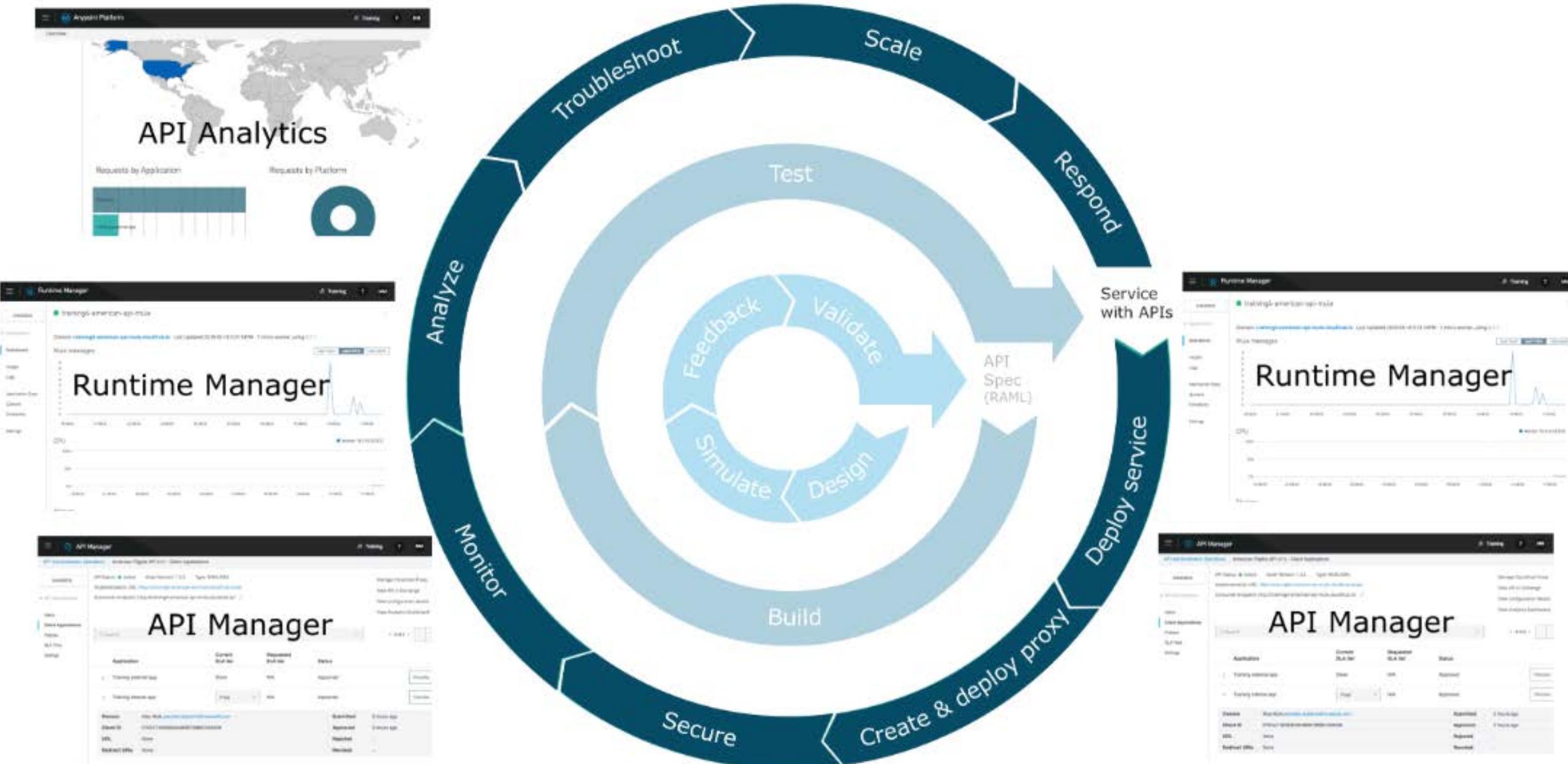




Module 5: Deploying and Managing APIs



Goal



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



- 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



MuleSoft-hosted runtime

- **Customer-hosted Mule runtimes**

- On bare metal or cloud service providers: AWS, Azure, and Pivotal Cloud Foundry



Customer-hosted runtime

- 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



- 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



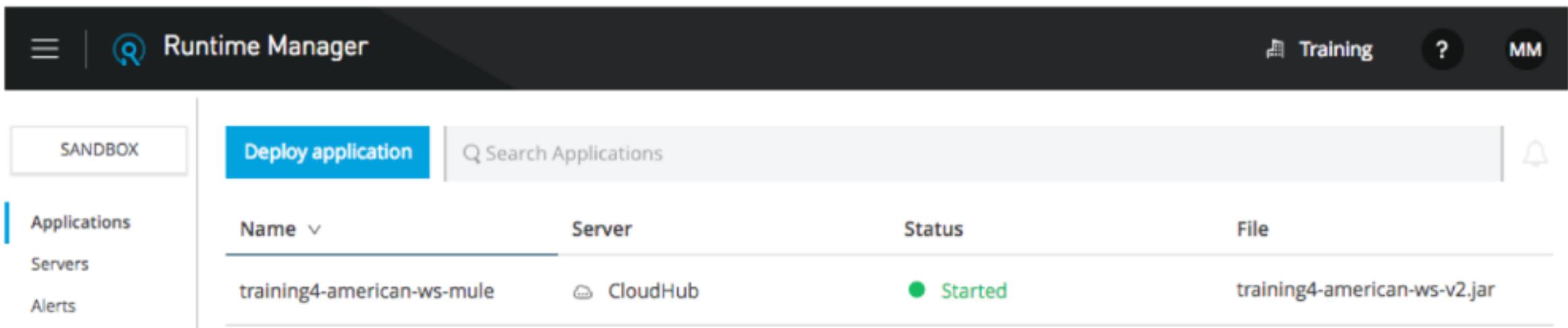
- Can deploy from Anypoint Studio or from Anypoint Platform using Runtime Manager
- You must set worker size and number
 - For apps deployed from flow designer, these values were set automatically

The screenshot shows the MuleSoft Runtime Manager interface. At the top, there's a navigation bar with 'Runtime Manager' and other options like 'Training', '?', and 'MM'. On the left, a sidebar has 'Sandbox' selected, along with links for 'Applications', 'Dashboard', 'Insight', 'Logs', 'Application Data', 'Queues', 'Schedules', and 'Settings'. The main area displays an application named 'training4-american-ws-mule'. It shows the 'Application File' as 'training4-american-ws-v2.jar', with options to 'Choose file' or 'Get from sandbox' and a 'Stop' button. Below this, under the 'Runtime' tab, it lists 'Runtime version' as '4.1.1', 'Worker size' as '0.1 vCores', and 'Workers' as '1'. There are dropdown menus for both 'Worker size' and 'Workers'. Under 'Runtime', there's a checked checkbox for 'Automatically restart application when not res...' and two unchecked checkboxes for 'Persistent queues' and 'Encrypt persistent'. At the bottom right, there's a 'Apply Changes' button.

- 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

| Worker size |
|-----------------------------|
| 0.1 vCores |
| 0.1 vCores 500 MB memory |
| 0.2 vCores 1 GB memory |
| 1 vCore 1.5 GB memory |
| 2 vCores 3.5 GB memory |
| 4 vCores |

- 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



The screenshot shows the MuleSoft Runtime Manager interface. At the top, there's a navigation bar with icons for Training, Help, and User Profile. On the left, a sidebar has links for SANDBOX, Applications (which is selected), Servers, and Alerts. The main area has tabs for Deploy application (selected) and Search Applications. A table lists deployed applications with columns for Name, Server, Status, and File. One application, "training4-american-ws-mule", is listed with "CloudHub" as the server, "Started" status, and "training4-american-ws-v2.jar" as the file.

| Name | Server | Status | File |
|----------------------------|----------|---------|------------------------------|
| training4-american-ws-mule | CloudHub | Started | training4-american-ws-v2.jar |

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.

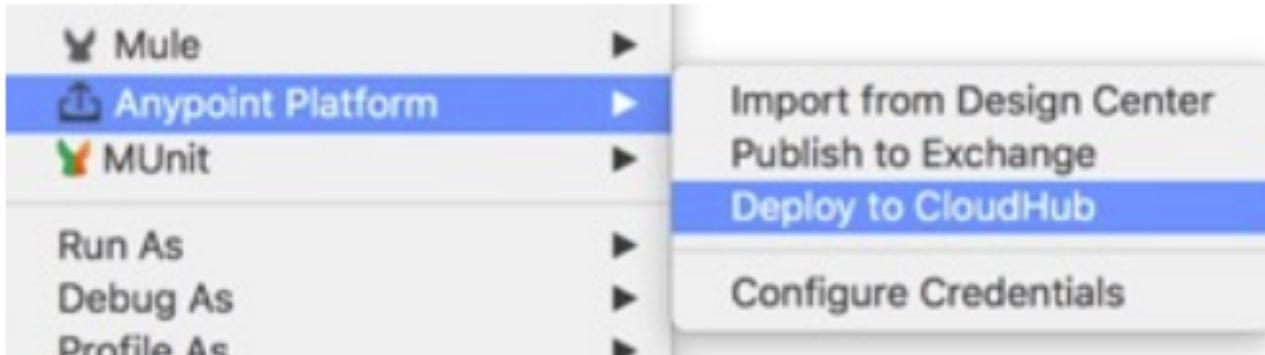
The screenshot shows the Anypoint Platform Runtime Manager. At the top, there's a navigation bar with icons for Training, Help, and Metrics (MM). Below the bar, a sidebar on the left has tabs for SANDBOX, Applications (which is selected), Servers, and Alerts. The main area has tabs for Deploy application (selected) and Search Applications. A table lists deployed applications with columns for Name, Server, Status, and File. One application, "training4-american-ws-mule", is listed with "CloudHub" as the server, "Started" status, and "training4-american-ws-v2.jar" as the file. There's also a small bell icon in the top right corner of the main area.

| Name | Server | Status | File |
|----------------------------|----------|---------|------------------------------|
| training4-american-ws-mule | CloudHub | Started | training4-american-ws-v2.jar |

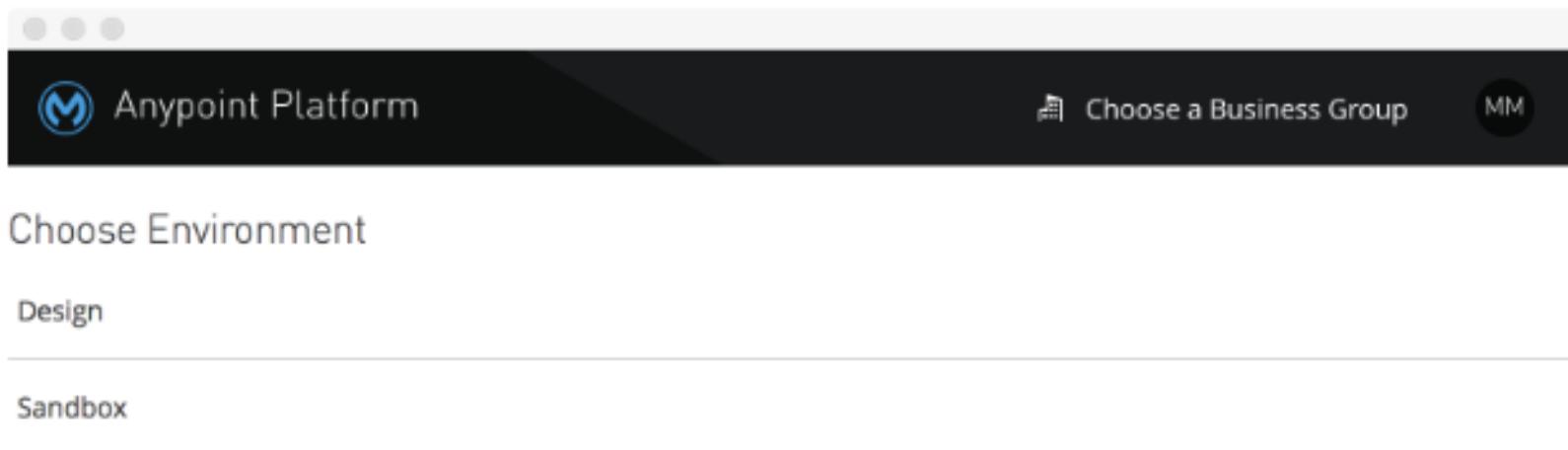
Note: If you do not have a working application at this point, import the training4-american-ws-wt4-6.jar solution into Anypoint Studio and work with that project.

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.

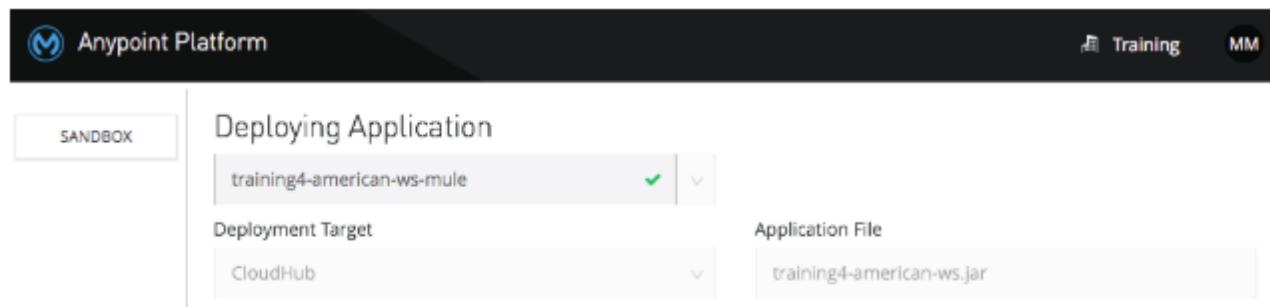


3. In the Choose Environment dialog box, select Sandbox.



- 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.



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

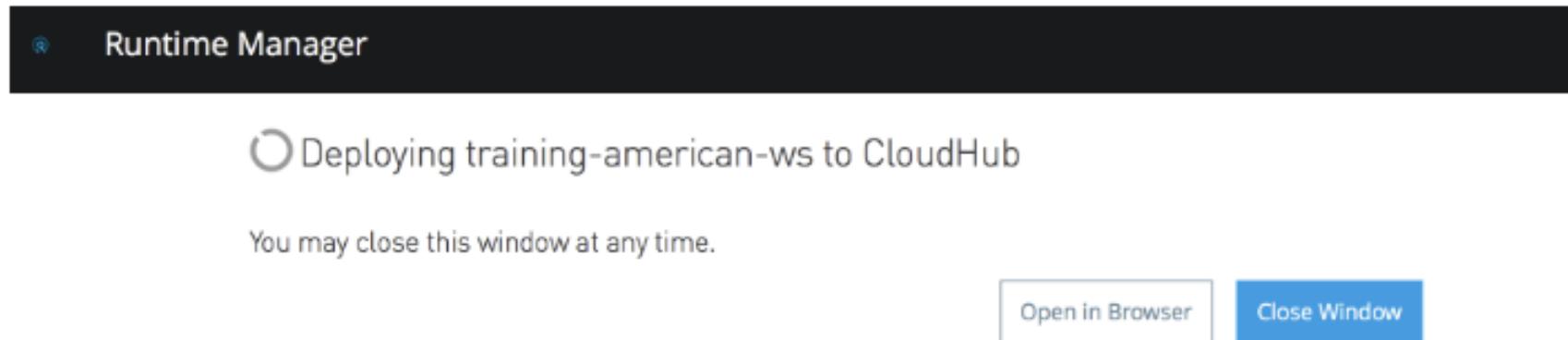
Note: If you don't know what version it is using, look at the Package Explorer and find a library folder with the name of the server being used, like Mule Server 4.1.1 EE.

- Make sure the worker size to 0.1 vCores.

| Runtime | Properties | Insight | Logging | Static IPs |
|--------------------------|---------------------------|--------------|---------|------------|
| Runtime version 4.1.3 | Worker size 0.1 vCores | Workers 1 | | |

- 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.

A screenshot of the Anypoint Platform Runtime Manager dashboard. The top navigation bar includes "Sandbox", "Runtime Manager" (selected), "Training", a help icon, and a user icon. On the left, a sidebar has "Sandbox" selected, and "Applications", "Servers", and "Alerts" are listed. The main area shows a table with columns: Name, Server, Status, and File. One row is visible: "training4-american-ws-mule" is deployed to "CloudHub" with status "Deploying" and file "training4-american-ws-1.0.0-SNAPSHOT-mule-application.jar".

| Name | Server | Status | File |
|----------------------------|----------|-----------|---|
| training4-american-ws-mule | CloudHub | Deploying | training4-american-ws-1.0.0-SNAPSHOT-mule-application.jar |

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.

