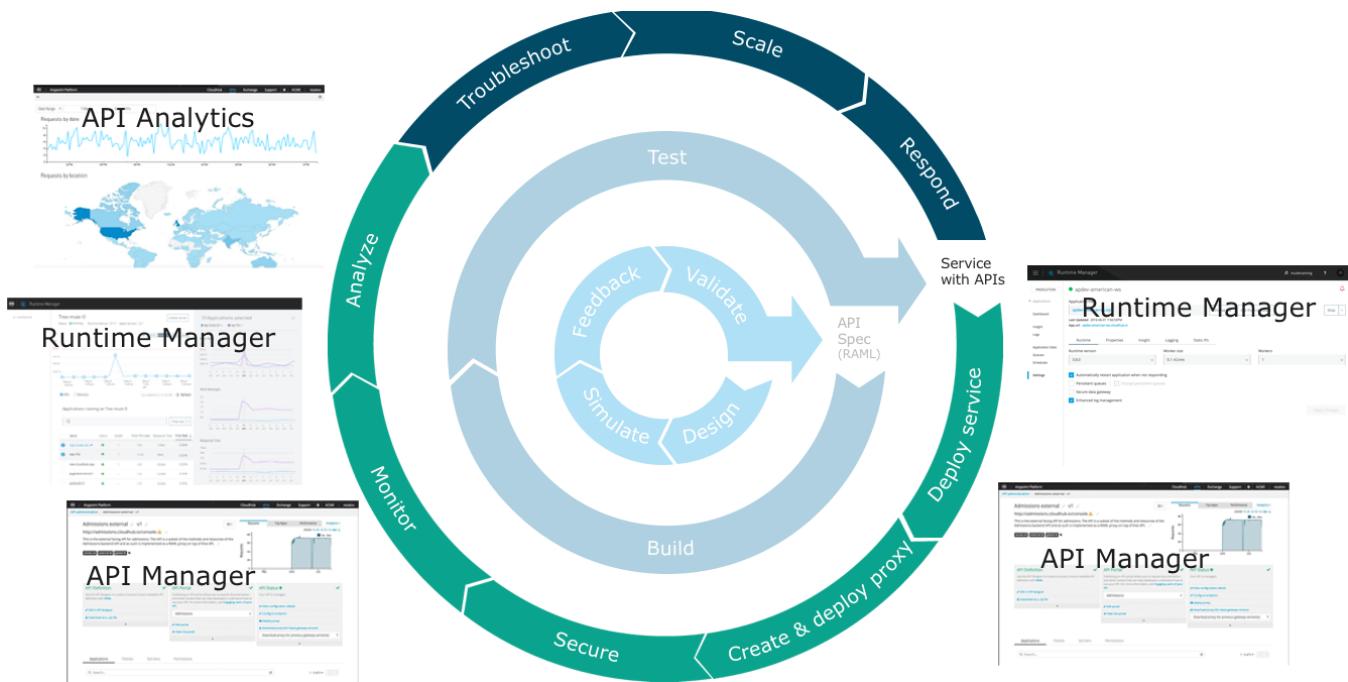


Module 5: Deploying and Managing APIs



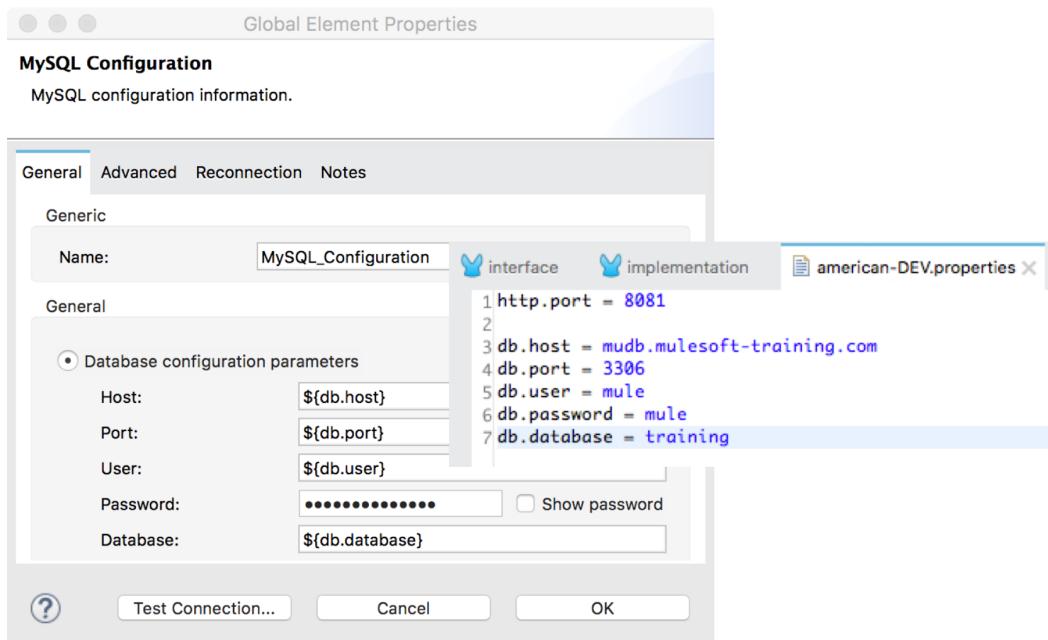
At the end of this module, you should be able to:

- Describe the options for deploying Mule applications.
- Use properties in Mule applications so they can be easily moved between environments.
- Deploy Mule applications to CloudHub.
- Use API Manager to create and deploy API proxies.
- Use API Manager to restrict access to API proxies.

Walkthrough 5-1: Prepare an API for deployment using properties

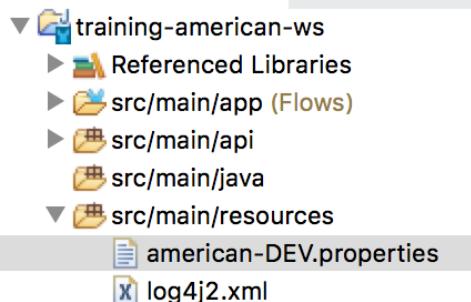
In this walkthrough, you introduce properties into your API implementation. You will:

- Create a properties file for an application.
- Create a Properties Placeholder global element to specify the properties file.
- Define and use Database connector properties.
- Specify a properties file dynamically.



Create a properties file

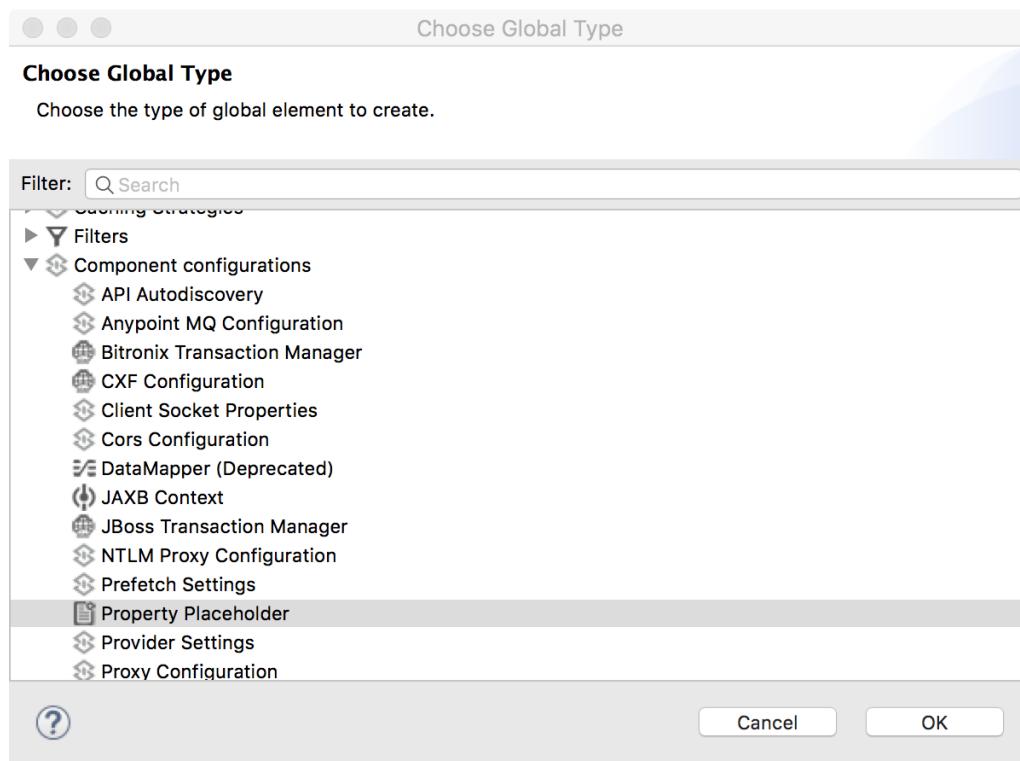
1. Right-click the src/main/resources folder in the Package Explorer and select New > File.
2. Set the file name to american-DEV.properties and click Finish.



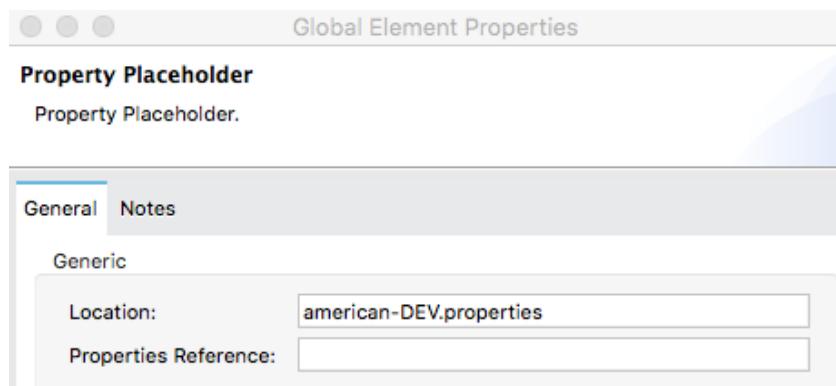
Create a Properties Placeholder global element

3. Return to interface.xml and go to its Global Elements view.

- In the Global Elements view, click Create.
- In the Choose Global Type dialog box, select Component configurations > Property Placeholder and click OK.



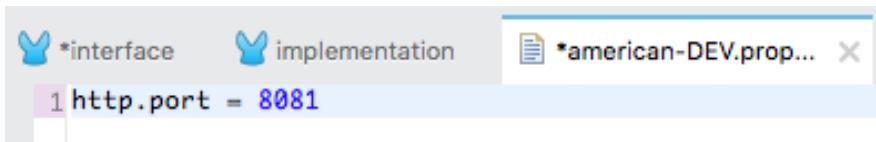
- In the Global Element Properties dialog box, set the location to american-DEV.properties and click OK.



Parameterize the HTTP Listener port

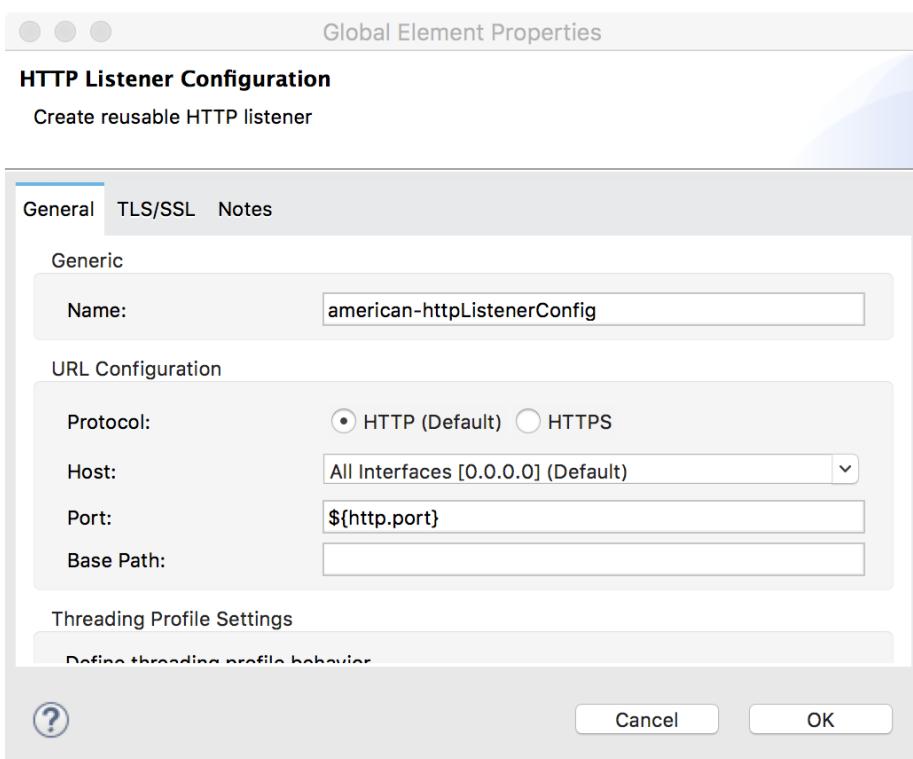
- Return to american-DEV.properties.

