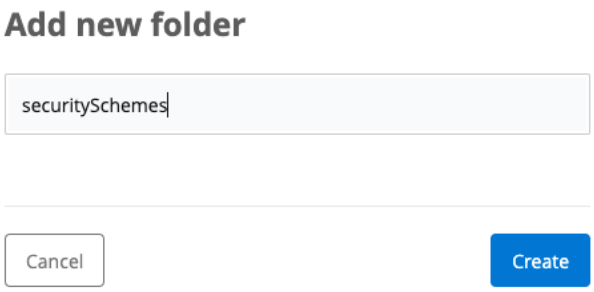
In this walkthrough, you will enforce OpenId token requirement to access an API. You will:

- Update an API RAML definition to specify the OAuth 2.0 security scheme.
- Apply the OpenId policy to the API.
- Create an API Client application to request access to the API implementation.
- Test the OpenId policy using Okta as the external provider.

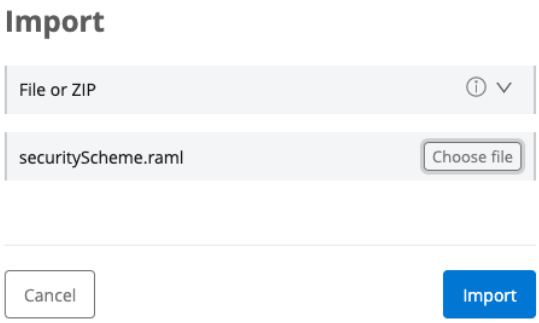
Update the API RAML to include OAuth 2.0 security scheme

1. From the main menu, navigate to Design Center.
2. Open the United Flights System API specification.
3. Create a new branch called v1.
4. From the left menu, click on the add button and click on New folder.

5. Name the folder securitySchemes.



6. Click on the menu next to the securitySchemes folder and click on Import.
7. Navigate to the student files folder and look for the file securityScheme.raml. Click on Import.



8. Open to edit the securityScheme.raml file in Design Center.
9. Replace the authorization and token endpoint with the values from the snippets file.
10. Open the united-flights-system-api.raml to edit.
11. Copy the securityScheme and securedBy fragments from the snippets file under the section Security Scheme and paste it in the united-flights-system-api.raml as shown below.

```
#%RAML 1.0
title: United Flights API
version: v1

types:
  Flight: !include datatypes/Flight.raml
securitySchemes:
  oauth_2_0: !include securitySchemes/securityScheme.raml

/flights:
  get:
    securedBy: [oauth_2_0]
    displayName: Get flights
    queryParameters:
      code:
        displayName: Destination airport code
        required: false
        enum:
          - SFO
          - LAX
          - PDX
          - CLE
          - PDF
    responses:
      200:
        body:
          application/json:
            type: Flight[]
            example: !include examples/UnitedFlightsExample.raml
  post:
    securedBy: [oauth_2_0]
    body:
```

Note: The securedby annotation must be applied to methods get and post.

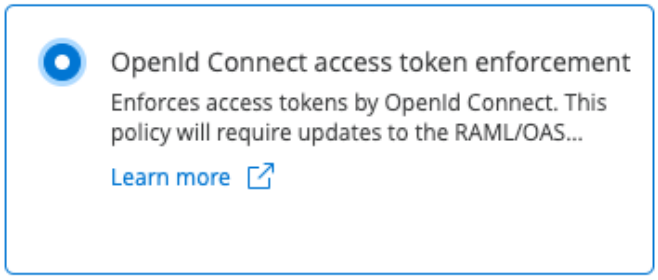
12. Click on Publish at the top of the page and click Publish to Exchange.
13. Ensure that the asset version shows as 1.1.0.
14. Click on Publish to Exchange.
15. Once the API is successfully published, click Done on the confirmation window.

Update API version and client provider in API Manager

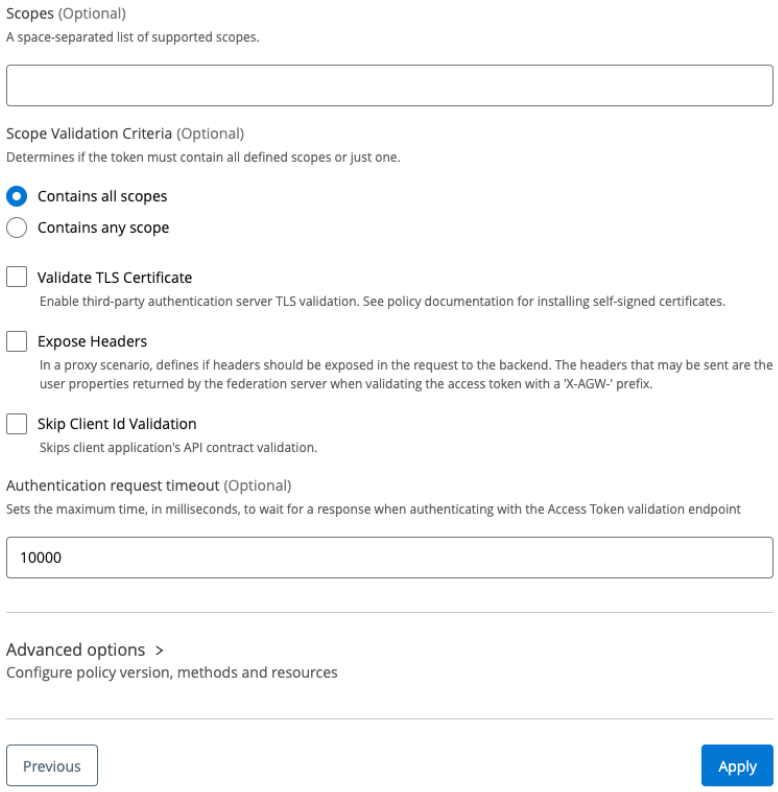
16. From the Main menu, navigate to API Manager.
17. Switch to Production environment.
18. Click the name of the United Flights System API v1.
19. Update the asset version to 1.1.0 which is the latest.
20. Expand Runtime & Endpoint Configuration section.
21. Click Save & Apply.
22. Wait for the application to restart and have an Active status.

Apply the Open ID Connect access token enforcement policy

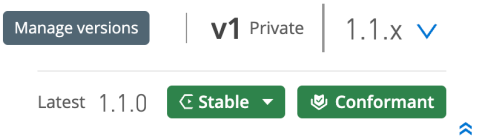
23. From the left menu, navigate to Policies.
24. Under the section API-level policies, click on Add policy.
25. Click on the Security Category.
26. Verify that you can now see the OpenId Connect access token enforcement policy.
27. Select the OpenId Connect access token enforcement policy.



28. In the policy configuration window, leave all values as they are and click on apply.

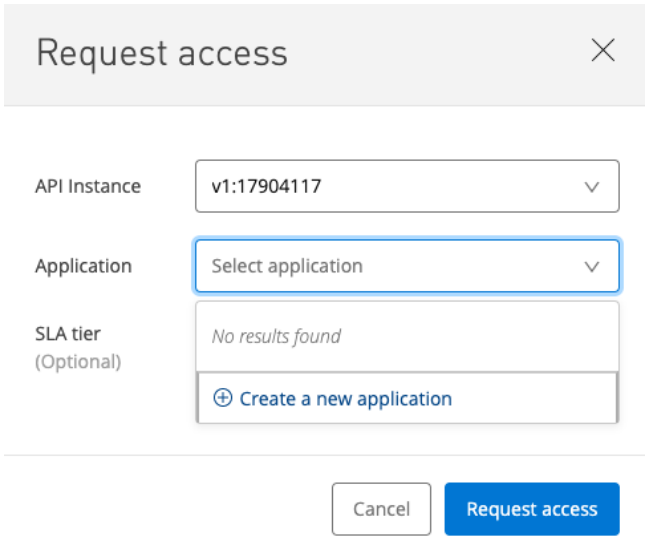


29. Verify the policy is created successfully.
30. From the main menu, navigate to Exchange.
31. Click on United Flights System API.
32. Note the latest version of the United Flight System API is showing as conformant. This is due to the security scheme applied to all operations resolved the authentication best practices violations.

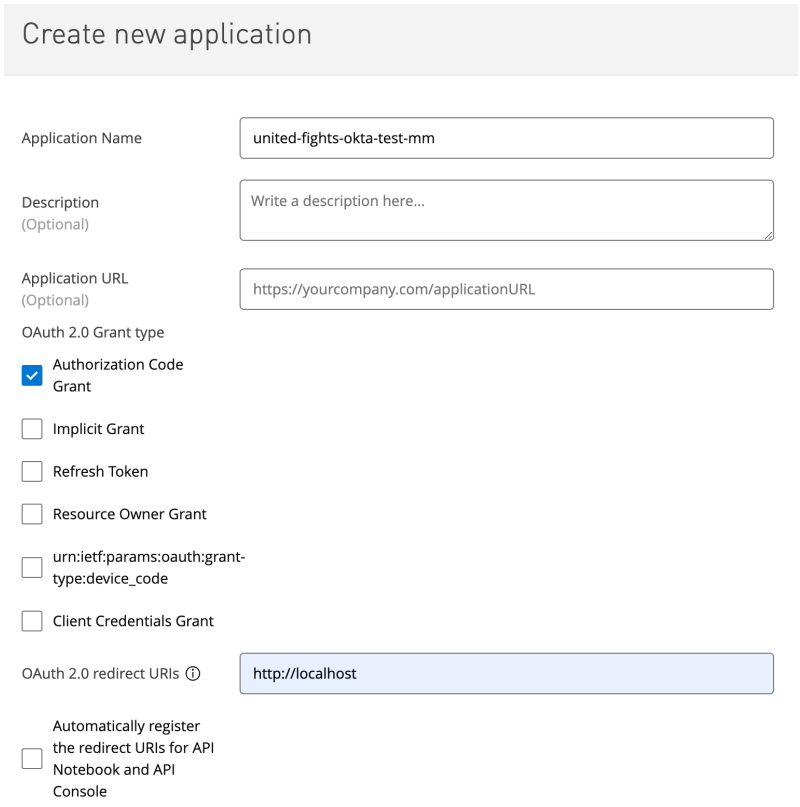


33. Click on Request access on the top.
34. In the Request access window that pops up, select the latest instance in the Production environment.
35. Click on the Select Application dropdown.

36. Click on Create a new application.

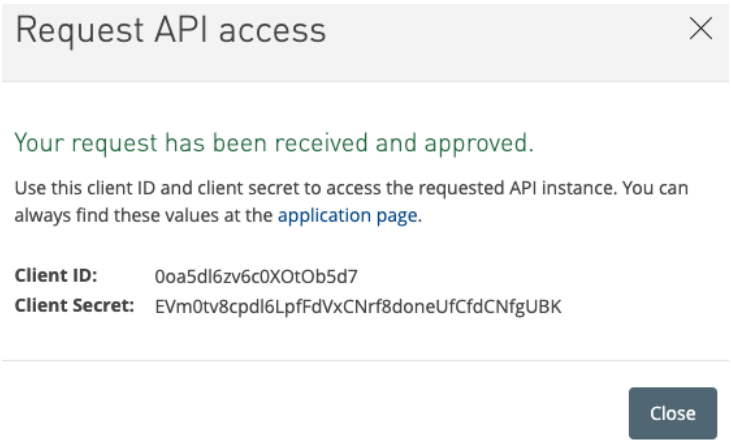


37. In the Create new application window, name the application as united-flights-okta-test-<initials>.
38. Select the OAuth 2.0 Grant types Authorization Code Grant and Client Credentials Grant.
39. In the OAuth 2.0 redirect URIs text field, enter http://localhost



40. Click on Create.
41. Click on Request access.

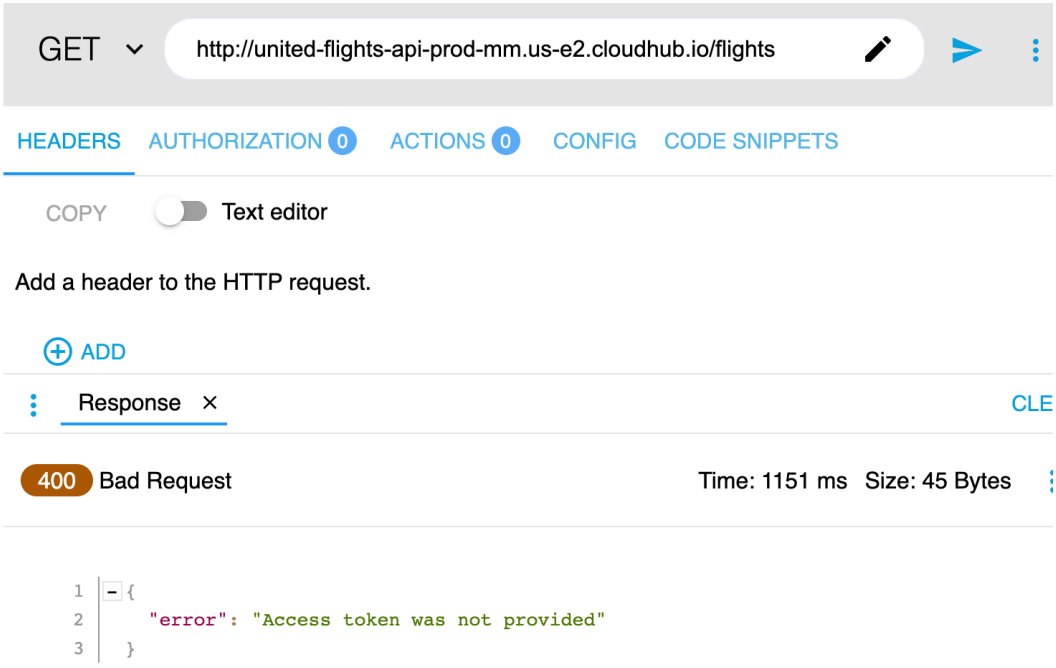
42. In the confirmation window, make a note of the Client ID and Client Secret in the snippets file under the section Client application details.



43. Click on Close.

Test the API in a Rest Client

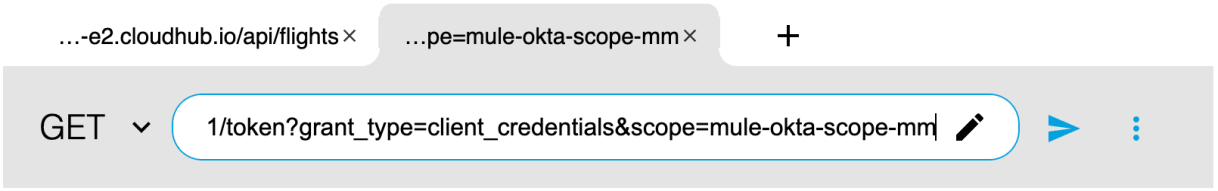
44. In a web client such as Advanced REST Client, send a GET request to the /flights resource of united-flights-api.



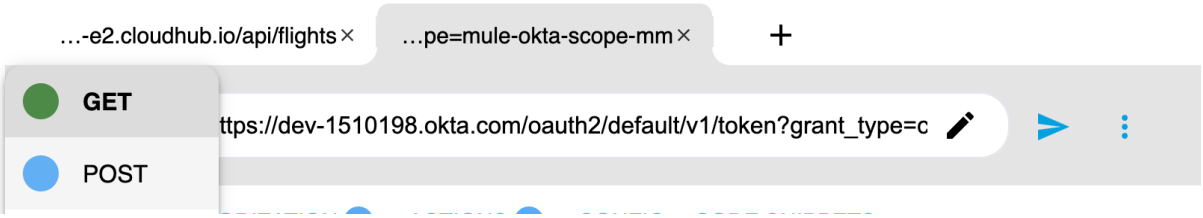
45. Verify that you get a 400 Bad Request response with error message “Access token not provided”.

46. Click on + to create a new request.
47. Copy your token_endpoint from your snippets file.
48. Paste it in the URL bar in Advanced REST Client.
49. Paste the following after the URL you pasted in the step before.

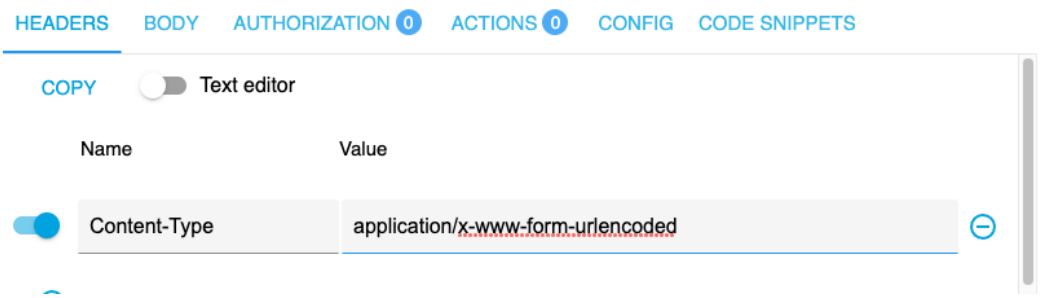
```
<your token_endpoint>?grant_type=client_credentials&scope=mule-okta-scope-
<your initials>
```



50. Click on the dropdown menu next to GET. Change the request to a POST.

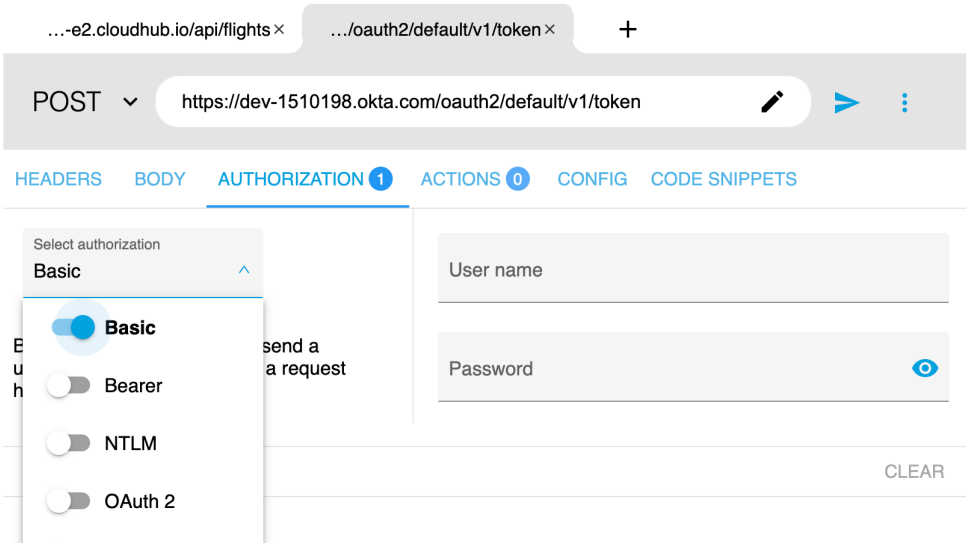


51. In the Headers Tab. Click ADD.
52. From the dropdown menu, select Content-Type. Put in the Header value as application/x-www-form-urlencoded.