The screenshot shows the Runtime Manager interface. On the left, there's a sidebar with options: SANDBOX, Applications (which is selected), Servers, Alerts, VPCs, and Load Balancers. The main area has tabs for Deploy application and Search Applications. A table lists applications by Name, Server, Status, and File. One row is selected: "training4-american-ws-mu" running on "CloudHub" with a green "Started" status. To the right, a detailed view of the application "training4-american-ws-mule" is shown. It includes a file input field ("Choose file..."), last update information (2018-04-18 1:48:21PM), app URL (training4-american-ws-mule.cloudhub.io), runtime version (4.1.1), worker size (0.1 vCores), and workers (1). Buttons for Manage Application, Logs, and Insight are at the bottom, along with a link to View Associated Alerts.

11. Click the Logs button.

12. Watch the logs as the application is deployed.

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

The screenshot shows the Mule Runtime Manager interface. On the left, a sidebar lists navigation options: SANDBOX, Applications, Dashboard, Insight, **Logs**, Application Data, Queues, Schedules, and Settings. The main area displays the logs for the application "training4-american-ws-mule". The logs show the following output:

```
Starting Bean: listener
13:48:21.870 04/18/2018 Worker-0 qtp1298881551-39 INFO
*****
* Application: training4-american-ws-mule
* OS encoding: UTF-8, Mule encoding: UTF-8
*
*****
13:48:21.955 04/18/2018 Deployment system SYSTEM
Worker(18.217.62.85): Your application has started successfully.
13:48:22.622 04/18/2018 Deployment system SYSTEM
Your application is started.
```

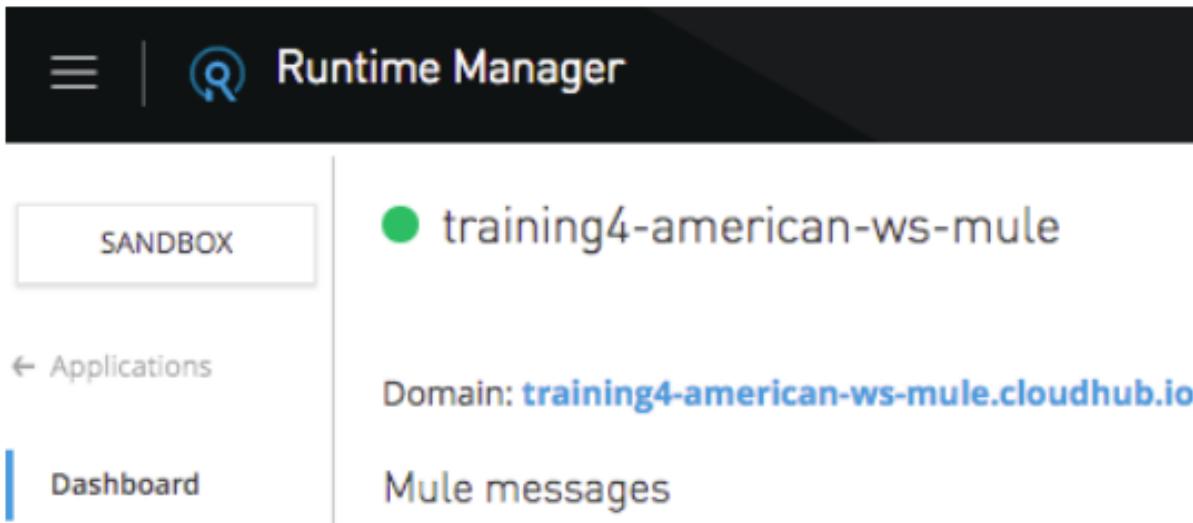
To the right, a "Deployments" panel is open, showing a deployment log for "13:46 - Deployment" on "Worker-0". The log entry is:

```
13:46 - Deployment
System Log
Worker-0
```

Note: If your application did not successfully deploy, read the logs to help figure out why the application did not start. If 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}.cloudbhub.io`.

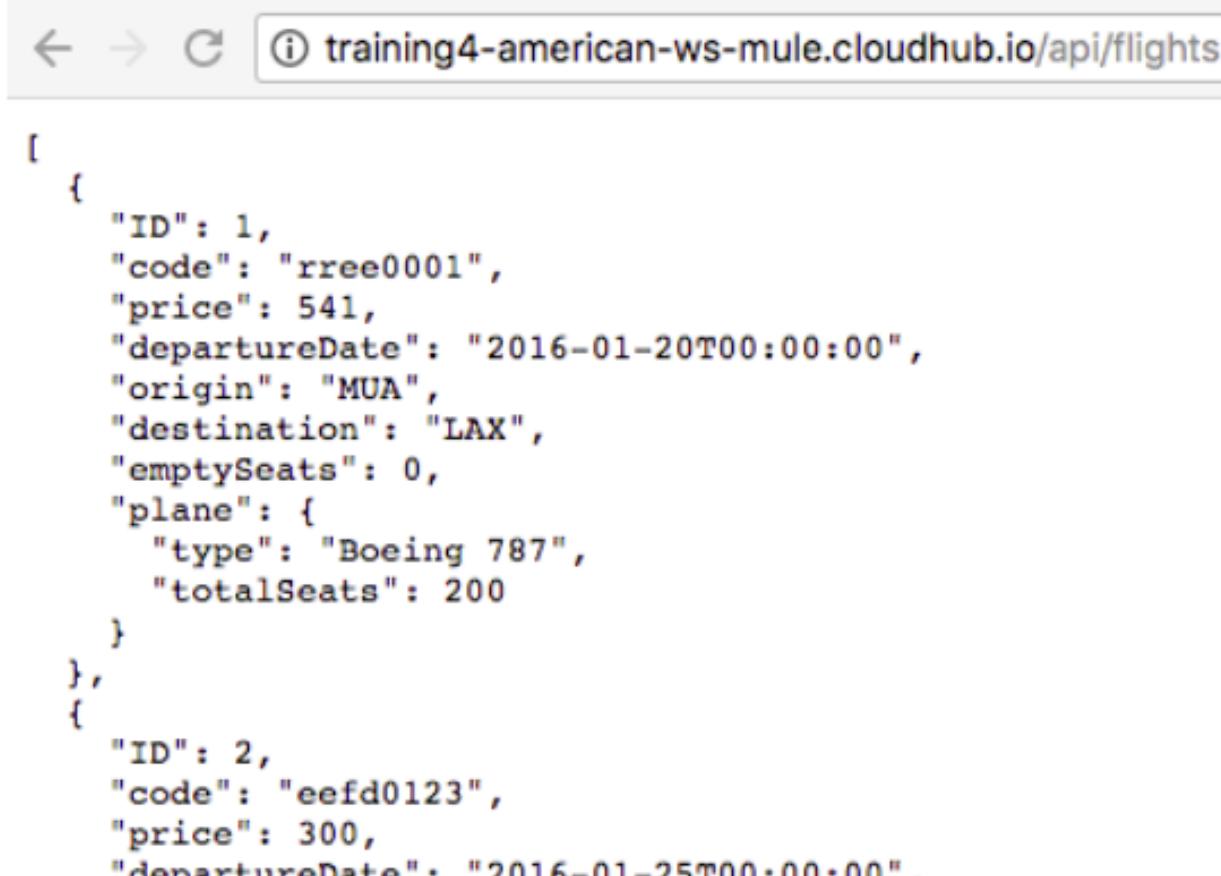


The screenshot shows the Runtime Manager interface. At the top, there's a dark header bar with three horizontal lines on the left, a magnifying glass icon, and the text "Runtime Manager". Below this is a navigation bar with three items: "SANDBOX" (highlighted with a light gray border), "Applications" (with a left arrow icon), and "Dashboard" (highlighted with a blue vertical bar). To the right of the navigation bar, the application "training4-american-ws-mule" is listed with a green circular icon. Below the application name, the text "Domain: training4-american-ws-mule.cloudbhub.io" is displayed. At the bottom of the dashboard, there are two links: "Mule messages" and "Mule metrics".

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-{/astname}.{region}.cloudbu.io/api/flights`; you should see the flights data.



A screenshot of a web browser window. The address bar shows the URL `training4-american-ws-mule.cloudbu.io/api/flights`. The page content displays a JSON array of flight data:

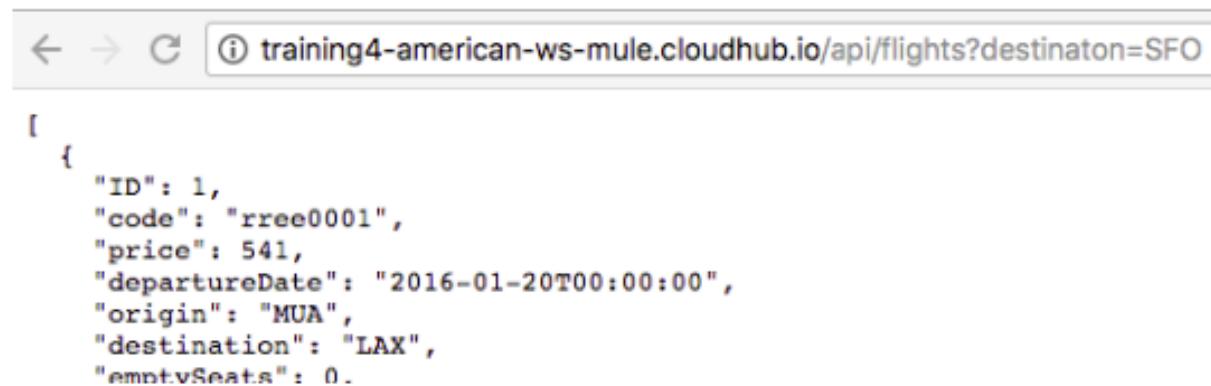
```
[  
  {  
    "ID": 1,  
    "code": "rree0001",  
    "price": 541,  
    "departureDate": "2016-01-20T00:00:00",  
    "origin": "MUA",  
    "destination": "LAX",  
    "emptySeats": 0,  
    "plane": {  
      "type": "Boeing 787",  
      "totalSeats": 200  
    }  
  },  
  {  
    "ID": 2,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



A screenshot of a web browser window. The address bar shows the URL: "training4-american-ws-mule.cloudhub.io/api/flights?destination=SFO". The page content displays a JSON array with one element:

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

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.

The screenshot shows the 'Runtime Manager' interface for an application named 'training4-american-ws-mule'. The left sidebar includes links for 'Sandbox', 'Applications', 'Dashboard', 'Insight', 'Logs', 'Application Data', 'Queues', 'Schedules', and 'Settings'. The 'Settings' link is currently selected. The main panel displays runtime configuration options: 'Runtime version' set to '4.1.1', 'Worker size' set to '0.1 vCores', and 'Workers' set to '1'. A warning message states, '⚠ Your current subscription allows only one worker per application'. Under the 'Properties' tab, there are checkboxes for 'Automatically restart application when not responding' (checked), 'Persistent queues' (unchecked), and 'Encrypt persistent queues' (unchecked). At the bottom right is a large blue 'Apply Changes' button.

SANDBOX

Training ? MM

training4-american-ws-mule

Applications

Dashboard

Insight

Logs

Application Data

Queues

Schedules

Settings

Application File

training4-american-ws-v2.jar Choose file ▾ Get from sandbox Stop ▾

App url: training4-american-ws-mule.cloudhub.io

| Runtime | Properties | Insight | Logging | Static IPs |
|---|---------------------------|--------------|--|------------|
| Runtime version 4.1.1 | Worker size 0.1 vCores | Workers 1 | ⚠ Your current subscription allows only one worker per application | |
| <input checked="" type="checkbox"/> Automatically restart application when not responding | | | | |
| <input type="checkbox"/> Persistent queues <input type="checkbox"/> Encrypt persistent queues | | | | |

Apply Changes

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.



The screenshot shows a browser window with the URL `training-american-ws-mule.cloudhub.io/api/flights?destination=SFO`. The page displays a JSON array containing two flight records. Both flights have a destination of "SFO".

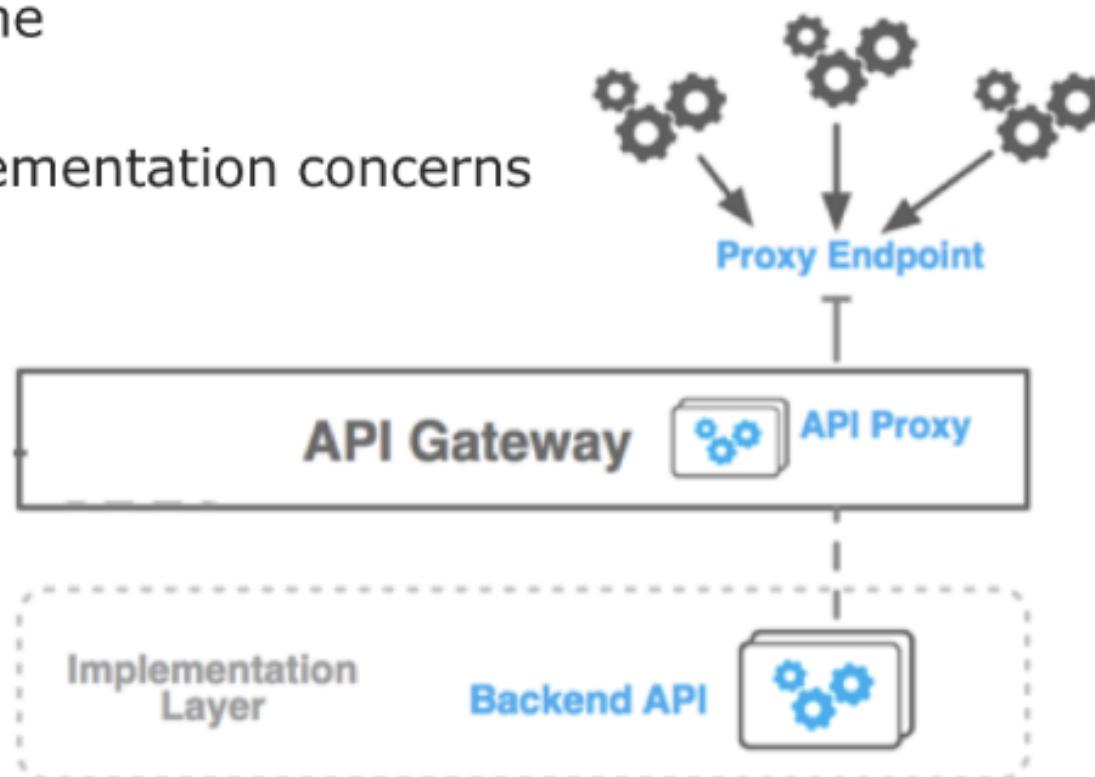
```
[  
  {  
    "ID": 5,  
    "code": "rree1093",  
    "price": 142,  
    "departureDate": "2016-02-11T00:00:00",  
    "origin": "MUA",  
    "destination": "SFO",  
    "emptySeats": 1,  
    "plane": {  
      "type": "Boeing 737",  
      "totalSeats": 150  
    }  
  },  
  {  
    "ID": 7,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

30. Close this browser tab.

Creating API proxies

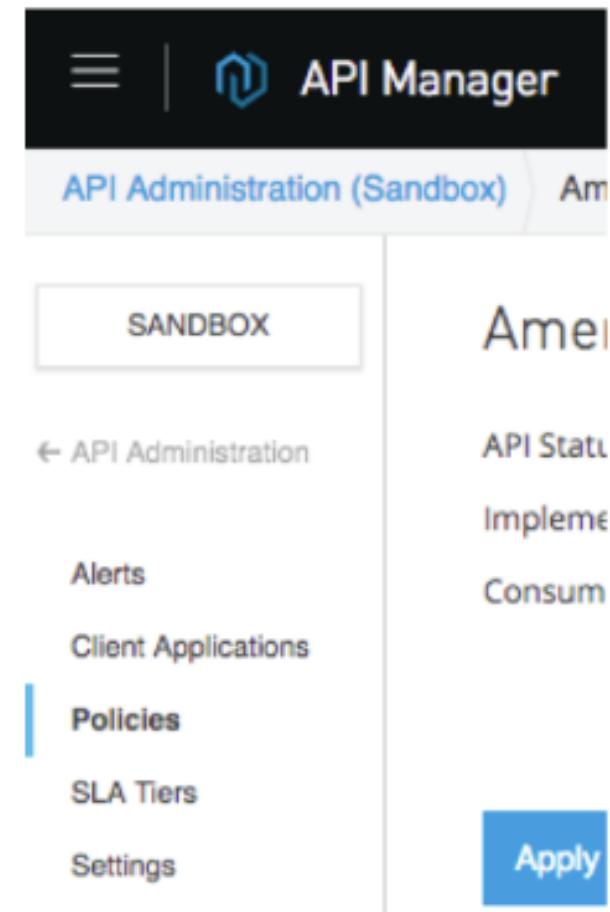


- An **API proxy** is an application that controls access to a web service, restricting access and usage through the use of an API gateway
- The **API Gateway** is a runtime designed and optimized to host an API or to open a connection to an API deployed to another runtime
 - Included as part of the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

