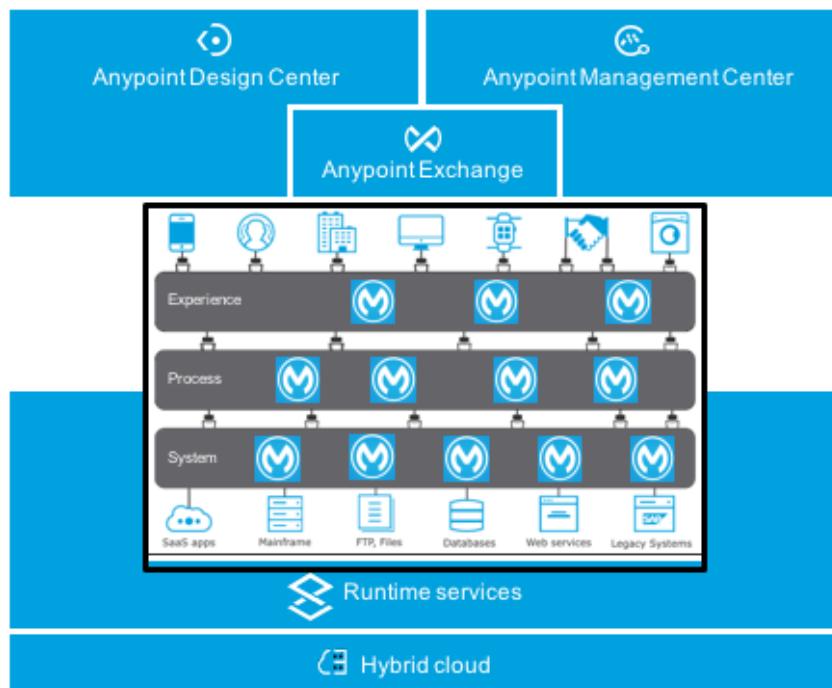


Module 2: Introducing Anypoint Platform



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 JSON example:

```
[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.

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.

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' (blue circle icon), 'Amazon RDS Connector' (orange circle icon), and 'Amazon EC2 Connector' (orange circle icon). Each connector has a star rating of five stars. 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'. A search bar and a 'New' button are visible at the top right.

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).

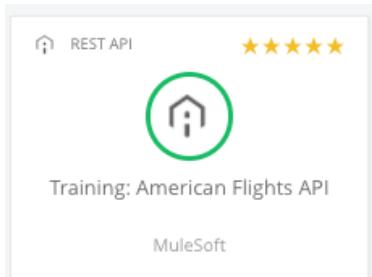
Browse REST APIs in Anypoint Exchange

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 'REST APIs' selected. The main area displays three REST API cards: 'Optymyze API' (MuleSoft), 'CardConnect REST API' (MuleSoft), and 'Nexmo SMS API' (MuleSoft). Each card includes a star rating (5 stars), a thumbnail icon, the API name, and the provider ('MuleSoft').

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 is a sidebar with a navigation menu. The main area shows the 'Training: American Flights API' details. The title is 'Training: American Flights API | 1.0' with a 5-star rating of '(7 reviews)' and a 'Rate and review' button. Below the title is a brief description: 'This example RAML 1.0 API is used in MuleSoft training courses. It defines the basic GET, POST, DELETE, and PUT operations for flights and uses data type and example fragments.' A code editor window displays the RAML 1.0 API definition. The right side of the screen shows the 'Overview' section with details like Type (REST API), Created By (MuleSoft Organization), Published On (Sep 14, 2017), and Visibility (Public). It also lists 'Asset versions for 1.0' with two entries: 1.0.1 and 1.0.0, each with a 'Mocking Service' link.

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 navigation links like 'Assets list', 'Training: American Flights API' (which is highlighted), 'API summary', 'Types', 'Resources', '/flights' (which is expanded), and 'API instances'. Under '/flights', there are four items: 'Get all flights' (GET), 'Add a flight' (POST), 'Get a flight by ID' (GET), and 'Delete a flight by ID' (DELETE). The main content area has a title 'Training: American Flights API | 1.0' with a 5-star rating. Below it, the endpoint '/flights : Get all flights' is listed with a 'Request' section containing a GET method and its URL. To the right, a detailed view of the 'Get all flights' endpoint is shown. It includes a 'Parameters' section with a table:

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

Below this is a 'Response' section showing a JSON schema for the 200 status code:

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

On the far right, there are tabs for 'Parameters' (selected), 'Headers', and 'Send' button.

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 200 status code. It starts with a 'Type application/json' header. Below it, the '200' status is selected, and the 'Type' tab is active. The schema is displayed as a JSON array:

```
[  
  {  
    "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 several icons: a checkmark, a download arrow, a copy icon, and a grid icon. The response body is displayed as a JSON array with one item:

```
[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

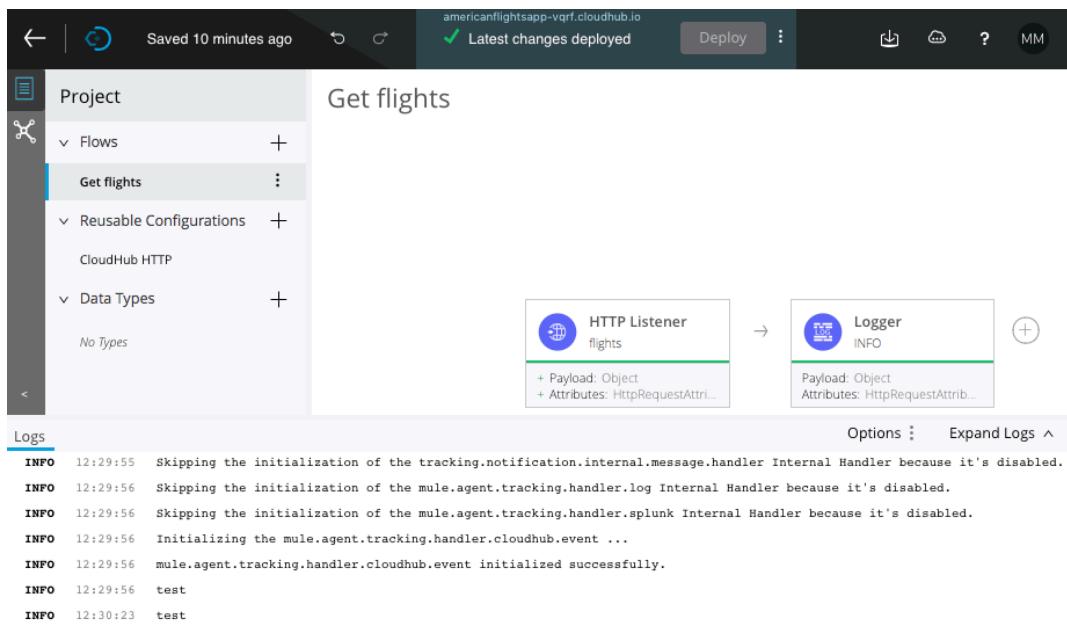
200 OK 1310.77 ms Details ▾

