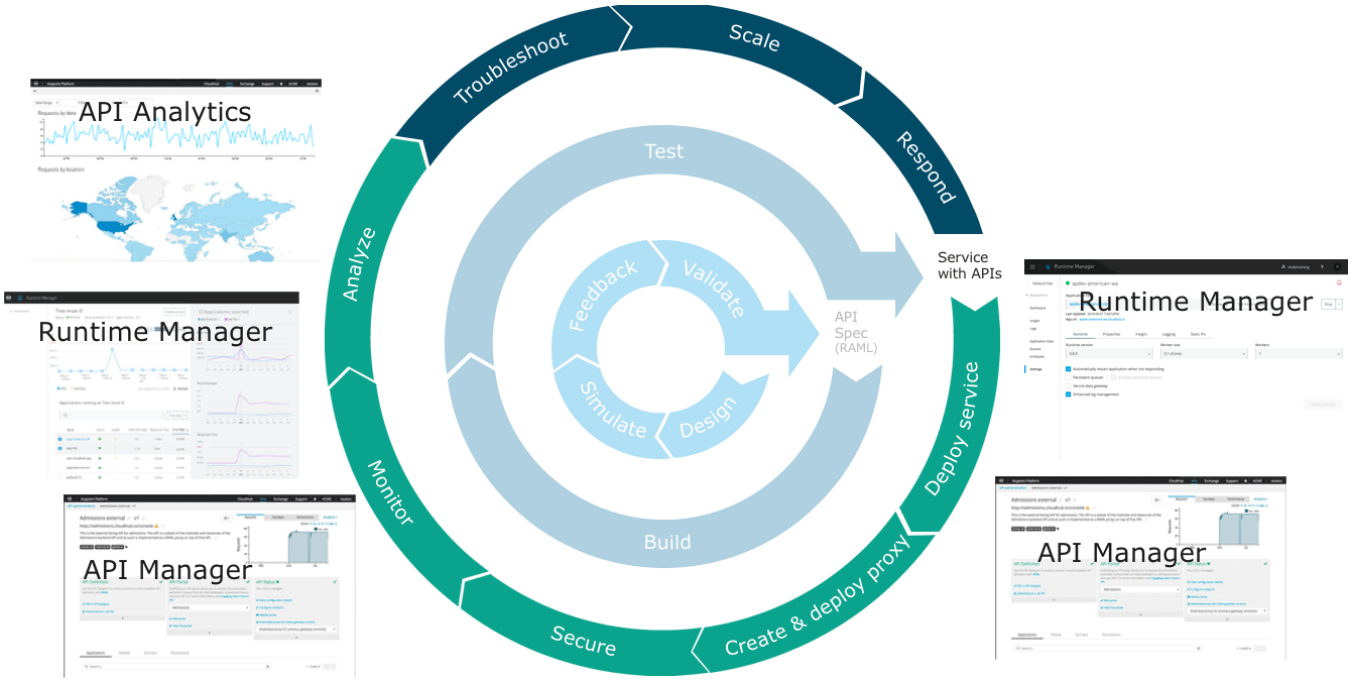
Module 4: Deploying and Managing APIs



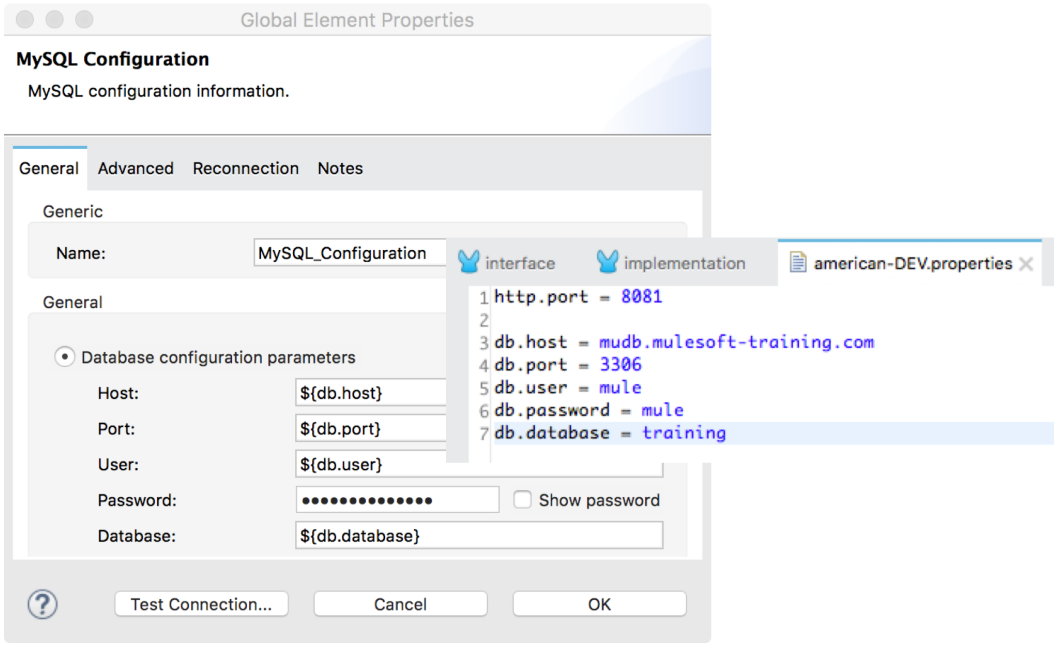
Objectives:

- 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

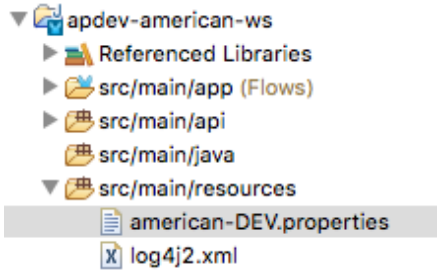
In this walkthrough, you introduce properties into your API implementation. You will:

- 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.



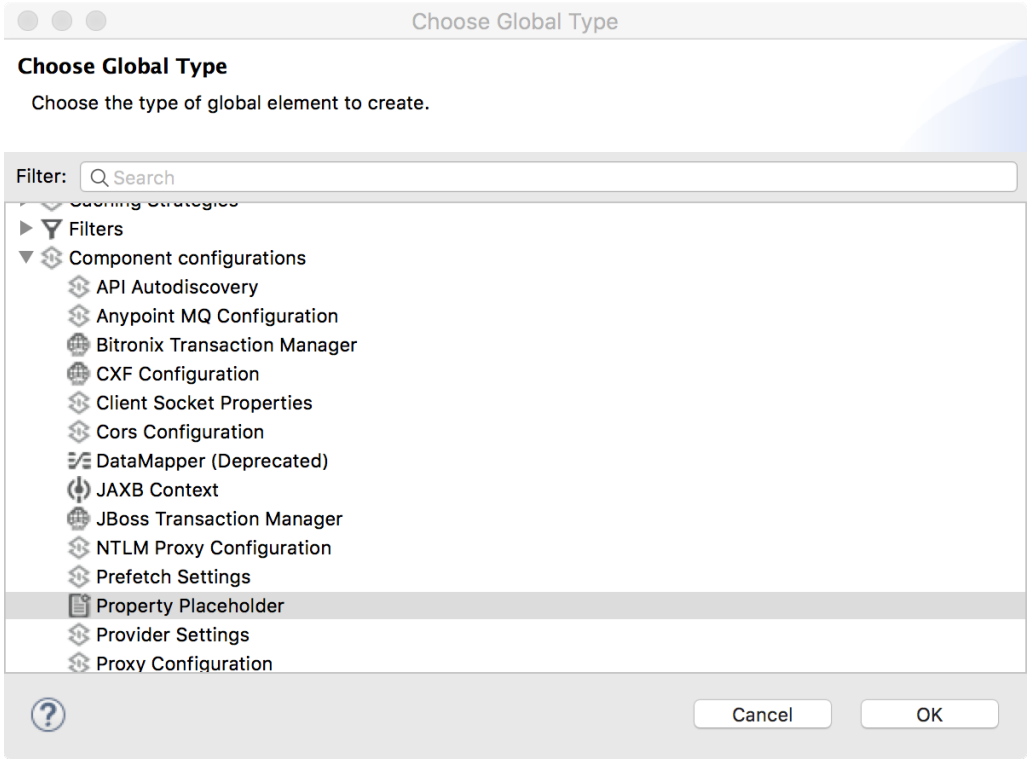
Create a properties file

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.

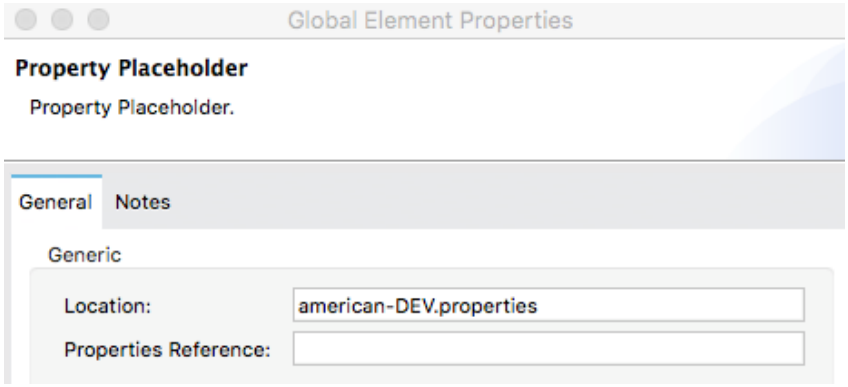


Create a Properties Placeholder global element

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.

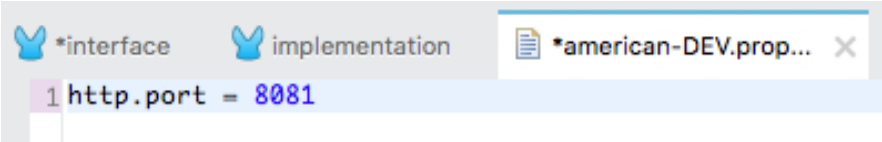


7. In the Global Element Properties dialog box, set the location to american-DEV.properties and click OK.

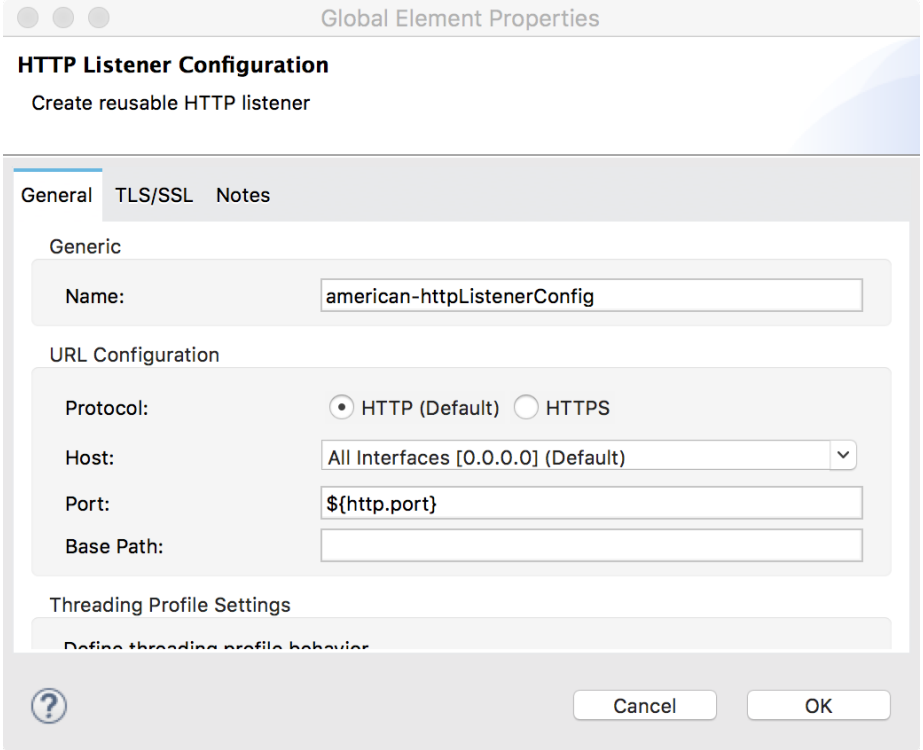


Parameterize the HTTP Listener port

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



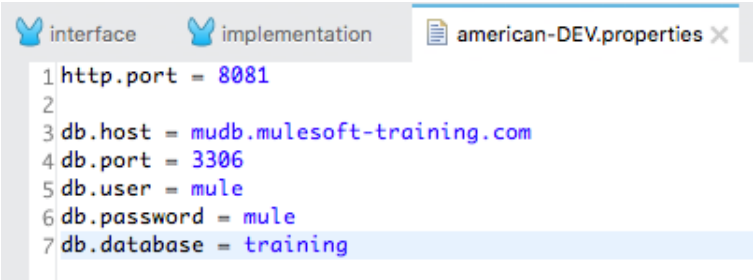
10. Return to the Global Elements view in interface.xml.
11. Double-click the HTTP Listener Configuration global element.
12. Change the port from 8081 to the application property, ${http.port}.
13. Click OK.



