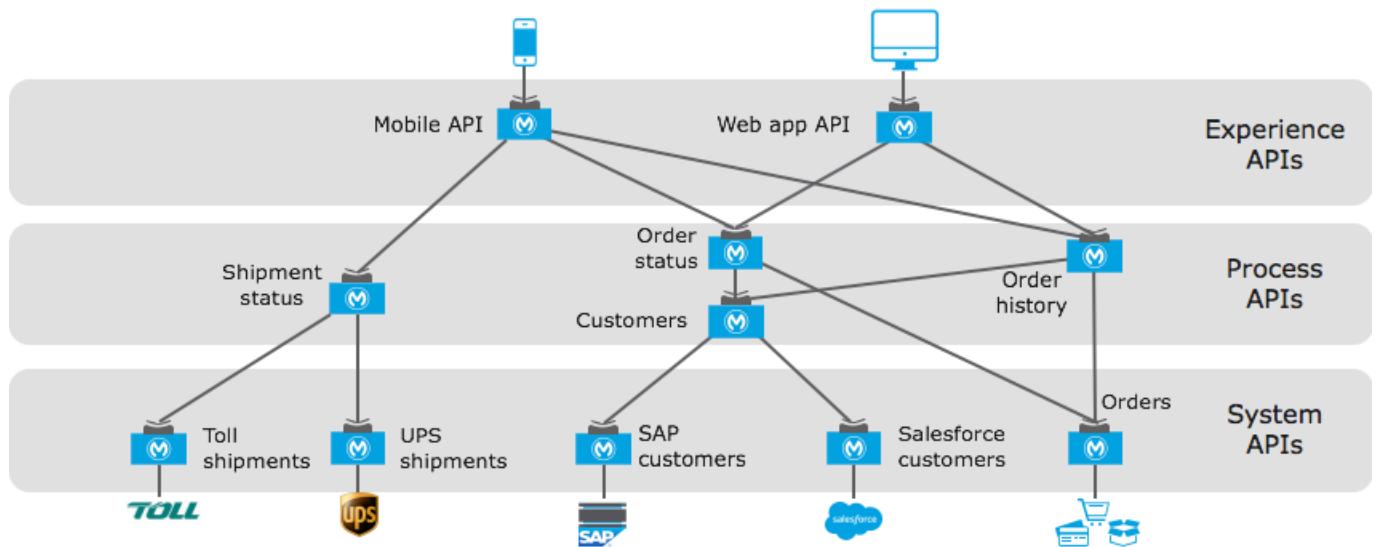


# Module 1: Introducing Application Networks and API-Led Connectivity



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

- Explain what an application network is and its benefits.
- Describe how to build an application network using API-led connectivity.
- Explain what web services and APIs are.
- Explore API directories and references.
- Make calls to secure and unsecured APIs.

# Walkthrough 1-1: Explore an API directory and an API reference

In this walkthrough, you make calls to a RESTful API. You will:

- Browse the ProgrammableWeb API directory.
- Explore the API Reference for the Twitter API.



## Browse the ProgrammableWeb API directory

1. In a web browser, navigate to <http://www.programmableweb.com/>.
2. Click the API directory link.

A screenshot of the ProgrammableWeb API directory website. The header includes a logo, a search bar, and links for API NEWS, API DIRECTORY, BECOME MEMBER, and LOGIN. Below the header is a navigation bar with categories: API UNIVERSITY, RESEARCH, SPORTS, SECURITY, TRAVEL, DESIGN, and ADD APIs &amp; MORE. To the right of the navigation are social media icons for RSS, Facebook, Twitter, Google+, and LinkedIn. The main content area features a map with a location pin and an advertisement for MuleSoft's "Build a better API strategy" guide.

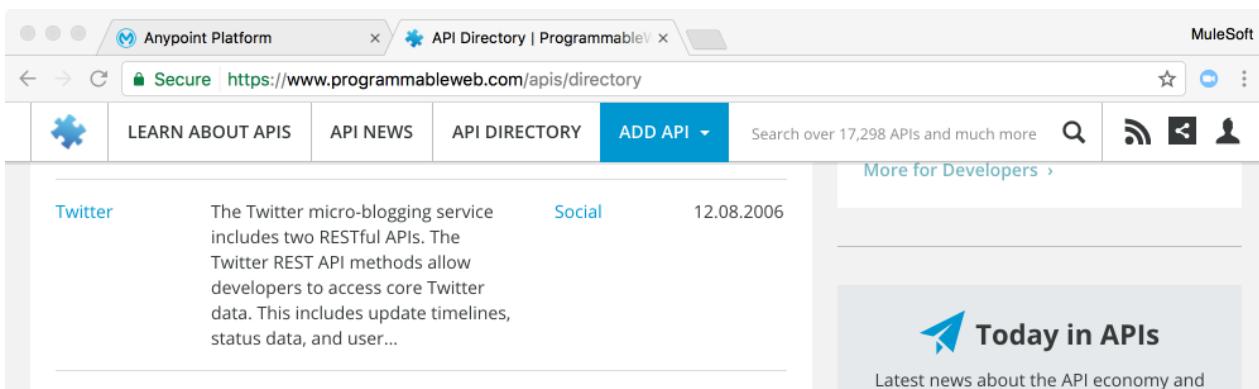
3. Browse the list of popular APIs.

		LEARN ABOUT APIS	API NEWS	API DIRECTORY	ADD API ▾	Search
Twitter	The Twitter micro-blogging service includes two RESTful APIs. The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user...	Social	12.08.2006			
YouTube	The Data API allows users to integrate their program with YouTube and allow it to perform many of the operations available on the website. It provides the capability to search for videos, retrieve...	Video	02.08.2006			
Flickr	The Flickr API can be used to retrieve photos from the Flickr photo sharing service using a variety of feeds - public photos and videos, favorites, friends, group pools, discussions, and more. The...	Photos	09.04.2005			
Facebook	The Facebook API is a platform for building applications that are available to	Social	08.16.2006			

## Explore the API reference for the Twitter API

4. Scroll down and click the link for the Twitter API.

*Note: If Twitter is no longer displayed on the main page, search for it.*



The screenshot shows a web browser window with the following details:

- Address Bar:** Shows "Secure https://www.programmableweb.com/apis/directory".
- Page Title:** "Anypoint Platform" and "API Directory | ProgrammableWeb".
- Header:** Includes links for "LEARN ABOUT APIS", "API NEWS", "API DIRECTORY", "ADD API ▾", and a search bar.
- Content Area:** Displays the Twitter API entry from the previous table, including its description, category (Social), and creation date (12.08.2006).
- Right Sidebar:** Contains a "More for Developers" section and a "Today in APIs" news feed.

5. In the Specs section, click the API Portal / Home Page link.

## SPECS

<b>API Endpoint</b>	<a href="http://twitter.com/statuses/">http://twitter.com/statuses/</a>
<b>API Portal / Home Page</b>	<a href="https://dev.twitter.com/rest/public">https://dev.twitter.com/rest/public</a>
<b>Primary Category</b>	Social

6. In the new browser tab that opens, click Tweets in the left-side navigation.
7. In the left-side navigation, click Post, retrieve and engage with Tweets.



Search all documentation...

# Post, retrieve and engage with Tweets

## Basics

### Accounts and users

#### Tweets

[Post, retrieve and engage with Tweets](#)

Get Tweet timelines

Curate a collection of Tweets

Optimize Tweets with Cards

Search Tweets

Filter realtime Tweets

Sample realtime Tweets

Get batch historical Tweets

[Overview](#) [Guides](#) [API Reference](#)

#### Tweets

#### Retweets

#### Likes (formerly favorites)

The following API endpoints can be used to programmatically create, retrieve and delete Tweets, Retweets and Likes:

- POST statuses/update
- POST statuses/destroy/:id
- GET statuses/show/:id
- POST statuses/retweet/:id
- POST statuses/unretweet/:id
- GET statuses/retweets/:id
- GET statuses/retweets\_of\_me
- GET statuses/retweeters/ids

- POST favorites/create/:id
- POST favorites/destroy/:id
- GET favorites/list

8. Browse the list of requests you can make to the API.
9. Select the API Reference tab beneath the title of the page.

10. Review the information for POST statuses/update including parameters, example request, and example response.

The screenshot shows the Twitter Developer API documentation for the POST statuses/update endpoint. At the top, there's a navigation bar with links for Developer, Use cases, Products, Docs, More, Apply, a search icon, and a Sign In button. Below the navigation, there's a table with one row. The first column contains the parameter name 'fail\_dm\_commands'. The second column indicates it's 'optional'. The third column provides a detailed description: 'When set to true, causes any status text that starts with shortcode commands to return an API error. When set to false, allows shortcode commands to be sent in the status text and acted on by the API.' The fourth column shows the default value 'true' and the fifth column shows the type 'false'.

fail_dm_commands	optional	When set to <i>true</i> , causes any status text that starts with shortcode commands to return an API error. When set to <i>false</i> , allows shortcode commands to be sent in the status text and acted on by the API.	<i>true</i>	<i>false</i>
------------------	----------	--	-------------	--------------

## Example Request

POST <https://api.twitter.com/1.1/statuses/update.json?status=Maybe%20he%27ll%20finally%20find%20his%20keys.%20%23peterfalk>

## Example Response

```
{  
  "coordinates": null,  
  "favorited": false,  
  "created_at": "Wed Sep 05 00:37:15 +0000 2012",  
  "truncated": false,  
  "id_str": "243145735212777472",  
  "entities": {  
    "urls": [  
      {"url": "http://t.co/..."},  
      {"url": "http://t.co/..."}  
    ]  
  }  
}
```

11. Close the two browser tabs.

## Walkthrough 1-2: Make calls to an API

In this walkthrough, you make calls to a RESTful API. You will:

- Use Postman to make calls to an unsecured API (an implementation).
- Make GET, DELETE, POST, and PUT calls.
- Use Postman to make calls to a secured API (an API proxy).

The screenshot shows the Postman interface with the following details:

- Method:** POST
- URL:** http://training-american-ws.cloudhub.io/api/flights/
- Body (JSON):**

```
1 {  
2   "code": "GQ574",  
3   "price": 399,  
4   "departureDate": "2016/12/20",  
5   "origin": "ORD",  
6   "destination": "SFO",  
7   "emptySeats": 200,  
8   "plane": {"type": "Boeing 747", "totalSeats": 400}  
9 }
```
- Status:** 201 Created
- Time:** 132 ms
- Size:** 221 B

### Make GET requests to retrieve data

1. Return to or open Postman.
2. Make sure the method is set to GET.
3. Return to the course snippets.txt file.
4. Copy the URL for the American Flights web service:  
<http://training-american-ws.cloudhub.io/api/flights>.
5. Return to Postman and paste the URL in the text box that says Enter request URL.

The screenshot shows the Postman interface with the following details:

- Method:** GET
- URL:** http://training-american-ws.cloudhub.io/api/flights
- Type:** No Auth

- Click the Send button; you should get a response.
- Locate and click the return HTTP status code of 200.
- Review the response body containing flights to SFO, LAX, and CLE.

The screenshot shows the Postman interface with a successful GET request to `http://training-american-ws.cloudhub.io/api/flights`. The response body is displayed in Pretty JSON format, showing two flight objects:

```

1 [
2   {
3     "ID": 1,
4     "code": "rree0001",
5     "price": 541,
6     "departureDate": "2016-01-20T00:00:00",
7     "origin": "MUA",
8     "destination": "LAX",
9     "emptySeats": 0,
10    "plane": {
11      "type": "Boeing 787",
12      "totalSeats": 200
13    }
14  },
15  {
16    "ID": 2,
17    "code": "eefd0123",
18    "price": 300
19  }
]

```

- Click the Params button next to the URL.
- In the area that appears, set the key to destination and the value to CLE.
- Click the Send button; you should get just flights to CLE returned.