```
[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

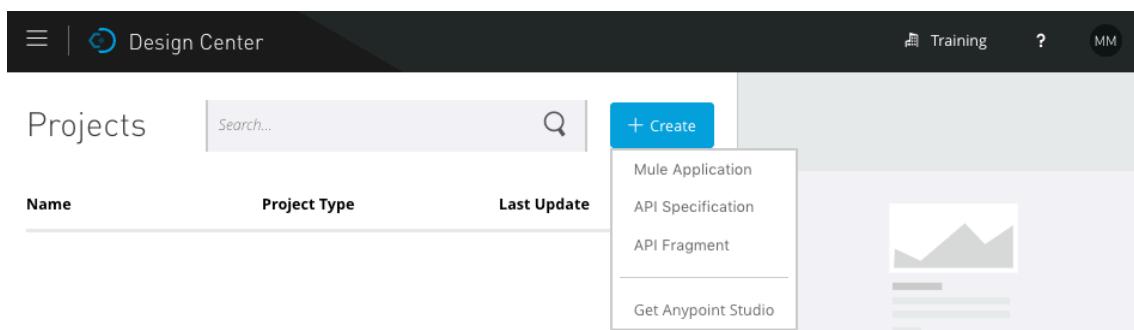
In this walkthrough, you build, run, and test a basic Mule application with flow designer. You will:

- 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.



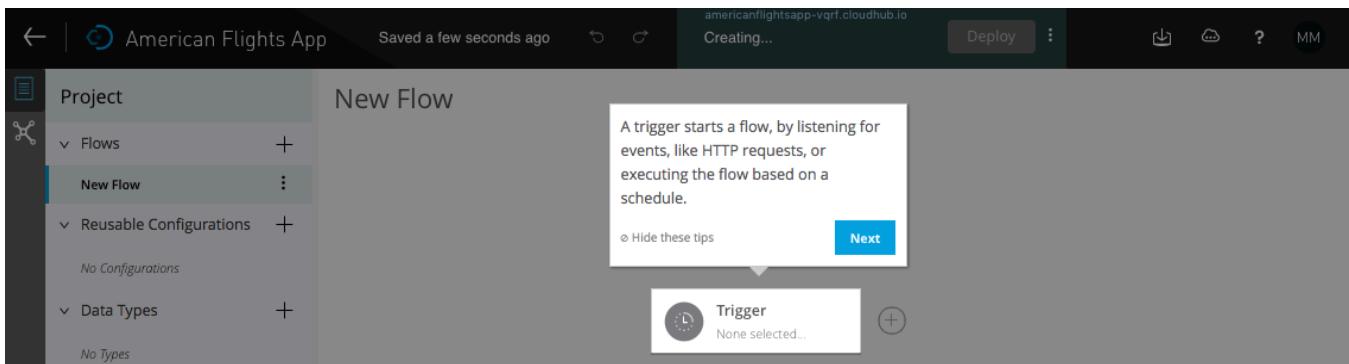
Create a Mule application project in Design Center

