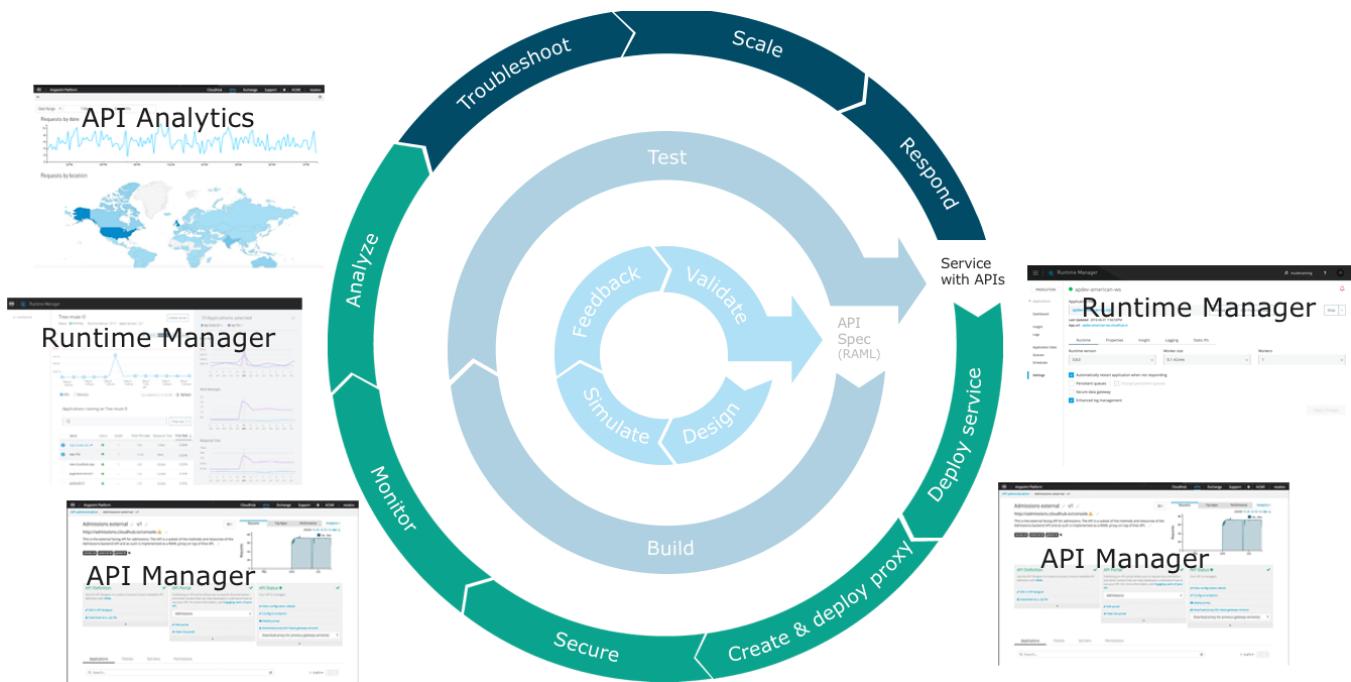


Module 4: Deploying and Managing APIs



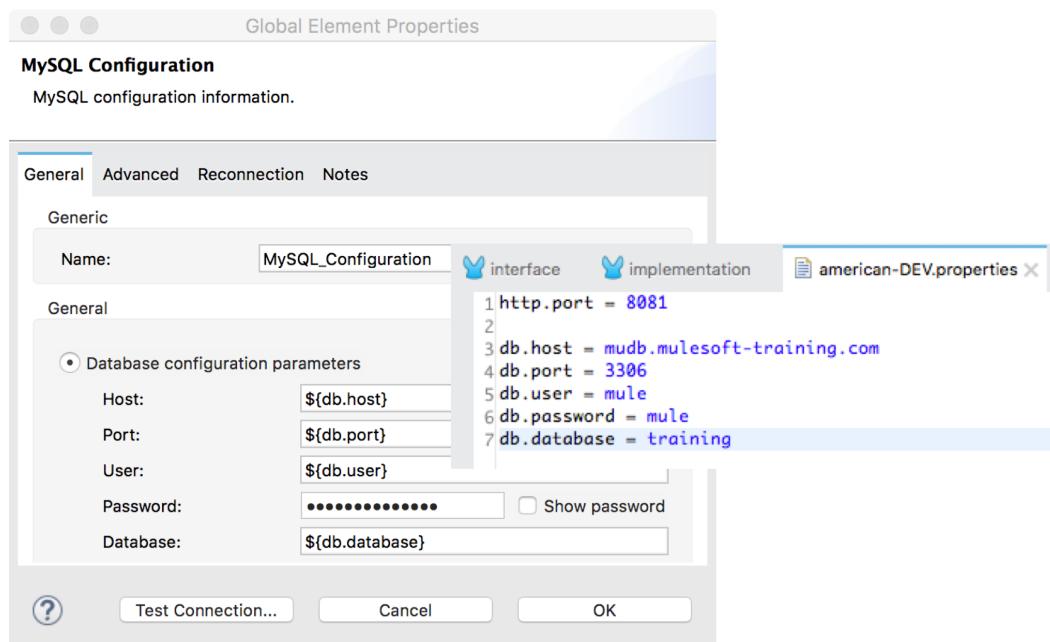
Objectives:

- Describe the options for deploying Mule applications.
- Use properties in Mule applications so they can be easily moved between environments.
- Deploy a Mule application to the cloud.
- Create and deploy a proxy for an API in the cloud.
- Restrict access to an API proxy.

Walkthrough 4-1: Prepare an API for deployment using properties

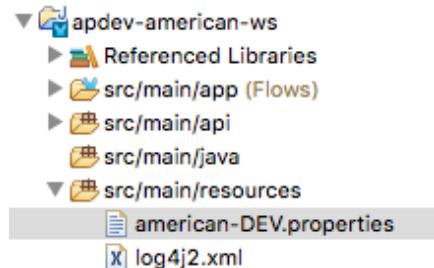
In this walkthrough, you introduce properties into your API implementation. You will:

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



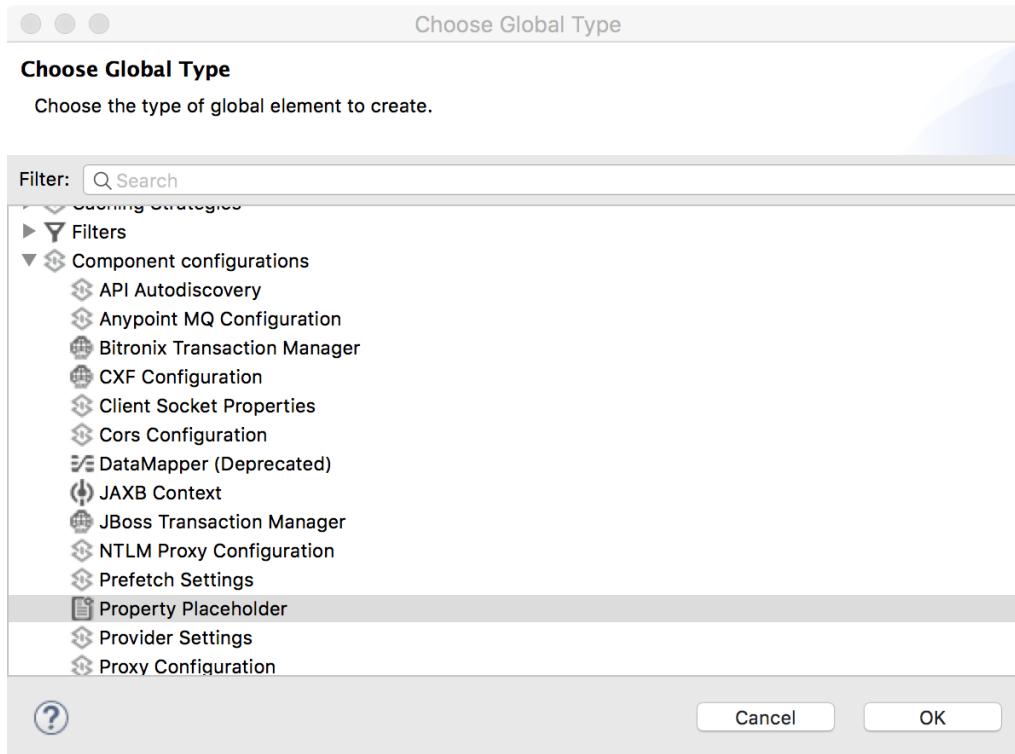
Create a properties file

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.

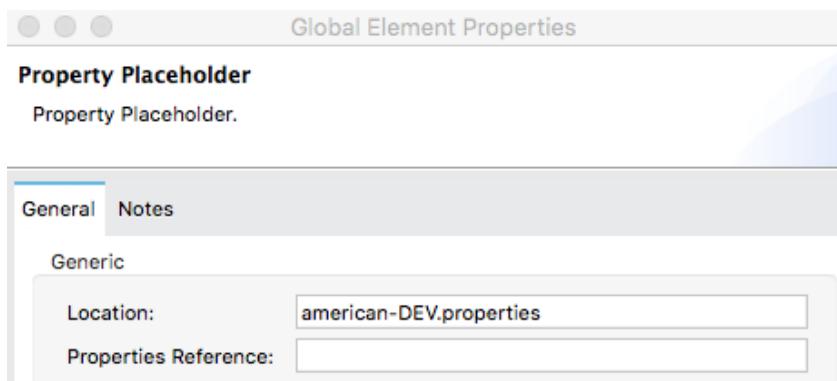


Create a Properties Placeholder global element

3. Return to interface.xml.
4. Click the Global Elements link at the bottom of the canvas.
5. Click Create.
6. In the Choose Global Type dialog box, select Component configurations > Property Placeholder and click OK.