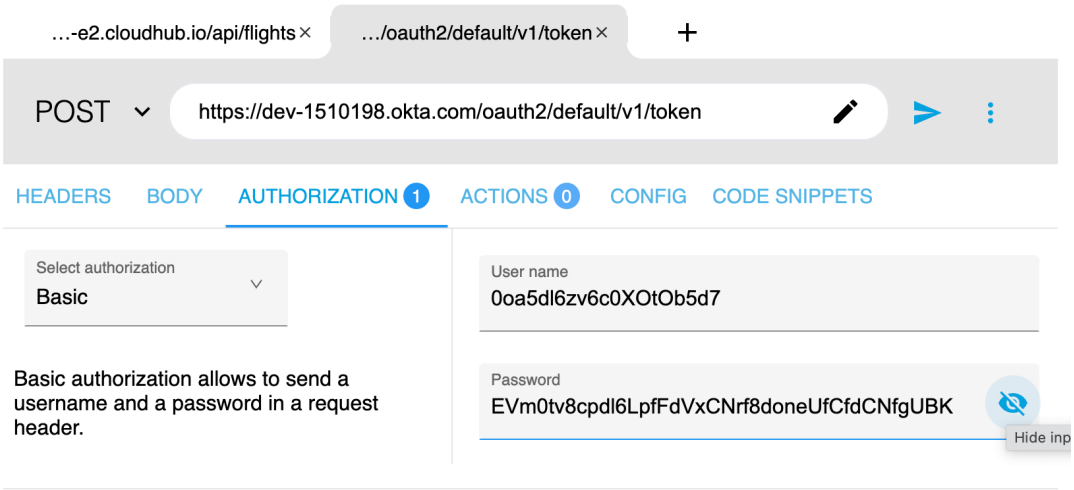


53. Switch to the Authorization tab.

54. From the Select Authorization drop down, select Basic Auth.



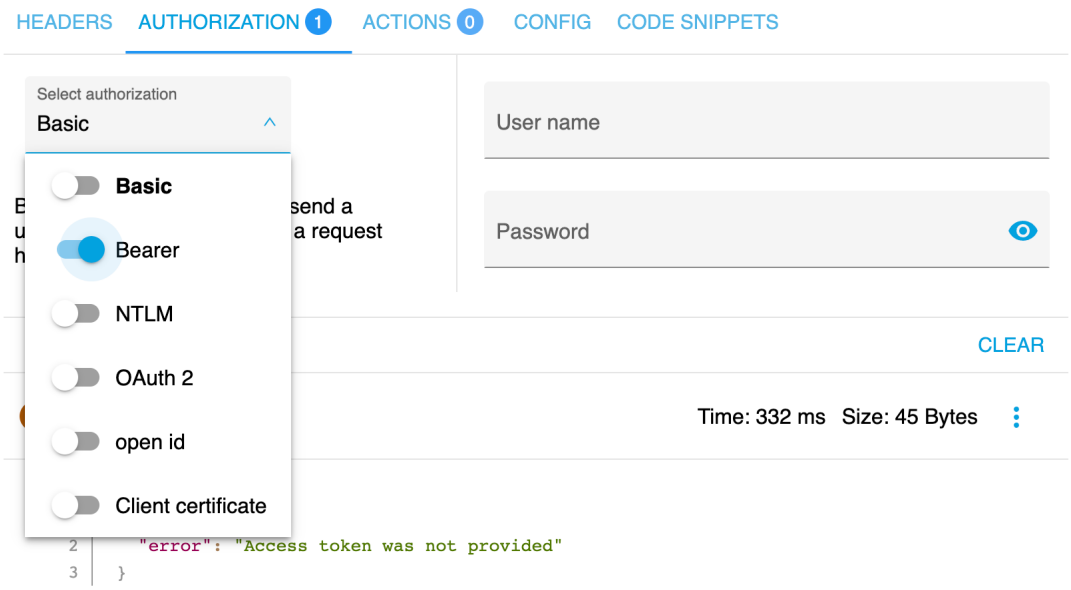
55. In the User name text field, enter the value of Client ID from snippets from the section Client application details.
56. In the Password text field, enter the value of Client Secret from snippets from the section Client application details.



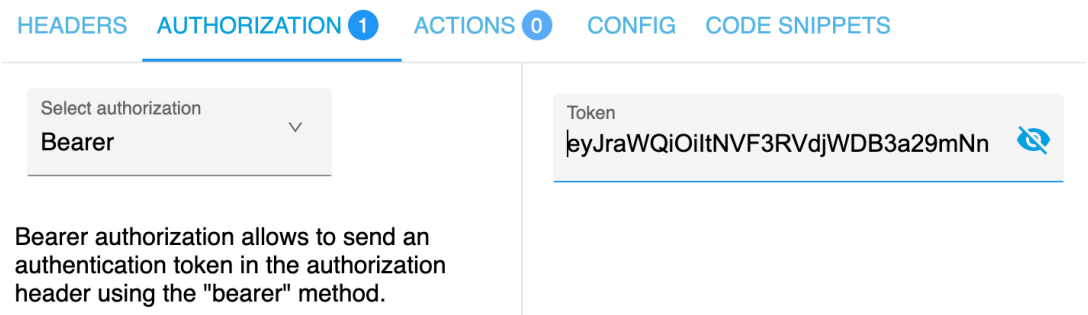
57. Click on Send.
58. Ensure you get a 200 OK response and the response body JSON contains a key value pair for access_token.

59. Copy the value of access_token and paste it in the snippets file.

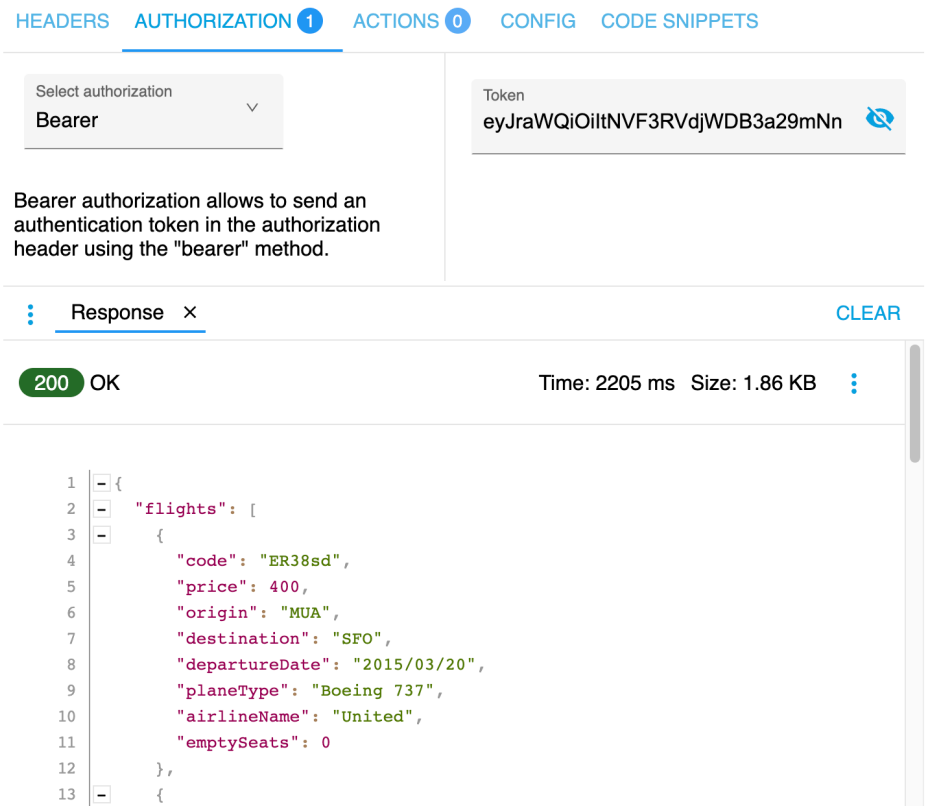
60. Navigate to the first request to United Flights API where we received a response of access token was not provided.
61. Navigate to the Authorization Tab.
62. From the Select authorization dropdown, select Bearer.



63. Paste the value of access_token from snippets file.



64. Send the request.
65. Verify you get a 200 OK response and the body of the response displays the flight information.



Module 6: Governing APIs with Policies and SLA tiers

At the end of this module, you should be able to:

- Distinguish between the types of API policies available in Anypoint API Manager.
- Enforce Service Level Agreement (SLA) tiers for APIs or groups of APIs.
- Apply the rate limiting - SLA-based policy to APIs.
- Apply the spike control policy to APIs.
- Define the order of execution of policies.

Walkthrough 6-1: Create, modify and publish an API group

In this walkthrough, you define an API group to bundle your APIs to facilitate group management. You then publish to Anypoint Exchange and request access. You will:

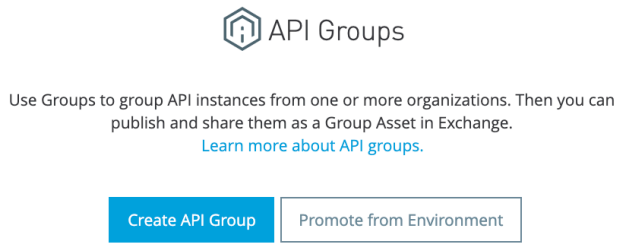
- Create a new API group within an environment.
- Add multiple API instances to an existing API group.
- Create an API group SLA tier.
- Apply a default SLA limit to an API group.
- Publish an API group to Anypoint Exchange.
- Request access to an API group.

Create a new API group

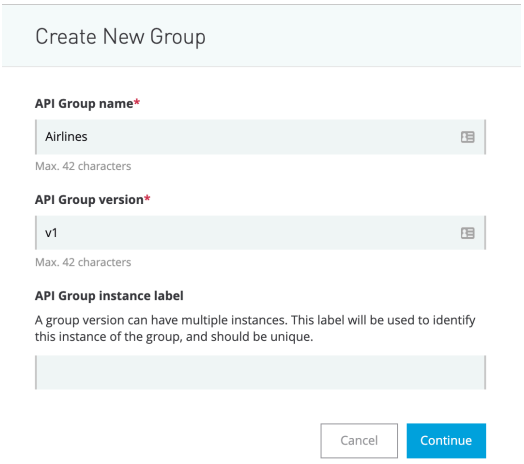
1. Return to API manager.
2. From the left menu select API Groups.

   Note: If a dialog introducing the features of API Groups appears, close it.

3. Click the Create API Group button.



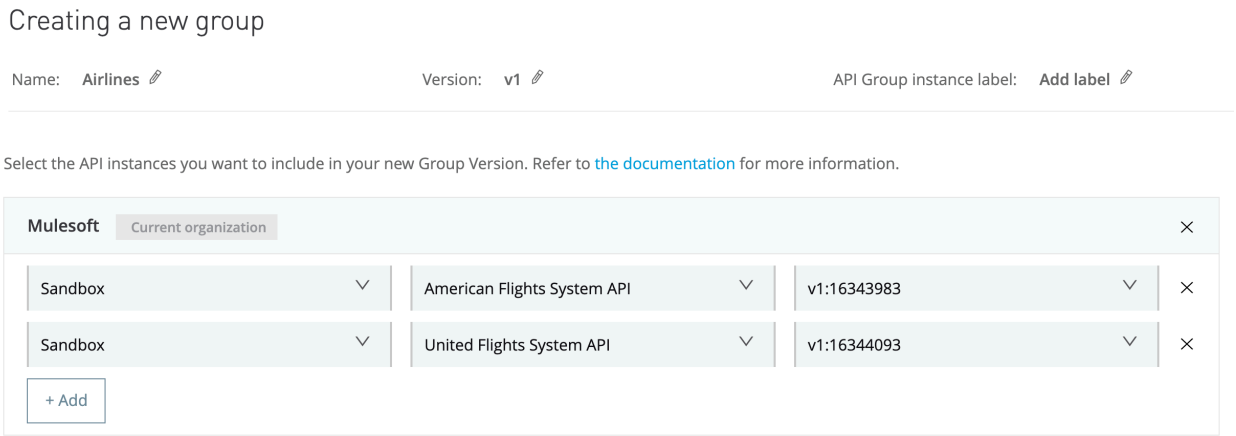
4. In the Create New Group dialog box, set the API Group name as Airlines.



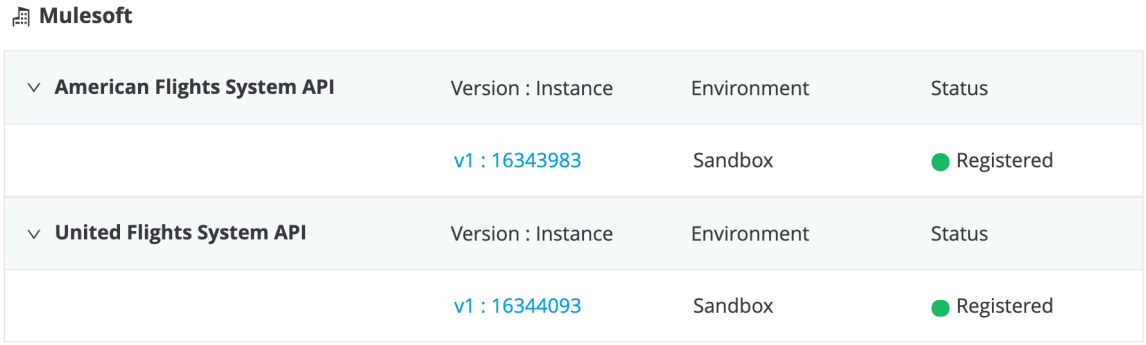
5. Click Continue.

Add existing API instances to the API group

6. On the Creating a new group page select Sandbox from the Select environment drop-down list.
7. From the Select an API list, select American Flights System API.
8. From the Select a version / instance select v1.
9. Click the +Add button.
10. From the Select environment drop-down list, select Sandbox.
11. From the Select an API list, select United Flights System API.
12. From the Select a version / instance select v1.

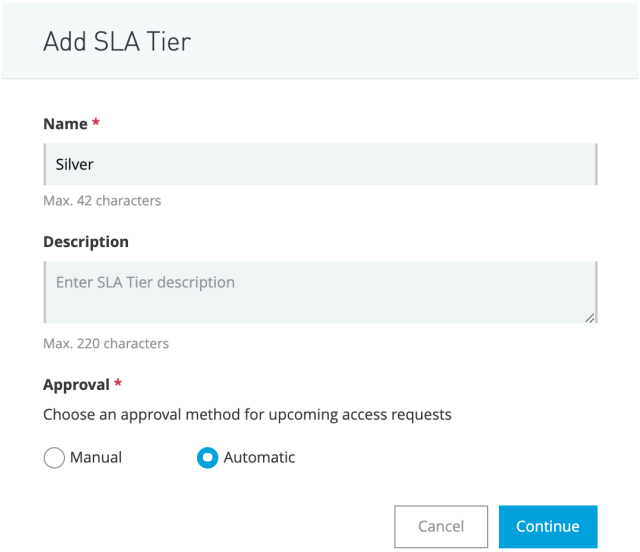


13. Click Save.



Create a group level SLA tier

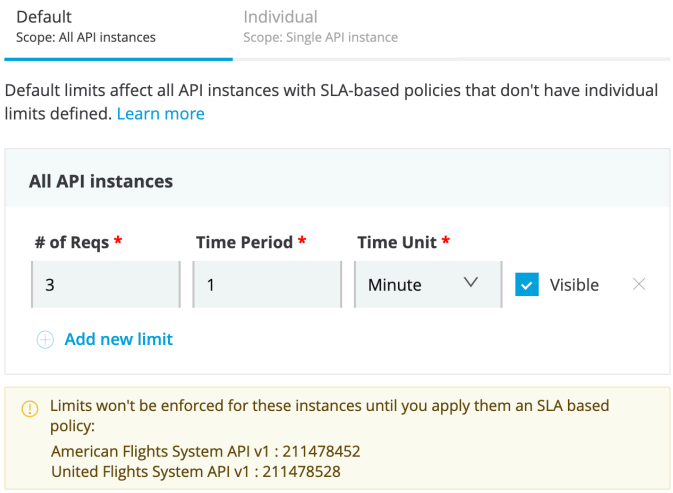
14. Select SLA Tiers from the left menu.
15. Click the Add SLA Tier button.
16. Set Name as Silver and select the Approval as Automatic.