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

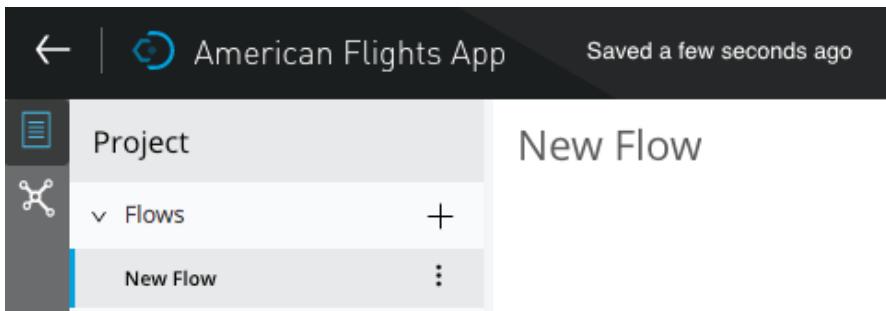


4. In the New Mule Application dialog box, set the project name to American Flights App.

- 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

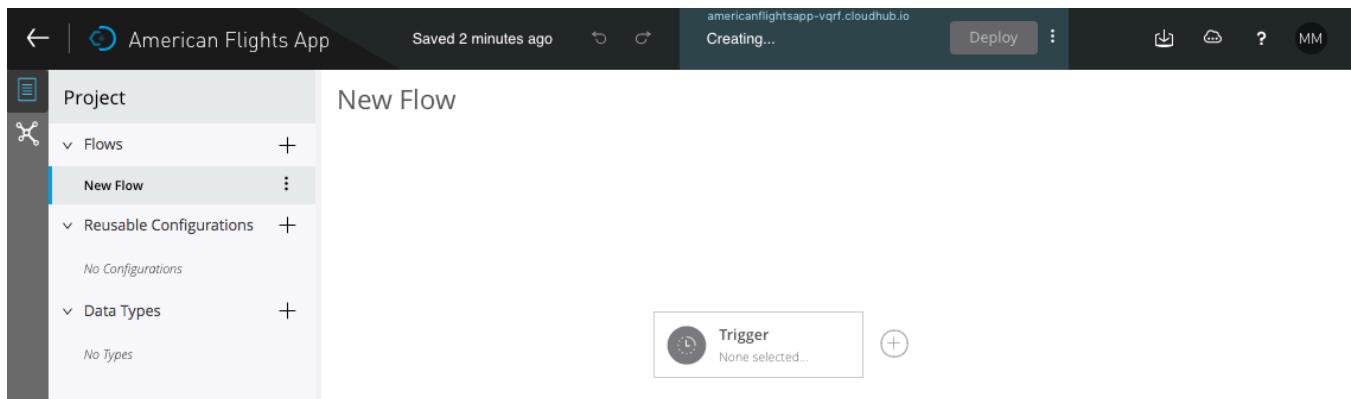
American Flights App

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

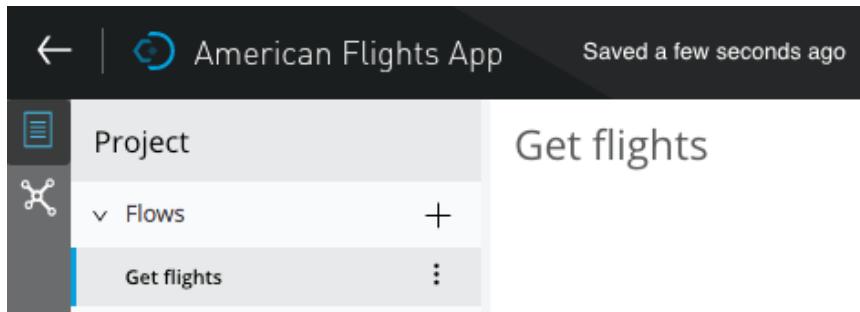
Open

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



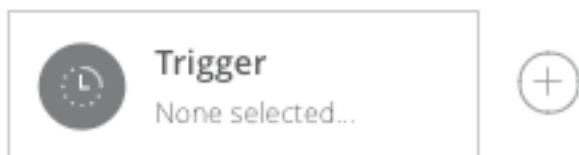
Rename the flow

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.

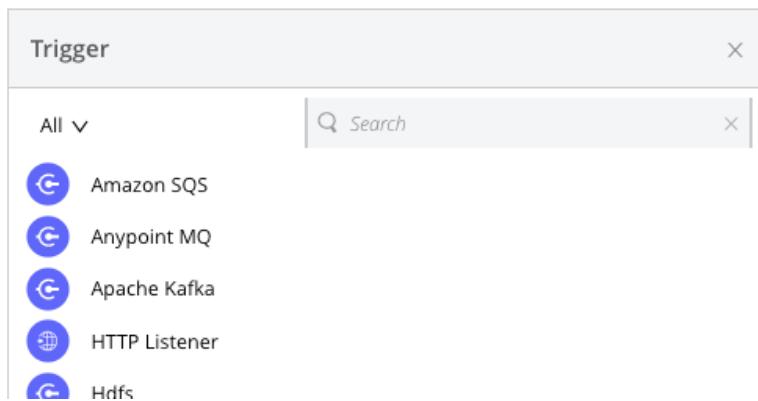


Create an HTTP trigger for a flow in the application

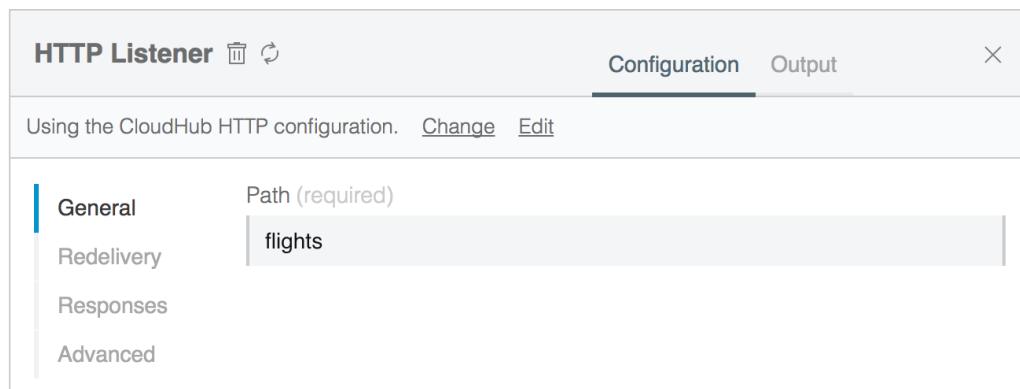
13. In flow designer, click the Trigger card.