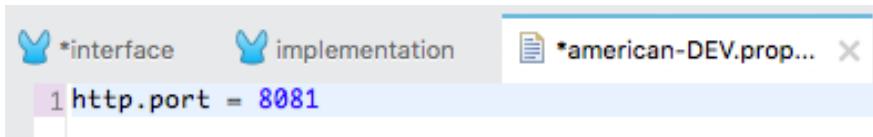


7. In the Global Element Properties dialog box, set the location to american-DEV.properties and click OK.



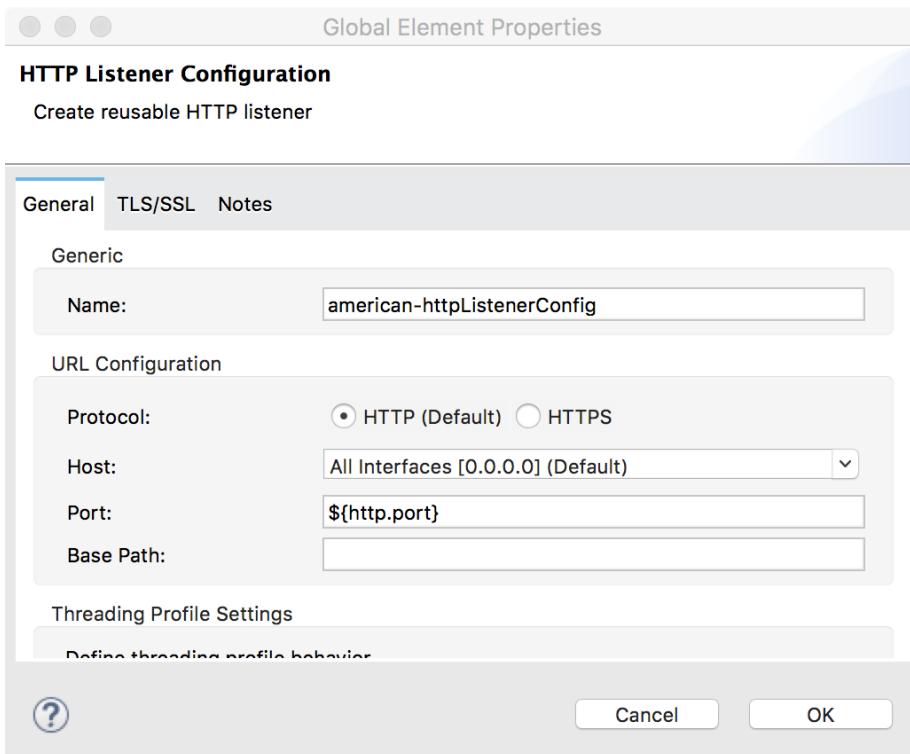
Parameterize the HTTP Listener port

8. Return to american-DEV.properties.
9. Create a property called http.port and set it to 8081.



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

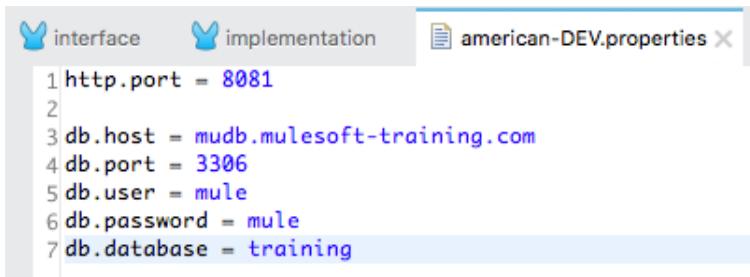
10. Return to the Global Elements view in interface.xml.
11. Double-click the HTTP Listener Configuration global element.
12. Change the port from 8081 to the application property, \${http.port}.
13. Click OK.



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.



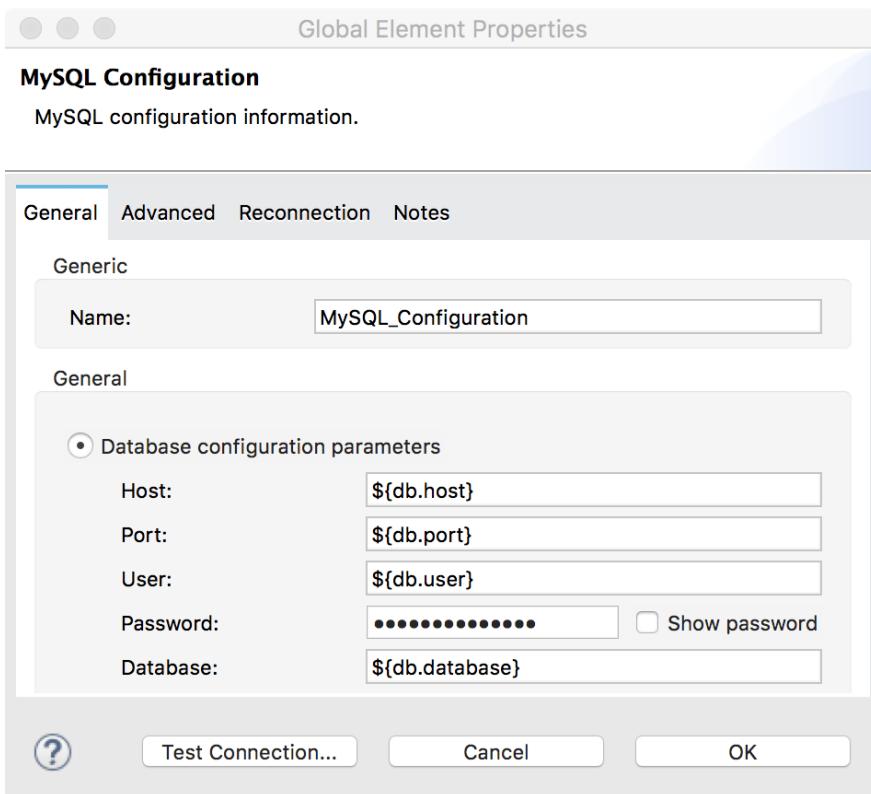
```
interface implementation american-DEV.properties
1 http.port = 8081
2
3 db.host = mudb.mulesoft-training.com
4 db.port = 3306
5 db.user = mule
6 db.password = mule
7 db.database = training
```

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

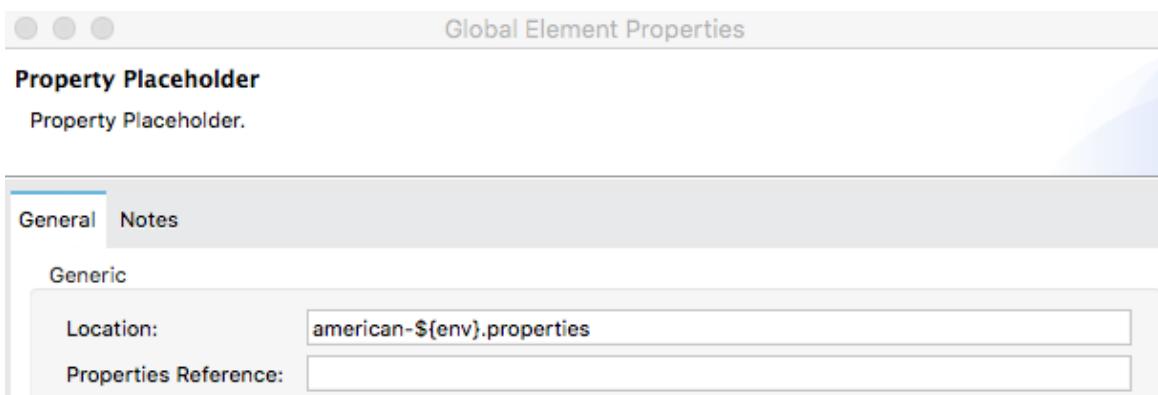
26. Return to Anypoint Studio and stop the project.
27. Open mule-app.properties located in the src/main/app folder.
28. Define a property called env and set it to DEV.



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 4-2: Deploy an application to the cloud

In this walkthrough, you deploy and run your application on Anypoint Platform in the cloud. You will:

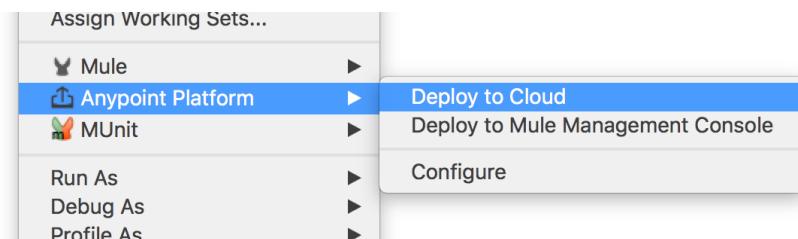
- Deploy an application from Anypoint Studio to the cloud.
- Run the application on its new, hosted domain.
- Make calls to the web service.
- Update the API implementation deployed to the cloud.

The screenshot shows the Anypoint Platform Runtime Manager. At the top, there's a navigation bar with 'Runtime Manager' and icons for 'Organization', '?', and 'MM'. Below it is a sidebar with tabs for 'PRODUCTION', 'Applications', 'Servers', and 'Alerts'. The main area has a 'Deploy application' button and a search bar. A table lists deployed applications with columns for Name, Server, Status, and File. One entry is shown: 'apdev-american-ws' on 'CloudHub' is 'Started' and its file is 'apdev-american-ws.zip'.

Name	Server	Status	File
apdev-american-ws	CloudHub	Started	apdev-american-ws.zip

Deploy the application to the cloud

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



3. At the top of the Anypoint Platform dialog box, set the application name to apdev-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. It has a 'PRODUCTION' tab selected. The application name 'apdev-american-ws' is entered, and a green checkmark icon indicates the domain is available.

4. 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 3.8.0. EE.

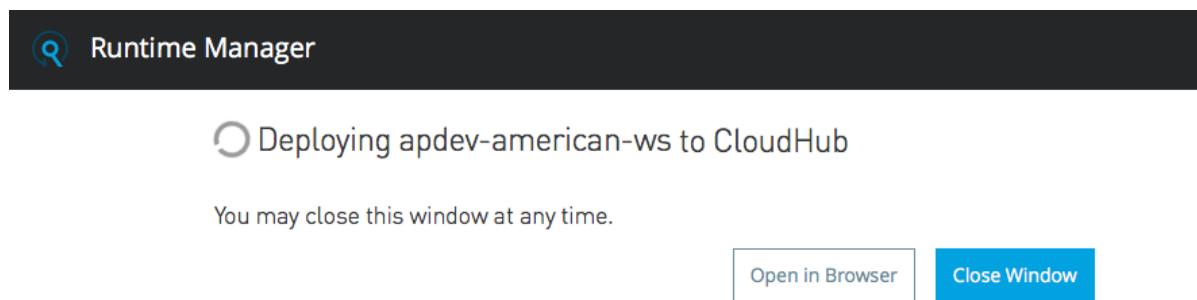
5. Set the worker size to 0.1 vCores.