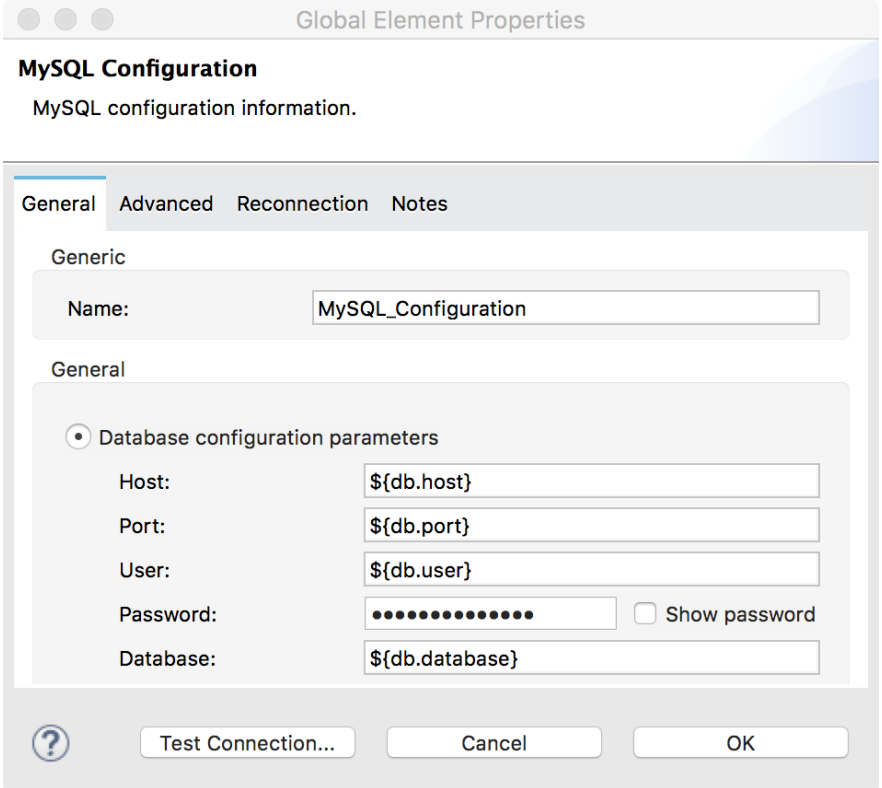
Parameterize the database credentials

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.



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.

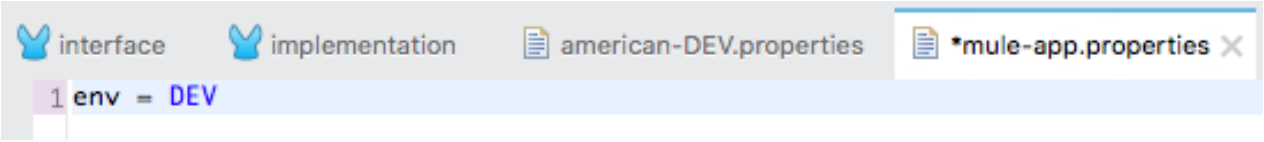
21. Click OK.
22. In the Global Element Properties dialog box, click OK.
23. Switch to the Message Flow view.

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

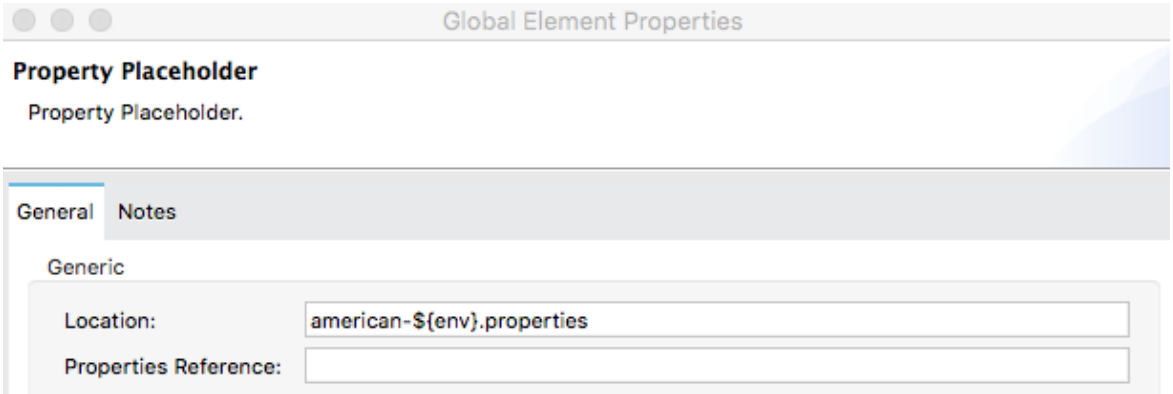
26. Return to Anypoint Studio and stop the project.
27. Open mule-app.properties located in the src/main/app folder.
28. Define a property called env and set it to DEV.



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.



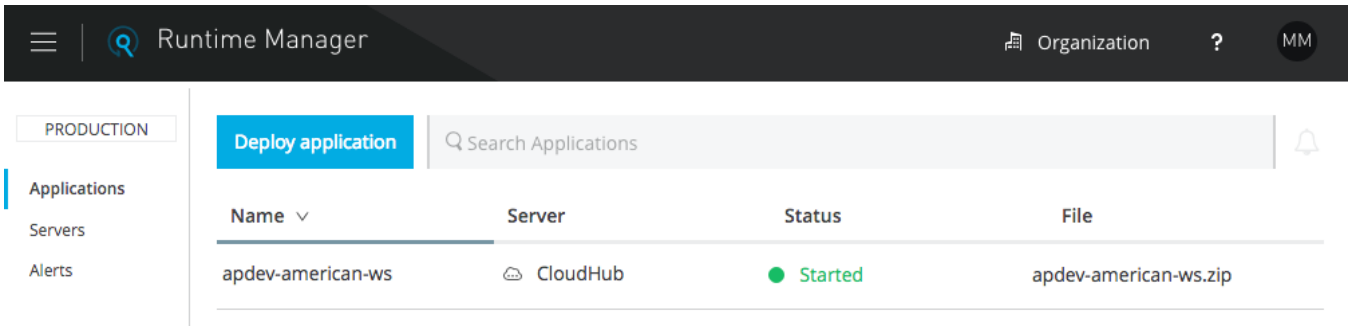
Test the application

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

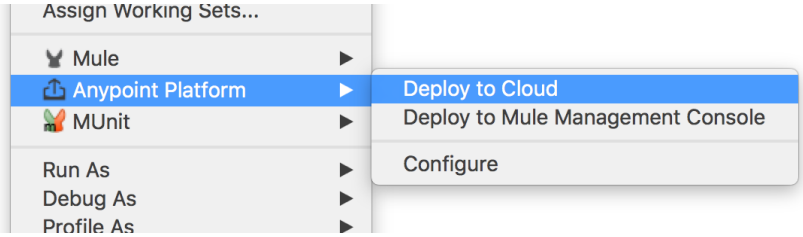
In this walkthrough, you deploy and run your application on Anypoint Platform in the cloud. You will:

- 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.



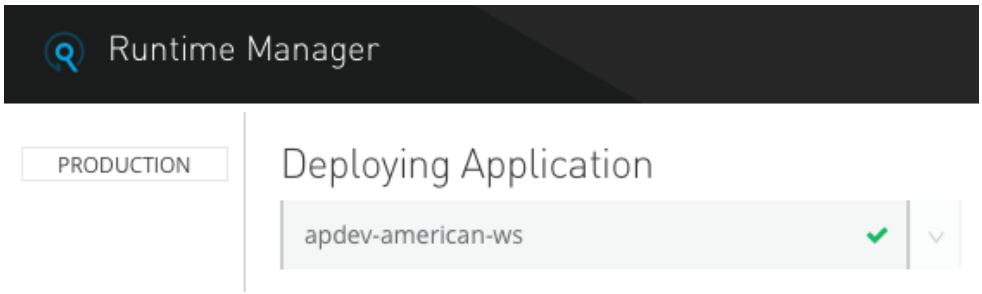
Deploy the application to the cloud

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.

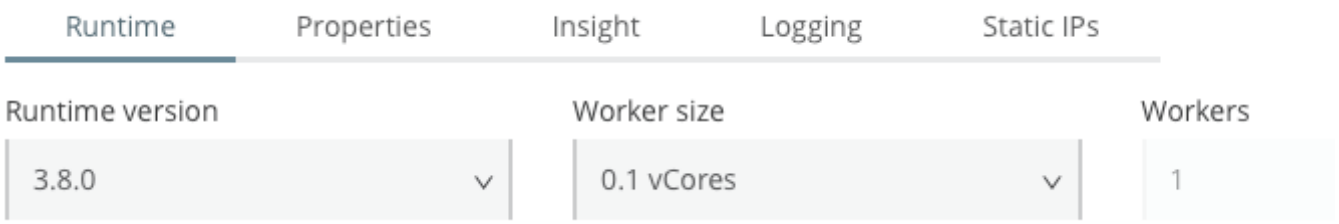
   Note: This name will be part of the URL used to access the application on CloudHub. It must be unique across all applications on CloudHub. The availability of the domain is instantly checked and you will get a green check mark if it is available.



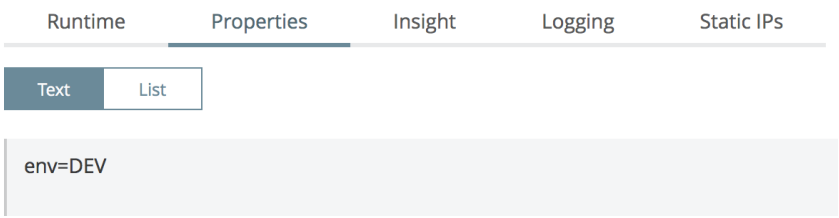
4. Make sure the runtime version is set to the version your project is using.

   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.

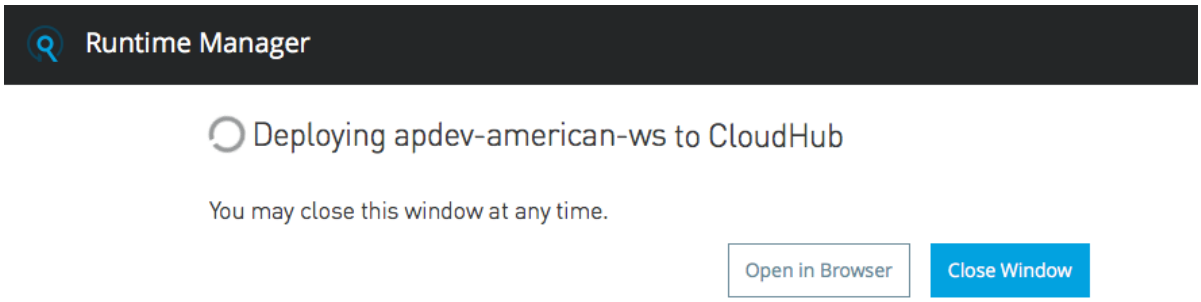
5. Set the worker size to 0.1 vCores.



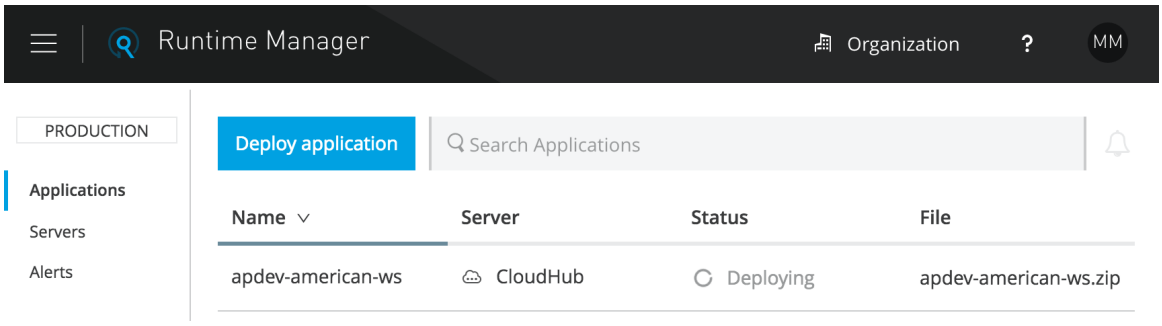
6. Click the Properties tab; you should see the env variable set to DEV.