17. Click Continue.
18. In the Adding limits to SLA tier screen under the Default scope tab enter:

- # of Reqs: 3
- Time Period: 1
- Time Unit: 1 minute
- Visible: selected

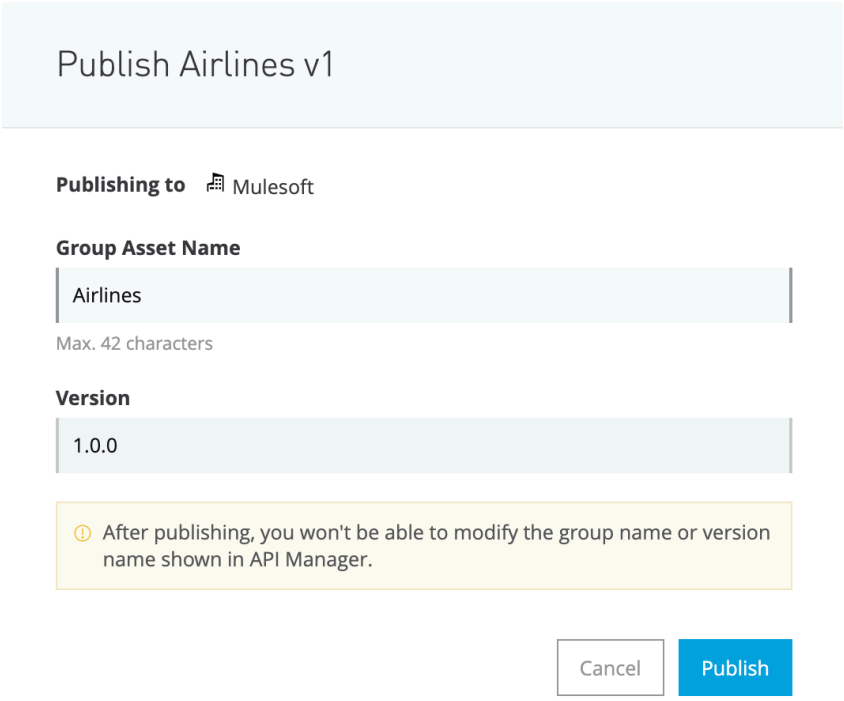
Note: Read the warning stating that limits won't be enforced for these instances until you apply an SLA based policy.



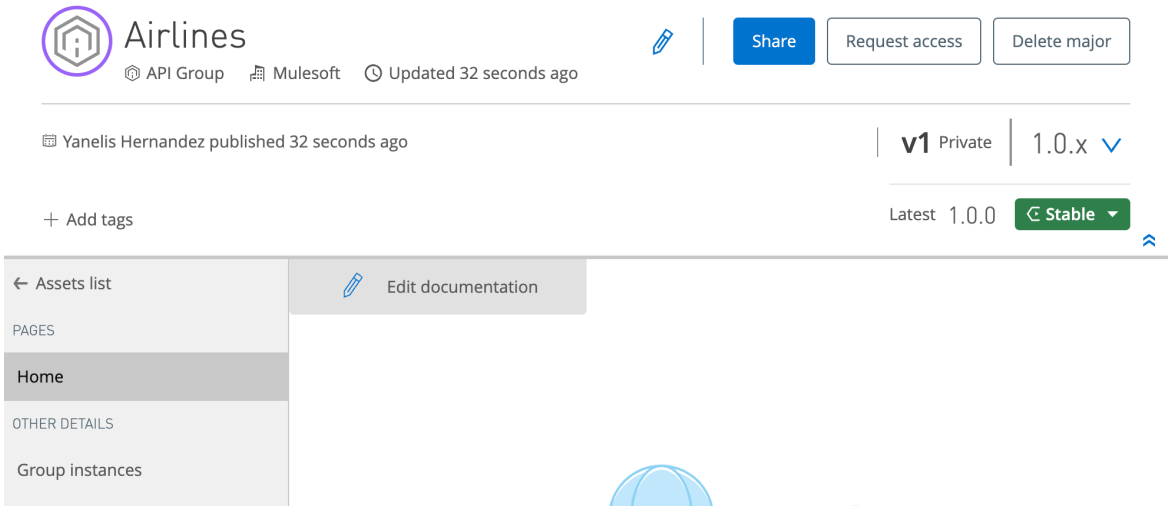
19. Click Save.

Publish the API group to Anypoint Exchange

20. Click Group Details from the left menu.
21. On the Group Details page, click publish to Exchange.
22. Leave the default values and click the Publish button.

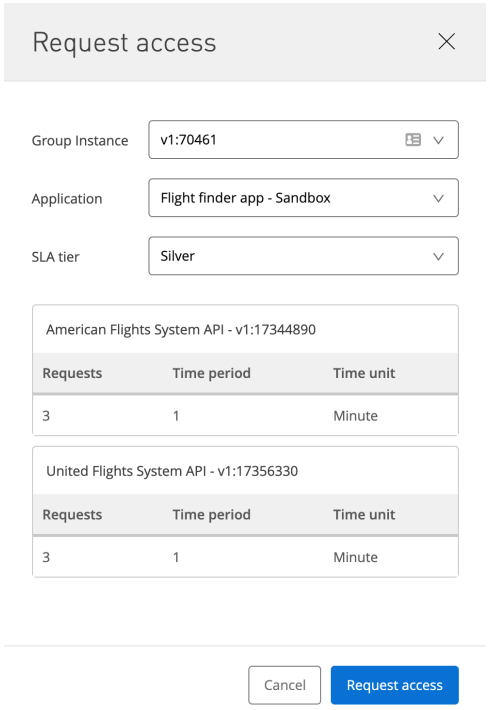


23. Click the Manage and share in exchange link.
24. Notice a new asset was created in Exchange of type API Group.



Request access to the entire API group

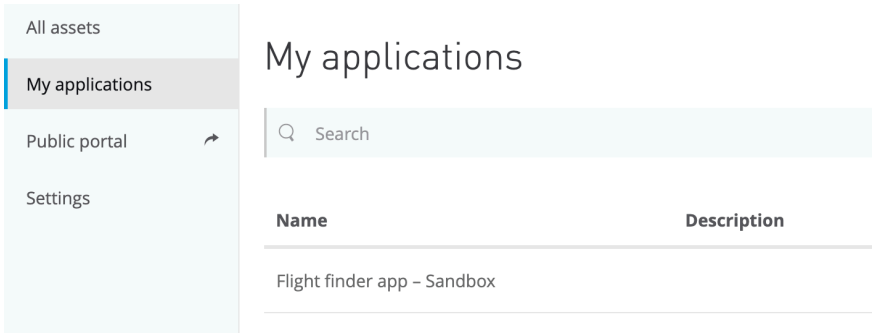
25. Press Request access button.
26. In the Request access dialog box, set the Group instance to v1.
27. In the Application drop down list, select Create new application.
28. Enter **Flight finder app – Sandbox** as the Application Name and press create.
29. Select SLA tier as Silver.



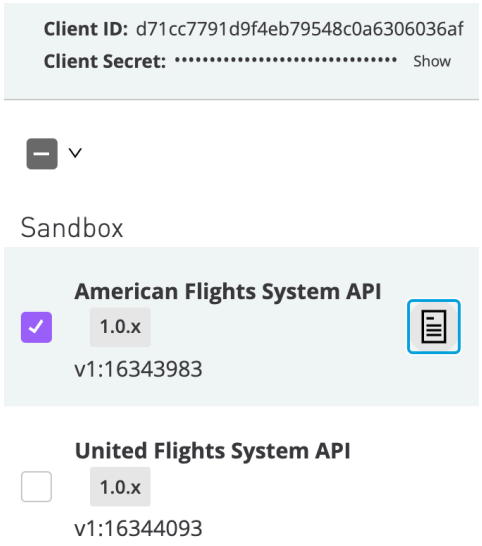
30. Click Request access.
31. Copy the Client ID and Client Secret from the confirmation dialog to your snippets file.

Verify the API contract in Exchange

32. In the Exchange portal left-side navigation, click Asset lists.
33. From the navigation menu select My applications.
34. In My applications, click Flight finder app - Sandbox.



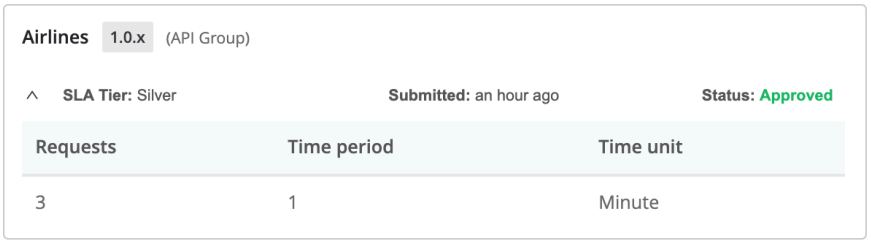
35. In the Flight finder app - Sandbox left-side navigation, click the American Flights System API checkbox.
36. Click the View SLA Tier icon to open Applied contract dialog.



37. Verify that you see the Group Silver SLA tier applied to the application.

American Flights API - v1:16311151

Applied contract



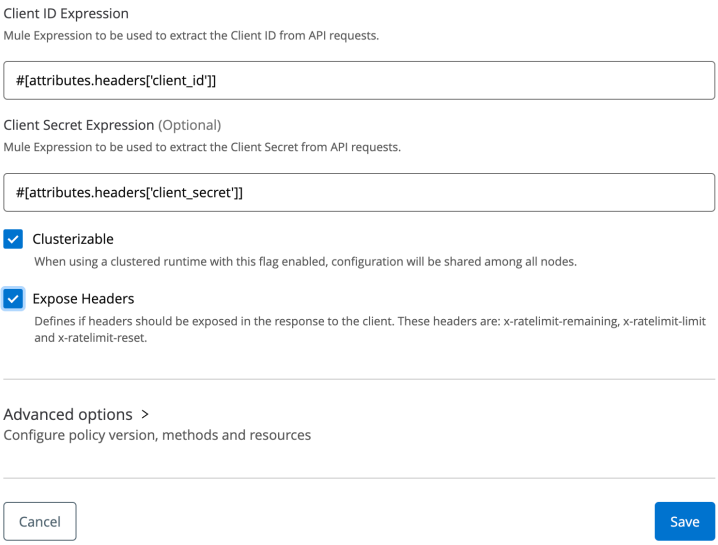
Walkthrough 6-2: Create and enforce SLA tiers through policies

In this walkthrough, you define SLA tiers to apply SLA-based policies to the API. You will:

- Apply an SLA-based rate limiting policy to enforce Group tiers.
- Create API instance SLA tiers for additional levels of throughput.
- Add multiple SLA limits to a tier.
- Update an API specification and its minor asset version number to support client ID enforcement in API policies.
- Apply an SLA tier to an API instance.

Apply an SLA-based rate limiting policy

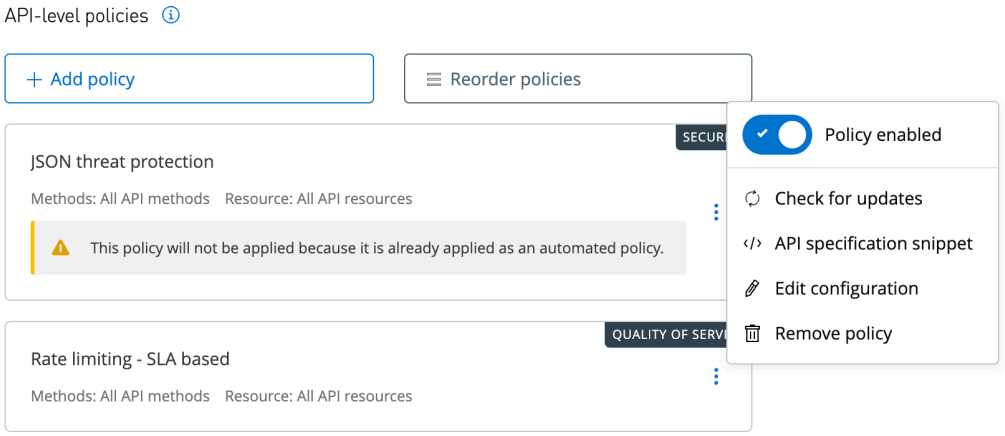
1. In API Manager navigate to the policies screen in the settings of American Flights System API.
2. Select +Add policy.
3. Under the Quality of Service, select Rate limiting – SLA based.
4. Click Next.
5. Read the default configuration settings and values.
6. Check the Expose Headers box.



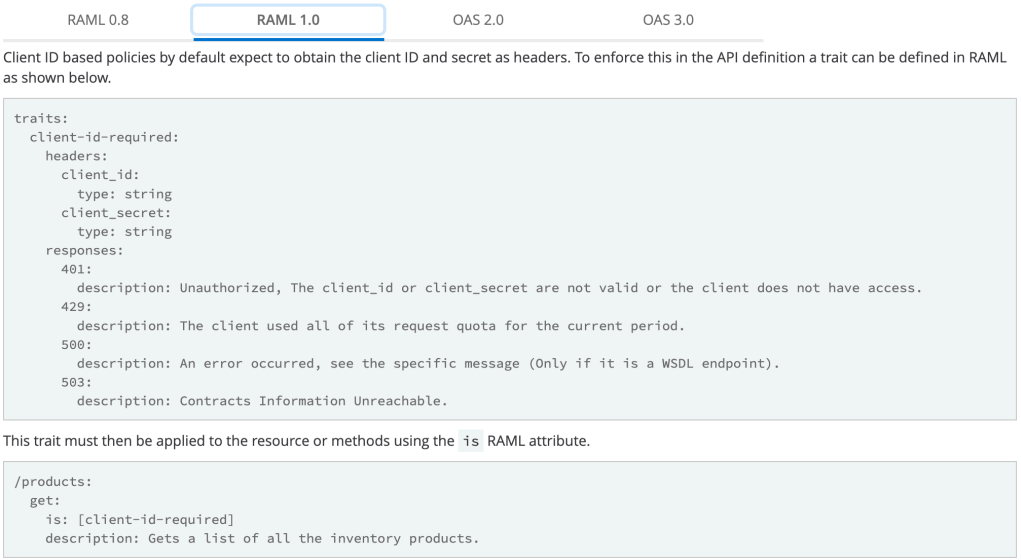
7. Click Apply.

Update API Specification to add client ID and secret enforcement

8. Locate the Rate limiting – SLA based policy under API – level policies and open the options menu.



9. Select API specification snippet.
10. Select the RAML 1.0 tab and copy the code the value for traits and click close.



11. Return to the browser tab with Design Center.
12. Open the project for American Flights System API.
13. Expand the arrow next to American Flights System API/master.
14. Type in v1 to create a new branch from master and click the plus sign.
15. Paste the traits copied in a new line after the types declaration.

American Flights System API/v1

```
#%RAML 1.0
title: American Flights System API
version: v1

types:
  AmericanFlight: !include datatypes/AmericanFlightDataType.raml

traits:
  client-id-required:
    headers:
      client_id:
        type: string
      client_secret:
        type: string
    responses:
      401:
        description: Unauthorized, The client_id or client_secret are not valid or the client does not have access.
      429:
        description: The client used all of it's request quota for the current period.
      500:
        description: An error ocurred, see the specific message (Only if it is a WSDL enpoint).
      503:
        description: Contracts Information Unreachable.

/flights:
  get:
    queryParameters:
      destination:
        required: false
```

16. Copy the trait enforcement from your snippets file on a new line after the /flights resource.

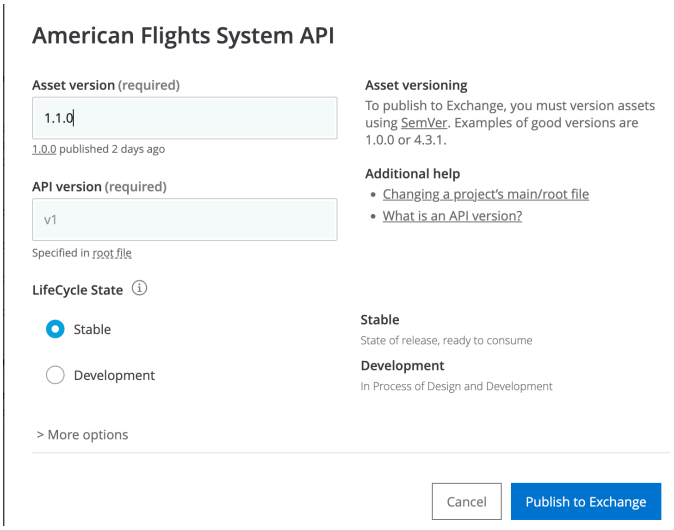
```
    503:
      description: Contracts Information Unreachable.

/flights:
  is: [client-id-required]
  get:
    queryParameters:
      destination:
```

Publish the updated API specification to the public Exchange portal

17. Click Publish then Publish to Exchange.

18. In the American Flights System API dialog box, change the Asset version to 1.1.0.

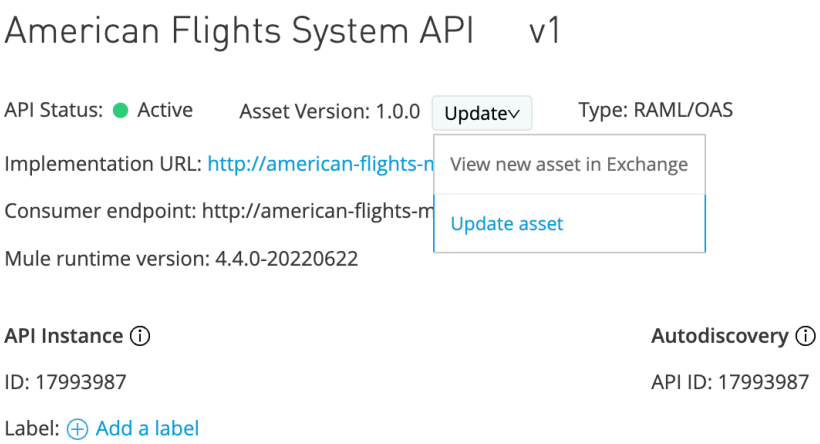


Note: Best practice is to increment the major version of the API so the new semantic version of the API should be 2.0.0. This is because we are adding more security which is not backward compatible. In order to keep the course simple, we set it to 1.1.0.