Runtime	Properties	Insight	Logging	Static IPs
Runtime version	Worker size			Workers
3.8.0	0.1 vCores			1

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

Runtime	Properties	Insight	Logging	Static IPs
	Text	List		
env=DEV				

7. Click the Deploy Application button.
8. Click the Open in Browser button.



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

The screenshot shows the "Runtime Manager" interface in the Anypoint Platform. On the left, a sidebar has tabs for "PRODUCTION" (selected), "Applications", "Servers", and "Alerts". The main area has a "Deploy application" button and a search bar. A table lists the deployment status of "apdev-american-ws" to "CloudHub".

Name	Server	Status	File
apdev-american-ws	CloudHub	Deploying	apdev-american-ws.zip

Watch the logs and wait for the application to start

10. Click the name of the application; you should see information about the application appear on the right-side of the window.

The screenshot shows the Runtime Manager interface. On the left, a sidebar has 'PRODUCTION' selected under 'Applications'. The main area shows a table with columns 'Name', 'Server', 'Status', and 'File'. A single row is selected for 'apdev-american-ws' on 'CloudHub' with a status of 'Deploying' and the file 'apdev-american-ws.zip'. To the right, a detailed view for 'apdev-american-ws' is shown, including its deployment status as 'Deploying', the runtime version '3.8.0', worker size '0.1 vCores', and one worker. Buttons for 'Manage Application', 'Logs', and 'Insight' are at the bottom.

11. Click the Logs button.
12. Watch the logs as the application is deployed.
13. Wait until the application starts (or fails to start).

The screenshot shows the Runtime Manager interface with 'Logs' selected in the sidebar. The main area displays the log console for 'apdev-american-ws'. It shows three log entries:

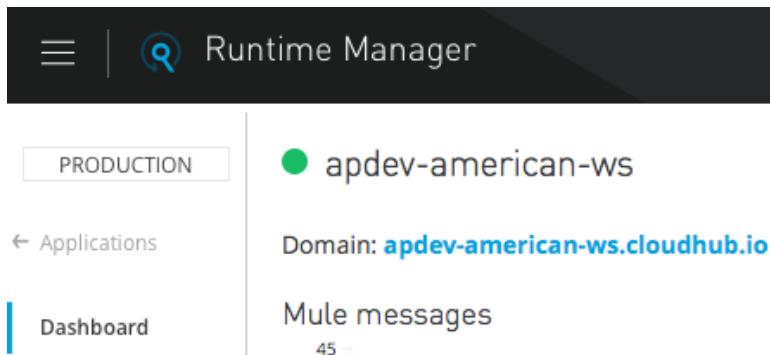
- 16:45:03.629 05/03/2016 Deployment system SYSTEM Your application is started.
- 17:03:34.821 05/03/2016 Worker-0 [apdev-american-ws].american- httpListenerConfig.worker.01 INFO No listener found for request: (GET) /cache/global/img/gs.gif
- 17:03:34.825 05/03/2016 Worker-0 [apdev-american-ws].american- httpListenerConfig.worker.01 INFO Available listeners are: [(*)/api/*]

A separate 'Deployments' panel on the right lists deployment events: '16:40 - Deployment' (checked), '16:33 - Deployment', and '16:28 - Deployment'.

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.

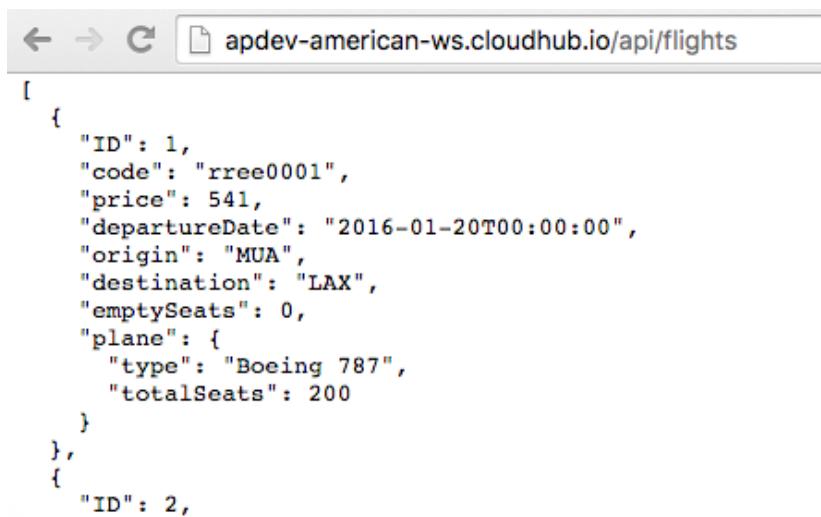
Test the application

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



The screenshot shows the Mule Runtime Manager interface. At the top, there's a dark header with a menu icon, a search icon, and the text "Runtime Manager". Below the header, there's a navigation bar with tabs: "PRODUCTION" (selected), "Applications" (with a back arrow), and "Dashboard" (selected). To the right of the navigation bar, there's a green circular status indicator followed by the text "apdev-american-ws". Below this, it says "Domain: **apdev-american-ws.cloudbhub.io**". Underneath, there's a section titled "Mule messages" with a count of "45".

16. Click the link; a request will be made to that URL and you should get a resource not found message.
17. Modify the path to `http://apdev-american-ws-{lastname}.cloudbhub.io/api/flights`; you should see the flights data.



The screenshot shows a browser window with the address bar containing "`apdev-american-ws.cloudbhub.io/api/flights`". The page content displays a JSON array of flight 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": "rree0002",  
      "price": 650,  
      "departureDate": "2016-01-21T00:00:00",  
      "origin": "MUA",  
      "destination": "SFO",  
      "emptySeats": 0,  
      "plane": {  
        "type": "Boeing 777",  
        "totalSeats": 250  
      }  
    }  
]
```

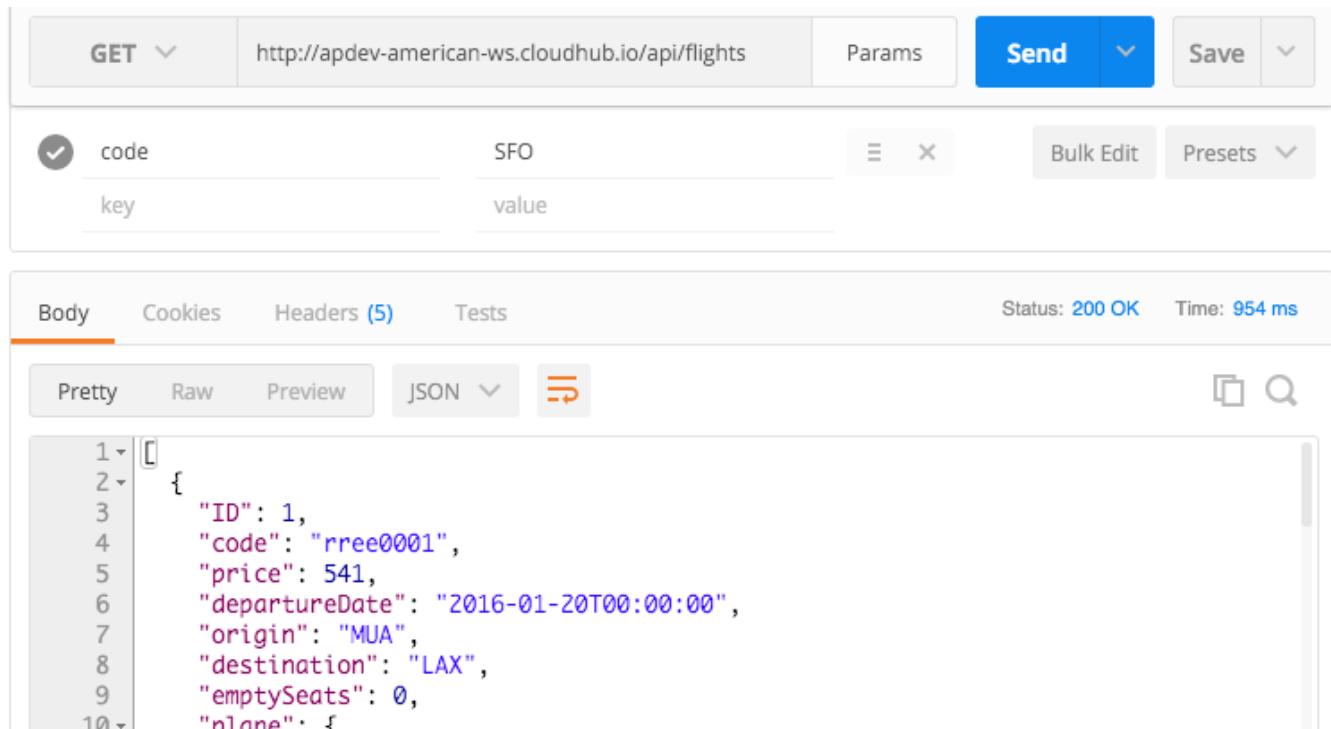
18. Copy the URL.
19. Return to Postman and paste the value in the URL.
20. Click Send; you should get the flights data.

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

21. Add a query parameter called code and set it equal to SFO.

22. 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.



The screenshot shows the MuleSoft Anypoint Platform API Tester. At the top, there's a header bar with 'GET', the URL 'http://apdev-american-ws.cloudhub.io/api/flights', 'Params', a 'Send' button, and a 'Save' button. Below the header, there's a parameter editor with a checked row for 'code' set to 'SFO'. Underneath, there are 'key' and 'value' columns. The main area is titled 'Body' and contains tabs for 'Pretty', 'Raw', 'Preview', and 'JSON'. The 'JSON' tab is selected, showing a JSON array of flight records. The first record is displayed in a pretty-printed format:

```
1 [ {  
2   "ID": 1,  
3   "code": "rree0001",  
4   "price": 541,  
5   "departureDate": "2016-01-20T00:00:00",  
6   "origin": "MUA",  
7   "destination": "LAX",  
8   "emptySeats": 0,  
9   "plane": {}  
10 } ]
```

Update the API implementation in the cloud

23. Return to the apdev-american-ws application in the Runtime Manager in Anypoint Platform.
24. In the left-side navigation, click Settings.
25. Click the Choose file button.
26. Browse to the resources folder in the course student files.
27. Select apdev-american-ws.zip and click Open.

Note: This updated version of the application adds functionality to return results for a particular destination. It also adds flows for deleting and updating a flight, though they don't actually modify the database.

28. Click the Apply Changes button.

The screenshot shows the Mule Runtime Manager interface. On the left, a sidebar lists navigation options: PRODUCTION, Applications (with a back arrow), Dashboard, Insight, Logs, Application Data, Queues, Schedules, and Settings (which is currently selected). The main panel displays the application 'apdev-american-ws'. It includes fields for the application file ('apdev-american-ws.zip'), a 'Choose file' button, and a 'Get from sandbox' button. Below this, the 'App url' is shown as 'apdev-american-ws.cloudhub.io'. A horizontal navigation bar at the top of the main panel has tabs for 'Runtime', 'Properties', 'Insight', 'Logging', and 'Static IPs', with 'Runtime' being the active tab. Under 'Runtime', the 'Runtime version' is set to '3.8.0', 'Worker size' is '0.1 vCores', and 'Workers' is '1'. There are also several checkboxes for settings: 'Automatically restart application when not responding' (checked), 'Persistent queues' (unchecked), 'Encrypt persistent queues' (unchecked), 'Secure data gateway' (unchecked), and 'Enhanced log management' (checked). At the bottom right of the main panel is a large blue 'Apply Changes' button.

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

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

Test the application

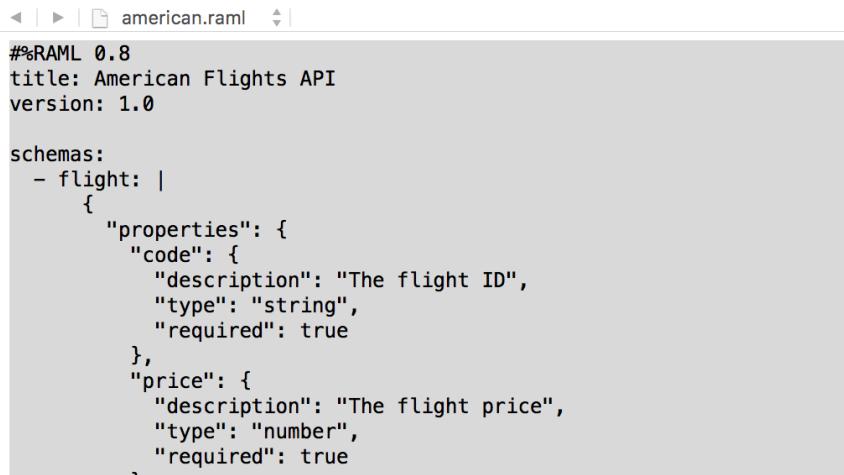
30. In Postman, make the same request with a code of SFO; you should now get only flights to SFO.

The screenshot shows a Postman request configuration and its response. The request method is 'GET' and the URL is 'http://apdev-american-ws-jms.cloudhub.io/api/flights?code=SFO'. The response status is '200 OK' and the time taken is '993 ms'. The response body is displayed in a JSON editor with line numbers. The JSON data is as follows:

```
1 [  
2 {  
3   "ID": 5,  
4   "code": "rree1093",  
5   "price": 142,  
6   "departureDate": "2016-02-11T00:00:00",  
7   "origin": "MUA",  
8   "destination": "SFO",  
9   "emntrvSnts": 1
```

Update the RAML file in the cloud

31. In your computer's file browser, locate and open the american.raml file in the student files.
32. Copy all the code in the file.



```
#%RAML 0.8
title: American Flights API
version: 1.0

schemas:
  - flight: |
    {
      "properties": {
        "code": {
          "description": "The flight ID",
          "type": "string",
          "required": true
        },
        "price": {
          "description": "The flight price",
          "type": "number",
          "required": true
        }
      }
    }
```

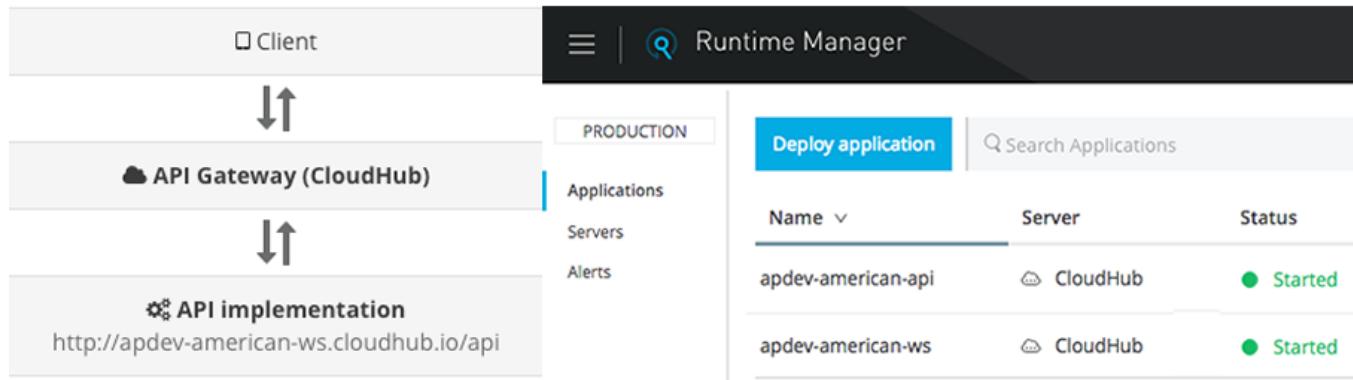
33. Return to Anypoint Platform in a web browser.
34. From the main menu, select API Manager.
35. Click the American Flights API version 1.0 link.
36. In the API Definition section, click the Edit in API designer link.
37. Turn off the mocking service.
38. Select all the text and delete it.
39. Paste the new text from the american.raml file.
40. Save the file.
41. Leave this page open.

Note: You updated the RAML file to match the updated, deployed application. At this time, you are not going to update the implementation source code in Anypoint Studio because it requires the addition of functionality that is taught later in the Developer courses.

Walkthrough 4-3: Create and deploy an API proxy

In this walkthrough, you create and deploy an API proxy for your API implementation in the cloud. You will:

- Set the API implementation URI in the RAML file.
- Use Anypoint Platform to automatically create an API proxy application.
- Deploy the API proxy application.



Set the implementation URI in the RAML file

1. Return to the american.raml definition page for American Flights API – 1.0 in Anypoint Platform.
2. Add a baseUri and set it equal to <http://apdev-american-ws-{lastname#}.cloudhub.io/api>.