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

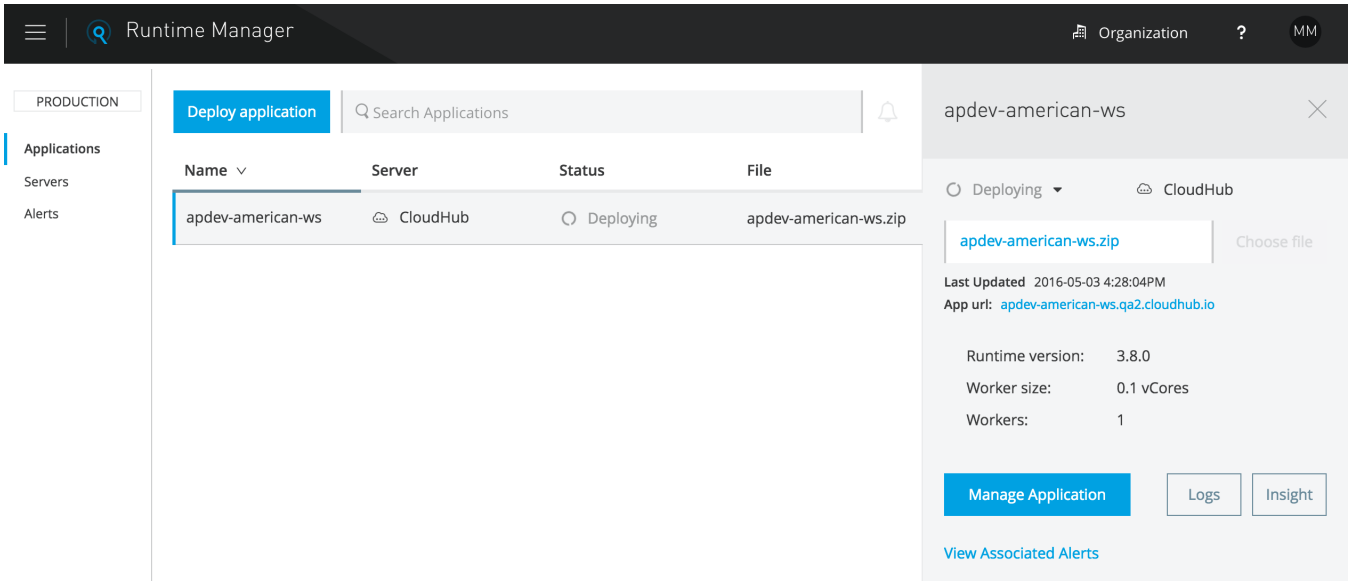


9. In the Anypoint Platform browser window that opens, locate the status of your deployment in the Runtime Manager.

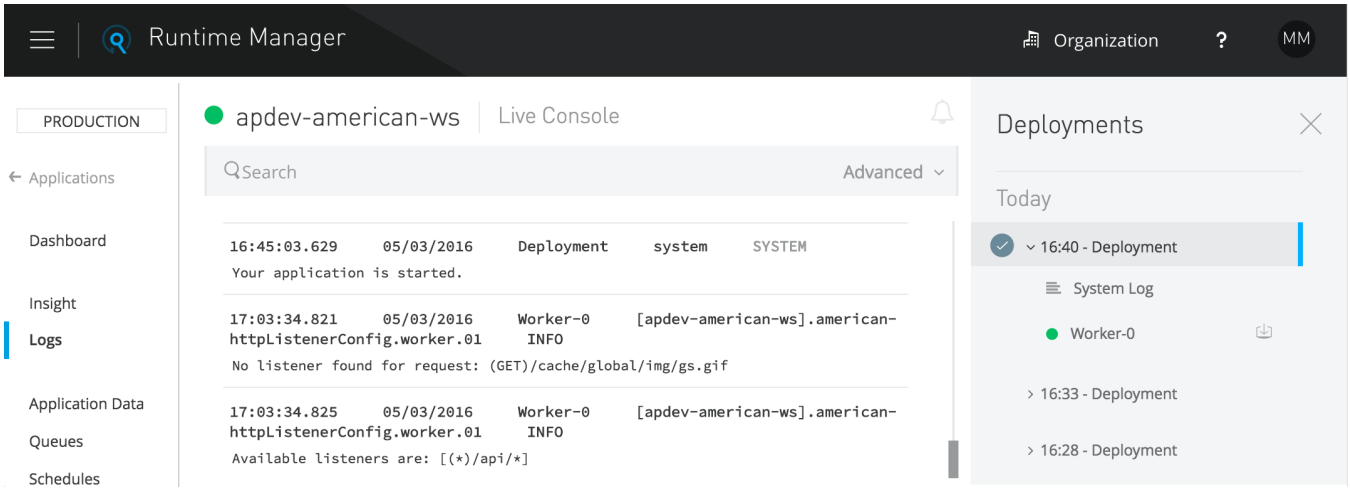


Watch the logs and wait for the application to start

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



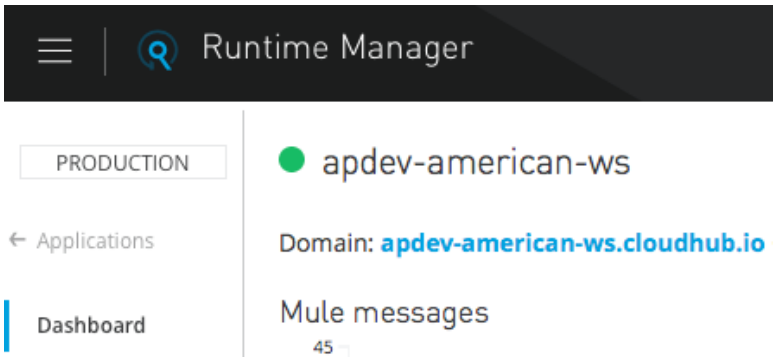
11. Click the Logs button.
12. Watch the logs as the application is deployed.
13. Wait until the application starts (or fails to start).



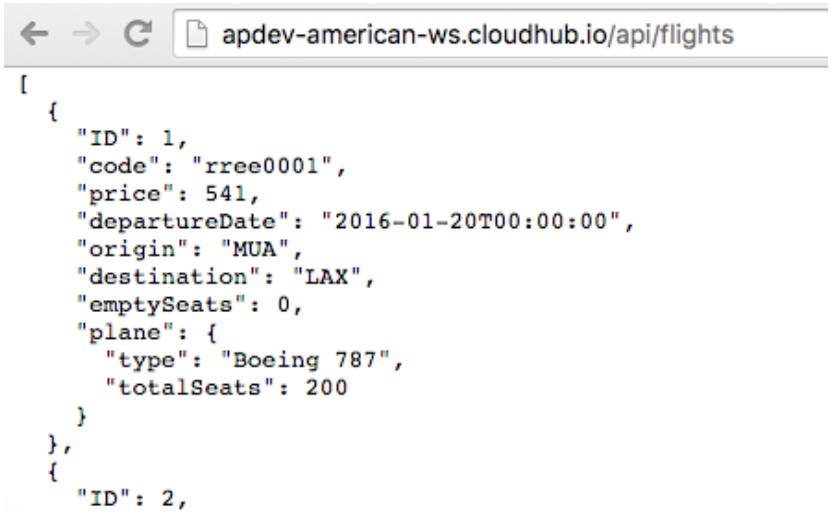
Note: If your application did not successfully deploy, read the logs to help figure out why the application did not deploy. If you had errors when deploying, troubleshoot them, fix them, and then redeploy.

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}*.cloudhub.io.



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}.cloudhub.io/api/flights; you should see the flights data.



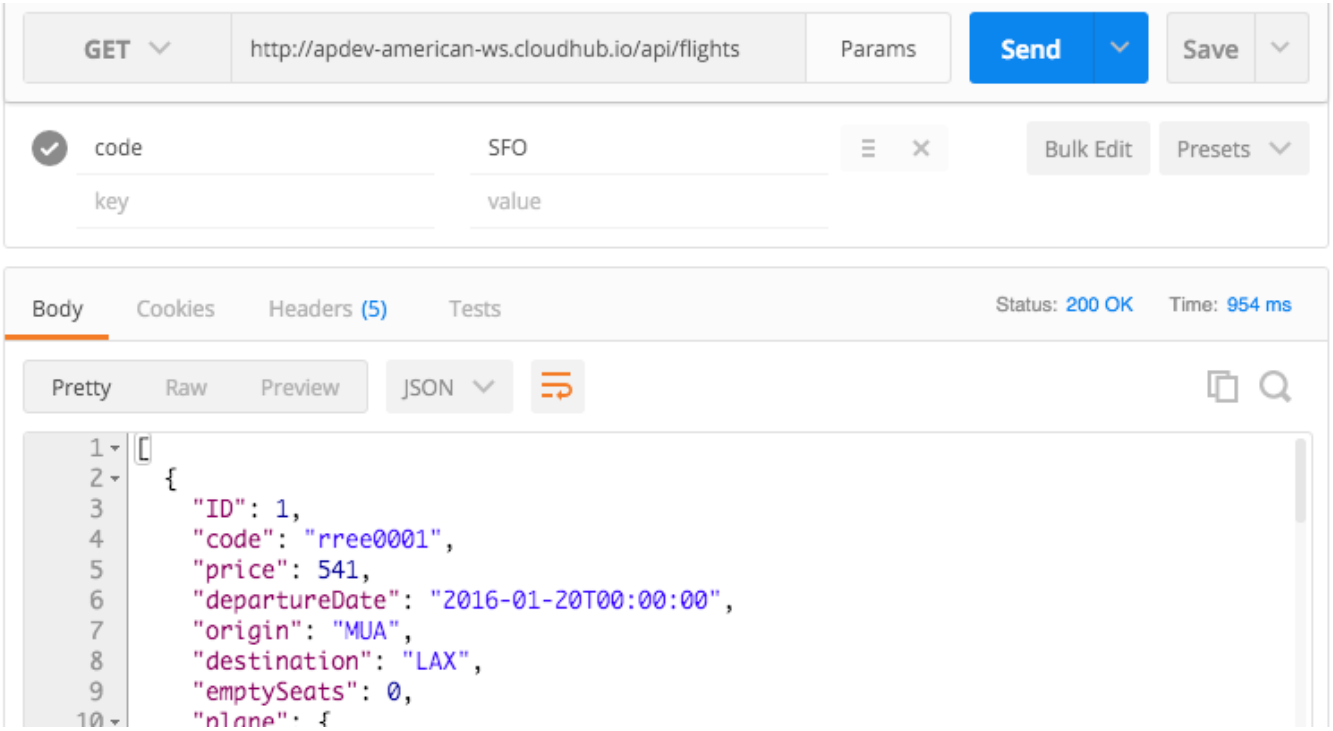
18. Copy the URL.
19. Return to Postman and paste the value in the URL.
20. Click Send; you should get the flights data.

    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.

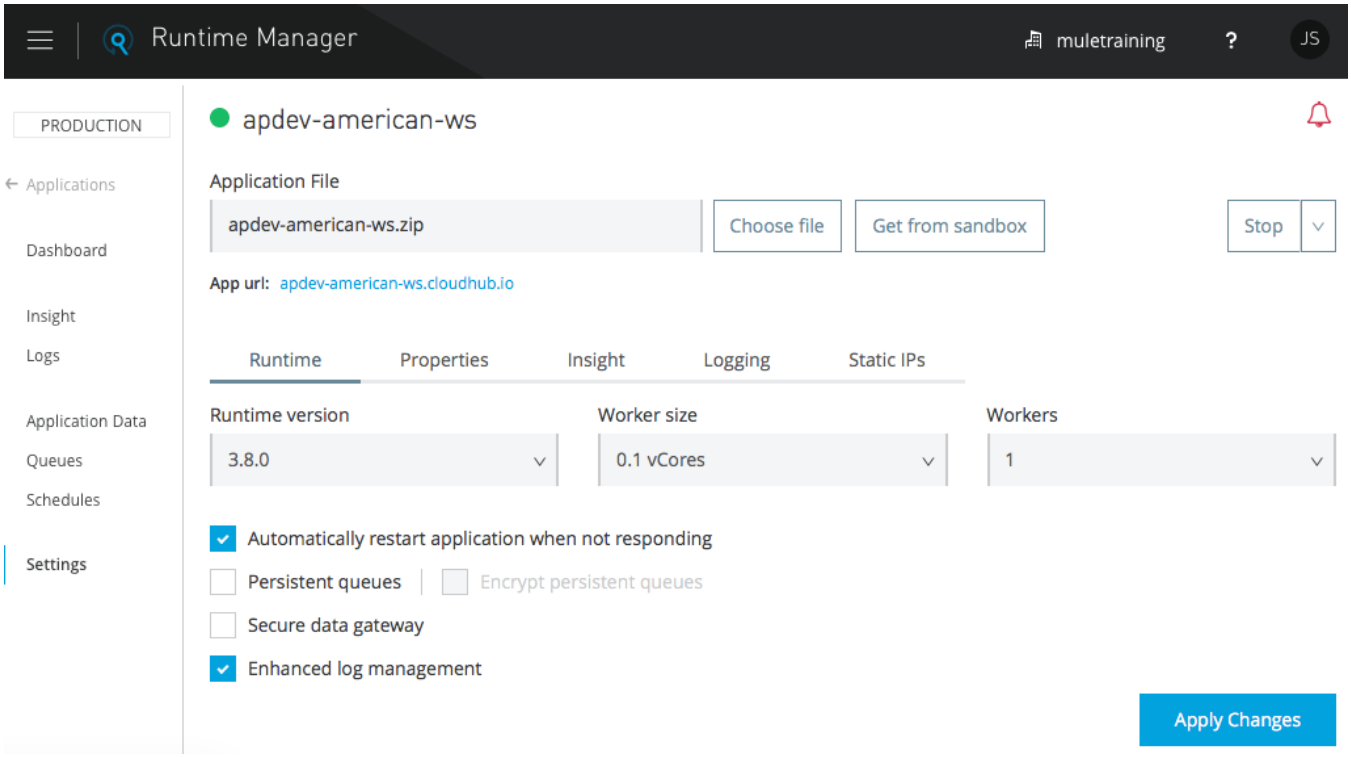


Update the API implementation in the cloud

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.



29. Wait until the application is uploaded and then redeploys 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.

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.