19. Wait for the success message to print in a dialog box.

Update the version of the API configured for the API instance in API Manager

20. Return to API Manager and navigate to the American Flights System API v1 settings page.
21. Next to the Asset Version, select the Update drop down list and select Update asset.



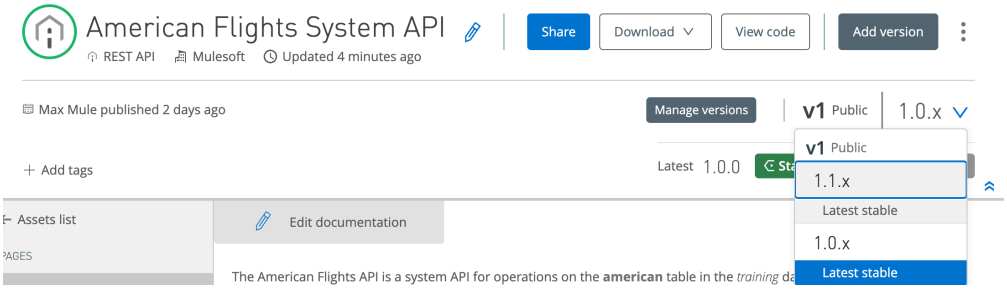
Note: If not visible then refresh the page.

22. In the Update asset version dialog, change the Asset Version to 1.1.0.

23. Click Change.

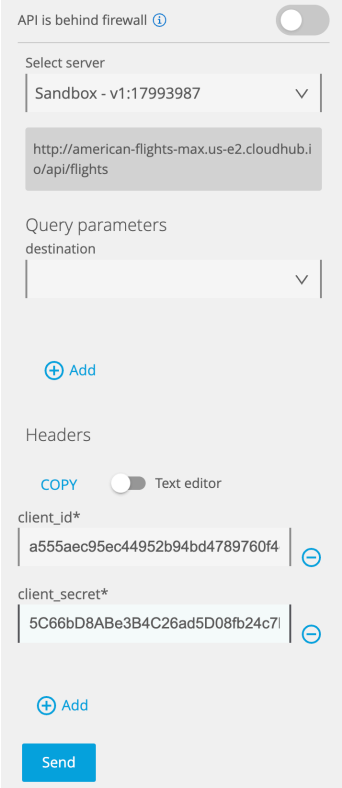
Test the rate limiting behavior

24. Return to the Exchange portal and refresh the page.
25. Select the newest version of the API, v1.1.x – latest stable.



26. Navigate to the GET method under /flights.
27. Select Sandbox – v1 server.

28. Fill in the client_id and client_secret from the snippets file.



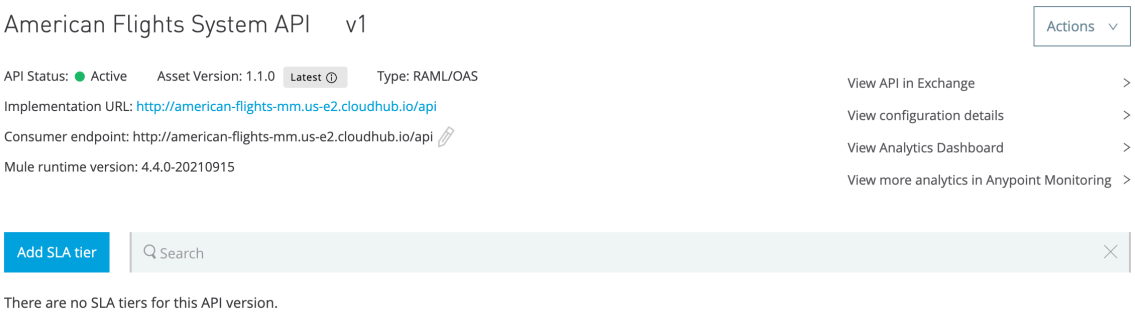
29. Click send and make sure you get a 200 OK response.
30. Repeat the request three or more times quickly.
31. Confirm that you get a 429 Too Many Requests.

```
{
  "error": "Quota has been exceeded"
}
```

Create additional SLA tiers for an API Instance

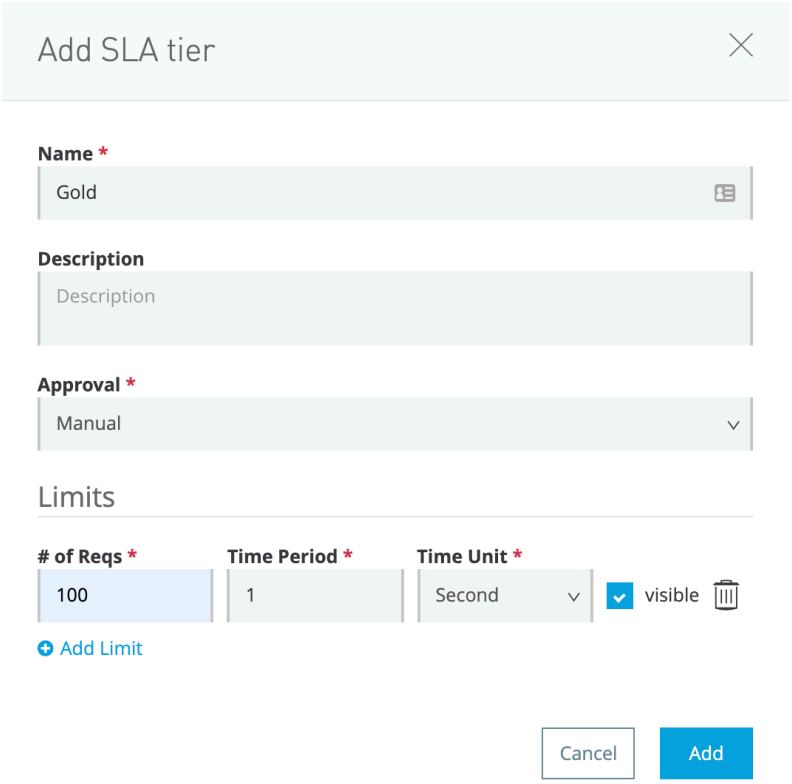
32. Return to API Manager.
33. Navigate to the American Flights System API (v1) – Settings page inside API Manager (Sandbox environment).
34. In the left-side navigation, click SLA Tiers.

35. Click the Add SLA tier button.



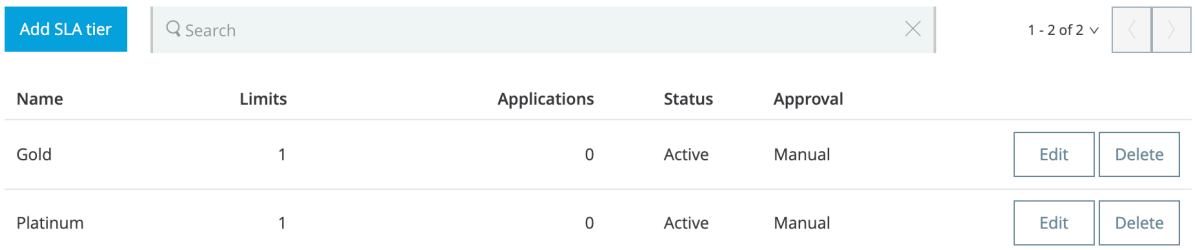
36. In the dialog box, fill in the following information:

- Name: Gold
- Approval: Manual
- # of Reqs: 100
- Time Period: 1
- Time Unit: Second
- Leave the visible checkbox checked.



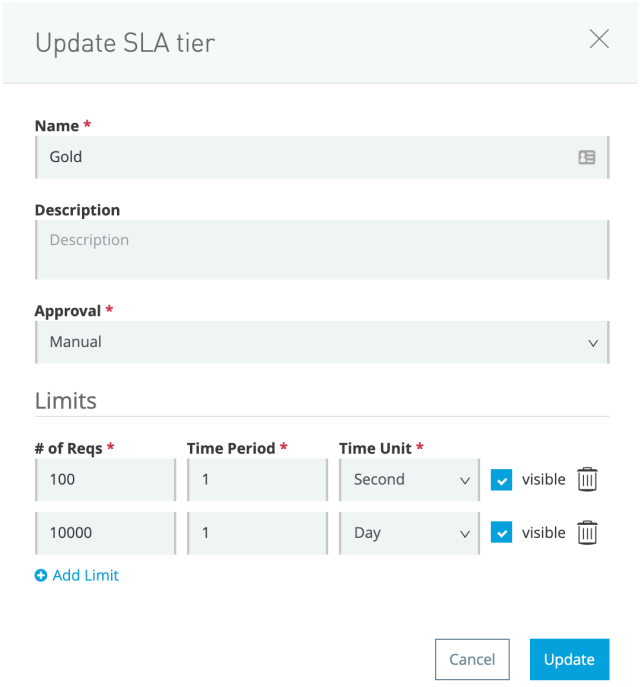
37. Click Add.
38. Similarly add a Platinum SLA tier with the following information:

- Name: Platinum
- Approval: Manual
- # of Reqs: 1000
- Time Period: 1
- Time Unit: Second
- visible: checked



39. Click the Edit button in Gold tier row to add another limit.
40. In the Update SLA tier dialog click the Add Limit link and fill in the following information:

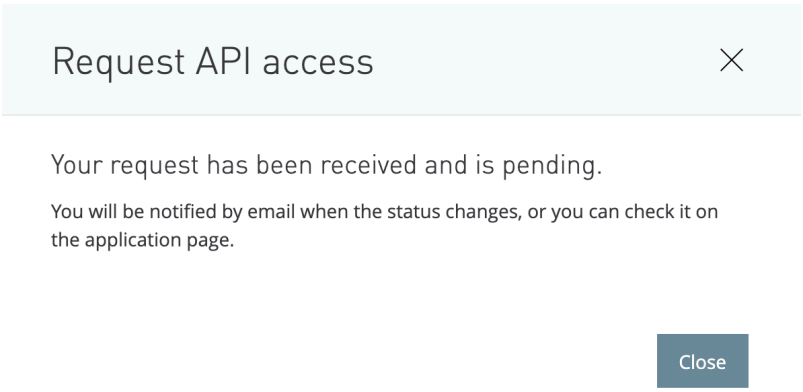
- # of Reqs: 10000
- Time period: 1
- Per: Day



41. Click Update.

Request Access to a single API instance

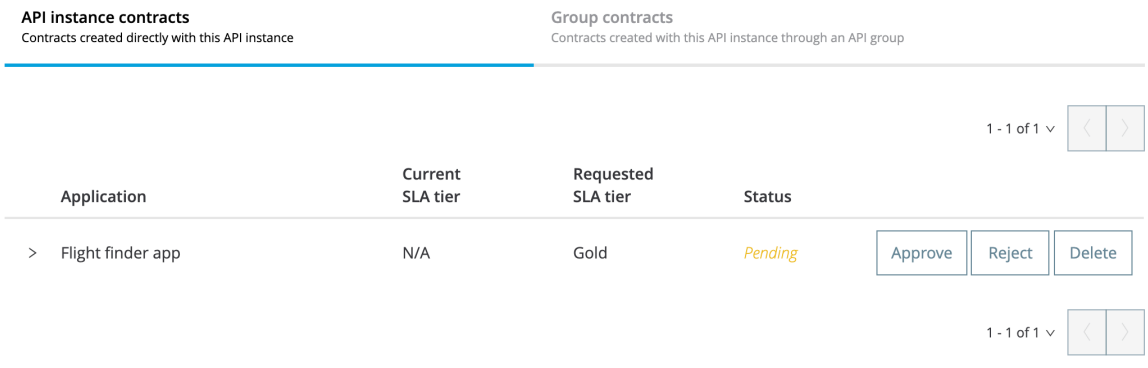
42. Select View API in to Exchange.
43. Select API version 1.1.x – latest stable.
44. Click the Request access button.
45. In the Request access dialog select:
    - API Instance: Sandox v1
    - Application: Flight Finder App - Sandbox
    - SLA Tier: Gold
46. Click Request access.



Note: The Gold tier approval is manual, therefore will remain pending until approved.

Approve an individual SLA Tier for an API instance to override the group SLA tier

47. Return to the API Manager settings page for American Flights System API v1.
48. Select Contracts from the left menu.
49. Under the API instance contracts, click **Approve** for the Gold tier.



50. Return to Anypoint Exchange.

51. Repeat a few GET request requests in version 1.1.x of the American Flights System API.
52. Confirm that there is no 429 too many requests error.

Walkthrough 6-3: Apply a spike control policy to limit requests from all API clients

In this walkthrough, you apply a Spike Control policy to an API proxy to limit the number of messages allowed through the API policy enforcement endpoint and forwarded to the API implementation endpoint. You then test the policy by sending too many API requests to the API proxy then verify the Spike Control policy buffers and retries the API requests. You will:

- Apply a Spike Control policy to an API.
- Specify the total number of messages to be processed in a time period from all API clients.
- Test an API policy enforcement endpoint governed by a Spike Control policy.

Configure and apply a spike control policy

1. Return to the API settings page for American Flights System API v1.
2. Select Policies from the left menu.
3. Click +Add policy.
4. Under Quality of service, select the Spike Control policy.
5. Click Next.
6. Read the description for each of the Spike Control policy properties and set the values to:

    - **Number of Reqs**: 1
    - **Time period**: 5000
    - **Delay time in milliseconds**: 1000 ms
    - **Delay attempts**: 2
    - **Queueing limit**: 5
    - **Expose Headers**: Selected

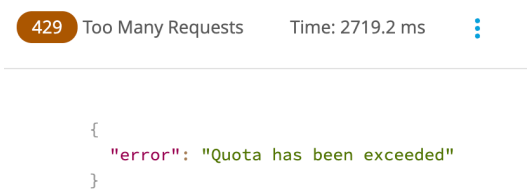
    Note: This configuration allows one request per 5 seconds to be processed by the API backend. It also allows 5 requests to be queued at the same time and the request retry will take place after 1 second. Each queued up request gets two attempts to retry the request before the API drops them.

7. Click Apply.

Test the managed API in Anypoint Exchange

8. Click the View API in Exchange Link.
9. Make sure version 1.1.x is selected; otherwise select it.

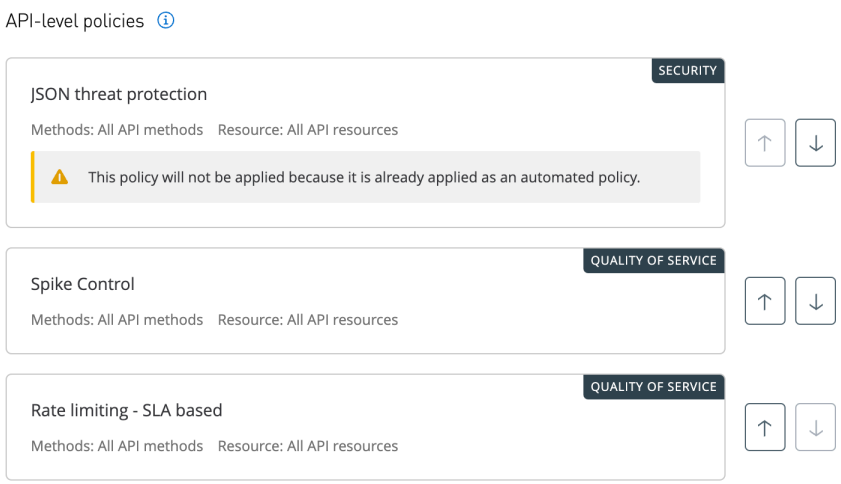
10. In the left-side navigation, click the /flights resource GET method.
11. In the API Console, click on the Mocking Service option and change it to Sandbox v1.
12. Fill in the client_id and client_secret from the snippets file.
13. Click Send.
14. Verify you get a status 200 OK and you can see the JSON response.
15. Click the Send button one more time.
16. Verify you get a status 429 when you click GET for the last time to show that the request limit has exceeded.



Change the order of execution of applied policies

17. Switch the API Manager American Flights System API (V1) settings page.
18. Select policies from the left menu.
19. Under API-level policies, click Reorder policies

20. Click the up arrow next to the row containing the Spike Control policy and to move it above Rate Limiting – SLA based.
21. Click Apply Order.



22. Observe that the order of Applied policies changed.

Module 7: Versioning Managed APIs

At the end of this module, you should be able to:

- Version APIs and API groups in Anypoint Design Center.
- Document changes in API versions inside portals.
- Deprecate old versions of APIs and API groups.