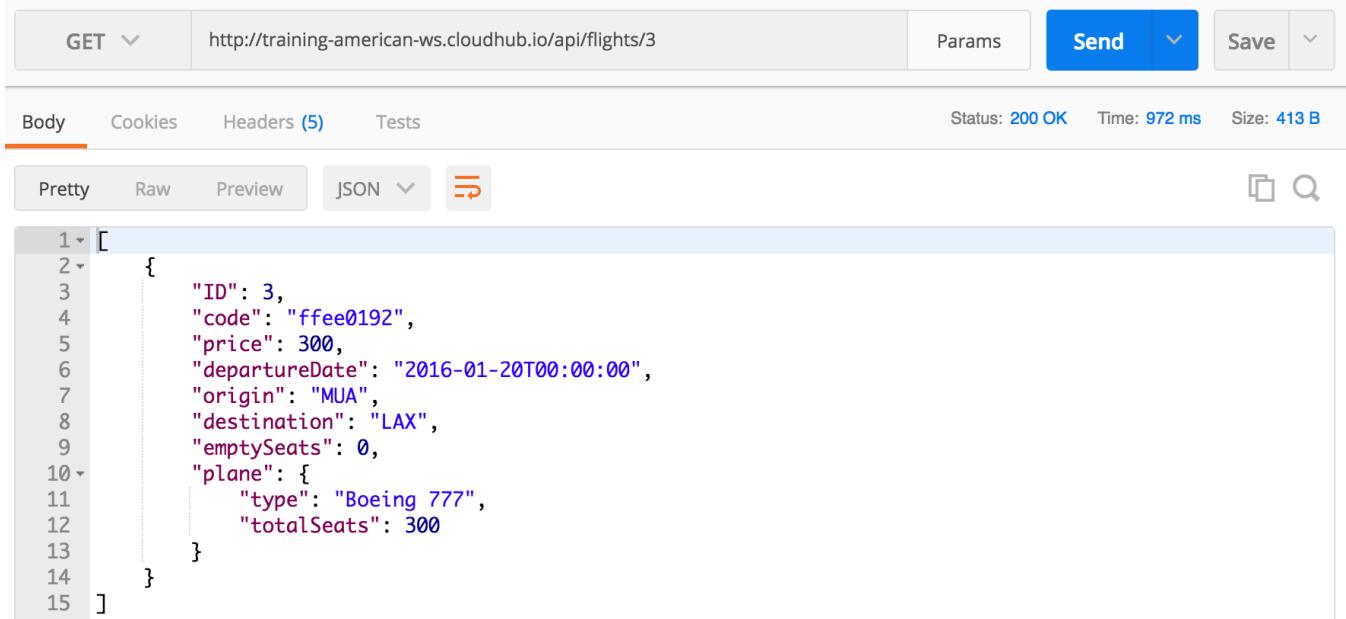
The screenshot shows the Postman interface with a filtered GET request to `http://training-american-ws.cloudhub.io/api/flights?destination=CLE`. The response body is displayed in Pretty JSON format, showing one flight object:

```

1 [
2   {
3     "ID": 2,
4     "code": "eefd0123",
5     "price": 300,
6     "departureDate": "2016-01-25T00:00:00",
7     "origin": "MUA",
8     "destination": "CLE",
9     "emptySeats": 7,
10    "plane": {
11      "type": "Boeing 787",
12      "totalSeats": 200
13    }
14  }
]

```

12. Click the X next to the parameter to delete it.
13. Change the request URL to use a uri parameter to retrieve the flight with an ID of 3:  
<http://training-american-ws.cloudhub.io/api/flights/3>
14. Click the Send button; you should see only the flight with that ID returned.



The screenshot shows a REST client interface. At the top, there's a header bar with 'GET' selected, the URL 'http://training-american-ws.cloudhub.io/api/flights/3', a 'Params' button, a 'Send' button, and a 'Save' button. Below the header, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Tests'. The 'Body' tab is active, showing a JSON response. The JSON is formatted with line numbers on the left:

```

1 [ {
2   "ID": 3,
3   "code": "ffee0192",
4   "price": 300,
5   "departureDate": "2016-01-20T00:00:00",
6   "origin": "MUA",
7   "destination": "LAX",
8   "emptySeats": 0,
9   "plane": {
10     "type": "Boeing 777",
11     "totalSeats": 300
12   }
13 }
14 ]
15 ]

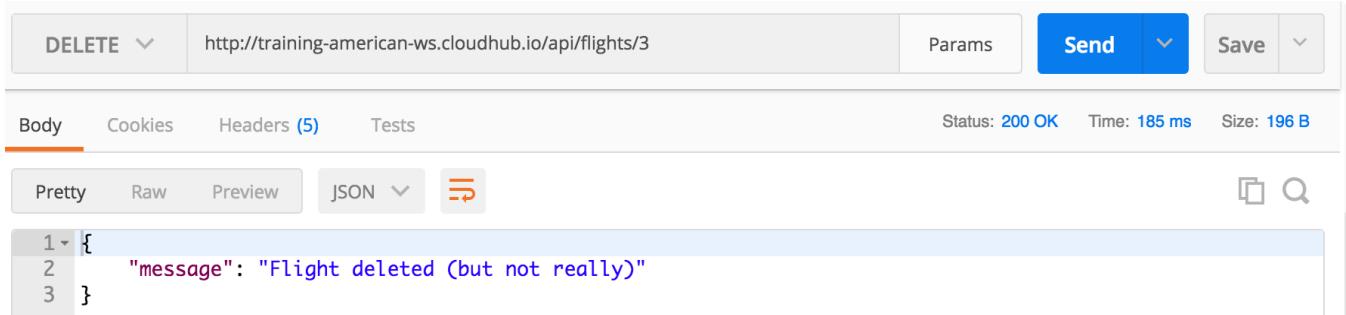
```

At the bottom right of the body panel are 'Pretty', 'Raw', 'Preview', and 'JSON' buttons, along with a search icon.

## Make DELETE requests to delete data

15. Change the method to DELETE.
16. Click the Send button; you should see a 200 response with a message that the Flight was deleted (but not really).

*Note: The database is not actually modified so that its data integrity can be retained for class.*



The screenshot shows a REST client interface. At the top, there's a header bar with 'DELETE' selected, the URL 'http://training-american-ws.cloudhub.io/api/flights/3', a 'Params' button, a 'Send' button, and a 'Save' button. Below the header, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Tests'. The 'Body' tab is active, showing a JSON response. The JSON is formatted with line numbers on the left:

```

1 { "message": "Flight deleted (but not really)"
2 }
3 }

```

At the bottom right of the body panel are 'Pretty', 'Raw', 'Preview', and 'JSON' buttons, along with a search icon.

17. Remove the URI parameter from the request: <http://training-american-ws.cloudhub.io/api/flights>.

18. Click the Send button; you should get a 405 response with a message of method not allowed.

The screenshot shows the Postman interface with a DELETE request to `http://training-american-ws.cloudhub.io/api/flights`. The response status is **405 Method Not Allowed**, Time: 106 ms, Size: 198 B. The Body tab is selected, showing a JSON response with the message "Method not allowed".

```
[{"message": "Method not allowed"}]
```

## Make a POST request to add data

19. Change the method to POST.

20. Click the Send button; you should get a 415 response with a message of unsupported media type.

The screenshot shows the Postman interface with a POST request to `http://training-american-ws.cloudhub.io/api/flights`. The response status is **415 Unsupported Media Type**, Time: 103 ms, Size: 206 B. The Body tab is selected, showing a JSON response with the message "Unsupported media type".

```
[{"message": "Unsupported media type"}]
```

21. Click the Headers link under the request URL.

22. Click in the Headers key field, type C, and then select Content-Type.

23. Click in the Value field, type A, and then select application/json.

The screenshot shows the Postman interface with a POST request to `http://training-american-ws.cloudhub.io/api/flights`. The Headers tab is selected, showing a table with one entry: Content-Type: application/json. There is also a New key row with a Value column and a Description column.

Key	Value	Description
Content-Type	application/json	
New key	Value	Description

24. Click the Body link under the request URL.

25. Select the raw radio button.

26. Return to the course snippets.txt file and copy the value for American Flights API post body.

27. Return to Postman and paste the code in the body text area.

The screenshot shows the Postman interface with a POST request to `http://training-american-ws.cloudhub.io/api/flights`. The Body tab is selected, showing a JSON payload:

```
1 [ {  
2   "code": "GQ574",  
3   "price": 399,  
4   "departureDate": "2016/12/20",  
5   "origin": "ORD",  
6   "destination": "SFO",  
7   "emptySeats": 200,  
8   "plane": {"type": "Boeing 747", "totalSeats": 400}  
9 } ]
```

28. Click the Send button; you should see a 201 response with the message Flight added (but not really).

The screenshot shows the Postman interface after sending the request. The response status is `201 Created`. The Body tab shows the response payload:

```
1 {  
2   "message": "Flight added (but not really)"  
3 }
```

29. Return to the request body and remove the plane field and value from the request body.

30. Remove the comma after the emptySeats key/value pair.

The screenshot shows the Postman interface with the request body modified. The `emptySeats` value now ends with a period instead of a comma:

```
1 {  
2   "code": "GQ574",  
3   "price": 399,  
4   "departureDate": "2016/12/20",  
5   "origin": "ORD",  
6   "destination": "SFO",  
7   "emptySeats": 200  
8 }
```

31. Send the request; the message should still post successfully.

32. In the request body, remove the emptySeats key/value pair.

33. Delete the comma after the destination key/value pair.

The screenshot shows the Postman interface with the 'Body' tab selected. The request method is POST and the URL is <http://training-american-ws.cloudhub.io/api/flights>. The 'Body' section is set to JSON (application/json). The JSON payload is:

```
1 {  
2   "code": "GQ574",  
3   "price": 399,  
4   "departureDate": "2016/12/20",  
5   "origin": "ORD",  
6   "destination": "SFO"  
7 }
```

The last character of the 'destination' value ('SFO') has a red underline, indicating a syntax error.

34. Send the request; you should see a 400 response with the message Bad request.

The screenshot shows the Postman interface after sending the request. The status bar indicates Status: 400 Bad Request, Time: 104 ms, and Size: 206 B. The response body is:

```
1 {  
2   "message": "Bad request"  
3 }
```

## Make a PUT request to update data

35. Change the method to PUT.

36. Add a flight ID to the URL to modify a particular flight.

37. Click the Send button; you should get a bad request message.

38. In the request body field, press Cmd+Z or Ctrl+Z so the emptySeats field is added back.

39. Send the request; you should see the response Flight updated (but not really).

The screenshot shows the Postman interface with a PUT request to `http://training-american-ws.cloudhub.io/api/flights/3`. The Body tab is selected, containing the following JSON payload:

```
1 {  
2   "code": "GQ574",  
3   "price": 399,  
4   "departureDate": "2016/12/20",  
5   "origin": "ORD",  
6   "destination": "SFO",  
7   "emptySeats": 200  
8 }
```

The response status is 200 OK, with a message: "Flight updated (but not really)".

## Make a request to a secured API

40. Change the method to GET.

41. Change the request URL to <http://training-american-api.cloudhub.io/flights/3>.

*Note: The -ws in the URL has been changed to -api and the /api removed.*

42. Click the Send button; you should get a message about a missing client\_id.

The screenshot shows the Postman interface with a GET request to `http://training-american-api.cloudhub.io/flights`. The Body tab is selected, containing the following text:

```
1 Unable to retrieve client_id from message
```

The response status is 401 Unauthorized, with the message: "Unable to retrieve client\_id from message".