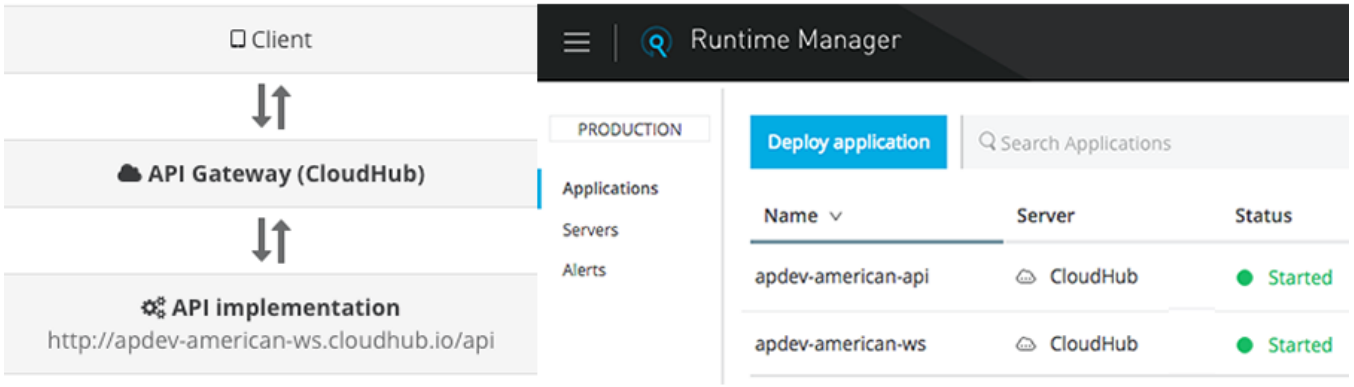
33. Return to Anypoint Platform in a web browser.
34. From the main menu, select API Manager.
35. Click the American Flights API version 1.0 link.
36. In the API Definition section, click the Edit in API designer link.
37. Turn off the mocking service.
38. Select all the text and delete it.
39. Paste 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