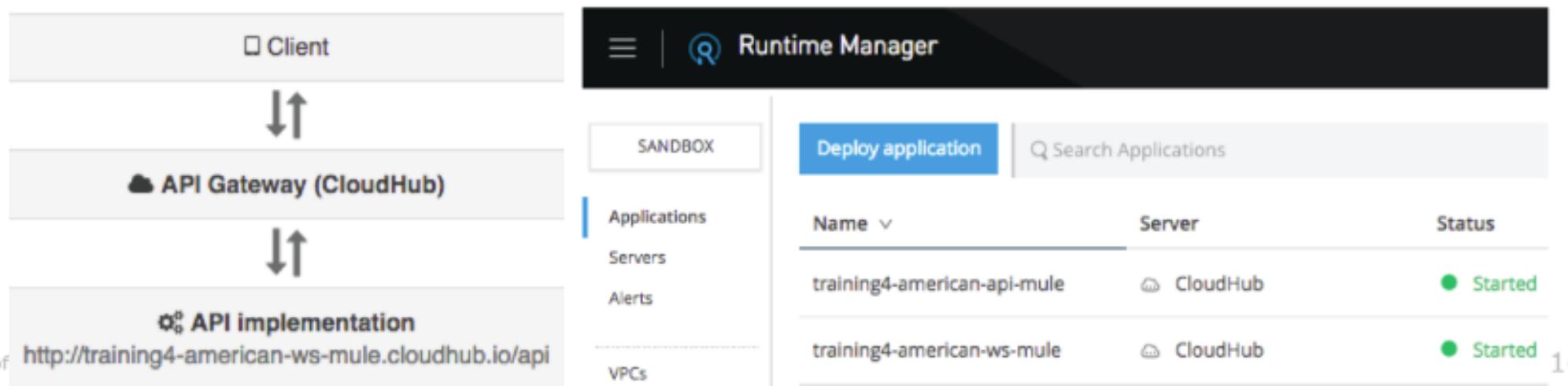


- **Determines which traffic** is authorized to pass through the API to backend services
- **Meters the traffic** flowing through
- **Logs** all transactions, collecting and tracking analytics data
- Applies runtime policies to **enforce governance** like rate limiting, throttling, and caching

- **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**



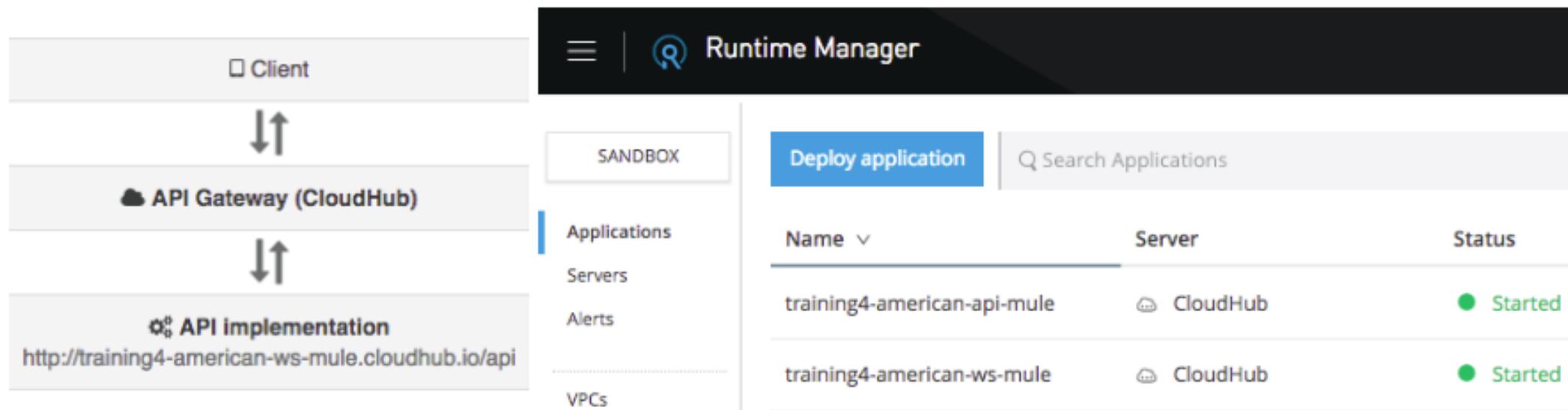
- Add an API to API Manager
- Use API Manager to create and deploy an API proxy application
- Set a proxy consumer endpoint so requests can be made to it from Exchange
- Make calls to an API proxy from API portals for internal & external users
- View API request data in API Manager.



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.

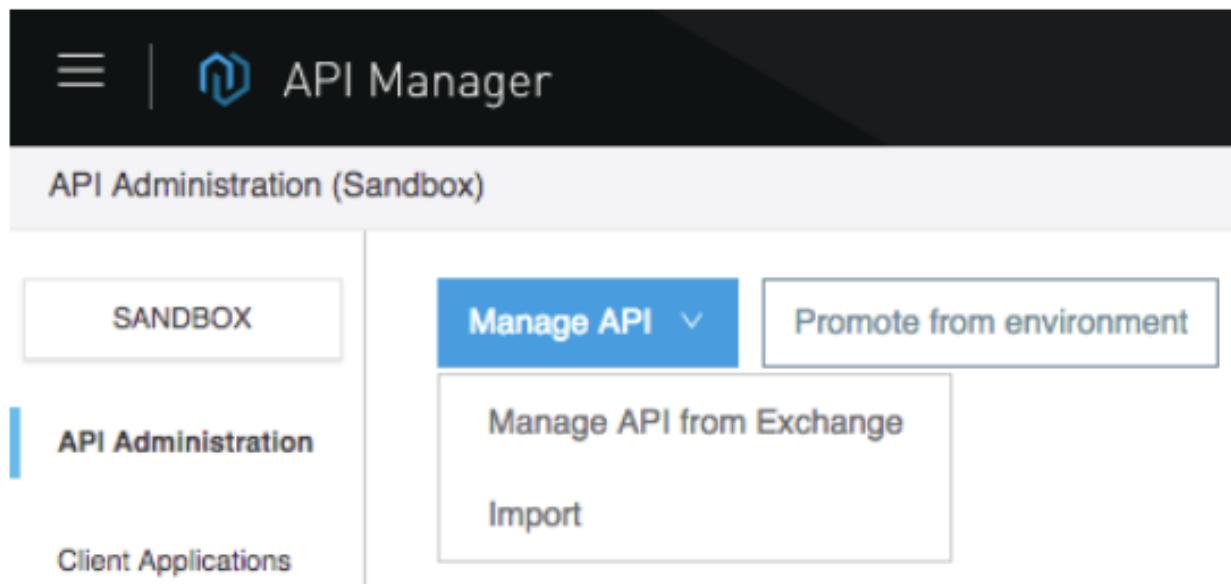


Create and deploy a proxy application

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

The screenshot shows the Anypoint Platform API Manager interface. At the top, there is a dark header bar with the "API Manager" logo and navigation links for "Training", "?", and "MM". Below the header, the title "API Administration (Sandbox)" is displayed. On the left, a sidebar lists "Sandbox", "API Administration" (which is selected and highlighted in blue), "Client Applications", "Custom Policies", and "Analytics". The main content area features a search bar with "Search" and "All" filters. A central message reads "No APIs to display. Get started by adding your first API." To the right, there is a placeholder image of a chart and a text link "Select an API version to see more details".

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-{astname}.cloudbhub.io/api`
- Proxy deployment target: CloudHub

The screenshot shows the Mule API Manager interface. The top navigation bar includes 'Training', a question mark icon, and a 'MM' icon. Below the navigation is a breadcrumb trail: 'API Administration (Sandbox) > Get from Exchange'. A left sidebar has a 'Sandbox' button and a link to 'API Administration'. The main content area is titled 'Manage API from Exchange' and shows the configuration for the 'American Flights API'. The configuration fields are as follows:

- API name: American Flights API
- Asset type: RAML/OAS
- API version: v1
- Asset version: 1.0.1
- Managing type: Endpoint with Proxy (selected)
- Implementation URI: `http://training4-american-ws-mule.cloudbhub.io/api`
- Proxy deployment target: CloudHub (selected)
- Path: /

A checkbox at the bottom states: 'Check this box if you are managing this API in Mule 4 or above.' The checkbox is checked.

≡ | API Manager

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.

Path:

/



Check this box if you are managing this API in Mule 4 or above.

[Advanced options >](#)

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.

Deployment Configuration ▾

Runtime version:

4.x.x

Proxy application name: ⓘ

training4-american-api-mule

.cloudhub.io

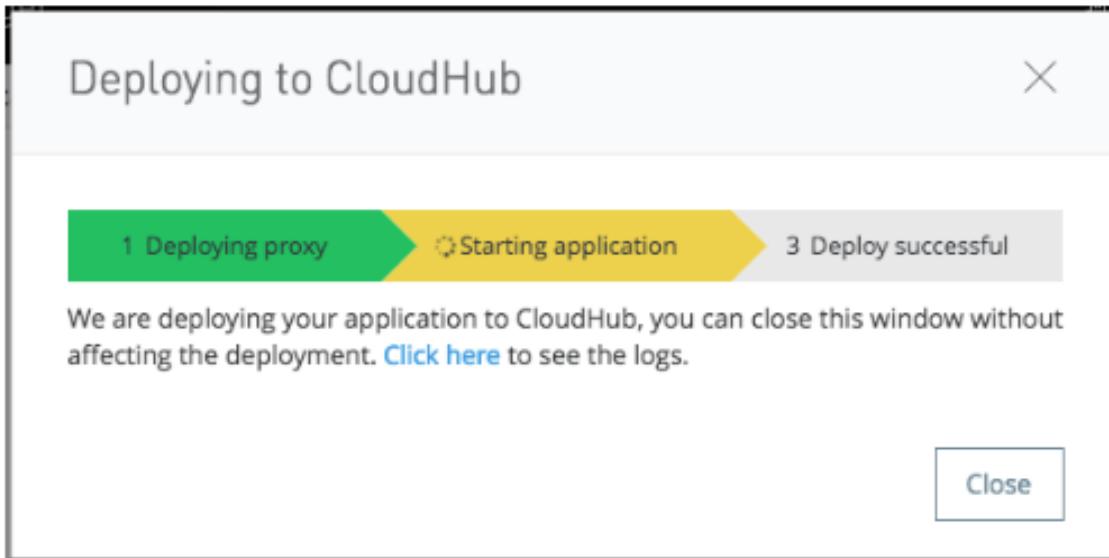


Update application if exists

Deploy

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.

A screenshot of the "Runtime Manager" interface. The top navigation bar includes "Training", a help icon, and a user profile icon. On the left, a sidebar shows "Sandbox" selected, along with "Applications" and "Dashboard". The main area displays an application named "training4-american-api-mule" with a "Live Console" tab. The console shows log entries: "14:12:08.656 04/18/2018 Deployment system SYSTEM Deploying application to 1 workers.". To the right, a "Deployments" panel is open, showing a single entry for "Today": "14:12 - Deployment".

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.

The screenshot shows the Mule Runtime Manager interface. On the left, there's a navigation sidebar with options: SANDBOX (selected), Applications, Dashboard, Insight, and Logs. The main area displays an application named "training4-american-api-mule" with a "Live Console" tab. The console shows the following log entries:

- API ApKey(id='11655548') is now unblocked (available).
- 14:13:27.252 04/18/2018 Deployment system SYSTEM Worker(18.219.16.104): Your application has started successfully.
- 14:13:27.902 04/18/2018 Deployment system SYSTEM Your application is started.

A "Deployments" modal is open on the right, showing a list of deployments for today, with one entry from 14:12.

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.

The screenshot shows the Runtime Manager interface. On the left, there's a sidebar with options: SANDBOX (selected), Applications (highlighted in blue), Servers, Alerts, VPCs, and Load Balancers. The main area has tabs for Deploy application and Search Applications. A table lists two applications:

| Name | Server | Status | File |
|-----------------------------|----------|---------|---------------------------------|
| training4-american-api-mule | CloudHub | Started | training4-american-api-mule.zip |
| training4-american-ws-mule | CloudHub | Started | training4-american-ws-v2.zip |

A modal window is open for the application "training4-american-api-mule". It shows the following details:

- Status: Started
- Server: CloudHub
- Last Updated: 2018-04-18 2:13:27PM
- App url: training4-american-api-mule.cloudhub.io
- Runtime version: 4.1.1
- Worker size: 0.1 vCores
- Workers: 1

Buttons at the bottom of the modal include Manage Application, Logs, and Insight.

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.

The screenshot shows the API Manager interface. The top navigation bar includes a menu icon, the 'API Manager' logo, 'Training' with a help icon, and a user icon. Below the header, the left sidebar has a 'Sandbox' button and links for 'API Administration', 'Alerts', 'Client Applications', 'Policies', 'SLA Tiers', and 'Settings'. The 'Settings' link is highlighted with a blue vertical bar. The main content area displays the 'American Flights API v1' settings. It shows the API is Active (green dot), Asset Version is 1.0.1, and Type is RAML/OAS. The Implementation URL is <http://training4-american-ws-mule.cloudhub.io/api>. There is a '+ Add consumer endpoint' button. Below this, the API Instance section shows ID 11655540 and Autodiscovery with API ID 11655540. The SLA Tiers section shows a 'Label' field with '+ Add a label'. Under the 'Proxy' section, it shows the Proxy Application as 'training4-american-api-mule' and the Proxy URL as <http://training4-american-api-mule.cloudhub.io>.

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.

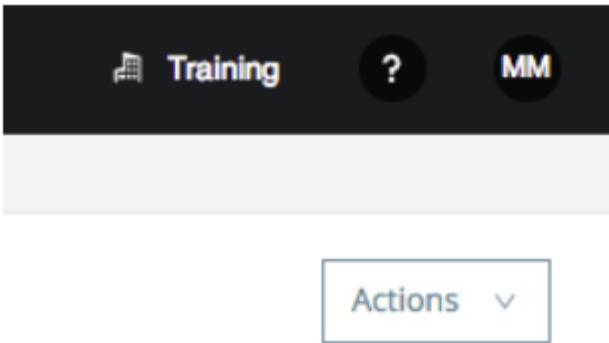
21. Review the API version info that appears on the right side of the window; you should see there are no policies, SLA tiers, or client applications.