8. Create a property called http.port and set it to 8081.



```
*interface implementation *american-DEV.prop...
1 http.port = 8081
```

9. Save the file.
10. Return to the Global Elements view in interface.xml.
11. Double-click the HTTP Listener Configuration global element.
12. Change the port from 8081 to the application property, \${http.port}.
13. Click OK.



Parameterize the database credentials

14. Return to the course snippets.txt file and copy the database parameters (the five starting with db).

15. Return to american-DEV.properties and paste the values.

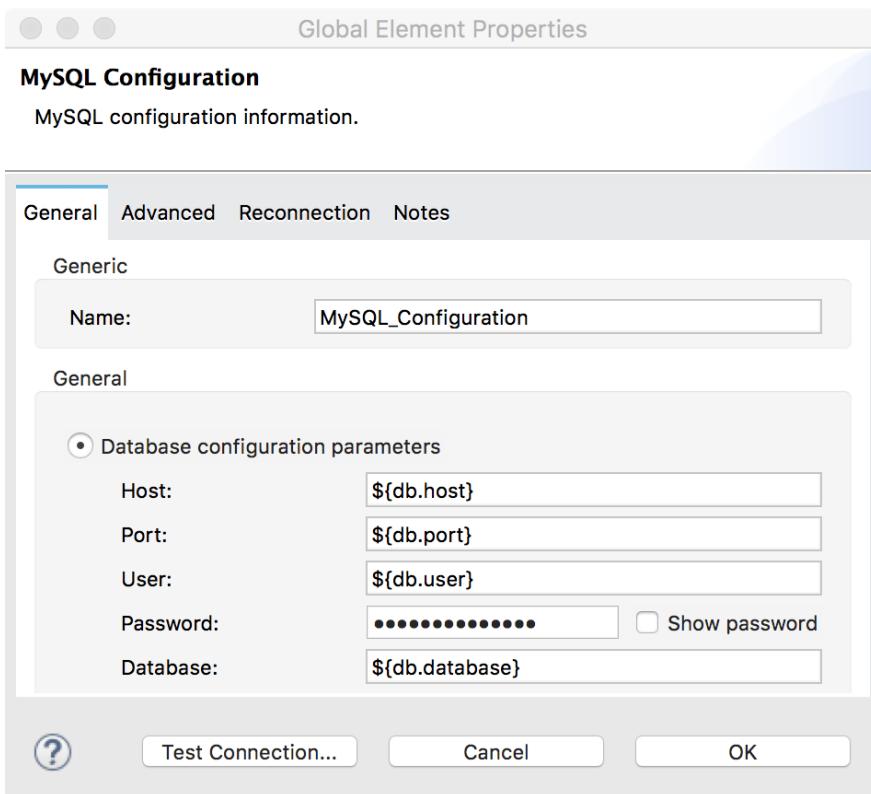
```
interface implementation american-DEV.properties
1 http.port = 8081
2
3 db.host = mudb.mulesoft-training.com
4 db.port = 3306
5 db.user = mule
6 db.password = mule
7 db.database = training
```

16. Save the file.

17. Return to the Global Elements view in implementation.xml.

18. Double-click the Database Configuration global element.

19. Change the values to use the application properties.



20. Click Test Connection and make sure it succeeds.

Note: If your connection fails, click OK and then go back and make sure you saved american-DEV.properties.

21. Click OK.

22. In the Global Element Properties dialog box, click OK.

23. Switch to the Message Flow view.

Test the application

24. Run the project.
25. In Postman, make a request <http://localhost:8081/api/flights> and confirm you still get data.

Define an environment property value in mule-app.properties

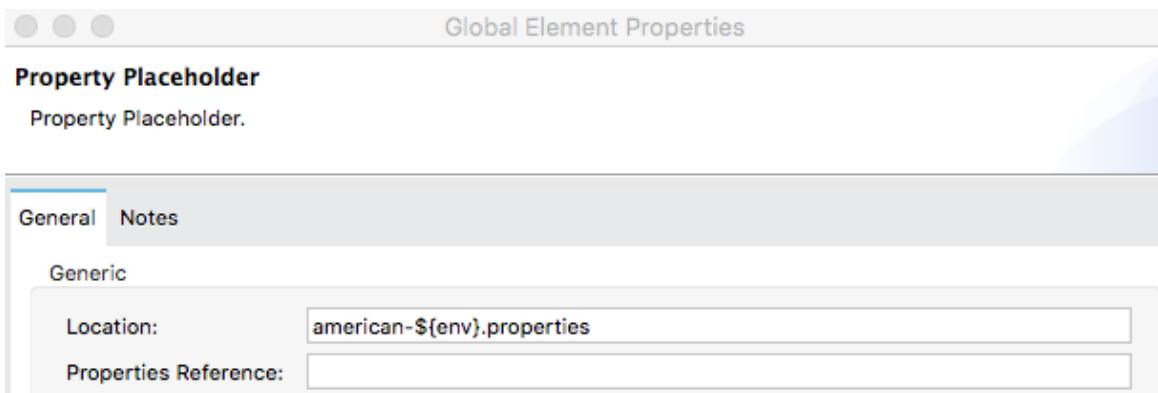
26. Return to Anypoint Studio and stop the project.
27. Open mule-app.properties located in the src/main/app folder.
28. Define a property called env and set it to DEV.



29. Save the file.

Use the environment property in the Property Placeholder

30. Return to the Global Elements view in interface.xml.
31. Double-click the Property Placeholder to edit it.
32. In the Global Element Properties dialog box, change the location to american-\${env}.properties and click OK.



Test the application

33. Return to the Message Flow view in interface.xml.
34. Run the project.
35. In Postman, make a request <http://localhost:8081/api/flights> and confirm you still get data.
36. Return to Anypoint Studio and stop the project.

Walkthrough 5-2: Deploy an application to CloudHub

In this walkthrough, you deploy and run your application on CloudHub. You will:

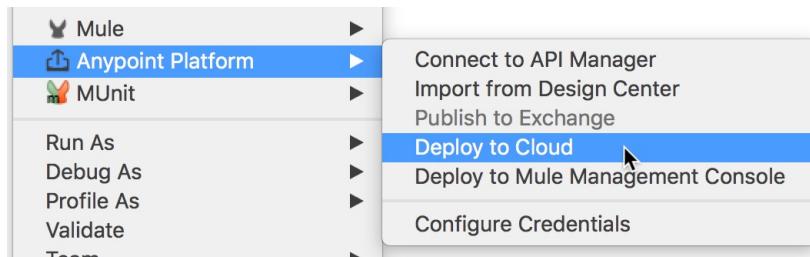
- Deploy an application from Anypoint Studio to CloudHub.
- Run the application on its new, hosted domain.
- Make calls to the web service.
- Update the API implementation deployed to CloudHub.

The screenshot shows the Anypoint Platform Runtime Manager. At the top, there's a navigation bar with icons for Training, Help, and MM. Below it is a search bar labeled "Search Applications". On the left, a sidebar has tabs for "Sandbox" (which is selected), "Applications", "Servers", and "Alerts". The main area displays a table of applications. The table has columns for Name, Server, Status, and File. One row is visible: "training-american-ws" is running on "CloudHub" with a green "Started" status and the file "training-american-ws.zip".

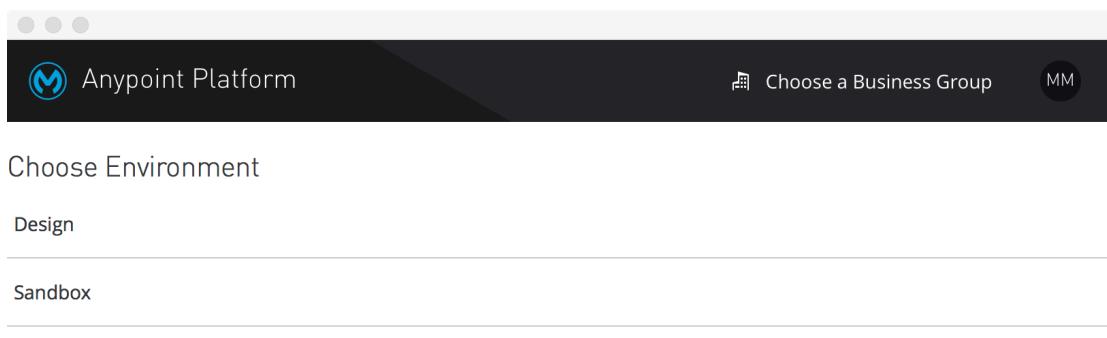
Note: If you do not have a working application at this point, import the training-american-ws-WT5.1 solution into Anypoint Studio and work with that project.

Deploy the application to CloudHub

1. Return to the training-american-ws project in Anypoint Studio.
2. In the Package Explorer, right-click the project and select Anypoint Platform > Deploy to Cloud.



3. In the Choose Environment dialog box, select Sandbox.



- At the top of the Anypoint Platform dialog box, set the application name to training-american-ws-{your-lastname} so it is a unique value.

Note: This name will be part of the URL used to access the application on CloudHub. It must be unique across all applications on CloudHub. The availability of the domain is instantly checked and you will get a green check mark if it is available.

The screenshot shows the 'Deploying Application' dialog box. On the left, there's a 'Sandbox' button. The main area has 'Deploying Application' at the top. Below it, 'Application Name' is set to 'training-american-ws-mule' with a green checkmark. Under 'Deployment Target', 'CloudHub' is selected. To the right, 'Application File' is listed as 'training-american-ws.zip'.

- Make sure the runtime version is set to the version your project is using.

Note: If you don't know what version it is using, look at the Package Explorer and find a library folder with the name of the server being used, like Mule Server 3.9.0 EE.

6. Change the worker size to 0.1 vCores.

The screenshot shows the 'Runtime' tab of the configuration interface. Under 'Runtime version', '3.9.0' is selected. Under 'Worker size', '0.1 vCores' is selected. Under 'Workers', '1' is selected.

- Click the Properties tab; you should see the env variable set to DEV.

The screenshot shows the 'Properties' tab. Under 'Text', the variable 'env=DEV' is listed.

- Click the Deploy Application button.

9. Click the Open in Browser button.

The screenshot shows a modal window titled "Runtime Manager". Inside, a circular progress bar indicates the status of a deployment. Below the progress bar, the text "Deploying training-american-ws to CloudHub" is displayed. A message below says "You may close this window at any time." At the bottom right are two buttons: "Open in Browser" (white background) and "Close Window" (blue background).

10. In the Anypoint Platform browser window that opens, locate the status of your deployment in the Runtime Manager.

The screenshot shows the main "Runtime Manager" interface. On the left, a sidebar lists "Sandbox", "Applications", "Servers", and "Alerts". The "Applications" tab is selected. The main area displays a table with columns: Name, Server, Status, and File. One row is visible: "training-american-ws-mule" is listed under "Server" as "CloudHub", "Status" is "Deploying", and the "File" is "training-american-ws.zip".

Watch the logs and wait for the application to start

11. Click in the row of the application (not on its name); you should see information about the application appear on the right side of the window.

The screenshot shows the "Runtime Manager" interface with the "Applications" tab selected. On the left, a sidebar lists "Sandbox", "Applications", "Servers", "Alerts", "VPCs", and "Load Balancers". The "Applications" tab is selected. The main area displays a table with columns: Name, Server, Status, and File. A row for "training-american-ws-mule" is selected, showing "CloudHub" as the server, "Deploying" as the status, and "training-american-ws.zip" as the file. To the right, a detailed view of the application is shown, including its name, deployment status, server, file, and runtime details. Buttons for "Manage Application", "Logs", and "Insight" are at the bottom.

12. Click the Logs button.

13. Watch the logs as the application is deployed.
14. Wait until the application starts (or fails to start).

The screenshot shows the Mule Runtime Manager interface. On the left, there's a sidebar with options like SANDBOX, Applications, Dashboard, Insight, Logs (which is selected), Application Data, Queues, Schedules, and Settings. The main area shows the application 'training-american-ws-mule' with a 'Live Console' tab. The console displays log messages:

```
*****
* Application: training-american-ws-mule
* OS encoding: /, Mule encoding: UTF-8
*
* Agents Running:
*   JMX Agent
*   Batch module default engine
*   DevKit Extension Information
*   Wrapper Manager
*****
09:33:45.650 11/20/2017 Deployment system SYSTEM
Worker(54.85.130.219): Your application has started successfully.

09:33:46.313 11/20/2017 Deployment system SYSTEM
Your application is started.
```

To the right, there's a 'Deployments' panel showing a successful deployment entry for '09:32 - Deployment' with a checkmark. It also includes links for 'System Log' and 'Worker-0'.

