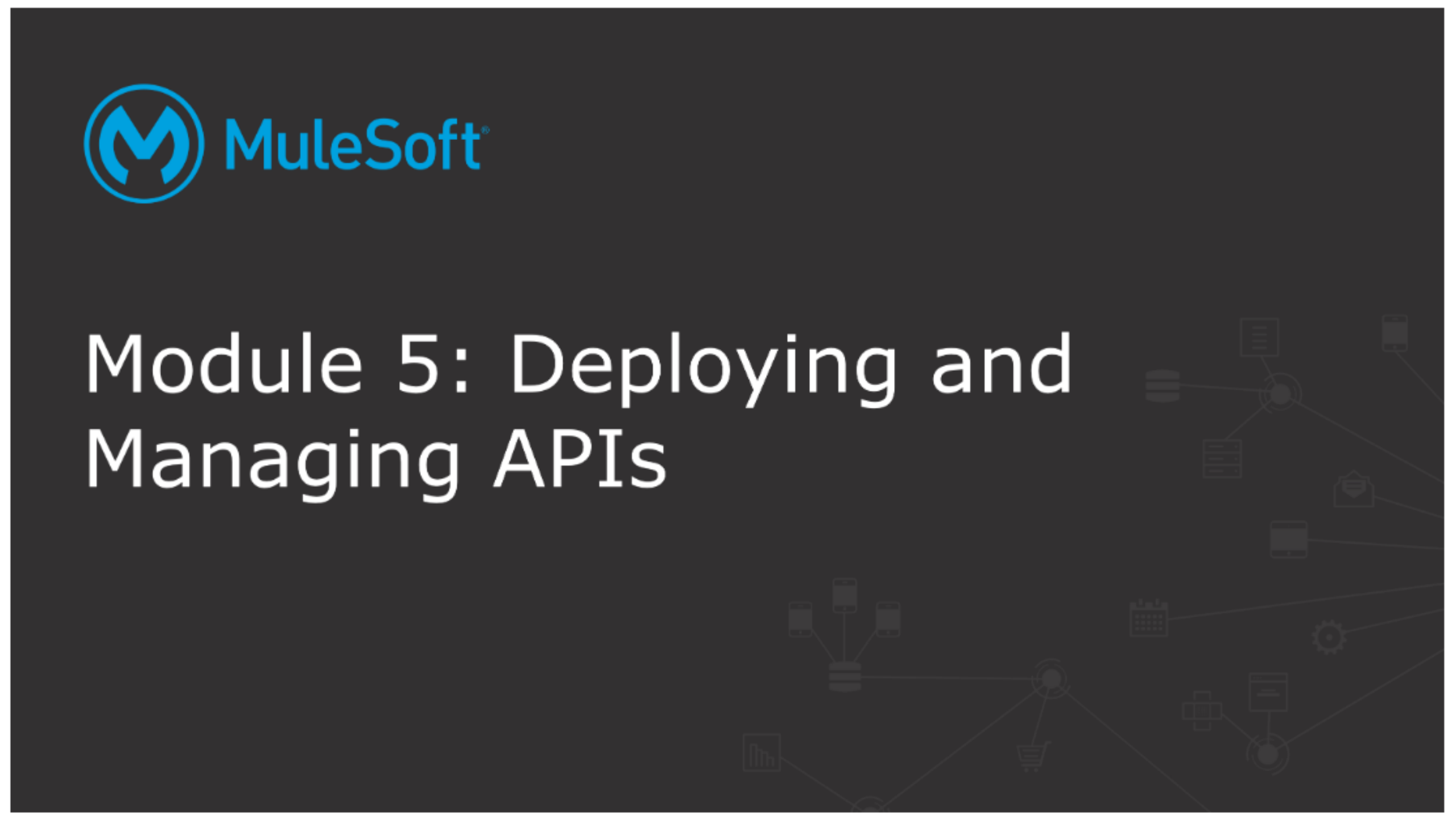


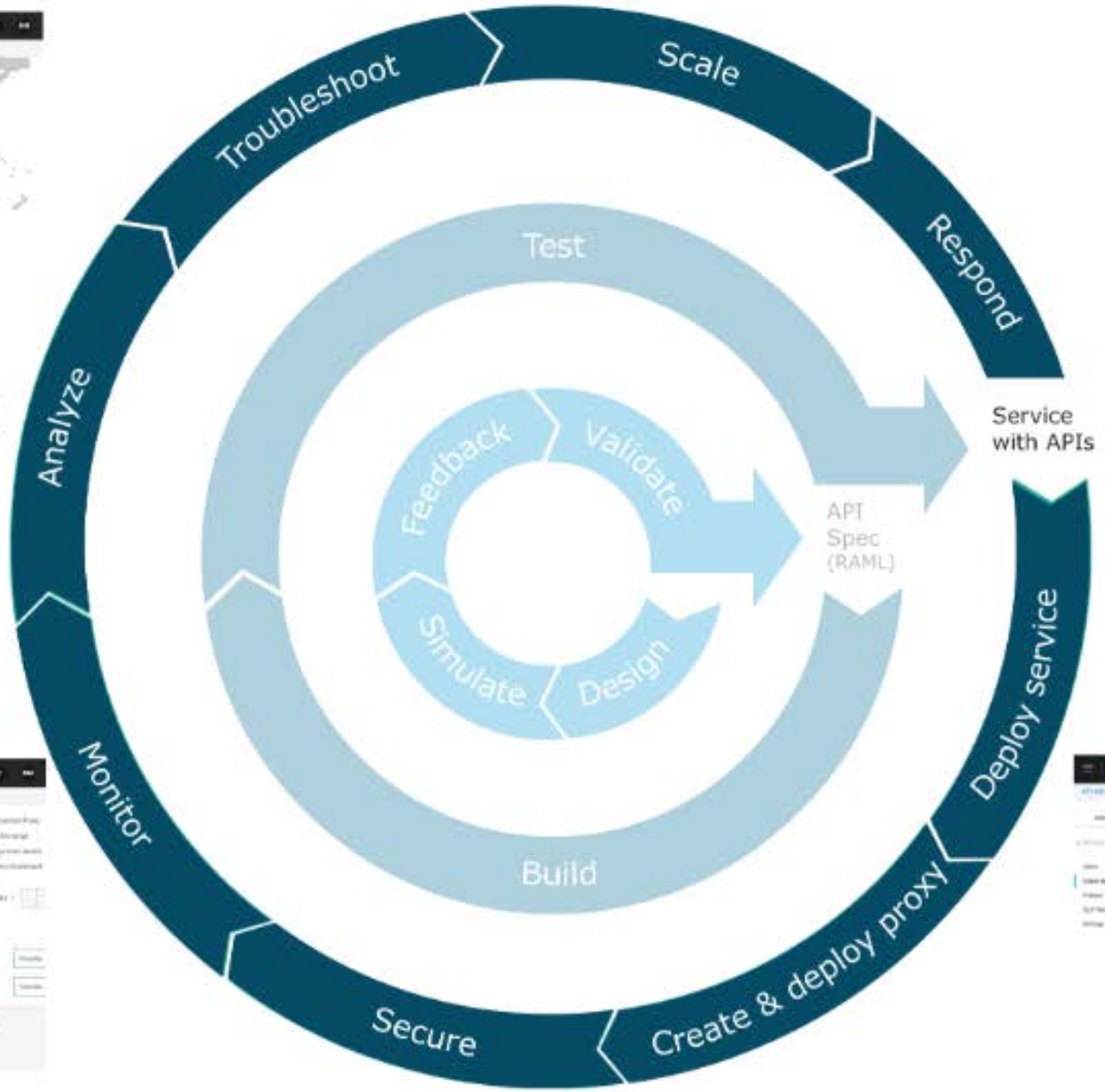
Module 5: Deploying and Managing APIs

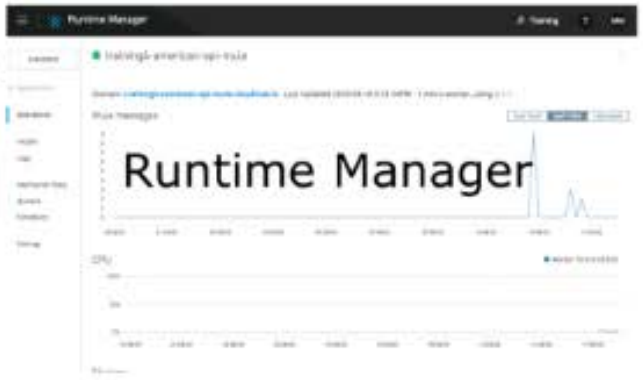
Goal

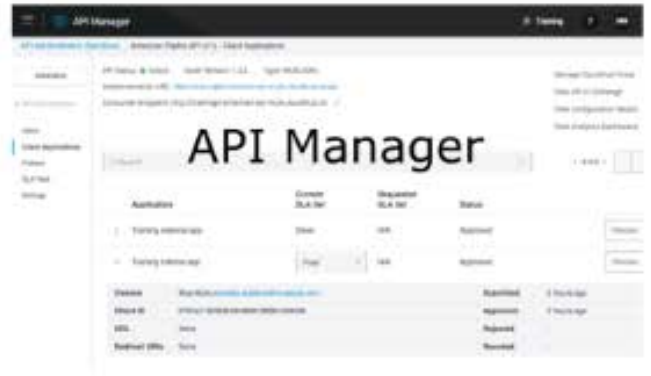

API Analytics


Runtime Manager


API Manager


Troubleshoot
Scale
Respond
Analyze
Monitor
Secure
Create & deploy proxy
Deploy service
Test
Feedback
Validate
Simulate
Design
Build
API
Spec
(RAML)
Service
with APIs


Runtime Manager


API Manager

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

- Describe the options for deploying Mule applications
- Deploy Mule applications to CloudHub
- Use API Manager to create and deploy proxies for APIs
- Use API Manager to restrict access to API proxies

Introducing deployment options

Deploying applications

- During development, applications are deployed to an embedded Mule runtime in Anypoint Studio

- For everything else (testing, Q&A, and production), applications can be deployed to

  - **CloudHub**
    - Platform as a Service (PaaS) component of Anypoint Platform
    - MuleSoft-hosted Mule runtimes on AWS (Amazon Web Services platform)
    - A fully-managed, multi-tenanted, globally available, secure and highly available cloud platform for integrations and APIs

  - **Customer-hosted Mule runtimes**
    - On bare metal or cloud service providers: AWS, Azure, and Pivotal Cloud Foundry

MuleSoft-hosted runtime

Customer-hosted runtime

CloudHub benefits

- No hardware to maintain
- Continuous software updates
- Provided infrastructure for DNS and load-balancing
- Built-in elastic scalability for increasing cloud capacity during periods of high demand
- Globally available with data centers around the world
- Highly available with 99.99% uptime SLAs (service level agreements) http://status.mulesoft.com/
- Highly secure
  - PCI, HiTrust, and SSAE-16 certified

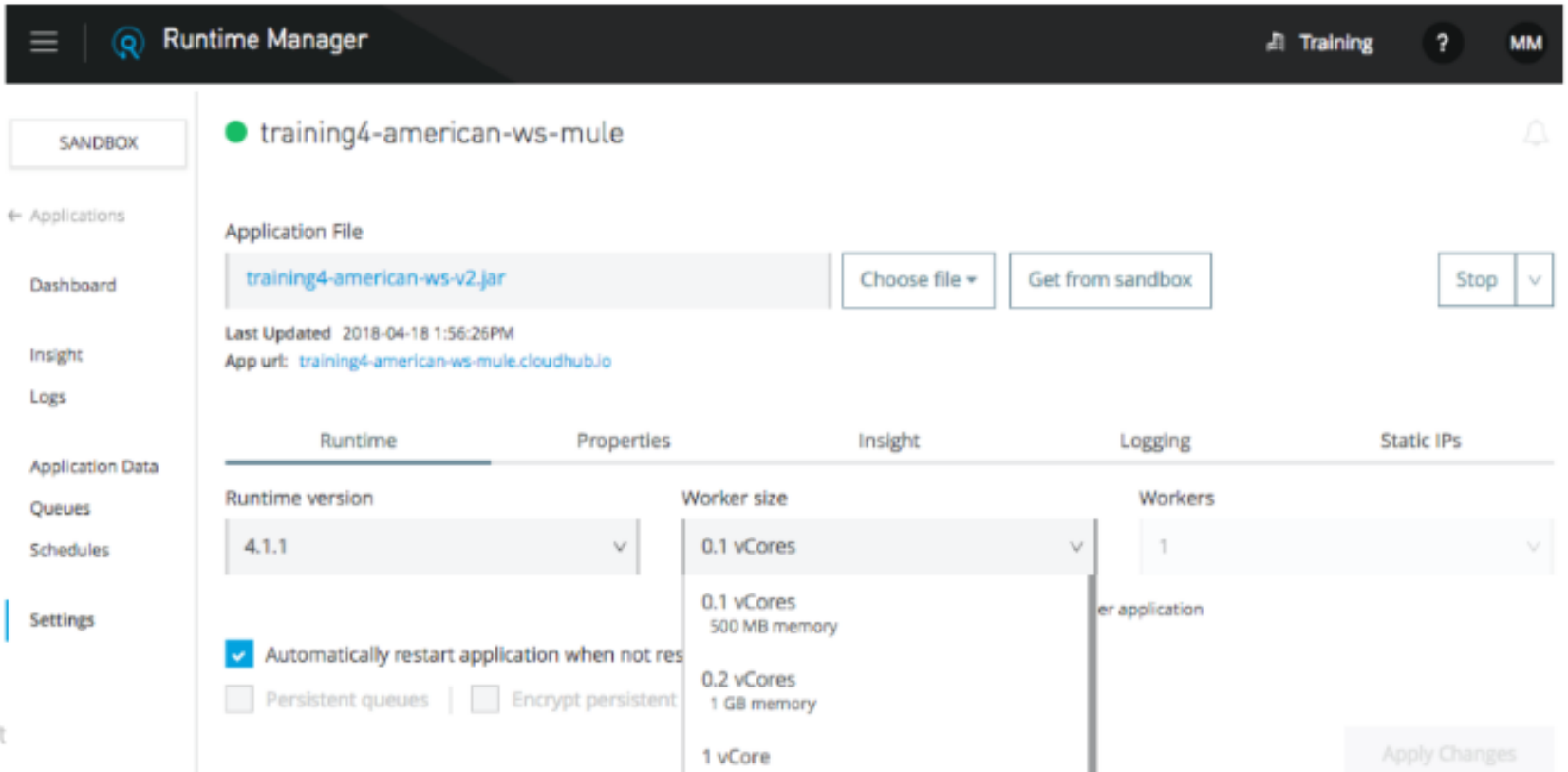
Customer-hosted Mule runtimes

- Easy to install
- Requires minimal resources
- Can run multiple applications
- Uses a Java Service Wrapper that controls the JVM from the operating system and starts Mule
- Can be managed by
  - Runtime Manager in MuleSoft-hosted Anypoint Platform
  - Runtime Manager in customer-hosted Anypoint Platform
    - Anypoint Platform Private Cloud Edition

Deploying applications to CloudHub

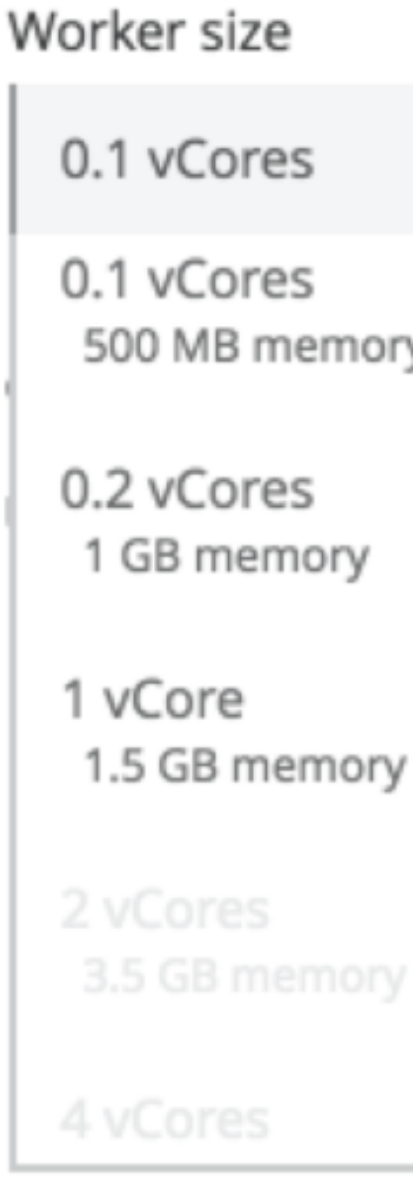
Deploying applications to CloudHub

- Can deploy from Anypoint Studio or from Anypoint Platform using Runtime Manager
- You must set worker size and number
  - For apps deployed from flow designer, these values were set automatically



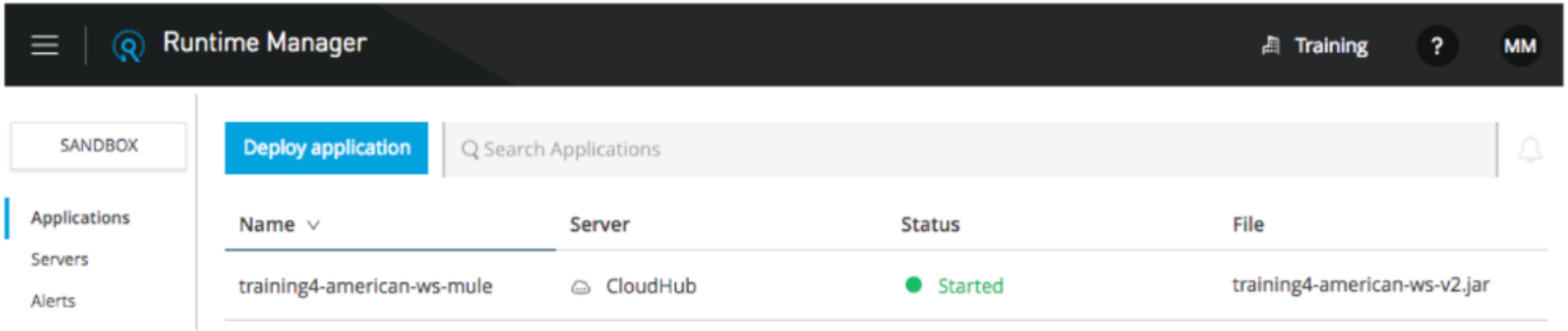
Review: CloudHub workers

- A worker is a dedicated instance of Mule that runs an app
- Each worker
  - Runs in a separate container from every other application
  - Is deployed and monitored independently
  - Runs in a specific worker cloud in a region of the world
- Workers can have a different memory capacity and processing power
  - Applications can be scaled vertically by changing the worker size
  - Applications can be scaled horizontally by adding multiple workers



Walkthrough 5-1: Deploy an application to CloudHub

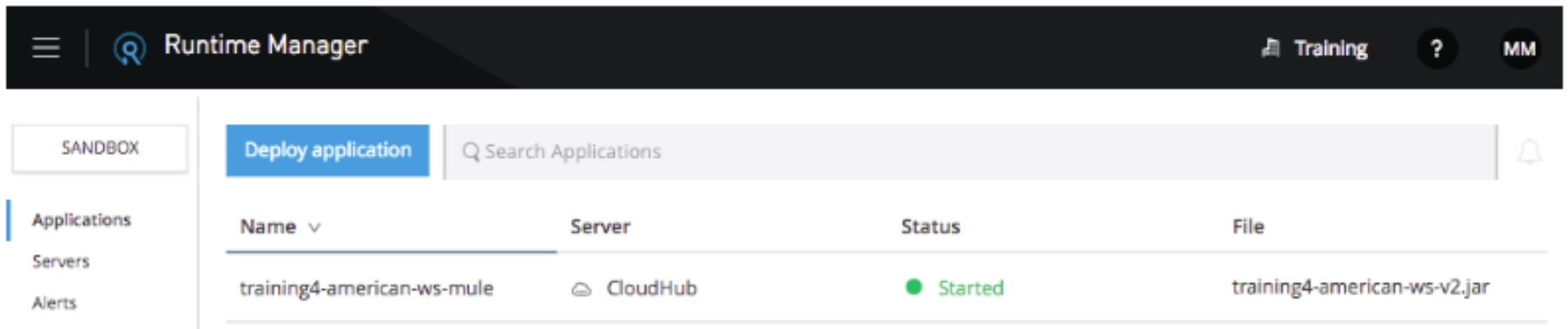
- Deploy an application from Anypoint Studio to CloudHub
- Run the application on its new, hosted domain
- Make calls to the web service
- Update an API implementation deployed to CloudHub



Walkthrough 5-1: 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 an API implementation deployed to CloudHub.

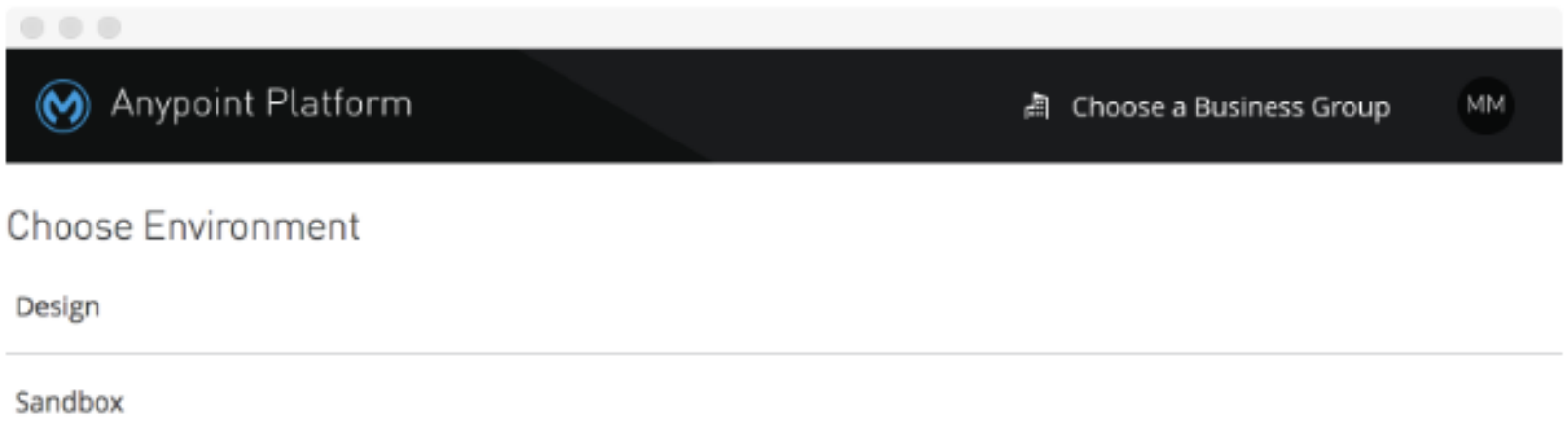


Note: If you do not have a working application at this point, import the training4-american-ws-wt4-6.jar solution into Anypoint Studio and work with that project.

Deploy the application to CloudHub

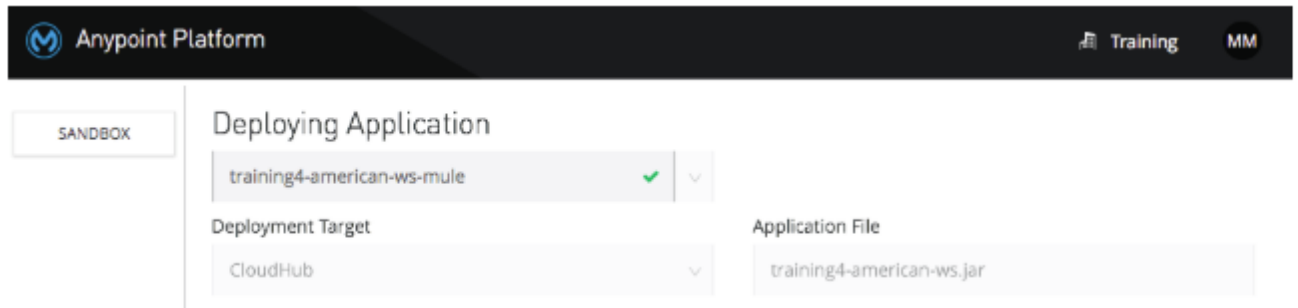
1. Return to Anypoint Studio.
2. In the Package Explorer, right-click the project and select Anypoint Platform > Deploy to CloudHub.

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



4. At the top of the Anypoint Platform dialog box, set the application name to training4-american-ws-*{your-lastname}* so it is a unique value.

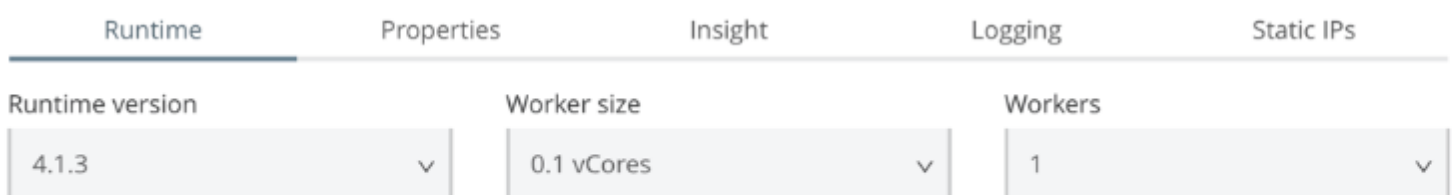
   Note: This name will be part of the URL used to access the application on CloudHub. It must be unique across all applications on CloudHub. The availability of the domain is instantly checked and you will get a green check mark if it is available.



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

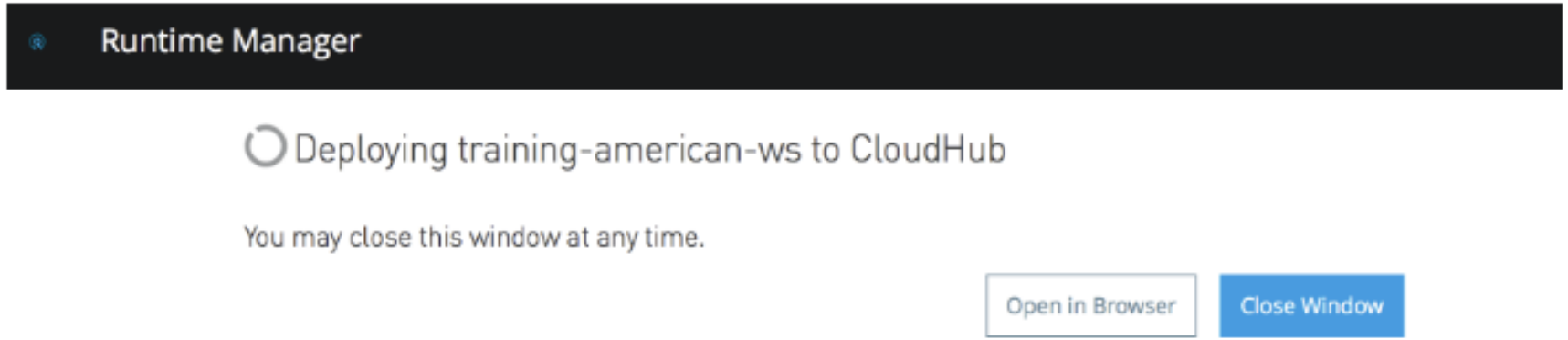
   Note: If you don't know what version it is using, look at the Package Explorer and find a library folder with the name of the server being used, like Mule Server 4.1.1 EE.