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

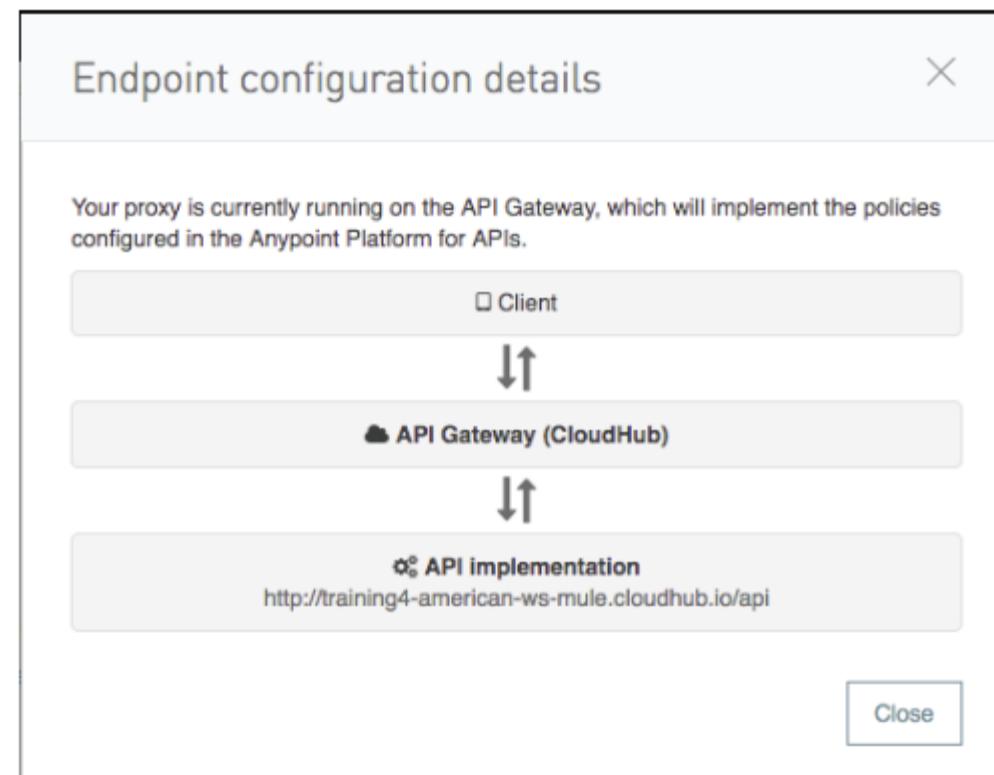
- Header:** API Manager, Training, ?, MM
- Left Sidebar:** SANDBOX, API Administration (selected), Client Applications, Custom Policies, Analytics
- Main Content:**
 - Manage API** dropdown, Promote from environment, Search bar, 1 - 1 of 1 results, All, Favorites, Active filters.
 - Table Headers:** API Name, Version, Status, Client Applications, Creation Date.
 - Table Data:** American Flights API (v1, Active, 0 Client Applications, Creation Date: 04-18-2018 14:11).
- Right Panel:** American Flights API v1 details.
 - Actions: Manage CloudHub Proxy, View API in Exchange, View Analytics Dashboard.
 - Tabs: Applications (selected), Policies, SLA tiers.
 - Note: There are no policies configured for this API version.

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.

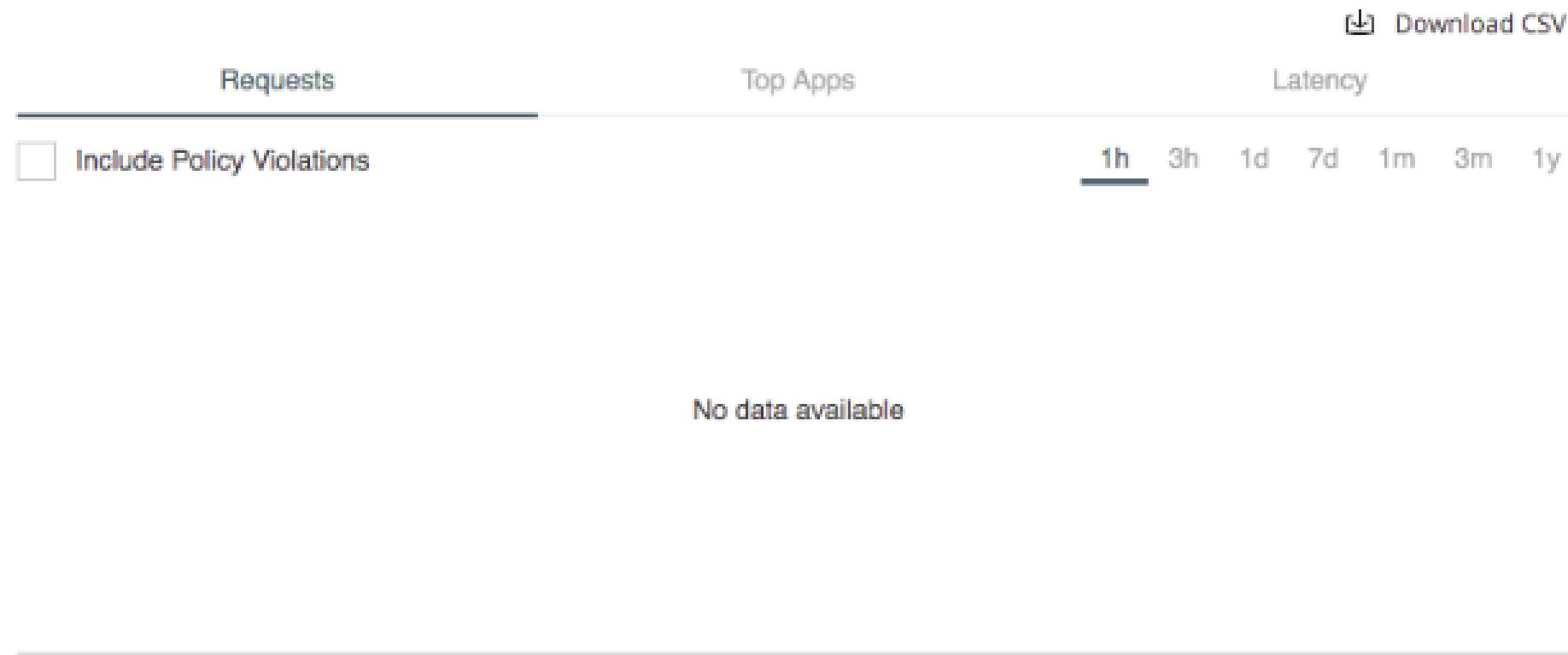


24. Click the View configuration details link.



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.

| Asset versions for v1 | |
|-----------------------|---|
| Version | Instances |
| 1.0.1 |  Mocking Service  Sandbox - v1:5831632 |
| 1.0.0 | |

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.

The screenshot shows the MuleSoft Anypoint Platform interface. The top navigation bar includes 'Training', '?', and 'MM' buttons. The left sidebar has 'Assets list' and 'American Flights API' sections, with 'API summary' expanded, showing 'Types' (selected), 'Resources', '/flights' (selected), 'GET' (highlighted), 'POST', and '/{ID}'. The main content area displays the 'American Flights API' details (version v1) and its 'API instances' table. The table has columns: Instances, Environment, URL, and Visibility. It lists two instances: 'Mocking Service' (Environment: Sandbox, URL: https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1, Visibility: Public) and 'v1:5831632' (Environment: Sandbox, URL: https://localhost:5831632, Visibility: Private). A 'Add new instance' button is at the bottom of the table.

| Instances | Environment | URL | Visibility |
|-----------------|-------------|--|------------|
| Mocking Service | Sandbox | https://mocksvc-proxy.anypoint.mulesoft.com/exchange/800b1585-b6be-4c62-82de-02d8c2adf413/american-flights-api/1.0.1 | Public |
| v1:5831632 | Sandbox | https://localhost:5831632 | Private |

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.

API Instance ⓘ

ID: 5831632

Label: No policy 

Set a consumer endpoint for the proxy in API Manager

36. Locate the proxy URL.

Proxy

Proxy Application: training4-american-api-mule

Proxy URL: training4-american-api-mule.cloudhub.io

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

38. Click the Add consumer endpoint link.

American Flights API v1

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

Implementation URL: <http://training4-american-ws-mule.cloudhub.io/api> ⊕ Add consumer endpoint

39. Paste the value of the proxy URL.

40. Press Enter/Return.

American Flights API v1

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

Implementation URL: <http://training4-american-ws-mule.cloudhub.io/api>

Consumer endpoint: <http://training4-american-api-mule.cloudhub.io/> /

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.

The screenshot shows the MuleSoft Anypoint Platform interface. At the top, there's a navigation bar with icons for Training, Help, and MM. On the left, a sidebar menu includes 'Assets list', 'American Flights API' (which is highlighted), 'API summary', 'Types' (with a dropdown for 'Resources'), '/flights' (with a dropdown for 'GET' and 'POST' methods), and '{ID}' (with a dropdown). The main content area is titled 'American Flights API' with a 'v1 Public' status. It shows a table of 'API instances' with columns for 'Instances', 'Environment', 'URL', and 'Visibility'. There are two entries: 'Mocking Service' (Environment: Public, URL: https://mocksvc-proxy.anypoint.mulesoft.com/exchange/74922056-9245-48e0-99df-a8141d1d3e9f/american4-flights-api/1.0.1) and 'No policy' (Environment: Sandbox, URL: http://training4-american-api-mule.cloudhub.io/). A 'Private' visibility option is shown next to the second entry. A blue '+ Add new instance' button is at the bottom of the table.

| Instances | Environment | URL | Visibility |
|-----------------|-------------|---|------------|
| Mocking Service | Public | https://mocksvc-proxy.anypoint.mulesoft.com/exchange/74922056-9245-48e0-99df-a8141d1d3e9f/american4-flights-api/1.0.1 | Public |
| No policy | Sandbox | http://training4-american-api-mule.cloudhub.io/ | Private |

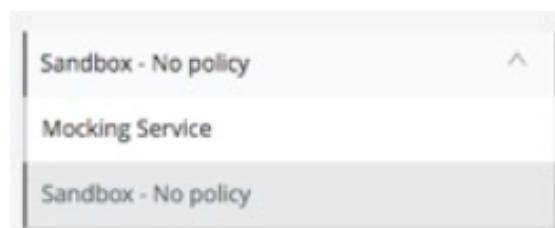
43. In the left-side navigation, select the name of the API to return to its main page.

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

| Asset versions for v1 | |
|-----------------------|--|
| Version | Instances |
| 1.0.1 |  Mocking Service  Sandbox - No policy |
| 1.0.0 | ⋮ |

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.

200 OK 9102.50 ms Details ✓

The screenshot shows a REST API response with a status of "200 OK" and a response time of "9102.50 ms". There is a "Details" link. Below the status, there are several icons: a checkmark, a download arrow, a copy icon, and a refresh/cross icon. The main content area displays an array of flight data. The first item in the array is expanded, showing its properties: ID (4), code (rree1000), price (200), departureDate (2016-01-20T00:00:00), origin (MUA), destination (CLE), emptySeats (5), plane (Boeing 737, totalSeats 150), and ID (5). The array has 11 items in total.

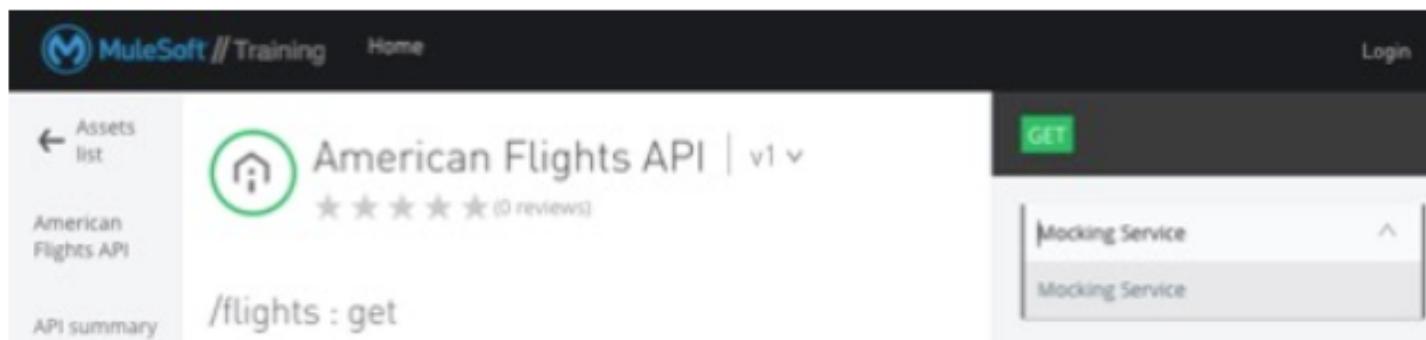
```
[Array[11]
-0: { ... }
-1: { ... }
-2: { ... }
-3: {
  "ID": 4,
  "code": "rree1000",
  "price": 200,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "CLE",
  "emptySeats": 5,
  "plane": {
    "type": "Boeing 737",
    "totalSeats": 150
  }
},
-4: {
  "ID": 5,
```

49. Make several more calls to this endpoint.

50. Make calls to different methods.

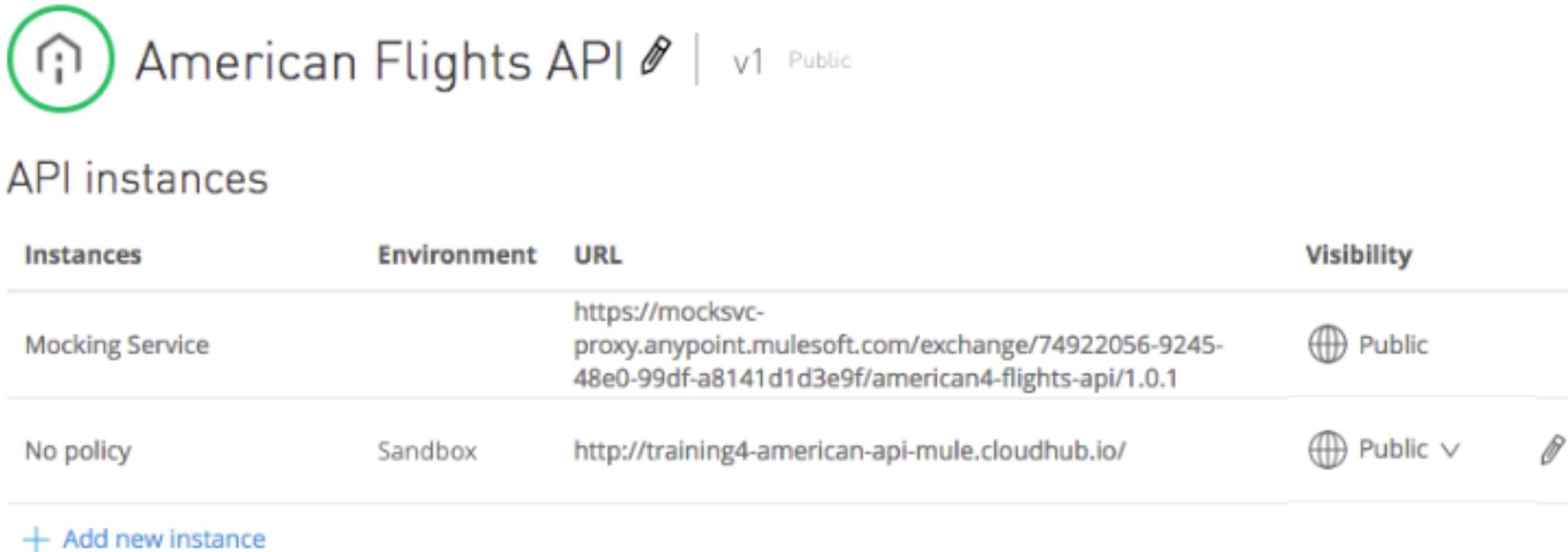
Make requests to the API proxy from the public portal

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.

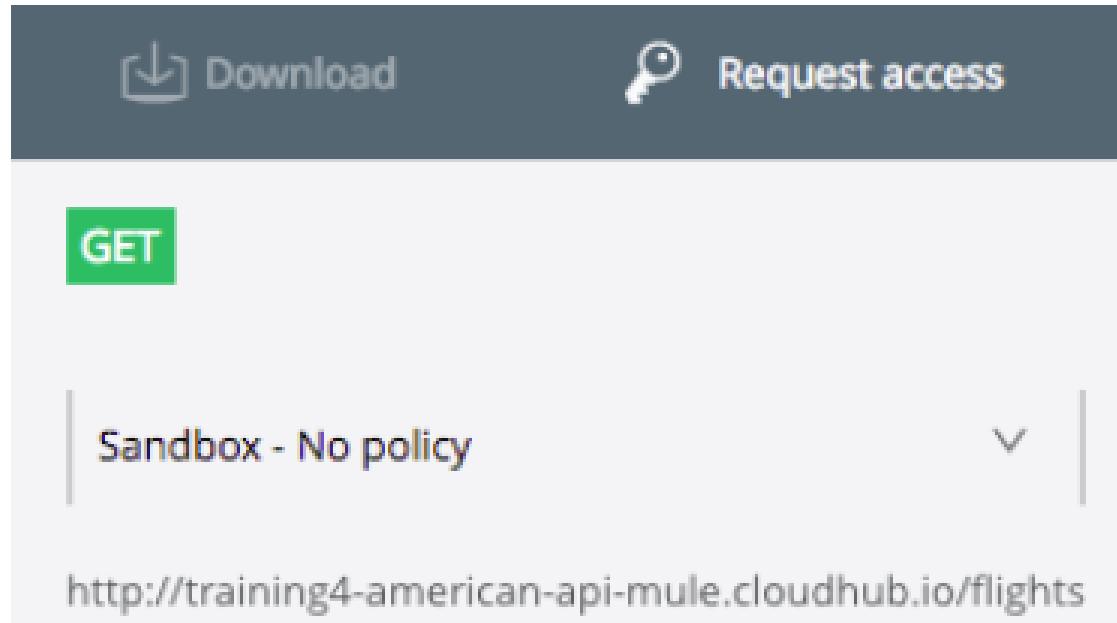


The screenshot shows the 'API instances' section of the Mule Exchange interface. At the top, there's a green circular icon with a house symbol, followed by the text 'American Flights API' and a pencil icon, indicating it's a managed API. Below this, the version 'v1' and visibility status 'Public' are shown. The main table lists two API instances:

| Instances | Environment | URL | Visibility |
|------------------------------------|-------------|---|--|
| Mocking Service | | https://mocksvc-proxy.anypoint.mulesoft.com/exchange/74922056-9245-48e0-99df-a8141d1d3e9f/american4-flights-api/1.0.1 |  Public |
| No policy | Sandbox | http://training4-american-api-mule.cloudhub.io/ |  Public  |
| + Add new instance | | | |

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.