43. Return to the course snippets.txt file and copy the value for the American Flights API client\_id.

44. Return to Postman and add a request parameter called client\_id.

45. Set client\_id to the value you copied from the snippets.txt file.

46. Return to the course snippets.txt file and copy the value for the American Flights API client\_secret.

47. Return to Postman and add a request parameter called client\_secret.

48. Set client\_secret to the value you copied from the snippets.txt file.

49. Click the Send button; you should get data for flight 3 again.

*Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.*

The screenshot shows the Postman interface with a successful API call. The request method is GET, the URL is [http://training-american-api.cloudhub.io/flights/3?client\\_id=d1374b15c6864c3...](http://training-american-api.cloudhub.io/flights/3?client_id=d1374b15c6864c3...), and the response status is 200 OK. The response body is a JSON object representing flight 3:

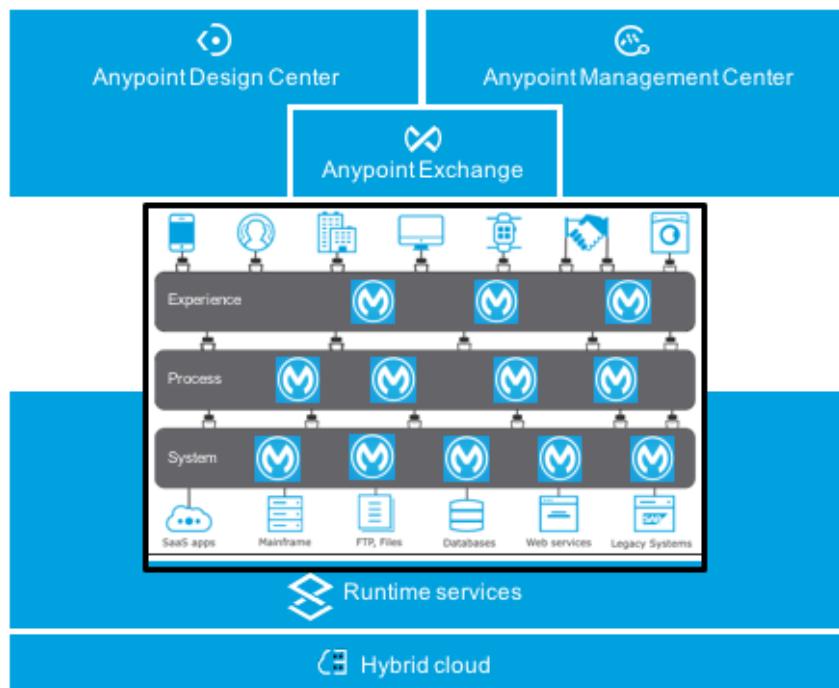
```
1  [
2   {
3     "ID": 3,
4     "code": "ffee0192",
5     "price": 300,
6     "departureDate": "2016-01-20T00:00:00",
7     "origin": "MUA",
8     "destination": "LAX",
9     "emptySeats": 0,
10    "plane": {
11      "type": "Boeing 777",
12      "totalSeats": 300
13    }
14  }
15 ]
```

50. Click the Send button three more times; you should get a 429 response and an API calls exceeded message.

*Note: The API service level agreement (SLA) for the application with this client ID and secret has been set to allow three API calls per minute.*

The screenshot shows the Postman interface with a 429 Too Many Requests error. The response body is a simple text message: "API calls exceeded".

# Module 2: Introducing Anypoint Platform



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

- Identify all the components of Anypoint Platform.
- Describe the role of each component in building application networks.
- Navigate Anypoint Platform.
- Locate APIs and other assets needed to build integrations and APIs in Anypoint Exchange.
- Build basic integrations to connect systems using flow designer.

# Walkthrough 2-1: Explore Anypoint Platform and Anypoint Exchange

In this walkthrough, you get familiar with the Anypoint Platform web application. You will:

- Explore Anypoint Platform.
- Browse Anypoint Exchange.
- Review an API portal for a REST API in Exchange.
- Discover and make calls to the Training: American Flights API.

The screenshot shows the Anypoint Exchange interface. On the left, there's a sidebar with 'Assets list' and a navigation tree for 'Training: American Flights API'. The main area displays the API summary for 'Training: American Flights API | 1.0'. It includes a star rating of 5 stars from 7 reviews, a thumbnail icon, and a link to '/flights : Get all flights'. Below this, under 'Request', is a GET method to 'http://training-american-ws.cloudhub.io/api/flights'. Under 'Parameters', there's a table for 'Query parameters' with a single entry: 'destination' (string, enum) with possible values SFO, LAX, CLE. Under 'Response', it shows a 200 OK status with a sample JSON response:

```
[Array[2]
  -0: {
    "ID": 1,
    "code": "ER38sd",
    "price": 400,
    "departureDate": "2017/07/26",
    "origin": "CLE",
    "destination": "SFO",
    "emptySeats": 1
  }
]
```

## Return to Anypoint Platform

1. Return to Anypoint Platform in a web browser.

*Note: If you closed the browser window or logged out, return to <https://anypoint.mulesoft.com> and log in.*

2. Click the menu button located in the upper-left in the main menu bar.

3. In the menu that appears, select Anypoint Platform; this will return you to the home page.

*Note: This will be called the main menu from now on.*

The screenshot shows the Anypoint Platform home page. On the left is a vertical sidebar with icons and labels: Design Center, Exchange, Management Center, Access Management, API Manager, Runtime Manager, and Data Gateway. The main content area has sections for 'Information' (Tier: Trial, Status: Expires on 12/12/2017), 'Sandboxes' (Test environments for QA of applications), 'vCores' (0 / 0), and 'Static IP' (0 / 1).

## Explore Anypoint Platform

4. In the main menu, select Access Management.
5. In the main menu, select Design Center.
6. In the main menu, select Runtime Manager.
7. If you get a Choose environment page, select Design.

The screenshot shows the 'Runtime Manager' page with a 'Choose Environment' dropdown. The 'Design' option is highlighted. Below it, there is a note: 'Open your profile to set the default environment...'.

8. In the main menu, select API Manager.

## Explore Anypoint Exchange

9. In the main menu, select Exchange.

10. In the left-side navigation, click MuleSoft; you should see all the content in the public Exchange.

The screenshot shows the Exchange interface with the 'MuleSoft' organization selected in the left sidebar. The main area displays three connectors: 'LDAP Connector', 'Amazon RDS Connector', and 'Amazon EC2 Connector', each with a five-star rating. A search bar and a 'New' button are visible at the top right.

11. In the left-side navigation, click the name of your organization beneath MuleSoft (Training in the screenshots); you should now see only the content in your private Exchange, which is currently empty.

The screenshot shows the Exchange interface with the 'Training' organization selected in the left sidebar. The main area is currently empty, indicating no content in the private Exchange.

12. In the left-side navigation, click All.

13. In the types menu, select Connectors.

The screenshot shows the Exchange interface with the 'All' option selected in the left sidebar. In the top navigation bar, the 'Connectors' type is selected. The main area displays the same three connectors as in the previous screenshot: 'LDAP Connector', 'Amazon RDS Connector', and 'Amazon EC2 Connector', each with a five-star rating.

14. Click one of the connectors and review its information.
15. In the left-side navigation, click the Assets list link.
16. Select the Salesforce connector (or search for it if it is not shown) and review its details.

*Note: The Salesforce connector is used in the Development Fundamentals course.*

The screenshot shows the Anypoint Exchange interface. The top navigation bar includes 'Training', a question mark icon, and a user profile icon. The main content area is titled 'Salesforce Connector' with a 5-star rating and '(0 reviews)'. It features a 'Rate and review' button. Below the title, there's a brief description: 'The Anypoint Salesforce Connector enables businesses to connect to Salesforce in a user friendly manner by leveraging Salesforce APIs in the backend.' A larger text block explains: 'Using this connector, businesses can create instant connectivity between Salesforce and popular ERP, analytics, billing, marketing automation, social applications and services.' Another block discusses MuleSoft's integration solutions. On the right side, there are sections for 'Download', 'Dependency Snippets', 'Overview' (with a 'Connector' type), 'Created By' (MuleSoft Organization), 'Published On' (Nov 9, 2017), and 'Versions' (listing 8.4.0 and 8.3.1).

17. In the left-side navigation, click Assets list.
18. In the types menu, select Templates (and remove anything from the search field).

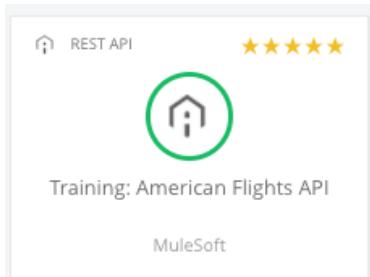
## Browse REST APIs in Anypoint Exchange

19. In the types menu, select REST APIs.
20. Browse the APIs.

The screenshot shows the 'All assets' page in Anypoint Exchange. The left sidebar has a 'Types' menu with 'All' selected, followed by 'MuleSoft', 'Training', 'My applications', and 'Public Portal'. The main area is titled 'All assets' and shows a grid of three REST API cards. Each card includes a small icon, the API name, a star rating, and the provider. The first card is 'Optymyze API' by MuleSoft. The second is 'CardConnect REST API' by MuleSoft. The third is 'Nexmo SMS API' by MuleSoft. A 'New' button is visible in the top right corner of the grid.

## Discover and review the API portal for the Training: American Flights API

21. Locate and click the Training: American Flights API.



22. Review the API portal.

A screenshot of the MuleSoft API Exchange interface. On the left, there's a sidebar with navigation links like 'Assets list', 'Training: American Flights API' (which is highlighted), 'API summary', 'Types', 'Resources', '/flights' (with 'GET Get all flig...', 'POST Add a fli...', 'PUT Updat...', 'DELETE De...'), and 'API instances'. The main content area shows the API details for 'Training: American Flights API | 1.0'. It includes a green house icon, five yellow stars, '(7 reviews)', and 'Rate and review'. Below this, it says 'This example RAML 1.0 API is used in MuleSoft training courses.' and 'It defines the basic GET, POST, DELETE, and PUT operations for flights and uses data type and example fragments.' A code editor window shows the RAML 1.0 API definition. The right side of the screen has sections for 'Download', 'Overview' (with fields for Type: REST API, Created By: MuleSoft Organization, Published On: Sep 14, 2017, Visibility: Public), and 'Asset versions for 1.0' (listing 1.0.1 and 1.0.0).

23. In the left-side navigation, expand and review the list of available resources.

24. Click the GET link for the /flights resource.

25. On the /flights: Get all flights page, review the information for the optional query parameter.

The screenshot shows the MuleSoft Anypoint Platform interface. On the left, there's a sidebar with 'Assets list' and 'Training: American Flights API'. The main area displays the 'Training: American Flights API | 1.0' page. It features a green circular icon with a house symbol, a 5-star rating (7 reviews), and the endpoint '/flights : Get all flights'. Below this, under 'Request', is the GET method at `http://training-american-ws.cloudhub.io/api/flights`. Under 'Parameters', there's a table:

Parameter	Type	Description
destination	string (enum)	Destination airport code Possible values: SFO, LAX, CLE

On the right, there's a 'Mocking Service' section with the URL `https://mocksvc-proxy.anypoint.mulesoft.com/exc`, tabs for 'Parameters' (selected) and 'Headers', and a 'Send' button. A checkbox 'Show optional parameters' is checked.

26. Scroll down and locate the type and details for the data returned from a call.

### Response

The screenshot shows the response schema for the /flights endpoint. It includes a '200' status indicator, a 'Type application/json' header, and tabs for 'Type' and 'Examples'. The 'Type' tab is selected, showing a JSON schema:

```
[  
  {  
    "ID": "integer",  
    "code": "string",  
    "price": "number",  
    "departureDate": "string",  
    "origin": "string",  
    "destination": "string",  
    "emptySeats": "integer",  
    "plane": {  
      "type": "string",  
      "totalSeats": "integer"  
    }  
  }  
]
```

Below the schema, there's a table for parameters:

Parameter	Type	Description
Array of AmericanFlight items		
item. ID	integer	

27. Select the Examples tab; you should see an example response.

## Use the API console to make calls to the Training: American Flights API

28. Scroll up and locate the API console on the right-side on the page.
29. Select to show optional query parameters and select a value.
30. Select a destination and click Send; you should get the example data returned using the mocking service.