14. In the Trigger card, select HTTP Listener.

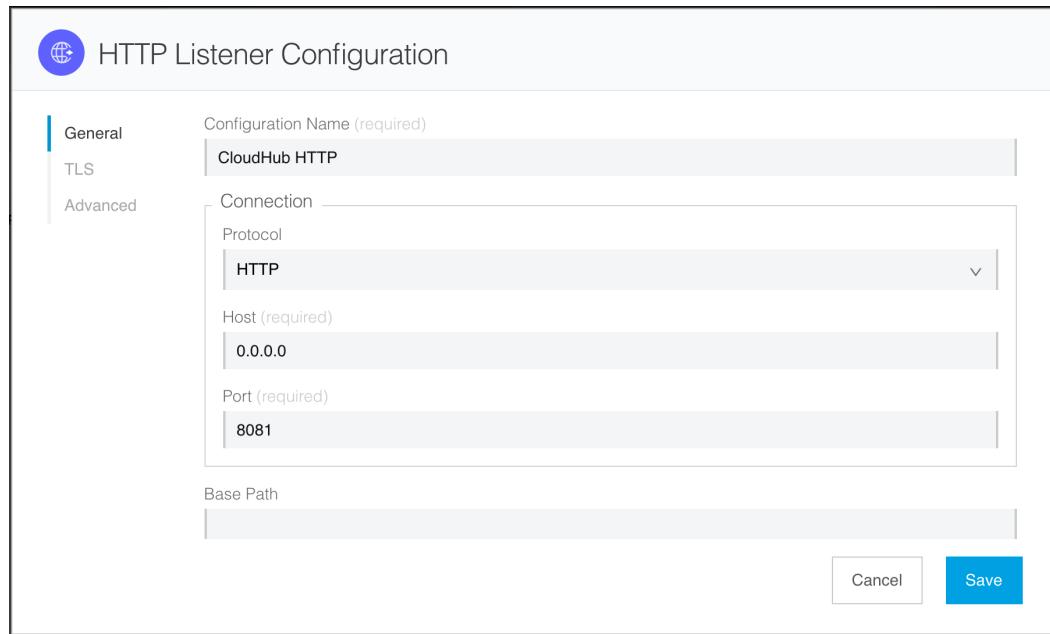


15. In the HTTP Listener dialog box, set the path to flights.



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

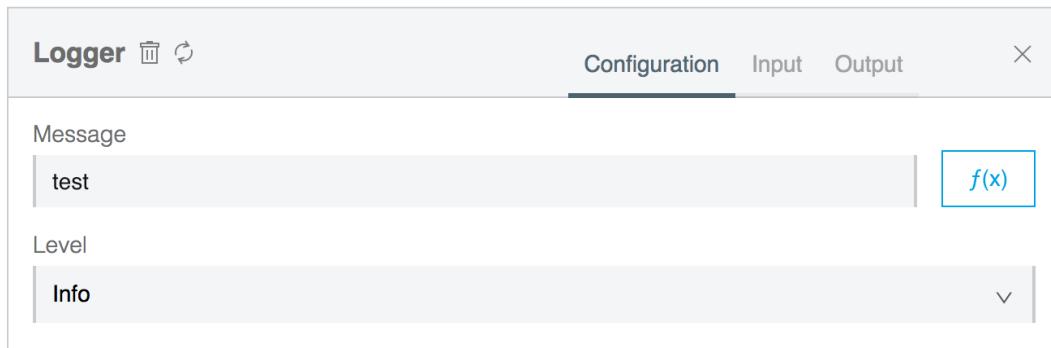
19. Click the add button next to the HTTP Listener card.

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.