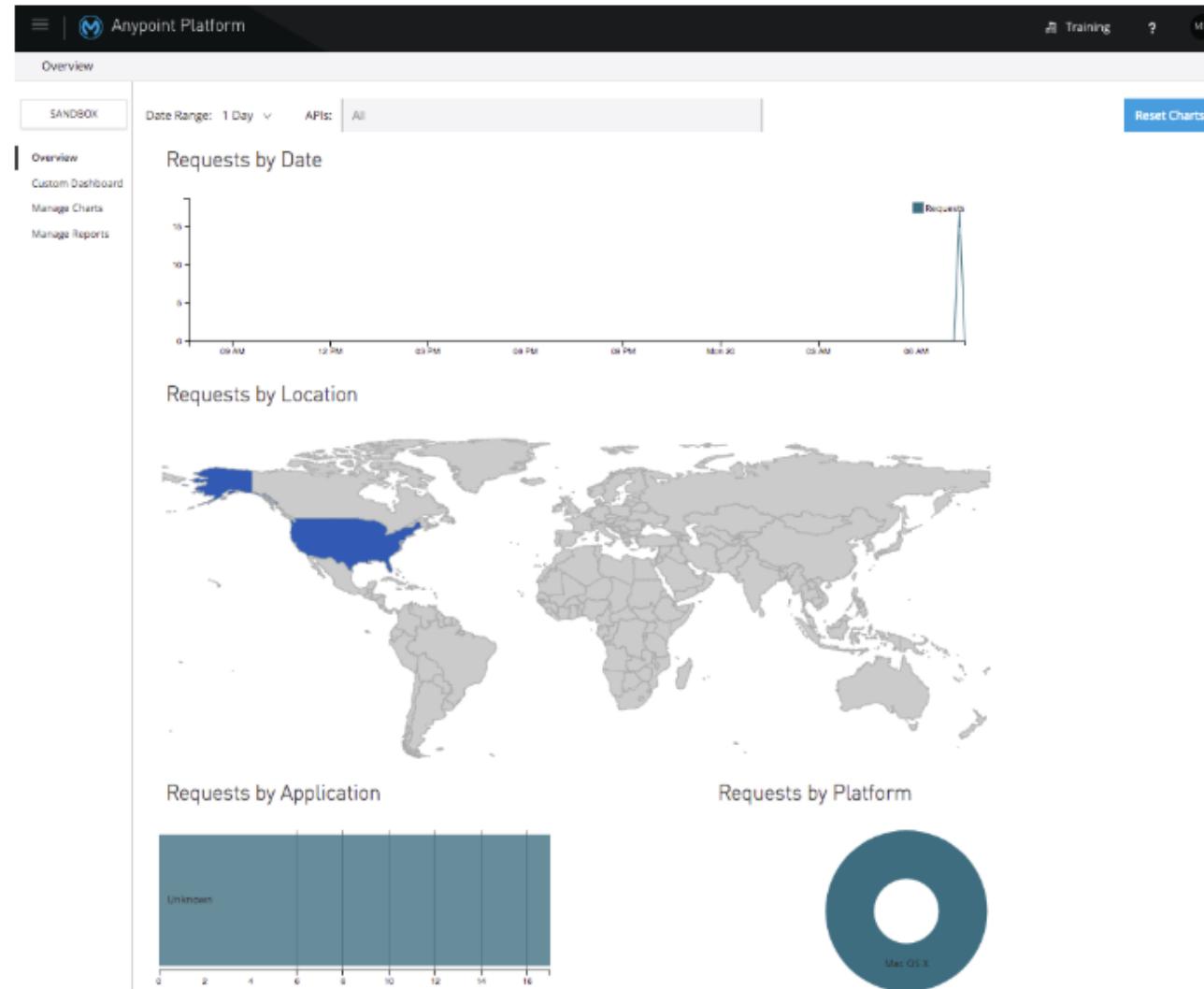
Look at the API request data

61. Return to the browser tab with API Manager.
62. Refresh the Settings page for your 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.

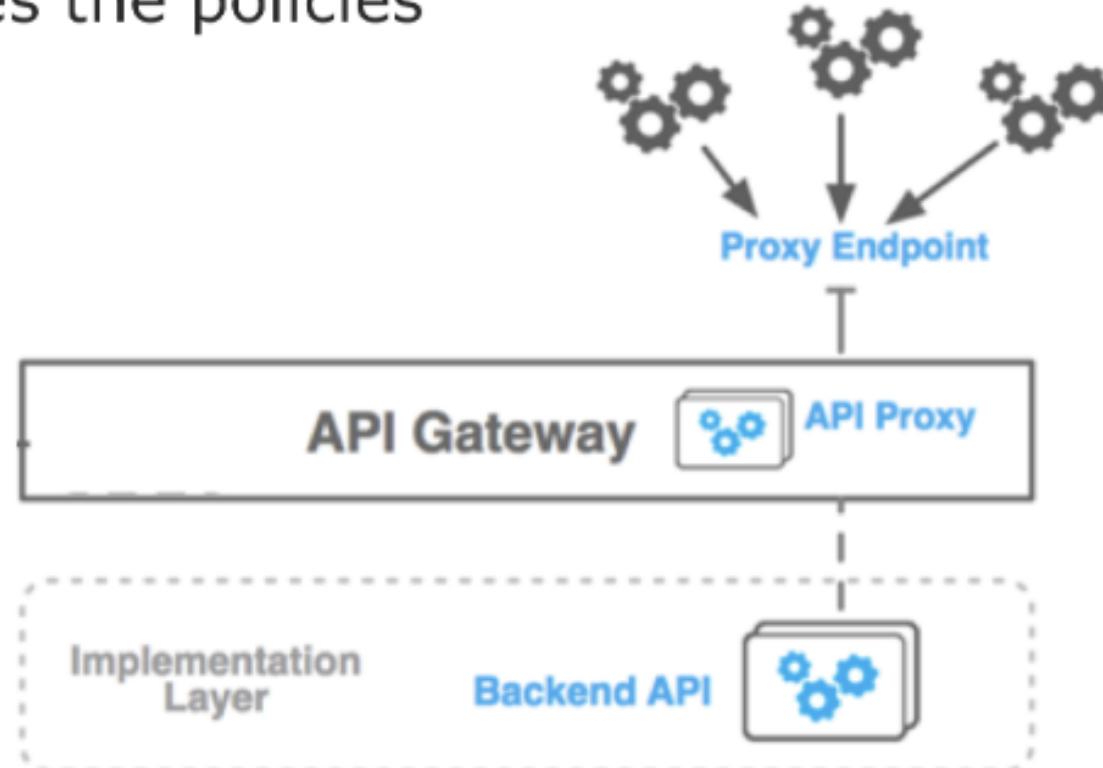


66. Close the browser tab.

Restricting access to APIs



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



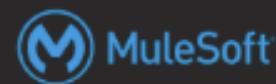
Applying policies to restrict access



- 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 **Service Level Agreement** 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 \$)

The screenshot shows the API Manager interface for the American Flights API (v1) - SLA Tiers. The left sidebar has a 'Sandbox' button highlighted, along with 'API Administration', 'Policies', 'SLA Tiers' (which is selected), and 'Settings'. The main content area shows a table of SLA tiers:

| Name | Limits | Applications | Status | Approval | Edit | Deprecate |
|--------|--------|--------------|--------|----------|-----------------------|----------------------------|
| Free | 1 | 1 | Active | Auto | <button>Edit</button> | <button>Deprecate</button> |
| Silver | 1 | 1 | Active | Manual | <button>Edit</button> | <button>Deprecate</button> |

At the top right, there are buttons for 'Training', '?', and 'MM'. Below the table, there are navigation arrows for 'View Analytics Dashboard'.

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

The screenshot shows the API Manager interface for managing policies. The top navigation bar includes 'Training', '?', and 'MM' buttons. The left sidebar has links for 'API Administration (Sandbox)', 'American Flights API (v1) - Policies', 'Sandbox', 'Alerts', 'Client Applications', **Policies**, 'SLA Tiers', and 'Settings'. The main content area displays a table for 'American Flights API (v1) - Policies'. A blue button 'Apply New Policy' is visible. The table has columns: Name, Category, Fulfils, Requires, Order, Method, and Resource URI. One policy listed is 'Rate limiting - SLA based', categorized under 'Quality of service', fulfills 'SLA Rate Limiting, Client ID required', and is associated with a 'RAML snippet'. The 'Order' column shows '1' and the 'Method' column shows 'All API Methods'. The 'Resource URI' column shows 'All API Resources'. Action buttons 'View Detail' and 'Actions' are at the bottom right of the table row.

| Name | Category | Fulfils | Requires |
|-----------------------------|--------------------|---------------------------------------|--------------|
| Rate limiting - SLA based ⓘ | Quality of service | SLA Rate Limiting, Client ID required | RAML snippet |

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.

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

- Header:** API Manager, Training, Help, MM
- Breadcrumbs:** API Administration (Sandbox) > American Flights API (v1) - Policies
- Buttons:** SANDBOX, Apply New Policy, Edit policy order
- Left Sidebar:** API Administration, Alerts, Client Applications, Policies (selected), SLA Tiers, Settings
- Policies Table:**

| Name | Category | Fulfils | Requires |
|-----------------------------|--------------------|---------------------------------------|--------------|
| Rate limiting - SLA based ⓘ | Quality of service | SLA Rate Limiting, Client ID required | RAML snippet |

Details for the applied policy:
Order: 1, Method: All API Methods, Resource URI: All API Resources

Actions: View Detail, Actions ▾

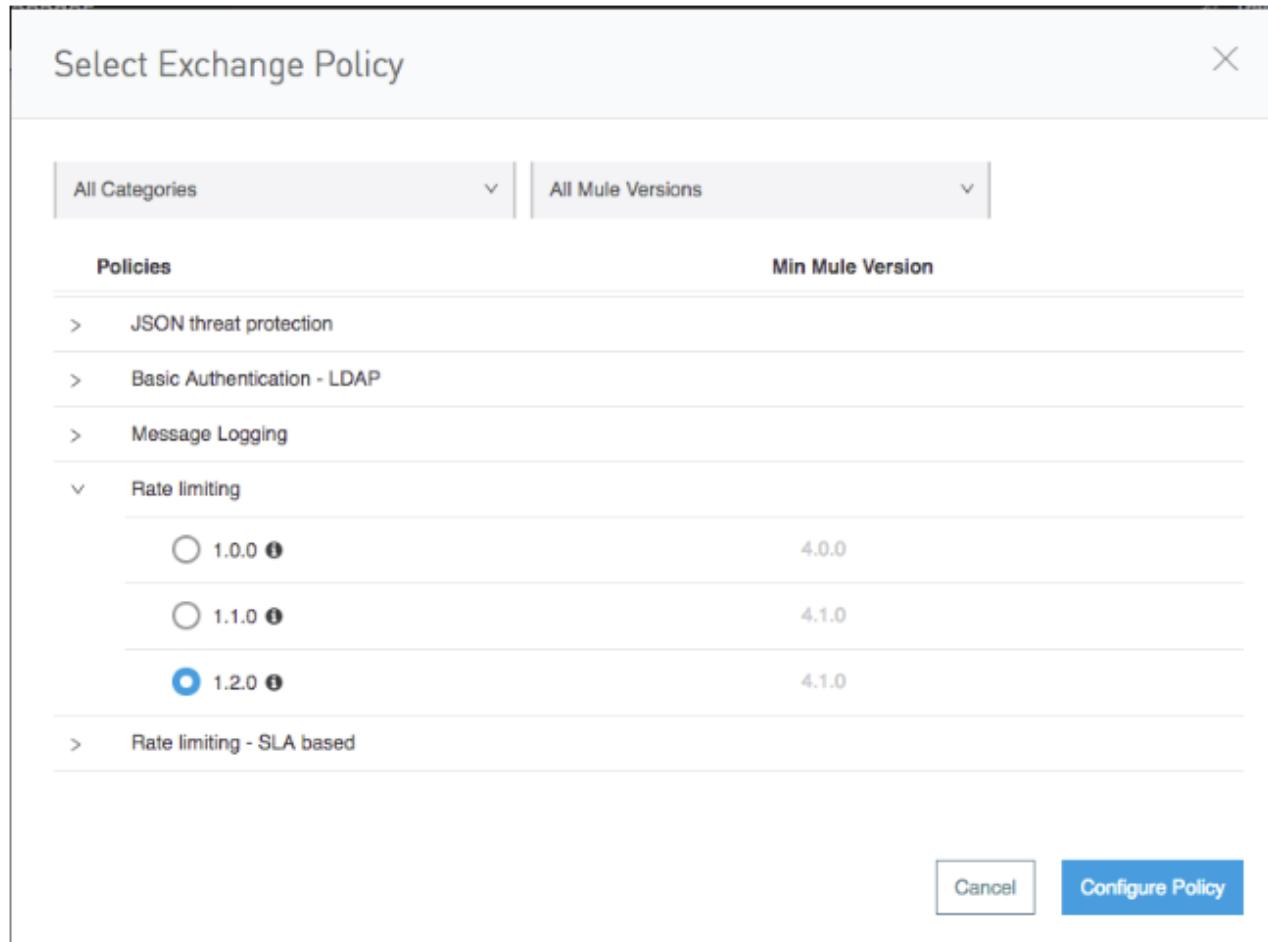
Create a rate limiting policy

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

The screenshot shows the Anypoint Manager interface. At the top, there's a navigation bar with 'API Manager' and icons for 'Training', '?', and 'MM'. Below it, a secondary navigation bar has tabs for 'API Administration (Sandbox)' and 'American Flights API (v1) - Policies'. On the left, a sidebar lists 'Sandbox', 'Alerts', 'Client Applications', 'Policies' (which is selected and highlighted in blue), 'SLA Tiers', and 'Settings'. The main content area displays the 'American Flights API v1' details: 'API Status: Active', 'Asset Version: 1.0.1', 'Type: RAML/DAS', 'Implementation URL: http://training4-american-ws-mule.cloudhub.io/api', and 'Consumer endpoint: http://training4-american-api-mule.cloudhub.io/'. To the right of these details are four actions: 'Manage CloudHub Proxy >', 'View API in Exchange >', 'View configuration details >', and 'View Analytics Dashboard >'. At the bottom of the main content area is a blue button labeled 'Apply New Policy'.

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

Apply Rate limiting policy

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

Identifier

For each identifier value, the set of Limits defined in the policy will be enforced independently. I.e.: # [attributes.queryParams['Identifier']].

| | |
|--|--|
| | |
|--|--|

Limits

Pairs of maximum quota allowed and time window.

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

[+ Add Limit](#)

Clusterizable

When using a clustered runtime with this flag enabled, configuration will be shared among all nodes.

Expose Headers
Defines if headers should be exposed in the response to the client. These headers are: x-ratelimit-remaining, x-ratelimit-limit and x-ratelimit-reset.