The screenshot shows the MuleSoft API console interface. At the top, a green button labeled "GET" is followed by the text "Get all flights". Below this, a dropdown menu is set to "Mocking Service". The URL field contains "https://mocksvc-proxy.anypoint.mulesoft.com/exc". There are two tabs: "Parameters" (which is selected) and "Headers". Under "Parameters", there is a section for "Query parameters" with a dropdown containing "LAX". A checkbox labeled "Show optional parameters" is checked. A blue "Send" button is located below the parameters section. At the bottom, the response status is shown as "200 OK" and "295.75 ms". To the right of the status, there is a "Details" link with a dropdown arrow. Below the status, there are four icons: a checkmark, a download arrow, a copy icon, and a grid icon. The response body is displayed in a code block:

```
[Array[2]
-0: {
  "ID": 1,
  "code": "ER38sd",
  "price": 400,
  "departureDate": "2017/07/26",
  "origin": "CLE",
  "destination": "SFO",
  "emptySeats": 0,
```

31. In the API console, change the API instance from Mocking Service to RAML Base URI.

32. Click Send again; you should get results from the actual API implementation for the destination you selected.

GET Get all flights

RAML Base URI

http://training-american-ws.cloudhub.io/api/flights

Parameters Headers

Query parameters  Show optional parameters

LAX

Send

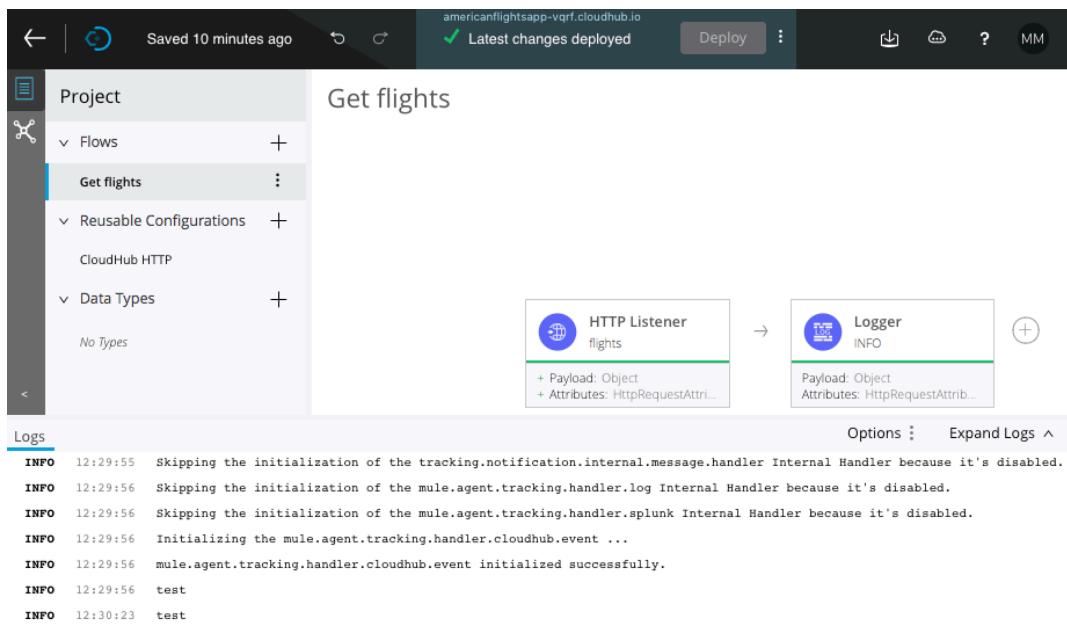
200 OK 1310.77 ms Details ▾

```
[Array[3]
-0: {
  "ID": 1,
  "code": "rree0001",
  "price": 541,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
```

## Walkthrough 2-2: Create a Mule application with flow designer

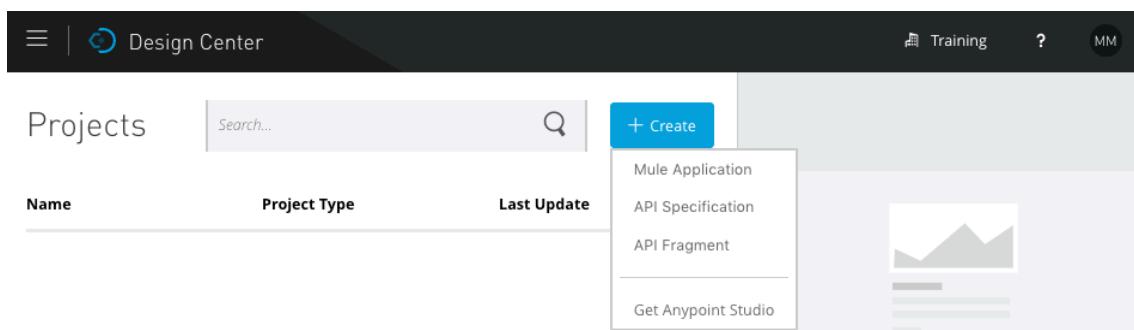
In this walkthrough, you build, run, and test a basic Mule application with flow designer. You will:

- Create a new Mule application project in Design Center.
- Create an HTTP trigger for a flow in the application.
- Add a Logger component.
- Deploy, run, and test the application.
- View application information in Runtime Manager.



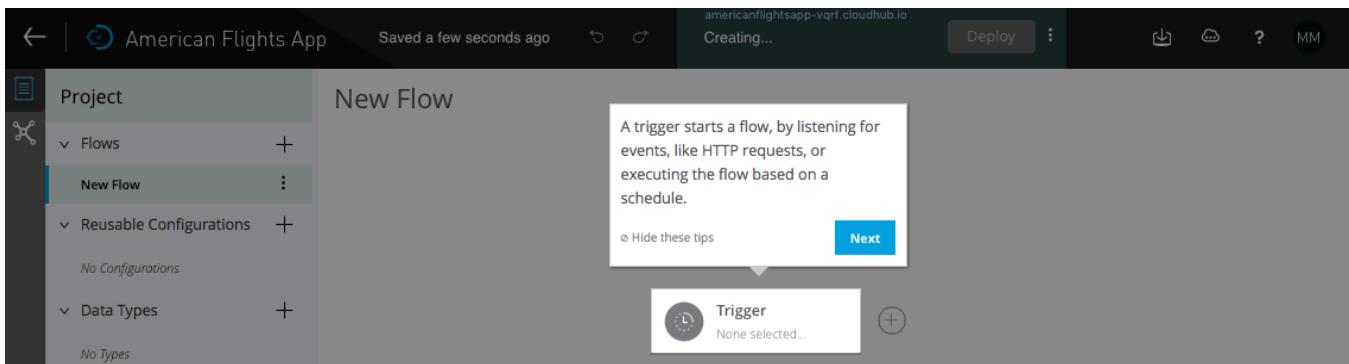
### Create a Mule application project in Design Center

1. Return to Anypoint Platform.
2. In the main menu, select Design Center.
3. Click the Create button and select Mule Application.

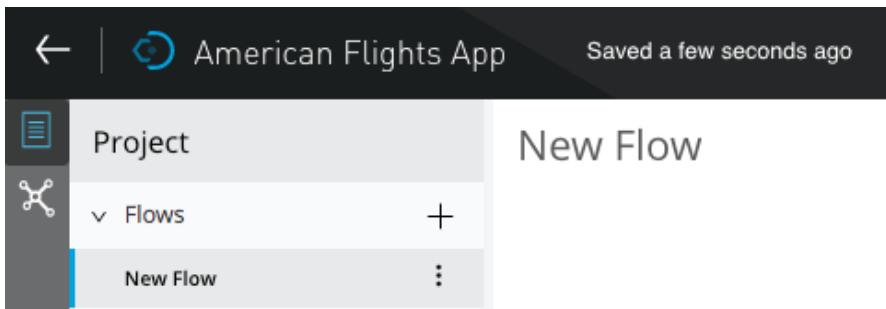


4. In the New Mule Application dialog box, set the project name to American Flights App.

- Click Create; flow designer should open.



- In the pop-up box in flow designer, click Hide these tips.
- Click the arrow icon in the upper-left corner; you should return to the Design Center.



- In the Design Center project list, click the row containing the American Flights App; you should see information about the project displayed on the right side of the page.

Name	Project Type	Last Update
American Flights App	Mule Application	November 19th, 2017

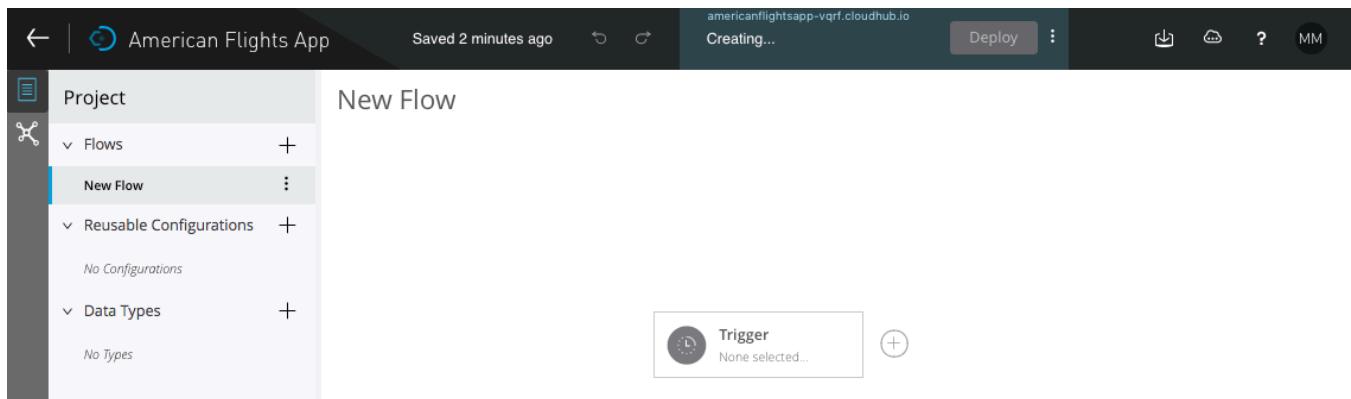
**American Flights App**

**Details**

Name	American Flights App
Modified	November 19th, 2017
Created	November 19th, 2017
Created by	maxmule00
Environment	b22ae821-3871-48a3-a1a7-4777cd3b520a
Status	Creating...
Deployment url	americanflightsapp-vqrf.cloudhub.io

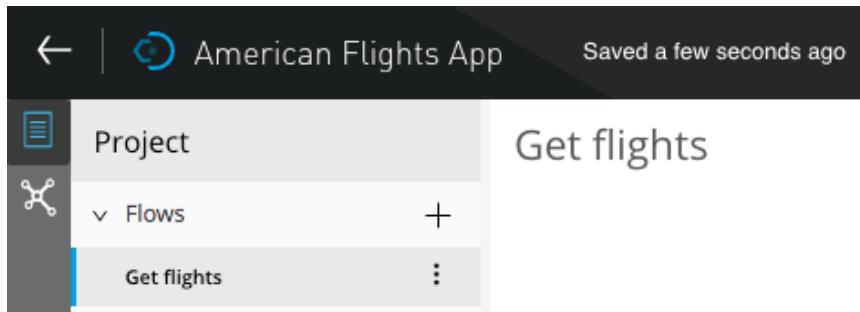
**Open**

9. Click the Open button or click the American Flights App link in the project list; the project should open in flow designer.



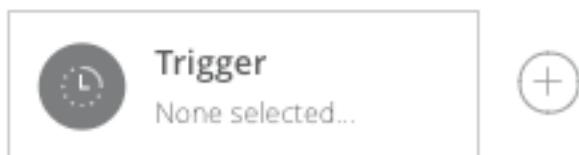
## Rename the flow

10. Locate New Flow in the project explorer.
11. Click its option menu and select Rename.
12. In the Rename Flow dialog box, set the name to Get flights and click OK.