6. Make sure the worker size to 0.1 vCores.

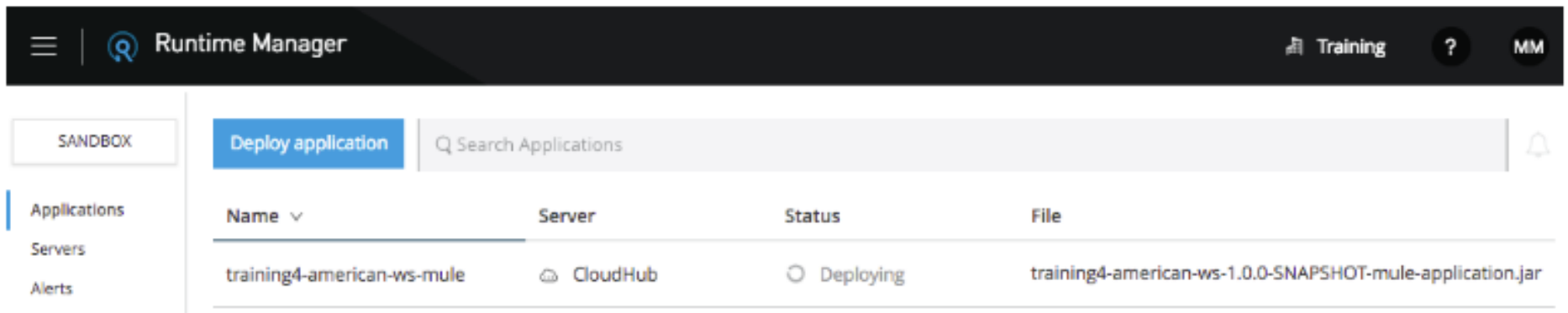


7. Click the Deploy Application button.

8. Click the Open in Browser button.

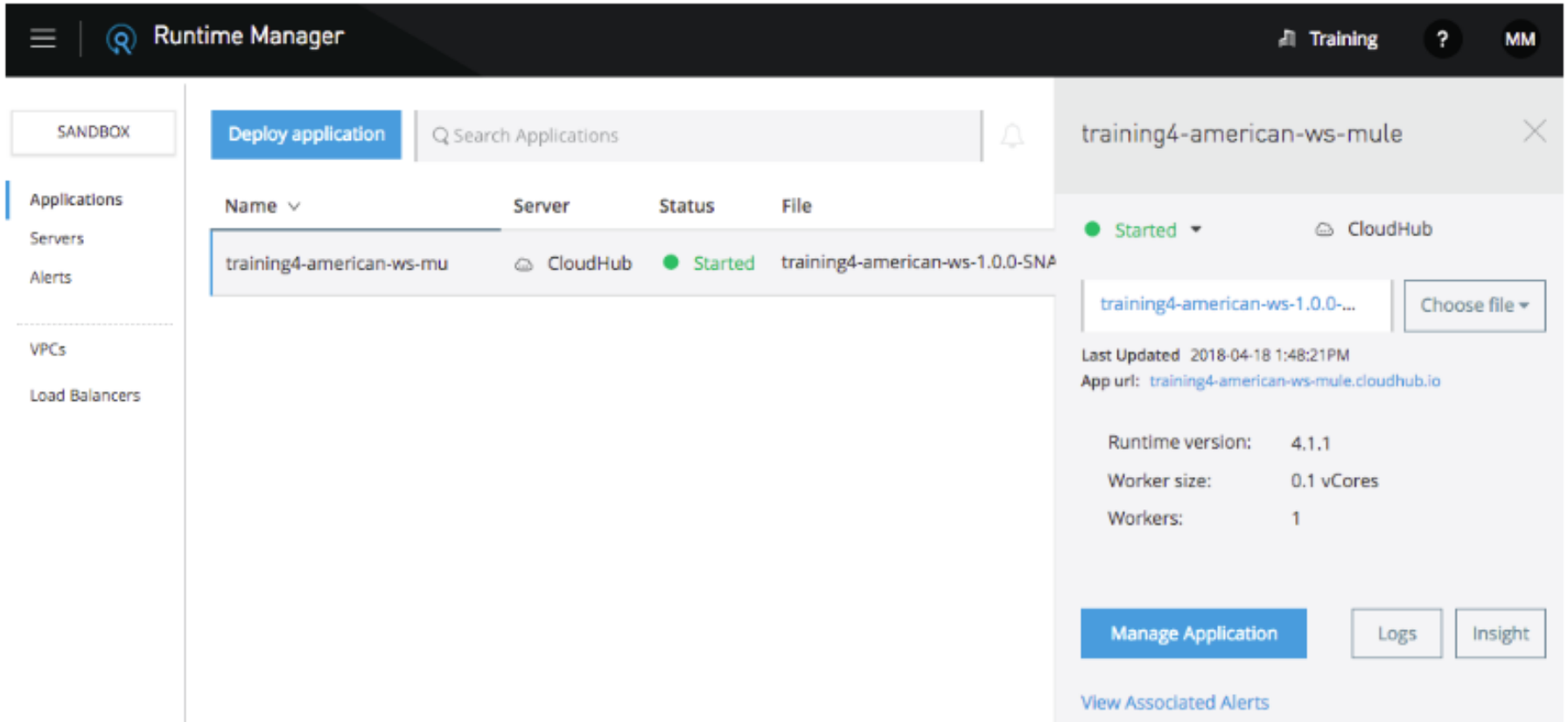


9. In the Anypoint Platform browser window that opens, locate the status of your deployment in the Runtime Manager.



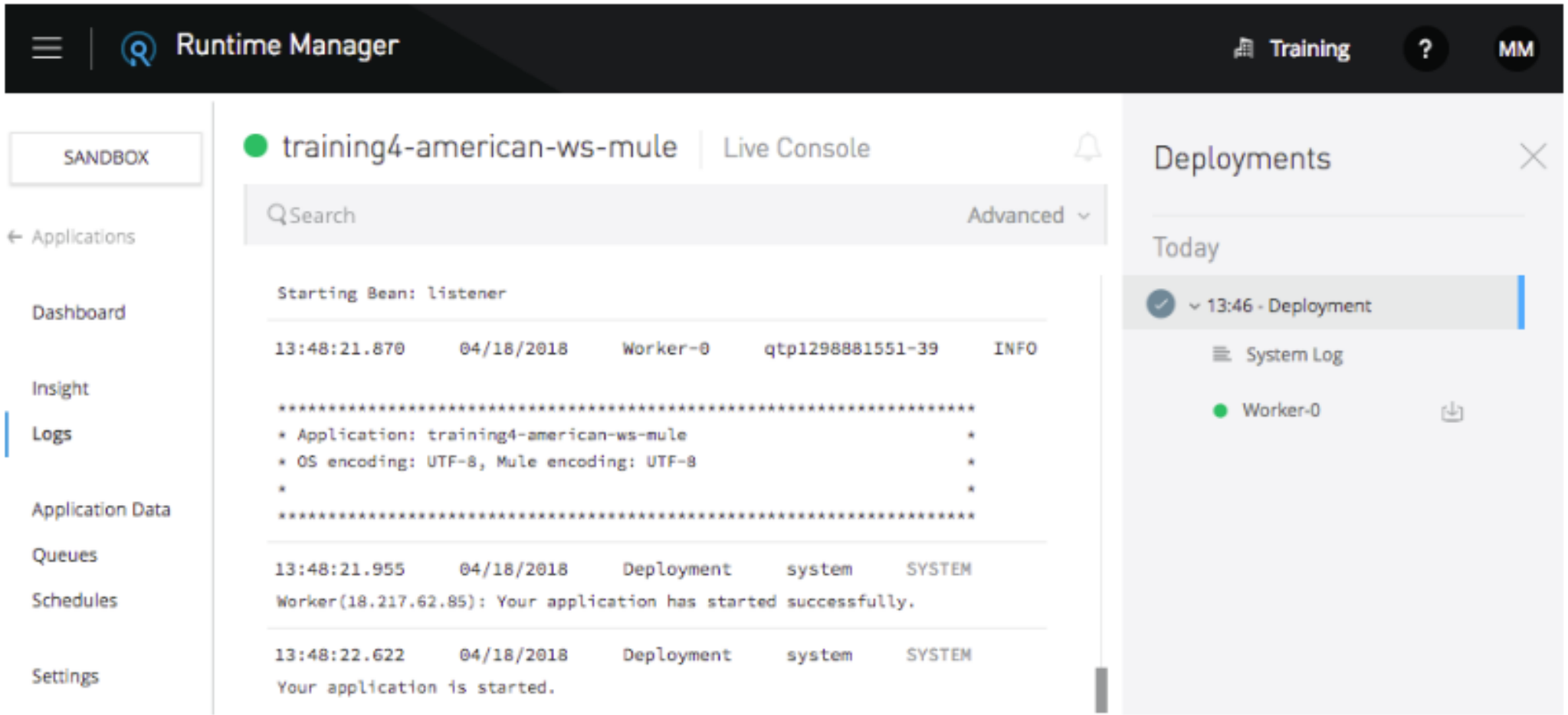
Watch the logs and wait for the application to start

10. Click 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.



11. Click the Logs button.
12. Watch the logs as the application is deployed.

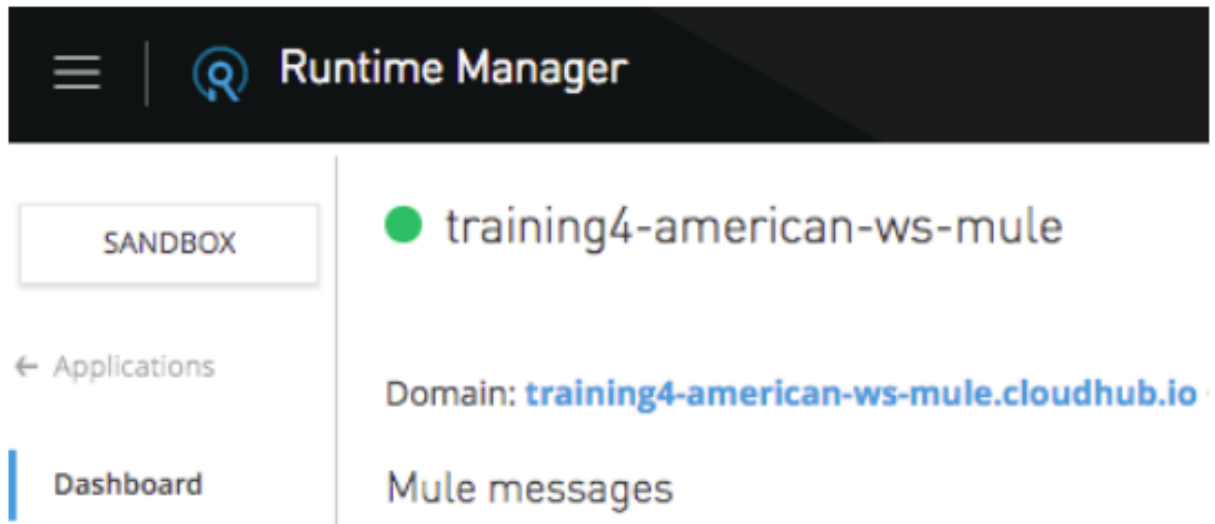
13. Wait until the application starts (or fails to start).



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

Test the application

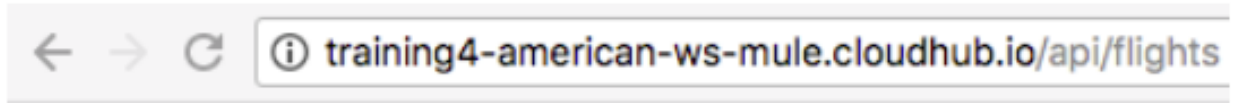
14. In the left-side navigation, select Dashboard.
15. Locate the link for the application on its new domain:
    training4-american-ws-*{lastname}.{region}*.cloudhub.io.



Note: {region} represents the worker region the Mule application is deployed. In North America, the default region is US East, and is denoted by us-e2 in the application URL.

16. 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.

17. Modify the path to http://training4-american-ws-{*lastname*}.{*region*}.cloudhub.io/api/flights; you should see the flights data.



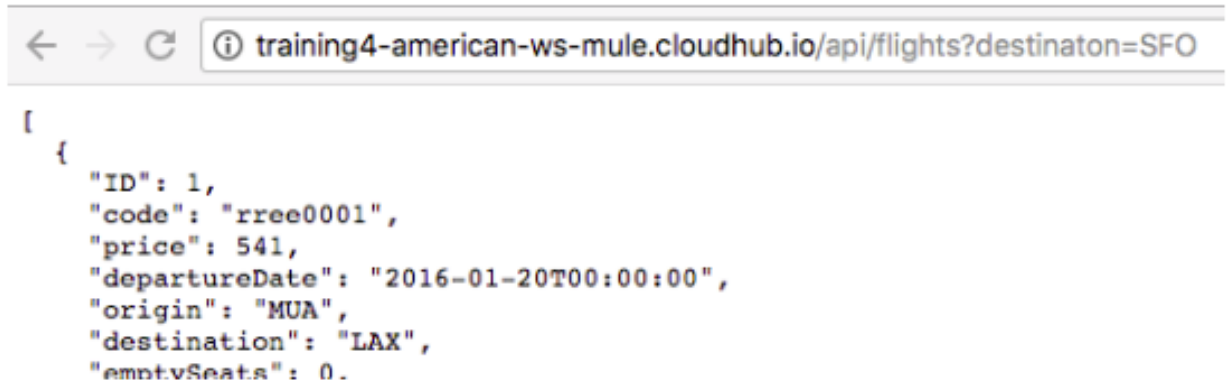

```
[
  {
    "ID": 1,
    "code": "rree0001",
    "price": 541,
    "departureDate": "2016-01-20T00:00:00",
    "origin": "MUA",
    "destination": "LAX",
    "emptySeats": 0,
    "plane": {
      "type": "Boeing 787",
      "totalSeats": 200
    }
  },
  {
    "ID": 2,
    "code": "eefd0123",
    "price":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

18. Add a query parameter called destination to the URL and set it equal to SFO.

19. 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.



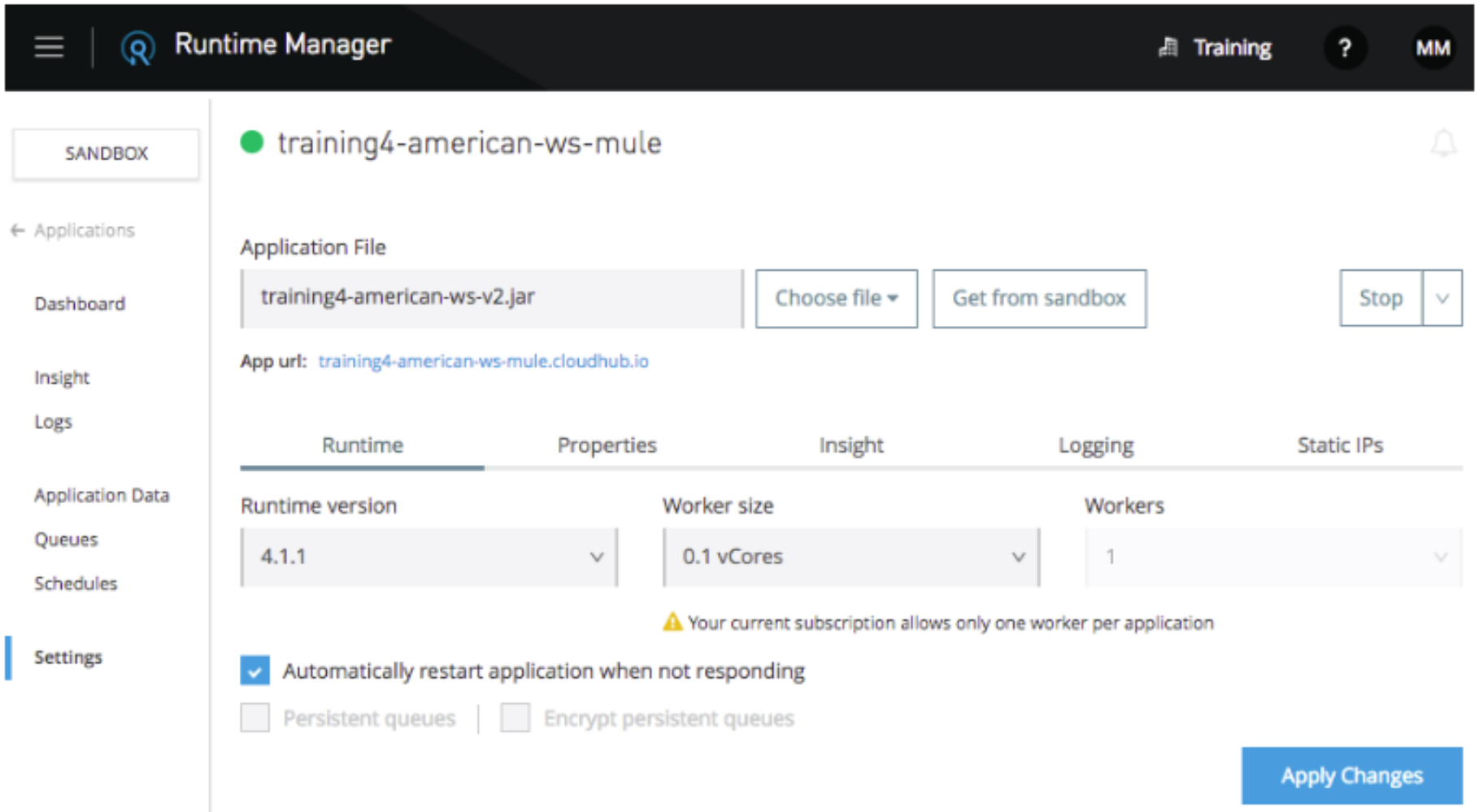
20. Leave this browser tab open.

Update the API implementation deployed to CloudHub

21. Return to the browser tab with Runtime Manager.
22. In the left-side navigation, select Settings.
23. Click the Choose file button and select Upload file.
24. Browse to the jars folder in the course student files.
25. Select training4-american-ws-v2.jar 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.

26. Click the Apply Changes button.



27. Wait until the application is uploaded and then redeploys successfully.

Test the updated application

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

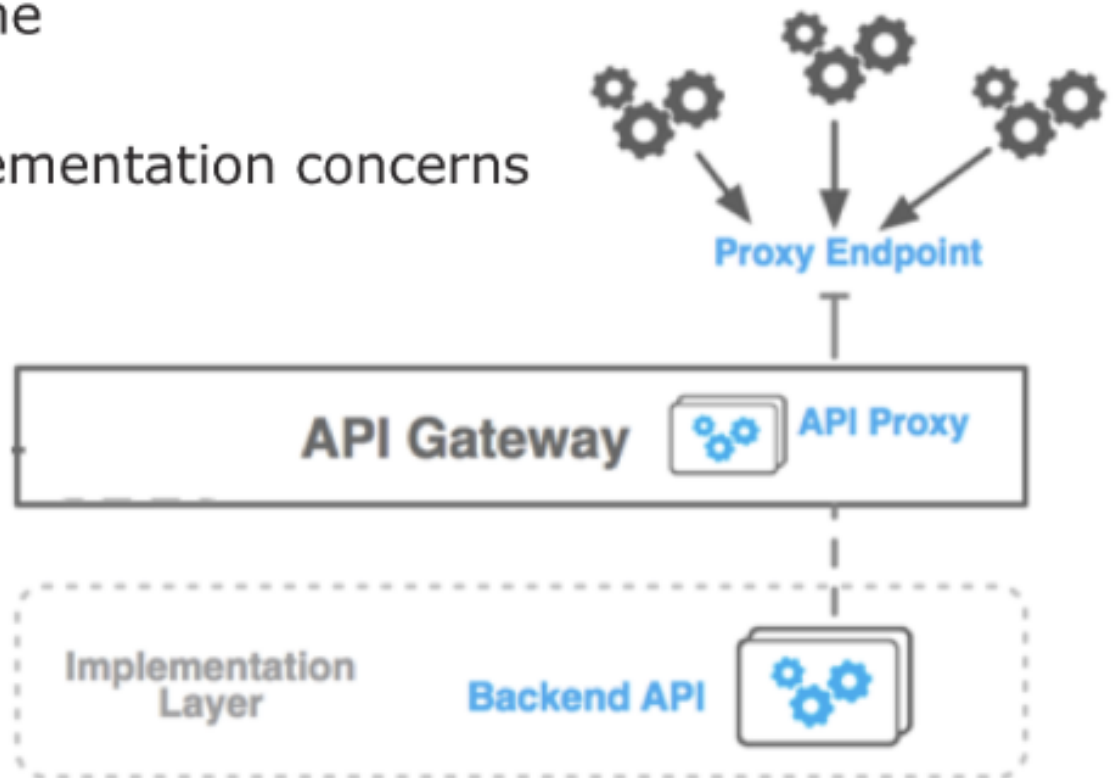

```
[
  {
    "ID": 5,
    "code": "rree1093",
    "price": 142,
    "departureDate": "2016-02-11T00:00:00",
    "origin": "MUA",
    "destination": "SFO",
    "emptySeats": 1,
    "plane": {
      "type": "Boeing 737",
      "totalSeats": 150
    }
  },
  {
    "ID": 7,
    "code": "eefd1994",
    "price":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


30. Close this browser tab.

Creating API proxies

Restricting access to APIs

- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway** is a runtime designed and optimized to host an API or to open a connection to an API deployed to another runtime
  - Included as part of the Mule runtime
    - Separate licenses required
  - Separates orchestration from implementation concerns



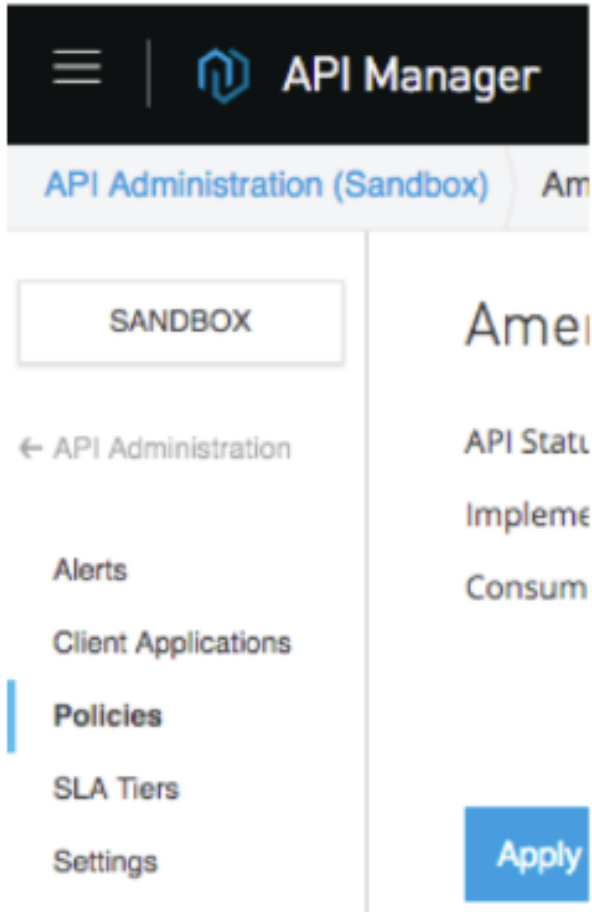
The API Gateway is the point of control

- **Determines which traffic** is authorized to pass through the API to backend services
- **Meters the traffic** flowing through
- **Logs** all transactions, collecting and tracking analytics data
- Applies runtime policies to **enforce governance** like rate limiting, throttling, and caching

Using API Manager to manage access to APIs

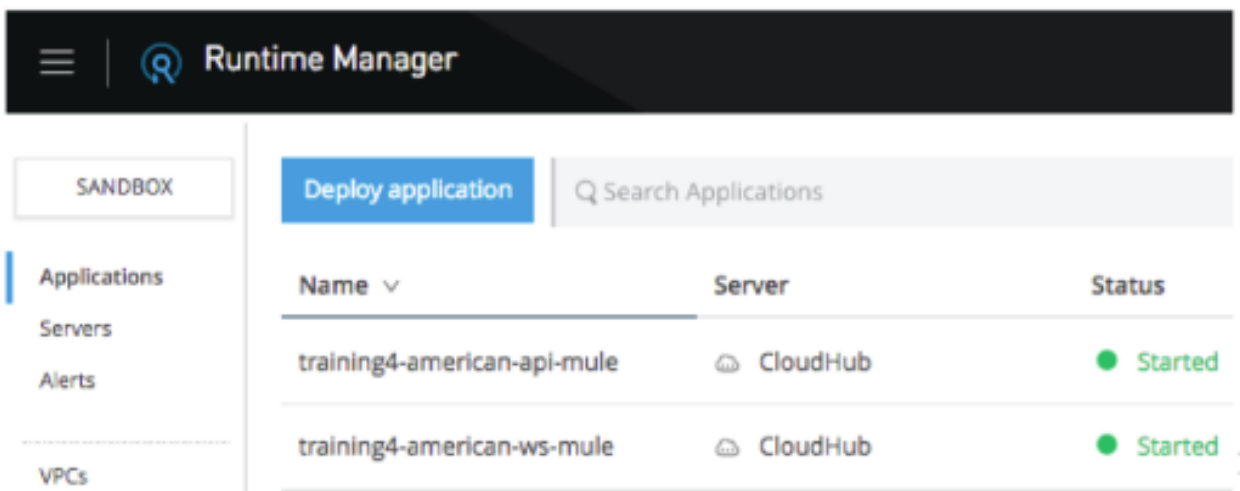
MuleSoft

- **Create** proxy applications
- **Deploy** proxies to an API Gateway runtime
  - On CloudHub or a customer-hosted runtime
- Specify throttling, security, and other **policies**
- Specify **tiers** with different types of access
- Approve, reject, or revoke **access** to APIs by clients
- **Promote** managed APIs between environments
- Review **analytics**



Walkthrough 5-2: Create and deploy an API proxy

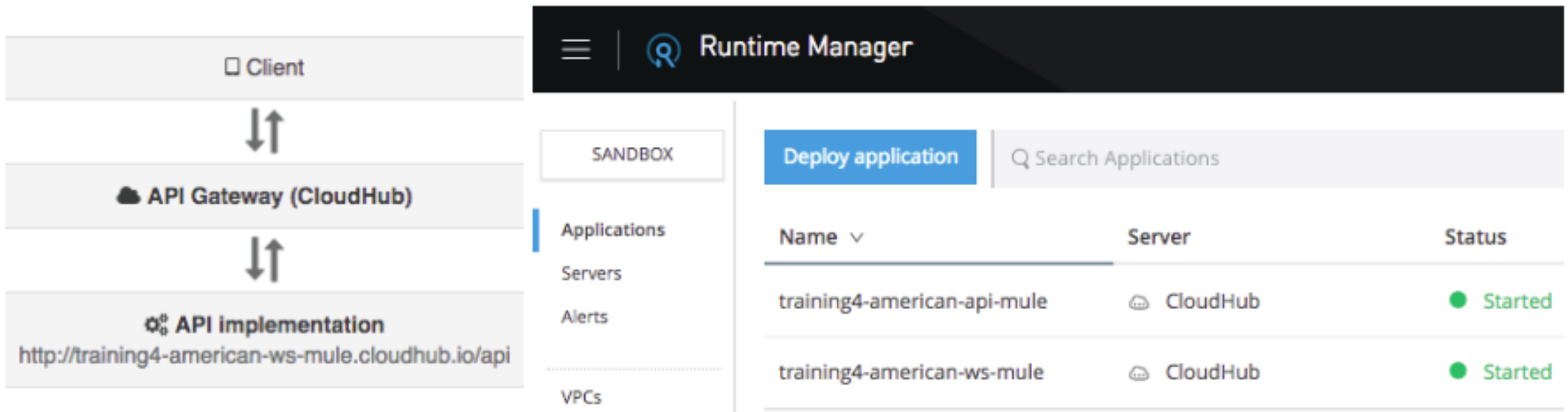
- Add an API to API Manager
- Use API Manager to create and deploy an API proxy application
- Set a proxy consumer endpoint so requests can be made to it from Exchange
- Make calls to an API proxy from API portals for internal & external users
- View API request data in API Manager.



Walkthrough 5-2: Create and deploy an API proxy

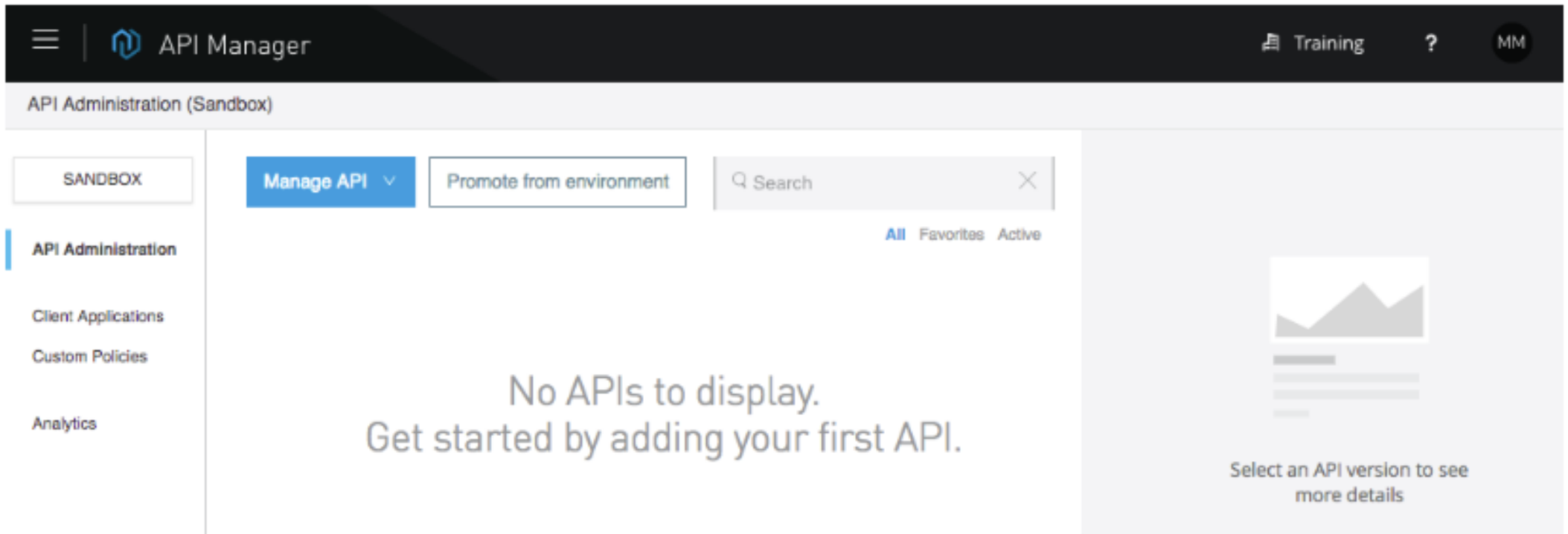
In this walkthrough, you create and deploy an API proxy for your API implementation on CloudHub. You will:

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

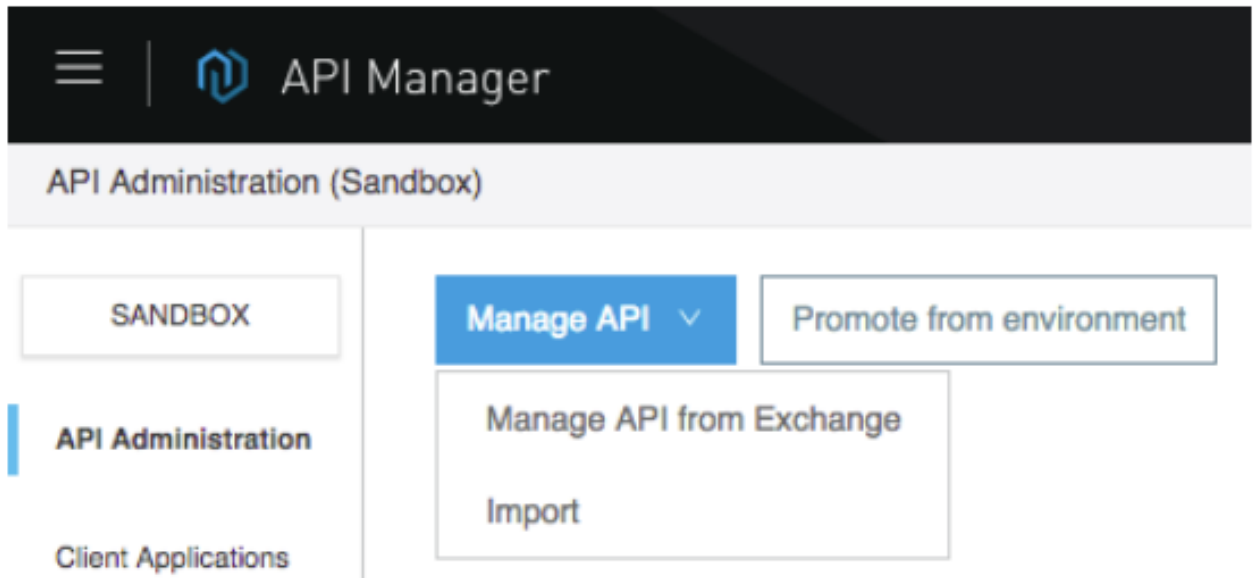


Create and deploy a proxy application

1. Return to Anypoint Platform.
2. In the main menu, select API Manager; you should see no APIs listed.



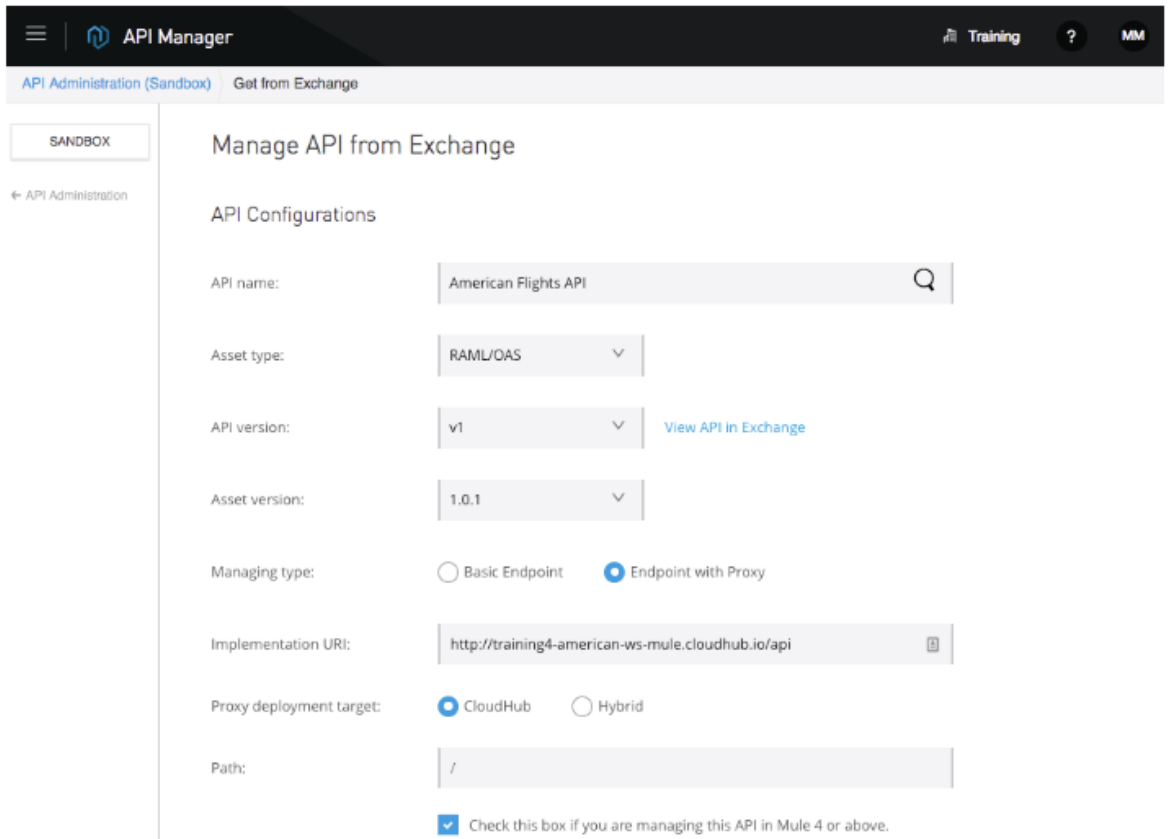
3. Click the Manage API button and select Manage API from Exchange.