Method & Resource conditions

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

Cancel Apply

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

The screenshot shows the API Manager interface. The top navigation bar includes 'Training', a help icon, and a 'MM' button. The left sidebar has links for 'API Administration (Sandbox)', 'Alerts', 'Client Applications', **Policies** (which is selected), 'SLA Tiers', and 'Settings'. The main content area displays the 'American Flights API v1' details. It shows the API is Active, Asset Version is 1.0.1, and Type is RAML/OAS. Implementation URL is <http://training4-american-ws-mule.cloudhub.io/api>. Consumer endpoint is <http://training4-american-api-mule.cloudhub.io/>. On the right, there are actions like 'Manage CloudHub Proxy', 'View API in Exchange', 'View configuration details', and 'View Analytics Dashboard'. Below these are buttons for 'Apply New Policy' and 'Edit policy order'. A table lists policies, showing one entry: 'Rate limiting' (Category: Quality of service, Fulfils: Baseline Rate Limiting).

| Name | Category | Fulfils | Requires |
|---------------|--------------------|------------------------|----------|
| Rate limiting | Quality of service | Baseline Rate Limiting | |

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

API Instance ⓘ

ID: 5831632

Label: Rate limiting policy

Test the new rate limiting policy

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

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

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

The screenshot shows a screenshot of an API testing tool's interface. At the top, there is a green 'GET' button. Below it, the 'Sandbox - Rate limiting policy' instance is selected from a dropdown menu. The URL is set to `http://training4-american-api-mule.cloudhub.io/flight:`. There are two tabs: 'Parameters' and 'Headers', with 'Parameters' being the active tab. Under 'Query parameters', there is a checkbox labeled 'Show optional parameters'. A large blue 'Send' button is located at the bottom of the request area. Below the request area, an error message is displayed in an orange box: '429 Too Many Requests'. To the right of the error message are the response time '180.90 ms' and a 'Details' link. At the bottom, there are several small icons: a checkmark, a download arrow, a file icon, and a refresh icon. Below these icons, the error message is shown in a code block:

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

.

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.

The screenshot shows the API Manager interface. The top navigation bar includes 'API Manager' (with a logo), 'Training', a help icon, and a user icon. Below the bar, the left sidebar has tabs: 'Sandbox' (selected), 'API Administration (Sandbox)', 'American Flights API (v1) - SLA Tiers', 'Alerts', 'Client Applications', 'Policies', 'SLA Tiers' (selected), and 'Settings'. The main content area displays the 'American Flights API v1' details: 'API Status: Active', 'Asset Version: 1.0.1', 'Type: RAML/OAS', 'Implementation URL: http://training4-american-ws-mule.cloudhub.io/api', 'Consumer endpoint: http://training4-american-api-mule.cloudhub.io/'. On the right, there are 'Actions' dropdown menus for 'Manage CloudHub Proxy', 'View API in Exchange', 'View configuration details', and 'View Analytics Dashboard'. At the bottom, there's a search bar with 'Add SLA tier' and a close 'X' button. A message at the bottom states: 'There are no SLA tiers for this API version.'

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

Add SLA tier

Name *

 (

Description

Approval *

 (

Limits

| # of Reqs * | Time Period * | Time Unit * |
|---|---|--|
| 1 | 1 | Minute () |
| <input checked="" type="checkbox"/> visible |  | |

 [Add Limit](#)

() Cancel () Add

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

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

Change the policy to rate limiting – SLA based

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

| Name | Category | Fulfils | Requires |
|---|--------------------|------------------------|---|
| Rate limiting  | Quality of service | Baseline Rate Limiting | |
| Order | Method | Resource URI | |
|  | All API Methods | All API Resources | <div><button>View Detail</button><button>Actions ▾</button><div>DisableEditRemove</div></div> |

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.

Select Exchange Policy

All Categories | All Mule Versions

| Policies | Min Mule Version |
|--|------------------|
| > IP whitelist | |
| > JSON threat protection | |
| > Basic Authentication - LDAP | |
| > Rate limiting | |
| < Rate limiting - SLA based | |
| <input type="radio"/> 1.0.0 ⓘ | 4.0.0 |
| <input checked="" type="radio"/> 1.1.0 ⓘ | 4.1.0 |
| > Spike Control | |
| > XML threat protection | |

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

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

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

- Header:** API Manager
- Breadcrumbs:** API Administration (Sandbox) > American Flights API (v1) - Policies > Apply Rate limiting - SLA based policy
- Sandbox:** A button labeled "SANDBOX".
- Backlink:** ← Policies
- ## Apply Rate limiting - SLA based policy

Specifies the maximum value for the number of messages processed per time period, and rejects any messages beyond the maximum.

This policy will require updates to the RAML definition in order to function. You can obtain the RAML snippet and learn more [here](#).

Client ID Expression *
Mule Expression to be used to extract the Client ID from API requests.
`##[attributes.headers['client_id']]`

Client Secret Expression
Mule Expression to be used to extract the Client Secret from API requests.
`##[attributes.headers['client_secret']]`

Clusterizable
When using a clustered runtime with this flag enabled, configuration will be shared among all nodes.

Expose Headers
Defines if headers should be exposed in the response to the client. These headers are: x-ratelimit-remaining, x-ratelimit-limit and x-ratelimit-reset.

Method & Resource conditions

Apply configurations to all API methods & resources

Apply configurations to specific methods & resources

Buttons: Cancel, Apply

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

API Instance ⓘ

ID: 5831632

Label: 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.

The screenshot shows the Microsoft Azure API Management test interface. At the top left, a green button labeled "GET" is visible. Below it, a dropdown menu is open, showing the option "Sandbox - Rate limiting - SLA based policy". The main URL input field contains "http://training4-american-api-mule.cloudhub.io/flight". Below the URL, there are two tabs: "Parameters" and "Headers", with "Parameters" being the active tab. Under "Query parameters", there is a checkbox labeled "Show optional parameters" which is unchecked. At the bottom center of the interface is a blue "Send" button. After sending the request, the results section displays an orange box containing the status code "401 Unauthorized", the execution time "819.80 ms", and a "Details" link. Below this, there are several small icons: a checkmark, a download arrow, a copy icon, and a refresh icon. A code editor window shows the following JSON response:

```
{  
    "error": "Invalid client id or secret"  
}
```

Granting access to APIs



Enforcing access to APIs using SLA tiers

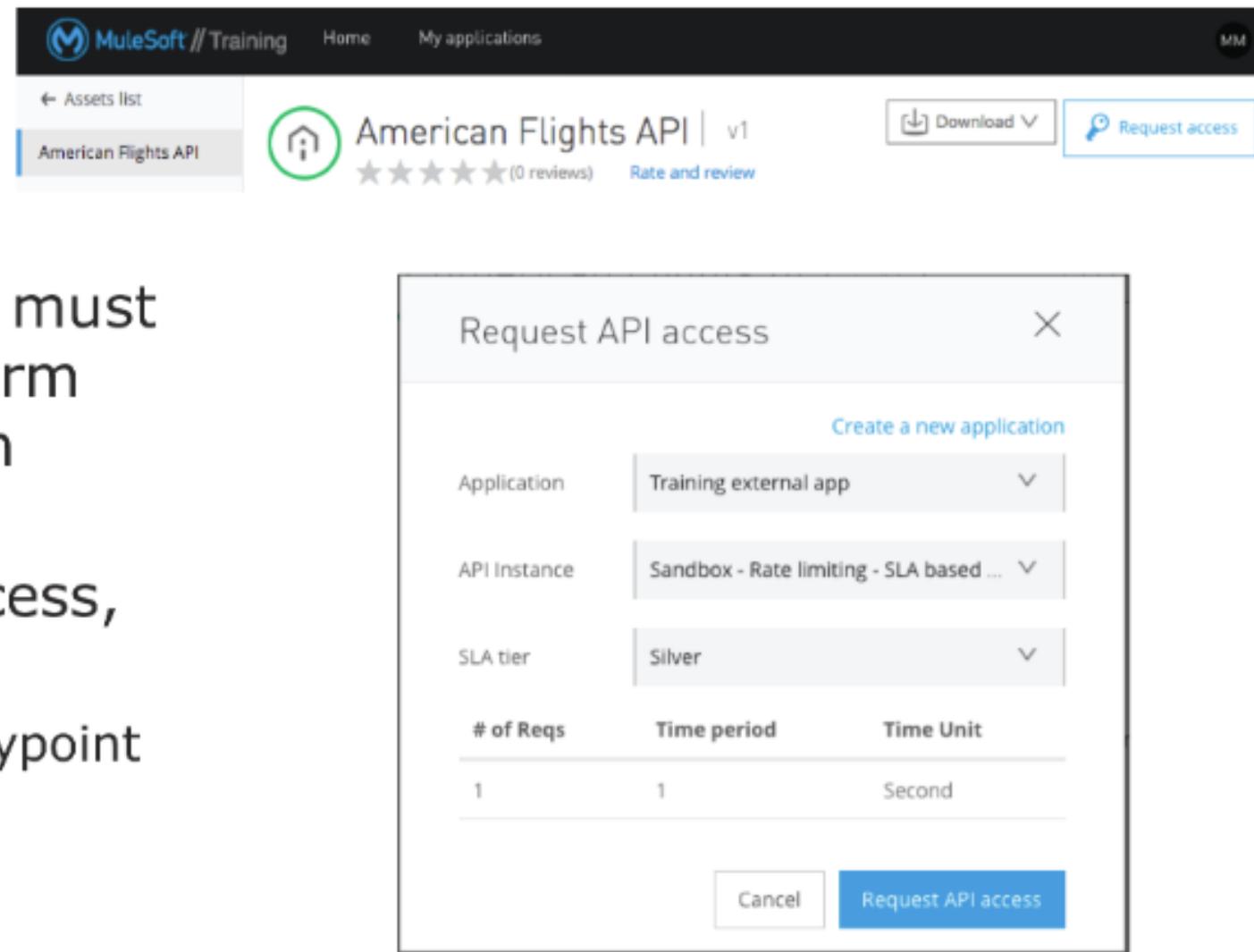


- To enforce, apply an **SLA based** rate limiting policy
- SLA based policies require all applications that consume the API to
 - **Register** for access to a specific tier
 - From an API portal in private or public Exchange
 - **Pass their client credentials** in calls made to the API

The screenshot shows a REST API request configuration. At the top, it says "GET". Below that is the URL "http://training4-american-api-mule.cloudhub.io/flights". There are two tabs: "Parameters" and "Headers", with "Headers" currently selected. Under "Headers", there are two entries: "Header name: client_id" with "Value: 7623dbcc2e1949d7" and "Header name: client_secret" with "Value: 52505680a6FB4d5". Below the headers is a blue "Send" button. At the bottom, the response is shown as "200 OK" with a duration of "1042.50 ms" and a "Details" link. The response body is an array of 11 flight objects, with the first one partially visible: { "ID": 1, "code": "rree0001", "origin": "ATL", "dest": "JFK", "airline": "AA", "gate": "12A", "status": "Arrived", "actualArrival": "2018-01-01T15:00:00Z", "estimatedArrival": "2018-01-01T15:00:00Z", "actualDeparture": "2018-01-01T14:55:00Z", "estimatedDeparture": "2018-01-01T14:55:00Z", "delay": 0, "cancelReason": null, "cancelReasonText": null, "gateAssignment": "12A", "airlineCode": "AA", "flightNumber": "rree0001", "carrier": "American Airlines", "carrierCode": "AA", "carrierName": "American Airlines", "carrierLogo": "https://www.americanairlines.com/-/media/assets/aa-com/aa-airlines/logo/aa-airlines-new-logo.ashx", "carrierLogoText": "American Airlines", "carrierLogoTextLink": "https://www.americanairlines.com", "carrierLogoTextLinkText": "American Airlines", "carrierLogoTextLinkTitle": "American Airlines", "carrierLogoTextLinkTarget": "_blank", "carrierLogoTextLinkRel": "noopener", "carrierLogoTextLinkType": "link", "carrierLogoTextLinkClass": "carrierLogoTextLink", "carrierLogoTextLinkStyle": "color: #000000; text-decoration: none; font-size: 14px; font-weight: bold; font-family: Arial, sans-serif; margin-bottom: 5px;"}.

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



The screenshot shows the MuleSoft API portal interface. At the top, there's a navigation bar with the MuleSoft logo, 'MuleSoft // Training', 'Home', and 'My applications'. Below the navigation, the 'American Flights API | v1' is displayed, along with a green circular icon containing a house symbol, a star rating of 5 stars with '(0 reviews)', and a 'Rate and review' link. To the right of the API name are 'Download' and 'Request access' buttons. A modal window titled 'Request API access' is open in the center. It contains fields for 'Application' (set to 'Training external app'), 'API Instance' (set to 'Sandbox - Rate limiting - SLA based ...'), and 'SLA tier' (set to 'Silver'). Below these fields is a table with three columns: '# of Reqs', 'Time period', and 'Time Unit'. The first row of the table has values '1', '1', and 'Second'. At the bottom of the modal are 'Cancel' and 'Request API access' buttons.

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

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

- Header:** API Manager, Training, Help, MM
- Breadcrumbs:** API Administration (Sandbox) > American Flights API (v1) - Client Applications
- Left Sidebar:** SANDBOX, Alerts, Client Applications (selected), Policies, SLA Tiers, Settings
- Table Headers:** Application, Current SLA tier, Requested SLA tier, Status
- Table Data:**
 - Application: Training external app, Current SLA tier: N/A, Requested SLA tier: Silver, Status: Pending, Actions: Approve, Reject, Delete
 - Application: Training internal app, Current SLA tier: Free, Requested SLA tier: N/A, Status: Approved, Action: Revoke
- Pagination:** 1 - 2 of 2, Next/Previous buttons

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

The screenshot shows the API Manager interface for the American Flights API (v1) - Client Applications. The sidebar on the left has tabs for Alerts, Client Applications (which is selected), Policies, SLA Tiers, and Settings. The main content area shows a table of client applications. The table has columns for Application, Current SLA tier, Requested SLA tier, and Status. It lists two entries: 'Training external app' and 'Training internal app'. The 'Training external app' entry has a 'Silver' current tier, 'N/A' requested tier, and is 'Approved'. There is a 'Revoke' button next to it. Below this entry is a detailed row with columns for Owners (Max Mule), Client ID (7623dbcc2e1949d7a861160fe4a3a1e6), URL (None), and Redirect URIs (None). The status for this row is 'Submitted' with a timestamp of '2 minutes ago'. The 'Training internal app' entry has a 'Free' current tier, 'N/A' requested tier, and is 'Approved'. There is a 'Revoke' button next to it.

| SANDBOX | Application | Current SLA tier | Requested SLA tier | Status | |
|----------------------|-------------------------|----------------------------------|--------------------|-----------|-------------------|
| ← API Administration | ▼ Training external app | Silver | N/A | Approved | Revoke |
| Alerts | Owners | Max Mule | | Submitted | 2 minutes ago |
| Client Applications | Client ID | 7623dbcc2e1949d7a861160fe4a3a1e6 | | Approved | a few seconds ago |
| Policies | URL | None | | Rejected | - |
| SLA Tiers | Redirect URIs | None | | Revoked | - |
| Settings | > Training internal app | Free | N/A | Approved | Revoke |

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:

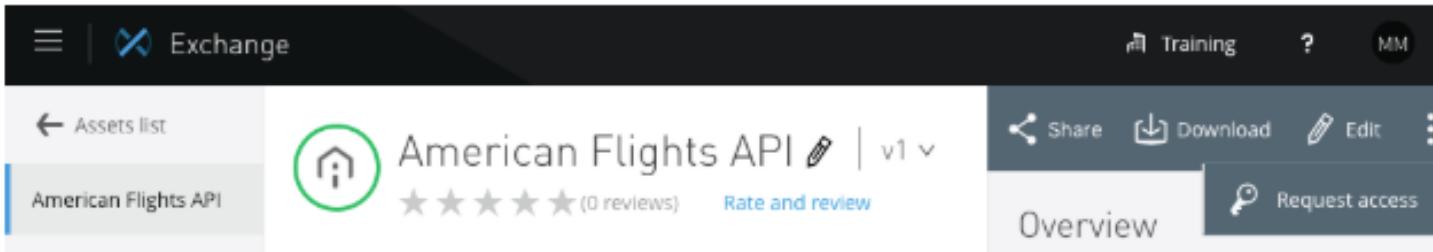
- Request application access to SLA tiers from private and public API portals.
- Approve application requests to SLA tiers in API Manager.