Note: If your application did not successfully deploy, read the logs to help figure out why the application did not deploy. If you had errors when deploying, troubleshoot them, fix them, and then redeploy.

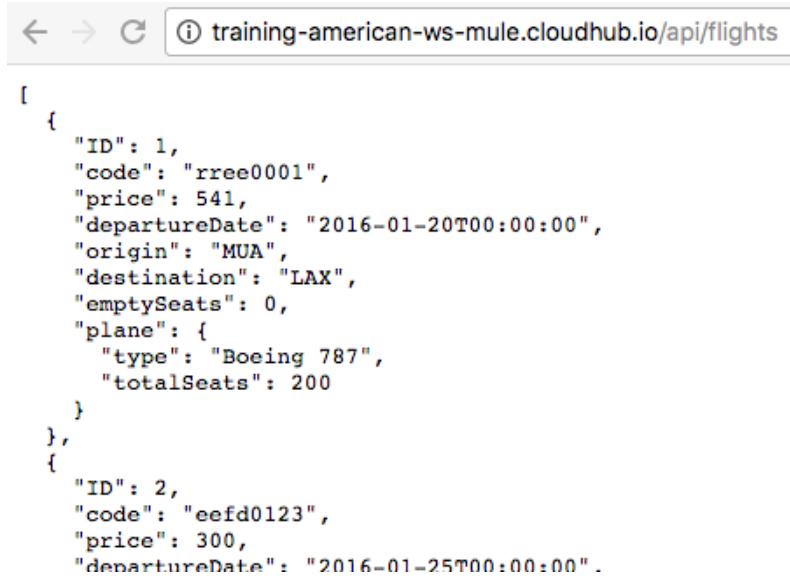
Test the application

15. In the left-side navigation, click Dashboard.
16. Locate the link for the application on its new domain:
training-american-ws-{lastname}.cloudbhub.io.

The screenshot shows the Mule Runtime Manager interface with the 'Dashboard' tab selected. It displays the application 'training-american-ws-mule' and its domain: training-american-ws-mule.cloudbhub.io. Below that, it says 'Mule messages'.

17. Click the link; a request will be made to that URL in a new browser tab and you should get a message that there is no listener for that endpoint.

18. Modify the path to <http://training-american-ws-{lastname}.cloudbhub.io/api/flights>; you should see the flights data.

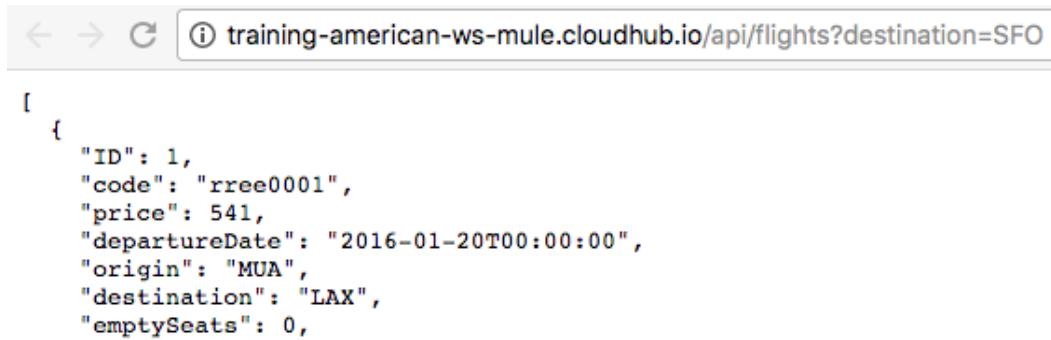


```
[  
  {  
    "ID": 1,  
    "code": "rree0001",  
    "price": 541,  
    "departureDate": "2016-01-20T00:00:00",  
    "origin": "MUA",  
    "destination": "LAX",  
    "emptySeats": 0,  
    "plane": {  
      "type": "Boeing 787",  
      "totalSeats": 200  
    }  
  },  
  {  
    "ID": 2,  
    "code": "eefd0123",  
    "price": 300,  
    "departureDate": "2016-01-25T00:00:00".  
  }]
```

Note: If you are using the local Derby database, your application will not return results when deployed to CloudHub. You will update the application with a version using the MySQL database in the next section so it works.

19. Add a query parameter called destination to the URL and set it equal to SFO.
20. Send the request; you should still get all the flights.

Note: You did not add logic to the application to search for a particular destination. You will deploy an application with this additional functionality implemented next.



```
[  
  {  
    "ID": 1,  
    "code": "rree0001",  
    "price": 541,  
    "departureDate": "2016-01-20T00:00:00",  
    "origin": "MUA",  
    "destination": "LAX",  
    "emptySeats": 0,
```

Note: If you wanted to test this API implementation from Exchange, you could do either 1) setting the baseUri in the API specification to the URL of the API implementation and then republishing the API specification or 2) clicking API instances in Exchange and adding a new API instance with this URL. Typically, however, you want to govern and manage access to an API. You will create a proxy to do this in the next walkthrough.

21. Leave this browser tab open.

Update the API implementation deployed to CloudHub

22. Return to the browser tab with Runtime Manager.
23. In the left-side navigation, click Settings for the training-american-ws-{lastname} application.
24. Click the Choose file button.
25. Browse to the resources folder in the course student files.
26. Select training-american-ws-v2.zip and click Open.

Note: This updated version of the application adds functionality to return results for a particular destination. You will learn to do this later in the Development Fundamentals courses.

27. Click the Apply Changes button.

The screenshot shows the Runtime Manager interface. At the top, there's a header with a menu icon, a search icon, the title 'Runtime Manager', a 'Training' link, a help icon, and a user profile icon. Below the header, on the left, is a sidebar with navigation links: 'Sandbox' (selected), 'Applications' (with a back arrow), 'Dashboard', 'Insight', 'Logs', 'Application Data', 'Queues', 'Schedules', and 'Settings' (selected). The main content area displays the application 'training-american-ws-mule'. It includes an 'Application File' section with a file input field containing 'training-american-ws-v2.zip', a 'Choose file' button, a 'Get from sandbox' button, a 'Stop' button, and a dropdown menu. Below this is an 'App url:' field with the value 'training-american-ws-mule.cloudhub.io'. The application details section has tabs for 'Runtime', 'Properties', 'Insight', 'Logging', and 'Static IPs'. Under 'Runtime', the 'Runtime version' is set to '3.9.0', 'Worker size' is '0.1 vCores', and 'Workers' is '1'. A warning message states: '⚠ Your current subscription allows only one worker per application'. There are checkboxes for 'Automatically restart application when not responding' (checked) and 'Persistent queues' (unchecked). At the bottom right is a large blue 'Apply Changes' button.

28. Wait until the application is uploaded and then redeloys successfully.

Note: Because this can take some time for trial accounts, your instructor may move on with the next topic and then come back to test this later.

29. Close the browser tab with Runtime Manager.

Test the updated application

30. Return to the browser tab making a request to the API implementation on CloudHub with a destination of SFO and refresh it; you should now get only flights to SFO.



The screenshot shows a browser window with the URL `training-american-ws-mule.cloudhub.io/api/flights?destination=SFO`. The page displays a JSON array containing two flight records. Both flights have an origin of "MUA" and a destination of "SFO". The first flight has an ID of 5, a code of "rree1093", a price of 142, and a departure date of "2016-02-11T00:00:00". It is currently at 1 empty seat out of 150 on a Boeing 737. The second flight has an ID of 7, a code of "eefd1994", a price of 676, and a departure date of "2016-01-01T00:00:00". It is currently at 0 empty seats out of 150 on a Boeing 737.

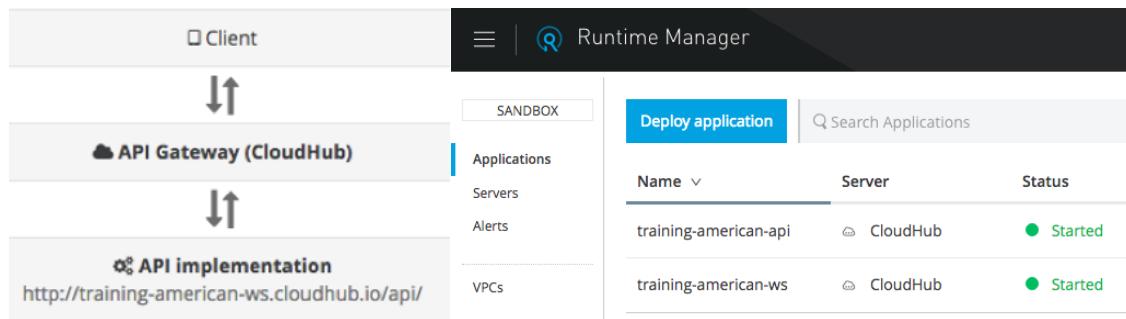
```
[{"ID": 5, "code": "rree1093", "price": 142, "departureDate": "2016-02-11T00:00:00", "origin": "MUA", "destination": "SFO", "emptySeats": 1, "plane": {"type": "Boeing 737", "totalSeats": 150}}, {"ID": 7, "code": "eefd1994", "price": 676, "departureDate": "2016-01-01T00:00:00", "origin": "MUA", "destination": "SFO", "emptySeats": 0, "plane": {"type": "Boeing 737", "totalSeats": 150}}]
```

31. Close this browser tab.

Walkthrough 5-3: Create and deploy an API proxy

In this walkthrough, you create and deploy an API proxy for your API implementation on CloudHub. You will:

- Add an API to API Manager.
- Use API Manager to automatically create and deploy an API proxy application.
- Set a label and consumer endpoint for a proxy so requests can be made to it from Exchange.
- Make calls to the API proxy from API portals for both internal and external developers.
- View API request data in API Manager.

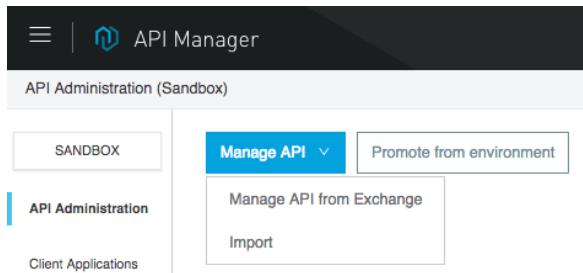


Create and deploy a proxy application

1. Return to the browser tab with Runtime Manager.
2. In the main menu, select API Manager; you should see no APIs listed.

The screenshot shows the API Manager interface. The top navigation bar includes 'Training', a question mark icon, and a user profile icon. The main content area is titled 'API Administration (Sandbox)' and features a search bar with 'Manage API' and 'Promote from environment' buttons. A sidebar on the left lists 'Sandbox', 'API Administration' (which is selected and highlighted in blue), 'Client Applications', 'Custom Policies', and 'Analytics'. The central area displays a message: 'No APIs to display. Get started by adding your first API.' Below this message is a small chart icon and the text 'Select an API version to see more details'.

- Click the Manage API button and select Manage API from Exchange.



- For API name, start typing American in the text field and then select your American Flights API in the drop-down menu that appears.
- Set the rest of the fields to the following values:
 - API version: v1
 - Asset version: 1.0.1
 - Managing type: Endpoint with Proxy
 - Implementation URI: `http://training-american-ws-{lastname}.cloudhub.io/api`
 - Proxy deployment target: CloudHub

The screenshot shows the 'Manage API from Exchange' configuration form. The 'API name:' field contains 'American Flights API'. The 'API version:' field is set to 'v1'. The 'Asset version:' field is set to '1.0.1'. Under 'Managing type:', the 'Endpoint with Proxy' radio button is selected. The 'Implementation URI:' field contains '`http://training-american-ws-mule.cloudhub.io/api`'. The 'Proxy deployment target:' field has 'CloudHub' selected. The 'Path:' field contains '/'. There is a checkbox at the bottom that says 'Check this box if you are managing this API in Mule 4 or above.' Below the form, there are 'Cancel' and 'Save' buttons.

- Click Save.

7. In the Deployment Configuration section, set the following values:

- Runtime version: 3.8.x (or a later value)
- Proxy application name: training-american-api-{lastname}

Deployment Configuration ▾

Runtime version: 3.8.x

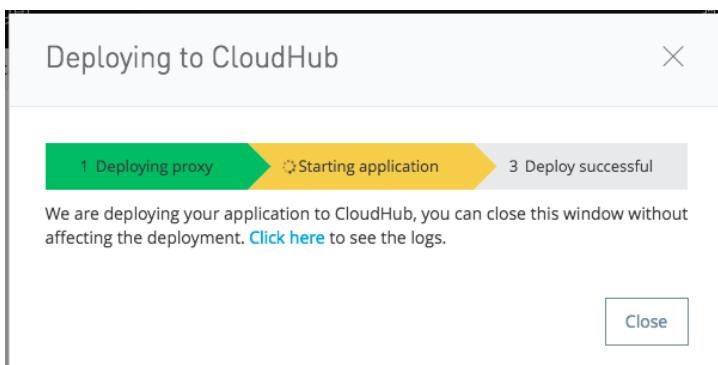
Proxy application name: .cloudhub.io

Update application if exists

Deploy

8. Click Deploy.

9. In the Deploying to CloudHub dialog box, click the Click here link to watch the logs.



10. In the new browser tab that opens, watch the logs in Runtime Manager.

11. Wait until the proxy application starts.

Note: If it does not successfully deploy, read the logs to help figure out why the application did not deploy. If you had errors when deploying, troubleshoot them, fix them, and then redeploy.