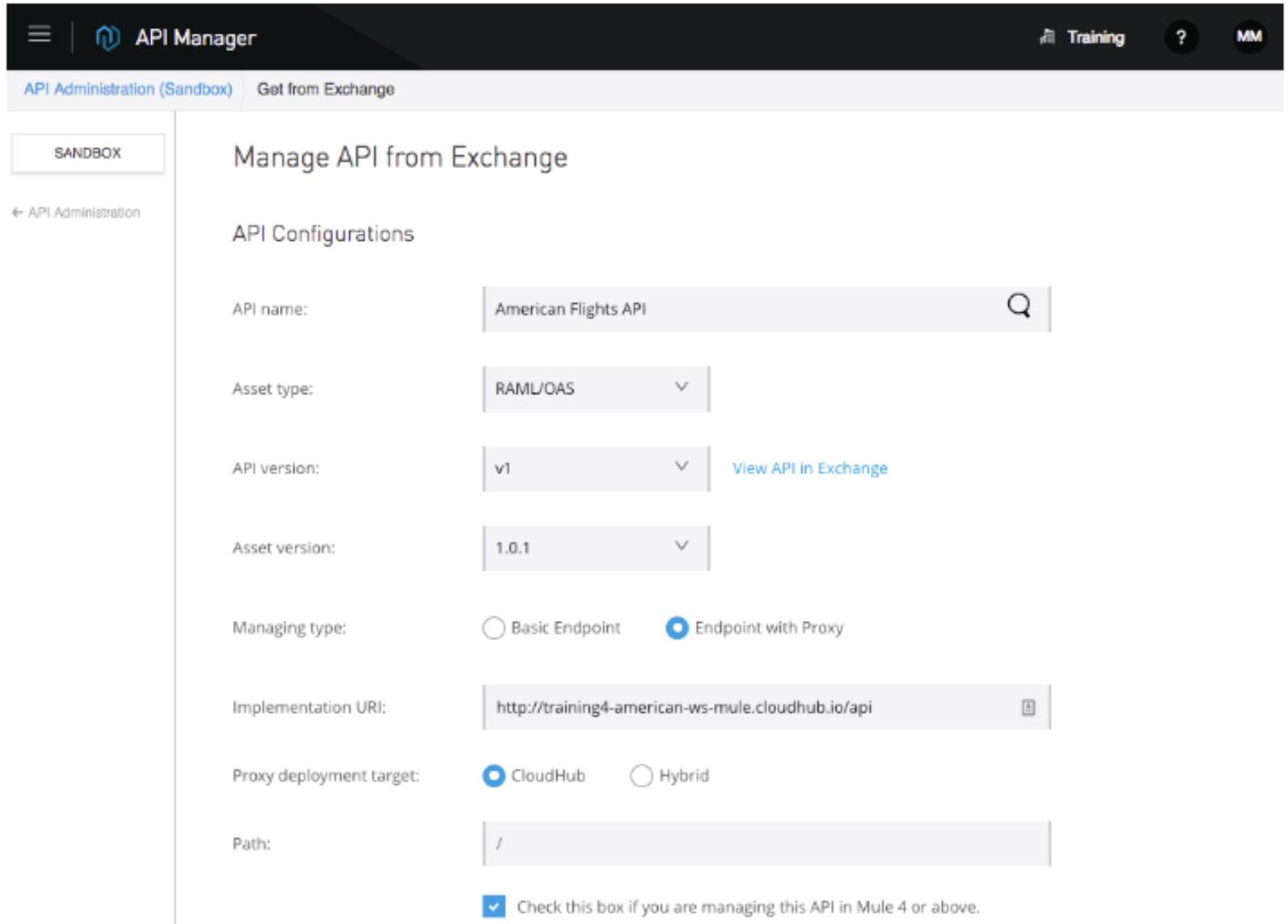


4. For API name, start typing American in the text field and then select your American Flights API in the drop-down menu that appears.

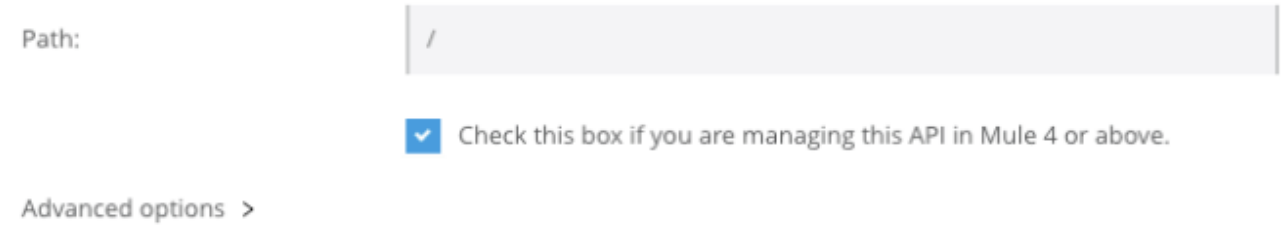
5. Set the rest of the fields to the following values:

   - Asset type: RAML/OAS
   - API version: v1
   - Asset version: 1.0.1
   - Managing type: Endpoint with Proxy
   - Implementation URI: http://training4-american-ws-{*lastname*}.cloudhub.io/api
   - Proxy deployment target: CloudHub

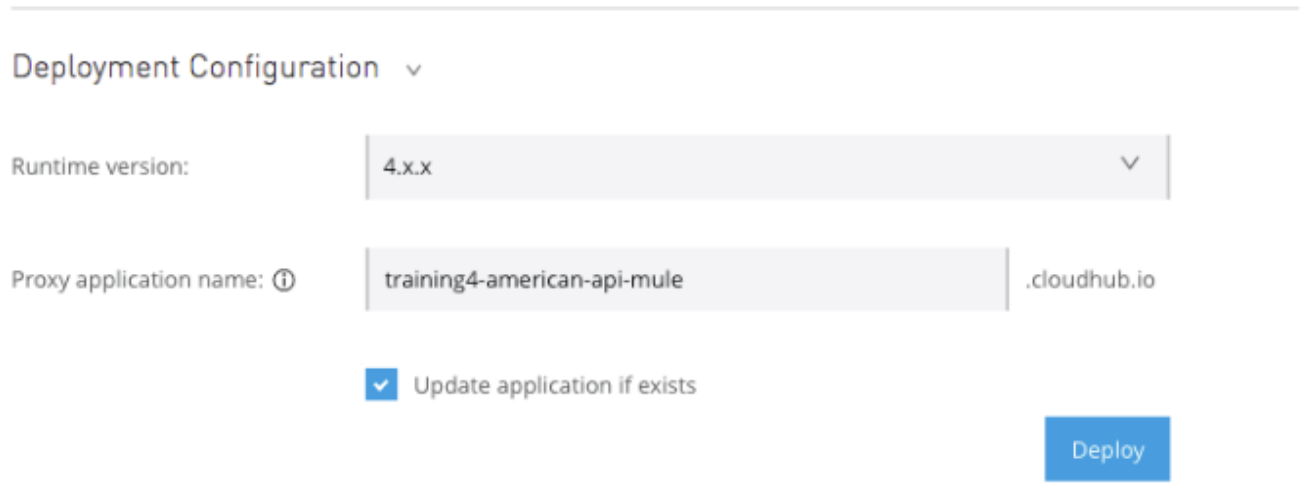



API Manager
Training
MM
API Administration (Sandbox)
Get from Exchange
SANDBOX
API Administration
Manage API from Exchange
API Configurations
API name:
American Flights API
Asset type:
RAML/OAS
API version:
v1
View API in Exchange
Asset version:
1.0.1
Managing type:
Basic Endpoint
Endpoint with Proxy
Implementation URI:
http://training4-american-ws-mule.cloudhub.io/api
Proxy deployment target:
CloudHub
Hybrid
Path:
/
Check this box if you are managing this API in Mule 4 or above.

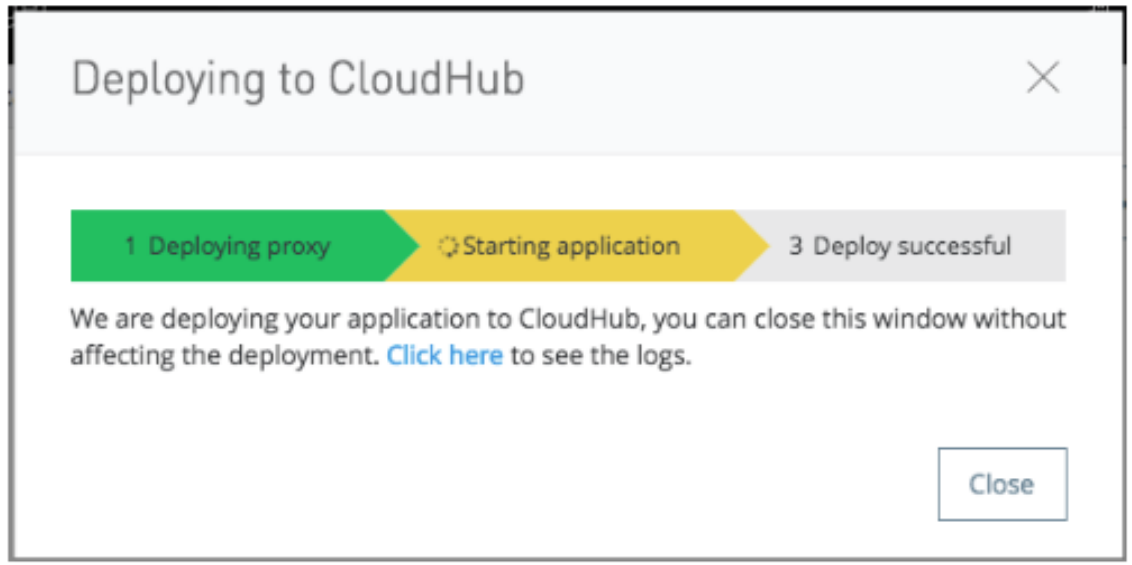
6. Select the checkbox: Check this box if you are managing this API in Mule 4 or above.



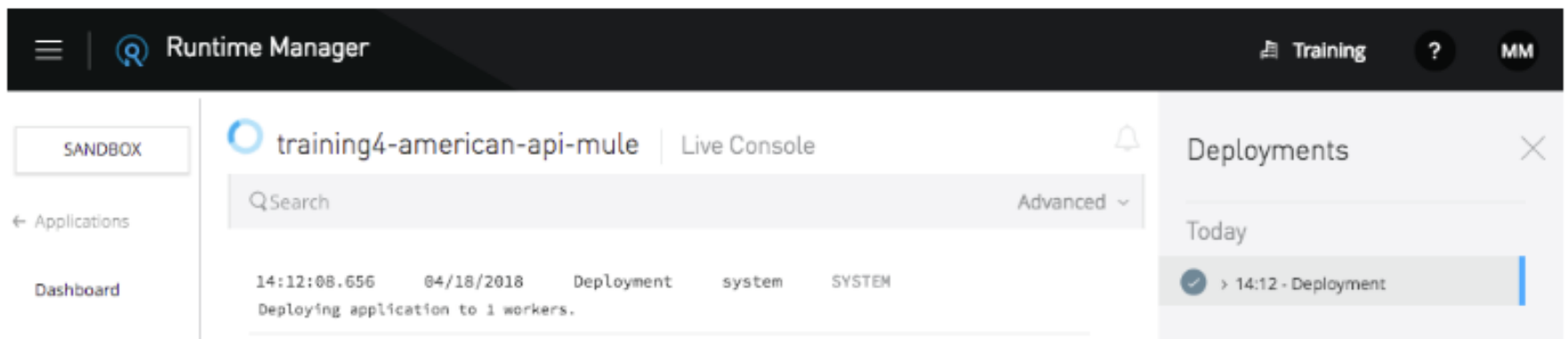
7. Click Save.
8. In the Deployment Configuration section, set the following values:
    - Runtime version: 4.x.x
    - Proxy application name: training4-american-api-{*lastname*}
9. Check Update application if exists.



10. Click Deploy.
11. In the Deploying to CloudHub dialog box, click the Click here link to see the logs.

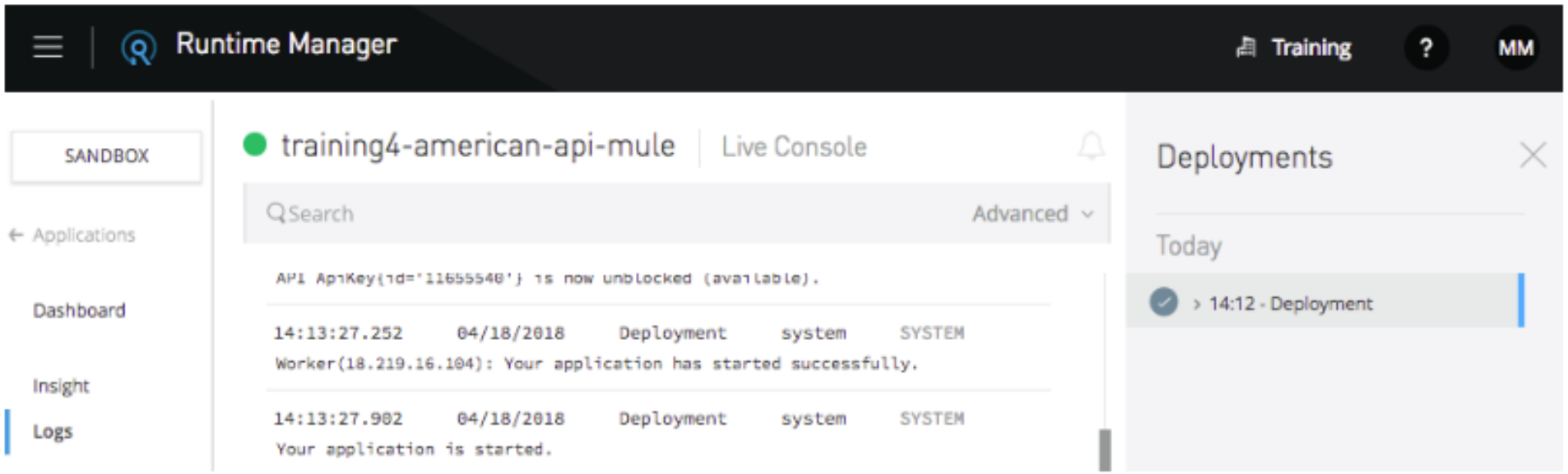


12. In the new browser tab that opens, watch the logs in Runtime Manager.



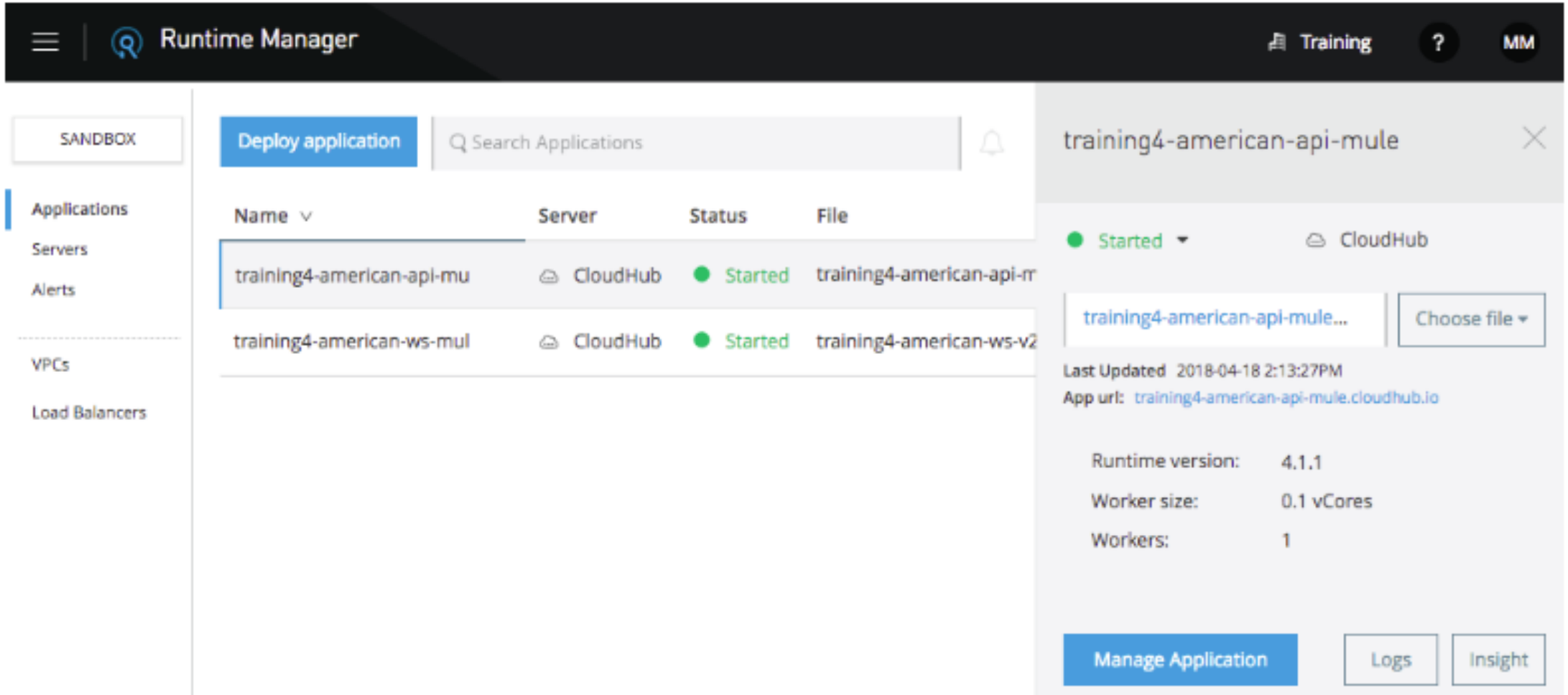
13. Wait until the proxy application starts.

    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.



14. In the left-side navigation, select Applications; you should see the proxy application.

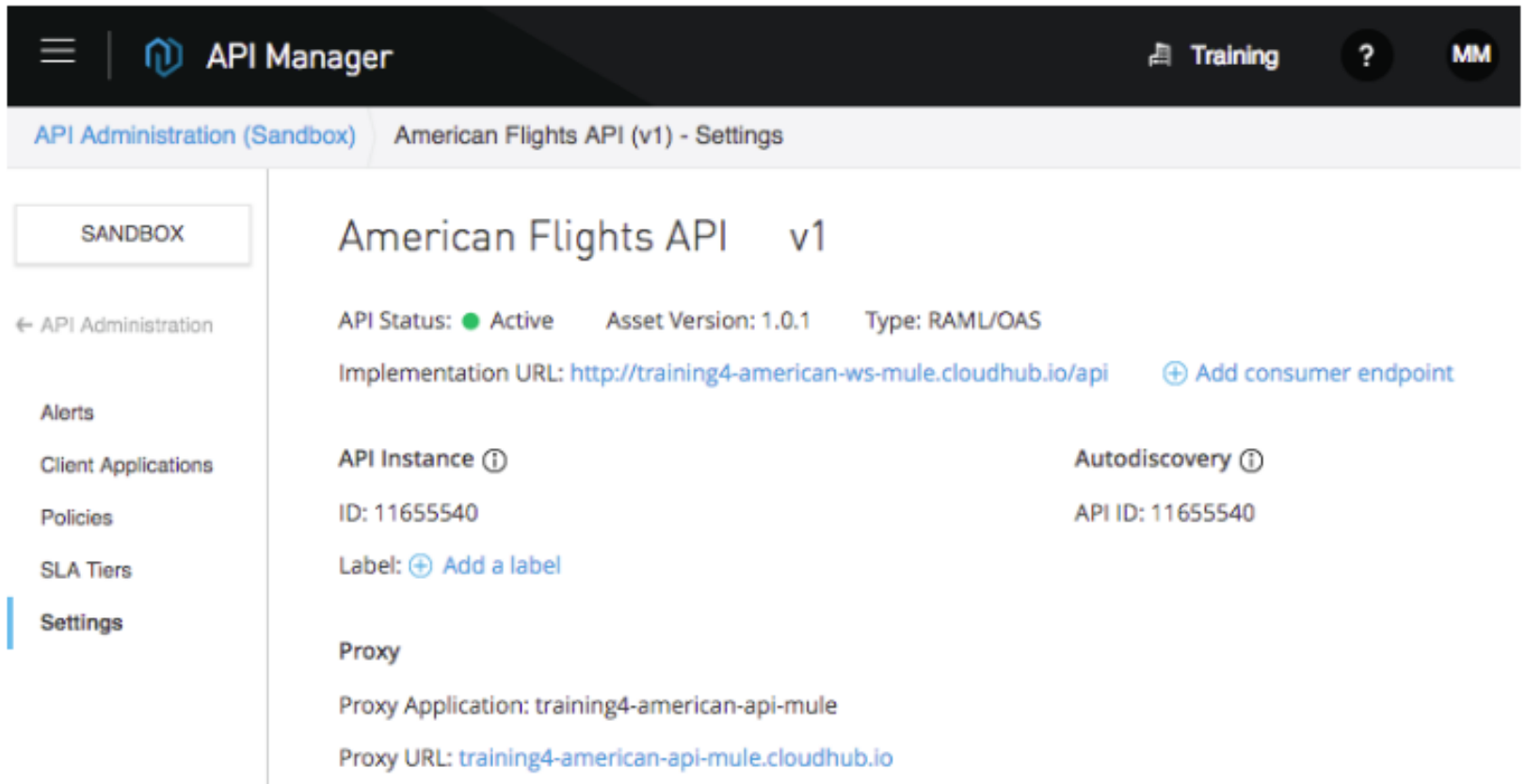
15. Click the row for the proxy and review its information in the right section of the window.



16. Close the browser tab.

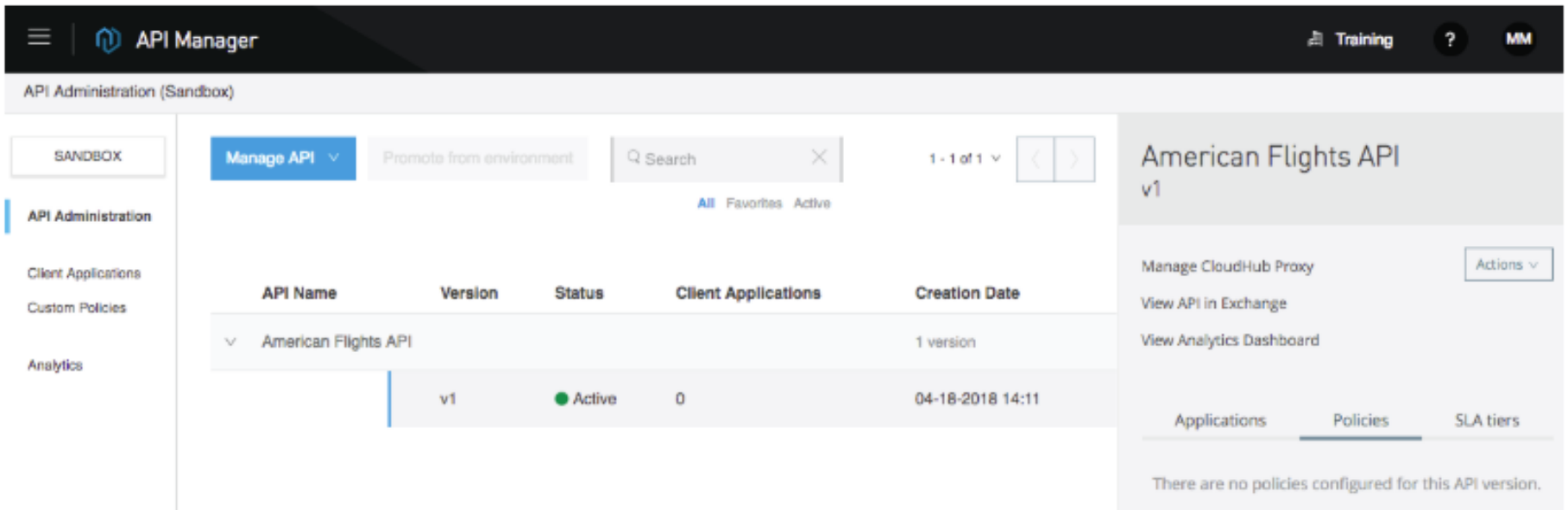
View API details in API Manager

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



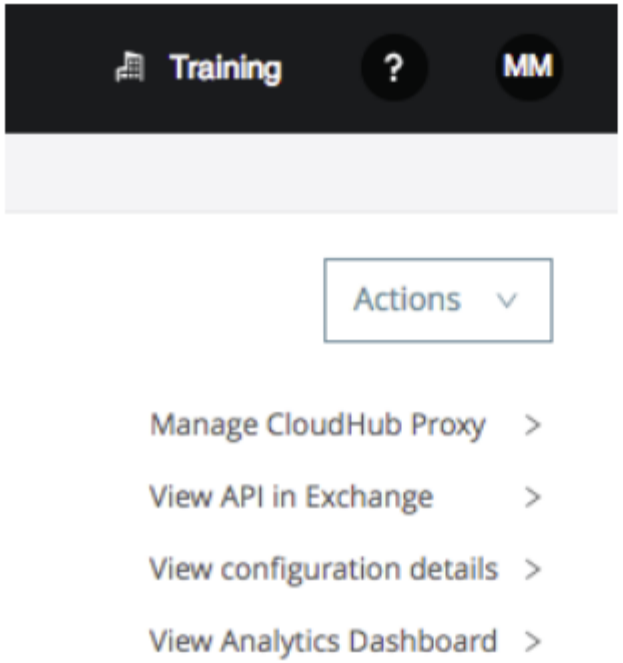
19. In the left-side navigation, click the API Administration link; you should now see your American Flights API listed.
20. Click in the row for the v1 version – but not on the v1 link.

21. 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.

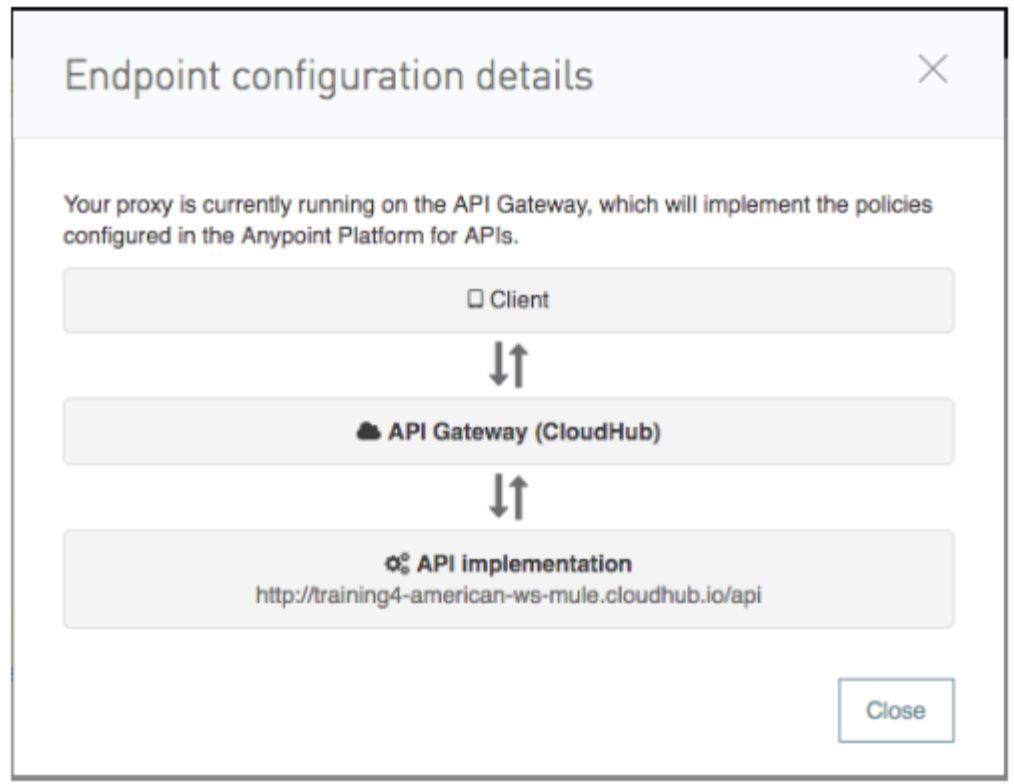


22. In the API list, click the v1 link for the API; you should be returned to the Settings page for the API.

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

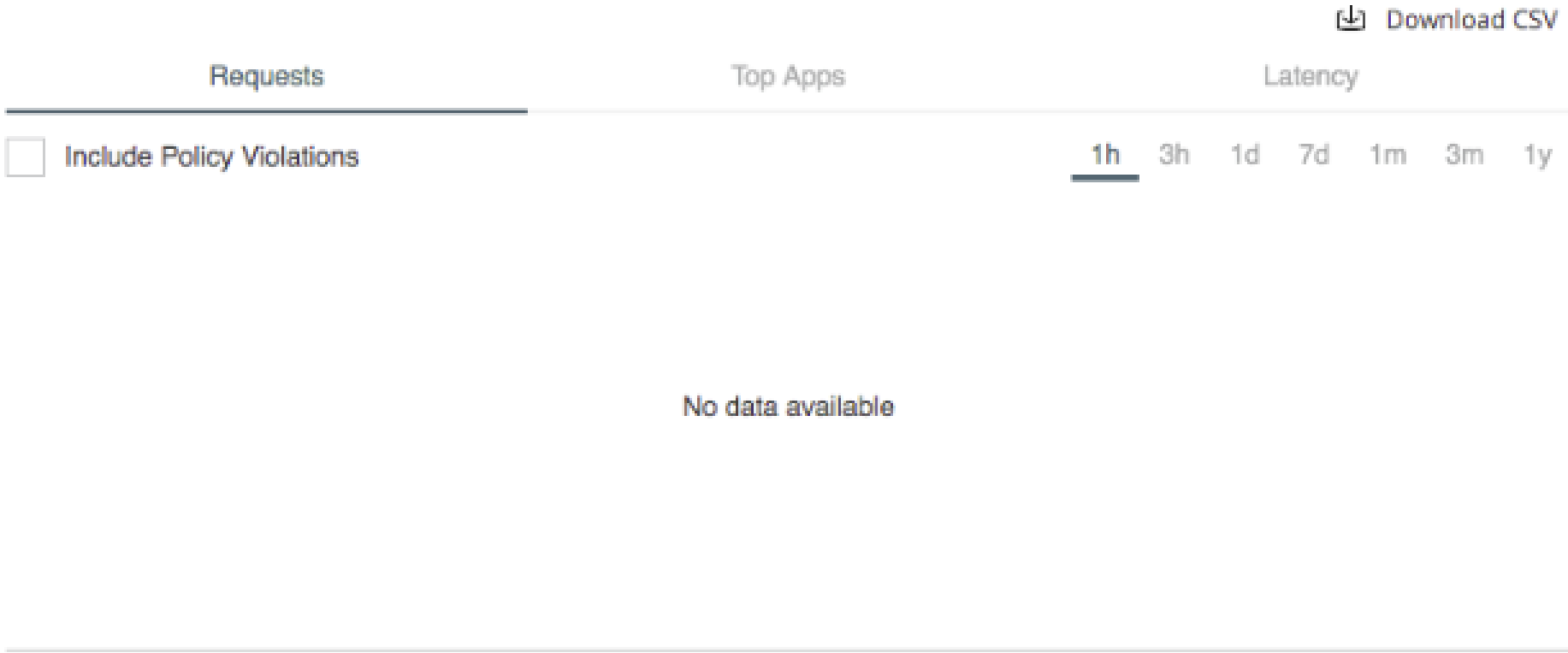


24. Click the View configuration details link.



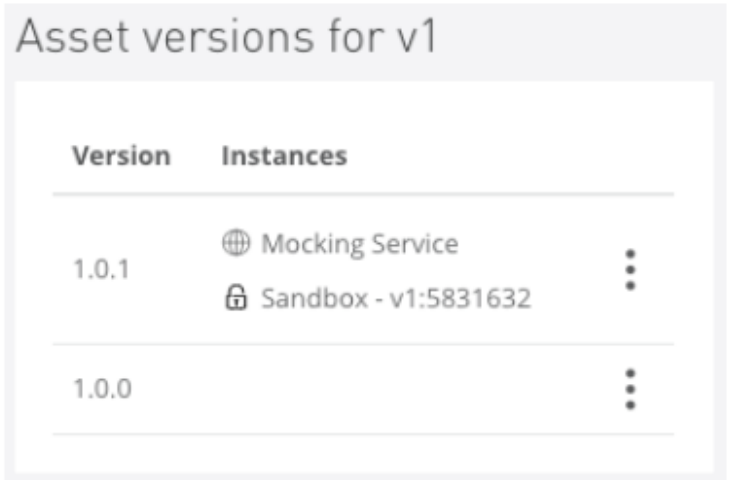
25. In the Endpoint configuration details dialog box, click Close.