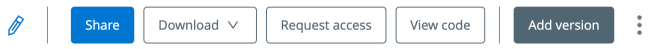
Walkthrough 7-1: Add a new major API version from API Manager

In this walkthrough, you create a new version of the Flights API. You will:

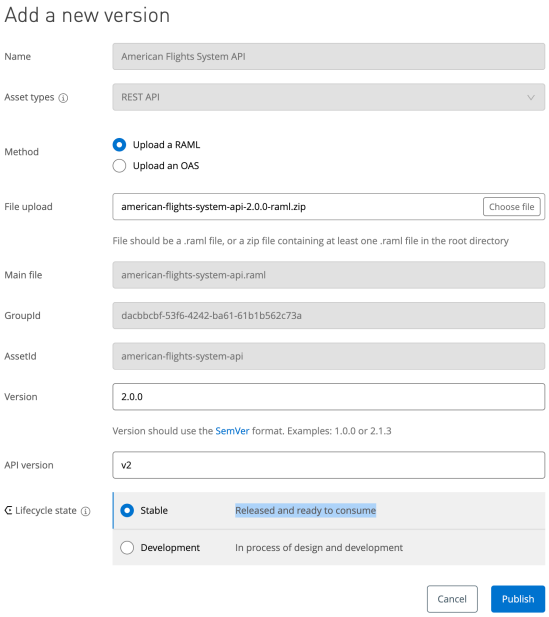
- Publish a new major version of an API from API Manager.
- Configure a proxy endpoint to use the new major API version.
- Deploy an API proxy that uses a new major API version.

Create a new API instance in API Manager from a RAML definition

1. Navigate to Exchange.
2. From the list of assets in your organization select American Flight System API.
3. Click Add version.



4. In the Add a new version screen set the following values:
    - **Method**: Upload a RAML
    - **File upload**: choose file american-flights-system-api-2.0.0-raml.zip located in the student files folder
    - **Version**: 2.0.0
    - **API version**: v2



5. Click Publish.
6. Navigate to API Manager.
7. Click on Add API and select Add new API.
8. In the Runtime screen, select Mule Gateway as the Runtime.
9. Select Deploy a proxy application as a Proxy type.
10. For the Proxy app name type american-flights-system-api-v2-{initials}.
11. Click Next.
12. In the API screen select American Flights System API.
13. Make sure the API version is v2 and Asset version is 2.0.0.
14. Click Next.
15. In the Endpoint screen set the implementation URI to the American External API Implementation from the snippets file.

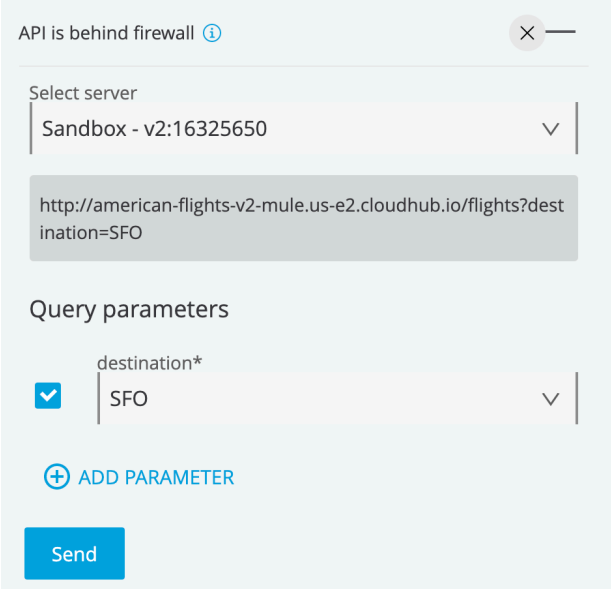
```
http://training4-american-ws.cloudhub.io/api/
```

16. Click Next.
17. Validate all values are correct in the Review screen and click Save & Deploy.
18. Verify that the deployment is successful.
19. Click Add consumer endpoint and set it to the proxy URL generated in the American Flights System API (v2) – Settings page.

Test the new version of the managed API

20. In the top-right corner of the American Flights System API (v2) – Settings page, click the View API in Exchange link.
21. In the left navigation of the American Flights System API v2 asset, click the /flights GET method. In the API Console on the right, click Mocking Service and select Sandbox – v2 from the drop-down list.

22. Click Send.



23. Verify that you get a 200 OK response back with an array of American Flight objects.

24. Select different query parameters values and test the API a few more times.

    Note: This step helps populate some Analytics data for this API.

Walkthrough 7-2: Deprecate an old version of an API

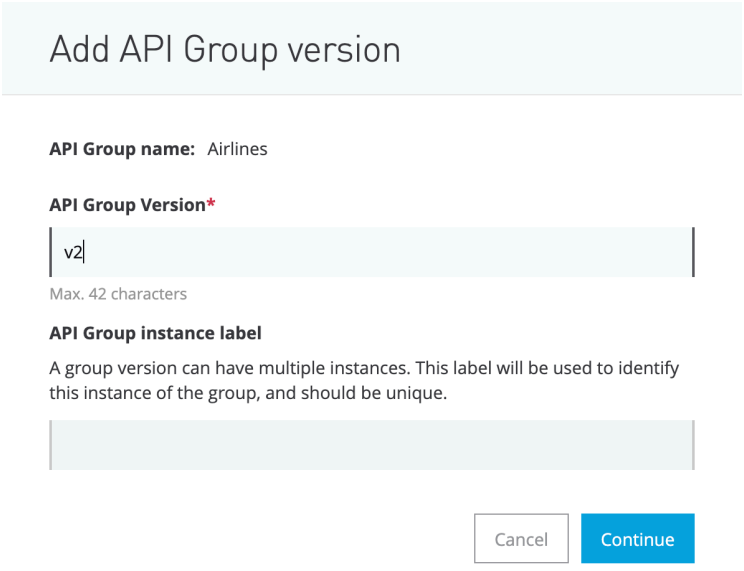
In this walkthrough, you set the older version of an API as deprecated. You will:

- Create a new group instance to remove an API version.
- Deprecate an API group version.
- Deprecate an old version of an API in Anypoint Exchange.
- Deprecate an API instance through API Manager.

Create a new API group version to update an API instance version

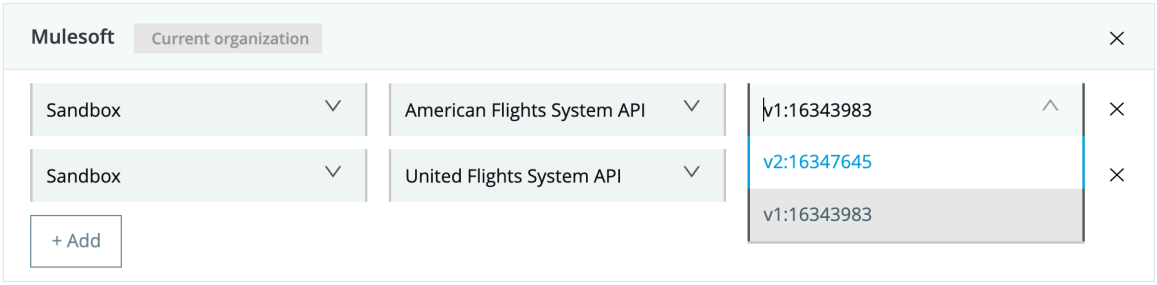
1. Return to API Manager and confirm you are in the Sandbox environment.
2. Select API Groups from the left menu.
3. Click Add group version button next to the Airlines Group.

4. In the Add API Group version dialog set the API Group Version as v2.

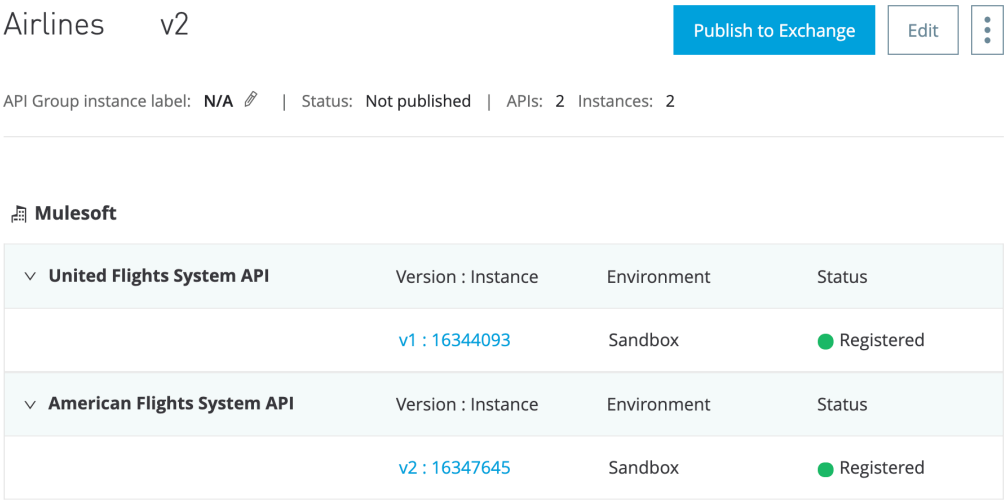


5. Click Continue.

6. In the Creating a new group version page, locate the American Flights System API v1 dropdown and change the value to v2.



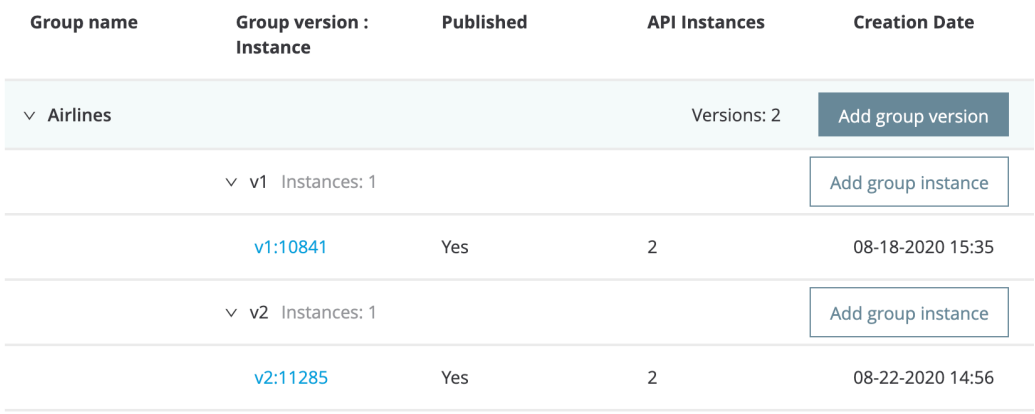
7. Click Save.



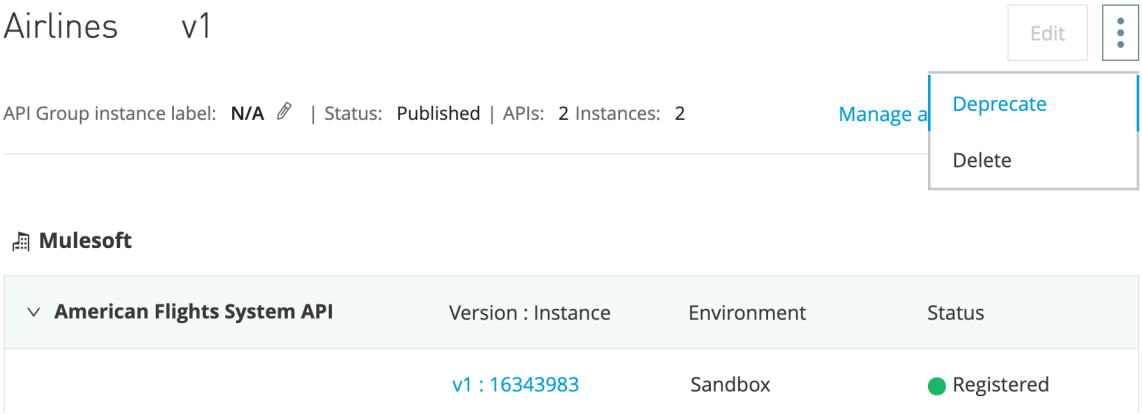
8. Click Publish to Exchange.
9. In the Publish Airlines v2 dialog click Publish and leave the version as default 2.0.0.

Deprecate an API group version

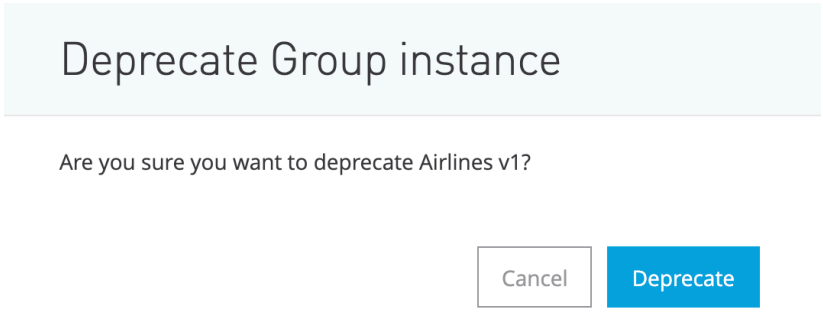
10. From the left menu select API Groups.



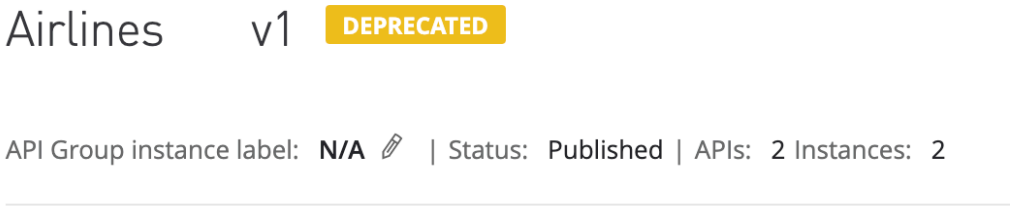
11. Select the Airlines v1 instance.
12. Click the three dots more options menu and select deprecate.



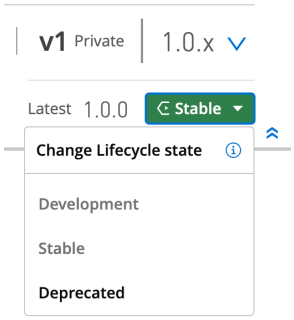
13. In the dialog Confirm deprecate group, click Deprecate.



14. Notice the deprecated badge next to Airlines v1.



15. Click Manage and share in exchange.
16. Click the Stable badge to change the Lifecyle state.



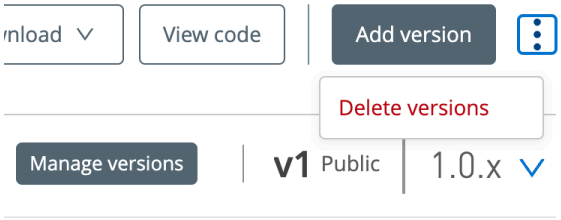
17. Verify that the group version shows as deprecated.

Deprecate an old version of an API in Anypoint Exchange

18. Return to the assets list in Exchange and select American Flight Systems API.
19. Click 2.0.x next to v2 of the asset and select v1.0.x.
20. Click on the Stable badge to update the lifecycle state to Deprecated.

Delete an old API major version in Anypoint Exchange

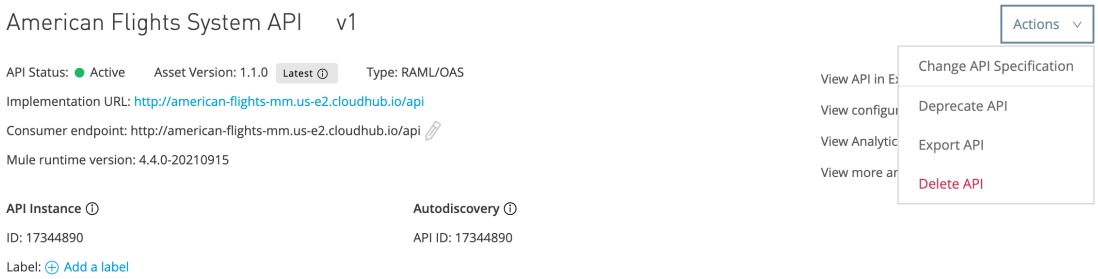
21. For asset version 1.0.x, click the three vertical dots icon and select Delete versions.



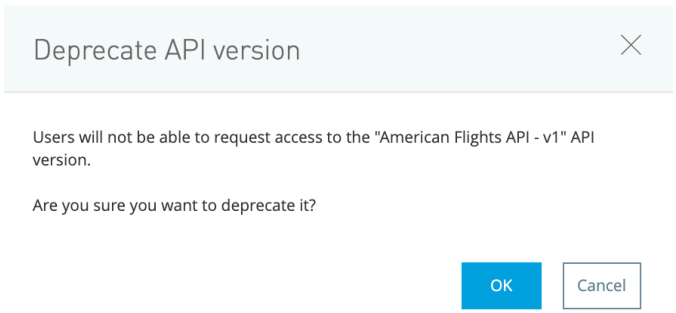
22. In the Delete patch version confirmation screen click the delete icon.
23. Confirm and Delete version.

Deprecate an API instance on API manager

24. Return to the API settings page for American Flights System API v1.
25. Under actions select Deprecate API.



26. In the Deprecate API version confirmation dialog box click OK.



27. Verify the deprecated badge shows next to the actions.

Note: If the badge doesn't show immediately, then refresh the browser page.

Module 8: Monitoring APIs

At the end of this module, you should be able to:

- Enable Anypoint Monitoring for applications and APIs.
- Monitor APIs using the built-in dashboards in Anypoint Monitoring.
- Create custom dashboard and charts with Anypoint Monitoring.
- Create API alert notifications using Anypoint API Analytics.
- Create and run custom reports from Anypoint API Analytics.
- Enable API analytics in third-party software.