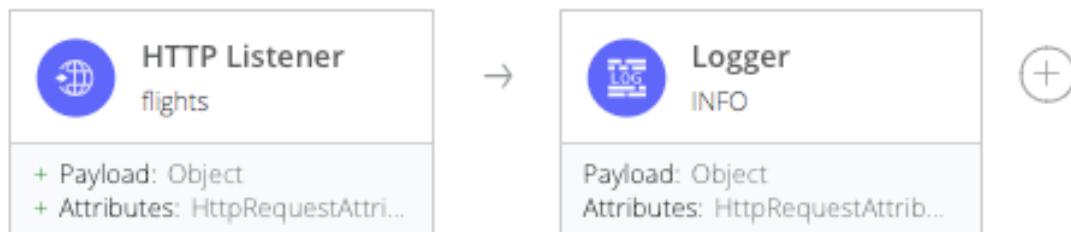
The screenshot shows the 'Select a component' dialog. At the top, it says 'Select a component' and has a search bar with a magnifying glass icon and a clear 'x' button. Below that, there is a dropdown menu set to 'All' and a 'Search' field. The list of components includes: Inventory API | R..., LDAP, 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.
```

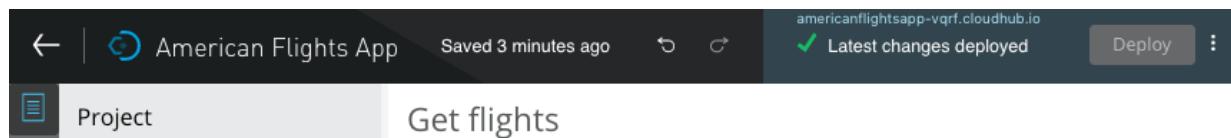
25. Locate the application status in the main menu bar; it should say Ready to deploy.

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.

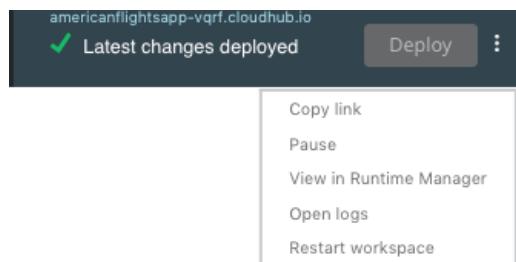
27. Click the Deploy button.

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

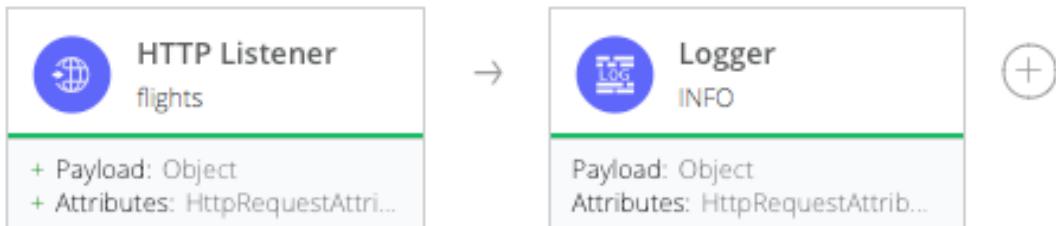
30. Return to Postman, paste the copied link, and click Send; you should get a No listener for endpoint: / message.

The screenshot shows the Postman interface with a GET request to 'americanflightsapp-vqrfl.cloudhub.io'. The response body is a single line of text: '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 the Postman interface with a GET request to '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.



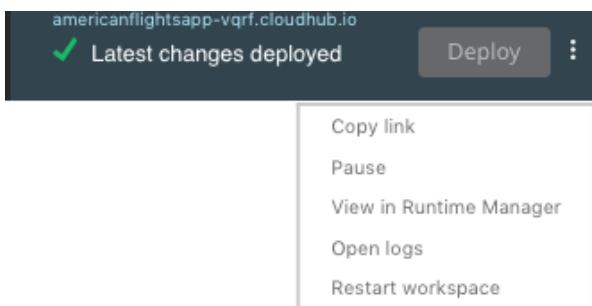
35. Look at the logs; you should see your Logger message displayed twice.

Logs

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

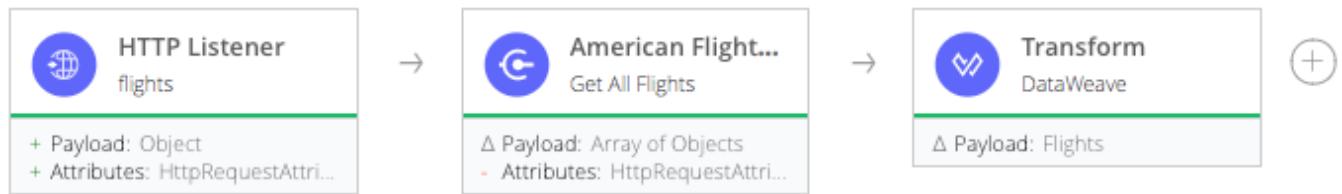
- To which environment it was deployed
- To what type of Mule runtime it was deployed
- To what size worker it was deployed

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 the last update time (2017-11-19 12:20:46PM) and the app URL (americanflightsapp-vqrf.cloudbu.io). The 'Runtime' tab is active, showing the runtime version as 4.0.0, worker size as 0.2 vCores, and one worker assigned. There are also sections for Properties, Insight, Logging, and Static IPs. At the bottom, there are checkboxes for automatically restarting the application when it's not responding, enabling persistent queues, and encrypting 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 X