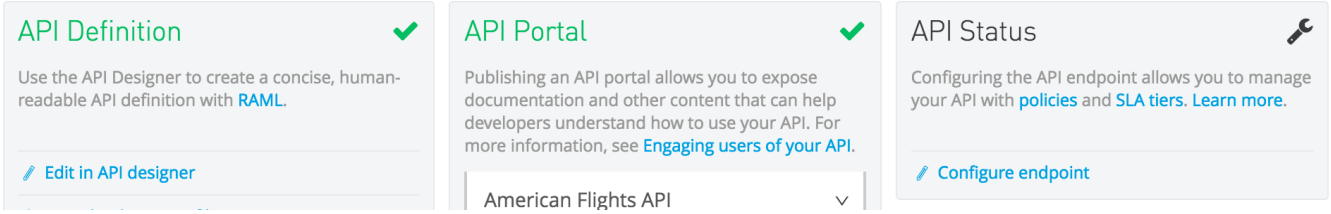
1. Return to the american.raml definition page for American Flights API – 1.0 in Anypoint Platform.
2. Add a baseUri and set it equal to http://apdev-american-ws-{lastname#}.cloudhub.io/api.

```
#%RAML 0.8
baseUri: http://apdev-american-ws.cloudhub.io/api
title: American Flights API
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.

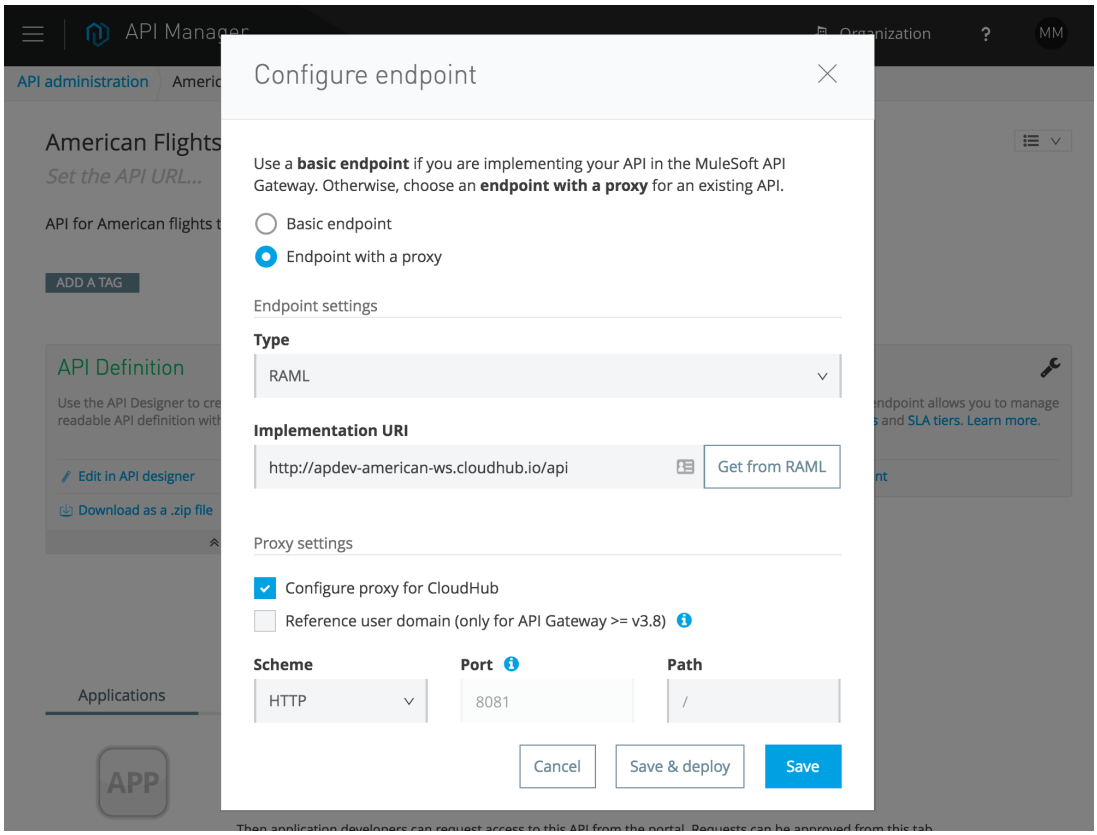


6. In the Configure endpoint dialog box, confirm the following values are set:

- Endpoint with a proxy
- Type: RAML
- Implementation URI: http://apdev-american-ws-*{lastname#}*.cloudhub.io/api

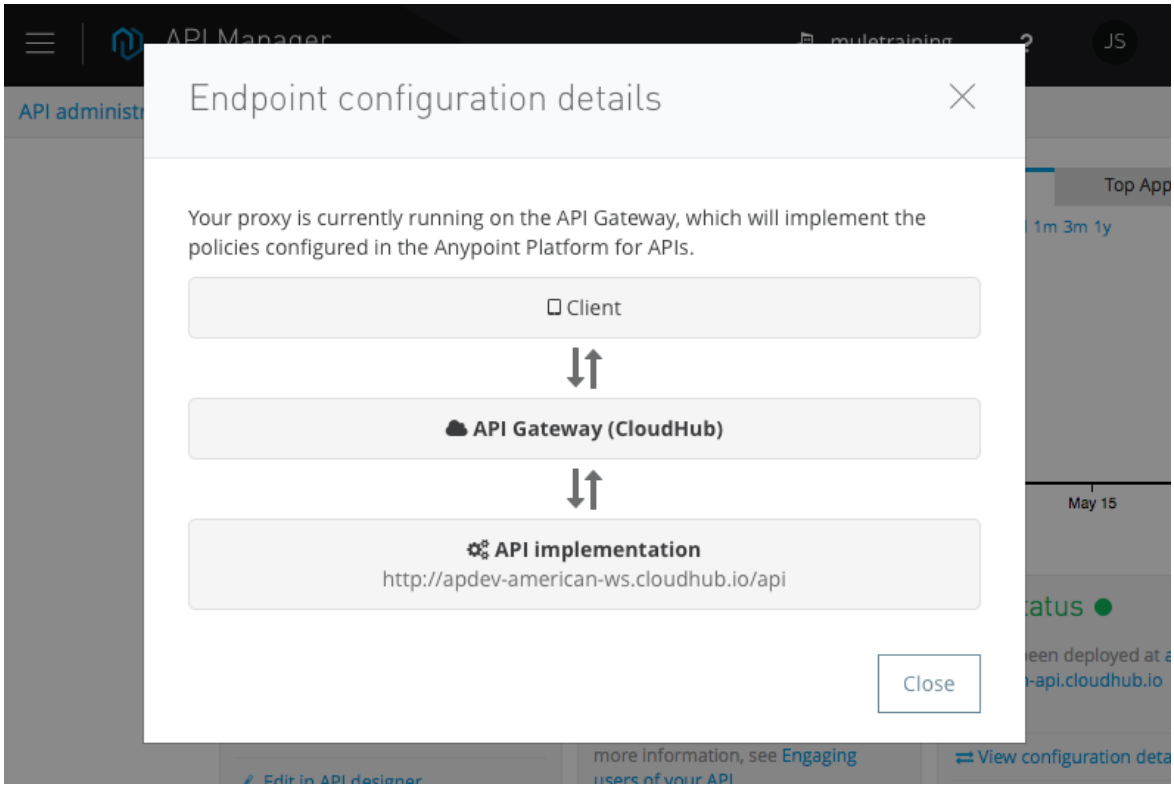
7. Select the Configure proxy for CloudHub checkbox.
8. Confirm the following values are set:

   - Scheme: RAML
   - Port: 8081
   - Path: /



9. Click Save.

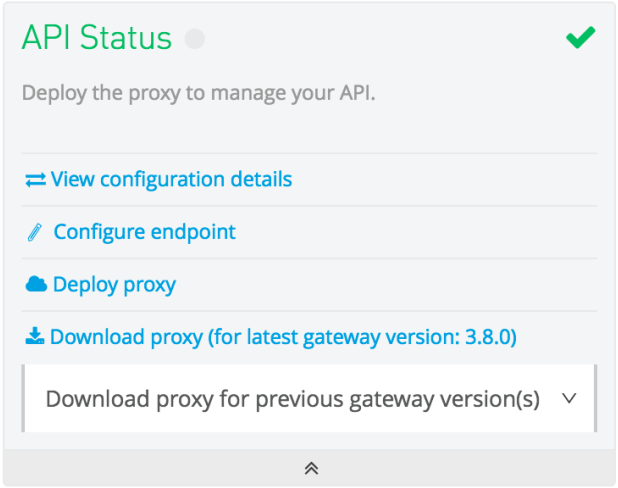
10. In the API Status section, click the View configuration details link.



11. Click Close.

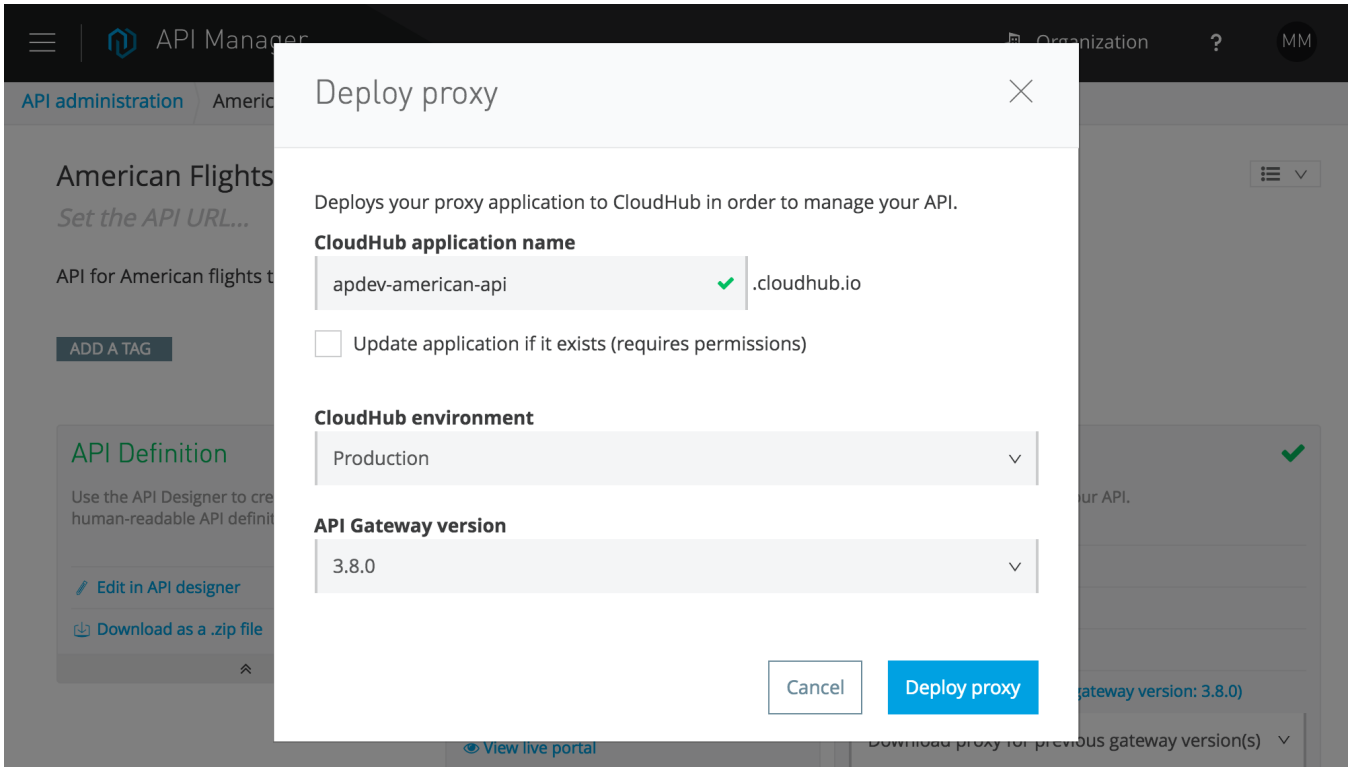
Deploy the proxy

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

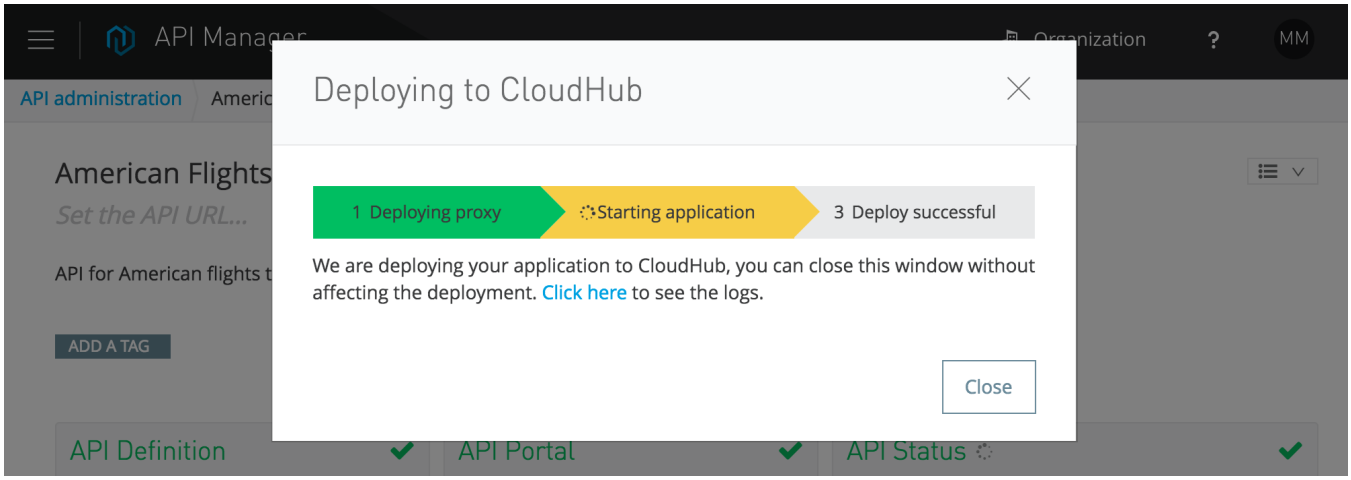


13. In the Deploy proxy dialog box, change the CloudHub application name to apdev-american-api-*{lastname#}*.cloudhub.io.

14. Click Deploy proxy.



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

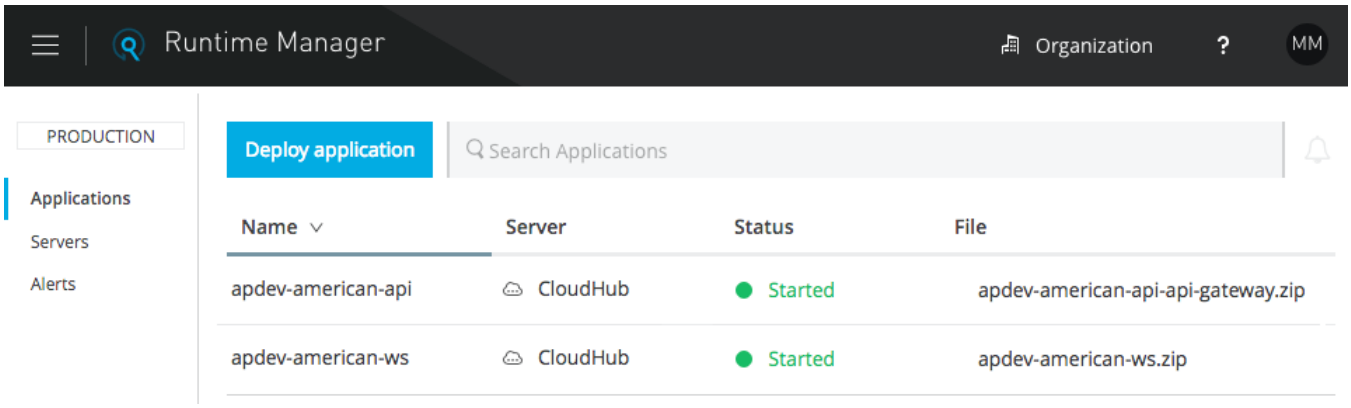


16. Watch the logs.
17. Wait until the proxy application starts.