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

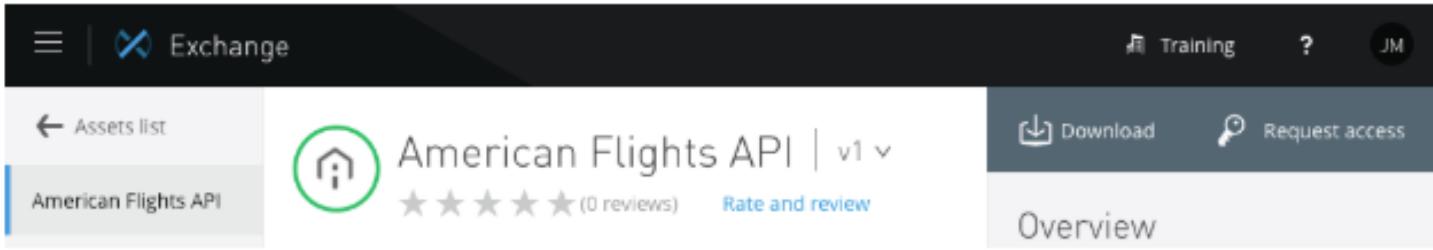
| Application | Current SLA tier | Requested SLA tier | Status | Action |
|---|------------------|--------------------|-------------------|--------|
| Training external app | Silver | N/A | Approved | Revoke |
| Owners: Max Mule | | Submitted | 2 minutes ago | |
| Client ID: 7623dbcc2e1949d7a861160fe4a3a1e6 | | Approved | a few seconds ago | |
| URL: None | | Rejected | - | |
| Redirect URIs: None | | Revoked | - | |
| Training internal app | Free | N/A | Approved | Revoke |

Request access to the API as an internal consumer

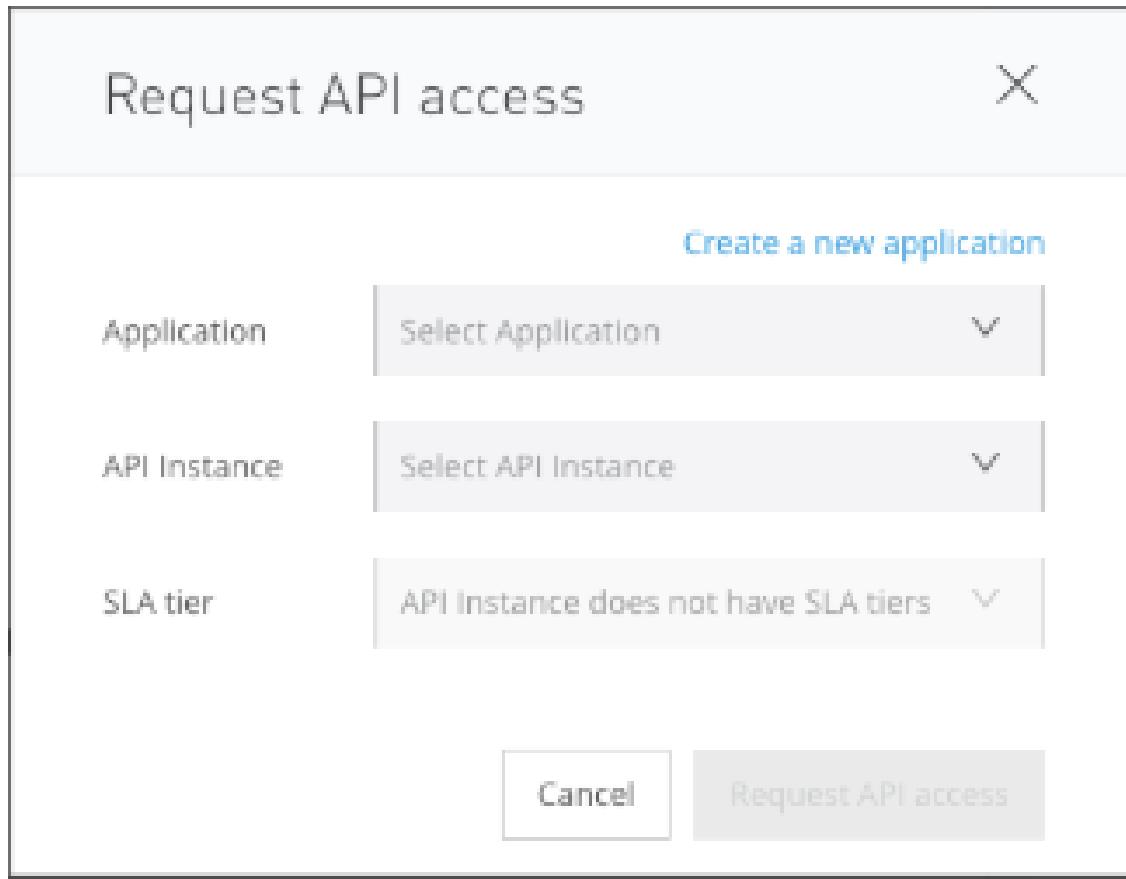
1. Return to the browser tab with Anypoint Exchange.
2. In the left-side navigation, select the name of the API to return to its home page.
3. Click the more options button in the upper-right corner and select Request access.



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.



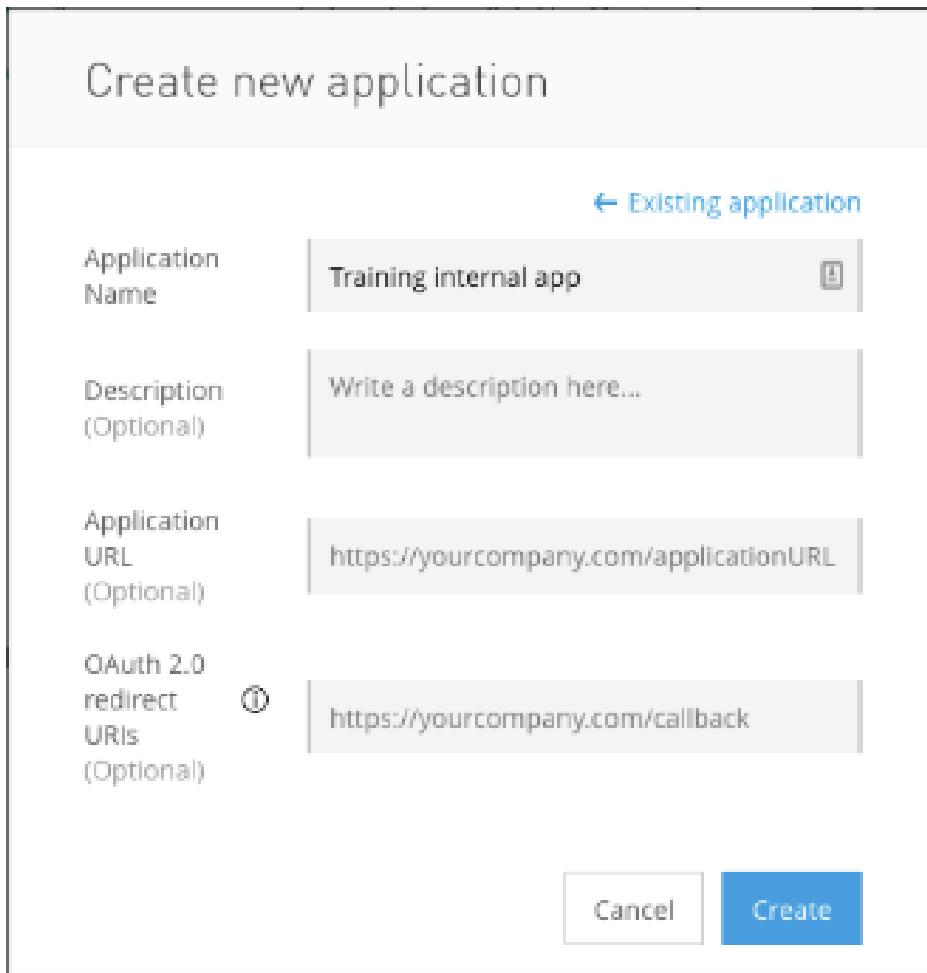
5. In the Create new application dialog box, set the name to Training internal app and click Create.

Create new application

[Existing application](#)

| | |
|------------------------------------|--|
| Application Name | Training internal app |
| Description (Optional) | Write a description here... |
| Application URL (Optional) | https://yourcompany.com/applicationURL |
| OAuth 2.0 redirect URIs (Optional) | https://yourcompany.com/callback |

[Cancel](#) [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.

Request API access

Create a new application

Application: Training internal app

API Instance: Sandbox - Rate limiting - SLA based ...

SLA tier: Free

| # of Reqs | Time period | Time Unit |
|-----------|-------------|-----------|
| 1 | 1 | Minute |

Cancel Request API access

8. Click Request API access.

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

Request API access

✓ API access has been successful!

Client ID: e708026bb0cf4c3e8594ca39138b9a00

Client secret: e10A1faF382D47DEA6A90cA8d4FC3891

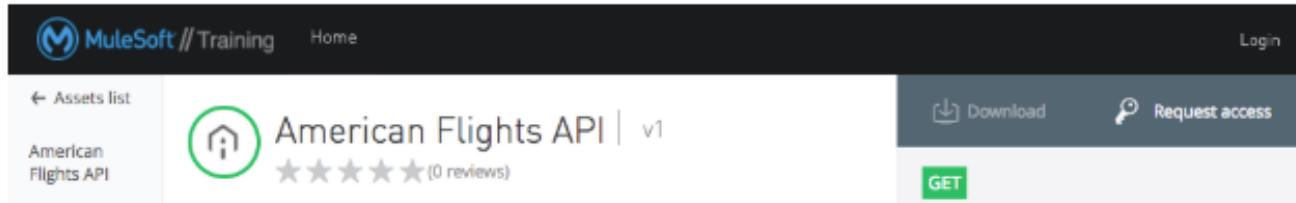
Application details have been opened in a new tab.

Close

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.



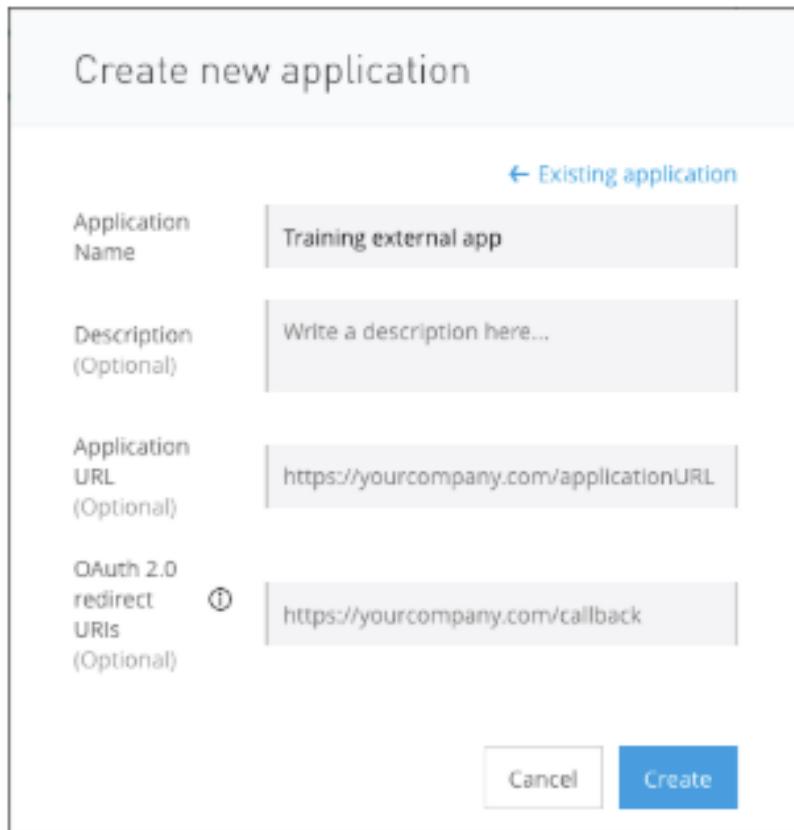
13. Click the Request access button; you should get a page to sign in or create an Anypoint Platform account.
14. Enter your existing credentials and click Sign in.

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

A screenshot of the Anypoint Platform sign-in page. It features a light blue header with the MuleSoft logo and the text 'Anypoint Platform'. Below this is a 'Sign in' section with two input fields: 'Username' containing 'maxmule4' and 'Password' containing several dots. To the right of these fields is a dark blue sidebar with the text 'Don't have an account?' and 'Create one for free.' at the top, and a 'Sign up' button at the bottom. At the very bottom of the page, there are links for 'Forgot sign-in credentials?' and 'Privacy policy'.

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.



The screenshot shows the 'Create new application' dialog box. It has a light gray header bar with the title 'Create new application' and a blue 'Existing application' link. Below the header are four input fields: 'Application Name' with the value 'Training external app', 'Description (Optional)' with the placeholder 'Write a description here...', 'Application URL (Optional)' with the placeholder 'https://yourcompany.com/applicationURL', and 'OAuth 2.0 redirect URIs (Optional)' with the placeholder 'https://yourcompany.com/callback'. At the bottom right are two buttons: a white 'Cancel' button and a blue 'Create' button.

| Field | Value |
|------------------------------------|--|
| Application Name | Training external app |
| Description (Optional) | Write a description here... |
| Application URL (Optional) | https://yourcompany.com/applicationURL |
| OAuth 2.0 redirect URIs (Optional) | https://yourcompany.com/callback |

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.

Request API access X

[Create a new application](#)

| | | |
|--------------|---|---|
| Application | Training external app | ▼ |
| API Instance | Sandbox - Rate limiting - SLA based ... | ▼ |
| SLA tier | Silver | ▼ |

| # of Reqs | Time period | Time Unit |
|-----------|-------------|-----------|
| 1 | 1 | Second |

Cancel Request API access

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.

The screenshot shows the 'My applications' page of the MuleSoft // Training portal. At the top, there is a navigation bar with the MuleSoft logo, 'MuleSoft // Training', 'Home', and 'My applications'. Below the navigation bar, the page title 'My applications' is displayed. A search bar with a magnifying glass icon and the placeholder 'Search' is present. To the right of the search bar is a close button ('X'). The main content area is a table with two rows. The first row has 'Name' and 'Description' columns. The second row contains the application names: 'Training internal app' and 'Training external app'.

| Name | Description |
|-----------------------|-------------|
| Training internal app | |
| Training external app | |

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.

The screenshot shows the details page for the 'Training external app'. At the top, there is a navigation bar with the MuleSoft logo, 'MuleSoft // Training', 'Home', and 'My applications'. Below the navigation bar, there is a back arrow labeled '← My applications'. The main title is 'Training external app'. To the right of the title are three buttons: 'Edit' (highlighted in blue), 'Reset client secret', and 'Delete'. On the left, there is a sidebar with a list: 'Show All (1)' (selected) and 'Sandbox - Rate limiti... v1'. The main content area displays application details: 'Application Description:' (empty), 'Application URL:' (empty), 'Redirect URIs:' (empty), 'Client ID: 7623dbcc2e1949d7a861160fe4a3a1e6', 'Client Secret: Show' (with a 'Show' link), and 'Grant Types: -'.

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.

The screenshot shows the 'Client Applications' section of the API Manager interface. On the left, there's a sidebar with options like Alerts, Client Applications (which is selected and highlighted in blue), Policies, SLA Tiers, and Settings. The main area has a search bar and a table with two rows. The first row for 'Training external app' is expanded, showing more details:

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

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.

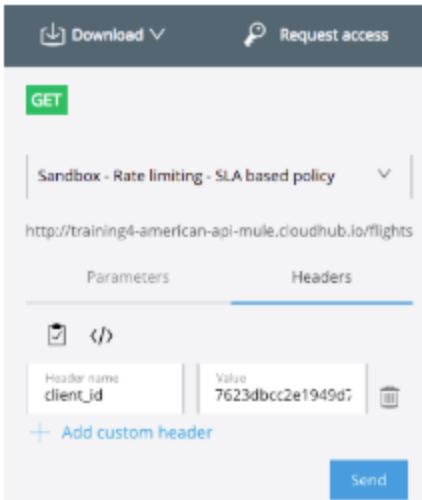
This screenshot provides a detailed view of the 'Training external app' row from the previous table. The 'Owners' section shows 'Max Mule'. The 'Client ID' section shows '7623dbcc2e1949d7a861160fe4a3a1e6'. The 'URL' and 'Redirect URIs' sections both show 'None'. The status is 'Approved' with a timestamp of 'a few seconds ago'. The 'Action' button is labeled 'Revoke'.

| Application | Current SLA tier | Requested SLA tier | Status | Action |
|-------------------------|------------------|--------------------|----------|--------|
| > Training external app | Silver | N/A | Approved | Revoke |

Below this, another row for 'Training internal app' is partially visible.