```
1  #%RAML 0.8
2  baseUri: http://apdev-american-ws.cloudhub.io/api
3  title: American Flights API
4  version: 1.0
```

3. Save the file.
4. Click the American Flights API - 1.0 link.

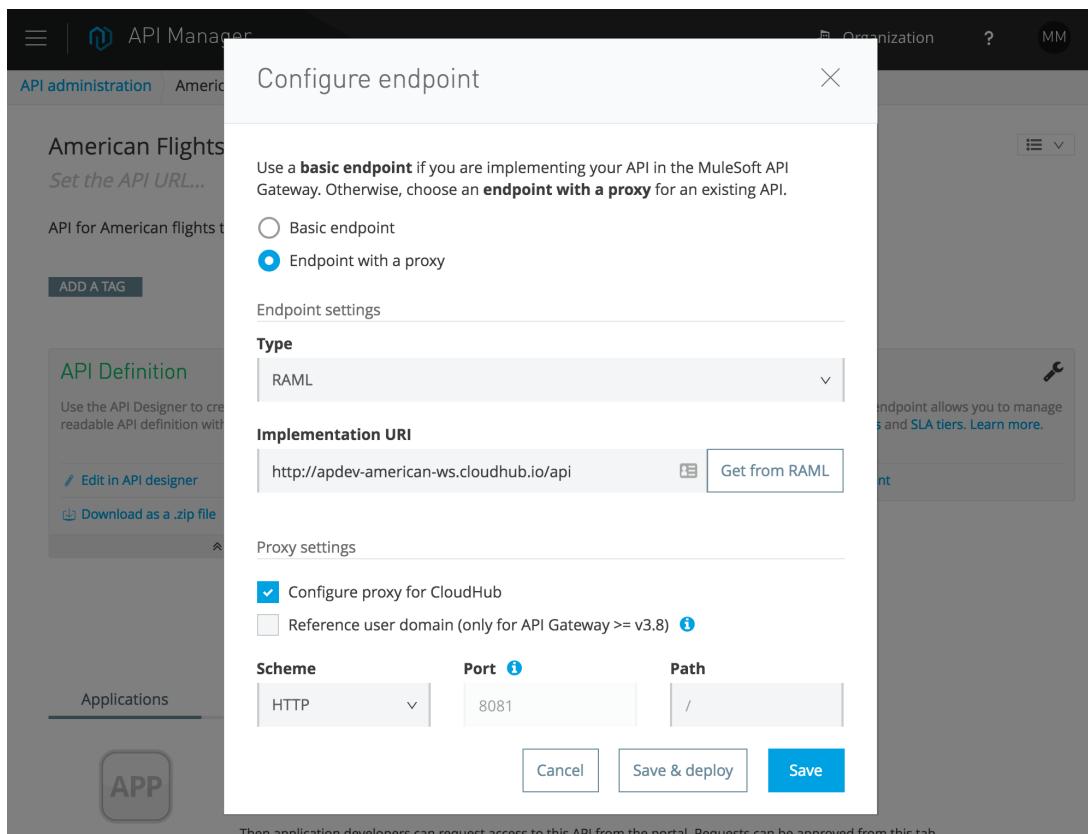
Create an API proxy

5. In the API Status section, click the Configure endpoint link.

The screenshot shows three cards in the 'API Status' section. The first card, 'API Definition', has a green checkmark and says 'Use the API Designer to create a concise, human-readable API definition with RAML.' It has a 'Edit in API designer' button. The second card, 'API Portal', also has a green checkmark and says 'Publishing an API portal allows you to expose documentation and other content that can help developers understand how to use your API. For more information, see Engaging users of your API.' It has a dropdown menu currently showing 'American Flights API'. The third card, 'API Status', has a wrench icon and says 'Configuring the API endpoint allows you to manage your API with policies and SLA tiers. Learn more.' It has a 'Configure endpoint' button.

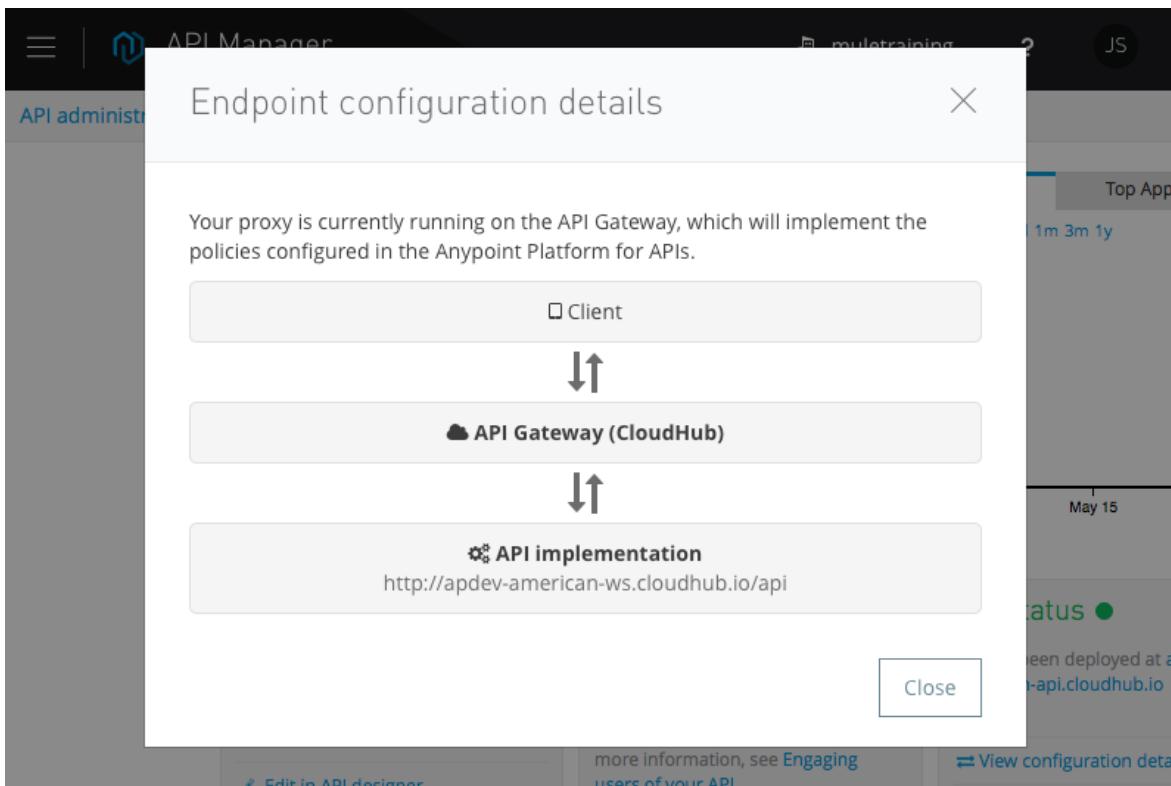
6. In the Configure endpoint dialog box, confirm the following values are set:

- Endpoint with a proxy
 - Type: RAML
 - Implementation URI: `http://apdev-american-ws-{lastname#}.cloudhub.io/api`
7. Select the Configure proxy for CloudHub checkbox.
8. Confirm the following values are set:
- Scheme: RAML
 - Port: 8081
 - Path: /



9. Click Save.

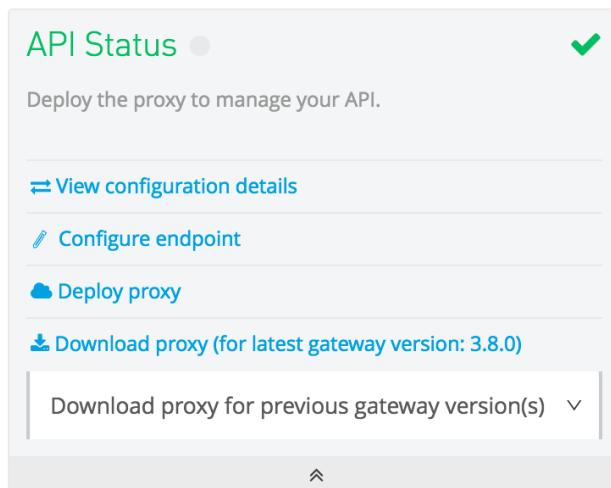
10. In the API Status section, click the View configuration details link.



11. Click Close.

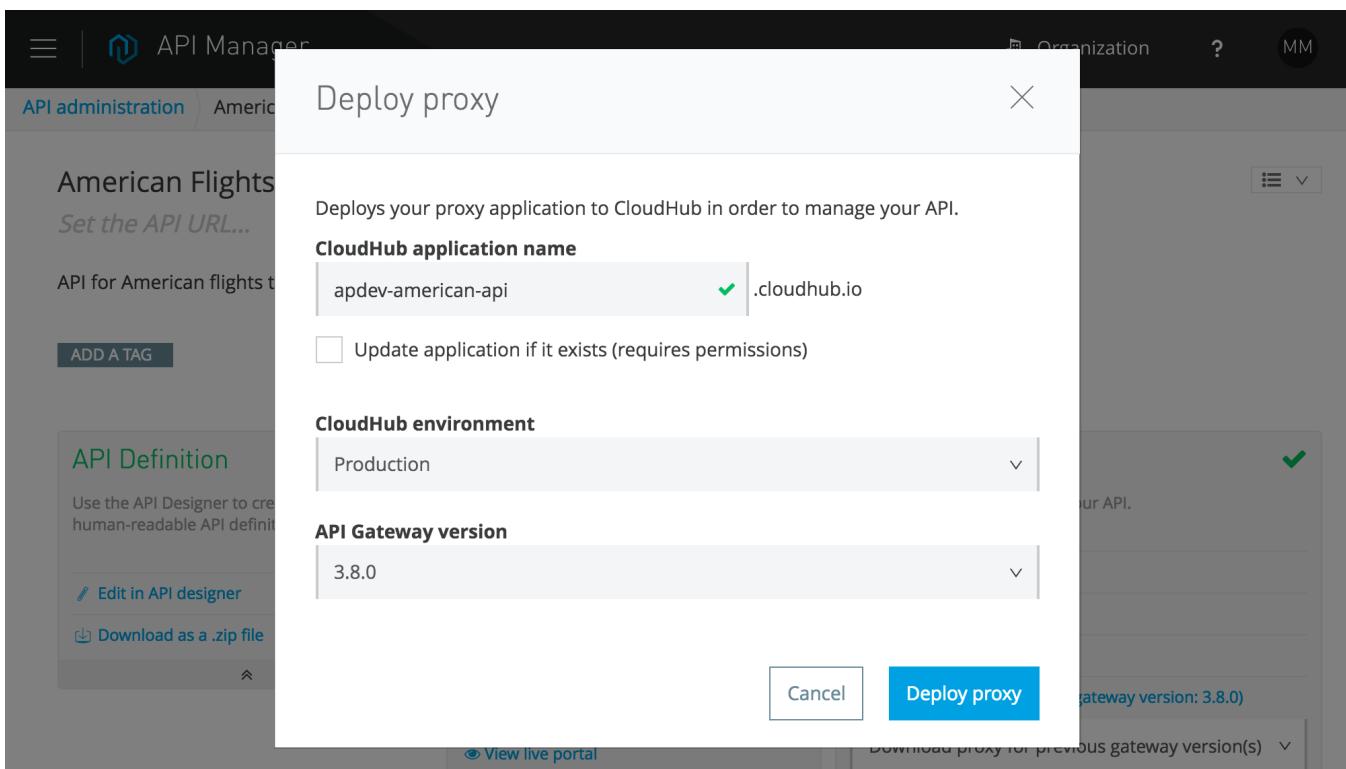
Deploy the proxy

12. In the API Status section, click the Deploy proxy link.

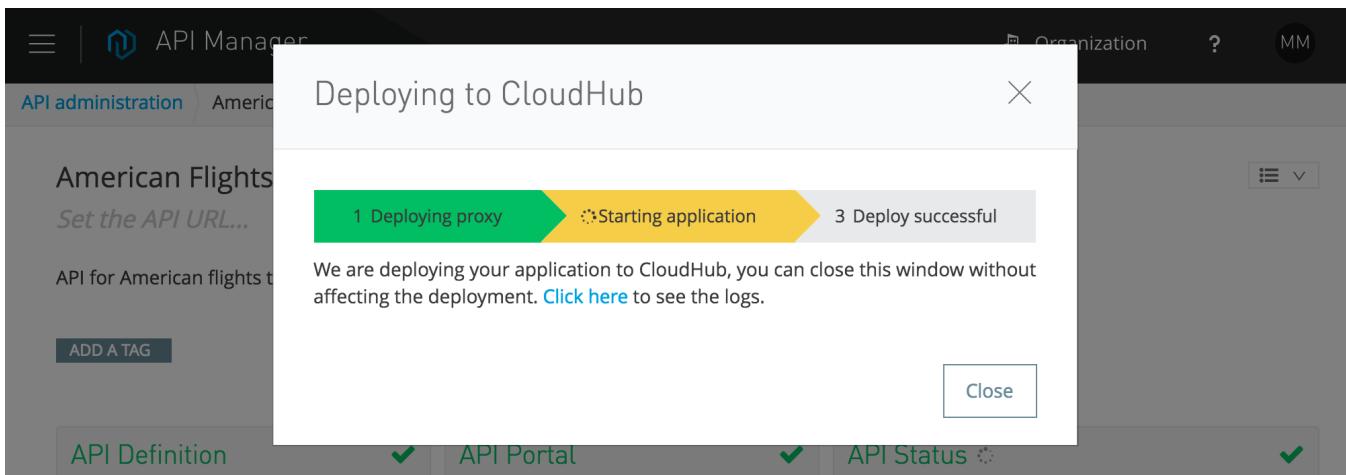


13. In the Deploy proxy dialog box, change the CloudHub application name to apdev-american-api-{lastname#}.cloudhub.io.

14. Click Deploy proxy.



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



16. Watch the logs.

17. 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.

Locate the API proxy application

18. In the main menu in Anypoint Platform, select Runtime Manager.
19. Locate the new apdev-american-api-{lastname#} application.

The screenshot shows the Runtime Manager interface. On the left, there's a sidebar with tabs for PRODUCTION, Applications, Servers, and Alerts. The PRODUCTION tab is selected. In the center, there's a table titled "Deploy application" with columns for Name, Server, Status, and File. Two applications are listed:

Name	Server	Status	File
apdev-american-api	CloudHub	Started	apdev-american-api-api-gateway.zip
apdev-american-ws	CloudHub	Started	apdev-american-ws.zip

Look at the API request data

20. From the main menu, select API Manager.
21. Click the American Flights API version 1.0 link.
22. Look at the Requests chart in the upper-right corner.

Call the API

23. Return to Postman.
24. Change the request URL to apdev-american-api-{lastname#}.cloudhub.io/flights; you should successfully get flights.

Note: Be sure to remove the /api from the URL.

The screenshot shows the Postman application interface. At the top, there's a header with "GET" and a URL field containing "http://apdev-american-api-jms.cloudhub.io/flights". Below the header, there are tabs for Body, Cookies, Headers (5), and Tests. The Headers tab is selected, showing "Status: 200 OK" and "Time: 1043 ms". Under the Body tab, there's a JSON response:

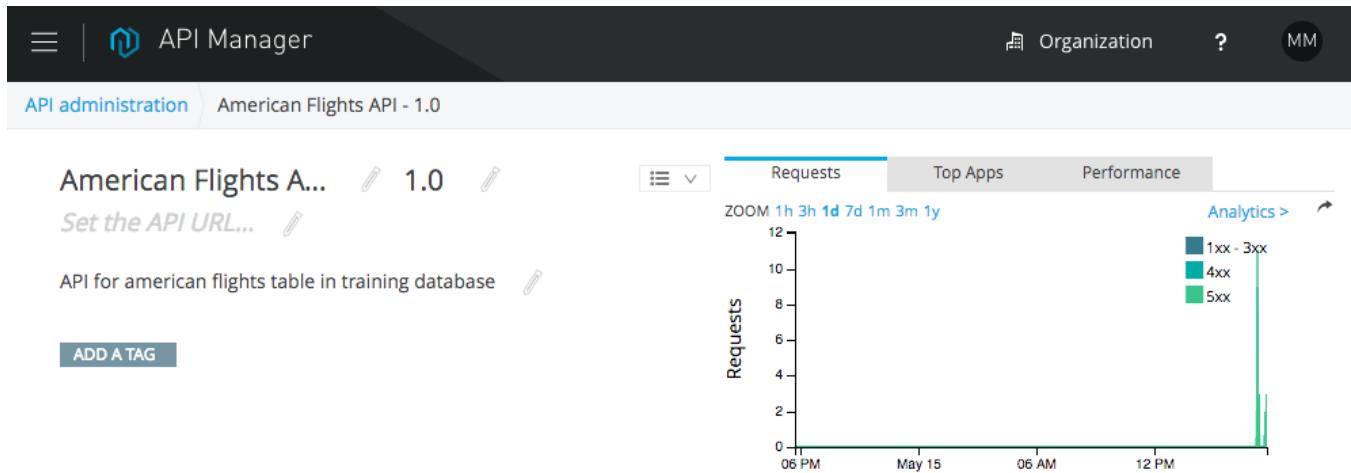
```
1. [ ] {
2.   "ID": 1,
3.   "code": "rree0001",
4.   "price": 541,
5.   "departureDate": "2016-01-20T00:00:00",
6.   "origin": "MUA",
7.   "destination": "LAX",