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.



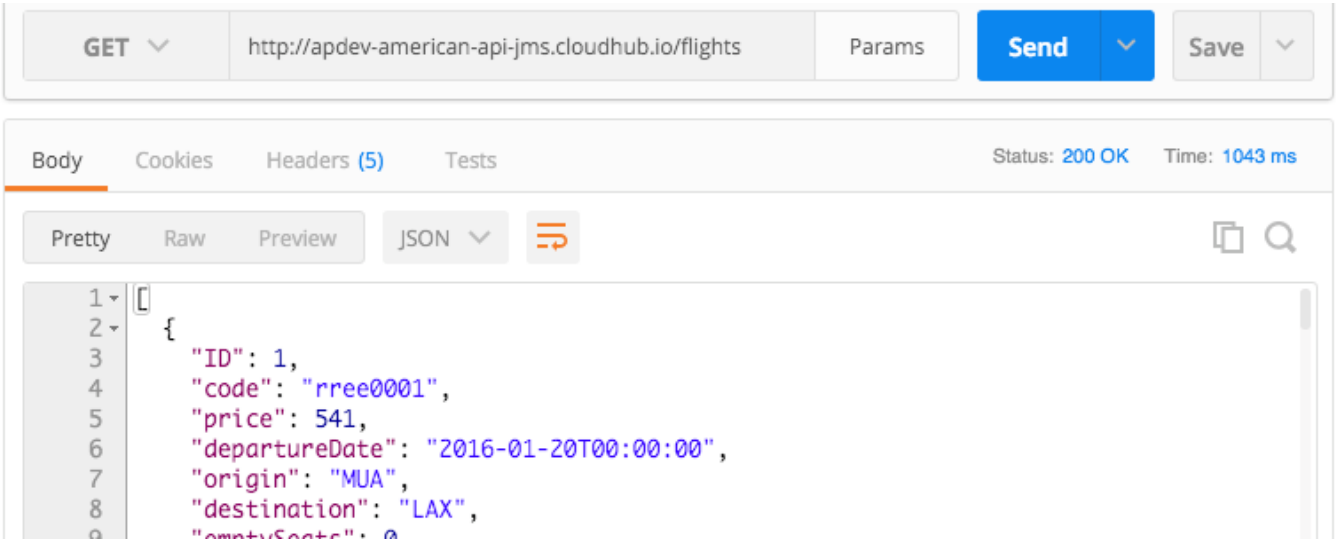
Look at the API request data

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.

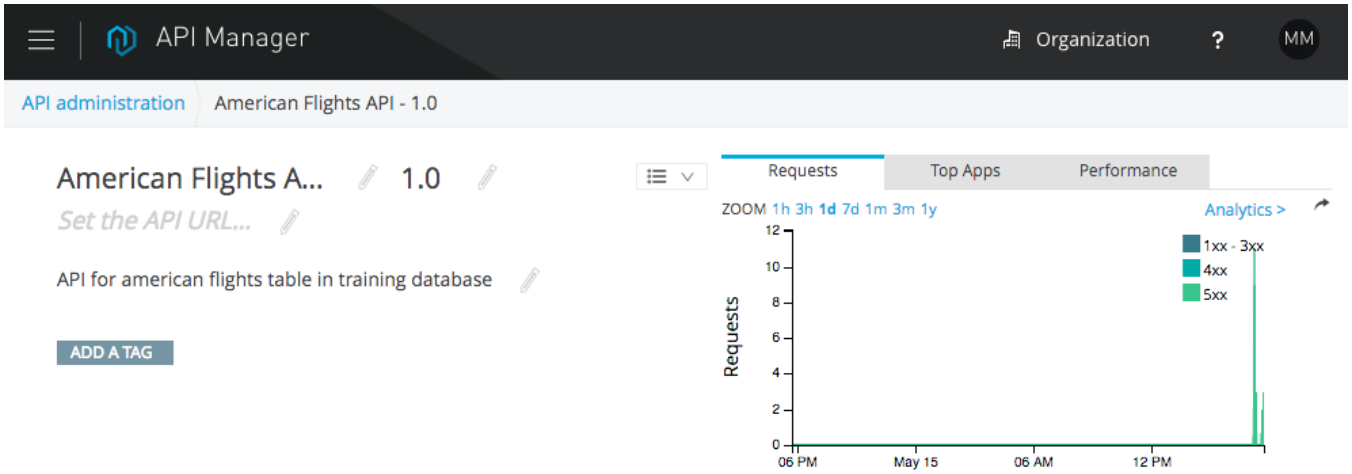
    Note: Be sure to remove the /api from the URL.



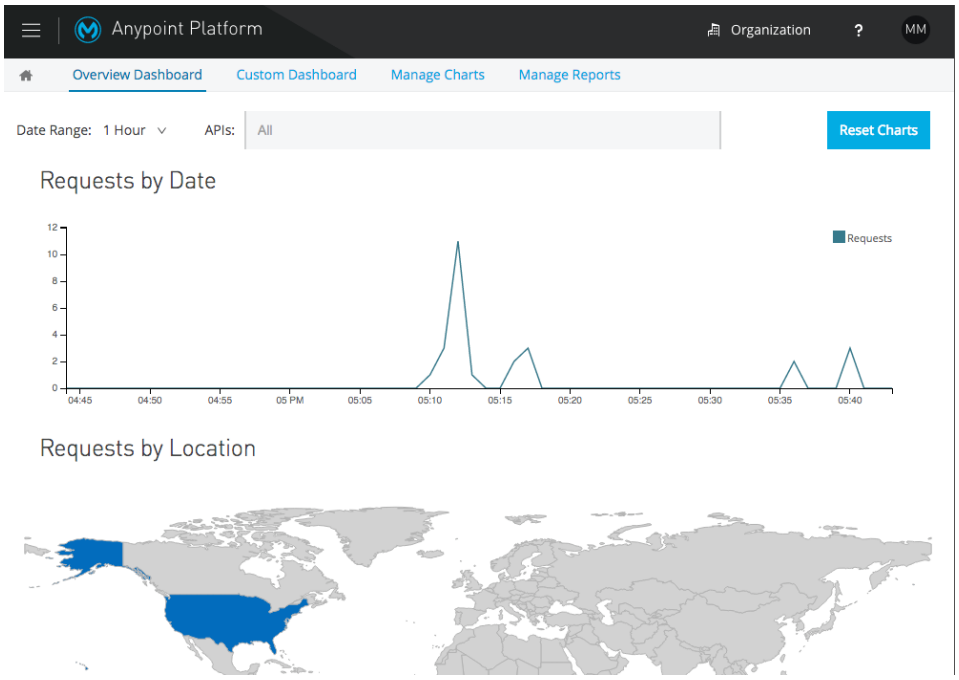
25. Make several more calls.

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.

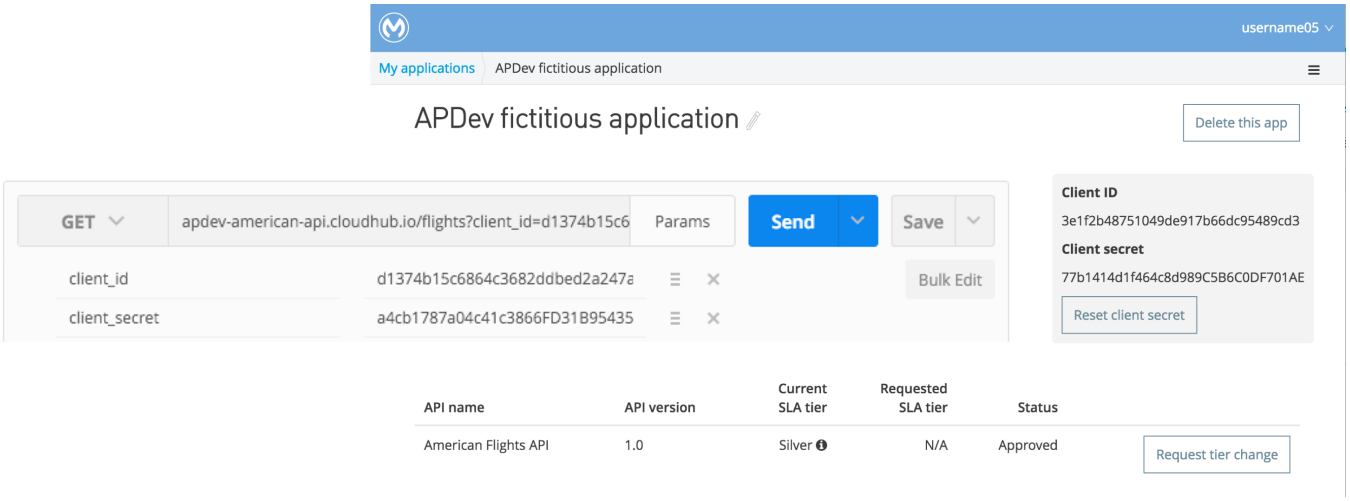


31. Close the web browser tab or window.

Walkthrough 4-4: Restrict API access

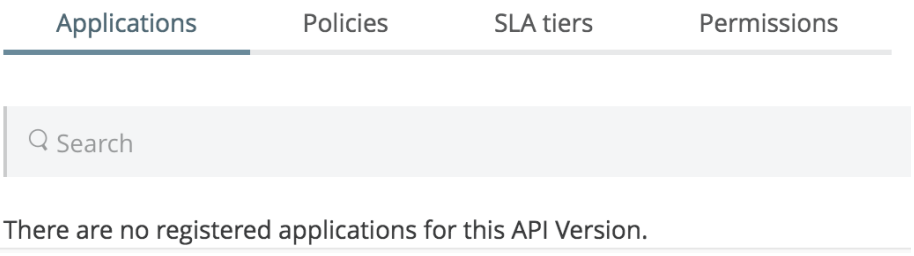
In this walkthrough, you govern access to the API proxy. You will:

- 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.

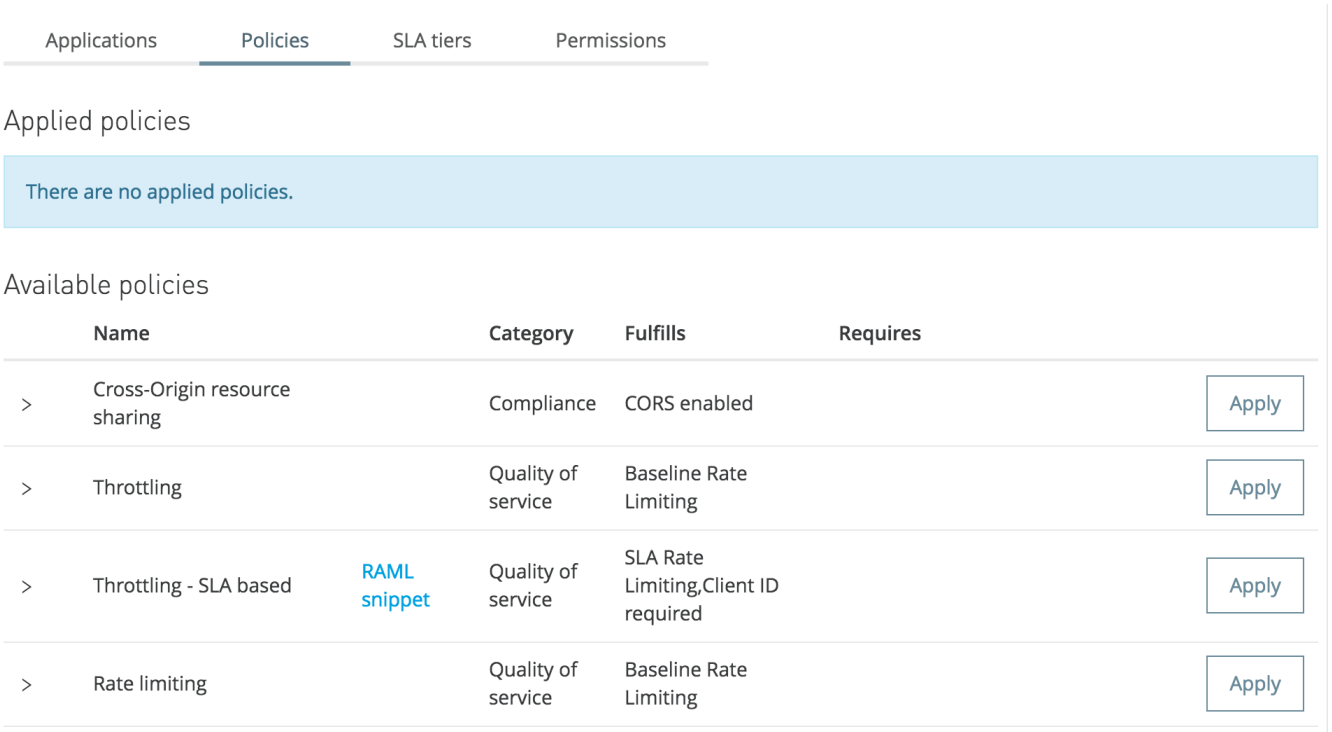


Create a rate limiting policy

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.

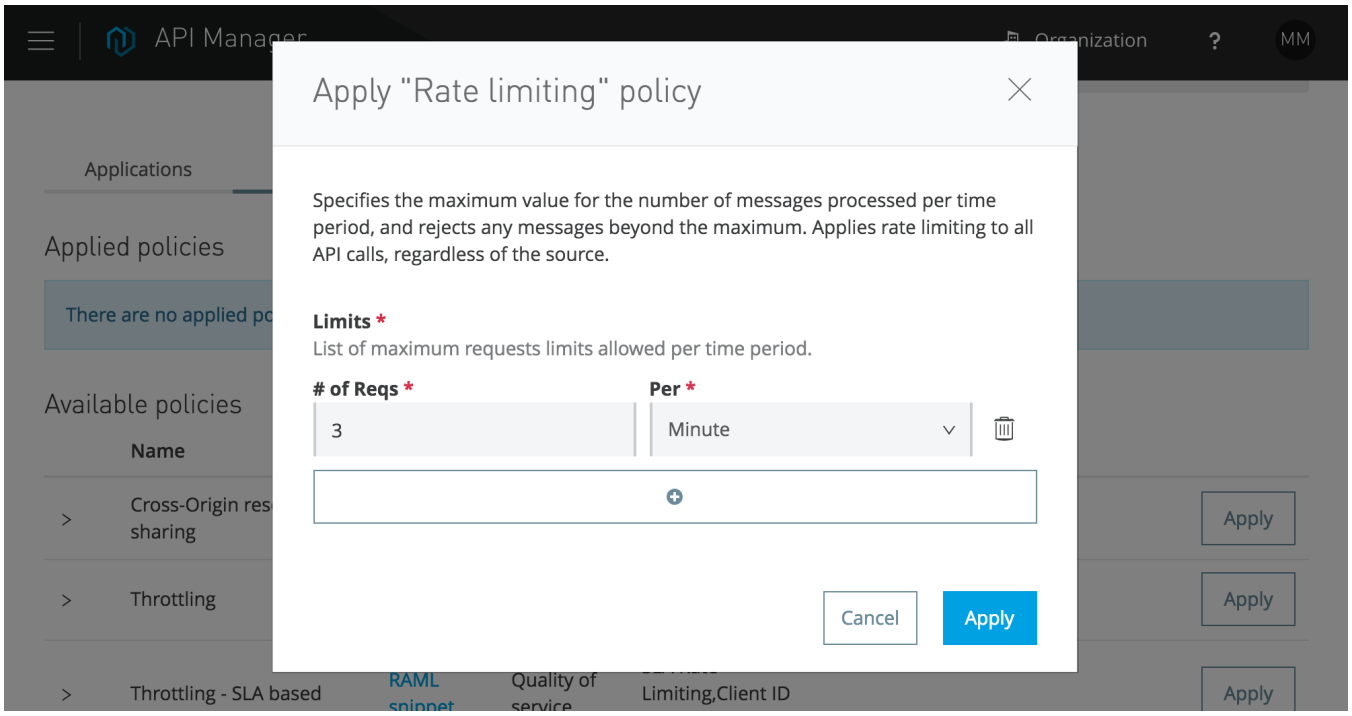


3. Click the Apply button next to the rate limiting policy.



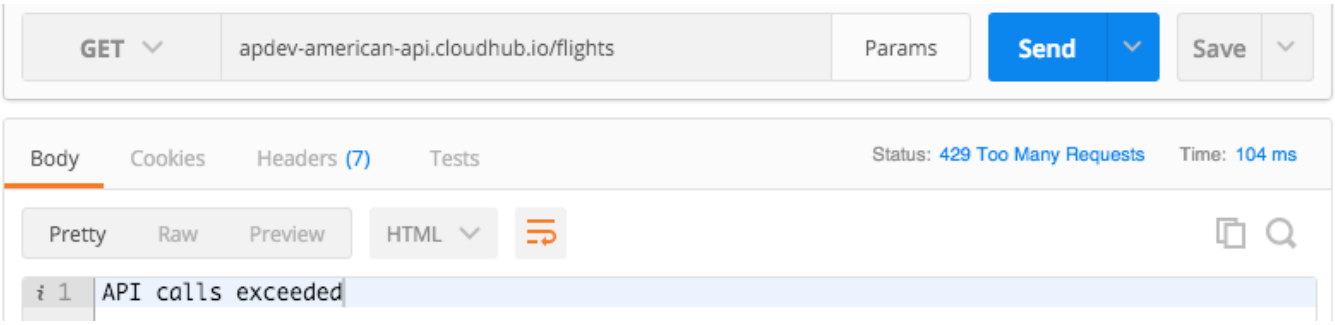
4. In the Apply "Rate limiting" policy dialog box, set the following values and click Apply:

   - # of Reqs: 3
   - Per: Minute



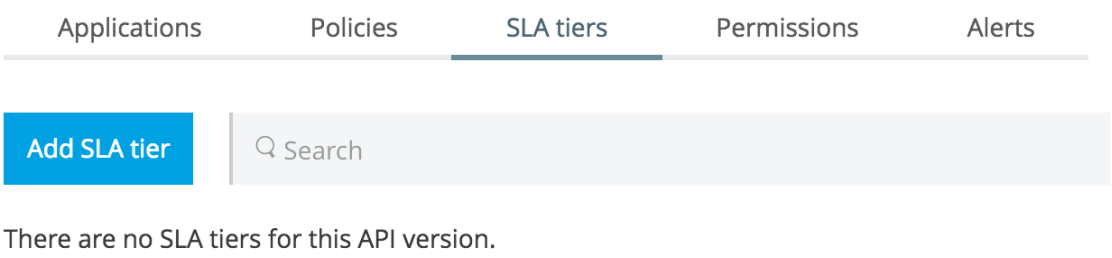
Test the rate limiting policy

5. In Postman, make another request to http://apdev-american-api-{lastname#}.cloudhub.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.



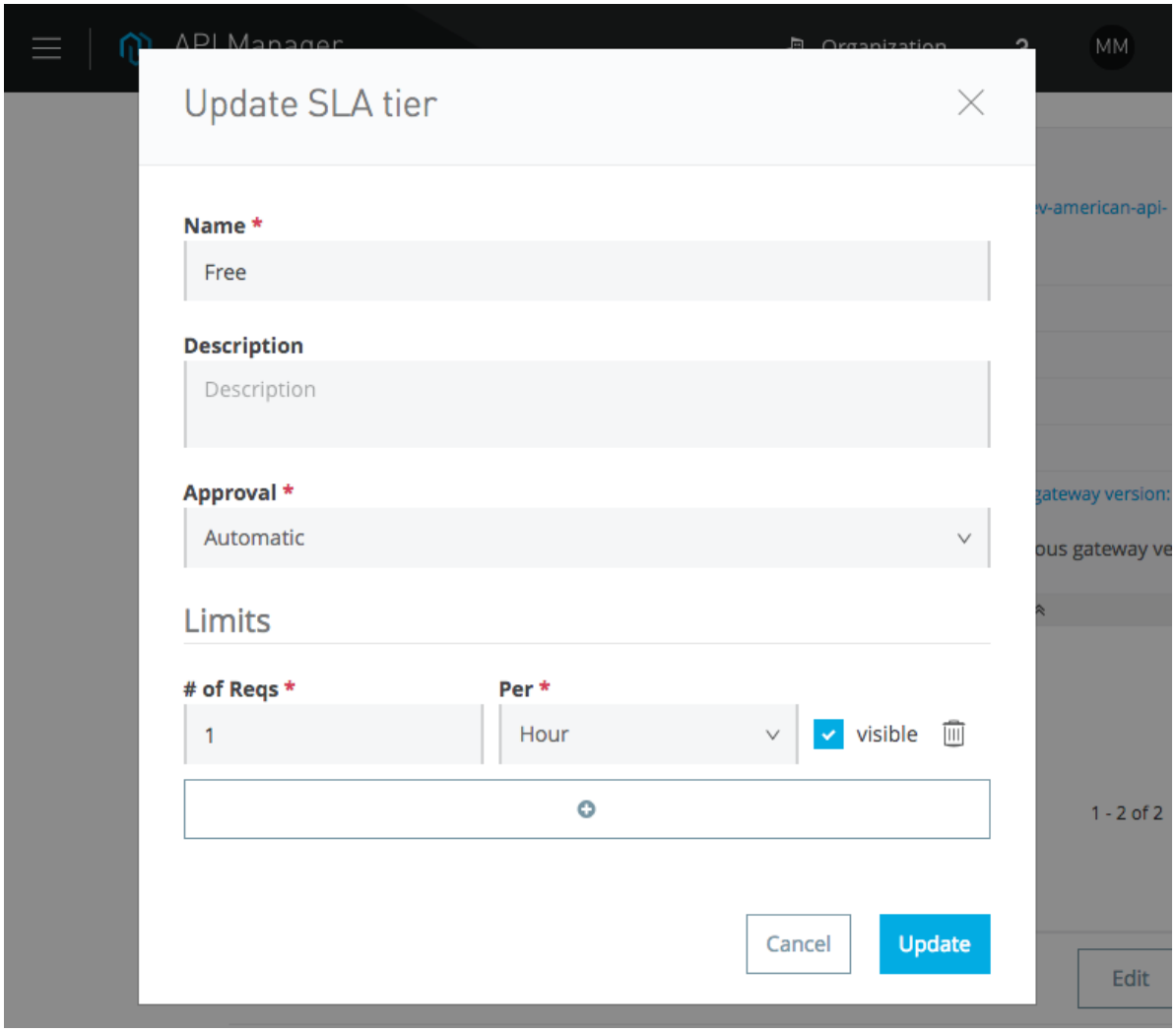
Create SLA tiers