The Runtime Manager interface shows the "Logs" tab selected. The left sidebar has links for "Sandbox", "Applications", "Dashboard", "Insight", and "Logs". The main area shows the application "training-american-api-mule" and a "Live Console" with a search bar and advanced options. The logs pane displays the following entries:

```
*****
10:04:11.700 11/20/2017 Deployment system SYSTEM
Worker(54.174.38.2): Your application has started successfully.

10:04:12.226 11/20/2017 Deployment system SYSTEM
Your application is started.
```

To the right, a "Deployments" panel shows a deployment log from "Today" at 10:03.

- In the left-side navigation, click Applications; you should see the proxy application.
- Click the row for the proxy and review its information in the right section of the window.

Name	Server	Status	File
training-american-api-mu	CloudHub	Started	training-american-api-mule-api
training-american-ws-mule	CloudHub	Started	training-american-ws-v2.zip

training-american-api-mule

Started CloudHub

training-american-api-mule... Choose file

Last Updated 2017-11-20 10:04:11AM
App url: training-american-api-mule.cloudhub.io

Runtime version: 3.8.5
Worker size: 0.1 vCores
Workers: 1

Manage Application Logs Insight

- Close the browser tab.

View API details in API Manager

- Return to the tab with API Manager and click the Close button in the Deploying to CloudHub dialog box.
- Review the API proxy information at the top of the page.

API Administration (Sandbox) American Flights API (v1) - Settings

American Flights API v1	
API Status: Active	Asset Version: 1.0.1 Type: RAML/OAS
Implementation URL: http://training-american-ws-mule.cloudhub.io/api + Add consumer endpoint	
API Instance ⓘ	Autodiscovery: ⓘ
ID: 5835551	API Name: groupId:800b1585-b6be-4c62-82de-02d8c2adf413:assetId:american-flights-api
Label: + Add a label	API Version: v1:5835551
Proxy	
Proxy Application: training-american-api-mule	
Proxy URL: training-american-api-mule.cloudhub.io	

17. In the left-side navigation, click the API Administration link; you should now see your American Flights API listed.

18. Click in the row for the v1 version – but not on the v1 link.

19. Review the API version info that appears on the right side of the window; you should see there are no policies, SLA tiers, or client applications.

The screenshot shows the API Manager interface. On the left, a sidebar has 'Sandbox' selected. Under 'API Administration', 'Client Applications', 'Custom Policies', and 'Analytics' are listed. The main area shows a table with columns: API Name, Version, Status, Client Applications, and Creation Date. One row is visible for 'American Flights API' with version 'v1'. To the right, a panel for 'American Flights API v1' displays links for 'Manage CloudHub Proxy', 'View API in Exchange', and 'View Analytics Dashboard'. Below these are tabs for 'Applications' (which is selected), 'Policies', and 'SLA tiers'. A note at the bottom says 'There are no applications for this API version.'

20. In the API list, click the v1 link for the API; you should be returned to the Settings page for the API.

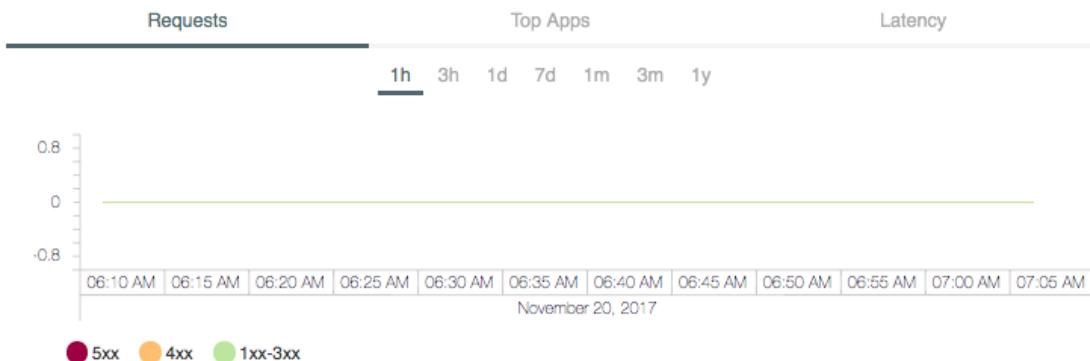
21. Locate and review the links on the right side of the page.

22. Click the View configuration details link.

This dialog box shows the flow of requests: 'Client' goes down to 'API Gateway (CloudHub)', which then goes down to 'API implementation' at the URL <http://training-american-ws-mule.cloudhub.io/api>. There are two downward arrows between the boxes, indicating the flow of data from client to gateway and then to the implementation.

23. In the Endpoint configuration details dialog box, click Close.

24. On the Settings page, look at the requests graph; you should not see any API requests yet.



View the new API proxy instance in Exchange

25. Return to the browser tab with Exchange.

26. Return to the home page for your American Flights API.

27. Locate the API instances now associated with asset version 1.0.1; you should see the Mocking Service instance and now the new proxy.

Version	Instances
1.0.1	Mocking Service Sandbox - v1:5831632
1.0.0	

Note: You may need to refresh your browser page.

28. Click the GET method for the flights resource.

29. In the API console, click the drop-down arrow next to Mocking Service; you should NOT see the API proxy as a choice.

GET

Mocking Service

Mocking Service

30. In the left-side navigation, click API instances; you should see that the new proxy instance does not have a URL.

The screenshot shows the MuleSoft API Manager interface. On the left, there's a sidebar with navigation links: Assets list, American Flights API, API summary, Types, Resources, /flights, GET, POST, and API instances. The API instances link is highlighted with a blue border. The main content area is titled "American Flights API" with a version "v1". Below it, the heading "API instances" is followed by a table. The table has columns: Instances, Environment, URL, and Visibility. It lists one instance: "Mocking Service" in "Sandbox" environment with the URL "https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1". The visibility is set to "Public". There's also a "Private" option and an edit icon next to it. At the bottom of the table, there's a link "+ Add new instance".

Set a friendly label for the API instance in API Manager

31. Return to the browser tab with API Manager.
32. On the Settings page for your American Flights API, click the Add a label link.
33. Set the label to No policy and press Enter/Return.

API Instance ⓘ
ID: 5831632
Label: No policy ⚙️

Set a consumer endpoint for the proxy in API Manager

34. Locate the proxy URL.

Proxy

Proxy Application: training-american-api-mule

Proxy URL: training-american-api-mule.cloudhub.io

35. Right-click it and copy the link address.

36. Click the Add consumer endpoint link.

American Flights API v1

API Status: ● Active Asset Version: 1.0.1 Type: RAML/OAS

Implementation URL: <http://training-american-ws-mule.cloudhub.io/api> [⊕ Add consumer endpoint](#)

37. Paste the value of the proxy URL.

38. Press Enter/Return.

American Flights API v1

API Status: ● Active Asset Version: 1.0.1 Type: RAML/OAS

Implementation URL: <http://training-american-ws-mule.cloudhub.io/api>

Consumer endpoint: <http://training-american-api-mule.cloudhub.io/> [✎](#)

Make requests to the API proxy from Exchange

39. Return to the browser tab with Exchange.

40. Refresh the API instances page for your American Flights API; you should see the new label and the URL.

The screenshot shows the Exchange interface with the following details:

- Header:** Exchange
- Left sidebar (Assets list):**
 - Assets list
 - American Flights API
 - API summary
 - > Types
 - ▼ Resources
 - ▼ /flights
 - GET
 - POST
 - > /{ID}
 - API instances
- Main Content Area:**
 - American Flights API** (with edit icon) | v1 ▾
 - API instances**
 - | Instances | Environment | URL | Visibility |
|-----------------|--|--|---------------------------|
| Mocking Service | proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1 | https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1 | Public |
| No policy | Sandbox | http://training-american-api-mule.cloudhub.io/ | Private ✎ |
 - [+ Add new instance](#)

41. In the left-side navigation, click the name of the API to return to its main page.

42. Locate the API instances now associated asset version 1.0.1; you should see the new label for the API instance.

The screenshot shows a table titled "Asset versions for v1". It has two columns: "Version" and "Instances". There are two rows. The first row is for version 1.0.1 and contains two instances: "Mocking Service" and "Sandbox - No policy". The second row is for version 1.0.0 and contains one instance: "Sandbox - No policy". Each instance has a three-dot menu icon to its right.

Version	Instances
1.0.1	Mocking Service Sandbox - No policy
1.0.0	Sandbox - No policy

43. Click the GET method for the flights resource.

44. In the API console, click the drop-down arrow next to Mocking Service; you should now see your API proxy instance as a choice.

45. Select the Sandbox - No policy instance.

46. Click the Send button; you should now see the real data from the database, which contains multiple flights.

The screenshot shows an API console interface. At the top, there is a green "GET" button. Below it, a dropdown menu is set to "Sandbox - No policy". The URL field contains "http://training-american-api0.cloudbhub.io/flights". Below the URL, there are tabs for "Parameters" and "Headers", with "Parameters" being active. A "Query parameters" section includes a checkbox labeled "Show optional parameters". At the bottom, there is a blue "Send" button. The response section shows a green "200 OK" status and "9163.95 ms" latency. A "Details" link is available. Below this, there are download and copy icons. The response body is shown as a JSON array:

```
[Array[11]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX".
```

47. Make several more calls to this endpoint.

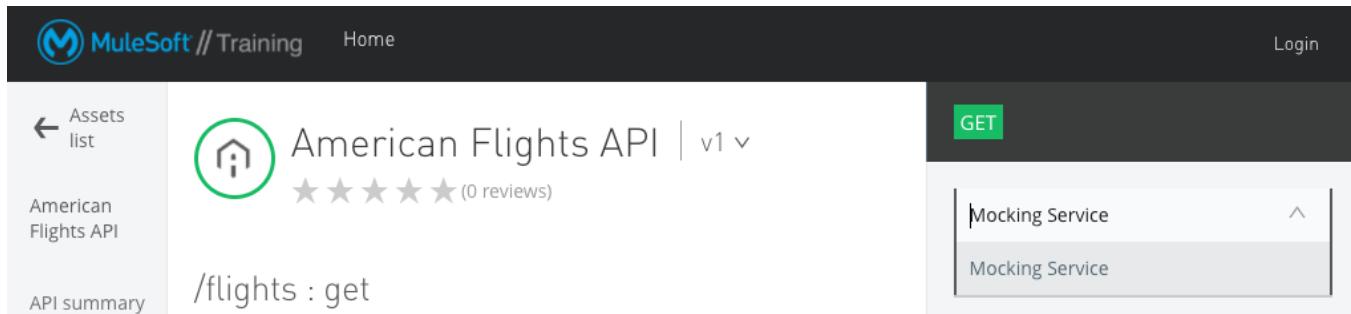
48. Make calls to different methods.

Make requests to the API proxy from the public portal

49. Return to the public portal in the private/incognito window.

50. Click the GET method for the flights resource.

51. In the API reference section, click the drop-down arrow next to Mocking Service; you should NOT see your API proxy instance as a choice.