8.   "amount": 541
}
```

25. Make several more calls.

Look at the API request data

26. Return to the API details page for American Flights API v1.0 in Anypoint Platform.

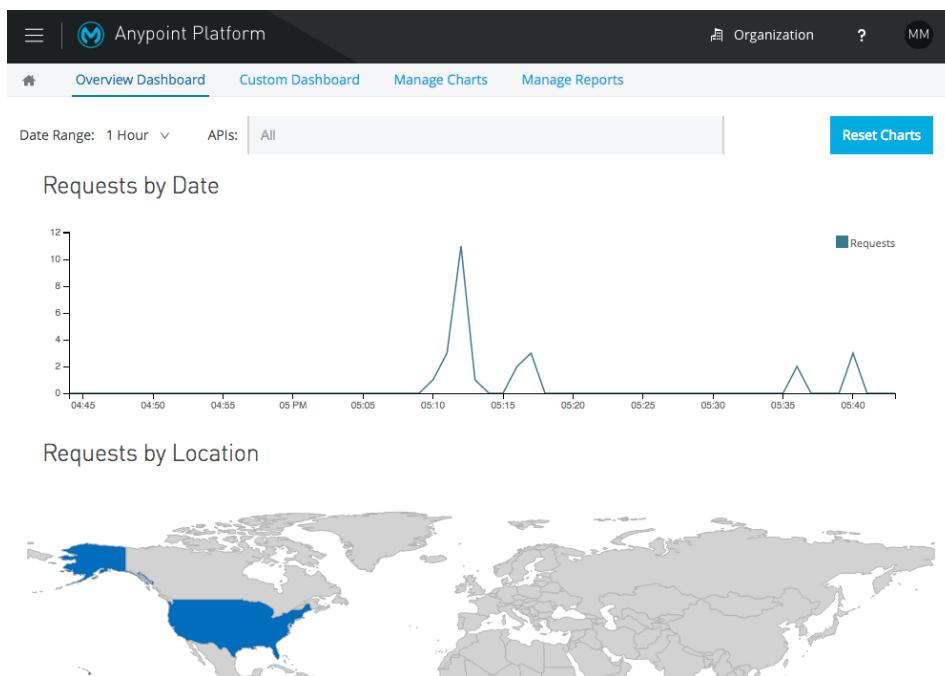
27. Look at the Request graph in the upper-right corner; you should see some calls.



28. Click the Analytics> link.

29. Take a quick look at the Overview Dashboard.

30. Change the Date Range to 1 hour.



31. Close the web browser tab or window.

Walkthrough 4-4: Restrict API access

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 a rate limiting SLA policy.
- Request access for an application to an SLA tier from the API Developer portal.
- Approve access for an application to an SLA tier.
- Call the governed API with client credentials.

The screenshot shows the 'APDev fictitious application' configuration page. At the top, there's a header with a user dropdown ('username05'). Below it, tabs for 'My applications' and 'APDev fictitious application' are selected. A 'Delete this app' button is on the right.

The main area has a 'Send' button and a 'Save' button. To the left is a table with columns for 'Method' (GET), 'URL' (apdev-american-api.cloudhub.io/flights?client_id=d1374b15c6), and 'Params'. It contains two rows: 'client_id' with value 'd1374b15c6864c3682ddbed2a247a' and 'client_secret' with value 'a4cb1787a04c41c3866FD31B95435'. There are 'Bulk Edit' and 'Delete' buttons next to the table.

To the right, there's a sidebar with 'Client ID' (3e1f2b48751049de917b66dc95489cd3) and 'Client secret' (77b1414d1f464c8d989C5B6C0DF701AE). Buttons for 'Reset client secret' and 'Request tier change' are also here.

At the bottom, a table lists API details: API name 'American Flights API', API version '1.0', Current SLA tier 'Silver', Requested SLA tier 'N/A', and Status 'Approved'.

Create a rate limiting policy

1. Return to the API details page for American Flights API v1.0 in Anypoint Platform.
2. At the bottom of the page, locate and click the Policies tab.

The screenshot shows the 'Policies' tab selected in the navigation bar. Below it is a search bar with placeholder text 'Search'.

A message at the bottom states: 'There are no registered applications for this API Version.'

- Click the Apply button next to the rate limiting policy.

Applications Policies SLA tiers Permissions

Applied policies

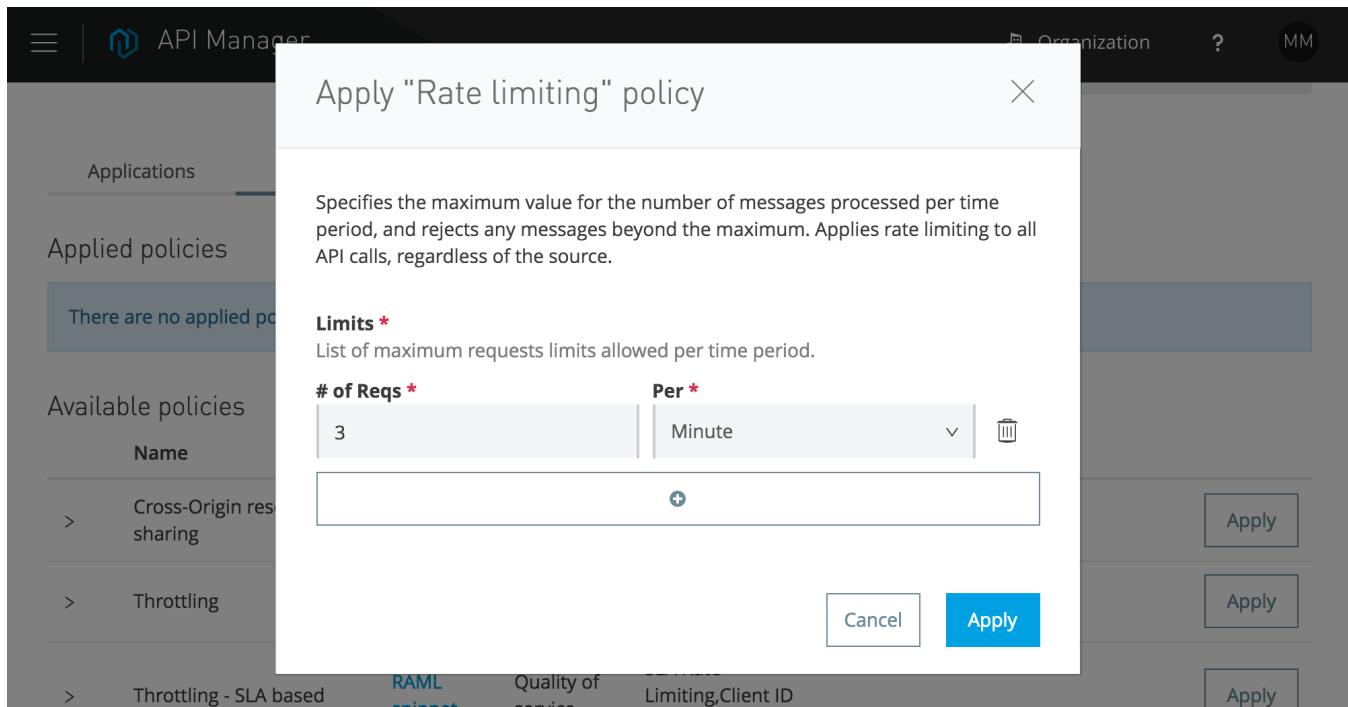
There are no applied policies.

Available policies

Name	Category	Fulfils	Requires
> Cross-Origin resource sharing	Compliance	CORS enabled	<button>Apply</button>
> Throttling	Quality of service	Baseline Rate Limiting	<button>Apply</button>
> Throttling - SLA based	RAML snippet	Quality of service SLA Rate Limiting, Client ID required	<button>Apply</button>
> Rate limiting	Quality of service	Baseline Rate Limiting	<button>Apply</button>

- In the Apply "Rate limiting" policy dialog box, set the following values and click Apply:

- # of Reqs: 3
- Per: Minute



Test the rate limiting policy

5. In Postman, make another request to <http://apdev-american-api-{lastname#}.cloudbhub.io/flights>; you should get flight results.
6. Click the Send button three more times until you get a response that the allowed number of API calls has been exceeded.

The screenshot shows a Postman request configuration for a GET request to 'apdev-american-api.cloudbhub.io/flights'. The 'Body' tab is selected, showing the response: '1 | API calls exceeded'. The status bar indicates 'Status: 429 Too Many Requests' and 'Time: 104 ms'.

Create SLA tiers

7. Return to the API administration page for American Flights API in Anypoint Platform.
8. At the bottom of the page, click the SLA tiers tab.