Configuration Output

Show: Payload ▼

Data type: Object ▼

Add Edit

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 X

		Configuration	Output	X
Show: Attributes ▾				
History	<pre>{ "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" } }</pre>			
Jul 25, 2017 05:27pm				
Jul 25, 2017 05:26pm				

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

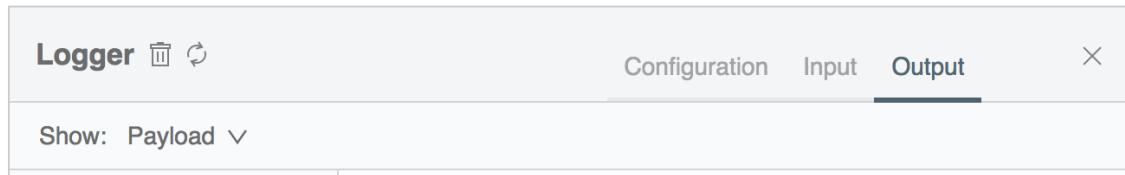
Logger X

		Configuration	Input	Output	X
Show: Attributes ▾					
History	<pre>{ "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" } }</pre>				
Jul 25, 2017 05:27pm					
Jul 25, 2017 05:26pm					

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

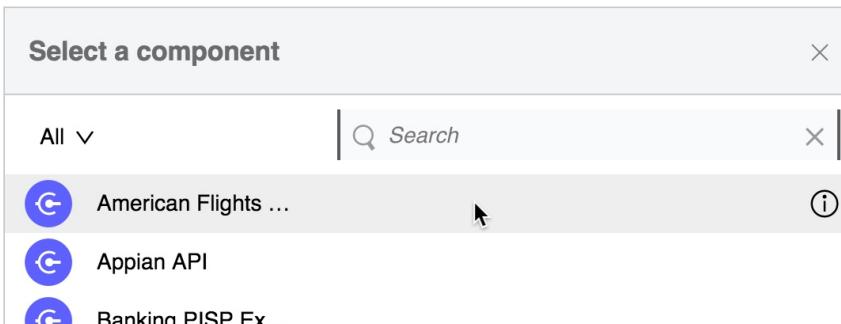
Delete a card

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

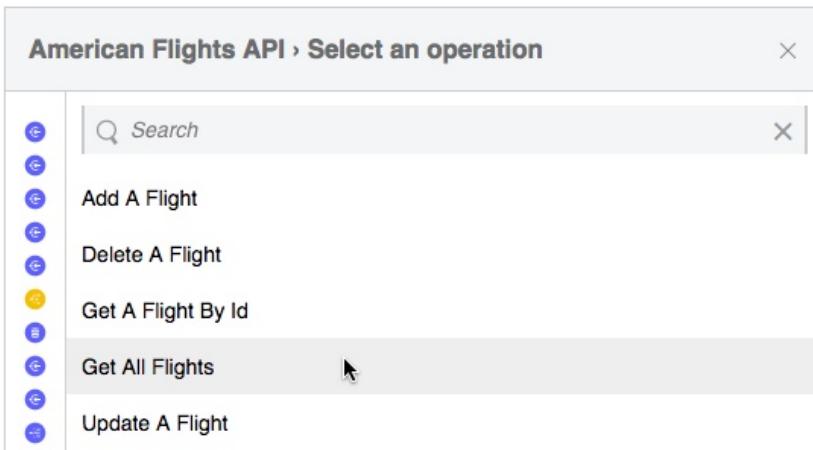


Use the American Flights API in Anypoint Exchange to get all flights

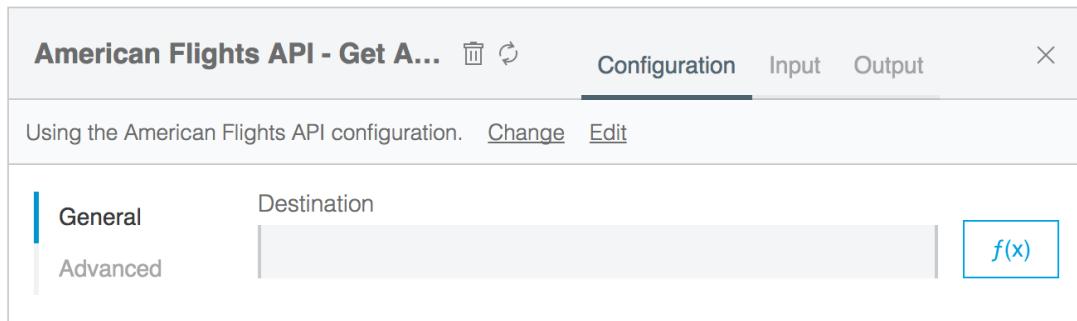
15. Click the Add button next to the HTTP Listener card.
16. In the Select a component dialog box, select the American Flights API.



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



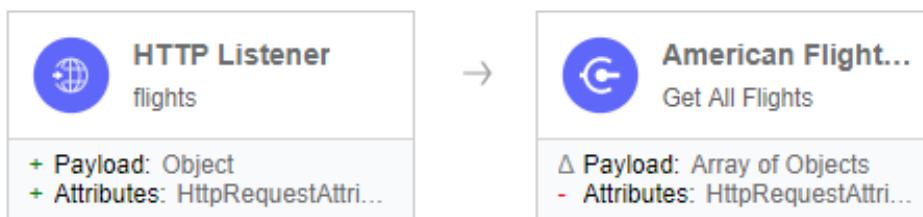
18. In the Get All Flights dialog box, click the Edit link for the American Flights API configuration.



19. Review the information and click Cancel.

The screenshot shows the "American Flights API Configuration" dialog. It contains fields for Configuration Name (set to "American Flights API"), Host (set to "training-american-ws.cloudhub.io"), Port (set to "80"), Base Path (set to "/api/"), and Protocol (set to "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.

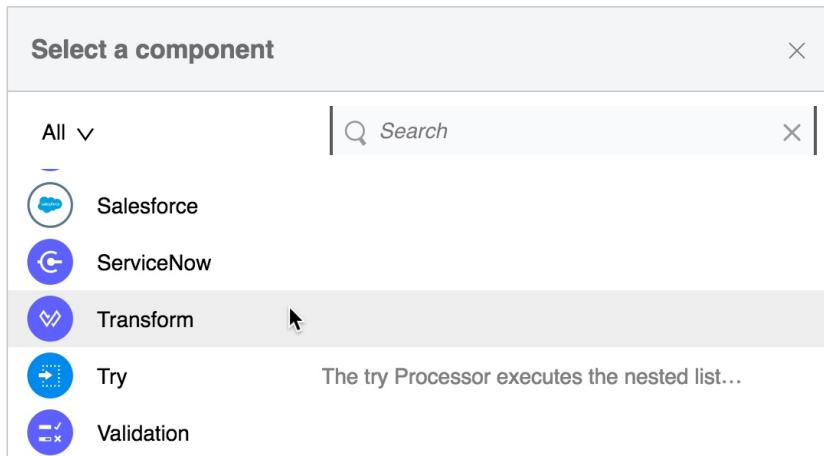
The screenshot shows the Mule Flow Designer with the American Flights API card open. The Output tab is selected, displaying the payload data. The payload is identical to the one shown in Postman, consisting of an array of flight objects.