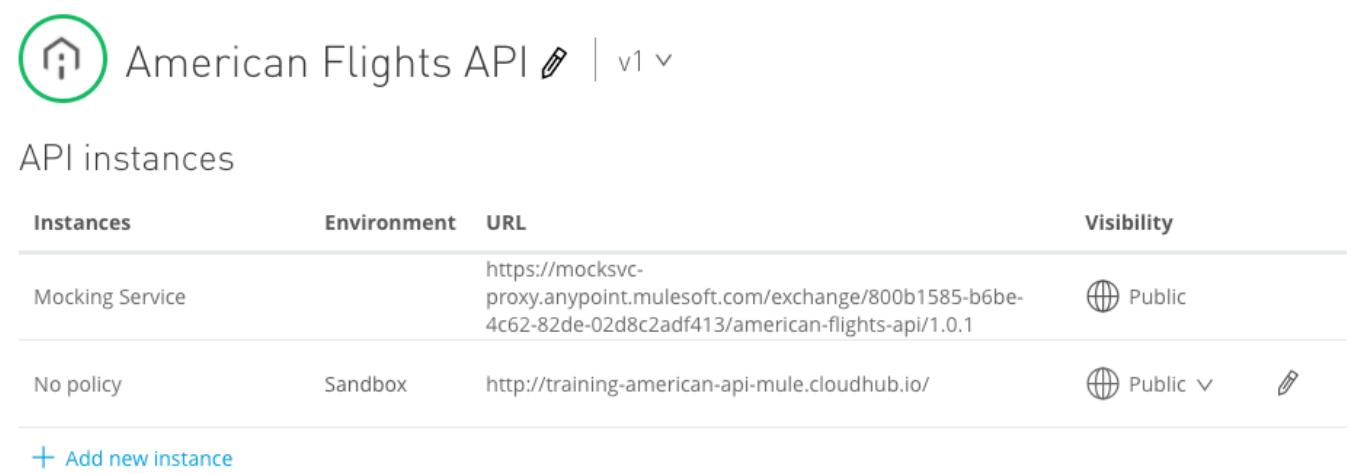


Make an API instance visible in the public portal

52. Return to the browser with Exchange.

53. In the left-side navigation, click API instances.

54. Change the visibility of the No policy instance from private to public.



Instances	Environment	URL	Visibility
Mocking Service		https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1	Public
No policy	Sandbox	http://training-american-api-mule.cloudhub.io/	Public

55. Return to the public portal in the private/incognito window.

56. Refresh the page.

57. In the API console, change the API instance from Mocking Service to Sandbox - No policy.

58. Click Send; you should get data.

The screenshot shows the API console interface for a GET request. At the top, there is a green 'GET' button. Below it, the instance is set to 'Sandbox - No policy'. The URL is listed as <http://training-american-api0.cloudhub.io/flights>. There are tabs for 'Parameters' and 'Headers', with 'Parameters' being active. A checkbox labeled 'Show optional parameters' is present. A large blue 'Send' button is centered below the parameters. The response section shows a green '200 OK' button and a latency of '1476.04 ms'. A 'Details' link is available. Below this, there are icons for copy, download, and refresh. The response body is displayed as an array of 11 elements, with the first element showing fields 'ID' and 'code'.

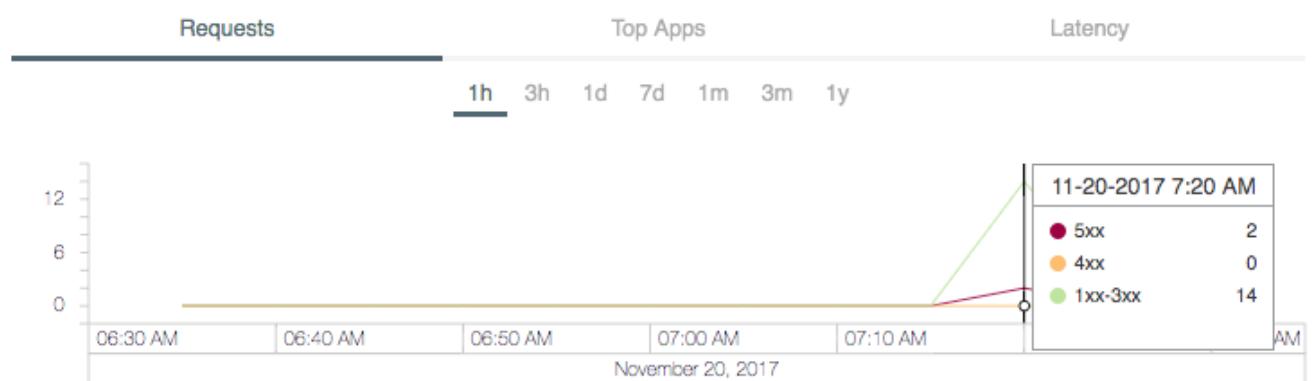
```
[Array[11]
-0: {
  "ID": 1,
  "code": "free0001"}
```

Look at the API request data

59. Return to the browser tab with API Manager.

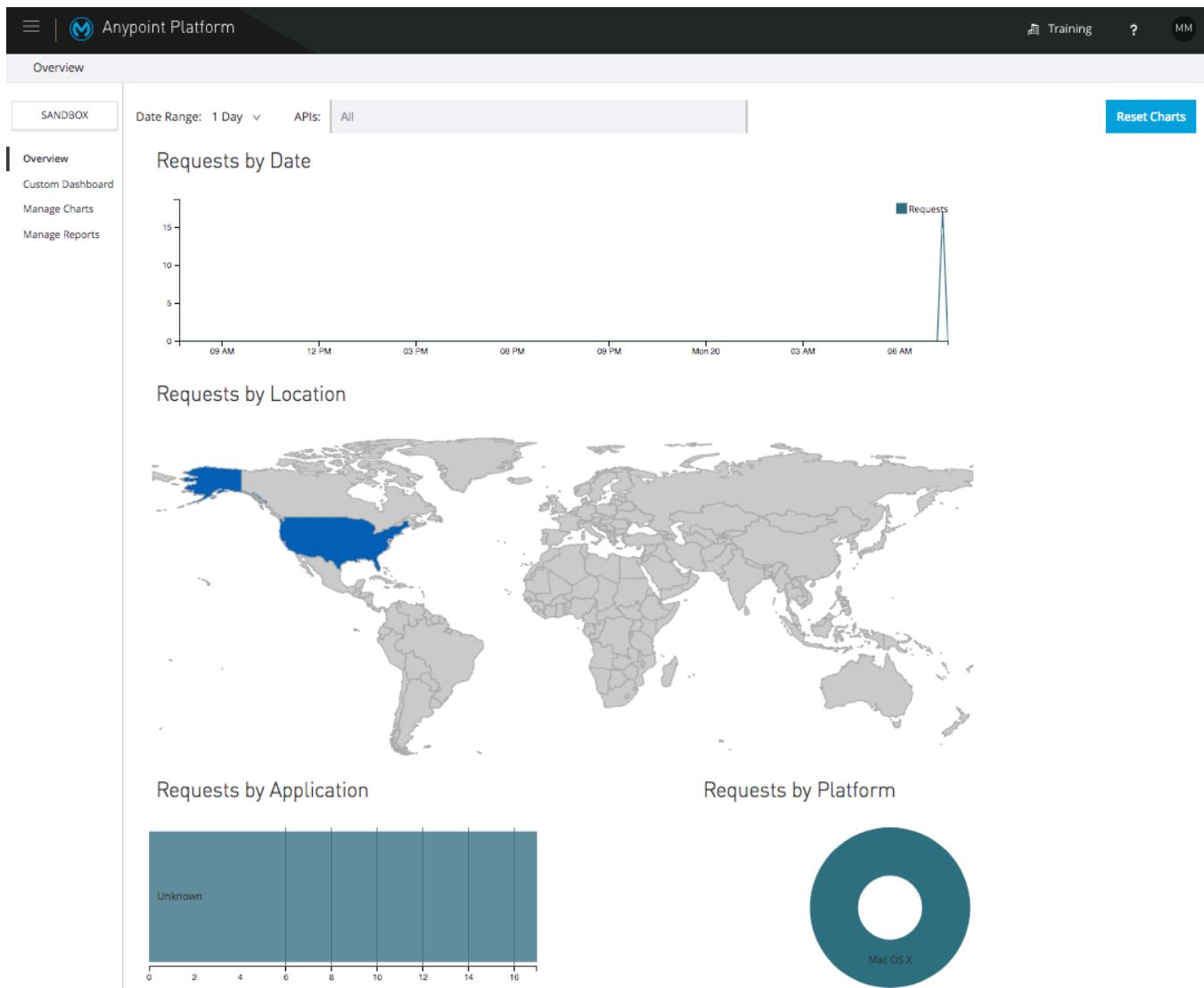
60. Refresh the Settings page for your American Flights API.

61. Look at the Request chart again; you should now see data for some API calls.



62. Click the View Analytics Dashboard link located in the upper-right corner.

63. Review the data in the dashboard.



64. Close the browser tab.

Walkthrough 5-4: Restrict API access with policies and SLAs

In this walkthrough, you govern access to the API proxy. You will:

- Add and test a rate limiting policy.
- Add SLA tiers, one with manual approval required.
- Add and test a rate limiting SLA policy.
- Request application access to SLA tiers from private and public API portals.
- Approve application requests to SLA tiers in API Manager.

The screenshot shows the API Manager interface with the title "API Administration (Sandbox) / American Flights API (v1) - Client Applications". On the left, there's a navigation sidebar with links for API Administration, Alerts, Client Applications (which is selected), Policies, SLA Tiers, and Settings. The main content area displays two client applications: "Training external app" and "Training internal app".

Application	Current SLA tier	Requested SLA tier	Status
Training external app	Silver	N/A	Approved
> Training internal app	Free	N/A	Approved

For the "Training external app", detailed information is shown in a table:

Owners	Max Mule jeanette.stallons@mulesoft.com	Submitted	3 minutes ago
Client ID	2f38b1ca94814b4aad02aab4e0c995a1	Approved	a few seconds ago
URL	None	Rejected	-
Redirect URIs	None	Revoked	-

Create a rate limiting policy

1. Return to the Settings page for your American Flights API in Anypoint Manager.
2. In the left-side navigation, click the Policies link.

The screenshot shows the API Manager interface with the title "API Administration (Sandbox) / American Flights API (v1) - Policies". On the left, there's a navigation sidebar with links for API Administration, Alerts, Client Applications, Policies (which is selected), SLA Tiers, and Settings. The main content area displays details for the "American Flights API v1".

API Status: Active Asset Version: 1.0.1 Type: RAML/OAS
Implementation URL: <http://training-american-ws-mule.cloudhub.io/api>
Consumer endpoint: <http://training-american-api-mule.cloudhub.io/>

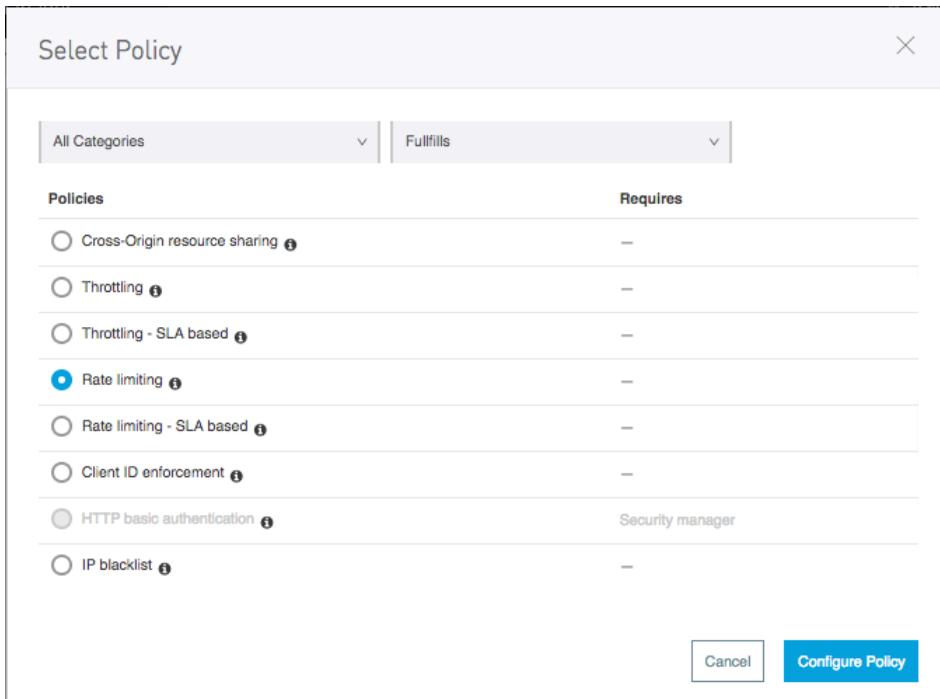
Actions: Manage CloudHub Proxy >, View API in Exchange >, View configuration details >, View Analytics Dashboard >

Buttons: Apply New Policy

Message: There are no applied policies.

3. Click the Apply New Policy button.

4. In the Select Policy dialog box, select Rate limiting.



5. Click Configure Policy.

6. On the Apply Rate limiting policy page, set the following values and click Apply:

- # of Reqs: 3
- Time Period: 1
- Time Unit: Minute
- Methods & Resource conditions: Apply configurations to all API methods & resources

Apply Rate limiting policy

Specifies the maximum value for the number of messages processed per time period, and rejects any messages beyond the maximum. Applies rate limiting to all API calls, regardless of the source.

Limits *

List of maximum requests limits allowed per time period.