9. Click the Add SLA tier button.

The screenshot shows the 'SLA tiers' tab selected in the navigation bar. A blue button labeled 'Add SLA tier' is visible. A search bar is also present.

There are no SLA tiers for this API version.

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

- Name: Free
- Approval: Automatic
- # of Reqs: 1
- Per: Hour

The screenshot shows the 'Update SLA tier' dialog box from the API Manager interface. The dialog has a title bar 'Update SLA tier' with a close button 'X'. It contains several input fields:

- Name ***: A text input field containing 'Free'.
- Description**: A text input field containing 'Description'.
- Approval ***: A dropdown menu showing 'Automatic'.
- Limits**: A section with two input fields:
 - # of Reqs ***: A text input field containing '1'.
 - Per ***: A dropdown menu showing 'Hour'.A checkbox labeled 'visible' is checked, and there is a trash can icon.

At the bottom right of the dialog are 'Cancel' and 'Update' buttons. The background shows a dark grey sidebar with some API names like 'n-american-api-' and 'gateway version:'.

11. Click the Add button.

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

- Name: Silver
- Approval: Manual
- # of Reqs: 1
- Per: Second

Applications		Policies	SLA tiers	Permissions	
Add SLA tier		<input type="text"/> Search X		1 - 2 of 2 < >	
Name	Limits	Applications	Status	Approval	
Free	1	0	Active	Auto	Edit Delete
Silver	1	0	Active	Manual	Edit Delete

Change the policy to rate limiting – SLA based

13. Click the Policies tab.

14. Click the Remove button for the Rate limiting policy.

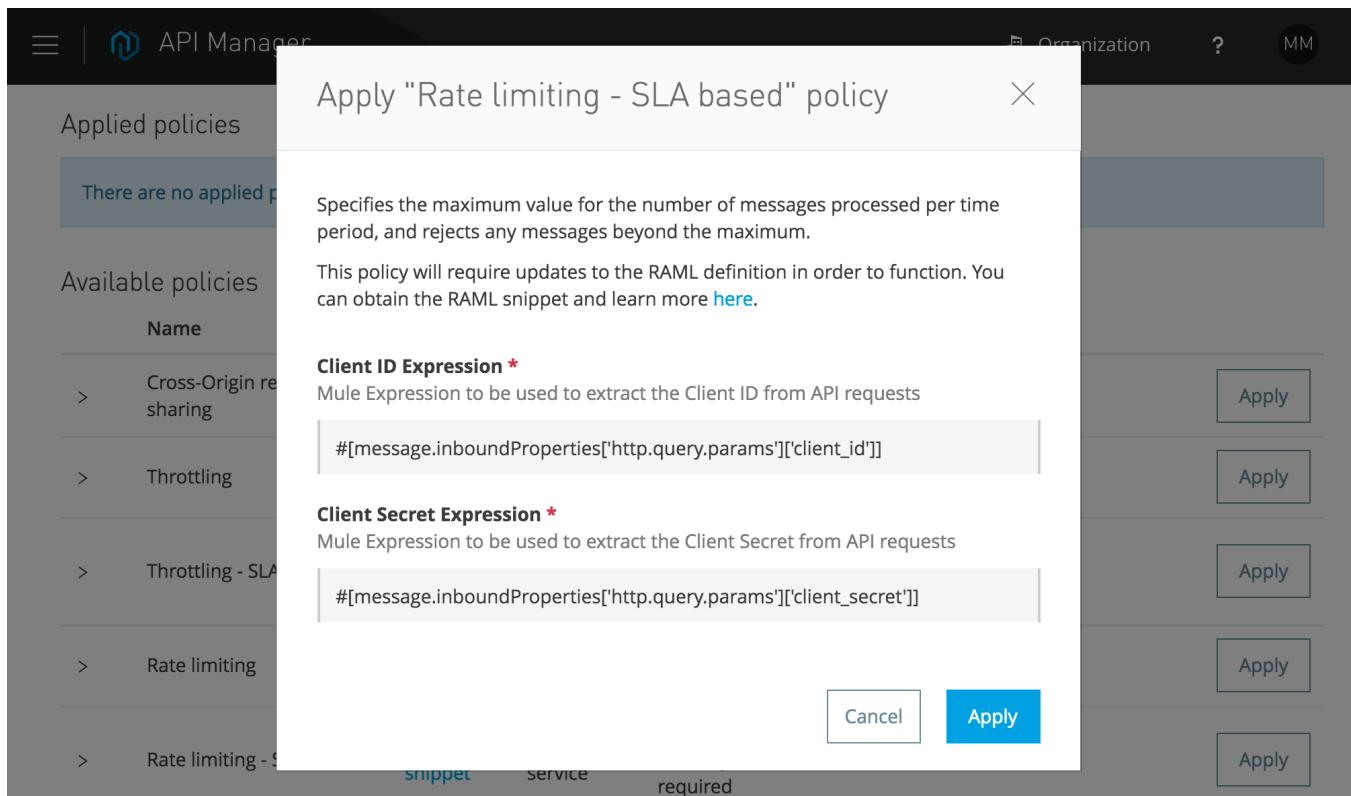
Applied policies					Edit policy order
Name	Category	Fulfils	Requires		
> 1 Rate limiting	Quality of service	Baseline Rate Limiting		Edit	Remove

15. In the Remove policy dialog box, click Remove.

16. Click the Apply button for the Rate limiting – SLA based policy.

>	Rate limiting - SLA based	RAML snippet	Quality of service	SLA Rate Limiting, Client ID required	Apply
---	---------------------------	------------------------------	--------------------	---------------------------------------	-----------------------

17. In the Apply "Rate-limiting – SLA based" policy dialog box, look at the expressions and see that a client ID and secret need to be sent with API requests as query parameters.



18. Click Apply.

Test the rate limiting – SLA based policy

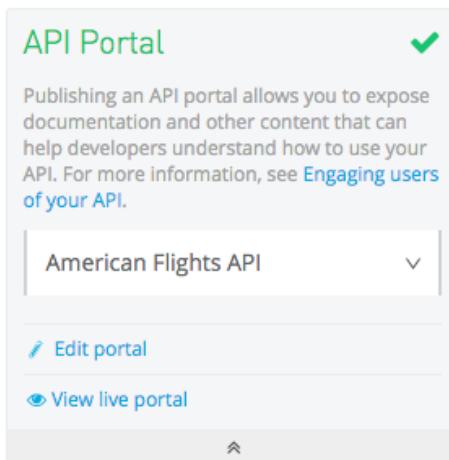
19. In Postman, make another request to your API; you should get a 401 response with a message that a client_id was not provided.

The screenshot shows a Postman request configuration. The method is set to 'GET', the URL is 'apdev-american-api.cloudhub.io/flights', and the status is '401 OK' with a duration of '275 ms'. The 'Body' tab is selected, showing the response content: 'i 1 | Unable to retrieve client_id from message'.

Request access to the API

20. Return to the API details page for American Flights API v1.0 in Anypoint Platform.

21. In the API Portal section, click the View live portal link.



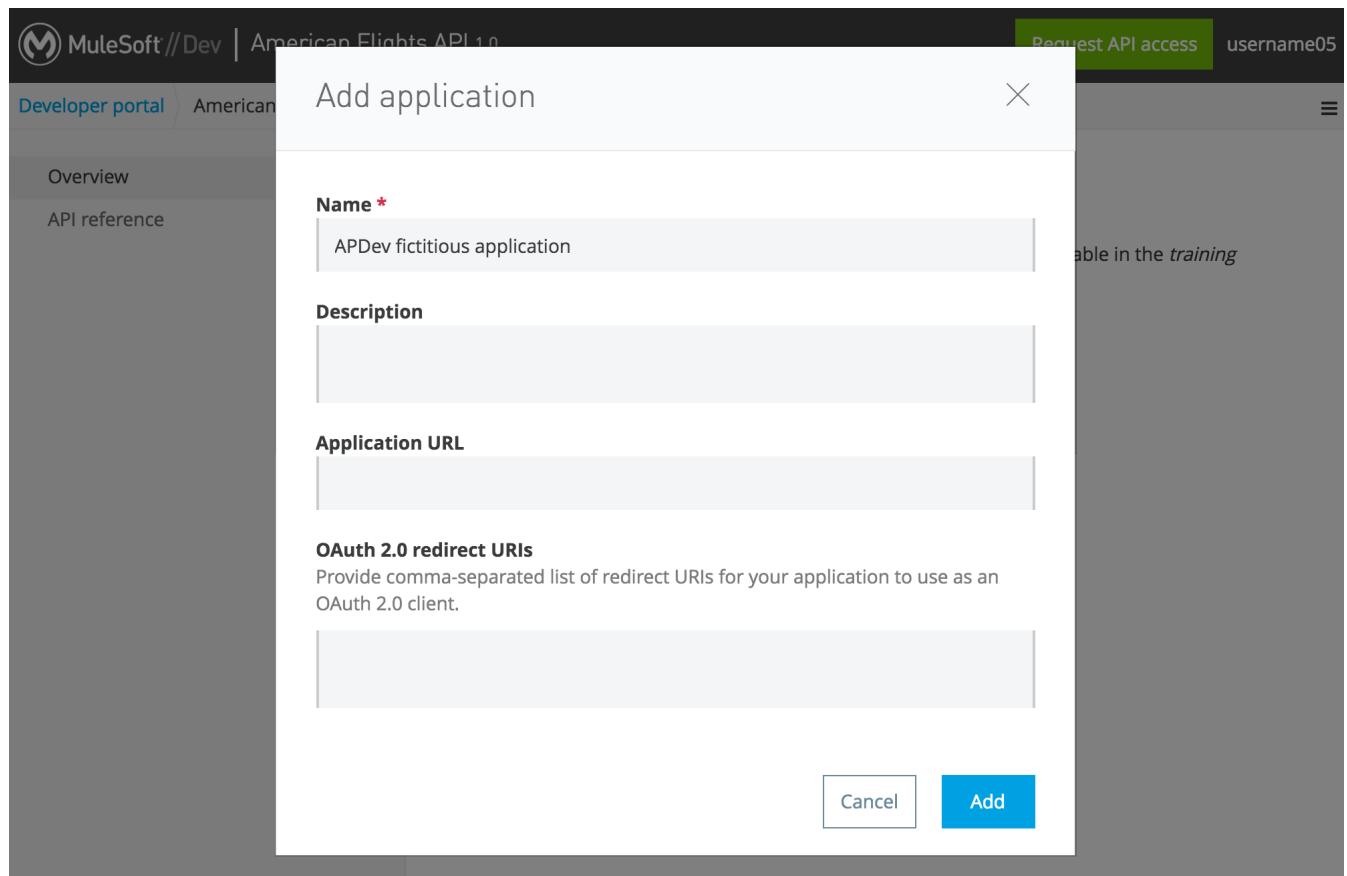
22. In the portal, click the Request API access button.

A screenshot of the American Flights API 1.0 Overview page. The top navigation bar includes the MuleSoft logo, the API name, and user information ("Request API access" and "username05"). Below the navigation, the page title is "American Flights API - 1.0" and the sub-page title is "Overview". On the left, there's a sidebar with "Overview" and "API reference" options. The main content area is titled "Overview" and contains a brief description of the API: "The American flights API is a system API for operations on the american table in the training database." A "Request API access" button is visible in the top right corner of the main content area.