```
[{"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} ]
```

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.

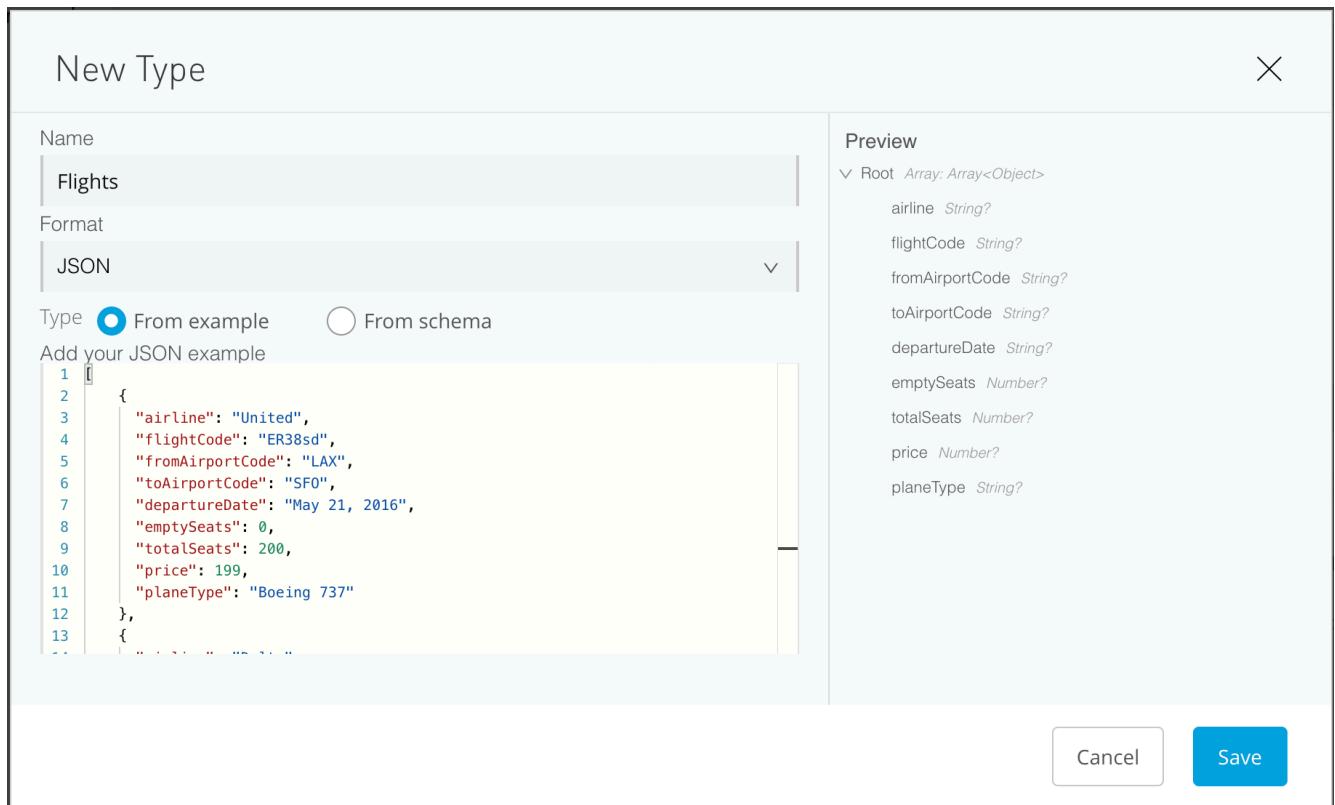
32. In the New Type dialog box, set the following values:

- 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

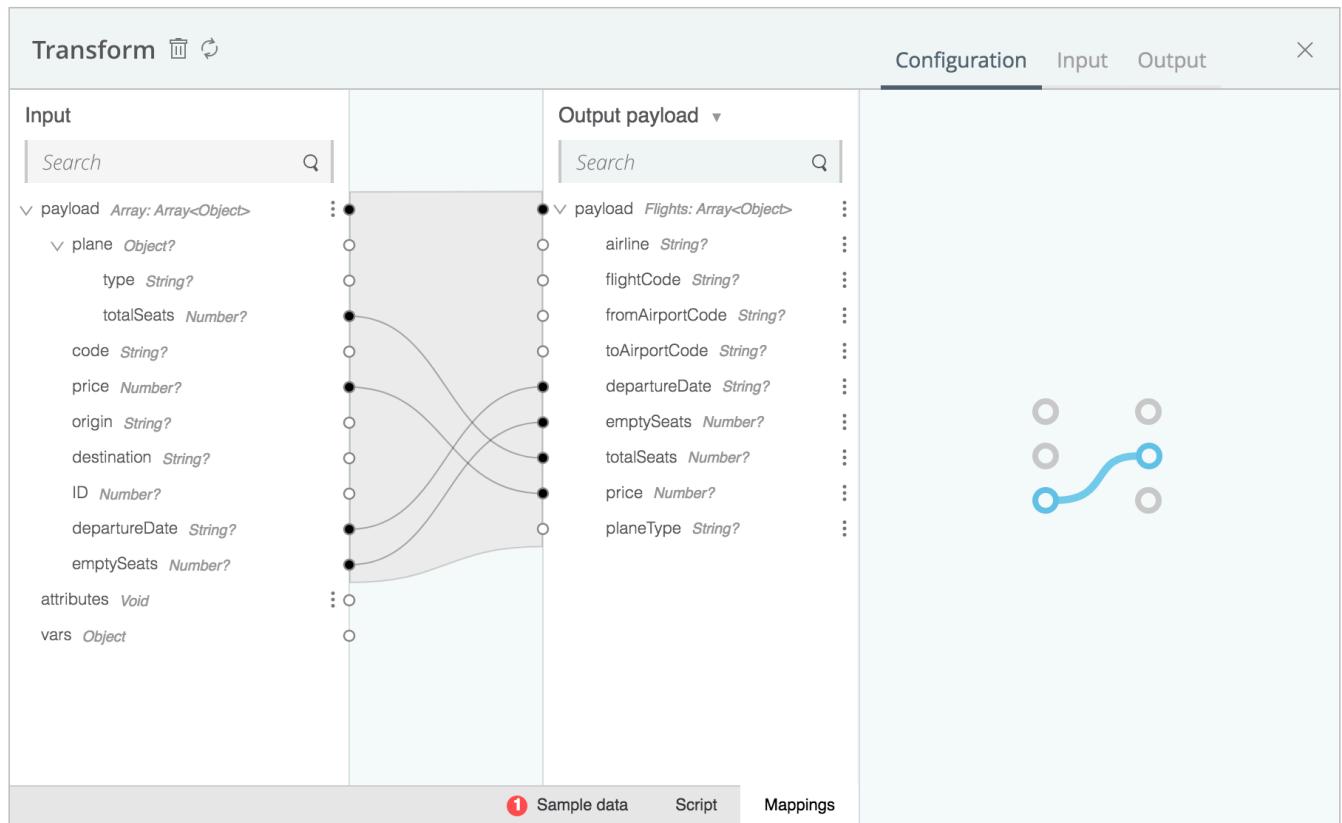
Input	Output payload	Preview
<p>Search Q</p> <p>payload Array<Object></p> <p> └ plane Object?</p> <p> type String?</p> <p> totalSeats Number?</p> <p> code String?</p> <p> price Number?</p> <p> origin String?</p> <p> destination String?</p> <p> ID Number?</p> <p> departureDate String?</p> <p> emptySeats Number?</p> <p> attributes Void</p> <p>vars Object</p>	<p>Search Q</p> <p>payload Flights: Array<Object></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

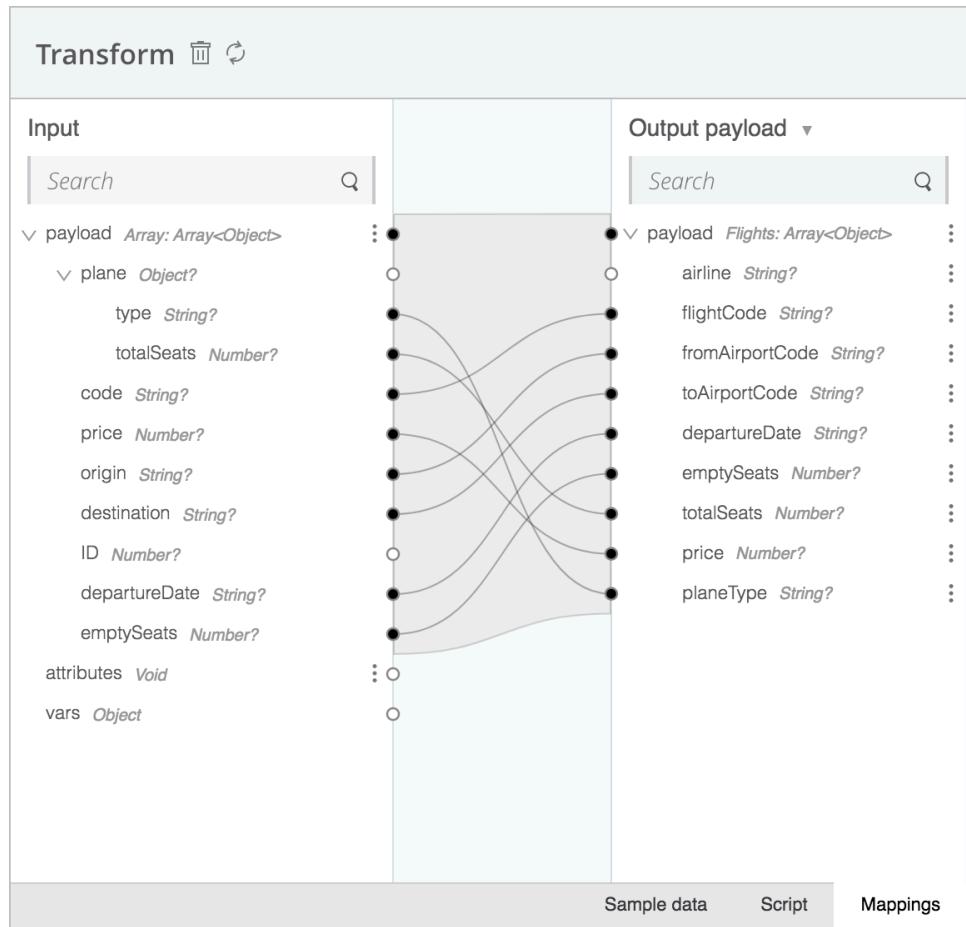
38. Map fields with the same names by dragging them from the input section and dropping them on the corresponding field in the output section.

- 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



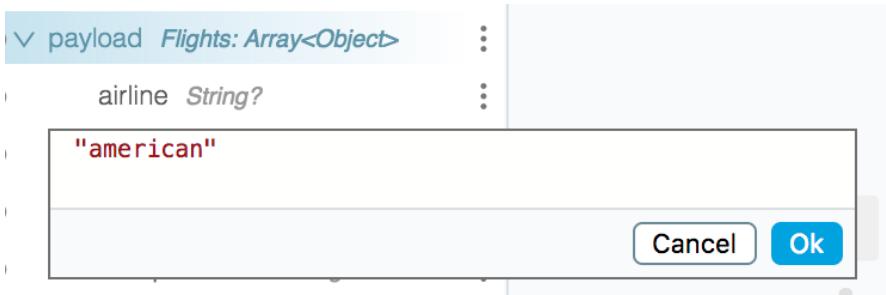
40. In the output section, click the options menu for the airline field and select Set Expression.

Output payload ▾

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

A context menu is open over the 'airline' field in the Output payload section. The menu is titled 'Data type actions' and includes the following options: Create, Edit, Set, Detach, and Set Expression. The 'Set Expression' option is highlighted with a blue background.

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.

Transform

Configuration Input Output X

Input

Sample data for payload (application/json)