26. 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

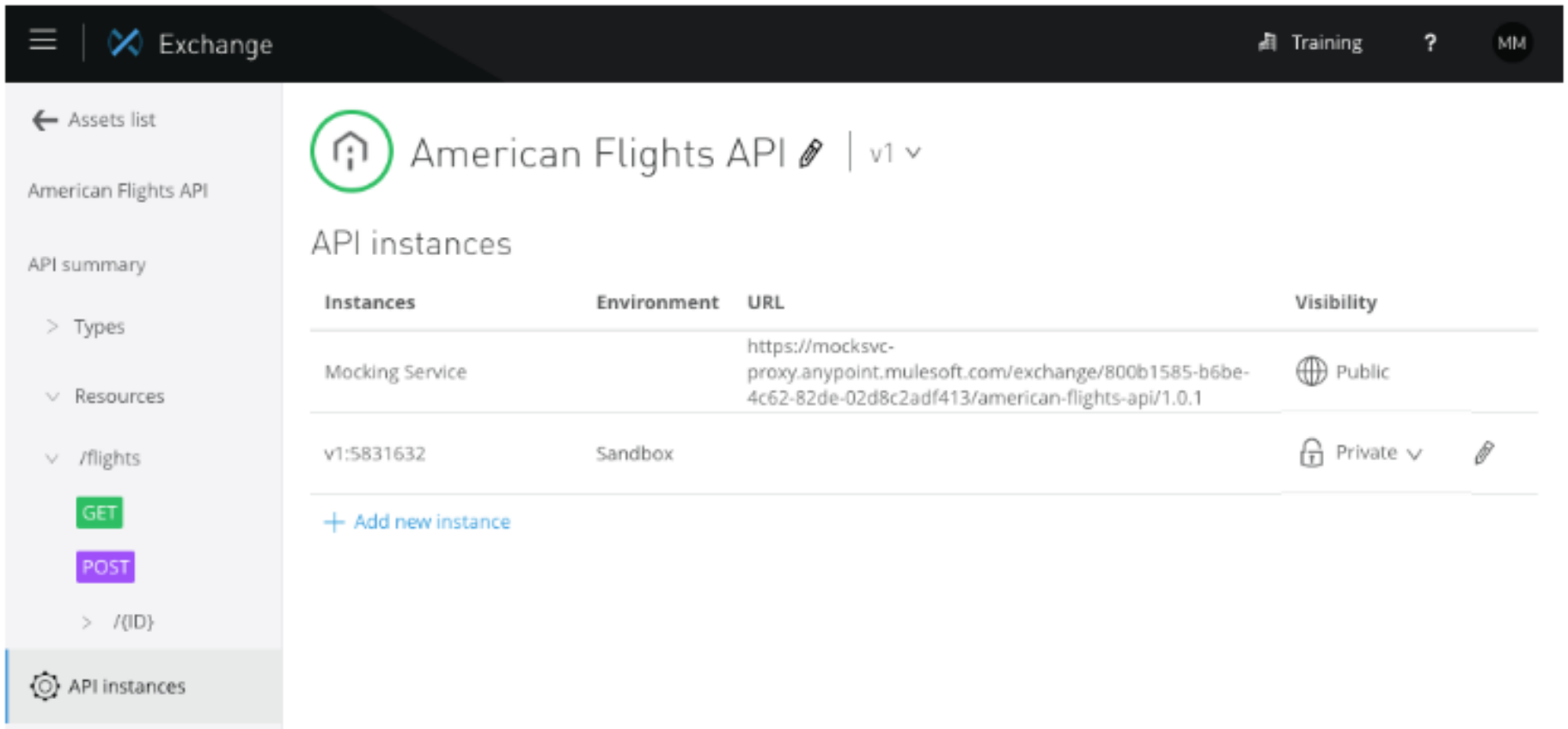
27. Return to the browser tab with Exchange.
28. Return to the home page for your American Flights API.
29. Locate the API instances now associated with asset version 1.0.1; you should see the Mocking Service instance and now the new proxy.



Note: You may need to refresh your browser page.

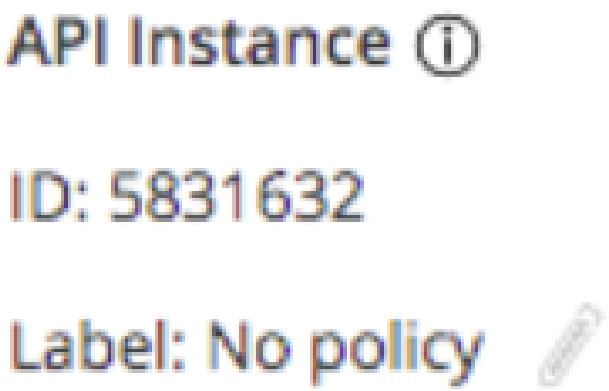
30. Click the GET method for the flights resource.
31. In the API console, click the drop-down arrow next to Mocking Service; you should NOT see the API proxy as a choice.

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



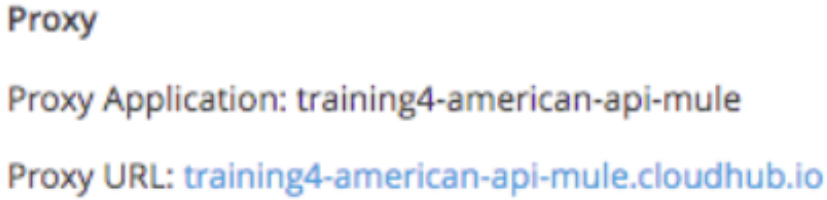
Set a friendly label for the API instance in API Manager

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



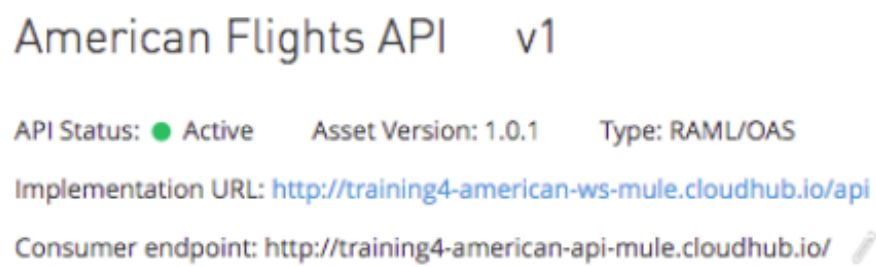
Set a consumer endpoint for the proxy in API Manager

36. Locate the proxy URL.



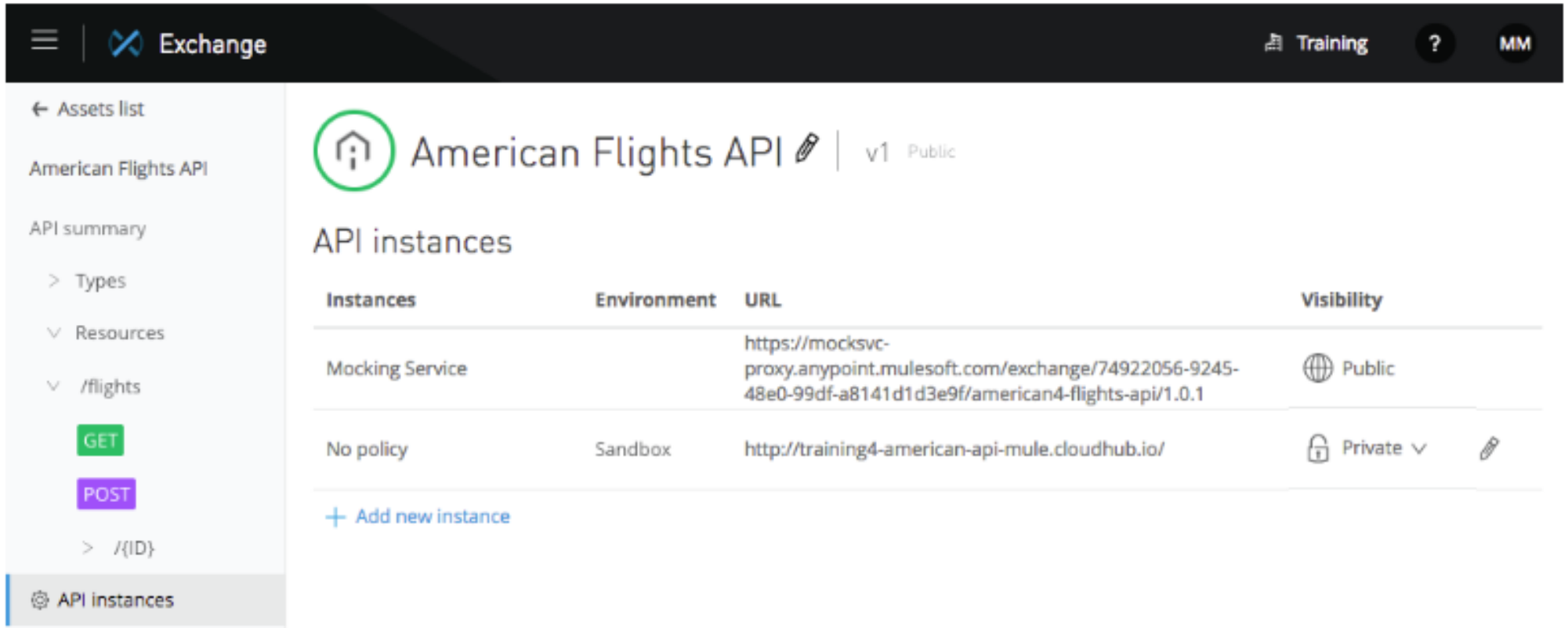
37. Right-click it and copy the link address.
38. Click the Add consumer endpoint link.

39. Paste the value of the proxy URL.
40. Press Enter/Return.



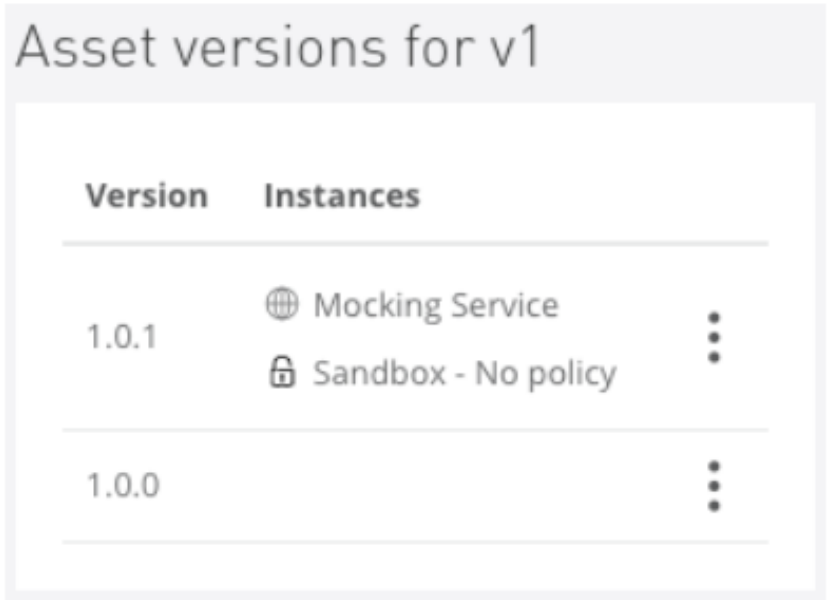
Make requests to the API proxy from Exchange

41. Return to the browser tab with Exchange.
42. Refresh the API instances page for your American Flights API; you should see the new label and the URL.

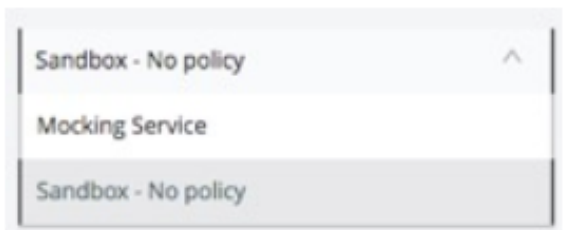


43. In the left-side navigation, select the name of the API to return to its main page.

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

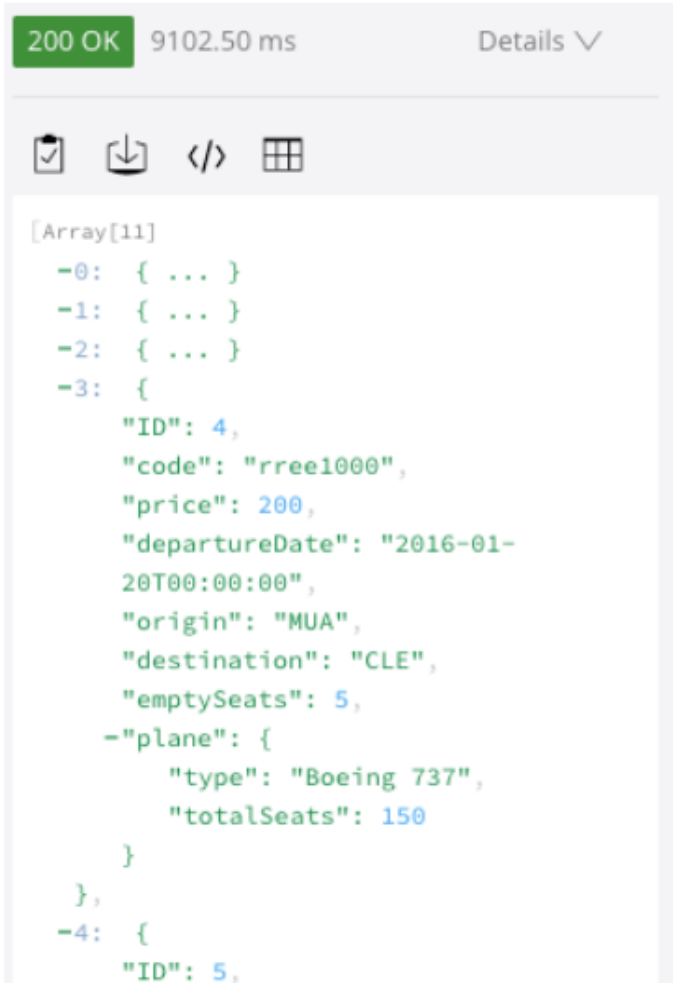


45. Click the GET method for the flights resource.
46. In the API console, click the drop-down arrow next to Mocking Service; you should now see your API proxy instance as a choice.



47. Select the Sandbox - No policy instance.

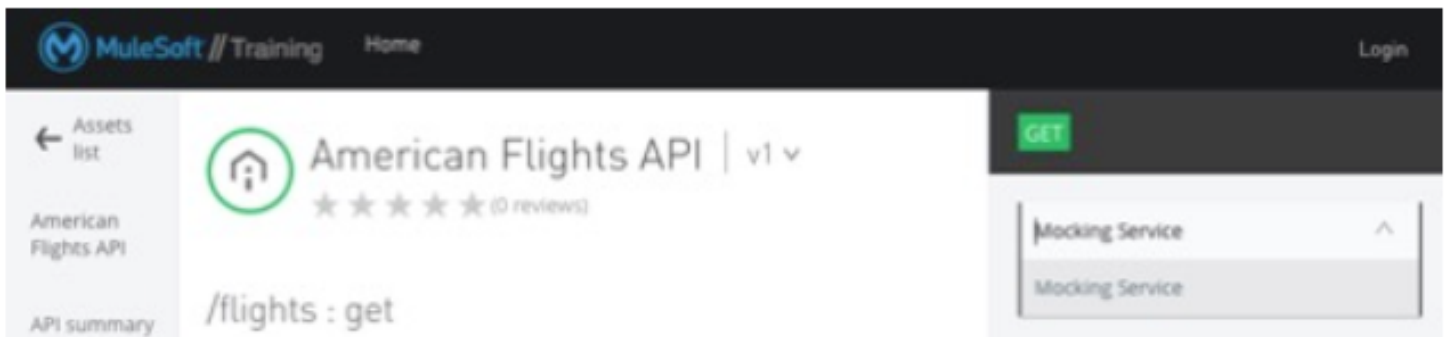
48. Click the Send button; you should now see the real data from the database, which contains multiple flights.



49. Make several more calls to this endpoint.
50. Make calls to different methods.

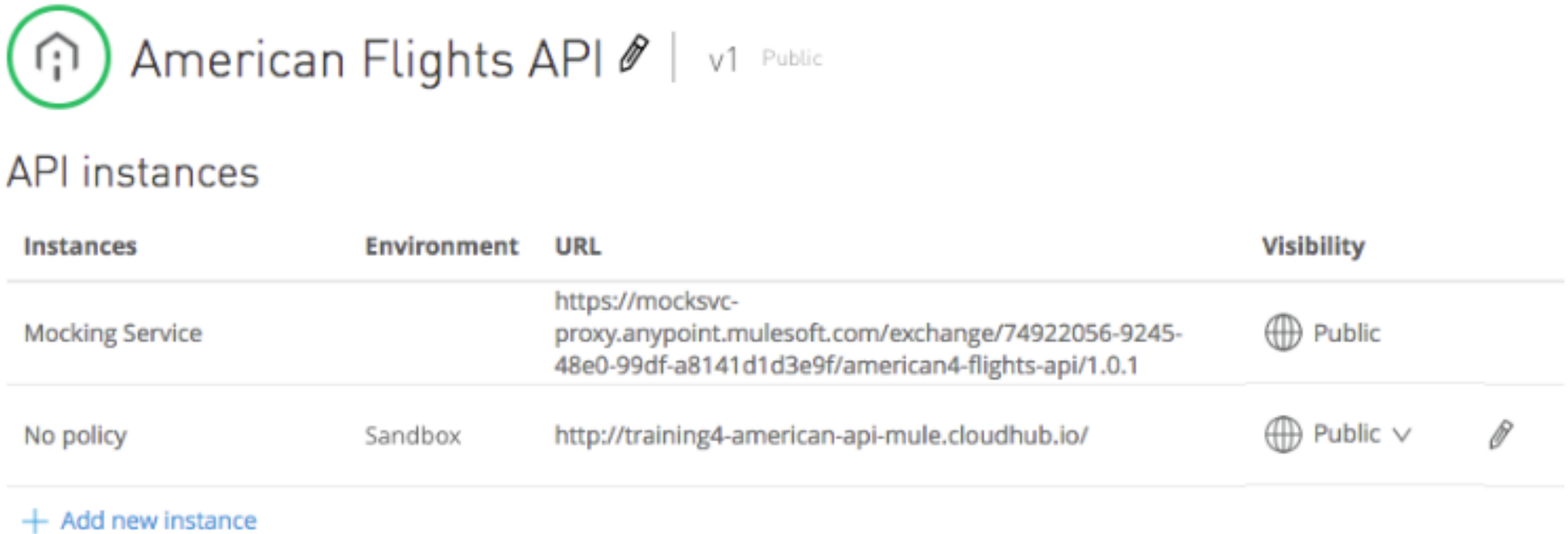
Make requests to the API proxy from the public portal

51. Return to the public portal in the private/incognito window.
52. Click the GET method for the flights resource.
53. 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.



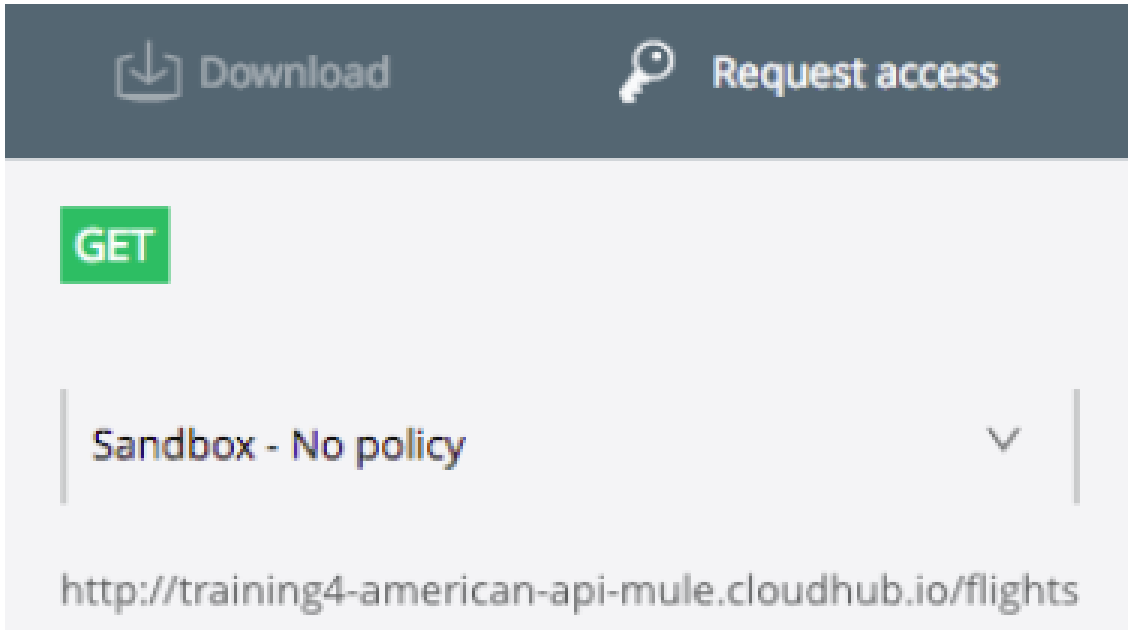
Make an API instance visible in the public portal

54. Return to the browser with Exchange.
55. In the left-side navigation, select API instances.
56. Change the visibility of the No policy instance from private to public.



57. Return to the public portal in the private/incognito window.
58. Refresh the page.

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



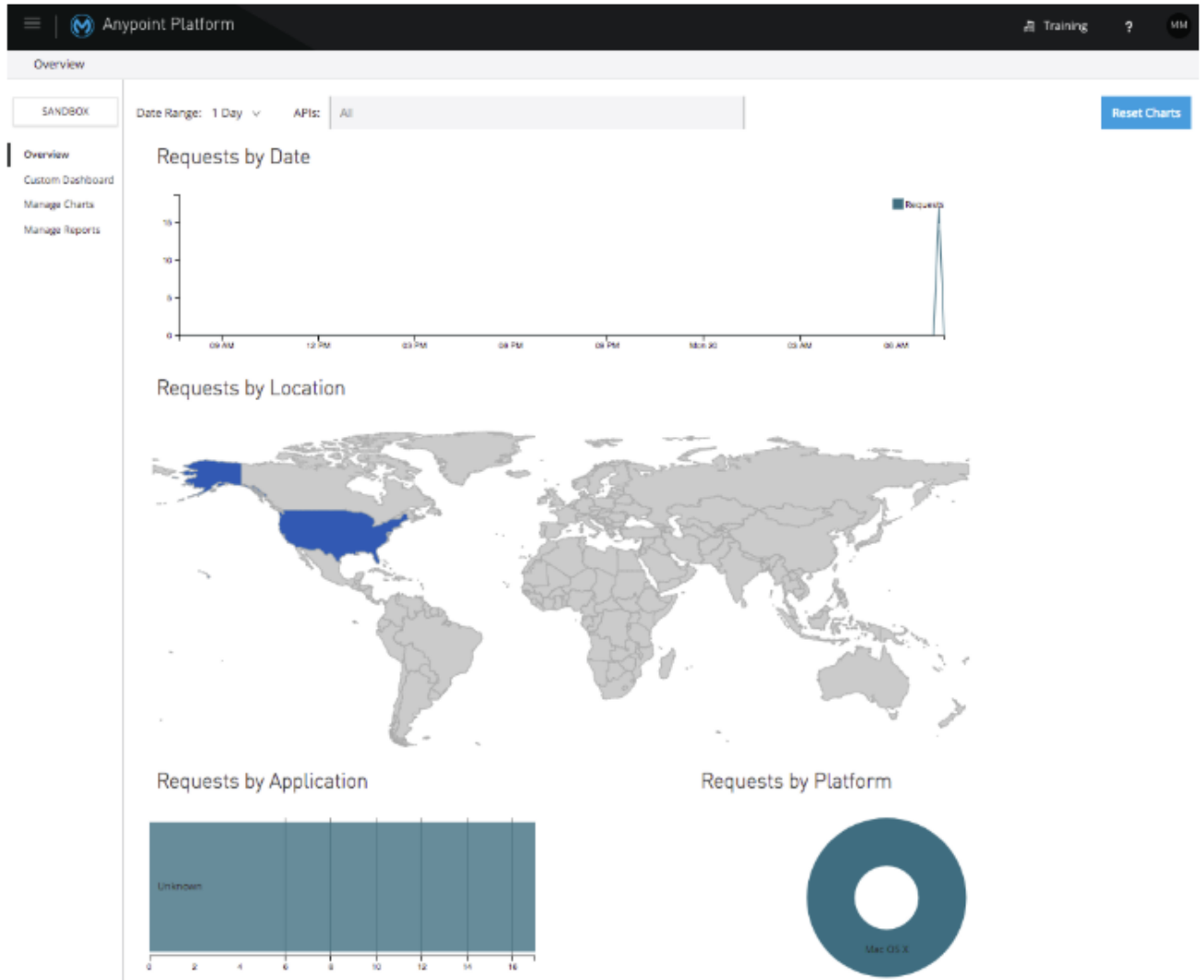
60. Click Send; you should a 200 response and flight data.

Look at the API request data

61. Return to the browser tab with API Manager.
62. Refresh the Settings page for you American Flights API.
63. Look at the Request chart again; you should now see data for some API calls.