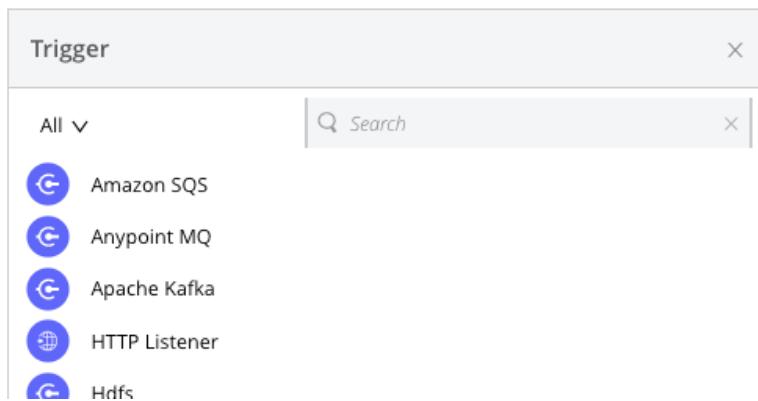


## Create an HTTP trigger for a flow in the application

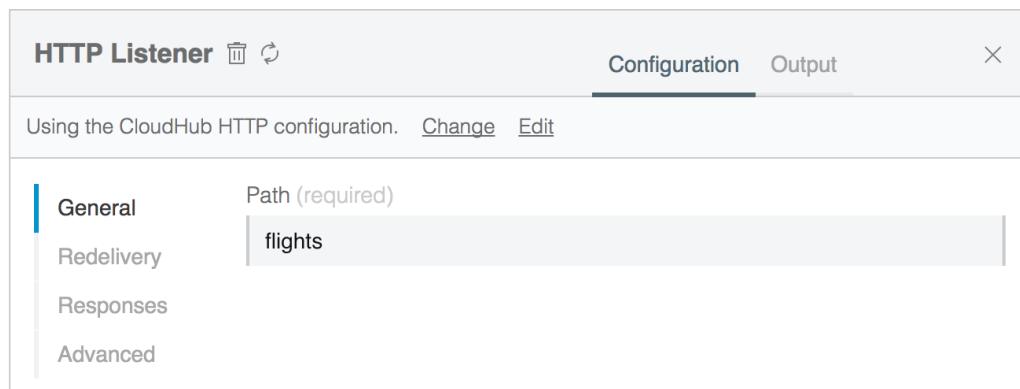
13. In flow designer, click the Trigger card.



14. In the Trigger card, select HTTP Listener.

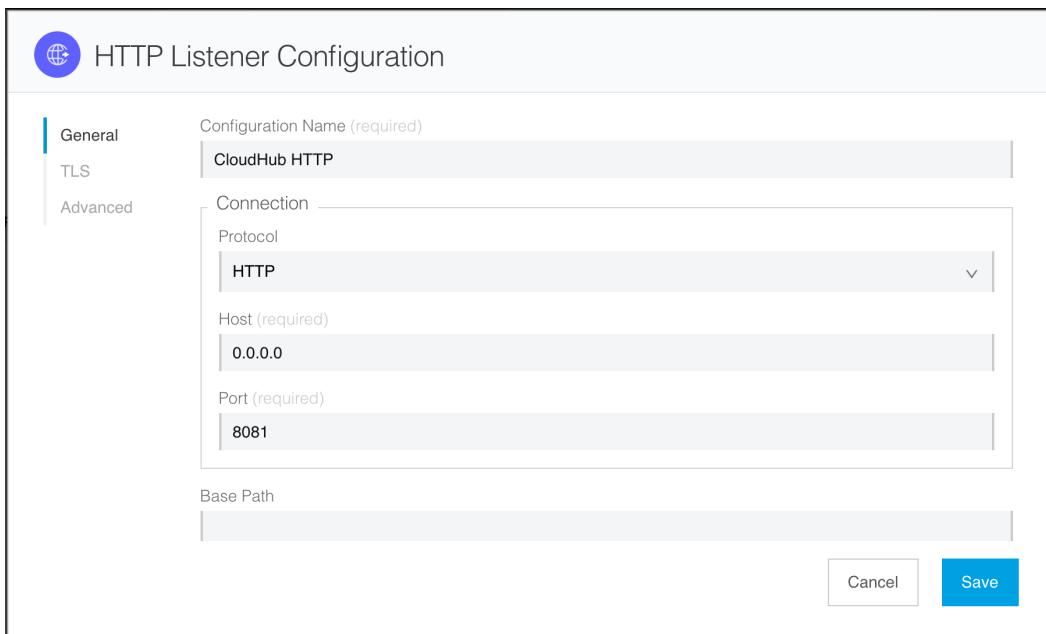


15. In the HTTP Listener dialog box, set the path to flights.



16. Click the Edit link for the CloudHub HTTP configuration.

17. In the HTTP Listener Configuration dialog box, review the information and click Cancel.



18. In the HTTP Listener dialog box, click the close button in the upper-right corner.

The screenshot shows the 'HTTP Listener' configuration dialog. At the top, there are tabs for 'Configuration' and 'Output'. Below that, it says 'Using the CloudHub HTTP configuration.' with links to 'Change' and 'Edit'. On the left, there are two tabs: 'General' (which is selected) and 'Redelivery'. The 'Path (required)' field contains 'flights'. There is also a large text input field below it containing 'flights'.

## Add a Logger

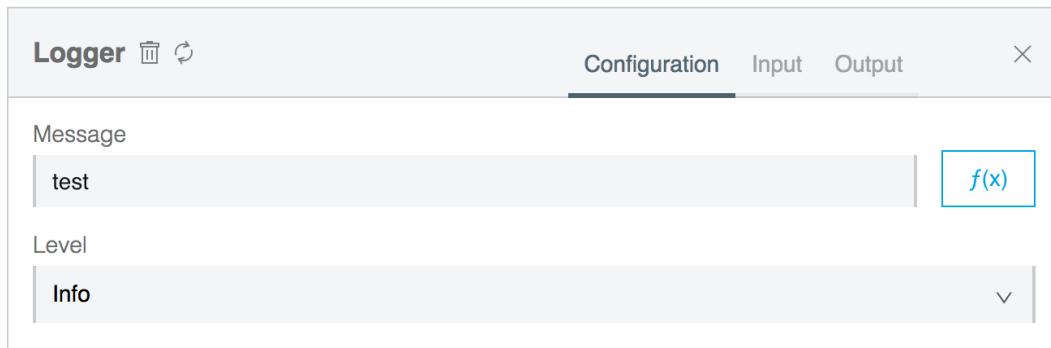
19. Click the add button next to the HTTP Listener card.

The screenshot shows a card for an 'HTTP Listener' named 'flights'. To the right of the card is a blue circular button with a white plus sign (+). Below the card, there are two green plus signs followed by text: '+ Payload: Object' and '+ Attributes: HttpRequestAttri...'. A small 'x' icon is located at the top right of the card.

20. In the Select a component dialog box, select Logger.

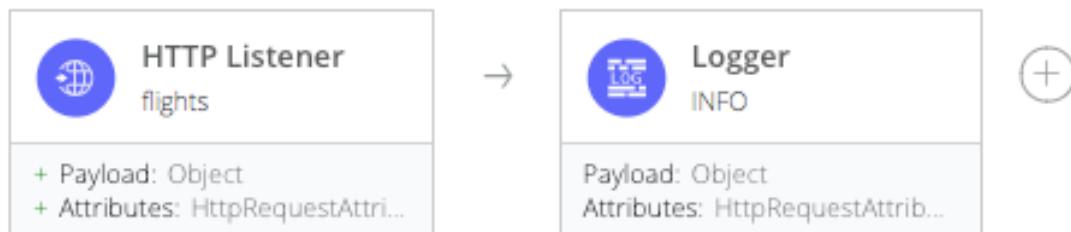
The screenshot shows the 'Select a component' dialog. At the top, it says 'Select a component' and has a search bar with a magnifying glass icon and a clear 'x' button. Below that, there is a dropdown menu set to 'All' and a 'Search' field. The list of components includes: Inventory API | R..., LDAP, LinkedIn RAML, Location API | RA..., Logger (with a description: 'Performs logging using an expression th...'), MongoDB, NetSuite, Nexmo SMS API, ObjectStore, and Omnichannel API... Each item has a blue circular icon with a white letter 'C' next to it.

21. In the Logger dialog box, set the message to test.



22. Close the card.

23. Notice that there are gray lines across both cards.



## Deploy the application

24. Click the Logs tab located in the lower-left corner of the window; you should see that your application is already started.

```
INFO 12:20:45 ****
* Application: americanflightsapp-vqrf
* OS encoding: UTF-8, Mule encoding: UTF-8
*
*****
SYSTEM 12:20:46 Worker(34.229.163.174): Your application has started successfully.
SYSTEM 12:20:47 Your application is started.
```

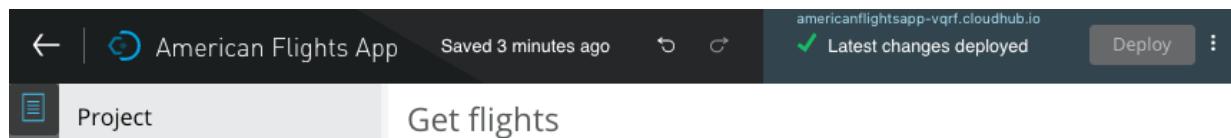
25. Locate the application status in the main menu bar; it should say Ready to deploy.

26. Look at the generated URL for the application.

*Note: The application name is appended with a four-letter suffix to guarantee that it is unique across all applications on CloudHub.*

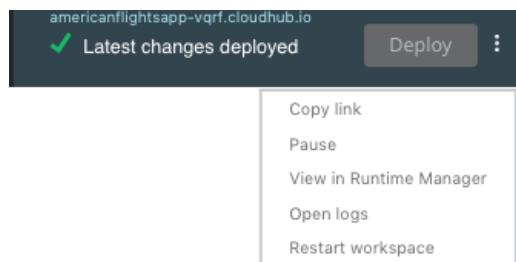
27. Click the Deploy button.

28. Watch the application status; it should change from Ready to Deploying then to Latest changes deployed.



*Note: If your application fails to deploy, look at the messages in the Logs panel. If there is no message about incorrect syntax, try restarting the workspace by clicking the options menu in the application status area and selecting Restart workspace.*

29. Click the options menu in the application status area and select Copy link.



## Test the application

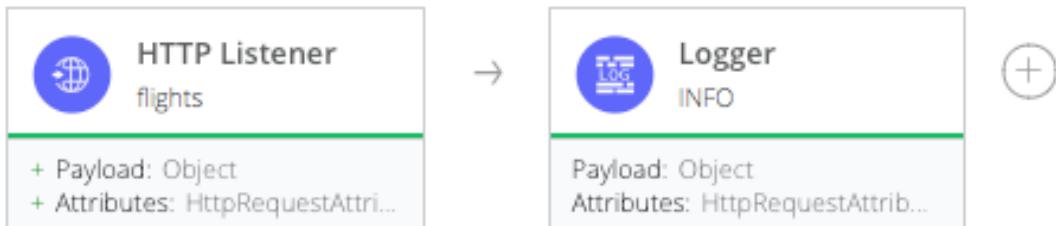
30. Return to Postman, paste the copied link, and click Send; you should get a No listener for endpoint: / message.

The screenshot shows a Postman request configuration. The method is set to 'GET' and the URL is 'americanflightsapp-vqrfl.cloudhub.io'. The response body is displayed as a single line: '1 No listener for endpoint: /'.

31. Add /flights to the path and click Send; you should get a 200 response with no body.

The screenshot shows a Postman request configuration. The method is set to 'GET' and the URL is 'americanflightsapp-vqrfl.cloudhub.io/flights'. The response body is empty, indicated by a single character '1'.

32. Click Send again to make a second request.
33. Return to flow designer.
34. Notice that there are now green lines across both cards.