23. In the Request API access to American Flights API - 1.0 dialog box, click New application.

A screenshot of the "Request API access to American Flights API - 1.0" dialog box. The title bar says "Request API access to American Flights API - 1.0". The main content asks "Which application would like access?" and shows a dropdown menu with "--- Select an application ---" and a "New application" button. At the bottom, there are "Cancel" and "Request API access" buttons.

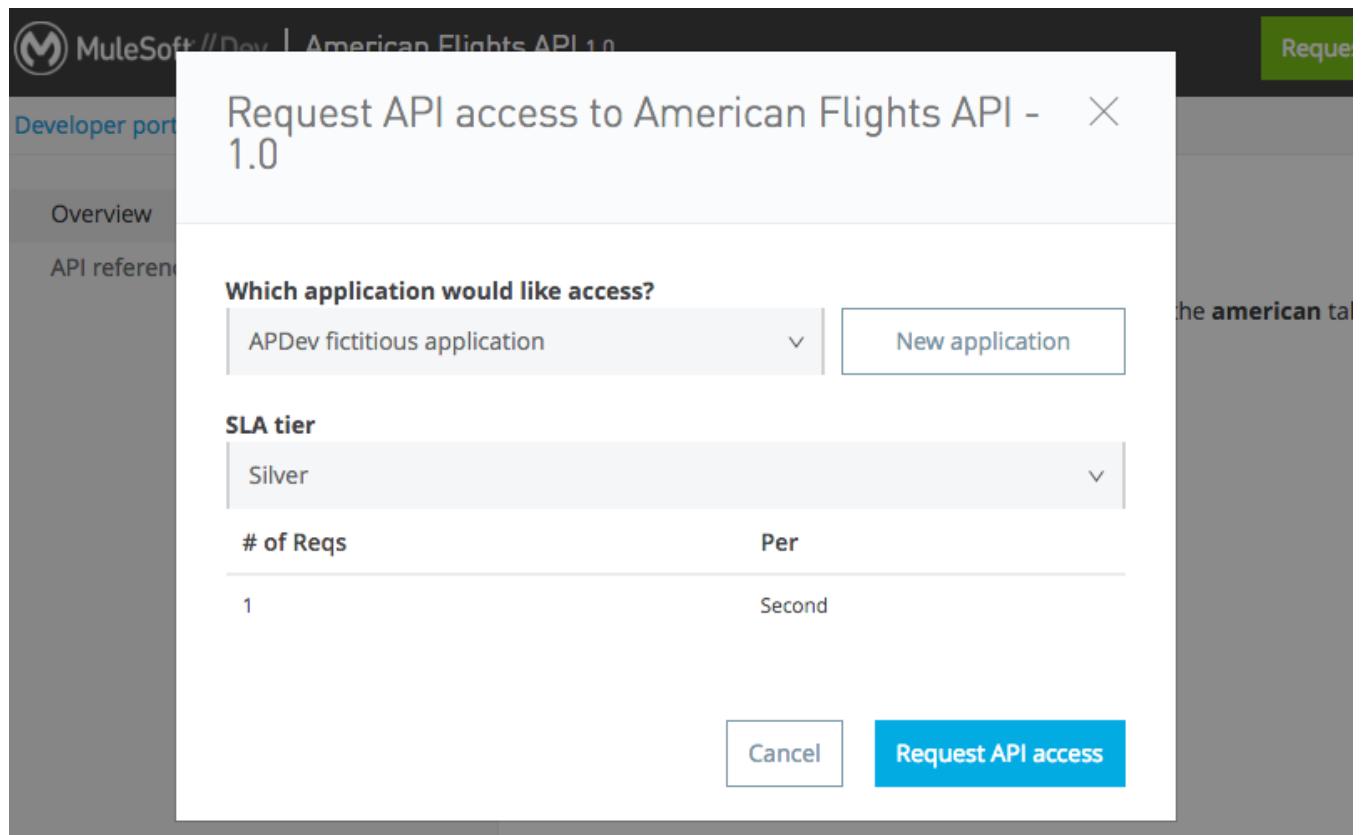
24. In the Add application dialog box, set the name to APDev fictitious application (or some other name).



25. Click Add.

26. In the Request API access to American Flights API – 1.0 dialog box, set the SLA tier to Silver.

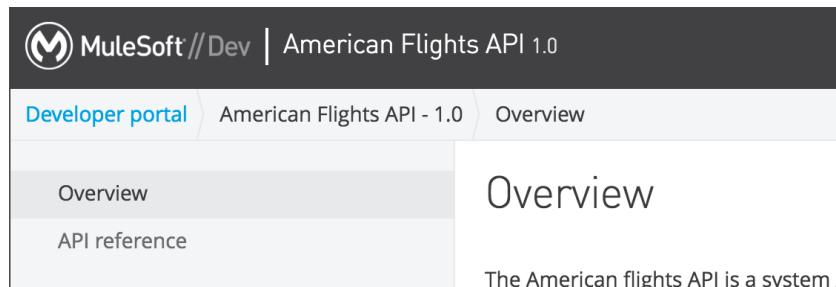
27. Click Request API access.



28. In the Request API access dialog box, click OK.

Retrieve client ID and secret

29. In the main menu bar, click the Developer portal link.



30. In the Developer portal, click My applications.

31. Click the APDev fictitious application link.

Name	Description
APDev fictitious application	

32. Locate the application client ID and client secret.

API name	API version	Current SLA tier	Requested SLA tier	Status
American Flights API	1.0	Silver ⓘ	N/A	Approved

33. Copy the client ID and paste it in the course snippets.txt file under your client_id.

34. Copy the client secret and paste it in the course snippets.txt file under your client_secret.

Approve the request

35. Click the menu button and select Manage your APIs; this should return you to the API administration page in Anypoint Platform.

36. Click the 1.0 link for American Flights API.

37. Select the Applications tab; you should see the request from APDev fictitious application.

Applications	Policies	SLA tiers	Permissions	
APDev fictitious application	N/A	Silver	Pending	<button>Approve</button> <button>Reject</button> <button>Delete</button>

38. Click Approve.

Test the API SLA

39. In Postman, make sure the URL is set to <http://apdev-american-api-{lastname#}.cloudbhub.io/flights>.

40. Click the Params button next to the URL.

41. Set the following key and value values:

- key: client_id
- value: *Get the value for your application from your course snippets.txt file*

42. Add a second key/value pair with the following values:

- key: client_secret
- value: *Get the value for your application from your course snippets.txt file*

The screenshot shows a Postman interface with a GET request to `apdev-american-api.cloudbhub.io/flights?client_id=d1374b15c6`. The Params section contains two entries: `client_id` with value `d1374b15c6864c3682ddbed2a247a` and `client_secret` with value `a4cb1787a04c41c3866FD31B95435`. The Send button is visible at the top right.

43. Click Send; you should get the flight data returned again.

The screenshot shows the Postman interface with a successful API call. The URL is `apdev-american-api.cloudhub.io/flights?client_id=d1374b15c6`. The request method is GET. There are two parameters: `client_id` with value `d1374b15c6864c3682ddbed2a247a` and `client_secret` with value `a4cb1787a04c41c3866FD31B95435`. The response status is 200 OK with a time of 1149 ms. The response body is a JSON array:

```
[{"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": "rree0002", "price": 541, "departureDate": "2016-01-20T00:00:00", "origin": "MUA", "destination": "LAX", "emptySeats": 0, "plane": {"type": "Boeing 787", "totalSeats": 200}}]
```