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

65. Review the data in the dashboard.

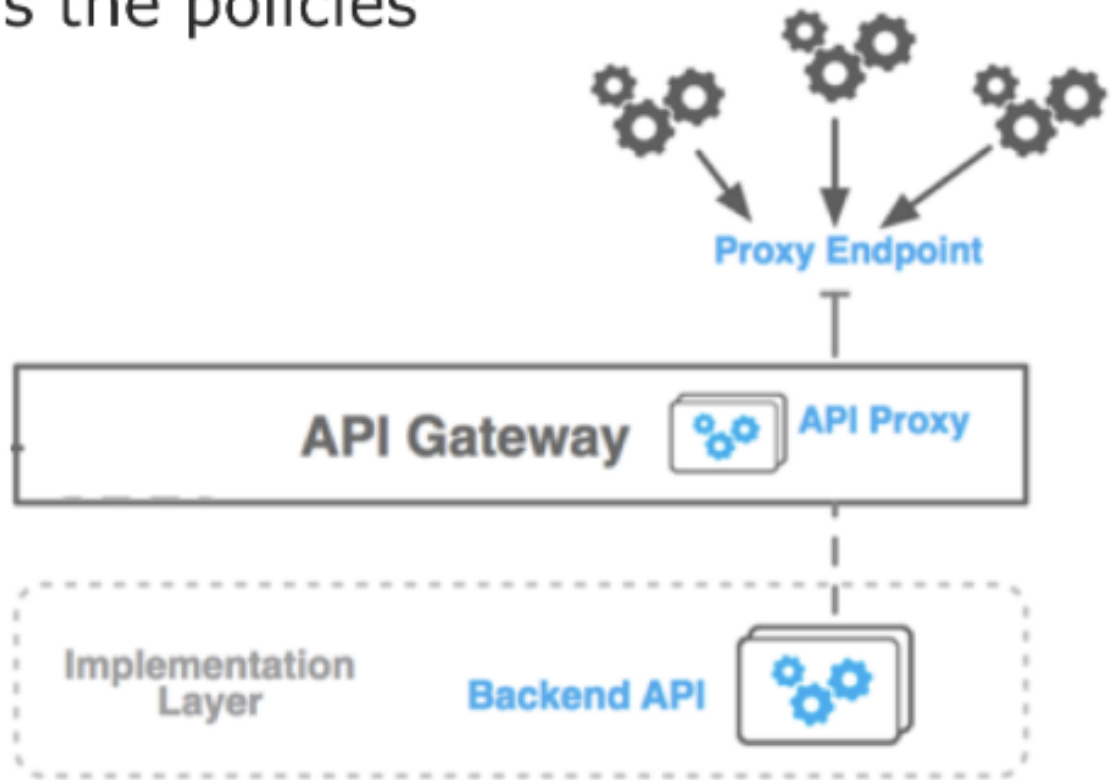


66. Close the browser tab.

Restricting access to APIs

Restricting access to APIs

- Use **API Manager** to manage access to API proxies
  - Define SLA tiers
  - Apply runtime policies
- The **API Gateway** enforces the policies



Applying policies to restrict access

- There are **out-of-the box** policies for many common use cases
  - Rate limiting
  - Spike control
  - Security

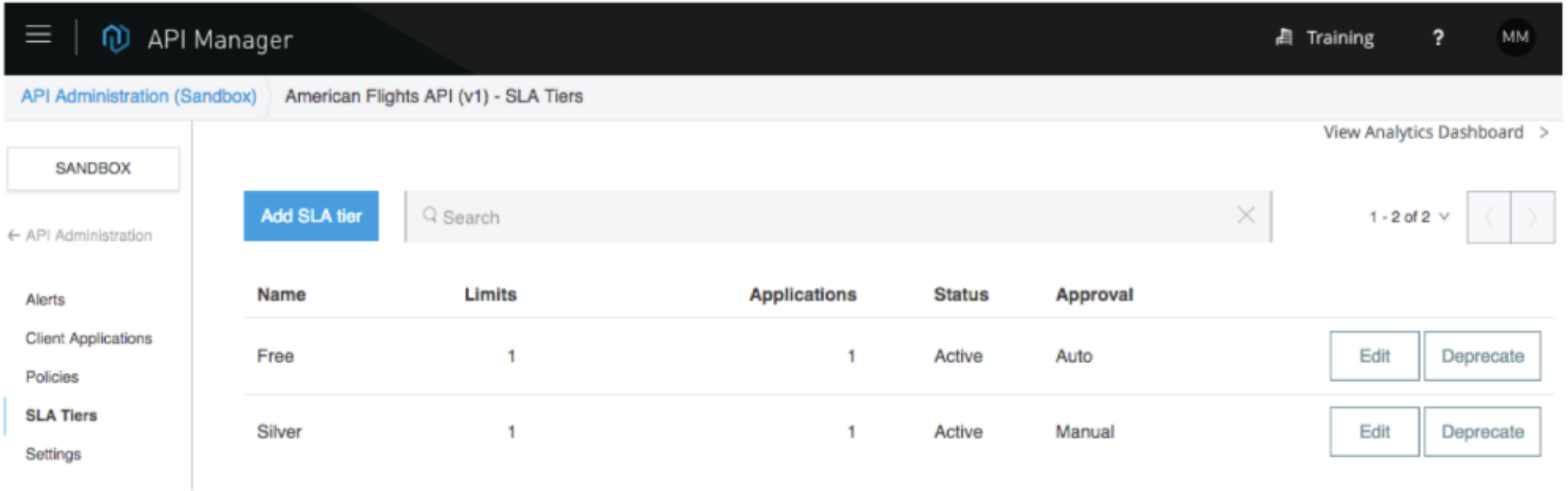
- You can define **custom** policies (using XML and YAML)

- You can apply **multiple** policies and set the order

Client ID enforcement	JSON threat protection
Cross-Origin resource sharing	Basic Authentication - LDAP
OAuth 2.0 access token enforcement	Message Logging
Header Injection	Rate limiting
Header Removal	Rate limiting - SLA based
Basic authentication - Simple	Spike Control
IP blacklist	XML threat protection
IP whitelist	

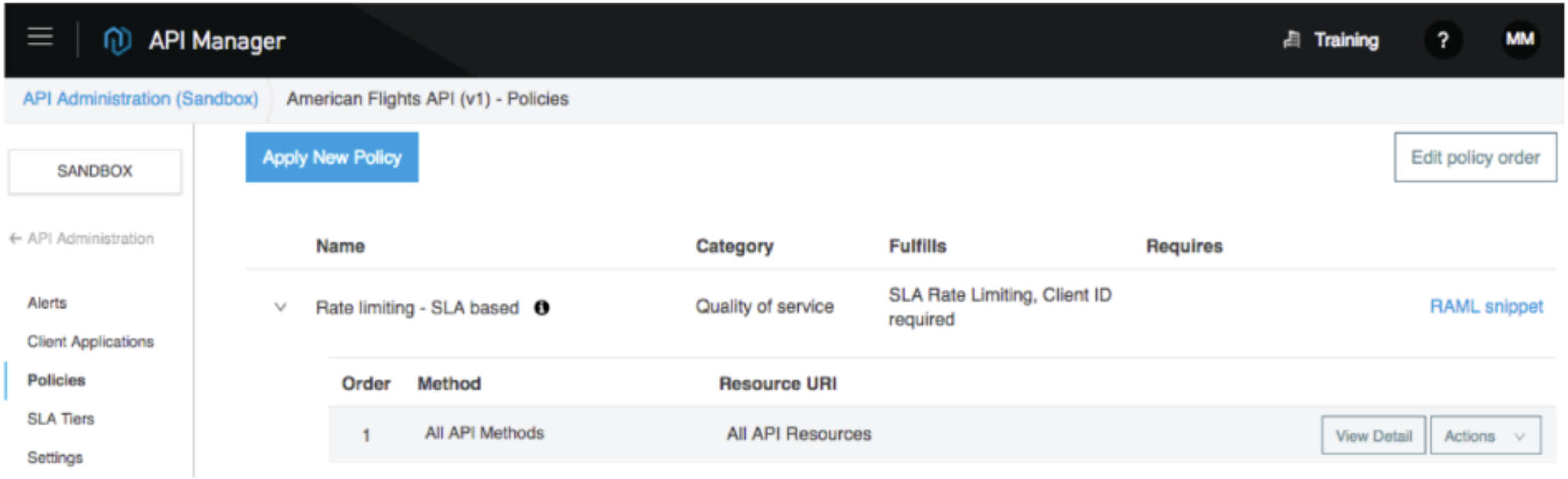
Using SLA tiers to restrict access by client ID

- A **S**ervice **L**evel **A**greement tier defines the # of requests that can be made per time frame to an API
  - Request approval can be set to automatic (free) or manual (for tiers that cost $)



Walkthrough 5-3: Restrict API access with policies and SLAs

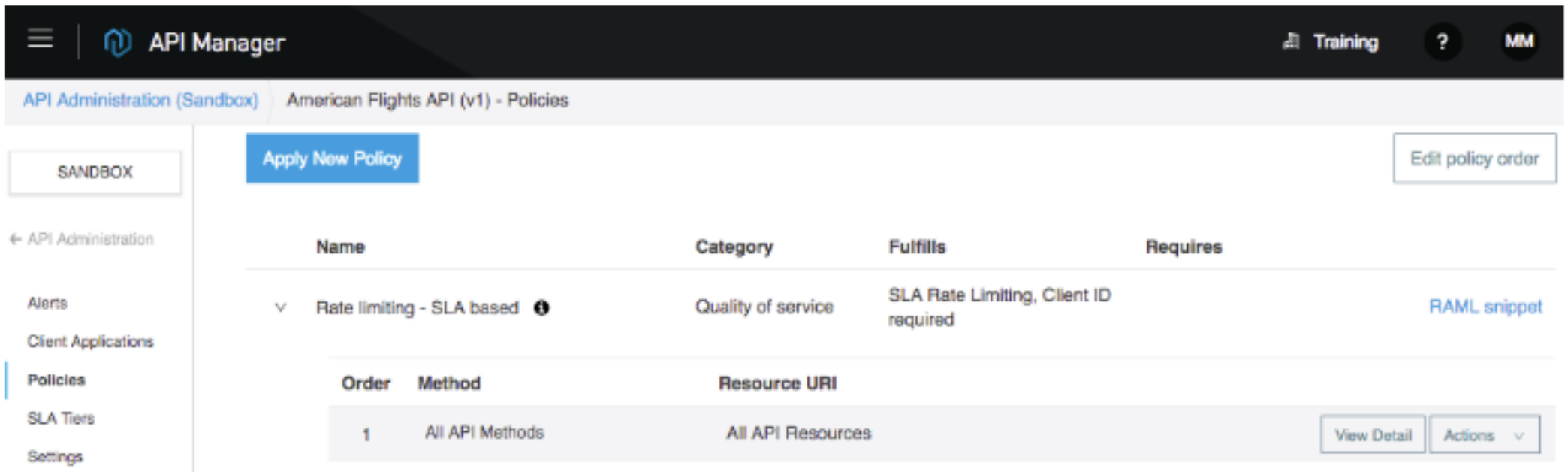
- Add and test a rate limiting policy
- Add SLA tiers, one with manual approval required
- Add and test a rate limiting SLA based policy



Walkthrough 5-3: Restrict API access with policies and SLAs

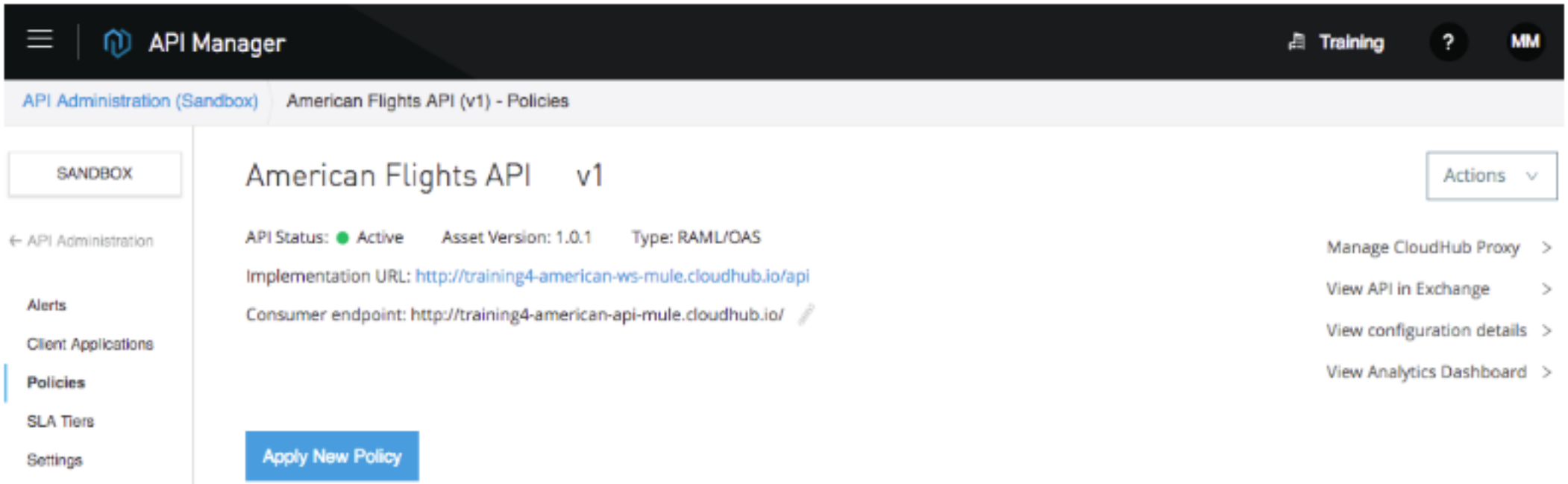
In this walkthrough, you govern access to the API proxy. You will:

- Add and test a rate limiting policy.
- Add SLA tiers, one with manual approval required.
- Add and test a rate limiting SLA based policy.



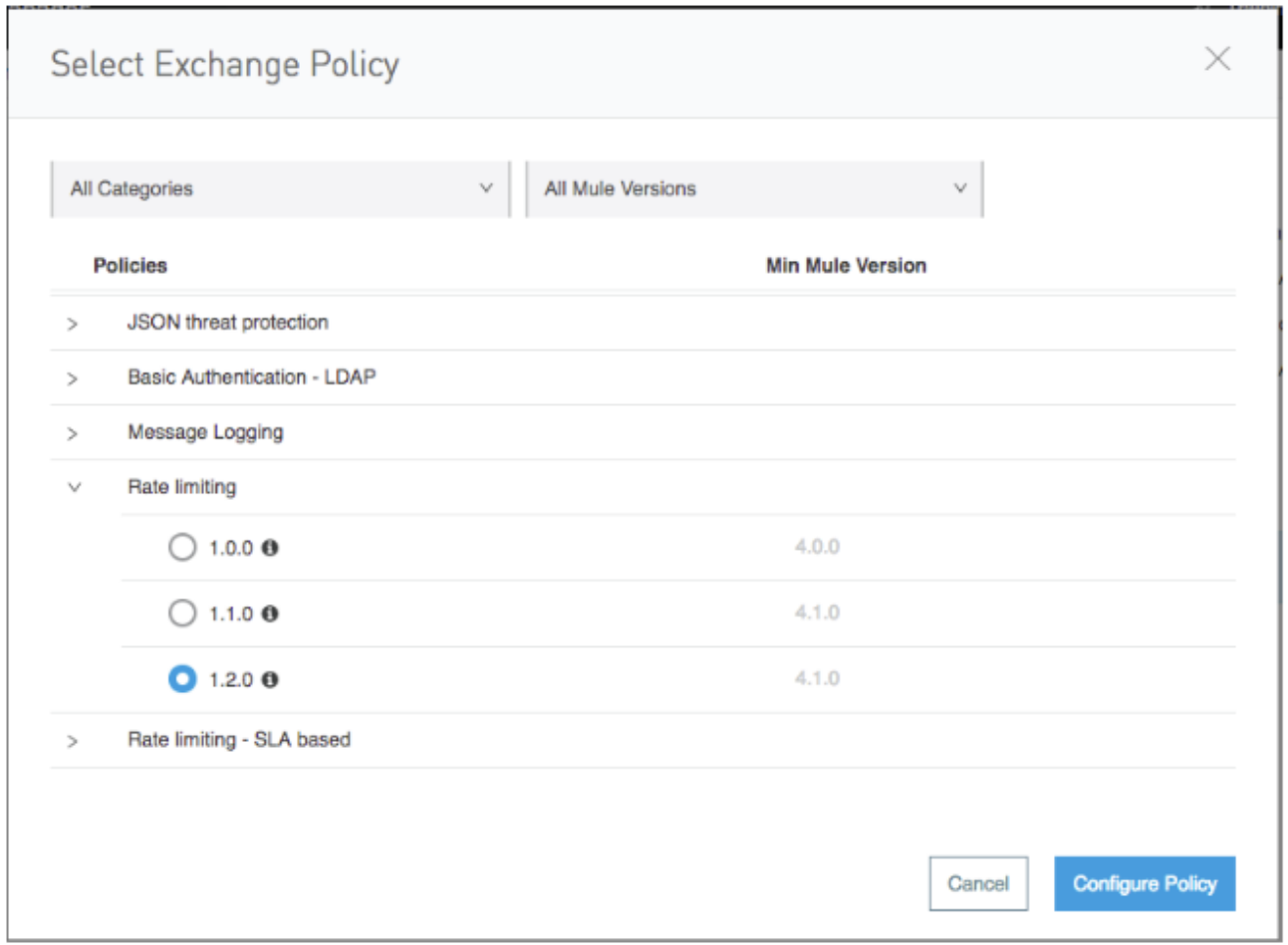
Create a rate limiting policy

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



3. Click the Apply New Policy button.

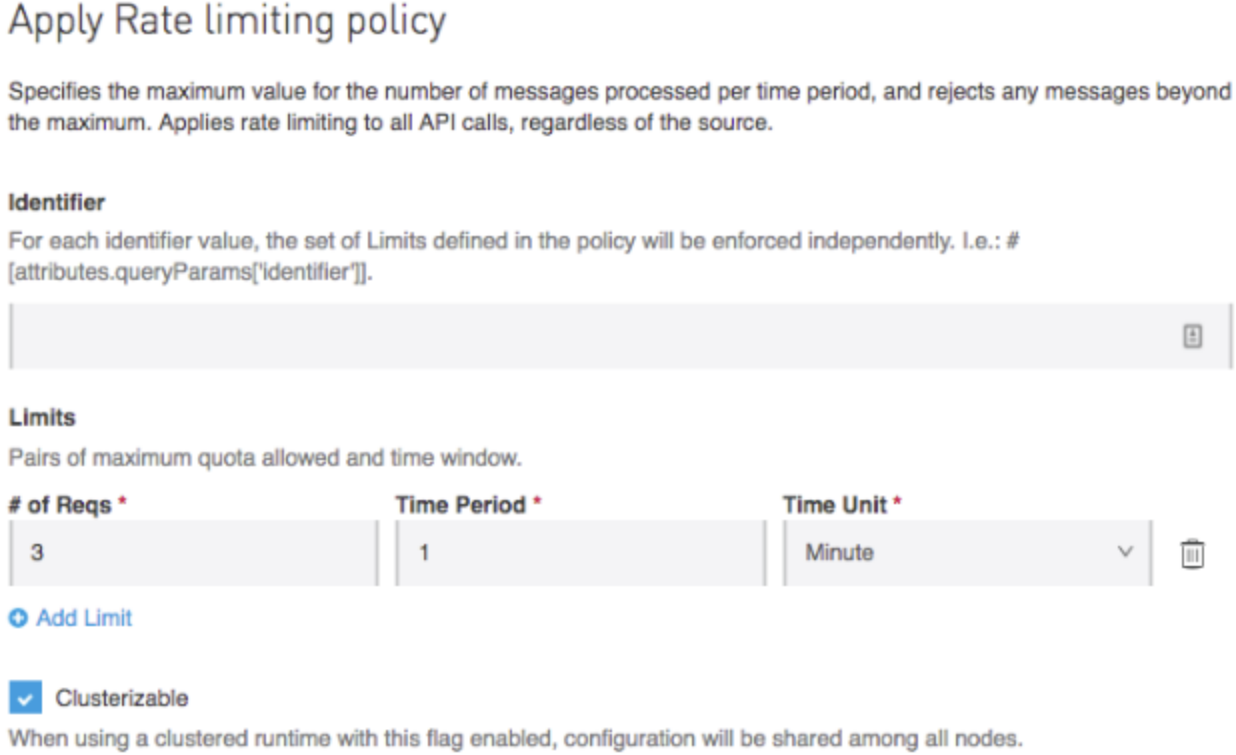
3. Click the Apply New Policy button.
4. In the Select Exchange Policy dialog box, expand Rate limiting and select the latest version for the Mule runtime version you are using.

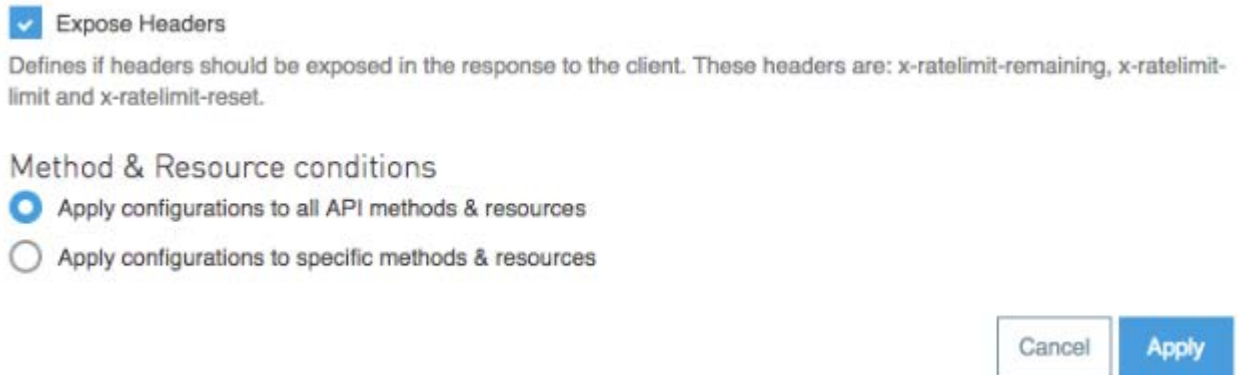


5. Click Configure Policy.

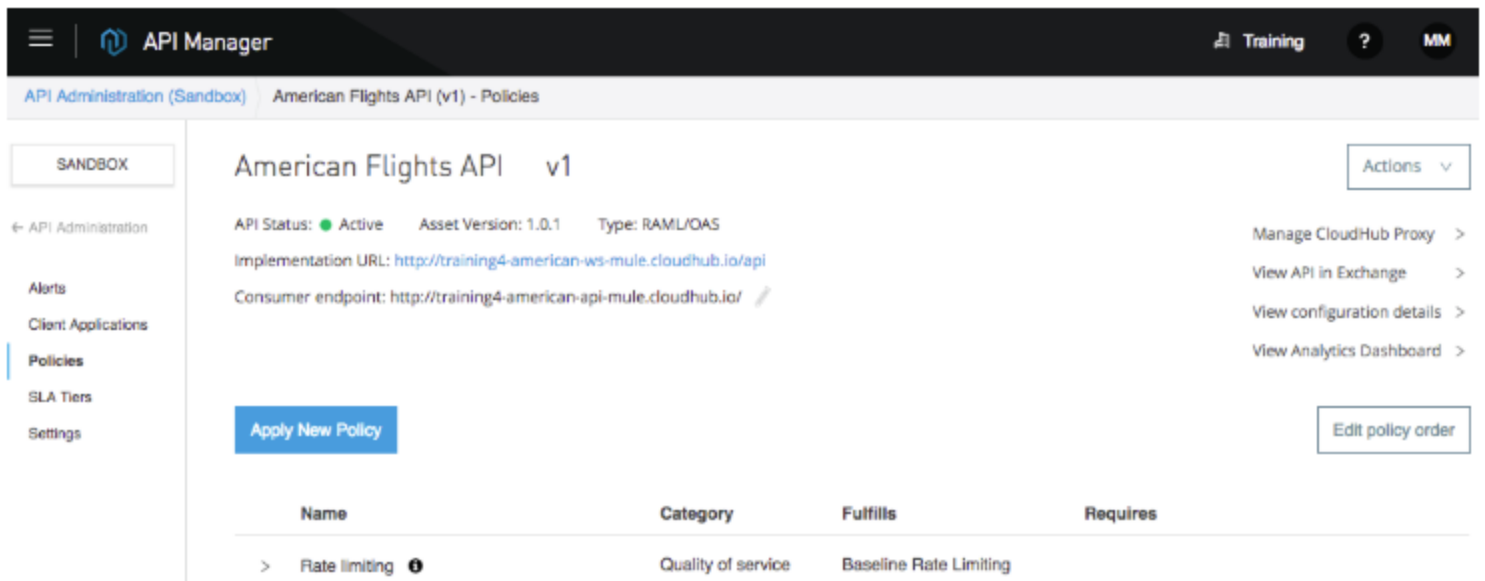
6. On the Apply Rate limiting policy page, set the following values and click Apply:

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

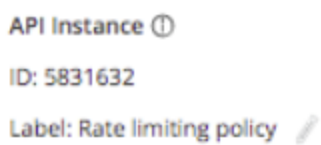




7. Click Apply; you should see the policy listed for your API.



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

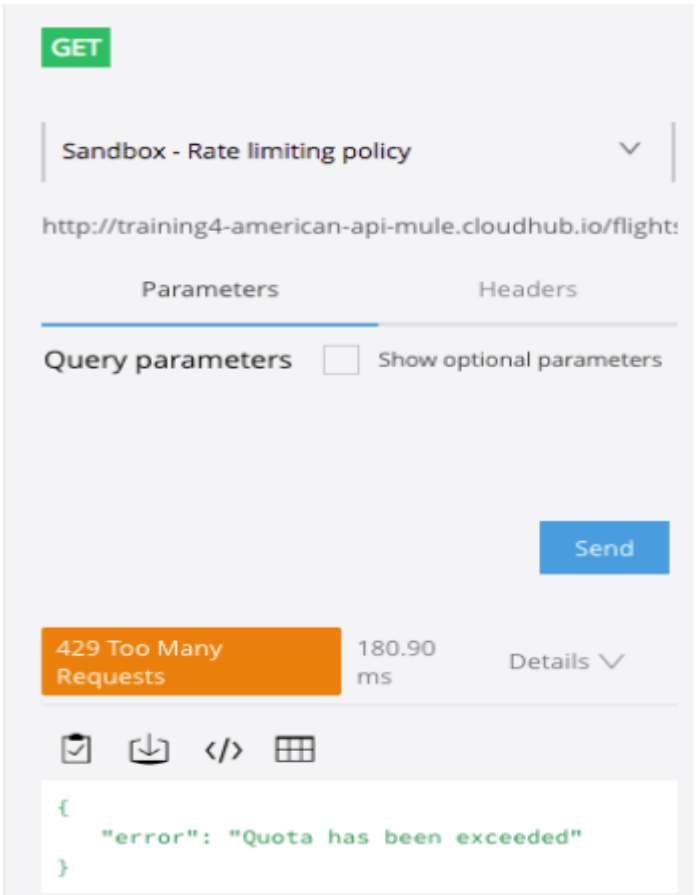


Test the new rate limiting policy

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.

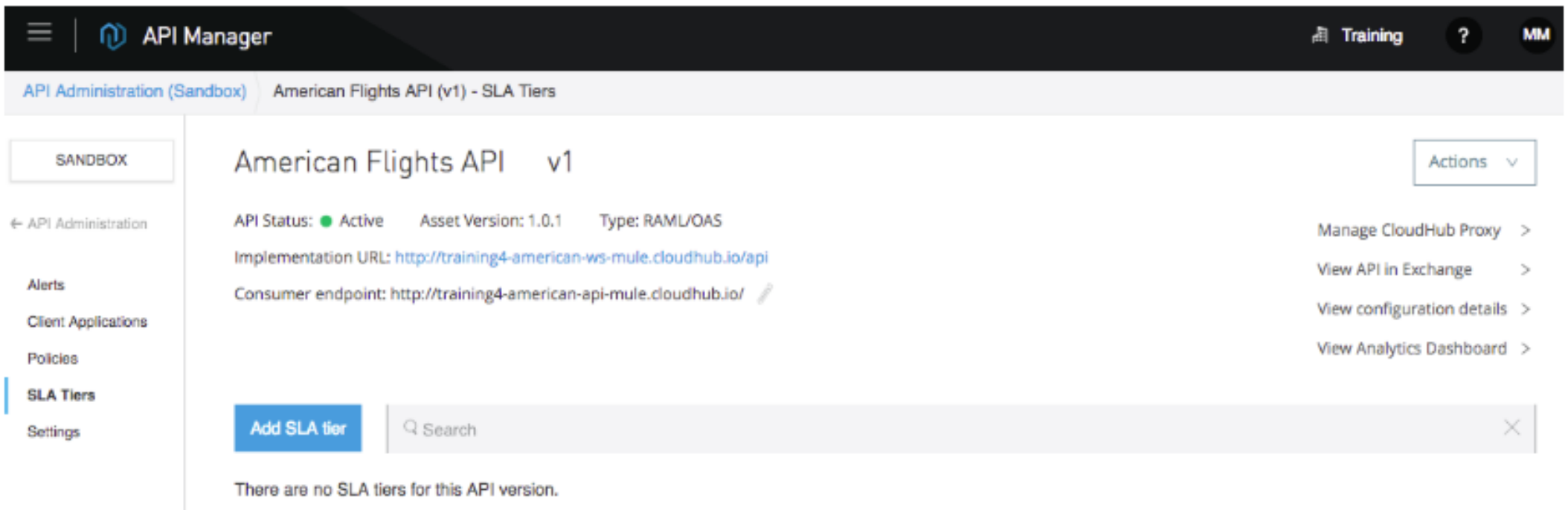
    Note: You may need to refresh the page to see the new label for the API instance.