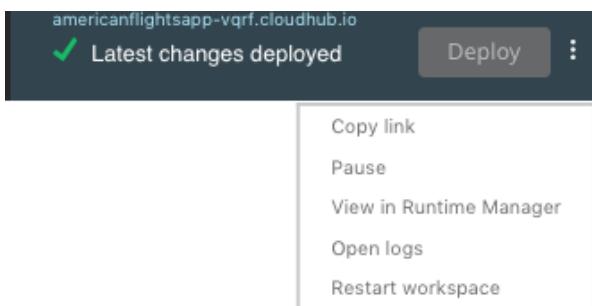
35. Look at the logs; you should see your Logger message displayed twice.

**Logs**

<b>INFO</b>	12:29:55	Skipping the initialization of the tracking.notification.internal.mess...
<b>INFO</b>	12:29:56	Skipping the initialization of the mule.agent.tracking.handler.log Int...
<b>INFO</b>	12:29:56	Skipping the initialization of the mule.agent.tracking.handler.splunk...
<b>INFO</b>	12:29:56	Initializing the mule.agent.tracking.handler.cloudhub.event ...
<b>INFO</b>	12:29:56	mule.agent.tracking.handler.cloudhub.event initialized successfully.
<b>INFO</b>	12:29:56	test
<b>INFO</b>	12:30:23	test

## View the application in Runtime Manager

36. Click the options menu in the application status area and select View in Runtime Manager; Runtime Manager should open in a new tab.



37. In the new browser tab that opens with Runtime Manager, review the application log file; you should see your test log messages.

The screenshot shows the Runtime Manager interface. On the left, there's a navigation sidebar with options like DESIGN, Applications (Dashboard, Insight, Logs, Object Store, Queues, Schedules, Settings), and a search bar. The main area displays the application 'americanflightsapp-vqrf' with a 'Live Console' tab. The console shows several log entries:

```
12:29:56.012 11/19/2017 Worker-0 http.listener.01:  
http.listener @77b46631 SelectorRunner INFO  
Initializing the mule.agent.tracking.handler.cloudhub.event ...  
  
12:29:56.032 11/19/2017 Worker-0 http.listener.01:  
http.listener @77b46631 SelectorRunner INFO  
mule.agent.tracking.handler.cloudhub.event initialized successfully.  
  
12:29:56.564 11/19/2017 Worker-0 [MuleRuntime].io.02:  
[americanflightsapp-vqrf].get_flights.BLOCKING @2eb8b9de INFO  
test  
  
12:30:23.324 11/19/2017 Worker-0 [MuleRuntime].io.02:  
[americanflightsapp-vqrf].get_flights.BLOCKING @2eb8b9de INFO  
test
```

To the right, there's a 'Deployments' panel showing a deployment for 'Worker-0' at 12:16. The 'Logs' section shows the 'System Log' for 'Worker-0'.

38. In the left-side navigation, click Settings.

39. Review the settings page and locate the following information for the application:

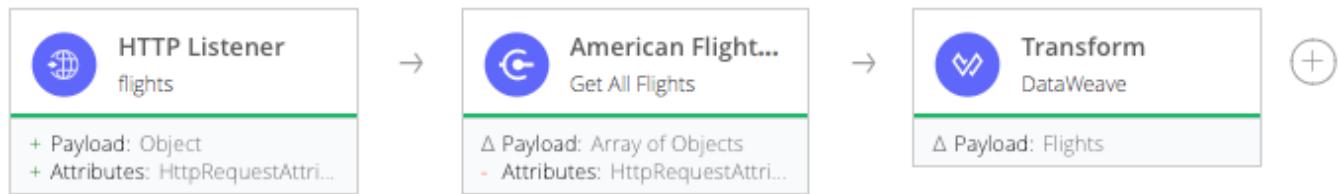
- To which environment it was deployed
- To what type of Mule runtime it was deployed
- To what size worker it was deployed

The screenshot shows the Runtime Manager interface with the 'Settings' option selected in the sidebar. The main area displays the application 'americanflightsapp-vqrf' with the 'Application File' set to 'americanflightsapp-vqrf.jar'. Below this, it shows 'Last Updated' as 2017-11-19 12:20:46PM and 'App url' as [americanflightsapp-vqrf.cloudhub.io](http://americanflightsapp-vqrf.cloudhub.io). The 'Runtime' tab is active, showing 'Runtime version' as 4.0.0, 'Worker size' as 0.2 vCores, and 'Workers' as 1. There are also checkboxes for 'Automatically restart application when not responding', 'Persistent queues', and 'Encrypt persistent queues'.

## Walkthrough 2-3: Create an integration application with flow designer that consumes an API

In this walkthrough, you build an integration application to consume an API from Anypoint Exchange. You will:

- Examine Mule event data for calls to an application.
- Use the American Flights API in Anypoint Exchange to get all flights.
- Transform data returned from an API to another format.



### Review Mule event data for the calls to the application

1. Return to the American Flights App in flow designer.
2. Click the title bar of the Logs panel to close it.
3. Expand the HTTP Listener card.
4. Click the Output tab.
5. Locate the Show drop-down menu that currently has Payload selected.

HTTP Listener		Configuration	Output	X
Show: <b>Payload</b> ▾	Data type: <b>Object</b>	v	Add Edit	X
History				
Dec 05, 2017 09:17am				
Dec 05, 2017 09:17am				
 No Payload				

- Locate your two calls to the application in the History panel; there should be no message payload for either call.
- Change the Show drop-down menu to Attributes.
- Review the attributes for the Mule event leaving the HTTP Listener processor.

```

HTTP Listener Configuration Output ×
Show: Attributes ▾
History Jul 25, 2017 05:27pm
{
  "listenerPath": "/flights",
  "relativePath": "/flights",
  "version": "HTTP/1.1",
  "scheme": "http",
  "method": "GET",
  "requestUri": "/flights",
  "queryString": "",
  "remoteAddress": "/54.161.15.67:59944",
  "queryParams": {},
  "uriParams": {},
  "requestPath": "/flights",
  "headers": {
    "x-forwarded-port": "80",
    "user-agent": "PostmanRuntime/6.1.6",
    "postman-token": "c00be330-3476-492d-87f7-291c783579b8",
    "x-forwarded-for": "73.231.218.202",
    "accept-encoding": "gzip, deflate",
    "cache-control": "no-cache",
    "x-real-ip": "73.231.218.202",
    "host": "americanflightsapp-wbsq.cloudhub.io",
    "accept": "*/*",
    "x-forwarded-proto": "http",
    "content-type": "application/json"
  }
}

```

- Close the card.
- Open the Logger card.
- Click the Input tab.
- Review the payload and attributes values for the two calls.

```

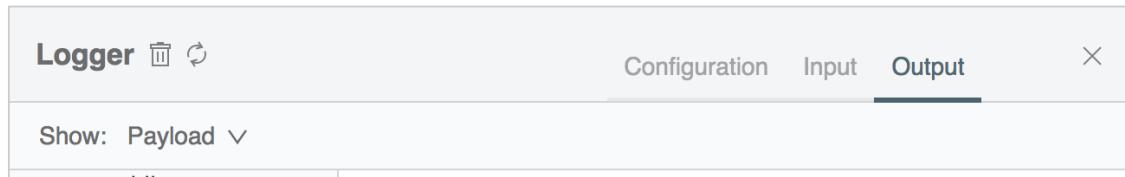
Logger Configuration Input Output ×
Show: Attributes ▾
History Jul 25, 2017 05:27pm
{
  "listenerPath": "/flights",
  "relativePath": "/flights",
  "version": "HTTP/1.1",
  "scheme": "http",
  "method": "GET",
  "requestUri": "/flights",
  "queryString": "",
  "remoteAddress": "/54.161.15.67:59944",
  "queryParams": {},
  "uriParams": {},
  "requestPath": "/flights",
  "headers": {

```

- Click the Output tab and review the payload and attributes values for the calls.

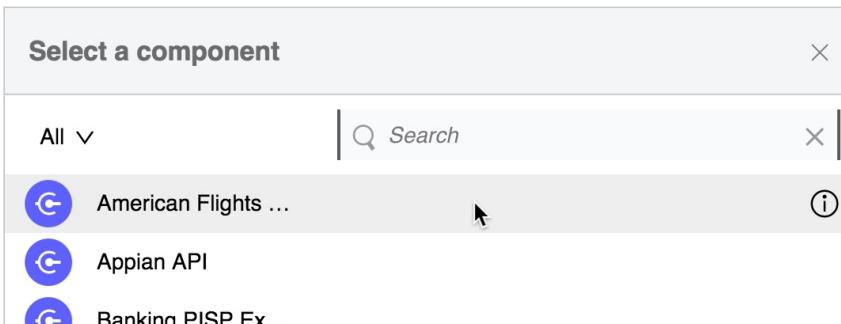
## Delete a card

14. Click the Remove button at the top of the card.

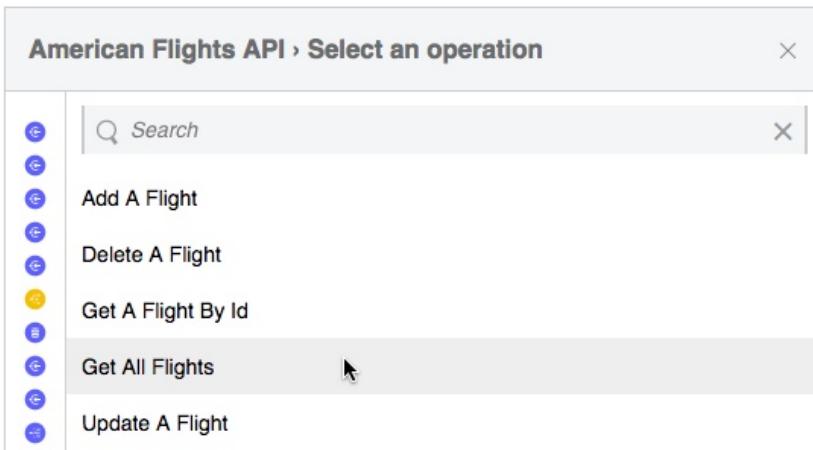


## Use the American Flights API in Anypoint Exchange to get all flights

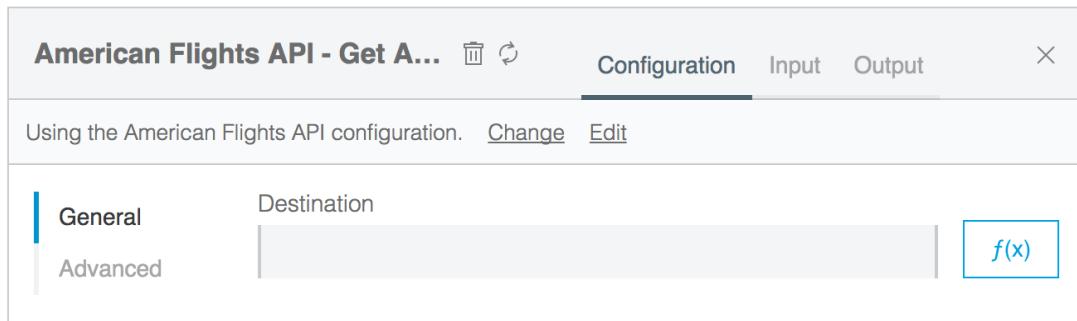
15. Click the Add button next to the HTTP Listener card.
16. In the Select a component dialog box, select the American Flights API.



17. In the American Flights API > Select an operation dialog box, select Get All Flights.



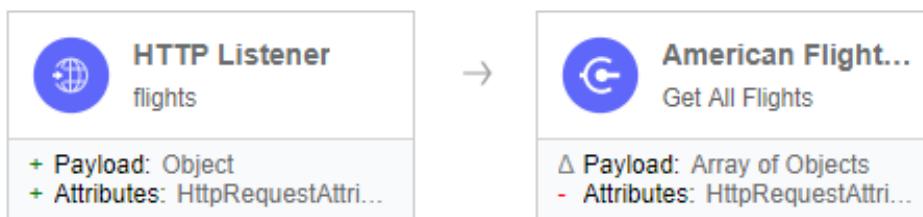
18. In the Get All Flights dialog box, click the Edit link for the American Flights API configuration.