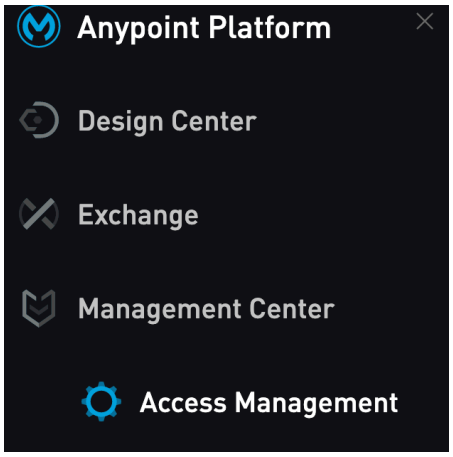
Walkthrough 8-1: Explore Anypoint Monitoring built-in dashboards

In this walkthrough, you explore the built-in dashboards provided by Anypoint Monitoring. You will:

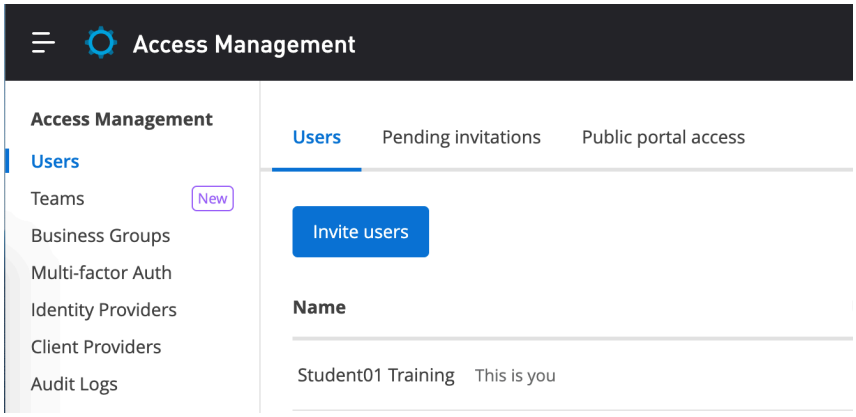
- Set required permissions to use Anypoint Monitoring.
- Enable Anypoint Monitoring for applications on CloudHub.
- Navigate to the Overview dashboard in Anypoint Monitoring.
- Discover other built-in dashboards available with advanced metrics.

Set Anypoint Monitoring User permission

1. In the main menu select Access Management.

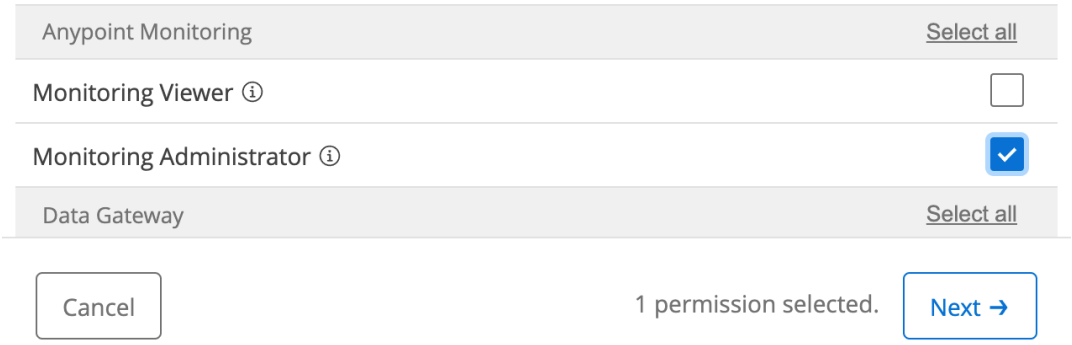


2. From the Access Management menu select **Users**.



3. Click on your Username for the admin user or the user that you have been using during the class.
4. On the Permissions tab, click **Add permissions**.

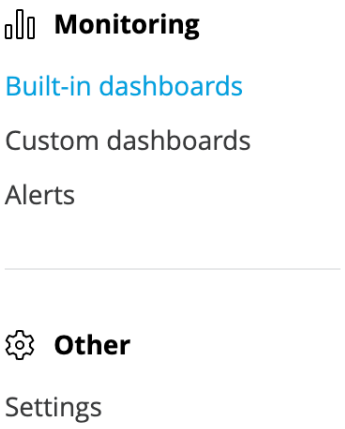
5. On the Select Permissions dialog, scroll down to the Anypoint Monitoring section and select the **Monitoring Administrator** permission.



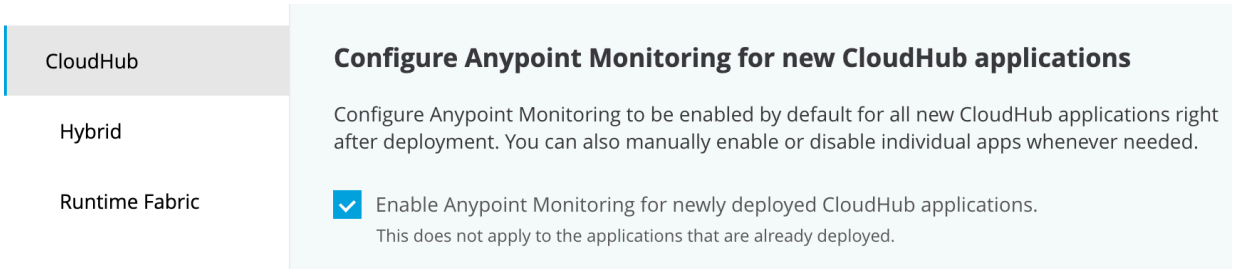
6. Click **Next**.
7. In the Select Business Groups dialog select the name of your business group.
8. Click **Next**.
9. Hover over Anypoint Monitoring permission to verify that **Monitoring Administrator** is listed.

Enable Anypoint Monitoring for current and new applications

10. Navigate to Anypoint Monitoring.
11. From the left menu click Settings.

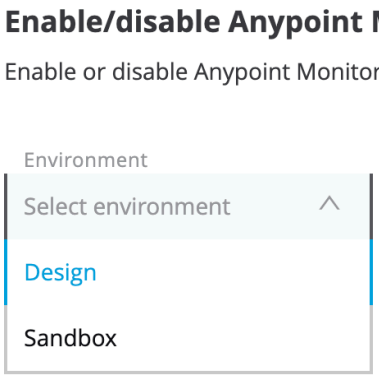


12. Under the Cloudhub tab select Enable Anypoint Monitoring for newly deployed CloudHub applications.

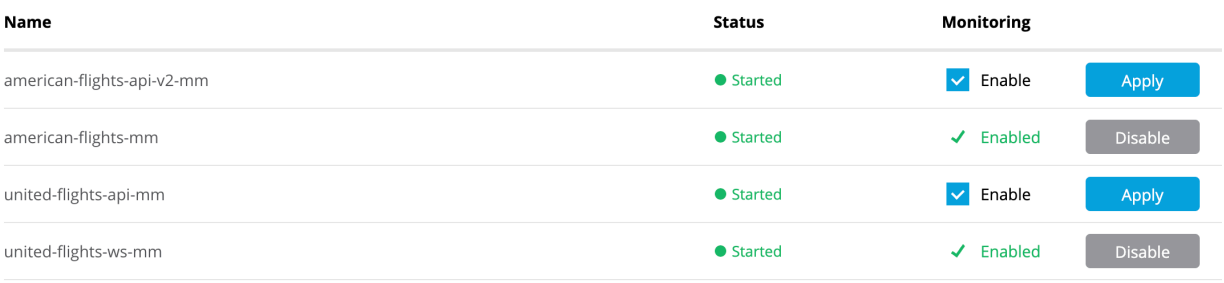


Note: This will not enable monitoring for the applications already deployed.

13. Select the Sandbox environment from the Select Environment dropdown list.

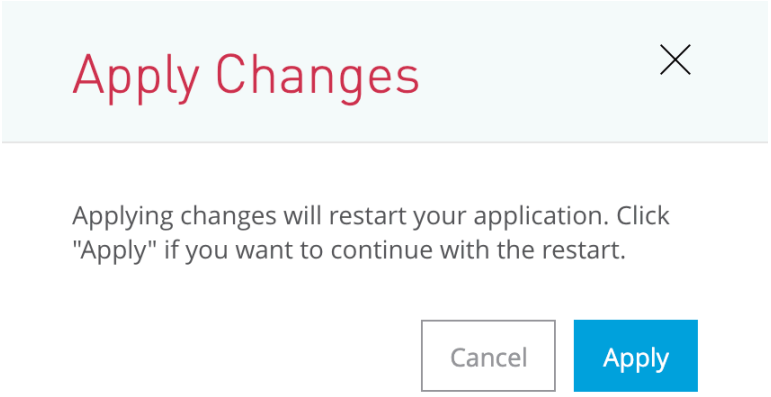


14. Under List of resources in Sandbox check Enable for all applications.



15. Click Apply for each of the applications to enable monitoring.

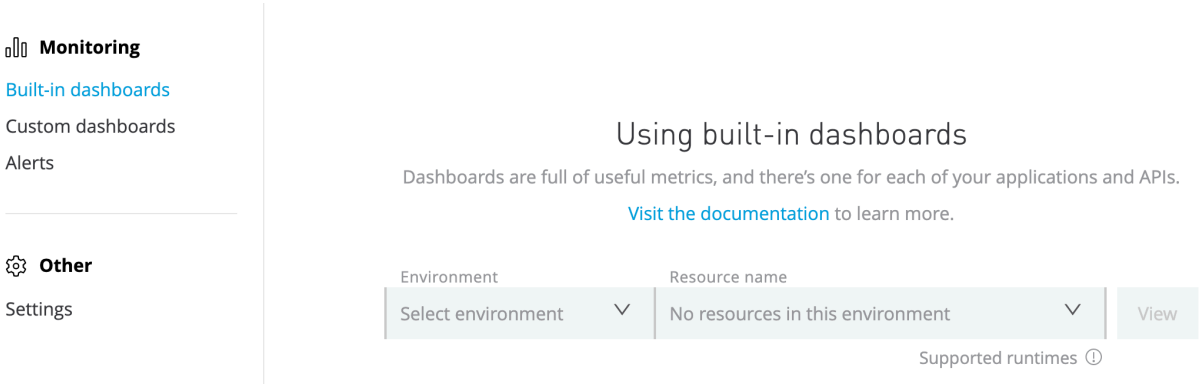
16. In the Apply Changes confirmation dialog, click Apply.



Note: This will restart the application and it will take some time to be available for Monitoring.

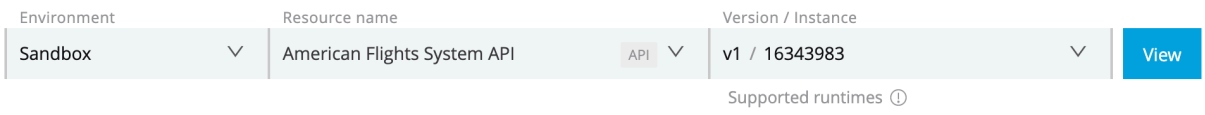
Explore the Overview built-in dashboard

17. From the left menu select Built-in dashboards
18. Notice the Built-in dashboards as the landing page.



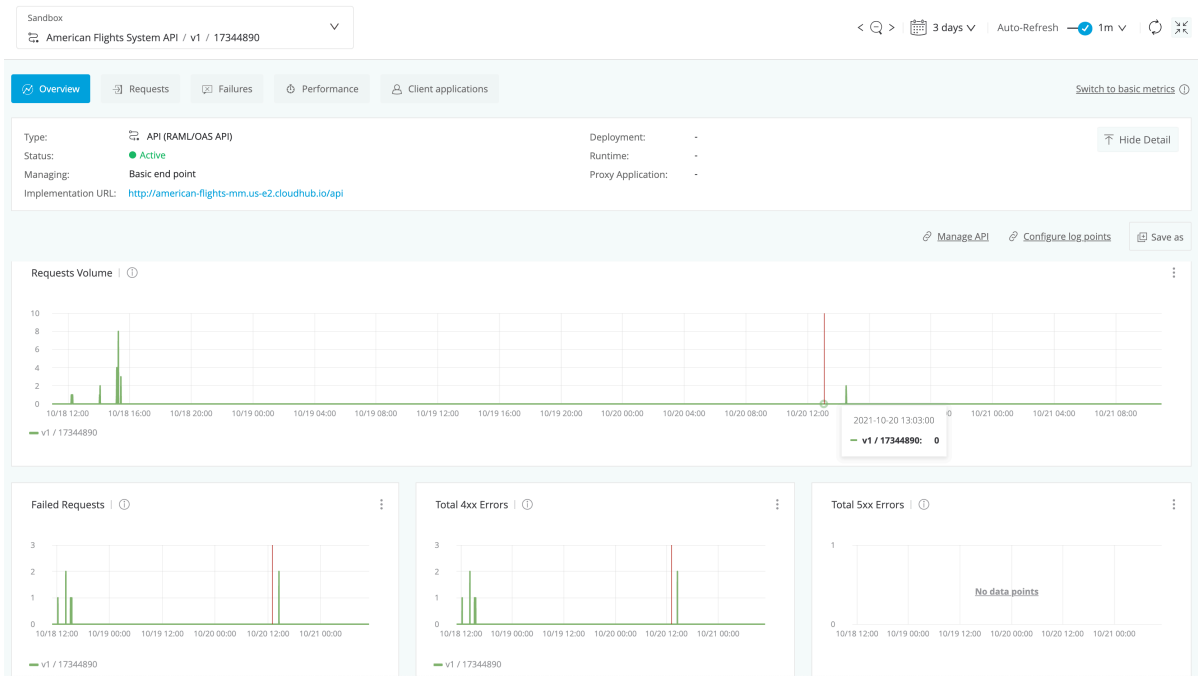
19. Select the following values from the drop downs:

- Environment: Sandbox
- Resource name: American Flights System API
- Version/Instance: v1



20. Click the View button.
21. Examine the built-in Overview dashboard.

Note: If your dashboard is already showing some data points change the interval in the upper corner to 15 minutes.



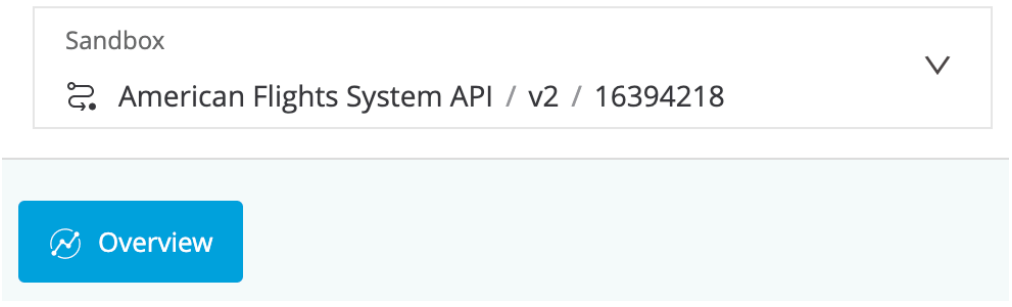
Explore other built-in dashboards available for APIs using enhanced metrics

22. Return to Anypoint Exchange to make a few more requests GET and POST requests to American Flights System API v2.

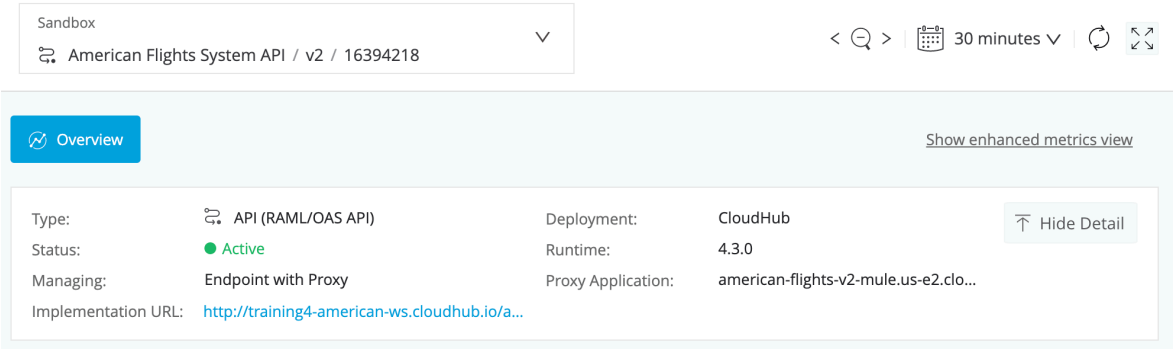
    Note: This will now record advanced metrics since monitoring is now enabled for all APIs.

23. Return to Anypoint Monitoring Built-in dashboards.
24. In the dashboard configuration section above change the version to v2.