# of Reqs *	Time Period *	Time Unit *
3	1	Minute

[+ Add Limit](#)

Method & Resource conditions

- Apply configurations to all API methods & resources
- Apply configurations to specific methods & resources

Cancel	Apply
--------	-------

7. Click Apply; you should see the policy listed for your API.

The screenshot shows the API Manager interface. The left sidebar has a 'Policies' tab selected. The main area displays the 'American Flights API v1' policies. A table lists one policy:

Name	Category	Fulfils
> Rate limiting ⓘ	Quality of service	Baseline Rate Limiting

Buttons include 'Apply New Policy' and 'Edit policy order'.

8. In the left-side navigation, click Settings.
9. Change the API instance label to Rate limiting policy.

API Instance ⓘ

ID: 5831632

Label: Rate limiting policy

Test the new rate limiting policy

10. Return to the browser tab with your American Flights API in Exchange.
11. Return to the page with the API console for the flights:/GET resource.
12. Select the Sandbox – Rate limiting policy API instance.

Note: You may need to refresh the page to see the new label for the API instance.

13. Press Send until you get a 429 Too Many Requests response.

The screenshot shows a browser interface for a GET request to 'http://training-american-api0.cloudhub.io/flights'. The request details include 'Sandbox - Rate limiting policy' and optional parameters. The 'Send' button is visible. The response is an orange box indicating a '429 Too Many Requests' error with a latency of '118.05 ms'. Below the error message are icons for refresh, download, and copy, followed by the text 'API calls exceeded'.

Create SLA tiers

14. Return to the browser tab with your American Flights API in API Manager.
15. In the left-side navigation, click the SLA Tiers link.
16. Click the Add SLA tier button.

The screenshot shows the API Manager interface for the 'American Flights API (v1) - SLA Tiers' page. The left sidebar has 'SLA Tiers' selected. The main area displays the API details: 'American Flights API v1', 'API Status: Active', 'Asset Version: 1.0.1', 'Type: RAML/OAS', 'Implementation URL: http://training-american-ws-mule.cloudhub.io/api', and 'Consumer endpoint: http://training-american-api-mule.cloudhub.io/'. There are buttons for 'Actions' and 'Add SLA tier'. A search bar and a note 'There are no SLA tiers for this API version.' are also present.

17. In the Add SLA tier dialog box, set the following values:

- Name: Free
- Approval: Automatic
- # of Reqs: 1
- Time Period: 1
- Time Unit: Minute

The screenshot shows the 'Add SLA tier' dialog box. It has sections for 'Name *' (containing 'Free'), 'Description' (containing 'Description'), 'Approval *' (containing 'Automatic'), and 'Limits'. Under 'Limits', there are fields for '# of Reqs *' (1), 'Time Period *' (1), and 'Time Unit *' (Minute). A dropdown arrow is shown next to the time unit field. Below these fields is a checkbox labeled 'visible' which is checked, and a trash can icon. At the bottom left is a blue 'Add Limit' button, and at the bottom right are 'Cancel' and 'Add' buttons.

18. Click the Add button.

19. Create a second SLA tier with the following values:

- Name: Silver
- Approval: Manual
- # of Reqs: 1
- Time Period: 1
- Time Unit: Second

Name	Limits	Applications	Status	Approval	
Free	1	0	Active	Auto	<button>Edit</button> <button>Delete</button>
Silver	1	0	Active	Manual	<button>Edit</button> <button>Delete</button>

Change the policy to rate limiting – SLA based

20. In the left-side navigation, click the Policies link.
21. Expand the Rate limiting policy.
22. Click the Actions button and select Remove.

Name	Category	Fulfils	Requires
Rate limiting <small>i</small>	Quality of service	Baseline Rate Limiting	

Order	Method	Resource URI	
	All API Methods	All API Resources	View Detail Actions ▾ Disable Edit Remove

23. In the Remove policy dialog box, click Remove.
24. Click the Apply New Policy button.
25. In the Select Policy dialog box, select Rate limiting - SLA based.
26. Click Configure Policy.

Select Policy

All Categories | Fulfils

Policies	Requires
<input type="radio"/> Cross-Origin resource sharing <small>i</small>	—
<input type="radio"/> Throttling <small>i</small>	—
<input type="radio"/> Throttling - SLA based <small>i</small>	—
<input type="radio"/> Rate limiting <small>i</small>	—
<input checked="" type="radio"/> Rate limiting - SLA based <small>i</small>	—
<input type="radio"/> Client ID enforcement <small>i</small>	—
<input type="radio"/> HTTP basic authentication <small>i</small>	Security manager
<input type="radio"/> IP blacklist <small>i</small>	—

[Cancel](#) [Configure Policy](#)

27. On the Apply Rate limiting – SLA based policy page, look at the expressions and see that a client ID and secret need to be sent with API requests as query parameters.

The screenshot shows the MuleSoft API Manager interface. The top navigation bar includes 'API Manager', 'Organization', a help icon, and a user profile icon. Below the navigation, the breadcrumb path shows 'API Administration / American Flights API (1.0) - Policies / Apply Rate limiting - SLA based policy'. The main content area is titled 'Apply Rate limiting - SLA based policy'. It contains a description: 'Specifies the maximum value for the number of messages processed per time period, and rejects any messages beyond the maximum.' Below this is a note: 'This policy will require updates to the RAML definition in order to function. You can obtain the RAML snippet and learn more [here](#)'. There are two sections for expressions: 'Client ID Expression *' with the value '#[message.inboundProperties['http.query.params']['client_id']]', and 'Client Secret Expression *' with the value '#[message.inboundProperties['http.query.params']['client_secret']]'. Under 'Method & Resource conditions', there are two radio buttons: 'Apply configurations to all API methods & resources' (selected) and 'Apply configurations to specific methods & resources'. At the bottom right are 'Cancel' and 'Apply' buttons.

28. Click Apply.
29. In the left-side navigation, click Settings.
30. Change the API instance label to Rate limiting – SLA based policy.

The screenshot shows the 'API Instance' settings page. It displays the 'ID: 5831632' and the 'Label: Rate limiting - SLA based policy' field, which has a pencil icon indicating it is editable.

Test the rate limiting – SLA based policy in Exchange

31. Return to the browser tab with your API in Exchange.
32. Refresh the page and select to make a call to the Sandbox – Rate limiting – SLA based policy.

33. Click Send; you should get a 401 response with a message that a client_id could not be retrieved from the message.

The screenshot shows a 'GET' request to 'http://training-american-api0.cloudhub.io/flights'. The 'Parameters' tab is selected. A 'Send' button is present. The response is a 401 Unauthorized error with a duration of 136.52 ms. The error message is 'Unable to retrieve client_id from message'.

Request access to the API as an internal consumer

34. In the left-side navigation, click the name of the API to return to its home page.

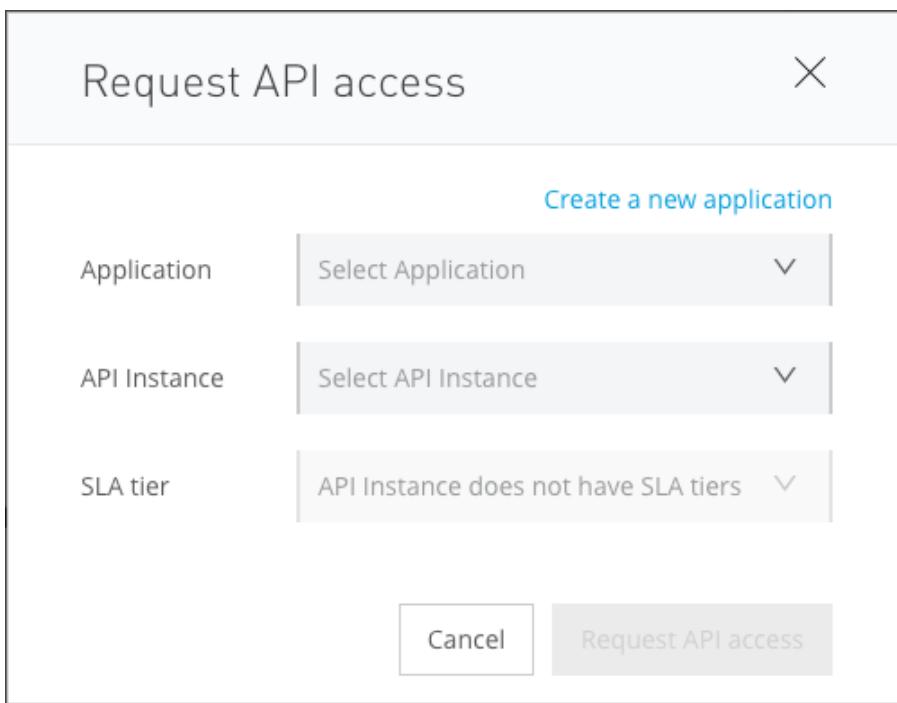
35. Click the more options button in the upper-right corner and select Request access.

The screenshot shows the 'American Flights API' page. The 'Request access' button is highlighted in the top right corner of the main content area.

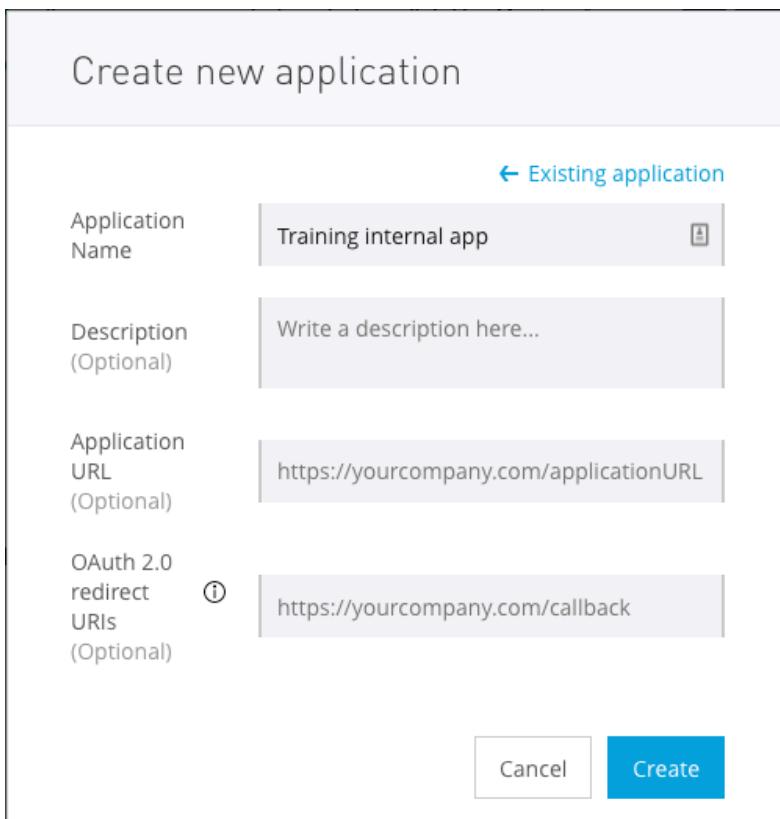
Note: Other internal users that you shared the API with that do not have Edit permissions will see a different menu.

The screenshot shows the 'American Flights API' page with a simplified menu in the top right corner, containing 'Download' and 'Request access'.

36. In the Request API access dialog box, click the Create a new application link.



37. In the Create new application dialog box, set the name to Training internal app and click Create.



38. In the Request API access dialog box, set the API instance to Sandbox – Rate limiting SLA - based policy.
39. Set the SLA tier to Free.

The dialog box has a title 'Request API access' at the top. Below it is a 'Create a new application' link. There are three dropdown menus:

- 'Application' set to 'Training internal app'
- 'API Instance' set to 'Sandbox - Rate limiting - SLA based ...'
- 'SLA tier' set to 'Free'

A table below shows the configuration:

# of Reqs	Time period	Time Unit
1	1	Minute

At the bottom are 'Cancel' and 'Request API access' buttons.

40. Click Request API access.
41. In the Request API access dialog box, view the assigned values for the client ID and client secret.

The dialog box has a title 'Request API access' and a close button. It displays a success message: '✓ API access has been successful!'. It shows the assigned values for Client ID and Client secret:

- 'Client ID' is e708026bb0cf4c3e8594ca39138b9a00
- 'Client secret' is e10A1faF382D47DEA6A90cA8d4FC3891

A note at the bottom says 'Application details have been opened in a new tab.' A 'Close' button is at the bottom right.

42. Click Close.

Request access to the API as an external consumer

43. Return to the public portal in the private/incognito window.
44. Refresh the page for the American Flights API; you should now see a Request access button.

The screenshot shows the MuleSoft // Training interface. At the top, there's a navigation bar with the MuleSoft logo, 'MuleSoft // Training', 'Home', and 'Login'. Below the navigation, there's a sidebar with 'Assets list' and 'American Flights API'. The main content area displays the 'American Flights API | v1' with a green house icon, a 5-star rating '(0 reviews)', and a 'Request access' button. To the right, there's a 'GET' button and a dropdown menu set to 'Mocking Service'. The 'Request access' button is highlighted with a red box.