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



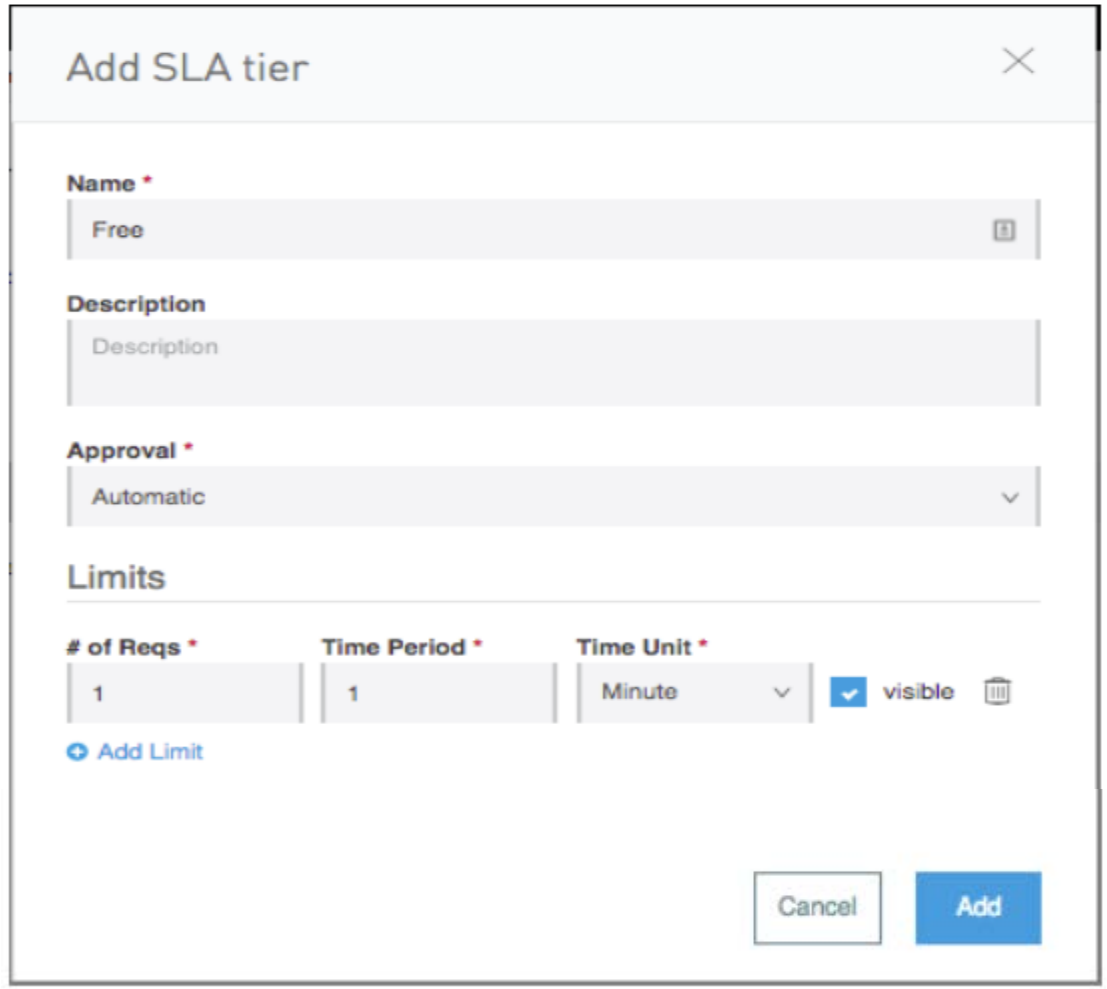
Create SLA tiers

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



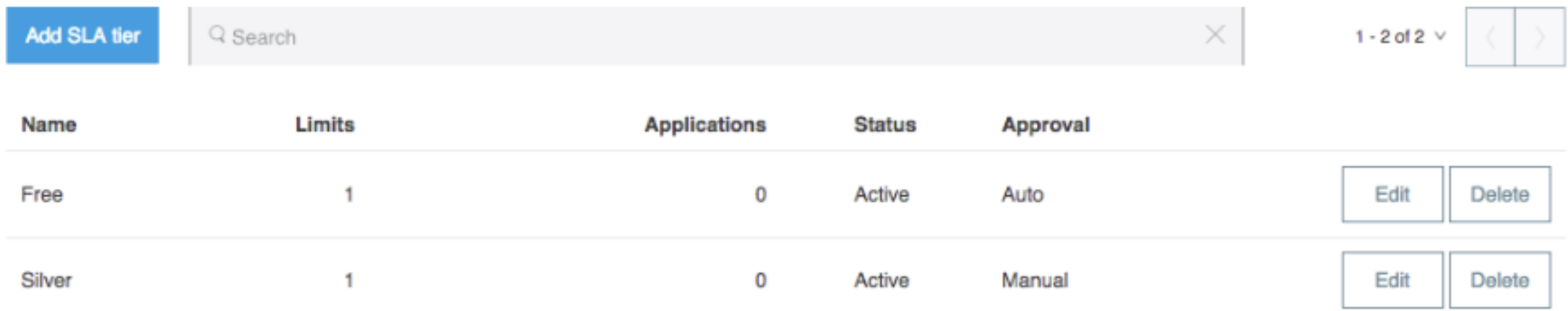
17. In the Add SLA tier dialog box, set the following values:

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



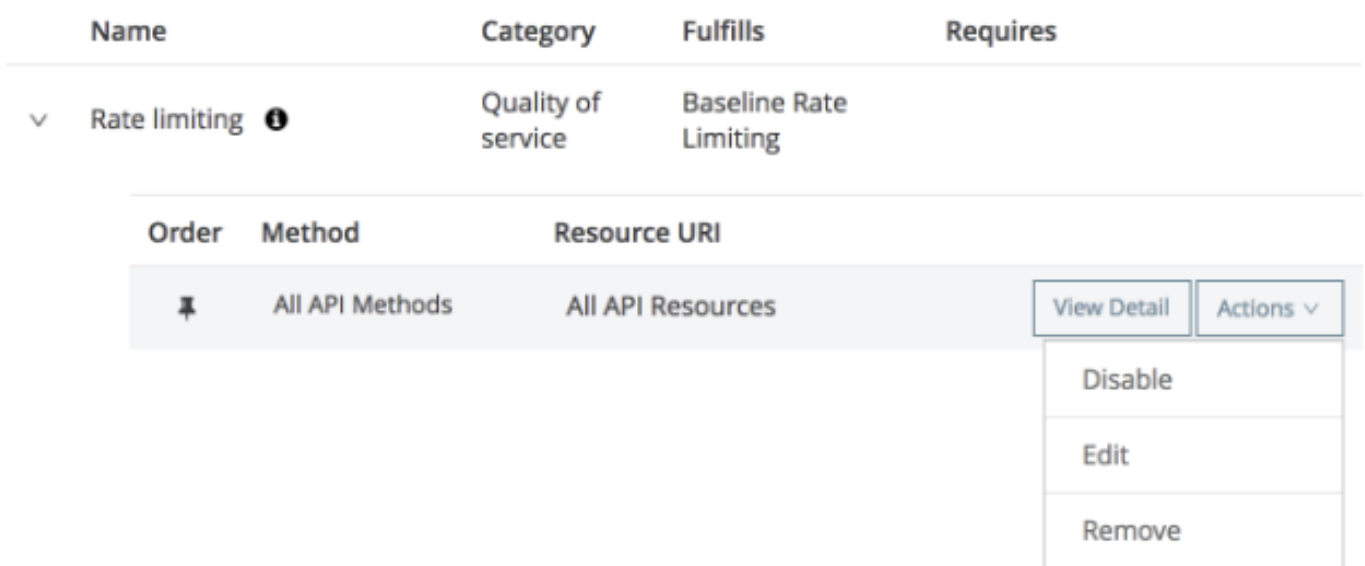
18. Click the Add button.
19. Create a second SLA tier with the following values:

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



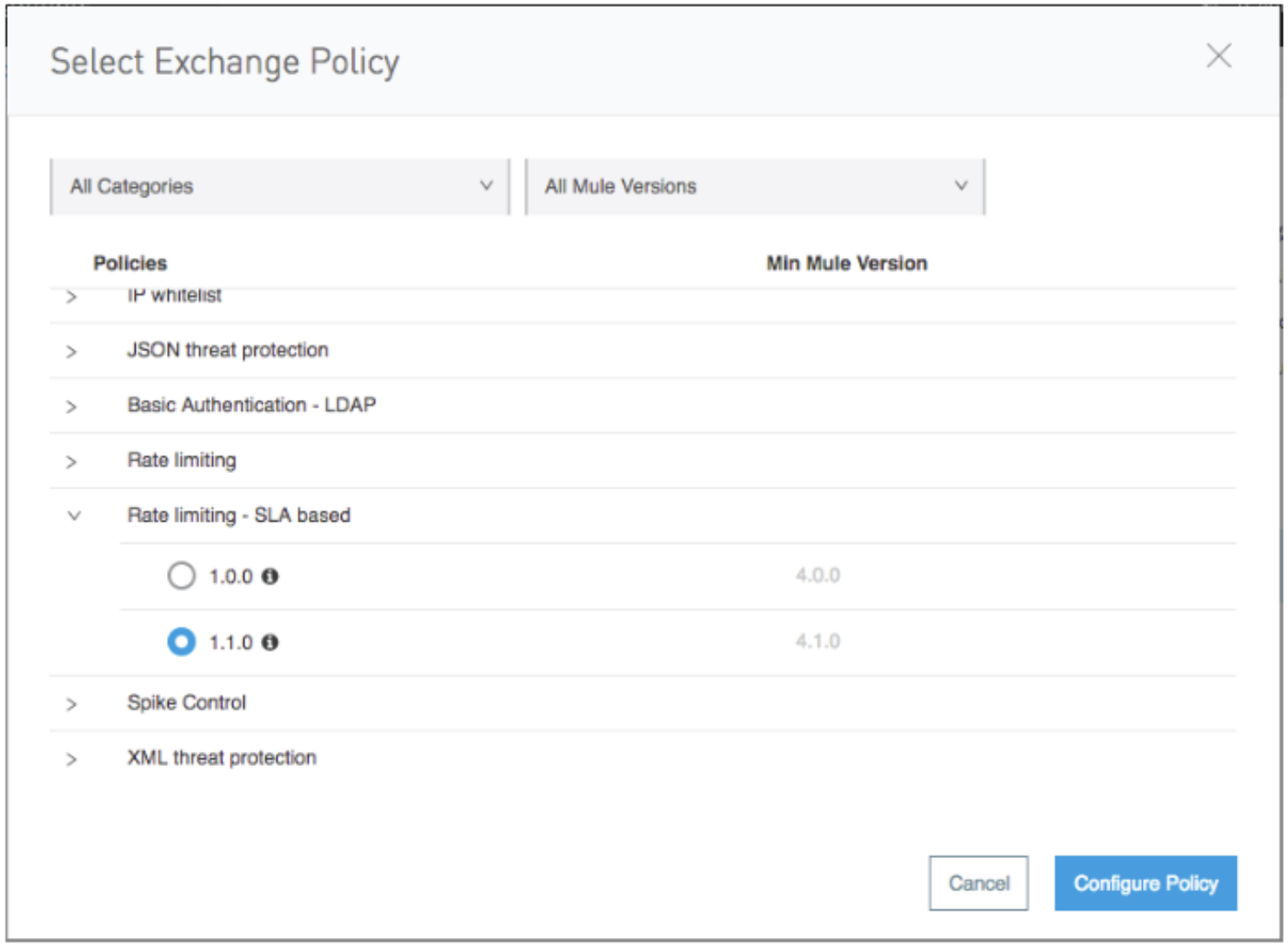
Change the policy to rate limiting – SLA based

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

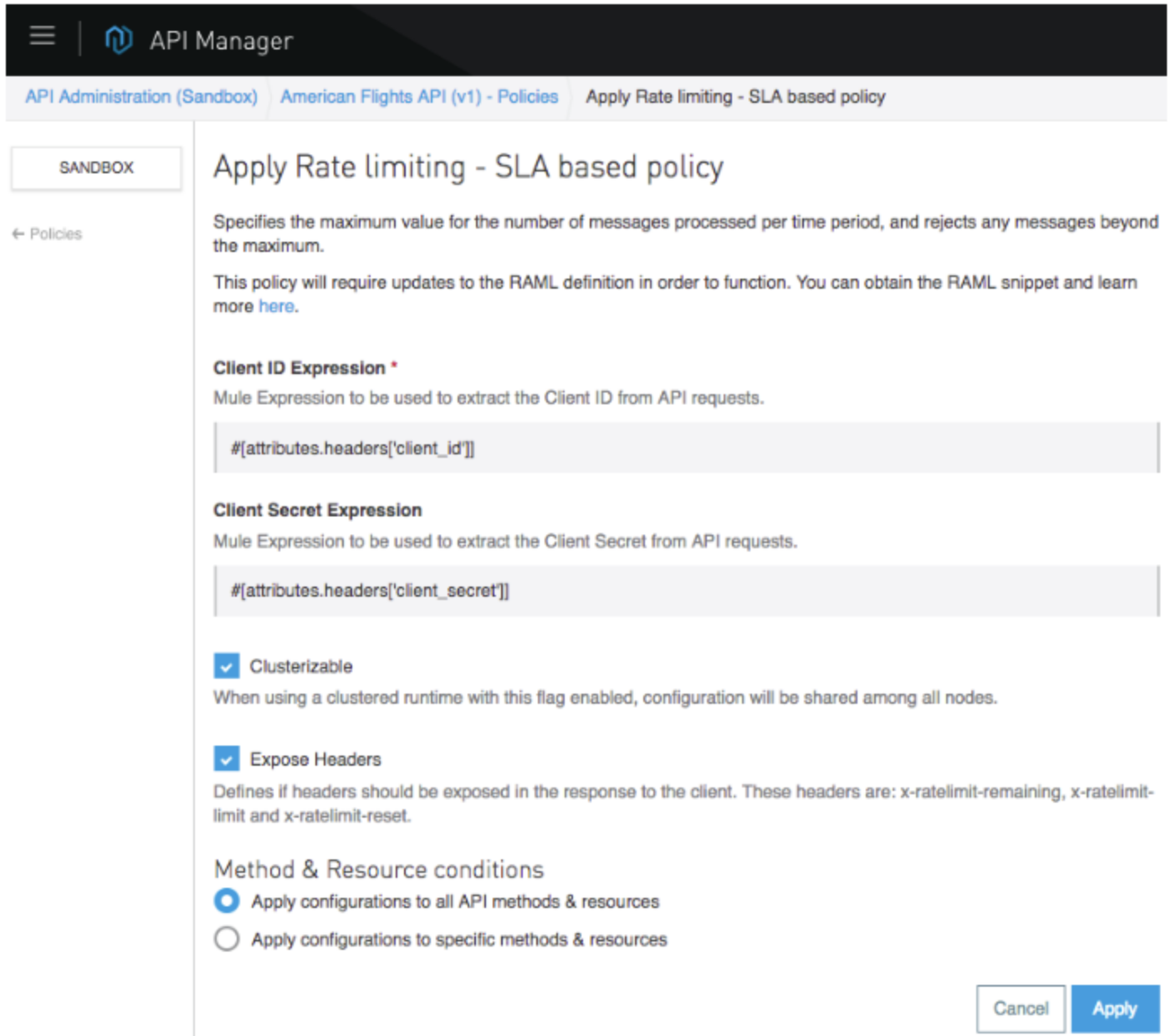


23. In the Remove policy dialog box, click Remove.
24. Click the Apply New Policy button.
25. In the Select Policy dialog box, expand Rate limiting - SLA based and select the latest version for the Mule runtime version you are using.

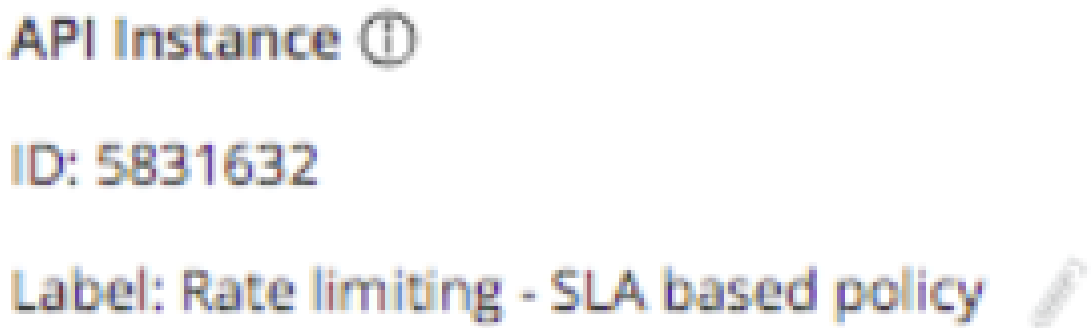
26. Click Configure Policy.



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 headers.

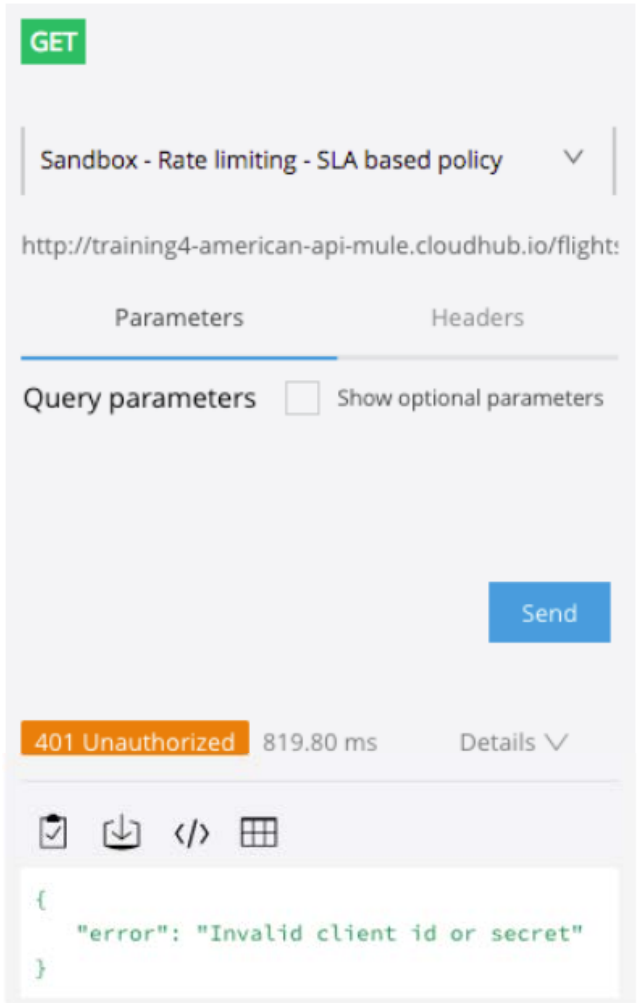


28. Click Apply.
29. In the left-side navigation, select Settings.
30. Change the API instance label to Rate limiting – SLA based policy.



Test the rate limiting – SLA based policy in Exchange

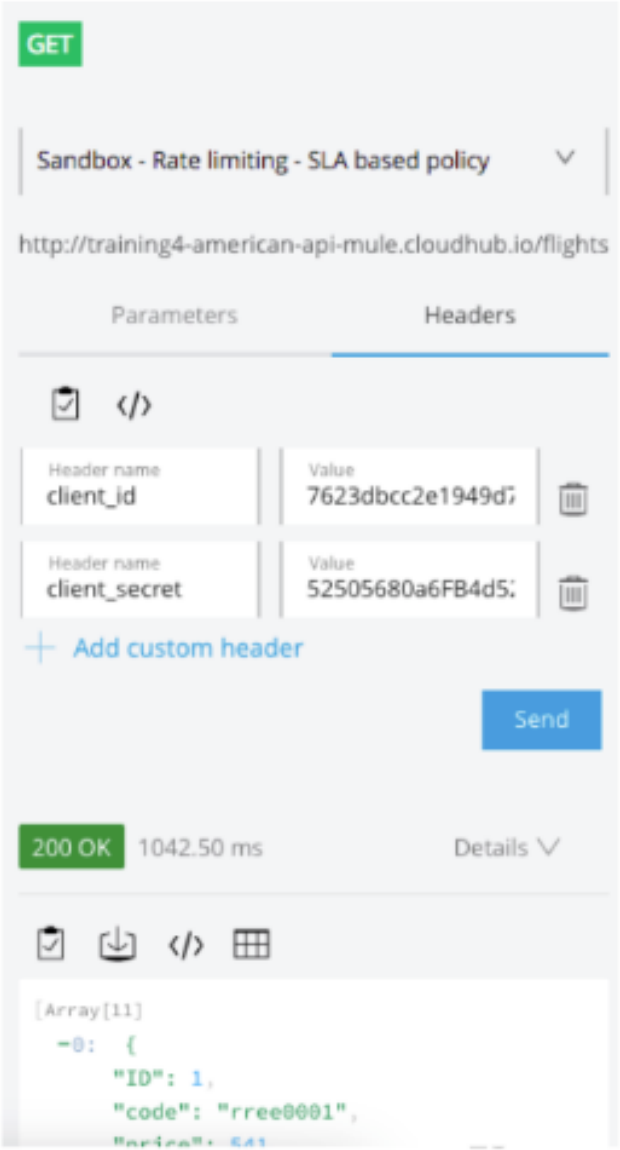
31. Return to the browser tab with your API in Exchange.
32. Refresh the page and select to make a call to the Sandbox – Rate limiting – SLA based policy.
33. Click Send; you should get a 401 Unauthorized response with a message that there is an invalid client id or secret.



Granting access to APIs

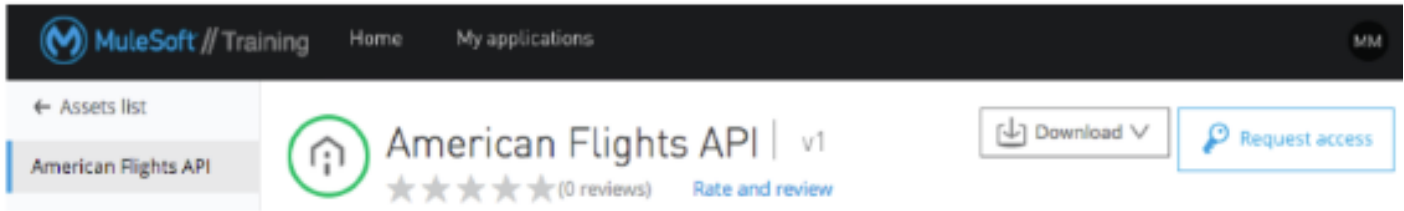
Enforcing access to APIs using SLA tiers

- To enforce, apply an **SLA based** rate limiting policy
- SLA based policies require all applications that consume the API to
  - **Register** for access to a specific tier
    - From an API portal in private or public Exchange
  - **Pass their client credentials** in calls made to the API



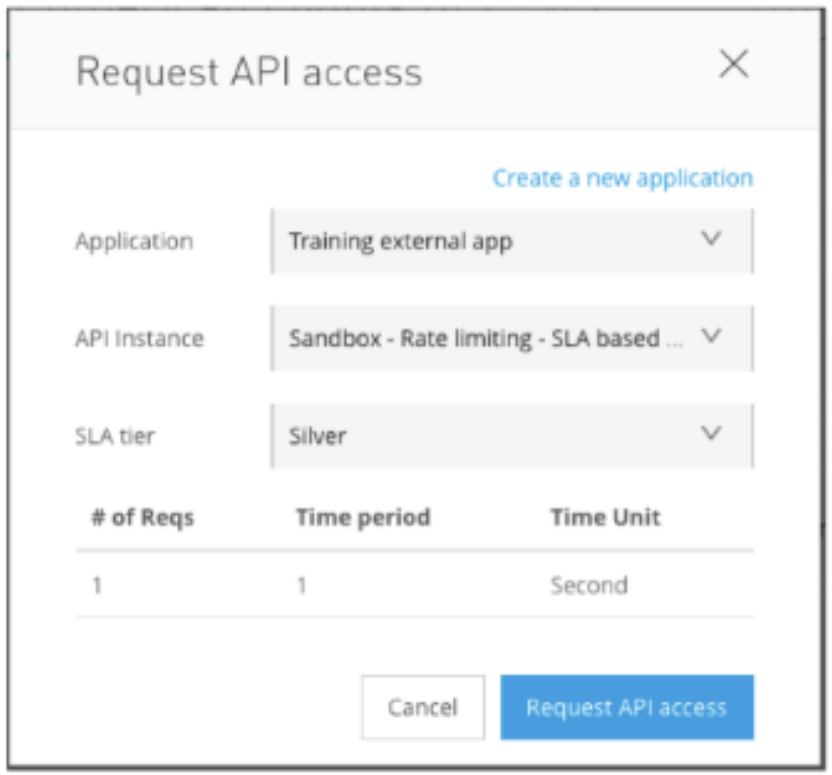
Requesting access to SLA tiers

- If an API has an SLA-based policy, a Request API access button appears in API portal



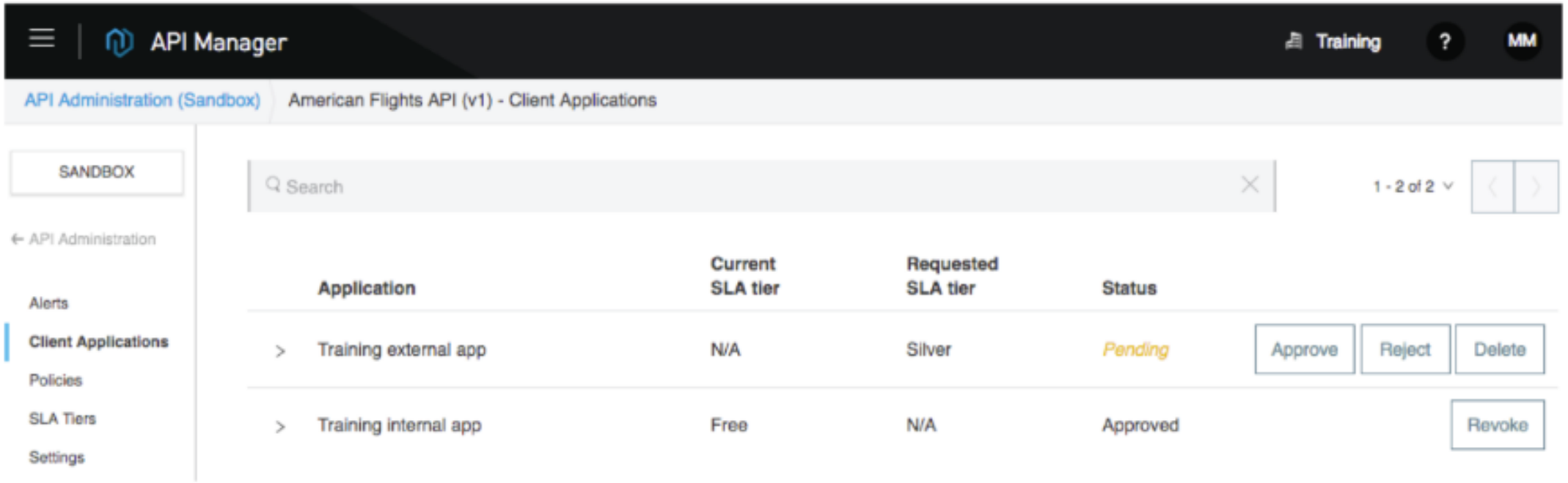
- To request access, developer must belong to the Anypoint Platform organization and be logged in

- When developers request access, they must
  - Register/add an app to their Anypoint Platform account
  - Select a tier



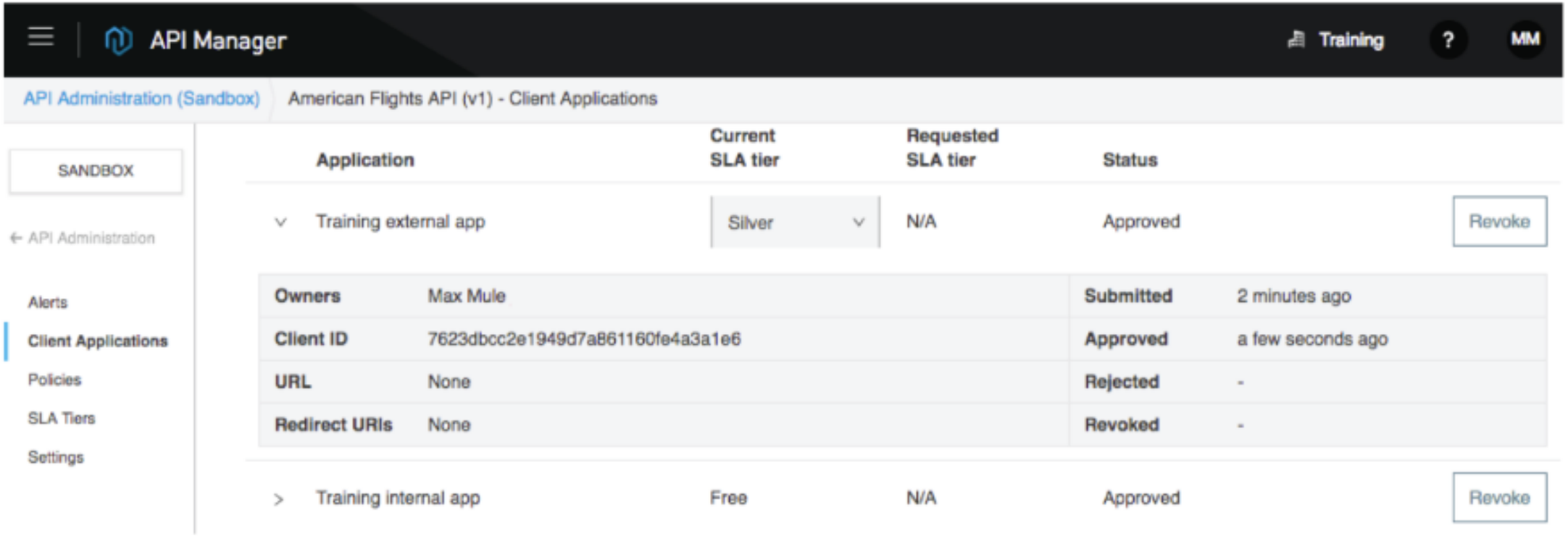
Approving SLA tier access requests

- For tiers with manual approval, emails are sent to the Organization Administrators when developers request access for applications
- Organization Administrators can review the applications in API Manager and approve, reject, or revoke requests



Walkthrough 5-4: Request and grant access to a managed API

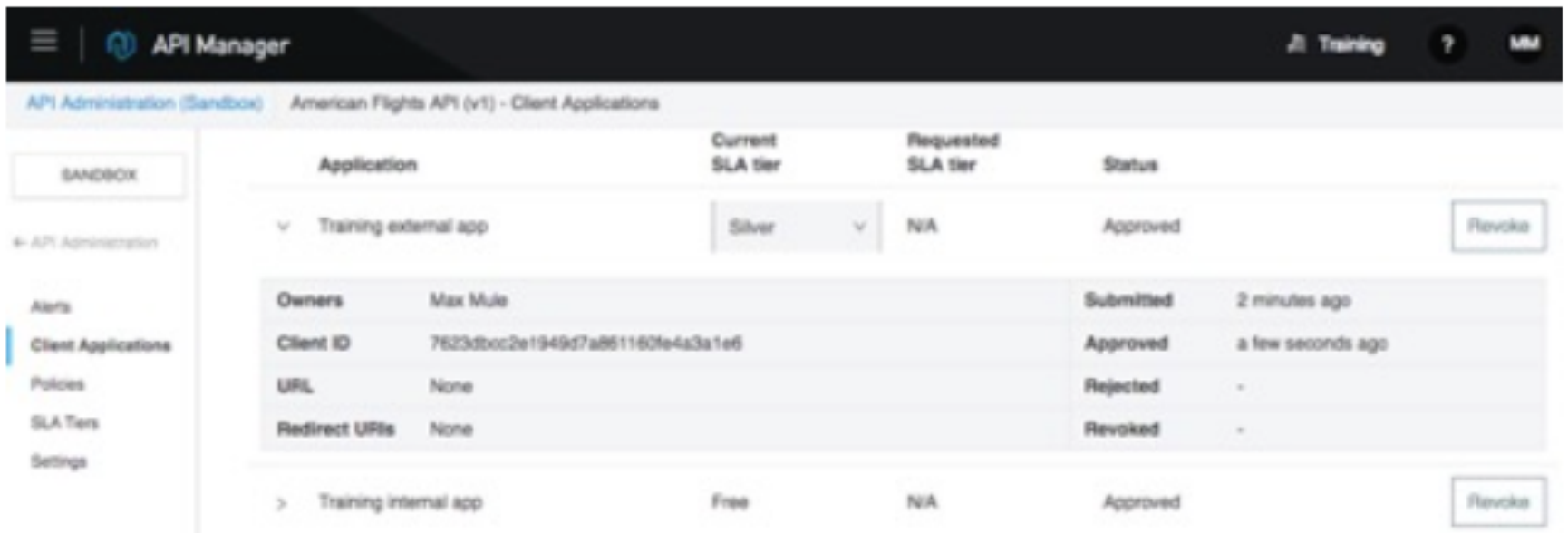
- Request application access to SLA tiers from private and public API portals
- Approve application requests to SLA tiers in API Manager



Walkthrough 5-4: Request and grant access to a managed API

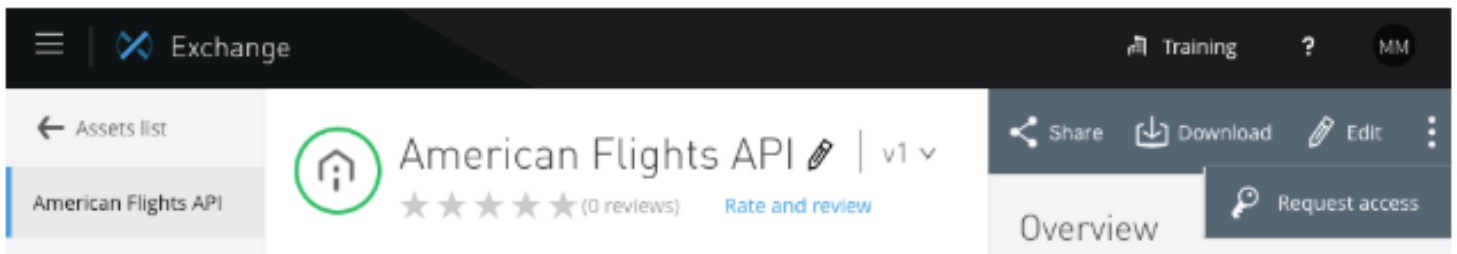
In this walkthrough, clients request access to an API proxy and administrators grant access. You will:

- Request application access to SLA tiers from private and public API portals.
- Approve application requests to SLA tiers in API Manager.