    - Environment: Sandbox
    - Resource name: American Flights System API
    - Version/Instance: v2

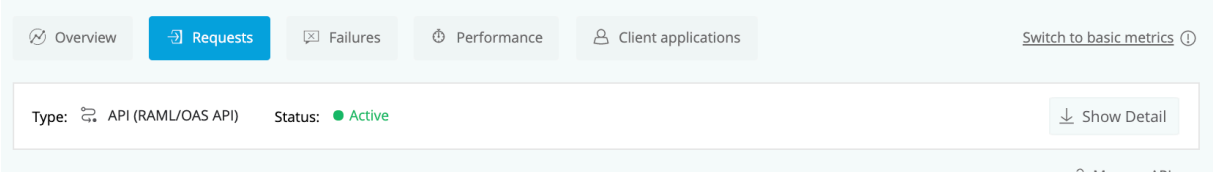


25. In the upper right corner of the dashboard click the link to Show enhanced metrics view.



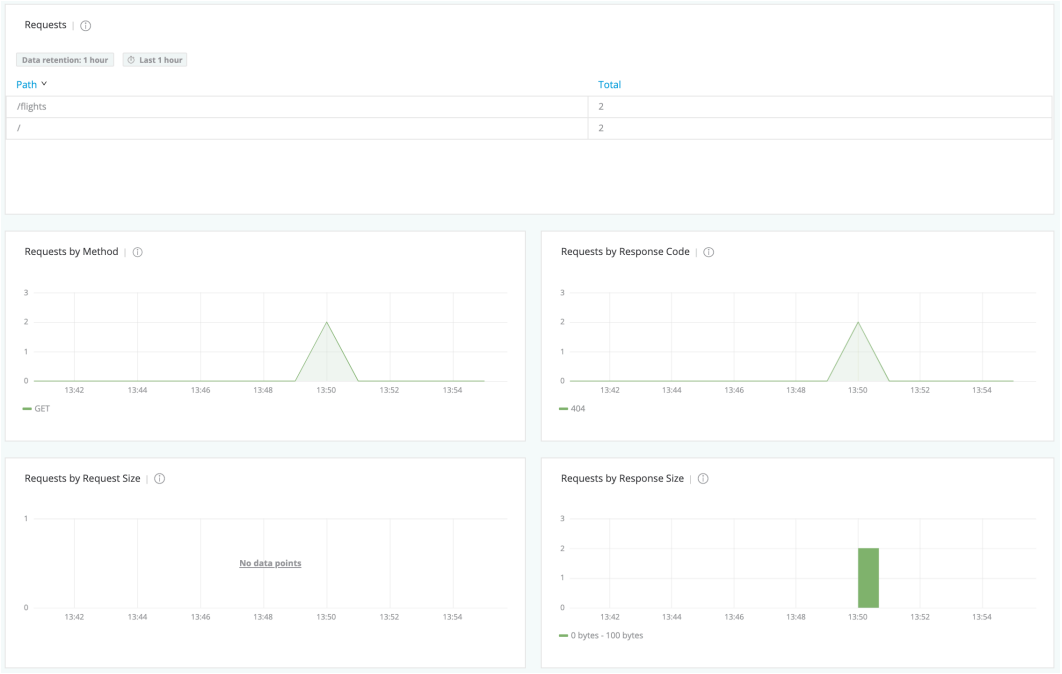
Note: If instead it shows switch to basic metrics, please ignore this step.

26. Notice the top bar shows four additional buttons to toggle between different dashboards.

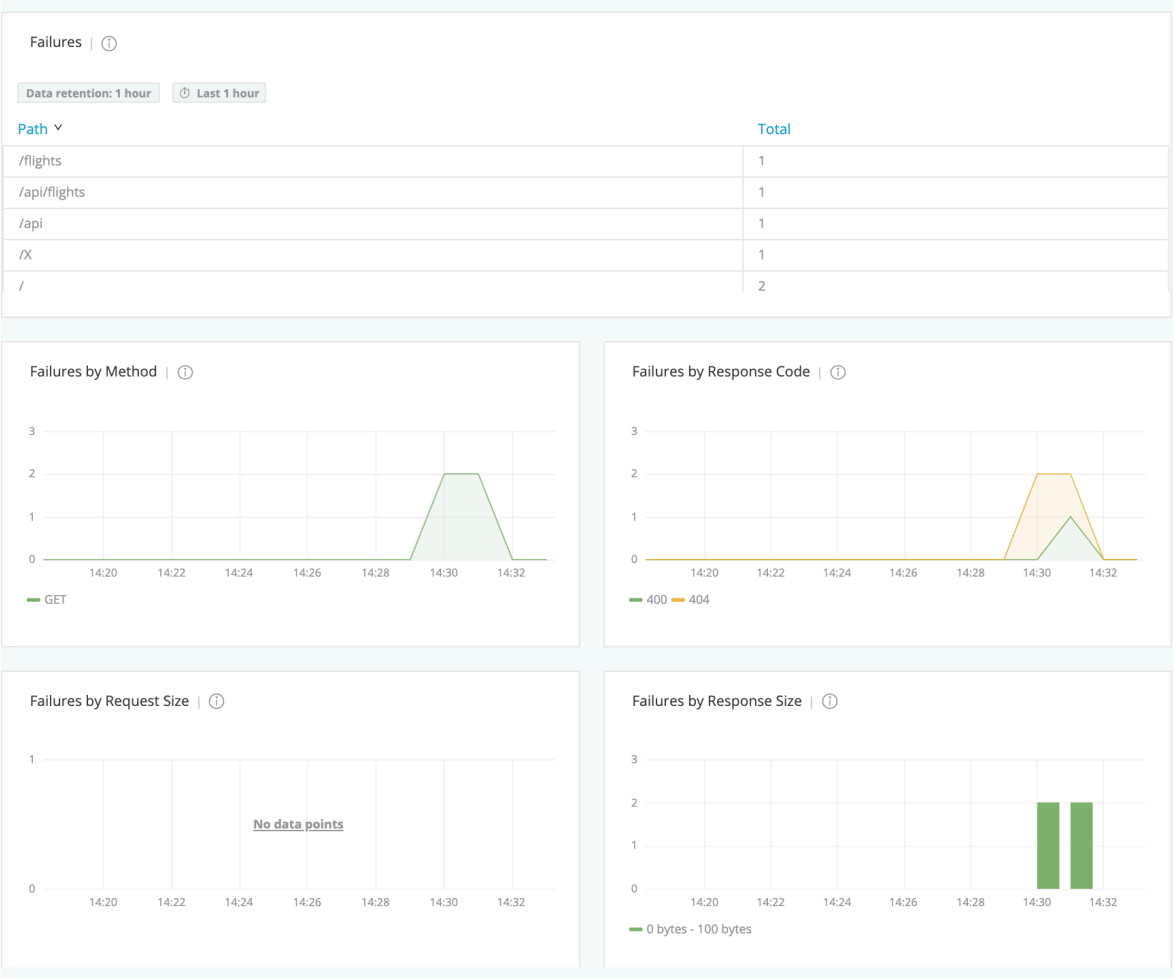


27. Click on Requests to see a dashboard with requests categorized by method, response code, request size and response size.

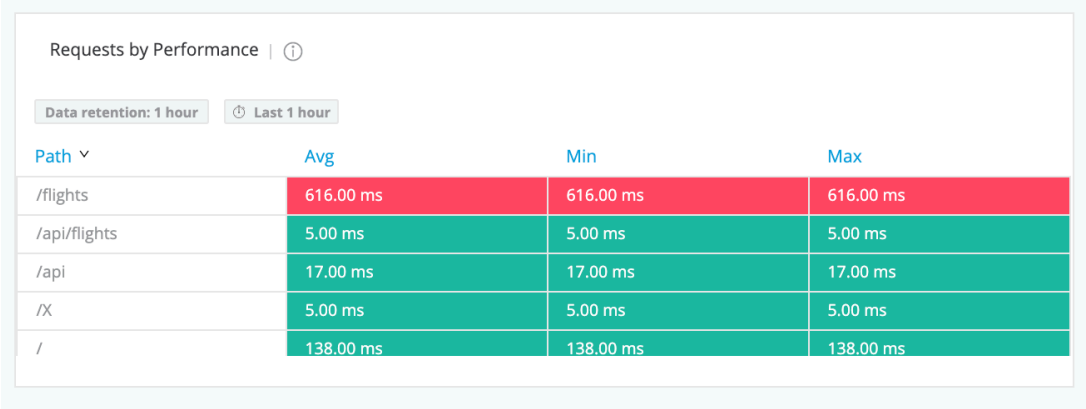
Note: It might take a minute or two for Anypoint Monitoring to reflect new requests.



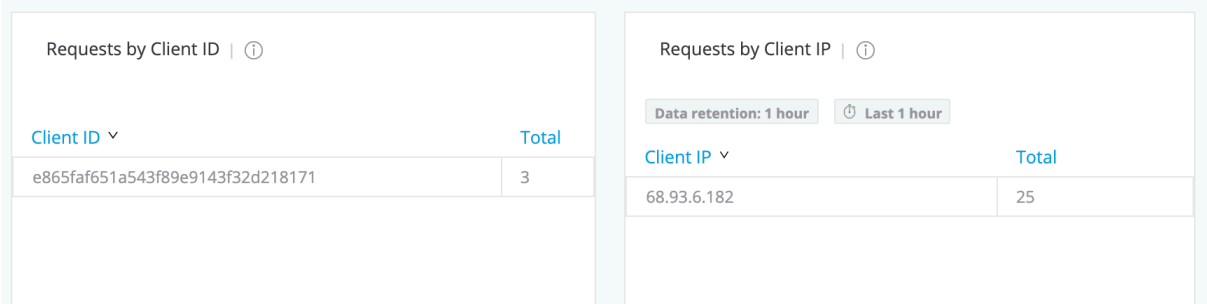
28. Similarly click on Failures to see a dashboard with the HTTP requests and responses where the response status code was 4xx or 5xx.



29. Click on Performance to see the dashboard with request-response latency (aka response time) by request URL path.



30. Click on Client Applications to see a breakdown of the HTTP requests by client ID or client IP.



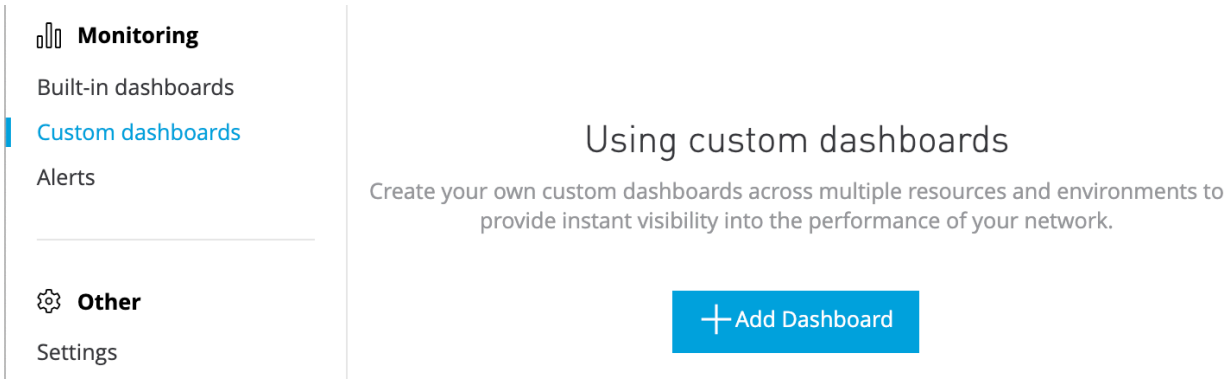
Walkthrough 8-2: Create a custom dashboard in Anypoint Monitoring

In this walkthrough, you create a custom chart in Anypoint Monitoring. You will:

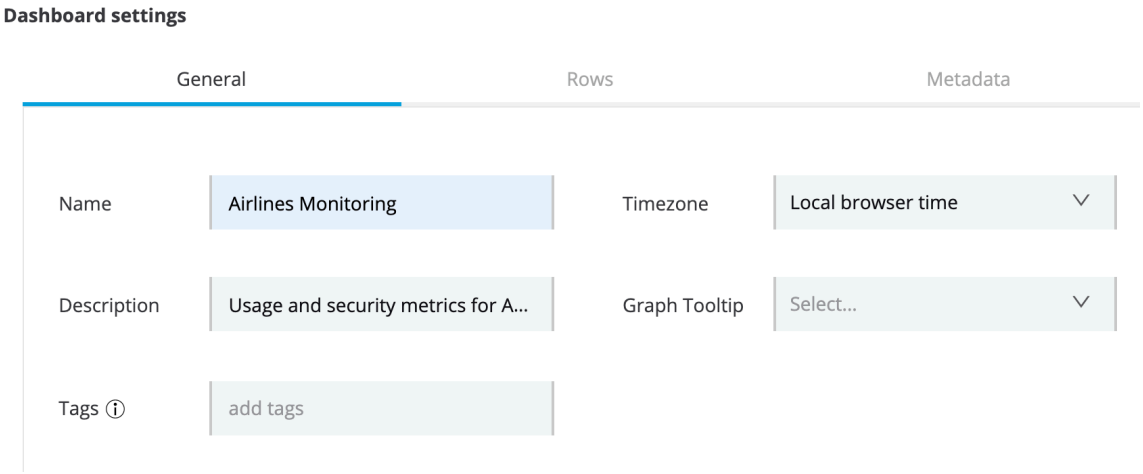
- Create a custom dashboard to monitor multiple APIs.
- Add custom charts to monitor usage and policy violations by API.

Create a new chart to visualize the total number of requests by API

1. In the left navigation menu in Anypoint Monitoring, select Custom Dashboards.

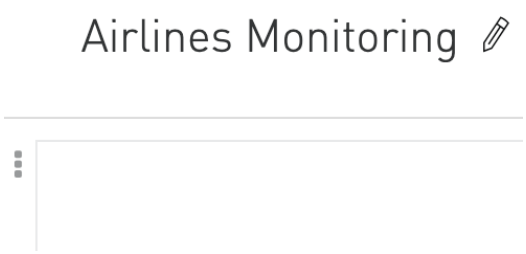


2. Click +Add Dashboard.
3. Click on the pencil next to New dashboard to edit the dashboard settings.
4. In the General tab set these values:
    - Name: Airlines Monitoring
    - Description: Usage and security metrics for American and United Airlines

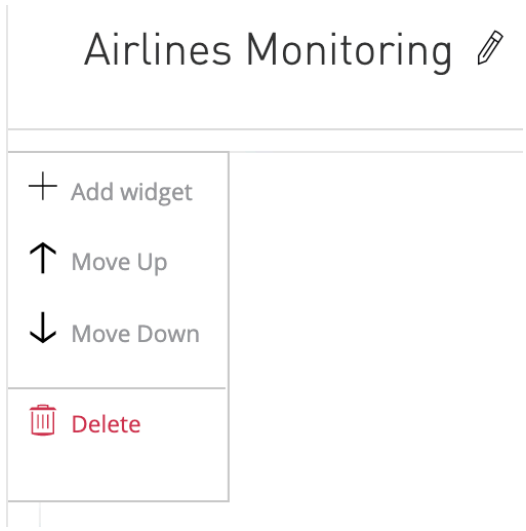


5. Click Apply changes.

6. On the Name before saving dialog box, leave the name as Airlines Monitoring and click Save.
7. Hover next to the three dots to modify the Empty Space.



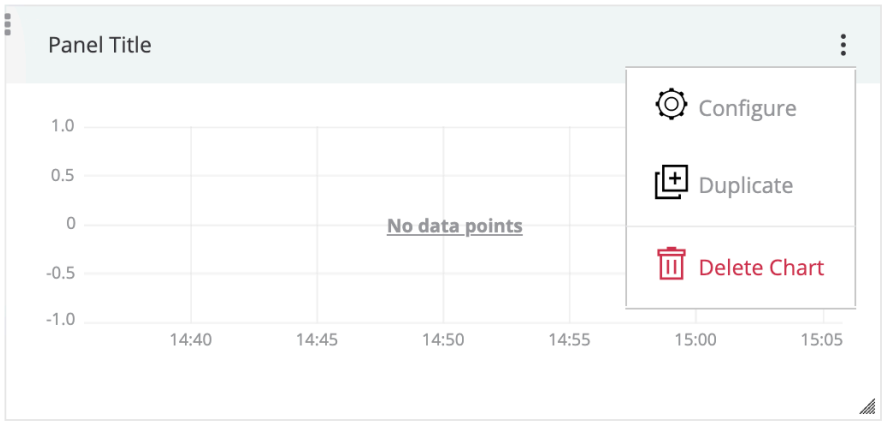
8. In the edit menu for the empty space select Add widget.



9. From the widget's options click to Add Graph.

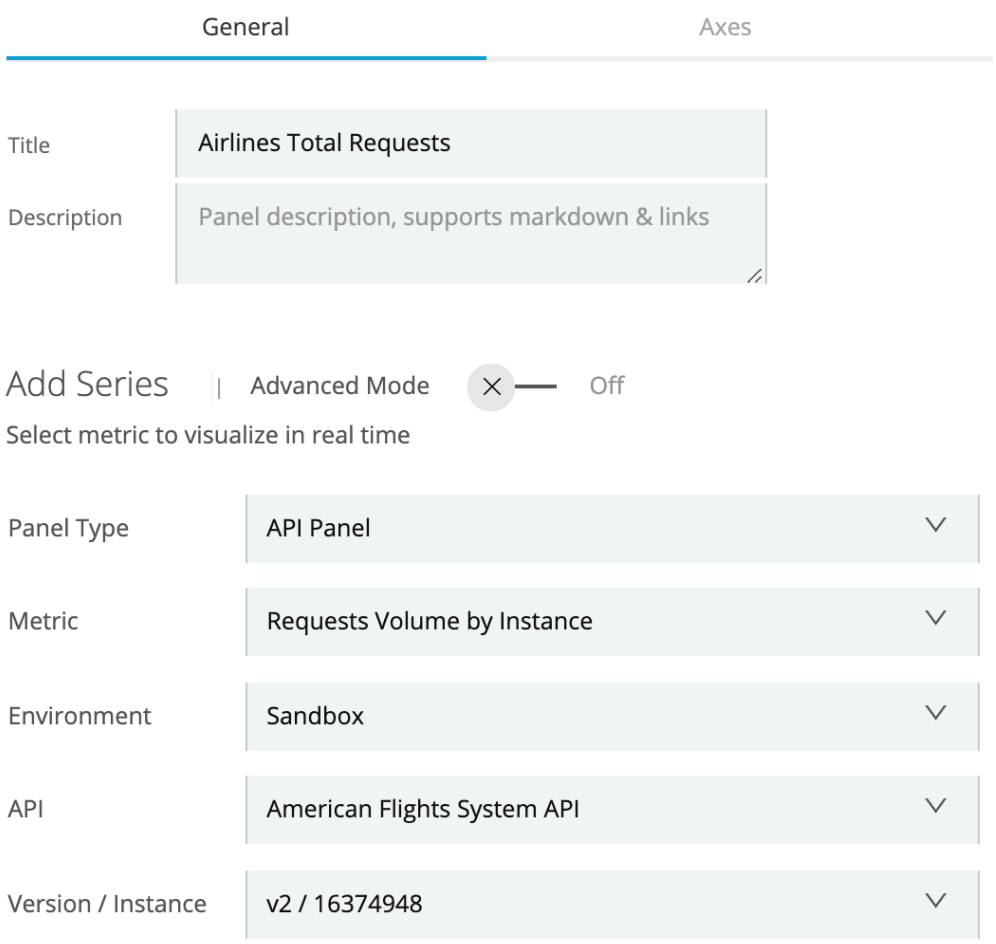
   Note: Notice a new panel will appear with no data points. It will be configured in the next steps.