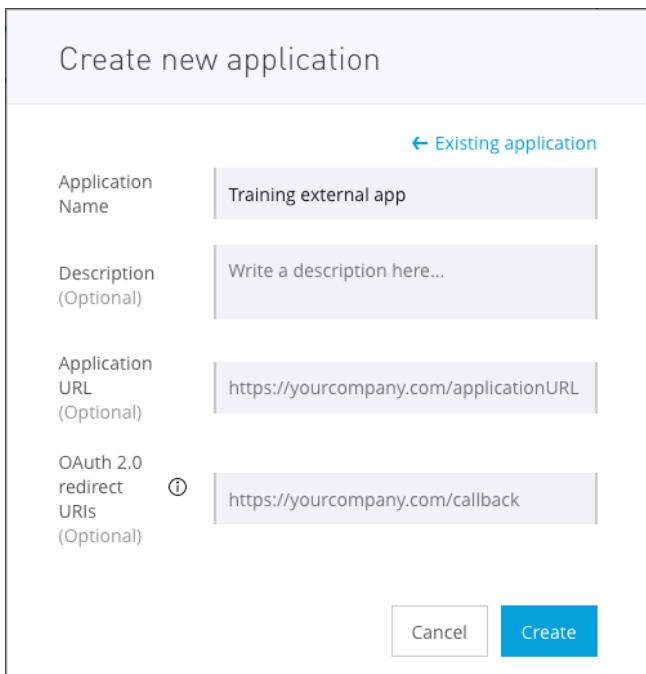
45. Click the Request access button; you should get a page to sign in or create an Anypoint Platform account.
46. Enter your existing credentials and click Sign in.

Note: Instead of creating an external user, you will just use your existing account.

The screenshot shows the Anypoint Platform sign-in dialog. It has two main sections: a light gray 'Sign in' section on the left and a dark gray 'Don't have an account?' section on the right. The 'Sign in' section contains fields for 'Username' (maxmule00) and 'Password' (redacted). Below these is a 'Sign in' button. The 'Don't have an account?' section contains the text 'Create one for free.' and a 'Sign up' button. At the bottom of the dialog, there are links for 'Forgot sign-in credentials?' and 'Privacy policy'.

47. Back in the public portal, click the Request access button again.
48. In the Request API access dialog box, click the Create a new application button

49. In the Create new application dialog box, set the name to Training external app and click Create.

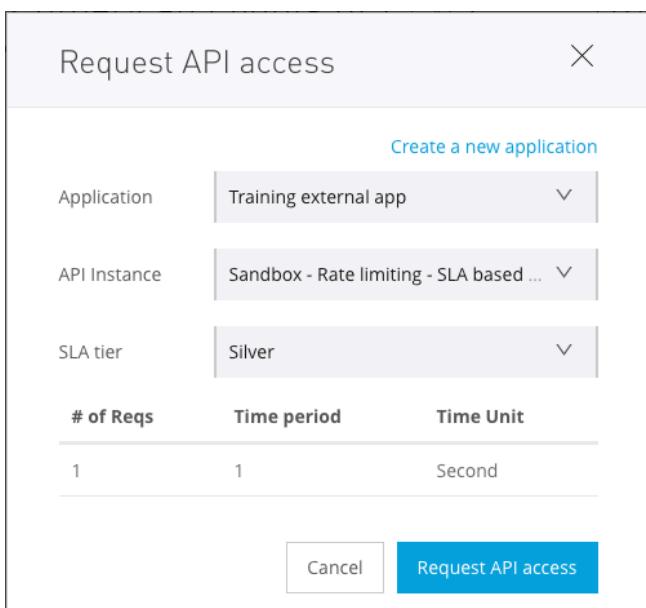


The dialog box has a title 'Create new application' and a back button 'Existing application'. It contains four input fields: 'Application Name' (Training external app), 'Description (Optional)' (Write a description here...), 'Application URL (Optional)' (https://yourcompany.com/applicationURL), and 'OAuth 2.0 redirect URIs (Optional)' (https://yourcompany.com/callback). At the bottom are 'Cancel' and 'Create' buttons.

50. Click Create.

51. In the Request API access dialog box, set the API instance to Sandbox – Rate limiting SLA-based policy.

52. Set the SLA tier to Silver.



The dialog box has a title 'Request API access' and a close button 'X'. It includes a 'Create a new application' link and three dropdown menus: 'Application' (Training external app), 'API Instance' (Sandbox - Rate limiting - SLA based ...), and 'SLA tier' (Silver). Below these are three input fields: '# of Reqs' (1), 'Time period' (1), and 'Time Unit' (Second). At the bottom are 'Cancel' and 'Request API access' buttons.

53. Click Request API access.

54. In the Request API access dialog box, click Close.

55. In the portal main menu bar, right-click My applications and select to open it in a new tab; you should see the two applications you created.

The screenshot shows the 'My applications' page in the MuleSoft // Training portal. At the top, there is a navigation bar with the MuleSoft logo, the text 'MuleSoft // Training', 'Home', and 'My applications'. Below the navigation bar, there is a breadcrumb trail 'Assets list' and a search bar with the placeholder 'Search'. The main content area is titled 'My applications' and contains a table with two rows. The table has columns for 'Name' and 'Description'. The first row is for 'Training internal app' and the second row is for 'Training external app'.

Name	Description
Training internal app	
Training external app	

56. Click the link for Training external app; you should see what APIs the application has access to, values for the client ID and secret to access them, and request data.

The screenshot shows the details page for the 'Training external app'. At the top, there is a navigation bar with the MuleSoft logo, the text 'MuleSoft // Training', and a user icon. Below the navigation bar, there is a breadcrumb trail 'My applications' and a title 'Training external app'. On the right side, there are three buttons: 'Edit', 'Reset client secret', and 'Delete'. On the left, there is a sidebar with a list of items: 'Show All (1)' (selected), 'Sandbox - Rate limiti...', and 'v1'. On the right, there is a detailed view of the application's configuration, including 'Application Description:', 'Application URL:', 'Redirect URIs:', 'Client ID: 2f38b1ca94814b4aad02aab4e0c995a1', 'Client Secret: Show', and 'Grant Types: -'.

57. Leave this page open in a browser so you can return to it and copy these values.

Approve the application requests

58. Return to the browser window and tab with the Settings page for American Flights API (v1) in API Manager.

59. In the left-side navigation, click Client Applications; you should see the two applications that requests access to the API.

The screenshot shows the API Manager interface with the following details:

- Header:** API Manager, Training, MM
- Breadcrumbs:** API Administration (Sandbox) / American Flights API (v1) - Client Applications
- Left Sidebar:** SANDBOX, API Administration, Alerts, **Client Applications**, Policies, SLA Tiers, Settings
- Page Title:** American Flights API v1
- API Status:** Active, Asset Version: 1.0.1, Type: RAML/OAS
- Implementation URL:** <http://training-american-ws-mule.cloudhub.io/api>
- Consumer endpoint:** <http://training-american-api-mule.cloudhub.io/>
- Actions:** Manage CloudHub Proxy, View API in Exchange, View configuration details, View Analytics Dashboard
- Search Bar:** Q Search
- Pagination:** 1 - 2 of 2
- Table:** Shows two rows of client application requests.

Application	Current SLA tier	Requested SLA tier	Status	Action Buttons
> Training external app	N/A	Silver	Pending	Approve, Reject, Delete
> Training internal app	Free	N/A	Approved	Revoke

60. Click the Approve button for the application requesting Silver tier access.

61. Expand the Training external app row and review its information.

Application	Current SLA tier	Requested SLA tier	Status	Action Buttons
> Training external app	Silver	N/A	Approved	Revoke
Owners	Max Mule jeanette.stallons@mulesoft.com		Submitted	3 minutes ago
Client ID	2f38b1ca94814b4aad02aab4e0c995a1		Approved	a few seconds ago
URL	None		Rejected	-
Redirect URLs	None		Revoked	-
> Training internal app	Free	N/A	Approved	Revoke

Try to test the rate limiting – SLA based policy from an API portal

62. Return to the browser window and tab with the API console in the public portal.
63. Try again to make a call to the Sandbox – Rate limiting – SLA based policy; you should still get a 401 response with a message that a client_id could not be retrieved from the message.

Note: There is currently no way to pass the client authentication parameters needed to call this API from an API portal; you will add this in the next walkthrough. You could, however, use

another tool like Postman to make calls to the API, explicitly passing client_id and client_secret query parameters as you did in module 2.

The screenshot shows a browser window with a dark header bar containing a green 'GET' button. Below the header, the URL 'http://training-american-api-mule.cloudhub.io/f' is visible. Underneath the URL, there are two tabs: 'Parameters' (which is selected) and 'Headers'. In the 'Parameters' section, there is a 'Query parameters' dropdown menu with 'destination' selected. A blue 'Send' button is located below the dropdown. At the bottom of the window, an orange box displays the error details: '401 Unauthorized' with '129.59 ms' and a 'Details' link. Below the error box, there are several icons: a checkbox, a download arrow, a copy icon, and an eye icon. A message box at the bottom states 'Unable to retrieve client_id from message'.

64. Close this browser window.

Walkthrough 5-5: Make calls to an API with a client ID based policy from API portals

In this walkthrough, you add the ability to make calls to APIs with client ID enforcement policies from internal and public API portals. You will:

- Add authentication query parameters to an API specification.
- Update a managed API to use a new version of an API specification.
- Call a governed API with client credentials from API portals.

The screenshot shows a REST client interface. On the left, a 'GET' button is highlighted. Below it, the URL is `http://training-american-api-mule.cloudhub.io/flight`. Under 'Parameters', there are two sections: 'Query parameters' and 'Headers'. In 'Query parameters', 'client_id*' is set to `e708026bb0cf4c3e8594ca39138b9a00` and 'client_secret*' is set to `e10A1faF382D47DEA6A90cA8d4FC3891`. A 'Send' button is at the bottom. On the right, the response is shown: a 200 OK status with a duration of 1486.61 ms. The response body is an array of 11 flight objects, each with fields like ID, code, price, departure date, origin, destination, and seat availability.

```
[Array[11]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 787",
    "totalSeats": 200
  }
},
-1: {
  "ID": 2,
  "code": "eefd0123",
  "price": 300
}]
```

Add authentication query parameters to the API specification

1. Return to the browser tab with the Settings page for American Flights API (v1) in API Manager.
2. In the left-side navigation, click the Policies link.
3. Click the RAML snippet link for the rate limiting – SLA based policy.



4. In the RAML snippet for Rate limiting – SLA based dialog box, select RAML 1.0.

5. Copy the value for the traits.

RAML snippet for Rate limiting - SLA based

RAML 0.8 RAML 1.0

Client ID based policies by default expect to obtain the client ID and secret as query parameters. To enforce this in the API definition a trait can be defined in RAML as shown below.

```
traits:  
  client-id-required:  
    queryParameters:  
      client_id:  
        type: string  
      client_secret:  
        type: string
```

This trait must then be applied to the resource or methods using the `is` RAML attribute.

```
/products:  
  get:  
    is: [client-id-required]  
    description: Gets a list of all the inventory products.
```

Please read [Applying Resource Types and Traits](#) section on RAML documentation for more information.

[Close](#)

6. Click Close.

7. Return to the browser tab with your API in Design Center.

8. Go to a new line after the types declaration and paste the traits code you copied.

```
1  #%RAML 1.0  
2  version: v1  
3  title: American Flights API  
4  
5  types:  
6    AmericanFlight: !include exchange_modu  
7  
8  traits:  
9    client-id-required:  
10      queryParameters:  
11        client_id:  
12          type: string  
13        client_secret:  
14          type: string  
15  
16 /flights:
```

9. Go to a new line after the flights resource declaration and add is: [].

```
16 /flights:  
17   is: []  
18   get:
```

10. Place the cursor inside the array brackets and in the shelf, click client-id-required.

```
16 /flights:  
17   is: [client-id-required]  
18   get:
```

11. Repeat this process so the trait is applied to all methods of the {ID} resource as well.

```
45 /{ID}:  
46   is: [client-id-required]  
47   get:
```

Test the authentication query parameters in API console

12. In the API console, turn on the mocking service.

13. Select to try any one of the resources; you should now see text fields to enter client_id and client_secret query parameters.

The screenshot shows the MuleSoft API Console interface. At the top, there's a header with a shield icon and the text "Mocking service: —✓". Below that is a "Request URL" field containing "https://mocksvc.mulesoft.com/mocks". Underneath the URL, there are two tabs: "Parameters" (which is currently selected) and "Headers". The "Parameters" section has a heading "Query parameters" and a checkbox labeled "Show optional parameters". There are two input fields: "client_id*" and "client_secret*". At the bottom right of the form is a blue "Send" button.

14. Enter any values for the client_id and client_secret and click Send; you should get a 200 response with the example results.