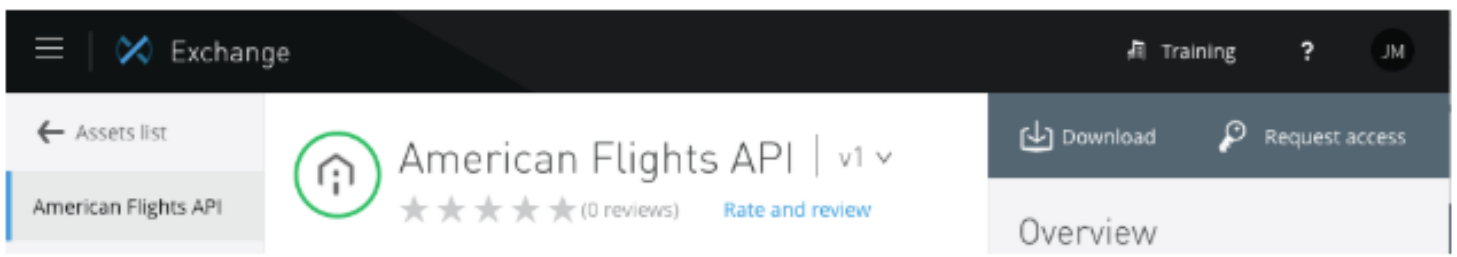


Request access to the API as an internal consumer

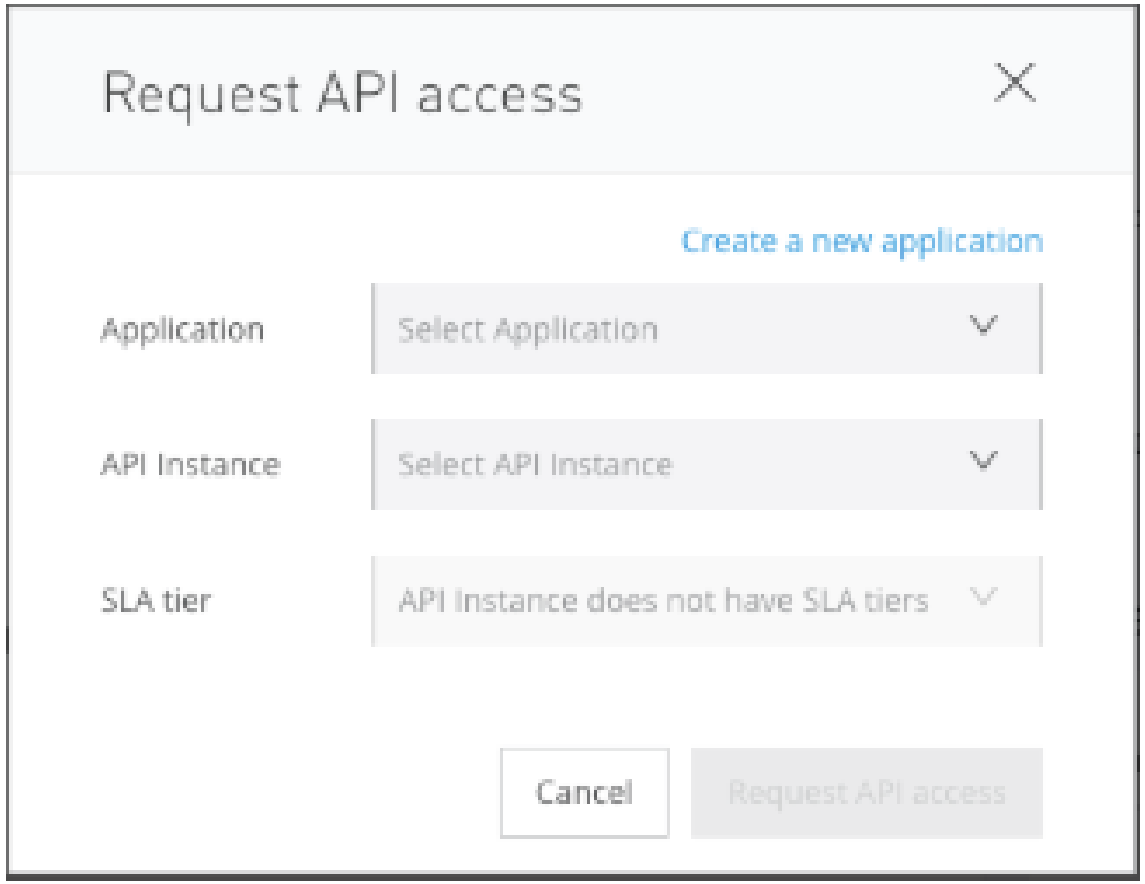
1. Return to the browser tab with Anypoint Exchange.
2. In the left-side navigation, select the name of the API to return to its home page.
3. Click the more options button in the upper-right corner and select Request access.



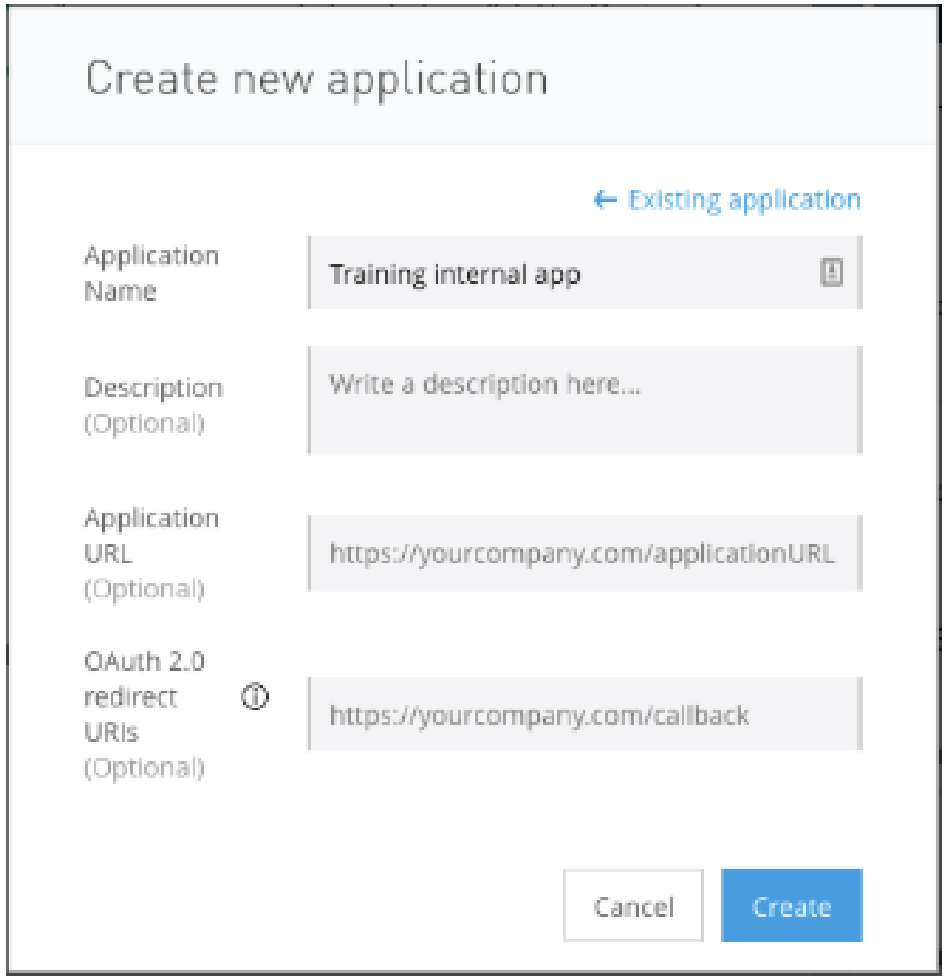
Note: Other internal users that you shared the API with that do not have Edit permissions will see a different menu.



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

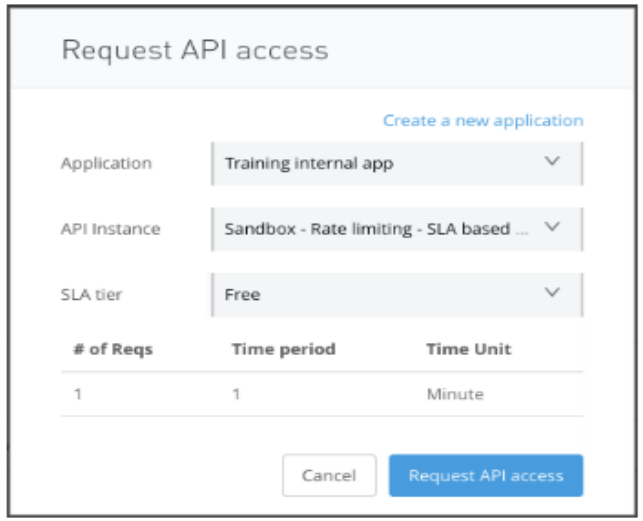


5. In the Create new application dialog box, set the name to Training internal app and click Create.

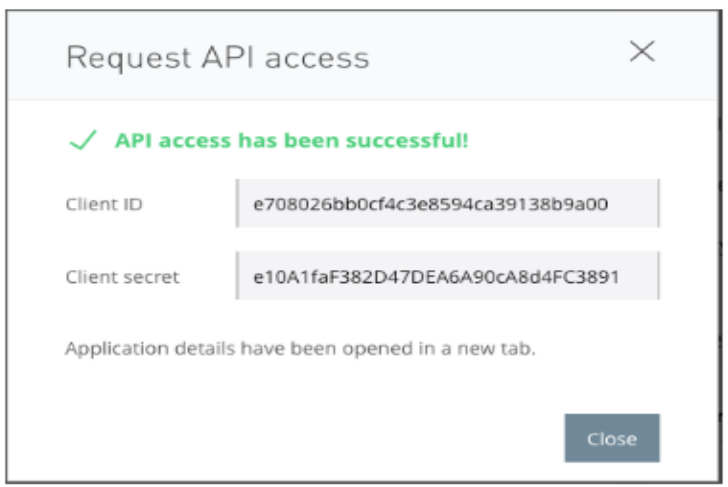


6. In the Request API access dialog box, set the API instance to Sandbox – Rate limiting SLA - base

7. Set the SLA tier to Free.



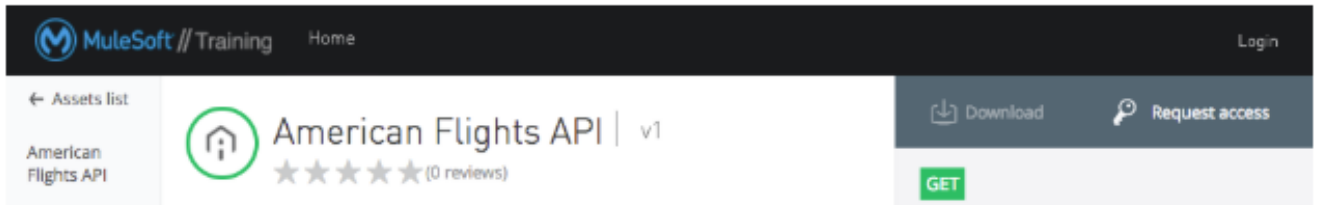
8. Click Request API access.
9. In the Request API access dialog box, view the assigned values for the client ID and client secret.



10. Click Close.

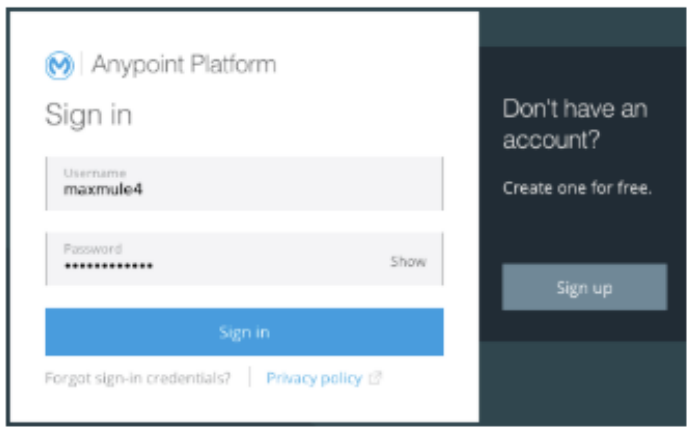
Request access to the API as an external consumer

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



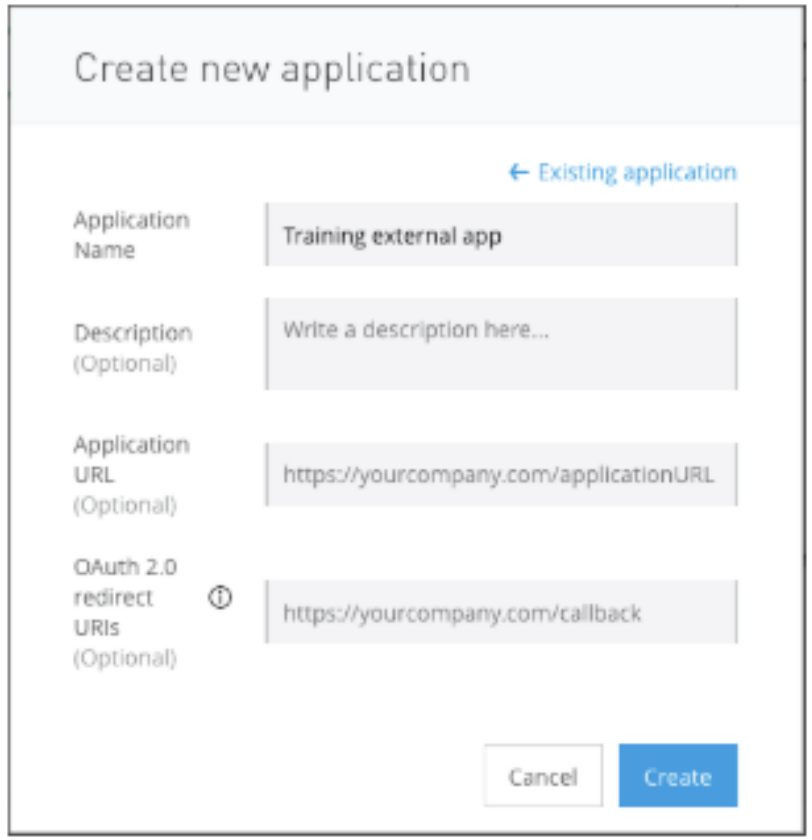
13. Click the Request access button; you should get a page to sign in or create an Anypoint Platform account.
14. Enter your existing credentials and click Sign in.

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



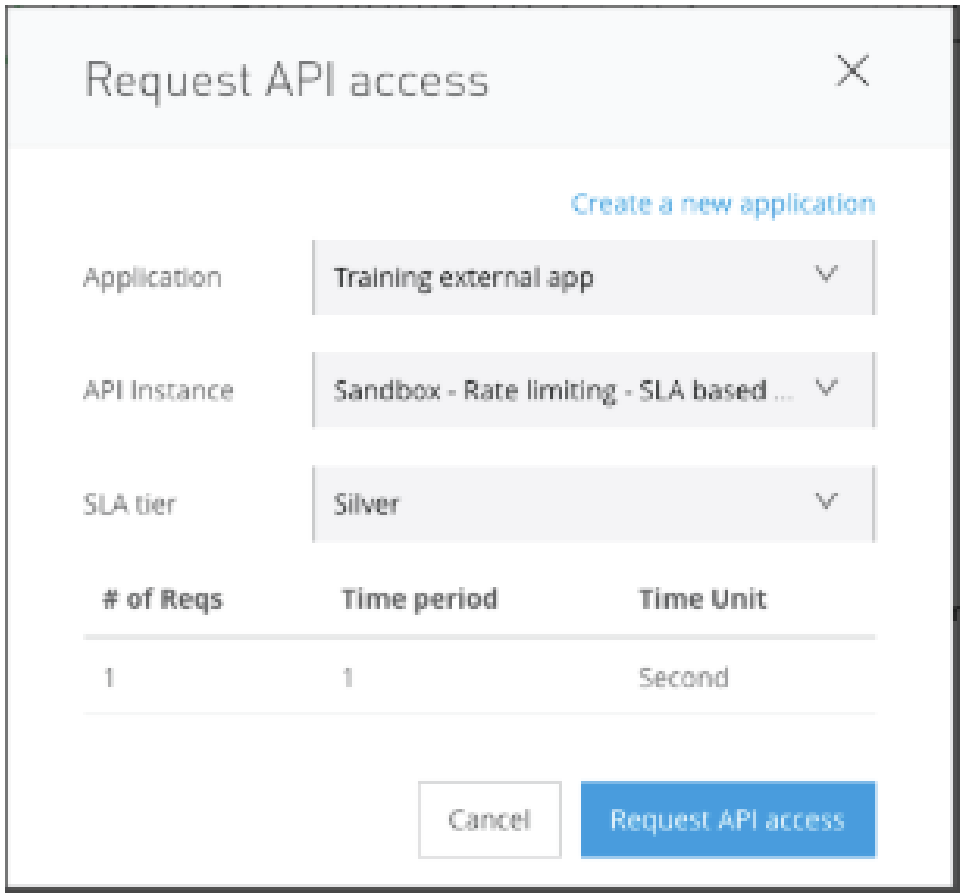
15. Back in the public portal, click the Request access button again.
16. In the Request API access dialog box, click the Create a new application button

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



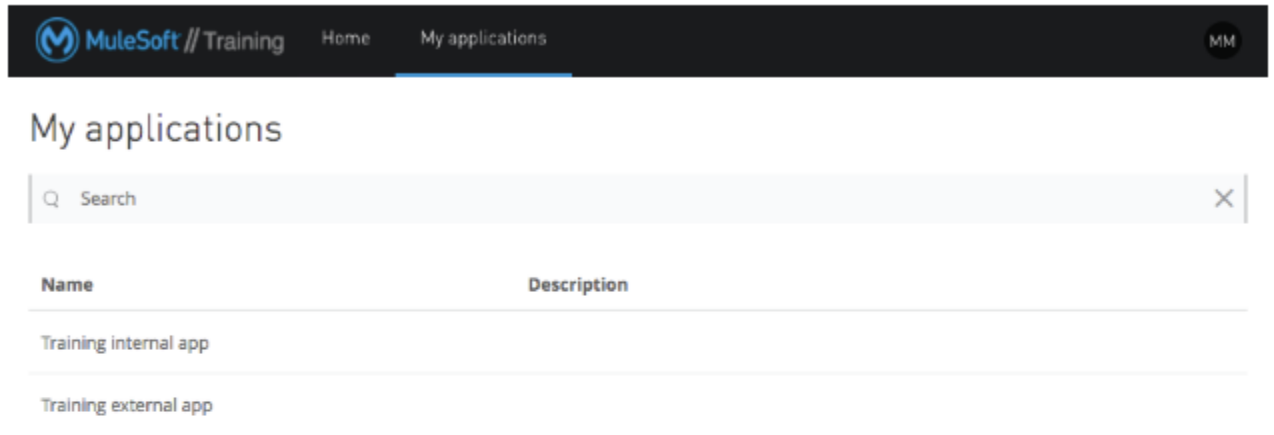
18. Click Create.
19. In the Request API access dialog box, set the API instance to Sandbox – Rate limiting SLA-based policy.

20. Set the SLA tier to Silver.

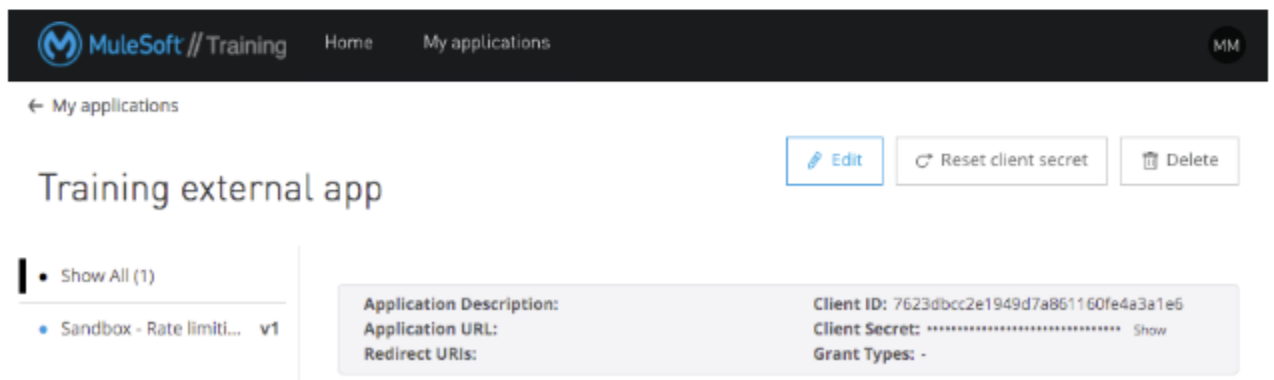


21. Click Request API access.
22. In the Request API access dialog box, click Close.

23. 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.



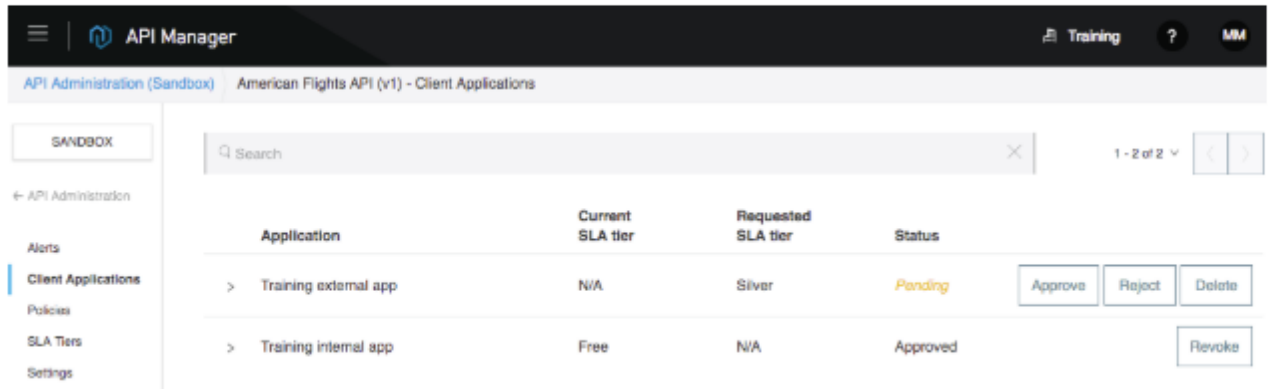
24. Click the link for Training external app; you should see what APIs the application has access to, values for the client ID and secret to access them, and request data.



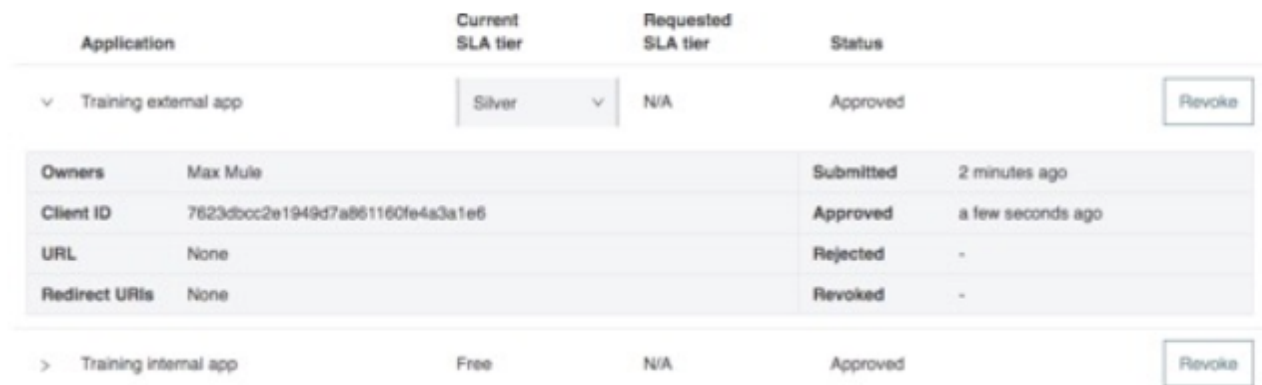
25. Leave this page open in a browser so you can return to it and copy these values.

Grant an application access

26. Return to the browser window and tab with the Settings page for American Flights API (v1) in API Manager.
27. In the left-side navigation, select Client Applications; you should see the two applications that requests access to the API.

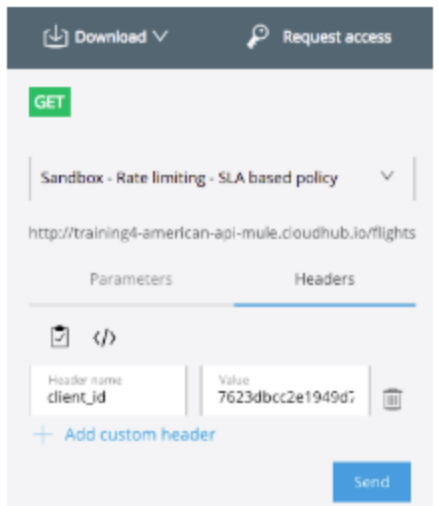


28. Click the Approve button for the application requesting Silver tier access.
29. Expand the Training external app row and review its information.
30. Copy the value of the client_id.



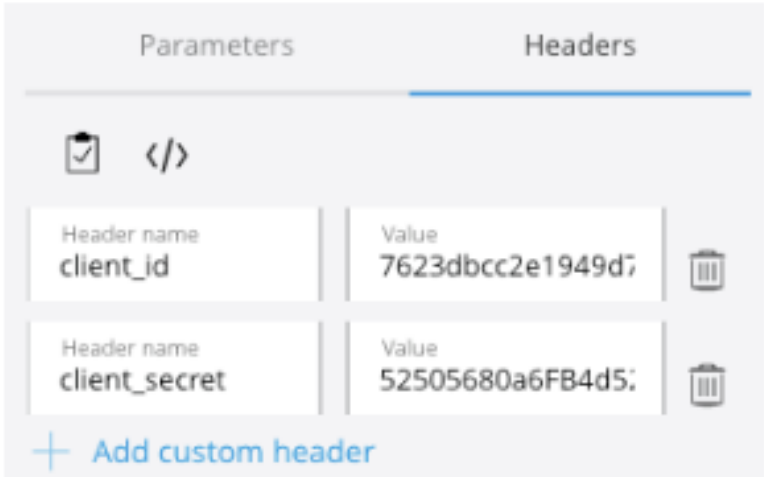
Add authorization headers to test the rate limiting – SLA based policy from an API portal

31. Return to the browser window and tab with the API console in the public portal.
32. Try again to make a call to the Sandbox – Rate limiting – SLA based policy; you should still get a 401 Unauthorized response.
33. Select the Headers tab.
34. Click Add custom header.
35. Set the header name to client_id.
36. Set the value of client_id to the value you copied.

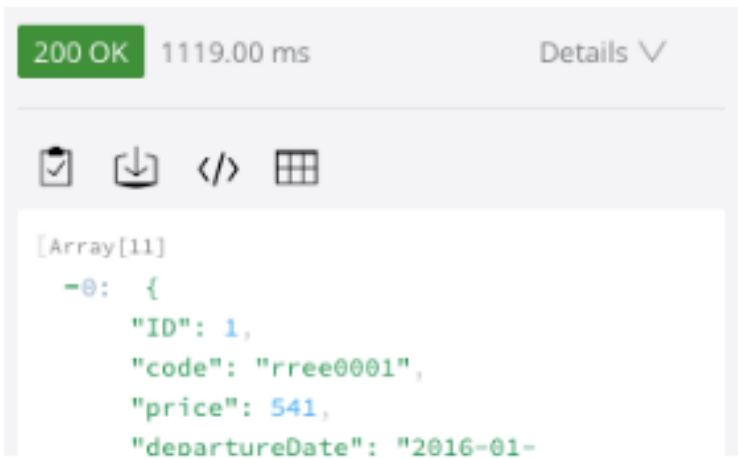


37. Return to the browser tab with My applications in the public portal.
38. Copy the value of the client_secret.
39. Return to the browser window and tab with the API console in the public portal.

40. Add another custom header and set the name to client_secret.
41. Set the client_secret header to the value you copied.