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.



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

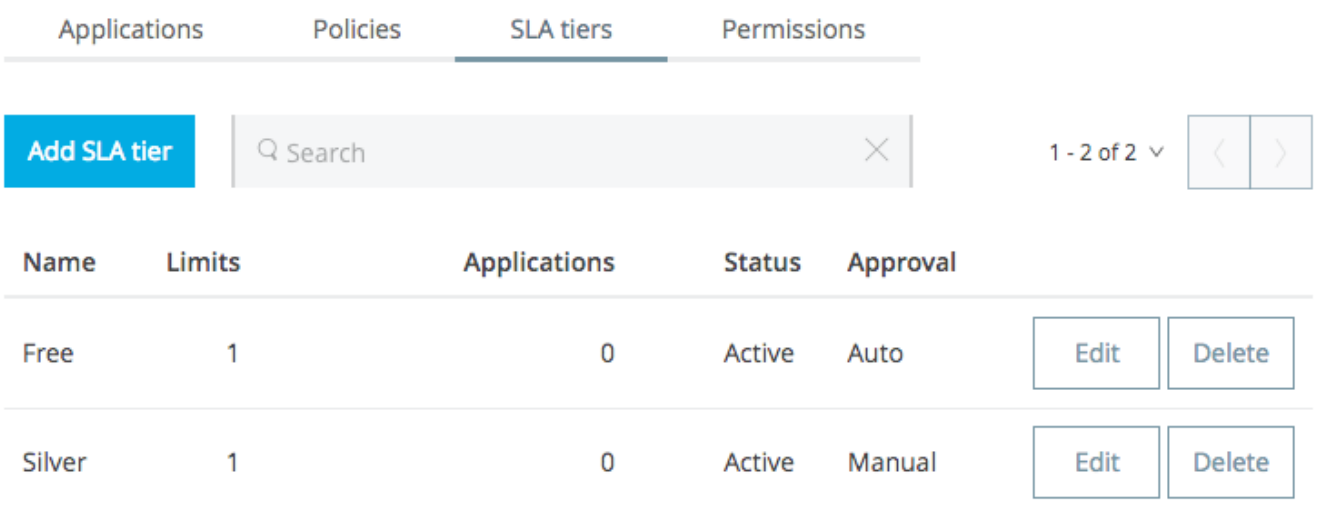
- Name: Free
- Approval: Automatic
- # of Reqs: 1
- Per: Hour



11. Click the Add button.

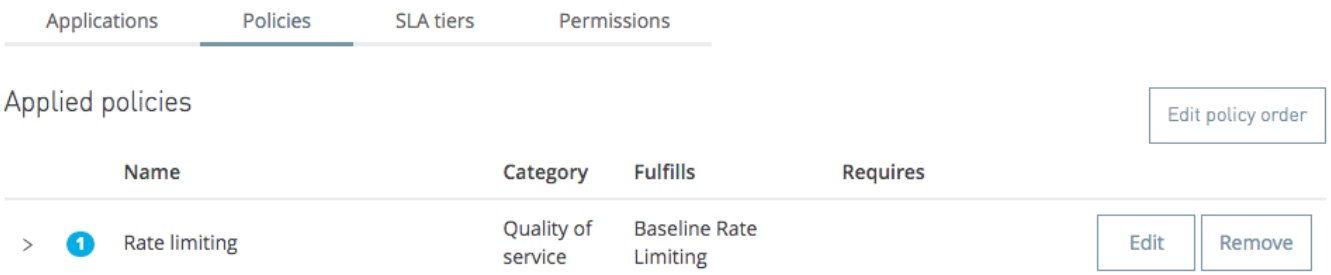
12. Create a second SLA tier with the following values:

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

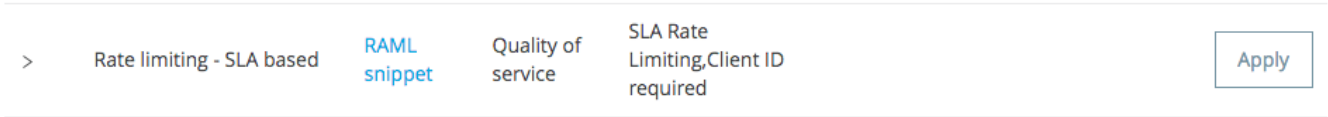


Change the policy to rate limiting – SLA based

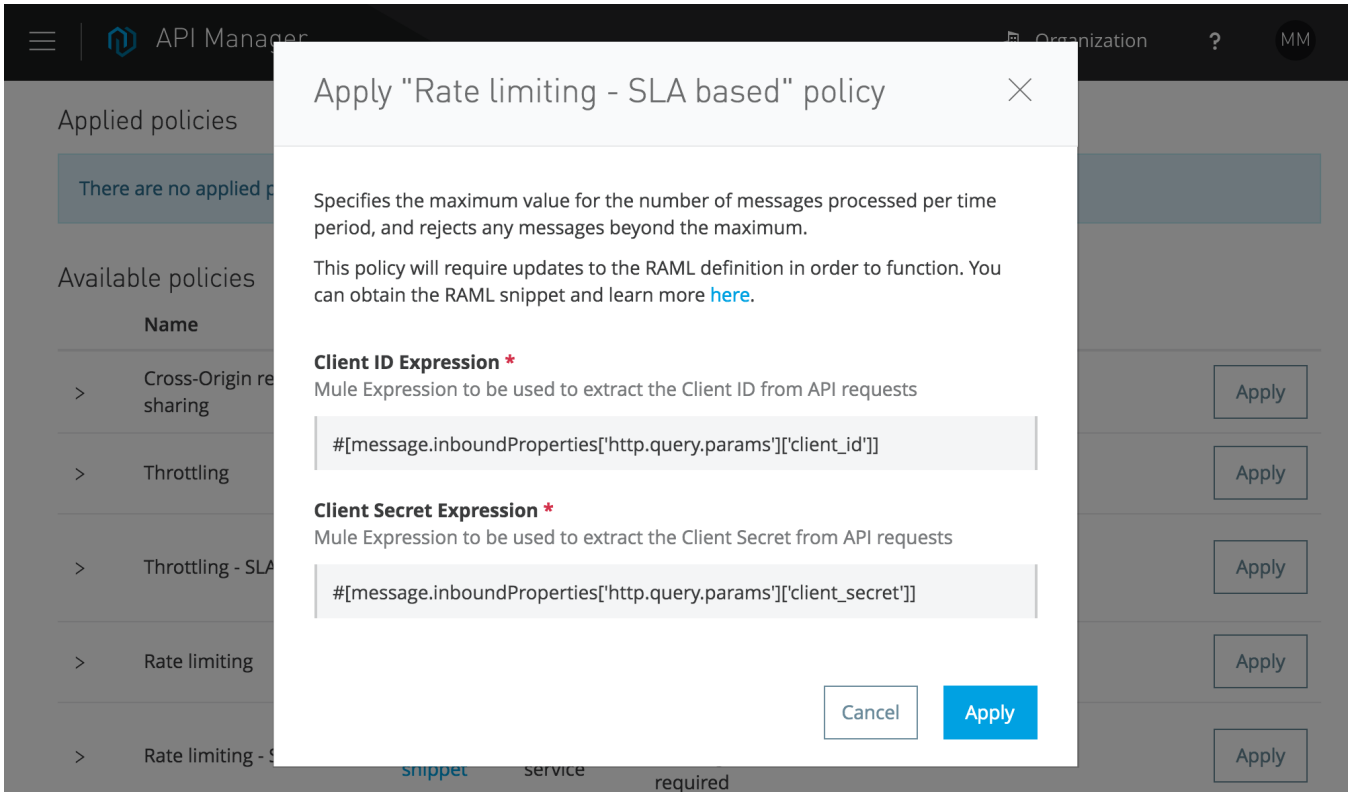
13. Click the Policies tab.
14. Click the Remove button for the Rate limiting policy.



15. In the Remove policy dialog box, click Remove.
16. Click the Apply button for the Rate limiting – SLA based policy.



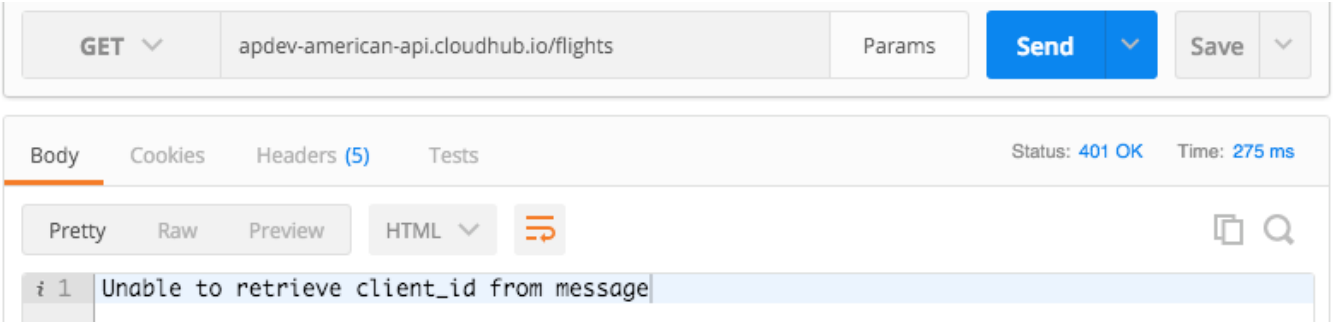
17. In the Apply "Rate-limiting – SLA based" policy dialog box, look at the expressions and see that a client ID and secret need to be sent with API requests as query parameters.



18. Click Apply.

Test the rate limiting – SLA based policy

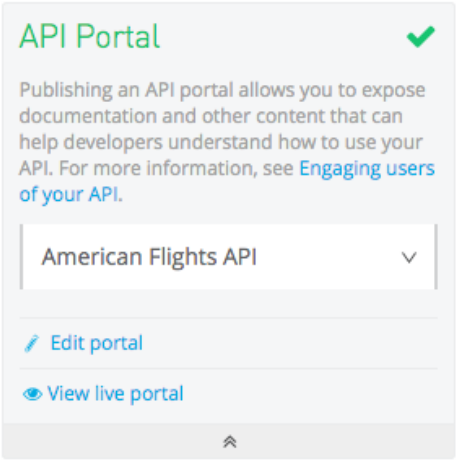
19. In Postman, make another request to your API; you should get a 401 response with a message that a client_id was not provided.



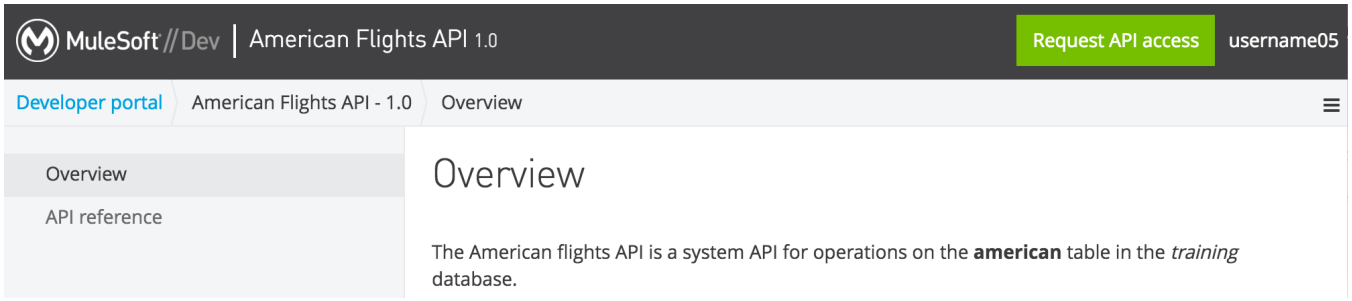
Request access to the API

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

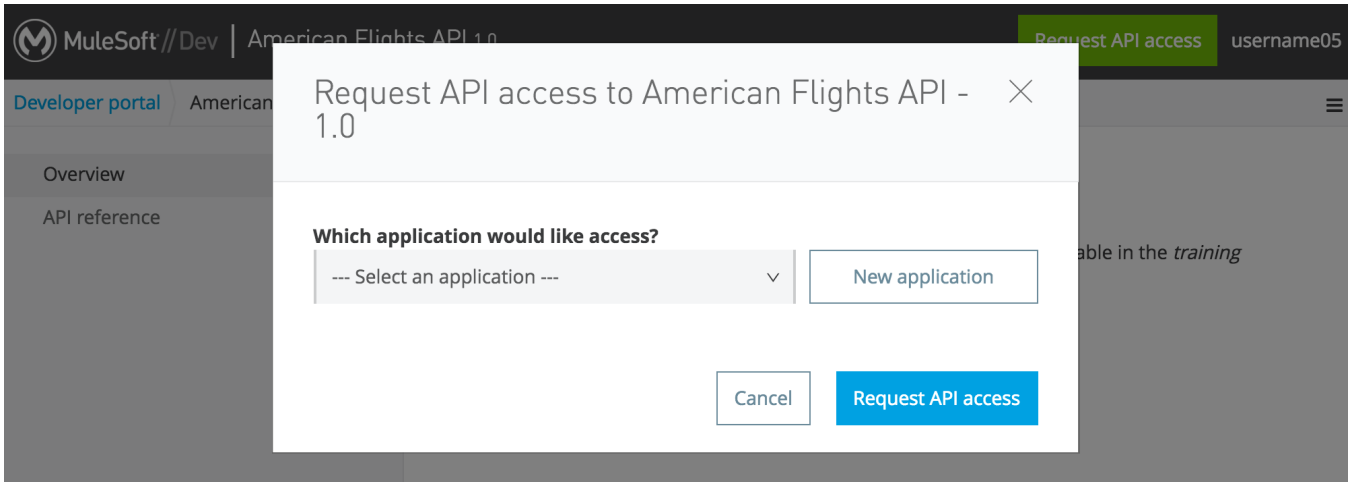
21. In the API Portal section, click the View live portal link.