The screenshot shows the MuleSoft Anypoint Platform Mocking service interface. At the top, it says "Mocking service: —✓". Below that is a "Request URL" field containing "https://mocksvc.mulesoft.com/mocks". Under the "Parameters" tab, there are two fields: "client_id*" with value "432" and "client_secret*" with value "765". A "Send" button is located below these fields. At the bottom, the response is shown as a green box with "200 OK" and "506.59 ms". Below the response, there are several icons: a square, a play button, a double arrow, and a list icon. The response body is displayed as JSON: [Array[2], -0: { "ID": 1, "code": "ER38sd", "price": 400 }].

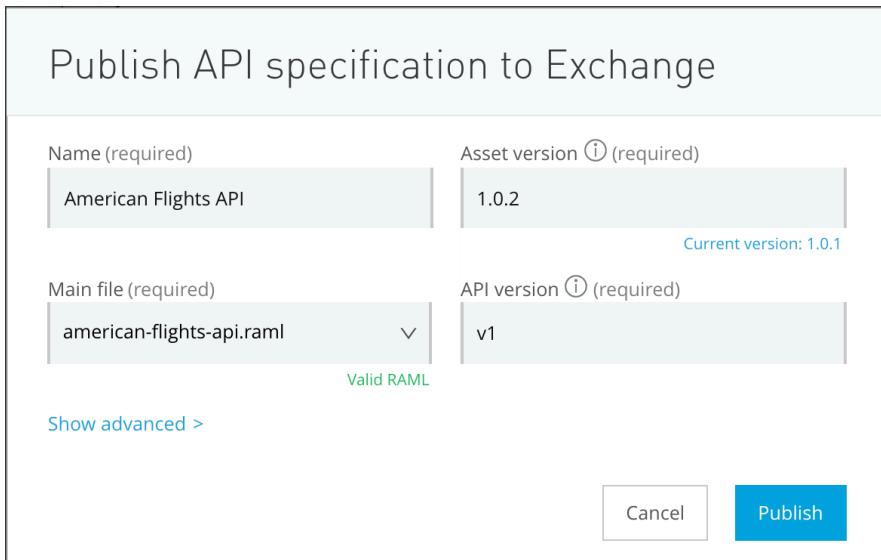
Publish the new version of the API to Exchange

15. Turn off the mocking service.

The screenshot shows the MuleSoft Anypoint Platform Mocking service interface. At the top, it says "Mocking service: X —". Below that is a "Request URL" field. The "Mocking service" status is now red with an "X" and a minus sign, indicating it is disabled.

16. Click the Publish to Exchange button.

17. In the Publish API specification to Exchange dialog box, note the asset version and click Publish.



18. After the API is published, click Done in the Publish API specification to Exchange dialog box.

Update the managed API instance to use the new version of the API specification

19. Return to browser tab with American Flights API (v1) in API Manager.
20. Locate the asset version displayed at the top of the page; you should see 1.0.1.

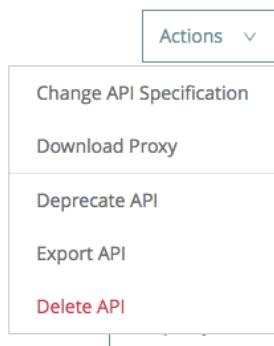
American Flights API v1

API Status: ● Active Asset Version: 1.0.1 Type: RAML/OAS

Implementation URL: <http://training-american-ws-mule.cloudhub.io/api>

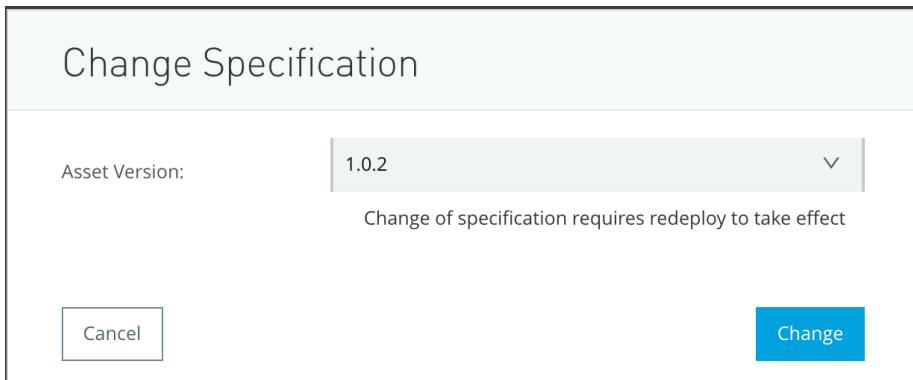
Consumer endpoint: <http://training-american-api-mule.cloudhub.io/>

21. Click the Actions button in the upper-right corner and select Change API Specification.



22. In the Change Specification dialog box, select the latest asset version, 1.0.2.

23. Click Change.



24. Wait until the API Status is green again indicating that the new proxy application has been redeployed.

American Flights API v1

API Status: ● Active Asset Version: 1.0.2 Type: RAML/OAS

Implementation URL: <http://training-american-ws-mule.cloudhub.io/api>

Consumer endpoint: <http://training-american-api-mule.cloudhub.io/>

Test the rate limiting – SLA based policy in an API portal with random client authentication values

25. Return to the browser tab with Exchange.

26. Return to the home page for the API; you should see the new asset version listed.

Asset versions for v1	
Version	Instances
1.0.2	Mocking Service
	Sandbox - Rate limiting - SLA based policy
1.0.1	
1.0.0	

27. Click the GET method for the flights resource; you should now see required text boxes to enter the required client_id and client_secret values.

The screenshot shows the configuration for a GET request to the 'Mocking Service' endpoint at <https://mocksvc-proxy.stgxdr.anypoint.mulesoft.com/exchan>. The 'Parameters' tab is selected, showing two required query parameters: 'client_id*' and 'client_secret*'. Both fields are empty. Below the fields is a 'Send' button.

28. With the Mocking Service endpoint selected, enter any values for the client_id and client_secret.

29. Click Send; you should get results.

The screenshot shows the configuration for a GET request to the 'Mocking Service' endpoint at <https://mocksvc-proxy.anypoint.mulesoft.com/exc>. The 'Parameters' tab is selected, showing two query parameters: 'client_id*' and 'client_secret*'. Both fields contain the value '432'. Below the fields is a 'Send' button.

After sending the request, the response is shown as a 200 OK status with a response time of 403.62 ms. The response body is displayed as an array of two objects:

```
[Array[2]
 -0: {
   "ID": 1,
   "code": "ER38sd".
```

30. Change the API instance to Sandbox – Rate limiting – SLA based policy and leave the random values for the authentication parameters.
31. Click Send; you should now get a 410 response with a message that the client ID is invalid.

GET

Sandbox - Rate limiting - SLA based policy ▾

http://training-american-api-mule.cloudhub.io/flig

Parameters Headers

Query parameters Show optional parameters

client_id*
432

client_secret*
432

Send

401 Unauthorized 162.26 ms Details ▾

Invalid client ID

Test the rate limiting – SLA based policy in an API portal with assigned client authentication values

32. In the left-side navigation, click Assets list.
33. In the left-side navigation, click Public Portal.
34. In the main menu, select My applications.
35. Click the Training internal application.

☰ | MuleSoft // Training

← My applications

Training internal app

Edit **Reset client secret** **Delete**

• Show All (1)

- Sandbox - Rate limiti... v1

Application Description:	Client ID: e708026bb0cf4c3e8594ca39138b9a00
Application URL:	Client Secret: Show
Redirect URIs:	Grant Types: -

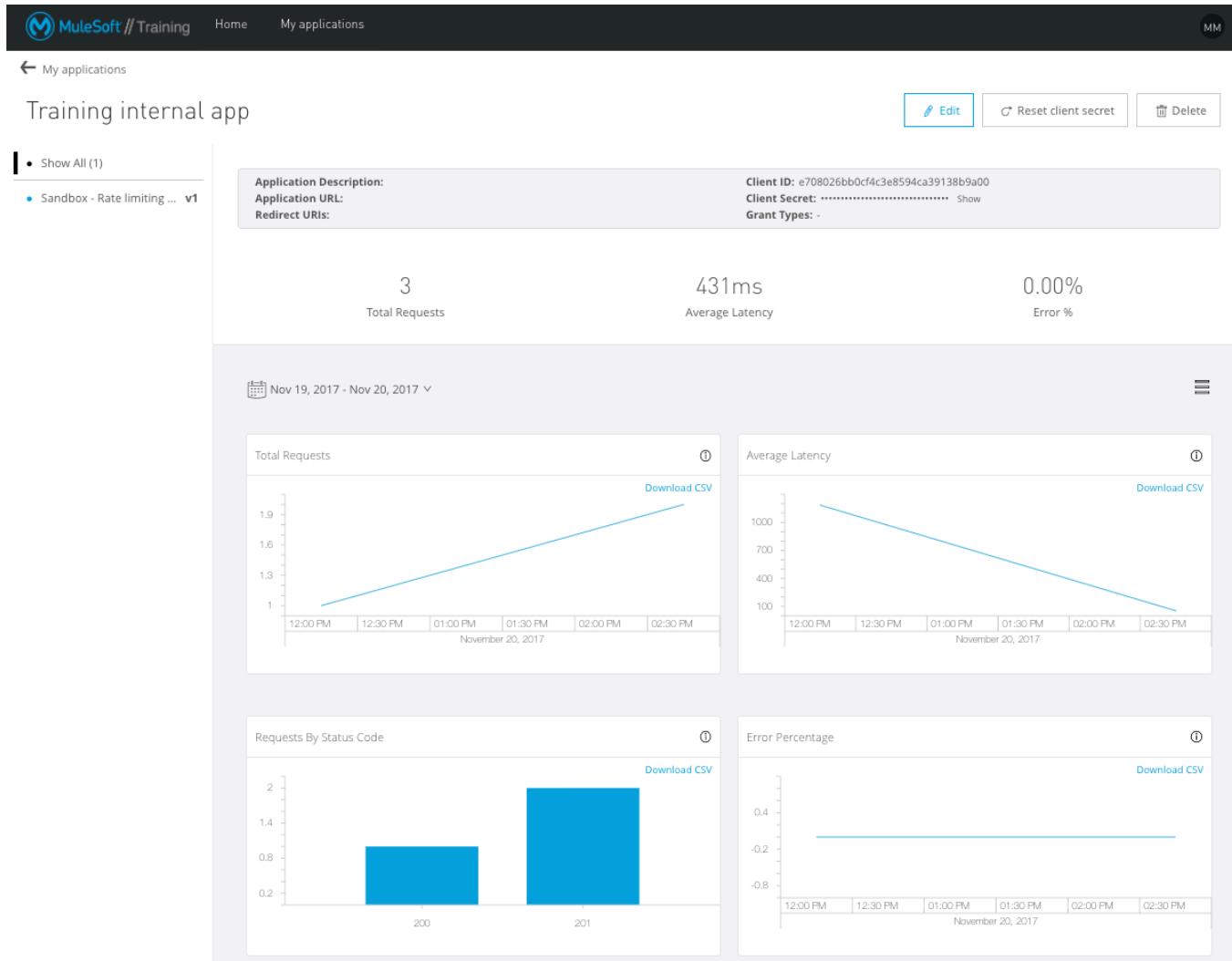
36. Copy the value of its client ID.
37. Right-click Home in the main menu bar and select to open the link in a new browser tab.
38. In that new tab, click the American Flights API.
39. Click the GET method for the flights resource.
40. In the API console, change the API instance to Sandbox – rate limiting – SLA based policy.
41. In the client_id field, paste the value of the client ID.
42. Return to the other browser tab and copy the value of the client secret.
43. Return to the other browser tab and paste the value of the client secret.
44. Click Send; you should now get a 200 response with live results.

The screenshot shows the MuleSoft API console interface. At the top, there is a green button labeled "GET". Below it, a dropdown menu shows "Sandbox - Rate limiting - SLA based policy". The URL entered is "http://training-american-api-mule.cloudhub.io/flight". There are two tabs: "Parameters" (which is selected) and "Headers". Under "Query parameters", there is a field containing "client_id*" followed by the value "e708026bb0cf4c3e8594ca39138b9a00". Another field for "client_secret*" contains the value "e10A1faF382D47DEA6A90cA8d4FC3891". At the bottom right is a blue "Send" button. The response section shows a green "200 OK" status with a timestamp of "1486.61 ms" and a "Details" link. Below the status are icons for copy, download, refresh, and grid. A preview window shows the JSON response: "[Array[11]]" with one item "-0: { "ID": 1, "code": "freeAAA1"}".

45. Make multiple calls to this method.

Review request data

46. Return to the other browser tab with the application information.
47. Change the date selection to Last Day; you should see data about your requests.



48. Close all the browser tabs with Anypoint Platform.