42. Click Send; you should now get a 200 response with flight results.

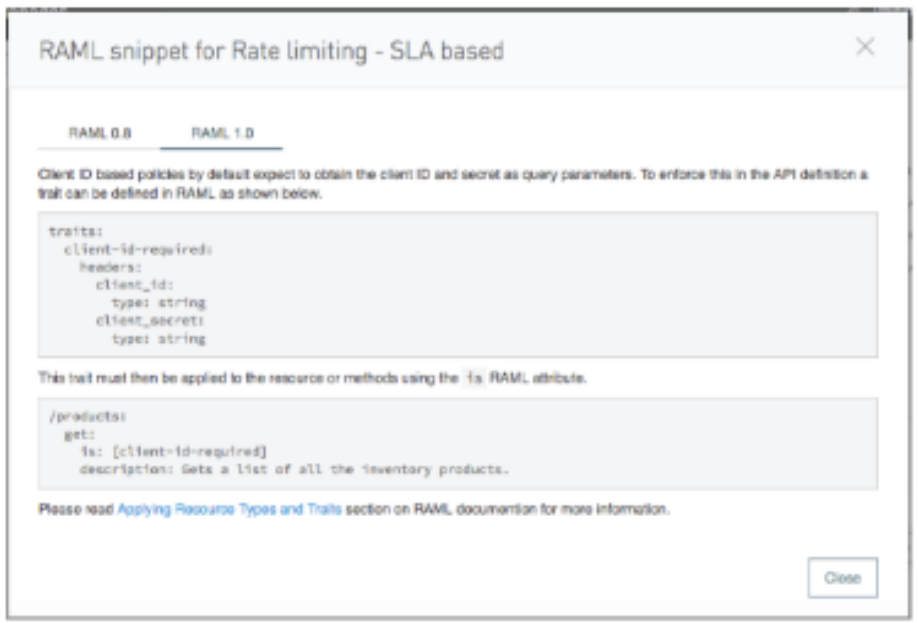


Adding client ID enforcement to API specifications

Adding client ID enforcement to API specifications

- You need to add client ID enforcement to the API spec for the
  - REST connector that is created for the API to enforce the authentication
  - Required headers to automatically show up in the API console so you don't have to manually add them for every call

- Instructions are in the RAML snippet for a policy in API Manager

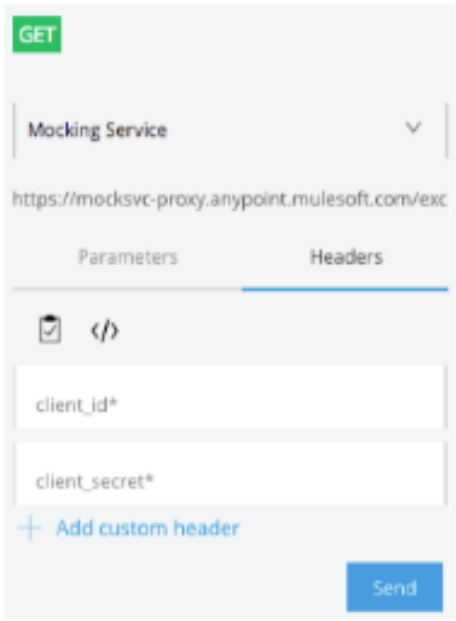


Walkthrough 5-5: (Optional) Add client ID enforcement to an API specification

- Modify an API specification to require client id and client secret headers with requests
- Update a managed API to use a new version of an API specification
- Call a governed API with client credentials from API portals

Note: If you do not complete this exercise for Fundamentals, the REST connector that is created for the API and that you use later in the course will not have client_id authentication

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

types:
  AmericanFlight: !include

traits:
  client-id-required:
    headers:
      client_id:
        type: string
      client_secret:
        type: string

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

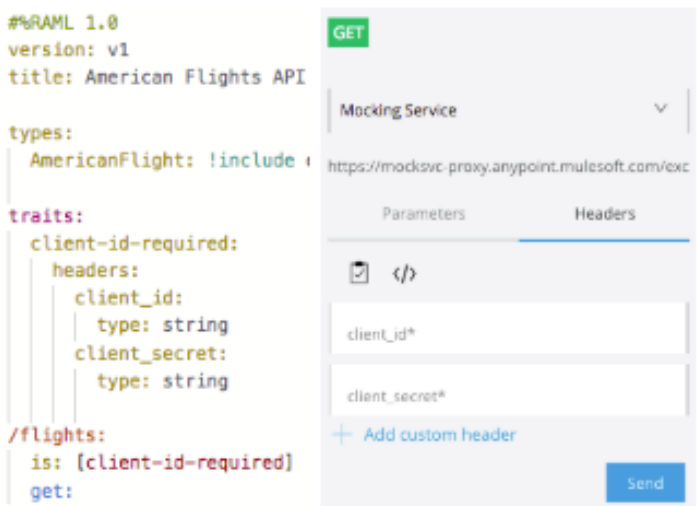


Walkthrough 5-5: Add client ID enforcement to an API specification

In this walkthrough, you add client ID enforcement to the API specification. You will:

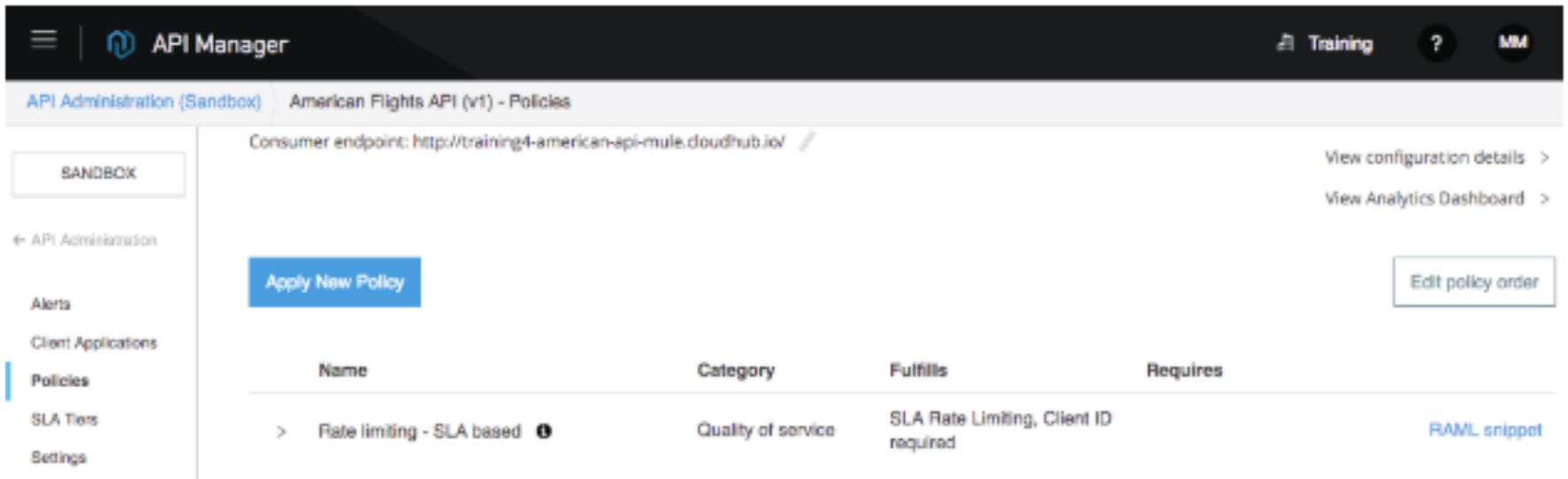
- Modify an API specification to require client id and client secret headers with requests.
- Update a managed API to use a new version of an API specification.
- Call a governed API with client credentials from API portals.

  Note: If you do not complete this exercise for Fundamentals, the REST connector that is created for the API and that you use later in the course will not have client_id authentication.



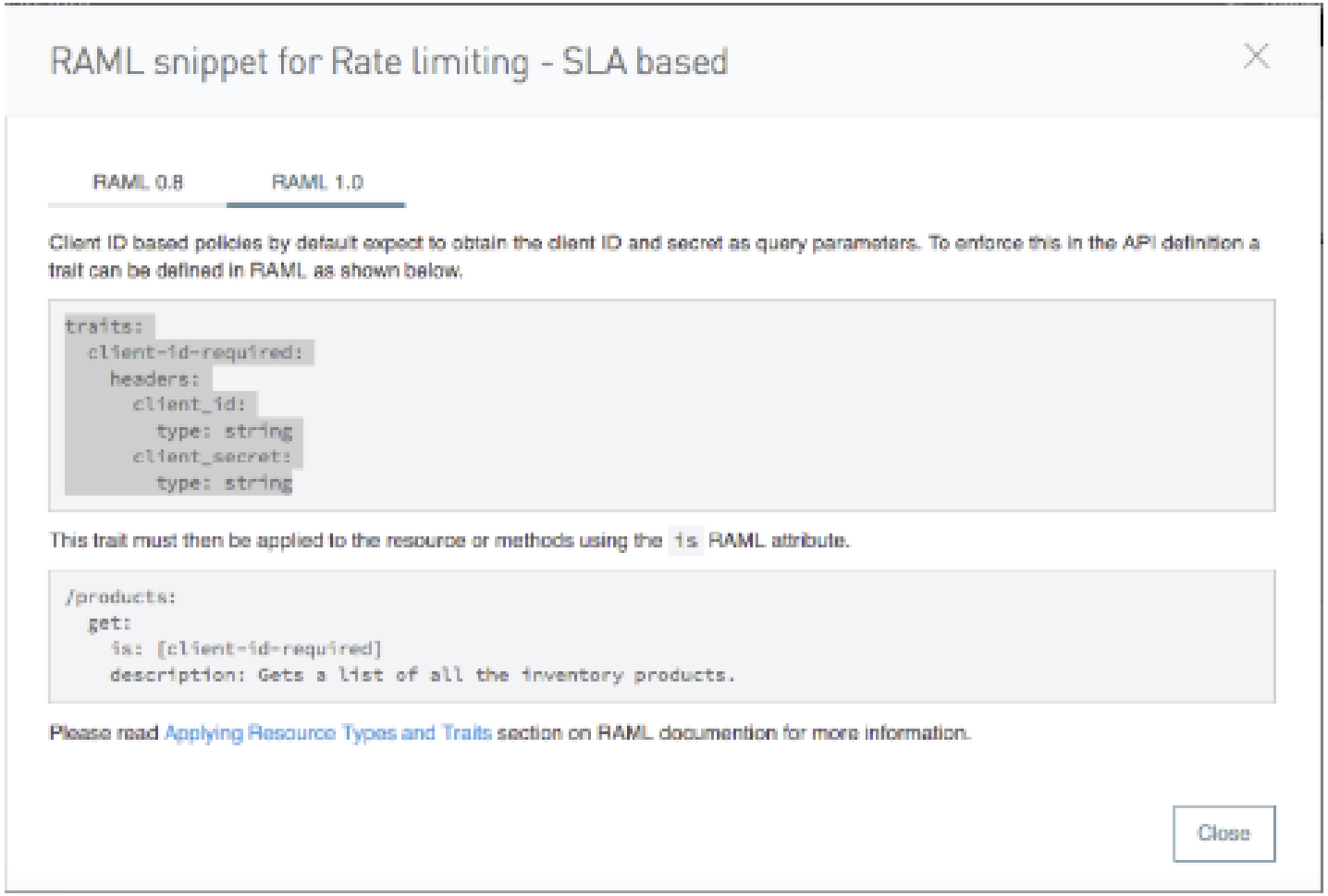
Copy the traits required to add authentication to the API specification

1. Return to the browser tab with the Settings page for American Flights API (v1) in API Manager.
2. In the left-side navigation, select Policies.
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.



6. Click Close.

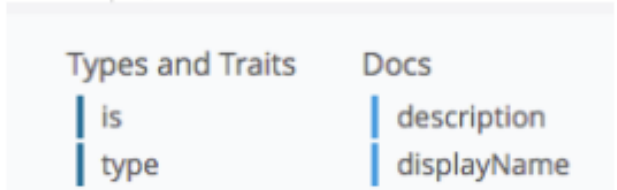
Add authentication headers to the API specification

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.

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

types:
  AmericanFlight: !include exchange_modules

traits:
  client-id-required:
    headers:
      client_id:
        type: string
      client_secret:
        type: string

/flights:
```

9. Go to a new line after the /flights resource declaration and indent.
10. In the shelf, select is.

```
/flights:

  get:
```



11. Add empty array brackets.

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

12. Make sure the cursor is inside the brackets and in the shelf, select client-id-required.

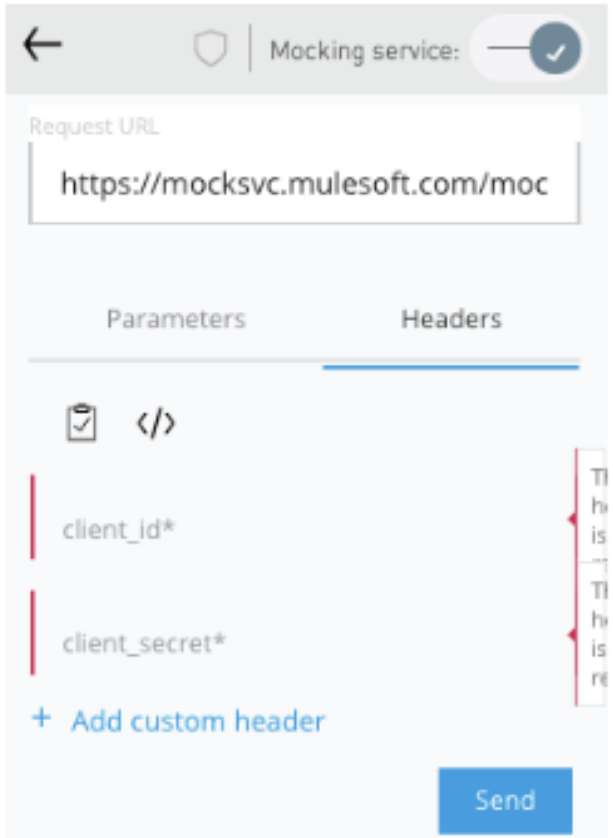
```
16    /flights:
17      is: [client-id-required]
18      get:
```

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

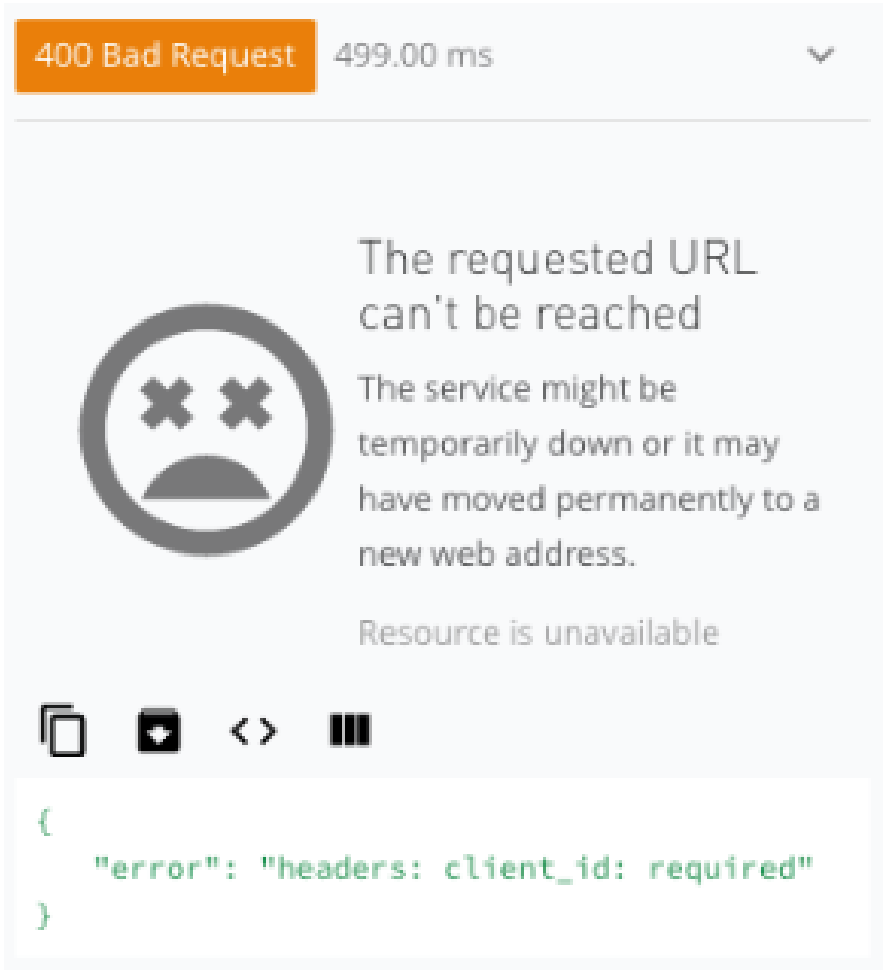
```
44    /{ID}:
45      is: [client-id-required]
46      get:
```

Test the API in the API console in Design Center

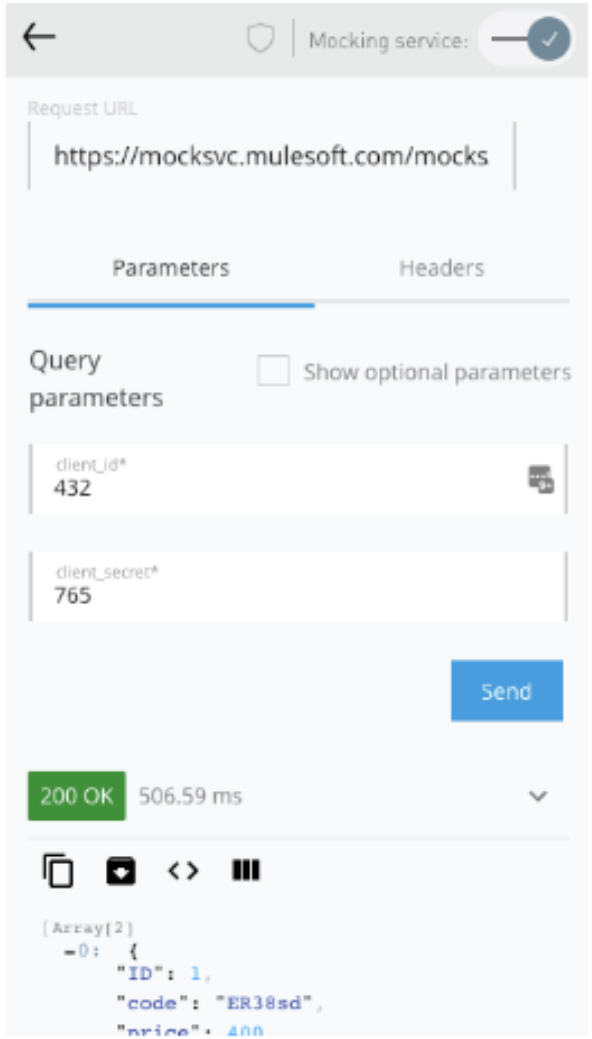
14. In the API console, turn on the mocking service.
15. Select one of the resources and click Try it.
16. Select the Headers tab; you should now see fields to enter client_id and client_secret.



17. Click Send; you should get a 400 Bad Request response with a message that a client_id h

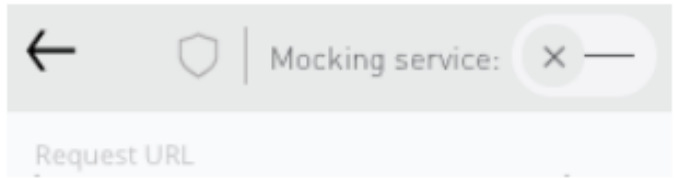


18. Enter *any* values for the client_id and client_secret and click Send; you should get a 200 response with the example results.

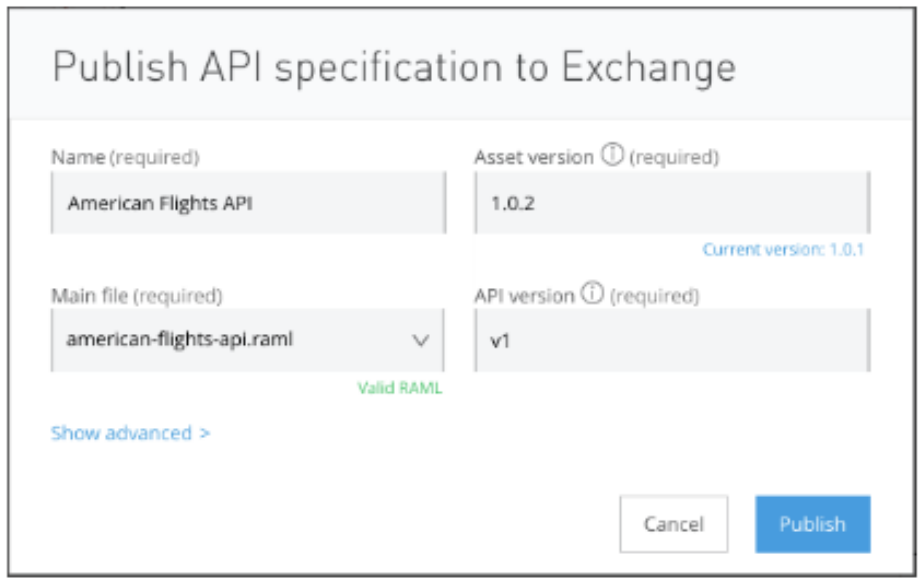


Publish the new version of the API to Exchange

19. Turn off the mocking service.



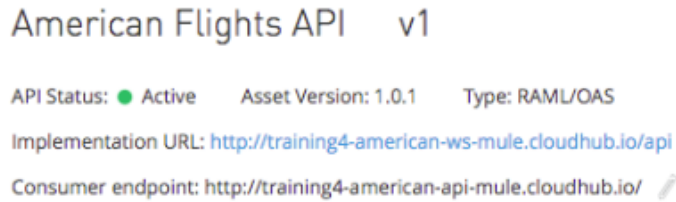
20. Click the Publish to Exchange button.
21. In the Publish API specification to Exchange dialog box, note the asset version and click Publish.



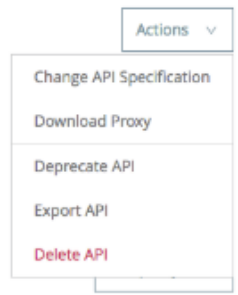
22. After the API is published, click Done in the Publish API specification to Exchange dialog box.

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

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



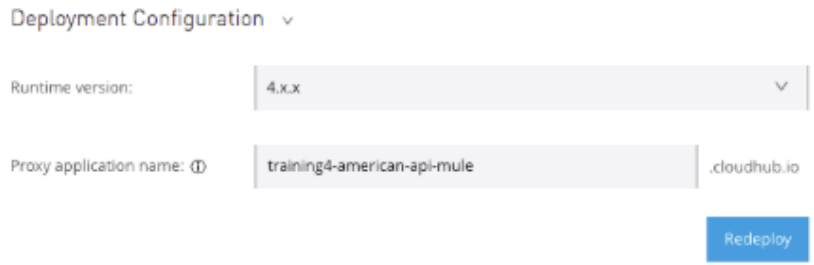
25. Click the Actions button in the upper-right corner and select Change API Specification.



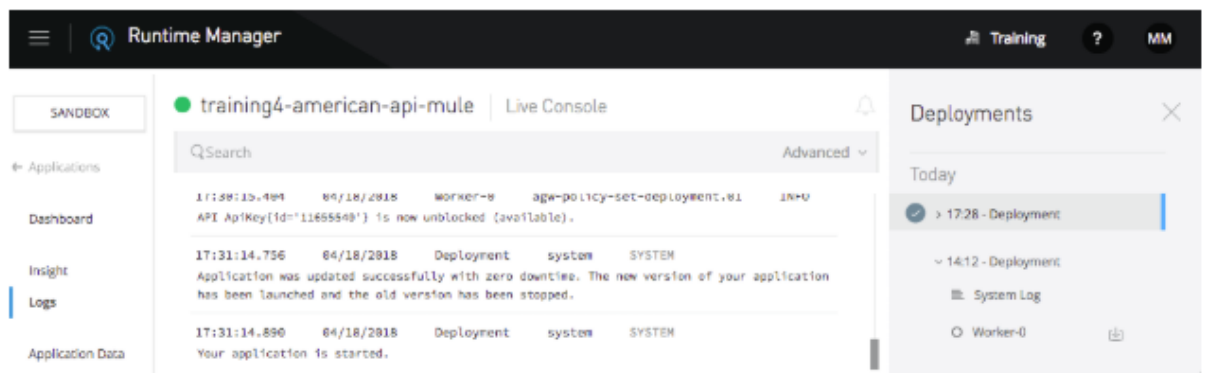
26. In the Change Specification dialog box, select the latest asset version, 1.0.2.
27. Click Change.

Redeploy a new proxy

28. In the left-side navigation, select Settings.
29. Scroll down to the Deployment Configuration settings; the Redeploy button should be disabled.
30. For the runtime version, select 4.x.x again; the Redeploy button should now be enabled.



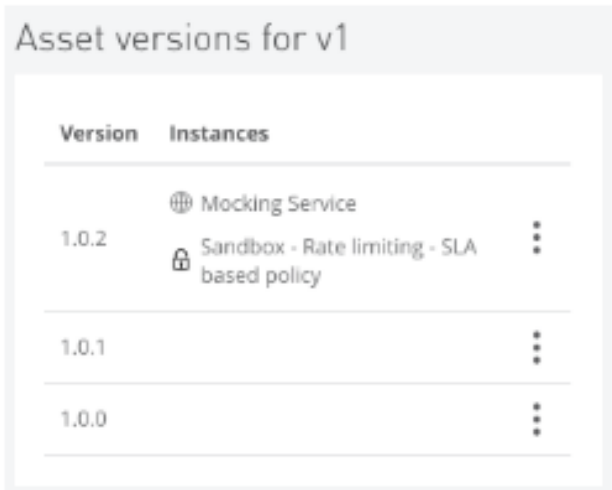
31. Click Redeploy.
32. In the Deploying to CloudHub dialog box, click the Click here link to see the logs.
33. Watch the logs and wait until the proxy application is redeployed.



34. Close the browser tab.
35. Return to the browser tab with API Manager and click Close in the Deploying to CloudHub dialog box.

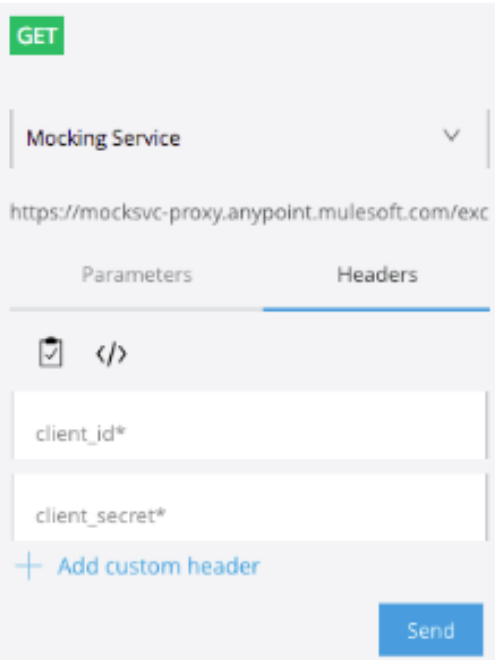
Test the rate limiting – SLA based policy in the API console in Exchange

36. Return to the browser tab with Exchange.
37. Return to the home page for the API (and refresh if necessary); you should see the new asset version listed.



38. Click the GET method for the flights resource and select the Headers tab; you should see required text fields for client_id and client_secret and no longer need to add the headers manually for each request.

    Note: You will test and use the authentication with the REST connector later in the Fundamentals course.



39. Close all Anypoint Platform browser windows and tabs.

Summary

Summary

- Deploy applications to MuleSoft-hosted or customer-hosted Mule runtimes
- **CloudHub** is the Platform as a Service (PaaS) component of Anypoint Platform
  - Hosted Mule runtimes (workers) on AWS
- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway runtime** controls access to APIs by enforcing policies
  - Is part of the Mule runtime but requires a separate license

Summary

- Use **API Manager** to
  - Create and deploy API proxies
  - Define SLA tiers and apply runtime policies
    - Anypoint Platform has out-of-the box policies for rate-limiting, throttling, security enforcement, and more
    - SLA tiers defines # of requests that can be made per time to an API
  - Approve, reject, or revoke access to APIs by clients
  - Promote managed APIs between environments
  - Review API analytics

Anypoint Platform Operations training courses

- This module was just an introduction to deploying and managing applications and APIs

- Anypoint Platform Operations:
  - CloudHub
  - Customer-Hosted Runtimes
  - API Management

DIY Exercise 5-1: Deploy and secure an API

Time estimate: 2 hours

Objectives

In this exercise, you deploy and secure an API implementation. You will:

- Deploy a Mule API application to CloudHub.
- Create simple test flows to automatically set sample required headers and parameters before calling an API endpoint.
- Create an API proxy for a Mule application and apply an SLA based rate limiting policy.

Scenario

The Accounts API has been implemented as a Mule application and you are now ready to deploy it to CloudHub. After deploying the Mule application, secure it by creating an API proxy for it and applying an SLA based rate limiting policy.

Import the starting project

Use your solution from exercise 4-1 or import /files/module04/accounts-mod04-api-implementation-solution.jar (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources) into Anypoint Studio.

Create API client test flows

Create test flows that proxy and test the accounts-api.raml endpoints so that each API endpoint can be tested by making simple GET requests to TCP port 9090. Add a new HTTP Listener global configuration for all the test flows to listen on HTTP port 9090.

The network flow should look like this:
Web client -----9090-----> Test HTTP Listener -----8081-----> API

For each flow, set example values for any required parameters and headers according to the accounts-api.raml specification in the project's src/main/api folder. In order to trigger each test flow, use one shared HTTP Listener configuration, listening on port 9090.

Note: This is an alternate way to test your API without using the mocking service or a separate REST client.

Test direct access to the API endpoints

Debug your Mule application in Anypoint Studio. Use the APIkit Console to make requests to http://localhost:8081/api/accounts. Be sure to set the required headers and query parameters.

Test access to the API endpoints via the test endpoints

Use a web client to make requests to your test flow URLs at http://localhost:9090/{yourTestEndpoint}.

Deploy the Mule application to CloudHub and test the API

In order to reduce the file size to be as small as possible, export the JAR file with only the option: Include project modules and dependencies. Deploy the Mule application to your Anypoint Platform account.

Try to test the flow endpoints (URIs) by making requests to http://appname.cloudhub.io:9090/{yourTestEndpoint}.

Answer the following questions

- What happens when you try to access your test flow endpoints on port 9090?
- What additional steps might you need to take to make your test endpoints available in CloudHub?
- What happens when you try to access your test flow endpoints through the CloudHub elastic load balancer via its public URL http://*appname*.cloudhub.io/{yourTestEndpoint}?
- What happens when you try to access your test flow endpoints via the direct Mule worker URL: http://mule-worker-*appname*.cloudhub.io:9090/{yourTestEndpoint}?
- What happens when you try to access your test flow endpoints via the direct Mule worker URL: http://mule-worker-*appname*.cloudhub.io:8081/{yourTestEndpoint}?

Test the API directly

Try to access the various API endpoints (including the correct URI parameters, query parameters, and headers) at various baseUris.

Answer the following questions

- What happens when you try to access the API:
  - Through the external elastic load balancer URL at http://*appname*.cloudhub.io/api?
  - Through the external elastic load balancer URL at http://*appname*.cloudhub.io:8081/api?
  - Directly at a Mule worker, bypassing the external elastic load balancer, at http://mule-worker-*appname*.cloudhub.io/api?
  - Directly at a Mule worker, bypassing the external elastic load balancer, at http://mule-worker-*appname*.cloudhub.io:8081/api?

Note: To learn more about using CloudHub, take the Anypoint Platform Operations: CloudHub course.

Apply a rate limiting SLA based policy to the API implementation

Now that the Mule application is deployed and tested, secure the API by creating an API proxy for it and then apply a rate limiting SLA based policy with the following SLA levels:

- Free (with automatic approval)
- Professional (with manual approval)
- Enterprise (with manual approval)

Verify your solution

Load the solution /files/module05/accounts-mod05-deploy-solution.jar (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources)
and compare your solution.

DIY Exercise 5-2: Use an API implementation as its own proxy

Time estimate: 1 hour

Objectives

In this exercise, you configure a Mule application that has already deployed to CloudHub to manage its own API proxies. You will:

- Configure a Mule application to enable autodiscovery for API Manager.
- Redeploy the Mule application to CloudHub.
- Test that the implementation URL is now enforcing API policies set in API Manager.

Scenario

The Accounts API has been implemented and deployed to CloudHub. Central IT has notified you that they need more vCores allocated to another project and you must shut down the API proxy that you just deployed in exercise 5-1.

In order to continue to enforce the current API policies, you now need to configure your implementation to enforce the API policy set in API Manager. This is referred to as configuring a basic endpoint as a managed API instance and can be achieved through enabling the Mule application (the API implementation) to use autodiscovery.

Note: To complete this exercise, you must have completed exercise 5-1.

Enable autodiscovery for the Mule application

Use your solution from exercise 5-1. In Anypoint Studio, configure your Mule application to include autodiscovery. Refer to the autodiscovery configuration documentation here: https://docs.mulesoft.com/api-manager/v/2.x/api-auto-discovery-new-concept.

Modify the RAML spec in your Mule application to support SLA policies.

Redeploy the Mule application

Redeploy the Mule application and ensure that the Anypoint Organization client_id and client_secret are passed to the Mule application.

Reconfigure the API in API Manager

Return to the API in API Manager and change the Managing type to Basic Endpoint.

Test the API implementation

Verify that the endpoints are now managed by the implementation by using a web client to make multiple requests to the implementation URL until the SLA limit is exceeded.

Note: Remember to also send values for the type query parameter and the Requester-ID header with the requests.

Verify your solution

Load the solution /files/module05/accounts-mod05-deploy-autodiscovery-solution.jar (in the MUFundamentals_DIYexercises.zip that you can download from the Course Resources)
and compare your solution.