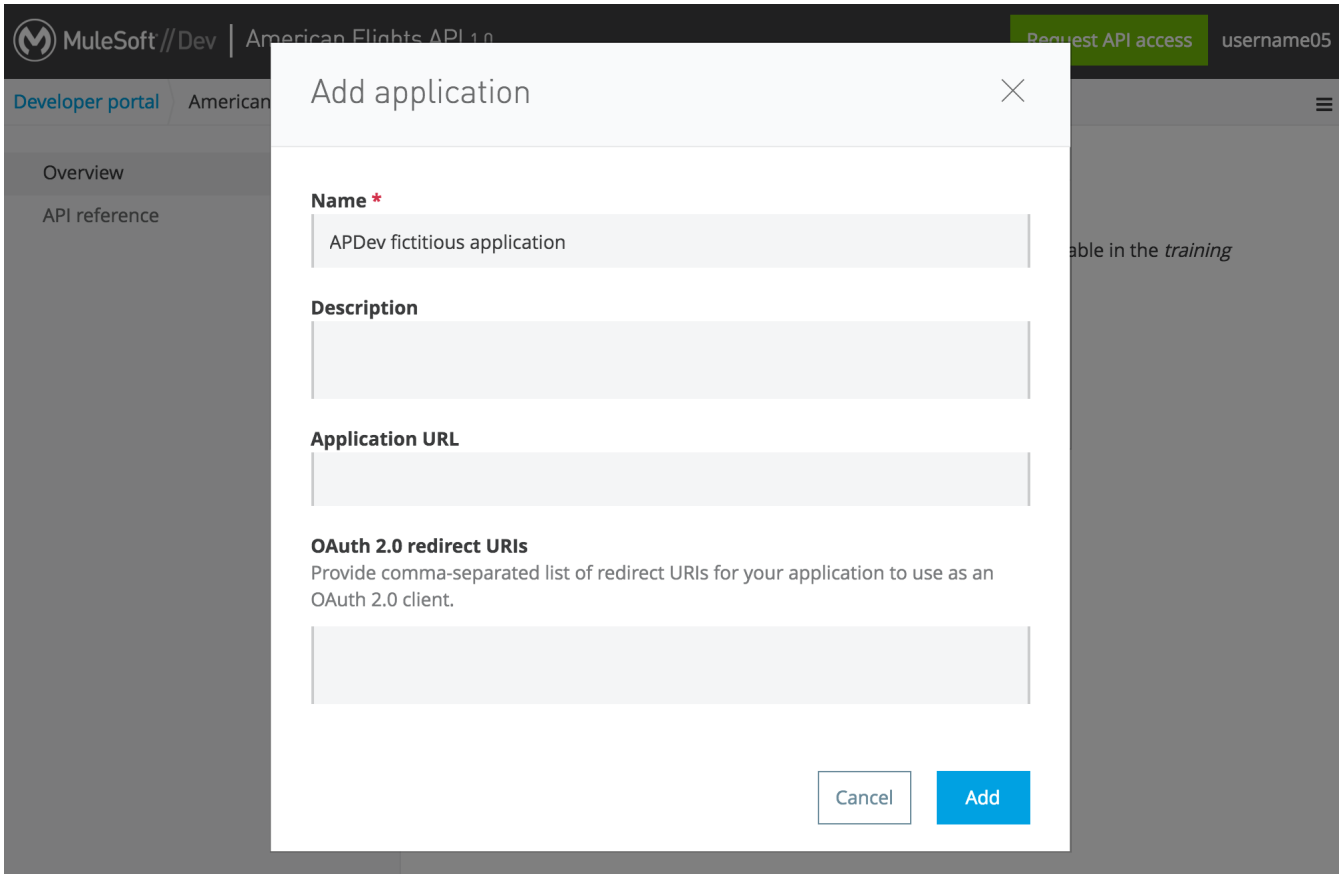
22. In the portal, click the Request API access button.



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

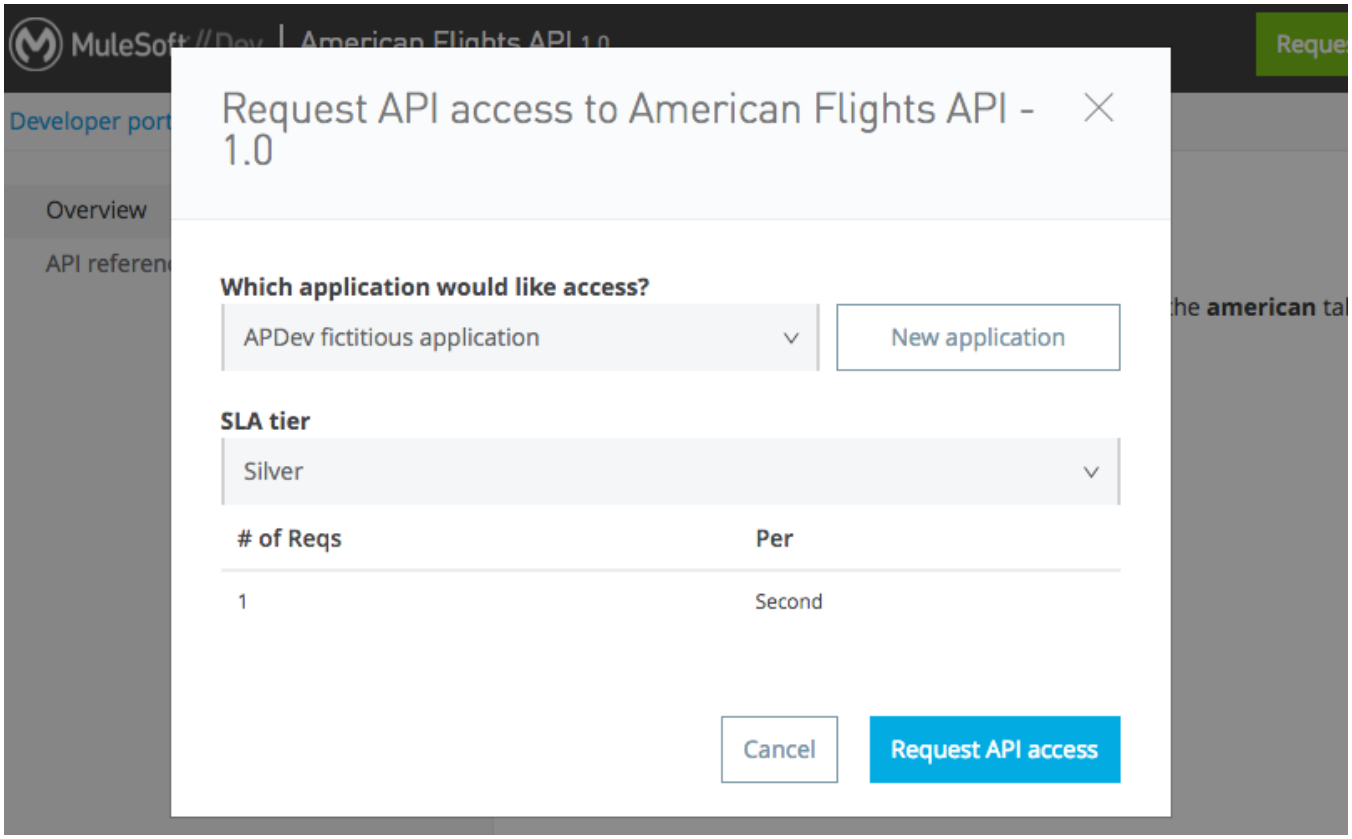


24. In the Add application dialog box, set the name to APDev fictitious application (or some other name).



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

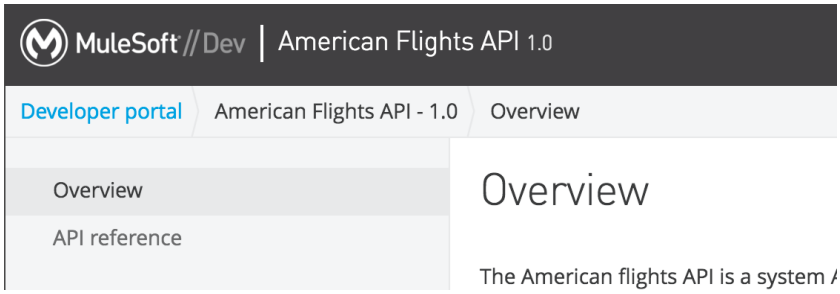
27. Click Request API access.



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

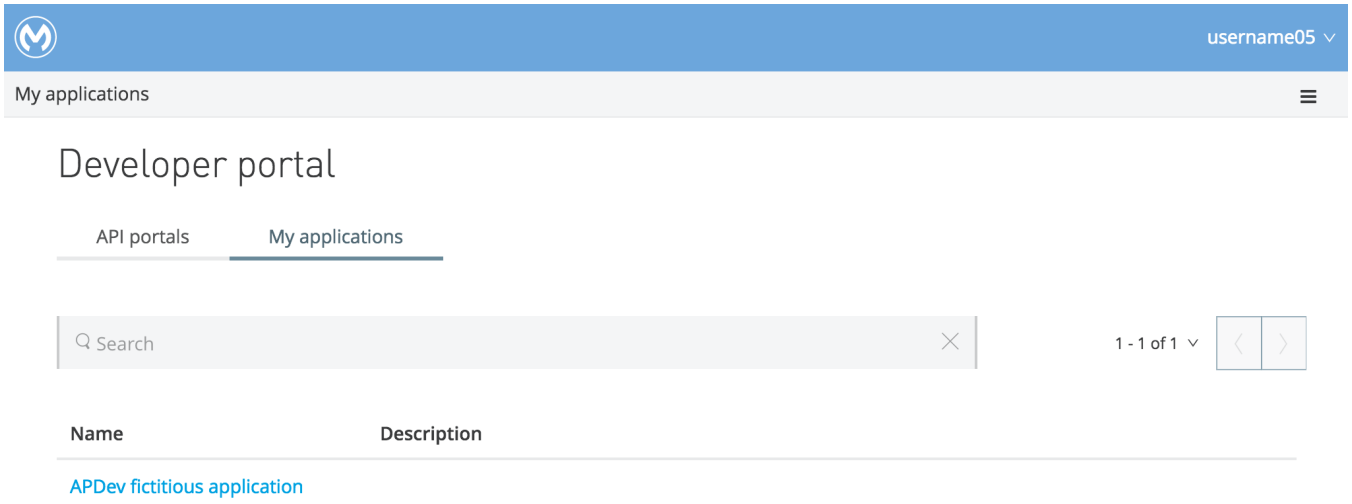
Retrieve client ID and secret

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

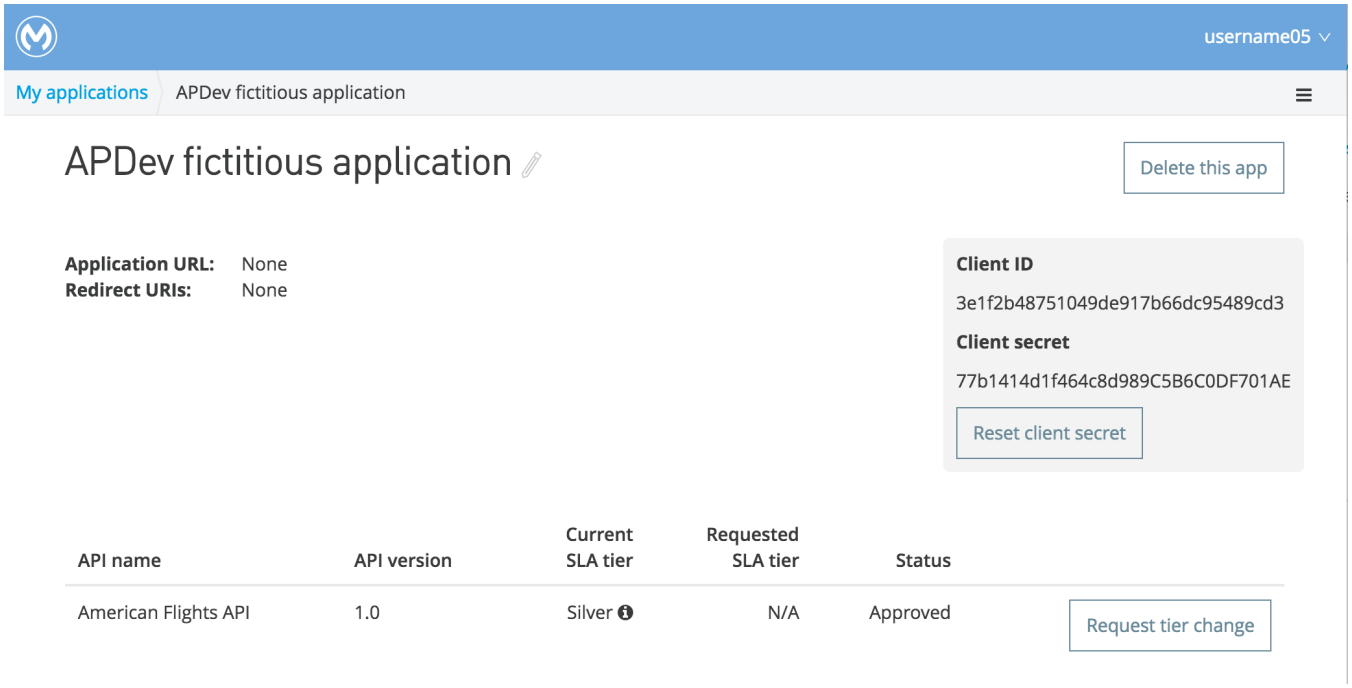


30. In the Developer portal, click My applications.

31. Click the APDev fictitious application link.



32. Locate the application client ID and client secret.

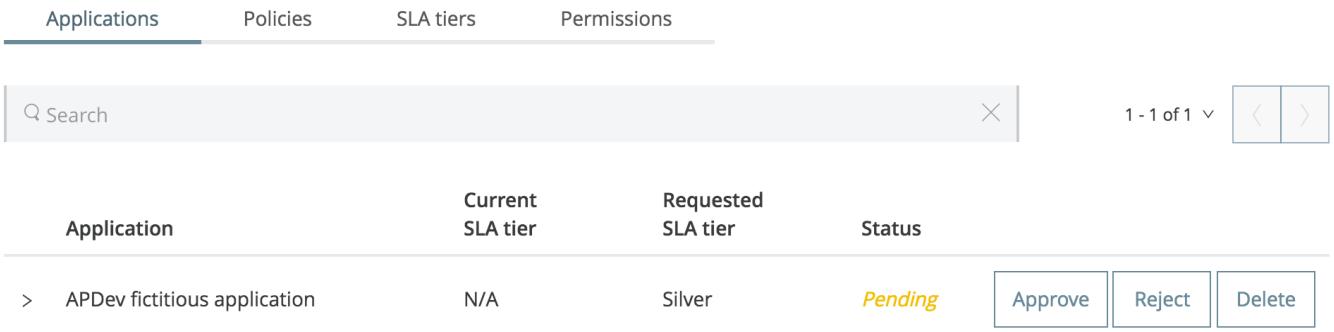


33. Copy the client ID and paste it in the course snippets.txt file under your client_id.
34. Copy the client secret and paste it in the course snippets.txt file under your client_secret.

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.



38. Click Approve.

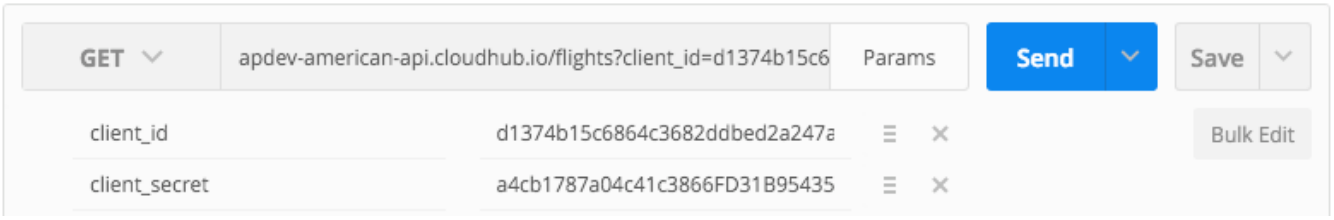
Test the API SLA

39. In Postman, make sure the URL is set to http://apdev-american-api-{lastname#}.cloudhub.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*



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