19. Review the information and click Cancel.

The screenshot shows the 'American Flights API Configuration' dialog. It contains fields for Configuration Name (American Flights API), Host (training-american-ws.cloudhub.io), Port (80), Base Path (/api/), and Protocol (HTTP). At the bottom right are 'Cancel' and 'Save' buttons, with 'Save' being highlighted with a blue border.

20. Close the American Flights API card.



21. Click the Deploy button in the main menu bar.

## Test the application

22. Return to Postman and click Send; you should see flight data.

The screenshot shows the Postman interface with a successful response from the American Flights API. The URL is `americanflightsapp-tngx.cloudhub.io/flights`. The response body is a JSON array containing two flight objects:

```
[{"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} ]
```

23. Click Send again to make a second request.

## Review Mule event data

24. Return to flow designer and open the American Flights API card.

25. Click the Input tab and examine the Mule event data.

26. Click the Output tab; you should see payload data.

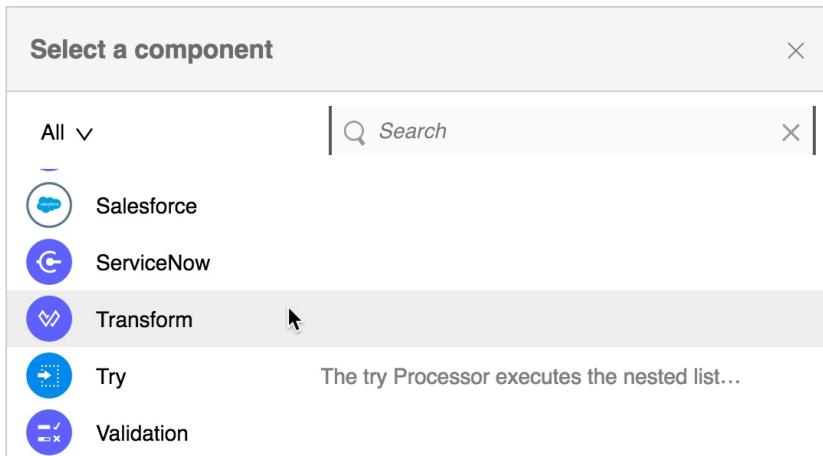
The screenshot shows the Mule Flow Designer with the American Flights API card open. The Input tab is selected, showing the history of events. The most recent event is from Jul 26, 2017 09:25am, which corresponds to the JSON payload shown in the Output tab. The Output tab displays the same JSON array as the Postman response.

```
[{"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} ]
```

27. Close the card.

## Add and configure a component to transform the data

28. Click the add button in the flow.
29. In the Select a component dialog box, select Transform.



30. In the Transform card, look at the Mule event structure in the input section.
31. In the output section, click the Create new Data Type button.

A screenshot of the 'Transform' configuration card. At the top right are tabs for 'Configuration', 'Input', 'Output', and 'X'. The 'Input' tab is active, showing a tree view of the Mule event structure under 'payload': 'plane' (Object?), 'code' (String?), 'price' (Number?), 'origin' (String?), 'destination' (String?), 'ID' (Number?), 'departureDate' (String?), 'emptySeats' (Number?), 'attributes' (Void), and 'vars' (Object). The 'Output payload' tab is active, showing a tree view with a single node. Below it is a message 'No data available' and a button 'Create new Data Type' with the text 'or set an existing one'. The 'Preview' tab shows a chart and a note: 'No data available, please perform some mappings and fill required sample data'. At the bottom are tabs for 'Sample data', 'Script', and 'Mappings', with 'Mappings' being the active tab.

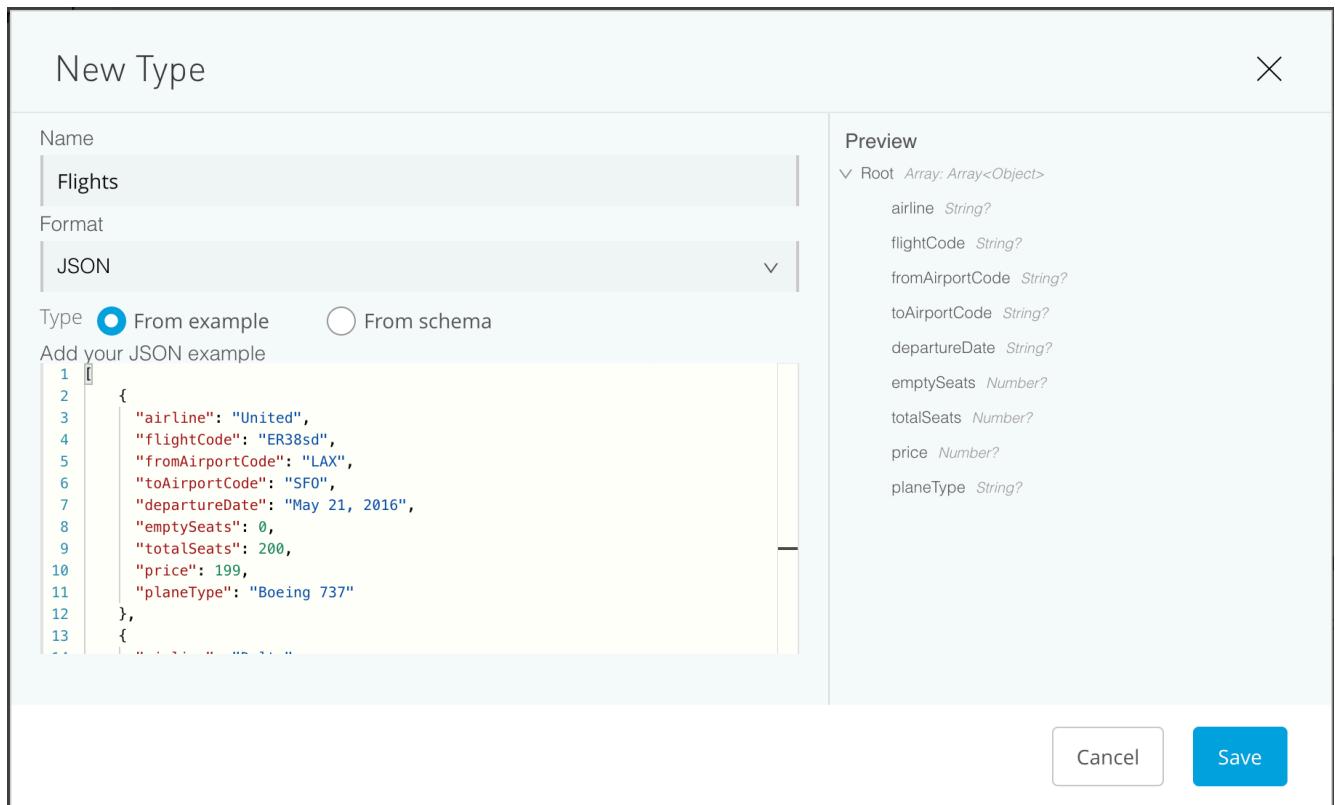
32. In the New Type dialog box, set the following values:

- Name: Flights
- Format: JSON
- Type: From example

33. In the computer's file explorer, return to the student files folder and locate the flights-example.json file in the examples folder.

34. Open the file in a text editor and copy the code.

35. Return to flow designer and paste the code in the section to add your JSON example.



36. Click Save.

37. In the input section, expand the plane object.

Transform Delete Save

Configuration Input Output X

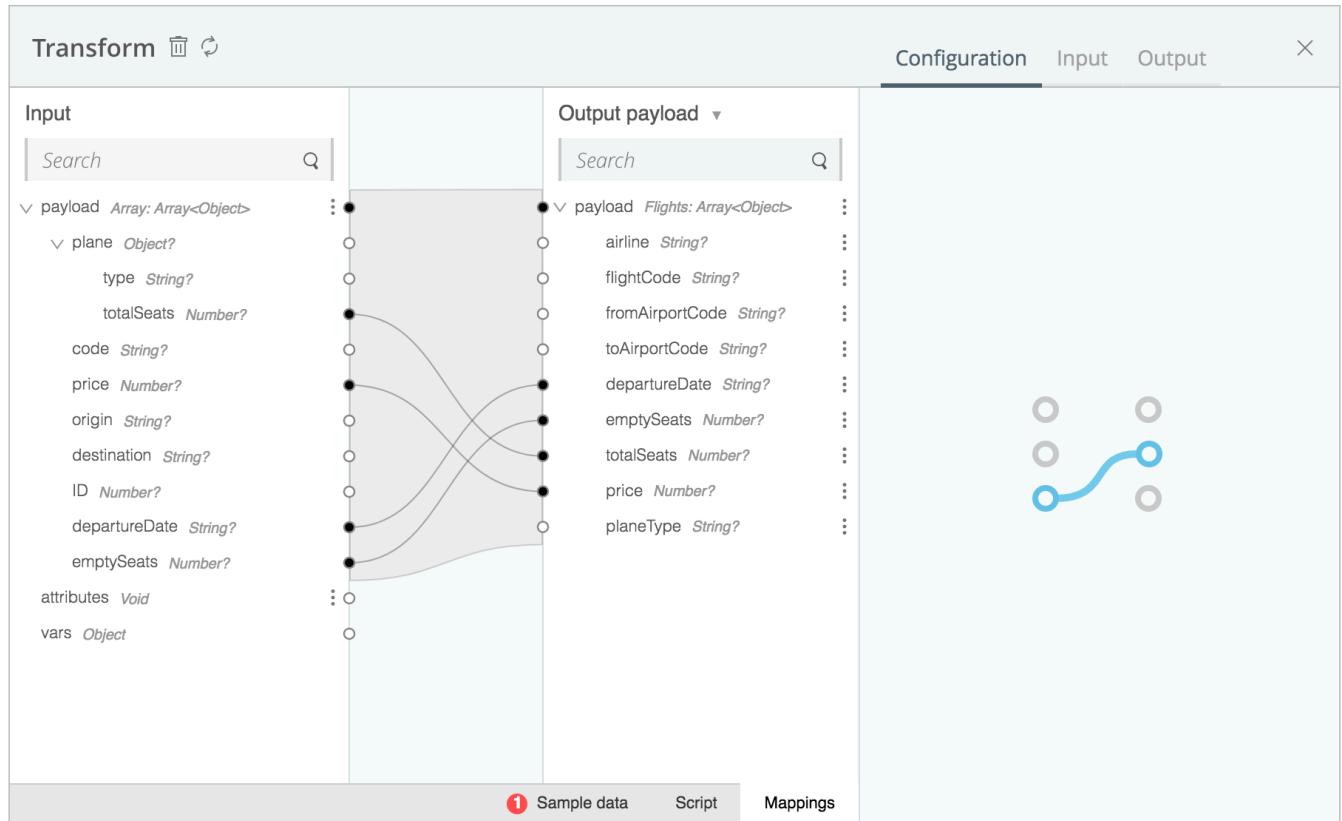
Input	Output payload	Preview
<p>Search <span style="float: right;">Q</span></p> <p>payload Array&lt;Object&gt;</p> <p>  └ plane Object?</p> <p>    type String?</p> <p>    totalSeats Number?</p> <p>    code String?</p> <p>    price Number?</p> <p>    origin String?</p> <p>    destination String?</p> <p>    ID Number?</p> <p>    departureDate String?</p> <p>    emptySeats Number?</p> <p>  attributes Void</p> <p>vars Object</p>	<p>Search <span style="float: right;">Q</span></p> <p>payload Flights: Array&lt;Object&gt;</p> <p>  airline String?</p> <p>  flightCode String?</p> <p>  fromAirportCode String?</p> <p>  toAirportCode String?</p> <p>  departureDate String?</p> <p>  emptySeats Number?</p> <p>  totalSeats Number?</p> <p>  price Number?</p> <p>  planeType String?</p>	<p>No data available, please perform some mappings and fill required sample data</p> 