10. Open the Panel menu from the right corner and select Configure.



11. Under the General tab enter the following properties:

- Title: Airline Total Requests
- Panel Type: API Panel
- Metric: Requests Volume by Instance
- Environment: Sandbox
- API: American Flights System API
- Version/Instance: v2



12. Click on Advanced Mode.
13. Click the plus sign next to the api_version_id value to add another condition.
14. Select api_version_id and select tag value as Sandbox - United Flights System API v1.
15. Modify the last AND to OR.

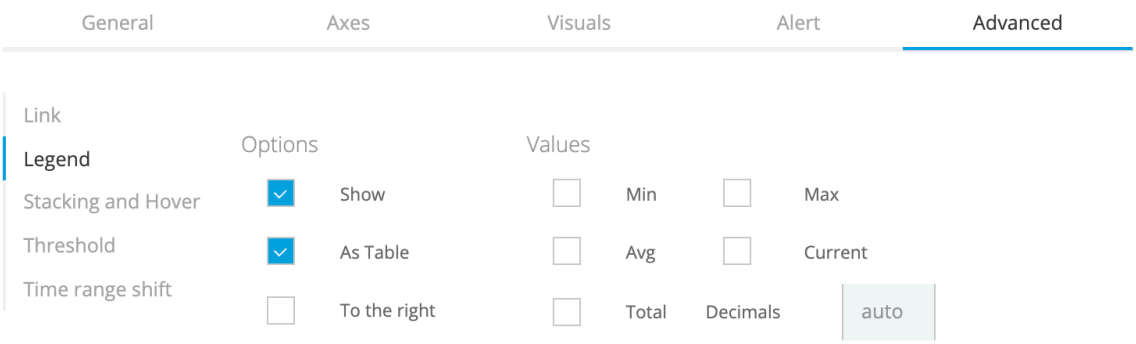
16. Verify the where clause shows as below, except for the instance numbers.

17. In the Group by clause click the plus sign and add tag(api_name).
18. Set the Alias by to:

```
$tag_api_name - $tag_api_version_id
```

Note: This value can also be found in the snippets file under Module 6.

19. Select the Advanced tab and select Legend.
20. Under Options mark As table.



21. Click Apply changes.
22. Change the time span in the calendar on top to display the last 12 hours and Confirm.

Note: Change the time to any other timeframe that reflects some data points.

23. Click the cross at the end of the tab to exit editing.

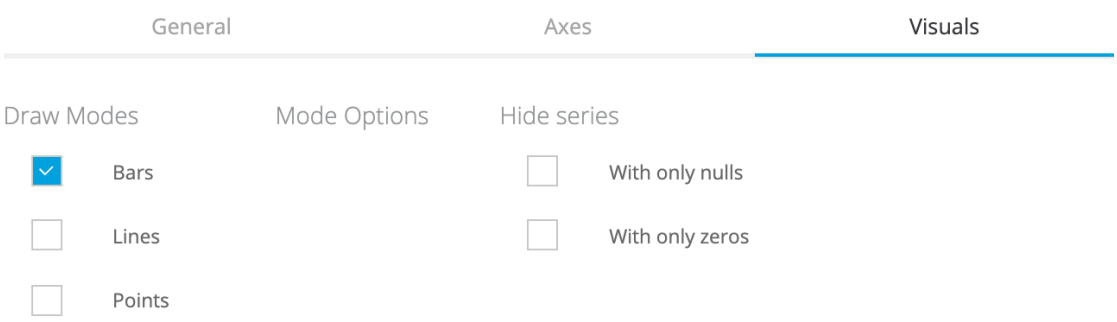
Add another chart to reflect the number of policy violations by API

24. Hover next to the three dots and select add widget.
25. From the widget's options click to Add Graph.
26. Open the Configure menu on the newly created Graph and select Configure.
27. In the General tab set the following settings:

- Title: Airlines Policy Violations.
- Panel Type: API Panel

- Environment: Sandbox
- API: American Flights System API
- Version/Instance: v1
- Metric: Policy Violations by Violation ID

28. Change to Visuals tab.
29. Under Draw Modes select Bars and unselect Lines.



30. Click Apply changes.
31. Click the cross at the end of the tab to exit editing.

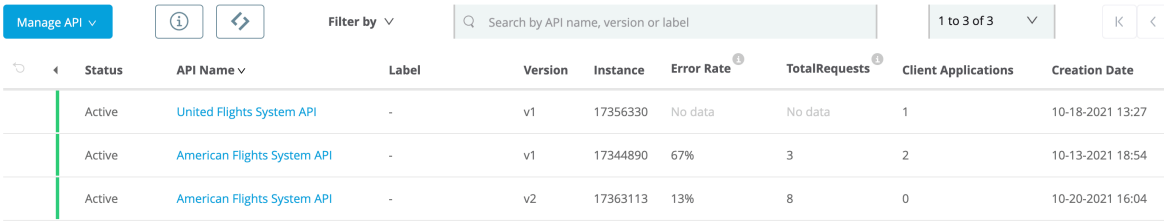
Walkthrough 8-3: Create a basic API alert in Anypoint Monitoring

In this walkthrough, you create a Request count alert notification for American Flights System API. You will:

- Explore the metrics available in the API list view.
- Create, configure, and trigger an API basic alert.

Explore the metrics available from the API list view in API manager

1. Open in a new browser tab the API Administration page of API Manager.
2. Make sure the environment selected is Sandbox.
3. Observe the API list containing the two versions of American Flight System API and United Flight System API.



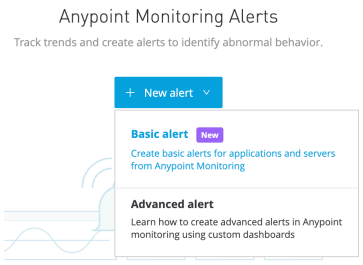
Status	API Name	Label	Version	Instance	Error Rate	TotalRequests	Client Applications	Creation Date
Active	United Flights System API	-	v1	17356330	No data	No data	1	10-18-2021 13:27
Active	American Flights System API	-	v1	17344890	67%	3	2	10-13-2021 18:54
Active	American Flights System API	-	v2	17363113	13%	8	0	10-20-2021 16:04

4. Notice the values on error rate and total requests columns for each API.

   Note: This view is a quick way to check for these key metrics per API in the last 24hrs. If there has been no requests these columns would show "No data".

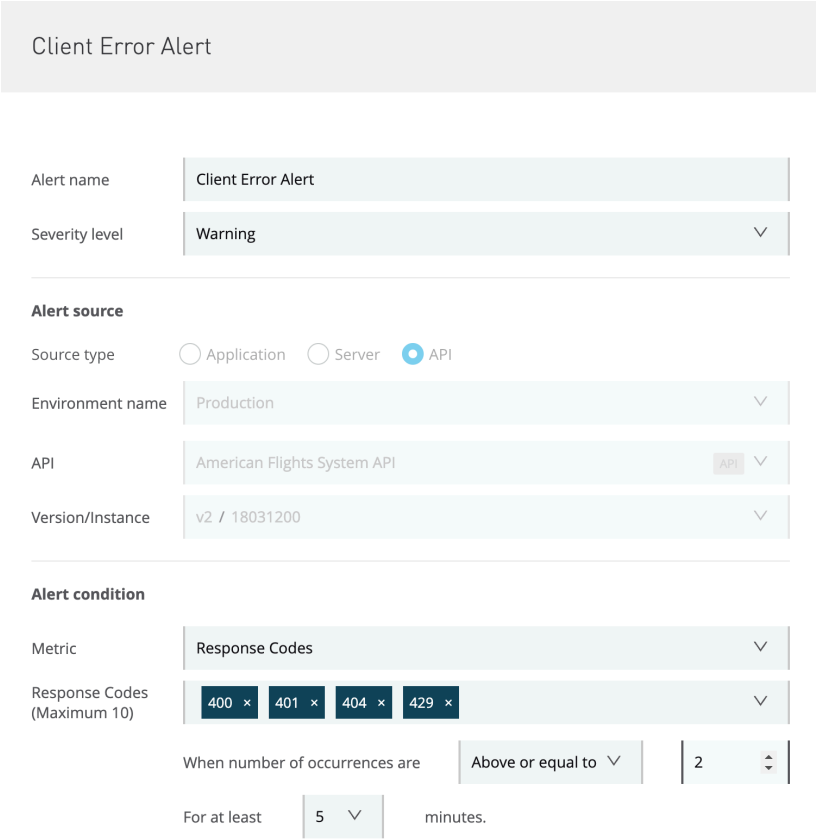
Create an API alert based on the response code

5. Return to Anypoint Monitoring.
6. From the left menu select Alerts.
7. In the Alerts page, click + New alert.
8. Select Basic Alert.



9. In the Add basic Alert page, enter the following information:

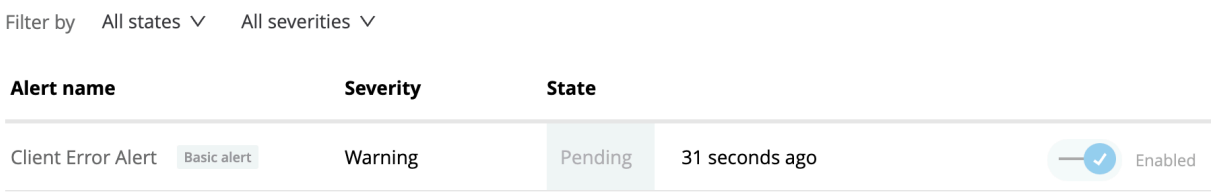
   - Alert name: Client Error Alert
   - Severity level: Warning
   - Source Type: API
   - Environment name: Sandbox
   - API: American Flight System API
   - Version/Instance: v2
   - Metric: Response Code
   - Response Codes: 400, 401, 404, 429
   - When number of occurrences are ***Above or equal 2*** times for at least ***5 minutes.***
   - Recipients: Select your admin user



10. Click Create.

    Note: When an alert is triggered, an email notification is sent to the recipients of the alert configured in the previous step.

11. Confirm the Alert is created and in pending State.



Note: It will take a few minutes for the alert to change to OK status.

Trigger the API Alert

12. Copy the consumer endpoint for the American Flights System API v2.

```
http://american-flights-api-v2-{initials}.{region}.cloudhub.io
```

13. Locate in your snippets file the watch command to trigger the alert and replace the endpoint placeholder.

```
watch -n 29 "curl -i <consumer-endpoint>"
```

Note: for Windows users use the following command

```
For /l %g in () do @(“curl -I <consumer-endpoint>” & timeout /t 29)
```

14. Open a new command line interface and execute the watch command.

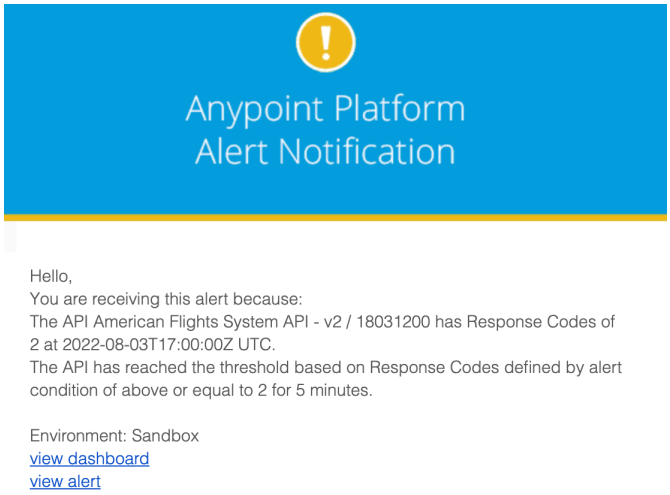
Note: You get error 404 resource not found.

15. Leave it running for at least 6 minutes.

Note: This will trigger the Client Error Alert, created previously.

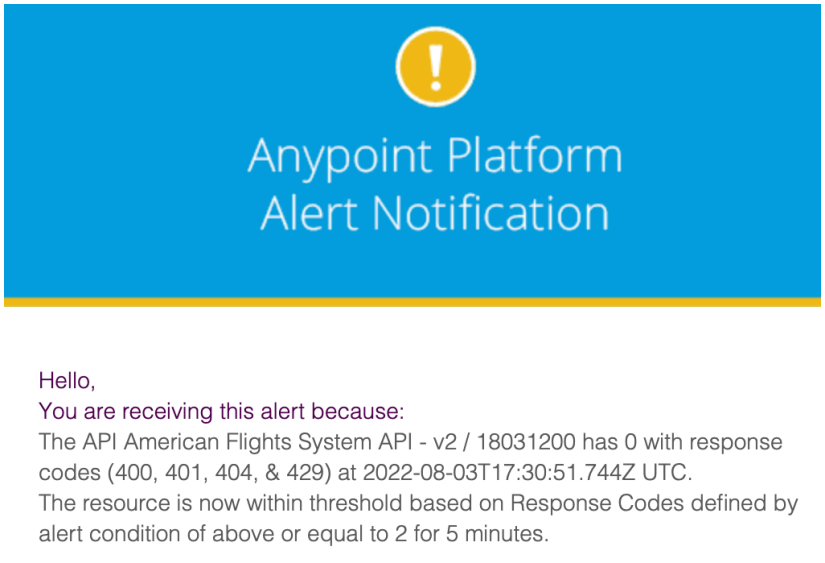
16. Refresh the alert screen in Anypoint Monitoring.
17. Verify that the State has changed from OK to Warning.

18. Check if there is a new message in the email address associated with the Anypoint account specified in the alert.



Note: It might take up to 15 minutes for the alert to trigger but the timestamp on the alert would be correct.

19. Close the command line window to stop the watch.
20. Wait 5-20 mins.
21. Refresh the alert screen in Anypoint Monitoring.
22. Verify that the alert state has changed from Warning to OK.
23. Check the email once more and you should have received another email stating that the API is no longer in alert state.



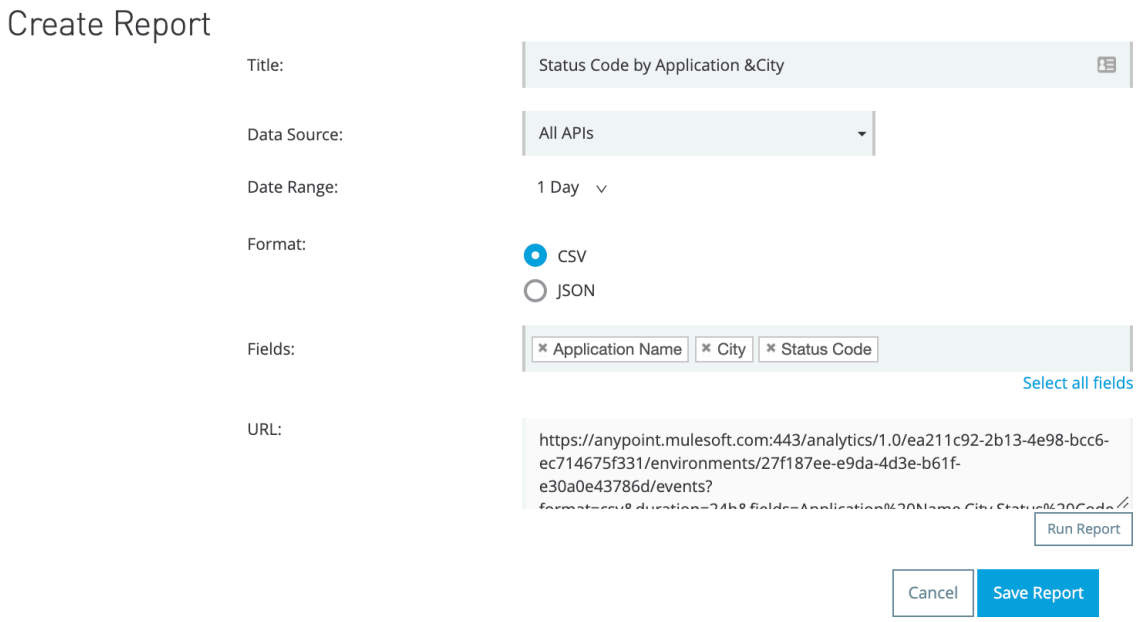
Walkthrough 8-4: Create a custom report in the Analytics Dashboard

In this walkthrough, you create a custom report in the Analytics Dashboard. You will:

- Create a custom CSV report in the Analytics Dashboard.
- Run a report in the Analytics Dashboard.

Create a custom report

1. Navigate to API Manager.
2. Select Analytics from the left menu.
3. In the left navigation menu in Analytics Dashboard, select Manage Reports.
4. Switch to Sandbox environment if not previously selected.
5. Click New.
6. In the Create Report page, fill in the following information:
    - Title: Status Code by Application & City
    - Data Source: American Flights System API
    - Select all versions from the drop-down menu
    - Set the date range as 1 day
    - Format: CSV
    - Fields: Click Select all fields link



7. Click Run Report to receive an immediate download.
8. Open the report in Excel (or another tool).
9. Go back to the web browser, click Save Report in the Analytics website so that you can run it anytime.