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.

A screenshot of a user interface for configuring API headers. The top navigation bar has tabs for "Parameters" and "Headers". The "Headers" tab is selected, indicated by a blue underline. Below the tabs, there is a checkbox followed by a placeholder text "</>". Underneath this, there are two rows of header definitions. Each row consists of a "Header name" field and a "Value" field, separated by a vertical line. To the right of each row is a small trash can icon for deletion. Row 1: Header name "client_id" with Value "7623dbcc2e1949d7"; Row 2: Header name "client_secret" with Value "52505680a6FB4d5;". At the bottom left of the header table is a blue "+ Add custom header" button.

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

A screenshot of a UI component displaying the results of an API request. At the top, it shows a green "200 OK" status box and a timestamp "1119.00 ms". To the right of the status is a "Details" button with a downward arrow. Below this, there is a toolbar with several icons: a checkbox, a download icon, a copy icon, and a refresh/circular arrow icon. The main content area starts with the text "[Array[11]]" followed by a list of flight results. Each result is an object with properties: ID, code, price, and departureDate. The first result is fully visible, showing ID: 1, code: "rree0001", price: 541, and departureDate: "2016-01-".

```
[Array[11]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-
```

Adding client ID enforcement to API specifications



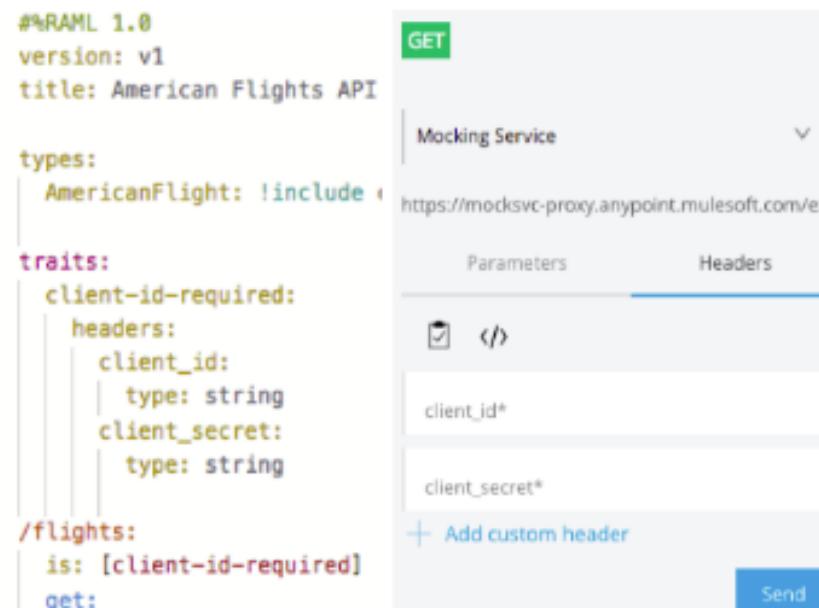
- You need to add client ID enforcement to the API spec for the
 - REST connector that is created for the API to enforce the authentication
 - Required headers to automatically show up in the API console so you don't have to manually add them for every call
- Instructions are in the RAML snippet for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

| Name | Category | Fulfils | Requires | RAML snippet |
|---|--------------------|---------------------------------------|----------|---|
| Rate limiting - SLA based  | Quality of service | SLA Rate Limiting, Client ID required | | <p>RAML 0.8 RAML 1.0</p> <p>Client ID biased policies by default expect to obtain the client ID and secret as query parameters. To enforce this in the API definition a trait can be defined in RAML as shown below.</p> <pre>traits: client-id-required: headers: client_id: type: string client_secret: type: string</pre> <p>This trait must then be applied to the resource or methods using the <code>is: RAML</code> attribute.</p> <pre>/products: get: is: [client-id-required] description: Gets a list of all the inventory products.</pre> <p>Please read Applying Resource Types and Traits section on RAML documentation for more information.</p> <p>Close</p> |

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



The image shows a side-by-side comparison of an API specification and its runtime interface.

API Specification (RAML 1.0):

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

types:
  AmericanFlight: !include ./types/AmericanFlight.raml

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

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

Runtime Interface (Mocking Service):

A screenshot of the MuleSoft Anypoint Platform's Mocking Service interface for a GET request. The URL is https://mocksvc-proxy.anypoint.mulesoft.com/exc. The Headers tab is selected, showing two required headers: 'client_id*' and 'client_secret*'. There is also a 'Parameters' tab and a 'Send' button at the bottom right.

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.

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, there is a code editor displaying RAML 1.0 API specifications. The code includes sections for types, traits, and a /flights endpoint. The traits section contains a 'client-id-required' trait with 'client_id' and 'client_secret' headers defined. On the right, a 'GET' request configuration panel is open. It shows the URL as 'Mocking Service https://mocksvc-proxy.anypoint.mulesoft.com/exc'. Below the URL, there are two tabs: 'Parameters' and 'Headers'. Under the 'Headers' tab, there are two input fields: 'client_id*' and 'client_secret*'. A blue '+' button labeled 'Add custom header' is located at the bottom of the header list. At the very bottom right of the panel is a blue 'Send' button.

```
#%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:
```

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.

The screenshot shows the API Manager interface. The top navigation bar includes 'Training', a question mark icon, and a user profile icon. Below the header, the URL 'http://training4-american-api-mule.cloudhub.io/' is displayed along with 'View configuration details' and 'View Analytics Dashboard' links. On the left, a sidebar menu lists 'Sandbox', 'API Administration', 'Alerts', 'Client Applications', **Policies** (which is selected), 'SLA Tiers', and 'Settings'. The main content area displays a table of policies. The table has columns for 'Name', 'Category', 'Fulfils', and 'Requires'. A blue 'Apply New Policy' button is located above the table. To the right of the table is a 'Edit policy order' button. The 'Rate limiting - SLA based' policy is listed in the table, showing it belongs to the 'Quality of service' category, fulfills 'SLA Rate Limiting, Client ID required', and has a 'RAML snippet' link under 'Requires'.

| Name | Category | Fulfils | Requires |
|---------------------------|--------------------|---------------------------------------|--------------|
| Rate limiting - SLA based | Quality of service | SLA Rate Limiting, Client ID required | RAML snippet |

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

5. Copy the value for the traits.

RAML snippet for Rate limiting - SLA based X

RAML 0.8 RAML 1.0

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

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

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

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

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

Close

6. Click Close.

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.

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

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

```
15
16 /flights:
17
18   get:
```

Types and Traits

is
type

Docs

description
displayName

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.

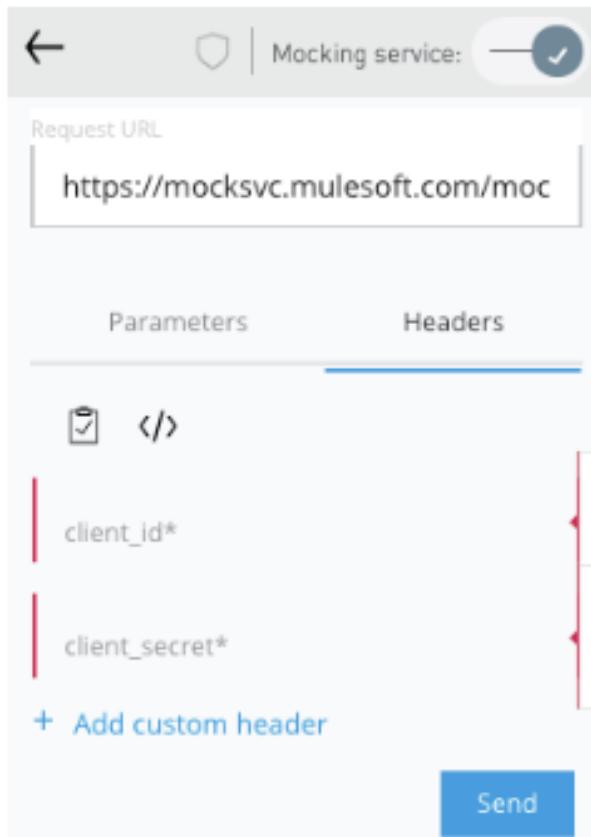
```
16 /flights:  
17   | is: [client-id-required]  
18   | get:
```

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

400 Bad Request 499.00 ms ▾

The requested URL
can't be reached



The service might be
temporarily down or it may
have moved permanently to a
new web address.

Resource is unavailable

□ ■ ◀ ▶ ⌂

```
{  
  "error": "headers: client_id: required"  
}
```

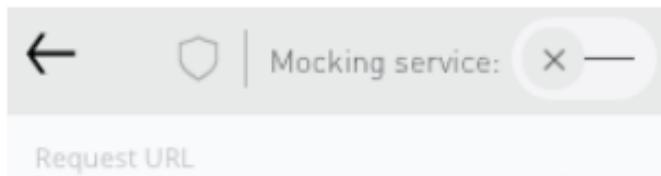
18. Enter *any* values for the client_id and client_secret and click Send; you should get a 200 response with the example results.

The screenshot shows a web-based interface for a 'Mocking service'. At the top, there's a back arrow, a shield icon, and the text 'Mocking service:'. Below that, the 'Request URL' is set to 'https://mocksvc.mulesoft.com/mocks'. There are two tabs: 'Parameters' (which is selected) and 'Headers'. Under 'Parameters', there are two entries under 'Query parameters': 'client_id*' with value '432' and 'client_secret*' with value '765'. A checkbox labeled 'Show optional parameters' is unchecked. A large blue 'Send' button is centered below the parameters. At the bottom, the status is shown as '200 OK' in a green box, followed by '506.59 ms'. Below this, there are icons for copy, download, and refresh. The response body is partially visible, showing an array of objects:

```
[Array[2]
  -0: {
    "ID": 1,
    "code": "ER38sd",
    "price": 400
  }
]
```

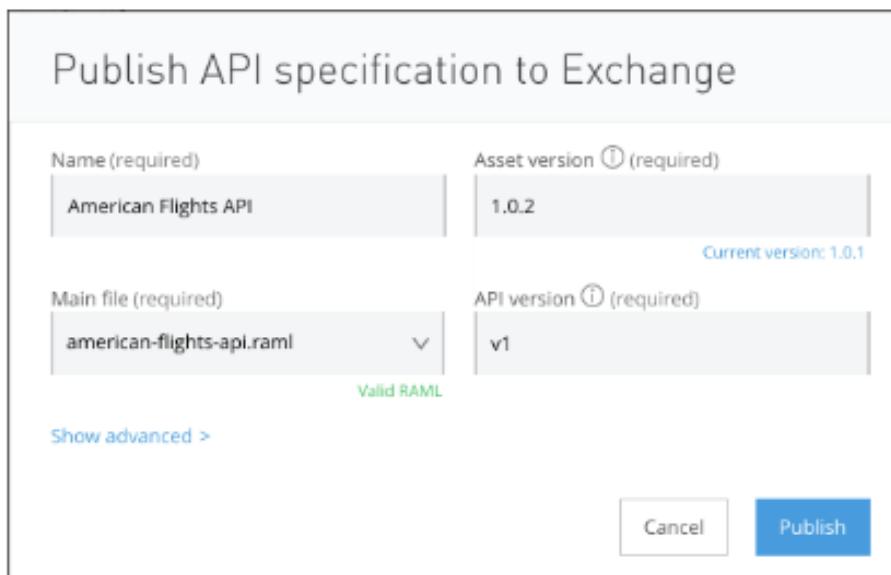
Publish the new version of the API to Exchange

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.

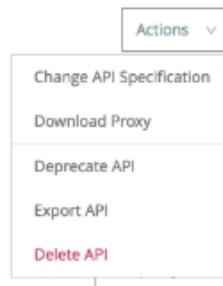
American Flights API v1

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

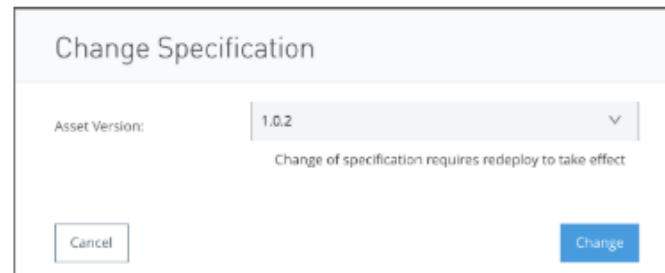
Implementation URL: <http://training4-american-ws-mule.cloudhub.io/api>

Consumer endpoint: <http://training4-american-api-mule.cloudhub.io/> 

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

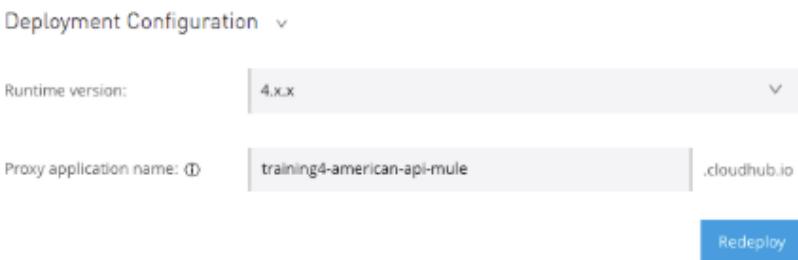


26. In the Change Specification dialog box, select the latest asset version, 1.0.2.
27. Click Change.

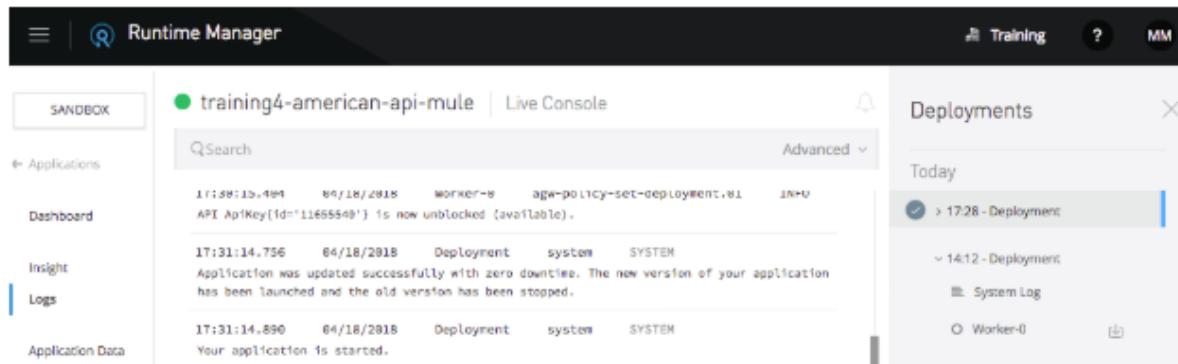


Redeploy a new proxy

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.

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

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.

The screenshot shows the Anypoint Platform Mocking Service interface. At the top left is a green 'GET' button. Below it is a dropdown menu set to 'Mocking Service'. The URL field contains 'https://mocksvc-proxy.anypoint.mulesoft.com/exc'. Underneath the URL are two tabs: 'Parameters' and 'Headers', with 'Headers' being the active tab. In the 'Headers' section, there is a checked checkbox labeled '</>'. Below it are two input fields: 'client_id*' and 'client_secret*'. At the bottom left of the 'Headers' section is a blue '+ Add custom header' button, and at the bottom right is a blue 'Send' button.

39. Close all Anypoint Platform browser windows and tabs.

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

- 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

- 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 -----> Test HTTP Listener -----> 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.cloudbu.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.cloudbuild.io/{yourTestEndpoint}`?
- What happens when you try to access your test flow endpoints via the direct Mule worker URL: `http://mule-worker-appname.cloudbuild.io:9090/{yourTestEndpoint}`?
- What happens when you try to access your test flow endpoints via the direct Mule worker URL: `http://mule-worker-appname.cloudbuild.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.cloudbu.io/api?`
 - Through the external elastic load balancer URL at `http://appname.cloudbu.io:8081/api?`
 - Directly at a Mule worker, bypassing the external elastic load balancer, at `http://mule-worker-appname.cloudbu.io/api?`
 - Directly at a Mule worker, bypassing the external elastic load balancer, at `http://mule-worker-appname.cloudbu.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.