Sample data    Script    **Mappings**

## Create the transformation

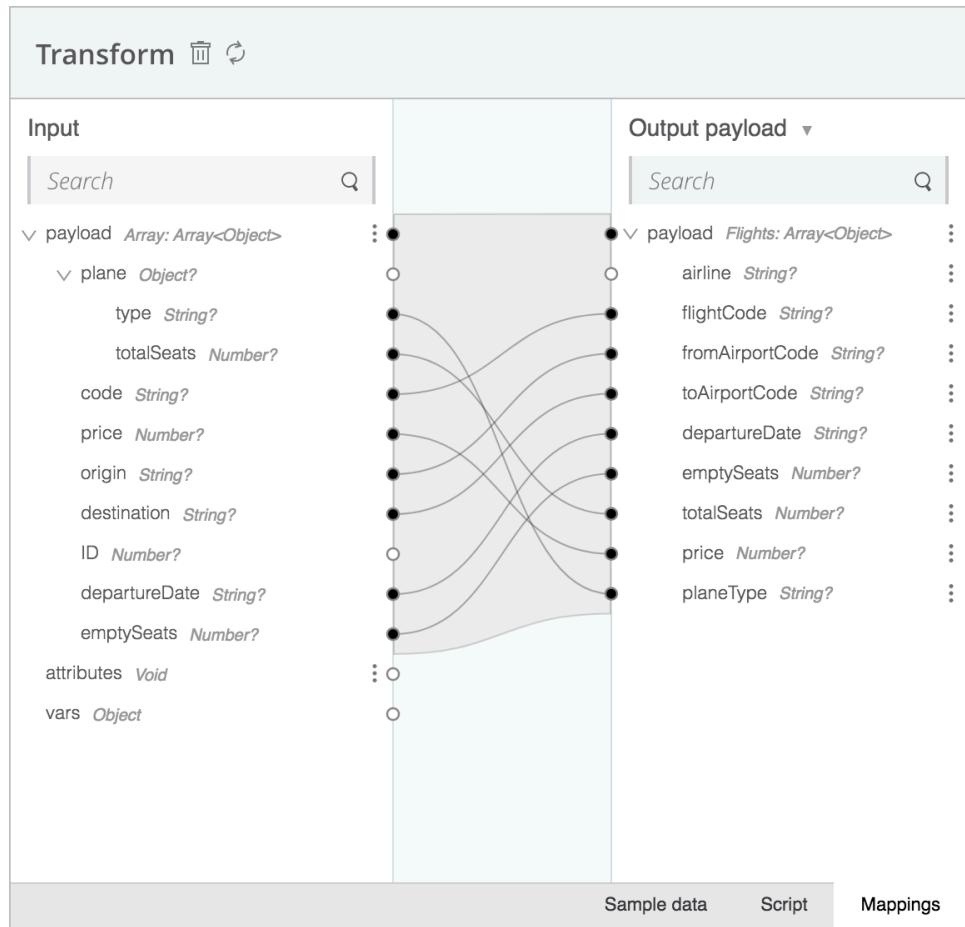
38. Map fields with the same names by dragging them from the input section and dropping them on the corresponding field in the output section.

- price to price
- departureDate to departureDate
- plane > totalSeats to totalSeats
- emptySeats to emptySeats



39. Map fields with different names by dragging them from the input section and dropping them on the corresponding field in the output section.

- plane > type to planeType
- code to flightCode
- origin to fromAirport
- destination to toAirport



40. In the output section, click the options menu for the airline field and select Set Expression.

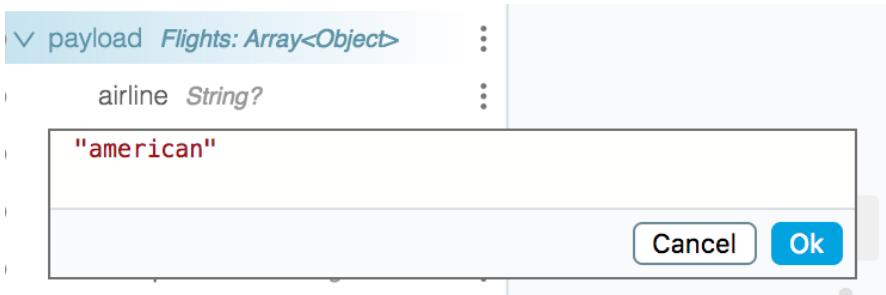
Output payload ▾

Search	Q
✓ payload Flights: Array<Object>	⋮
airline String?	⋮
flightCode String?	⋮
fromAirportCode String?	⋮

Data type actions

- Create
- Edit
- Set
- Detach
- Set Expression**

41. Change the value from null to "american" and click OK.



42. Click the Script tab at the bottom of the card; you should see the DataWeave expression for the transformation.

*Note: You learn to write DataWeave 1.0 expressions later in this course. You can also learn to write DataWeave expressions in the Flow Design course and the DataWeave course.*

The screenshot shows the 'Transform' tab in MuleSoft Anypoint Studio. The 'Input' pane on the left shows a schema for a 'payload' array of 'plane' objects. The 'Transformation script' pane in the center contains the following DataWeave code:

```
1 %dw 2.0
2
3 output application/json
4 ---
5 (payload map (value0, index0) -> {
6   flightCode: value0.code,
7   fromAirportCode: value0.origin,
8   toAirportCode: value0.destination,
9   departureDate: value0.departureDate,
10  emptySeats: value0.emptySeats,
11  totalSeats: value0.plane.totalSeats,
12  price: value0.price,
13  planeType: value0.plane.type,
14  airline: "american"
15 })
```

The 'Preview' pane on the right shows a tree structure for the transformed data. A message at the bottom states: "Sample data for payload is not well defined. Please review it." A 'Set sample data' button is available.

## Add sample data

43. Click the Set sample data button in the preview section.

44. In the computer's file explorer, return to the student files folder and locate the american-flights-example.json file in the examples folder.

45. Open the file in a text editor and copy the code.

46. Return to flow designer and paste the code in the sample data for payload section.

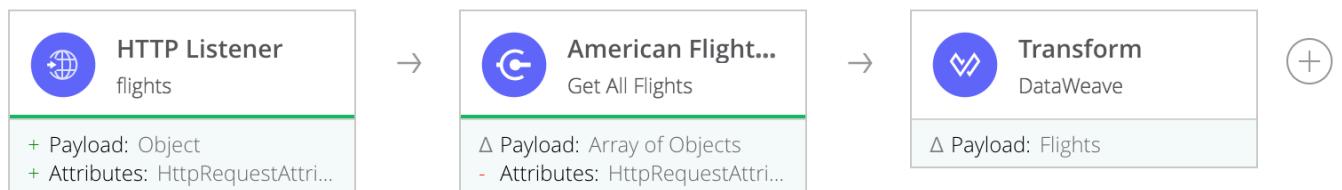
The screenshot shows the Mule ESB Flow Designer interface with a 'Transform' component selected. The 'Input' tab displays the JSON schema for the 'payload' array, which includes fields like ID, code, price, departureDate, origin, destination, ID, departureDate, emptySeats, attributes, and vars. The 'Sample data for payload (application/json)' tab contains the following JSON code:

```
1 [ {  
2     "ID": 1,  
3     "code": "ER38sd",  
4     "price": 400,  
5     "departureDate": "2016/03/20",  
6     "origin": "MUA",  
7     "destination": "SFO",  
8     "emptySeats": 0,  
9     "plane": {  
10         "type": "Boeing 737",  
11         "totalSeats": 150  
12     }  
13 }, {  
14     "ID": 2,  
15     "code": "ER45if",  
16     "price": 345.99,  
17     "departureDate": "2016/02/11",  
18     "origin": "MUA",  
19     "destination": "LAX",  
20     "emptySeats": 52,  
21     "plane": {  
22         "type": "Boeing 777",  
23         "totalSeats": 300  
24     }  
25 } ]
```

The 'Preview' tab shows the resulting transformed JSON output, which includes two flight objects with their respective details. The 'Script' and 'Mappings' tabs are also visible at the bottom of the component configuration window.

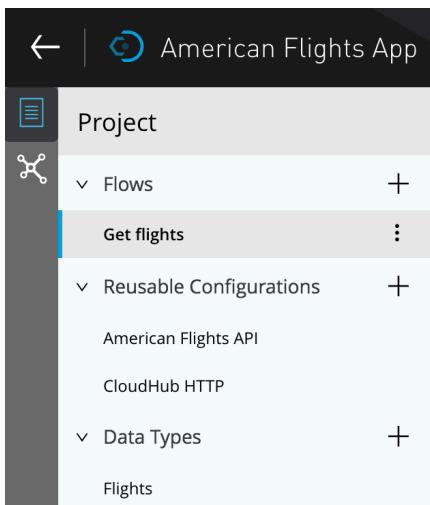
47. Look at the preview section, you should see a sample response for the transformation.

48. Close the card.



## Locate the data type and configuration definitions

49. Locate the connector configurations and the new Flight data type in the project explorer.



## Test the application

50. Deploy the project.

51. Return to Postman and click Send to make another request to <http://americanflightsapp-xxxx.cloudhub.io/flights>; you should see all the flight data as JSON again but now with a different structure.

```
1 [
2   {
3     "flightCode": "rree0001",
4     "fromAirportCode": "MUA",
5     "toAirportCode": "LAX",
6     "departureDate": "2016-01-20T00:00:00",
7     "emptySeats": 0,
8     "totalSeats": 200,
9     "price": 541,
10    "planeType": "Boeing 787",
11    "airline": "American"
12  },
13  {
14    "flightCode": "eefd0123",
15    "fromAirportCode": "MUA",
16    "toAirportCode": "CLE".
```

## Stop the application

52. Return to Runtime Manager.

53. In the left-side navigation, click Applications.

54. Select the row with your application; you should see information about the application displayed on the right side of the window.

The screenshot shows the MuleSoft Runtime Manager interface. On the left, there's a sidebar with 'DESIGN' selected, followed by 'Applications', 'Servers', 'Alerts', and 'VPCs'. The main area has tabs for 'Deploy application' and 'Search Applications'. A table lists applications: 'americanflightsapp-tng' under 'Server' 'CloudHub', 'Status' 'Started', and 'File' 'null'. To the right, a detailed view for 'americanflightsapp-tngx' shows it's 'Started' on 'CloudHub'. It was last updated on 2017-07-26 at 9:17:36AM with the URL [americanflightsapp-tngx.cloudhub.io](http://americanflightsapp-tngx.cloudhub.io). Configuration details include Runtime version 4.0.0-BETA.4, Worker size 0.2 vCores, 1 worker, and US West (N. California) region. Buttons for 'Manage Application', 'Logs', and 'Insight' are at the bottom, along with a link to 'View Associated Alerts'.

55. Click the drop-down menu button next to Started and select Stop; the status should change to Undeployed.

*Note: You can deploy it again from flow designer when or if you work on the application again.*

This screenshot shows the same Runtime Manager interface after the application status has been changed. The table now shows 'Status' as 'Undeployed' instead of 'Started'. The detailed view for 'americanflightsapp-tngx' also shows it's now 'Undeployed' on 'CloudHub'. The configuration details remain the same: Runtime version 4.0.0-BETA.4, Worker size 0.2 vCores, 1 worker, and US West (N. California) region. The 'Manage Application', 'Logs', and 'Insight' buttons are present at the bottom.

56. Close Runtime Manager.

57. Return to flow designer; you should see the application is undeployed.

The screenshot shows the MuleSoft flow designer. At the top, there's a header with the application name 'American Flights App', a save timestamp 'Saved 5 minutes ago', and navigation icons. Below the header, the status bar indicates the application is 'Undeployed'. There are 'Deploy' and three-dot more options buttons on the right.

58. Return to Design Center.