```
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 } ]
```

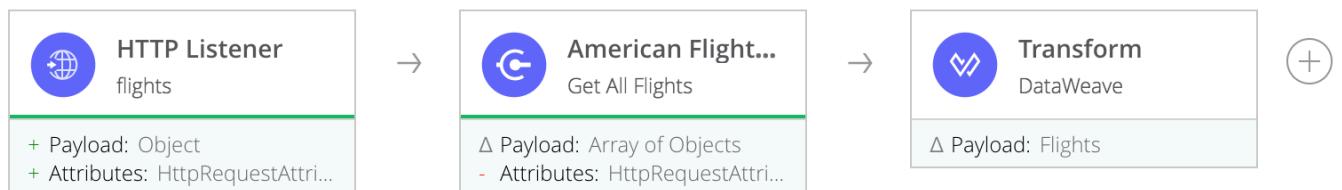
Preview

```
1 [ {  
2   "flightCode": "ER38sd",  
3   "fromAirportCode": "MUA",  
4   "toAirportCode": "SFO",  
5   "departureDate": "2016/03/20",  
6   "emptySeats": 0,  
7   "totalSeats": 150,  
8   "price": 400,  
9   "planeType": "Boeing 737",  
10  "airline": "american"  
11 }, {  
12   "flightCode": "ER45if",  
13   "fromAirportCode": "MUA",  
14   "toAirportCode": "LAX",  
15   "departureDate": "2016/02/11",  
16   "emptySeats": 52,  
17   "totalSeats": 300,  
18   "price": 345.99,  
19   "planeType": "Boeing 777",  
20   "airline": "american"  
21 }, {  
22   "flightCode": "ER45if",  
23   "fromAirportCode": "MUA",  
24   "toAirportCode": "LAX",  
25   "departureDate": "2016/02/11",  
26   "emptySeats": 52,  
27   "totalSeats": 300,  
28   "price": 345.99,  
29   "planeType": "Boeing 777",  
30   "airline": "american"  
31 }
```

Sample data Script Mappings

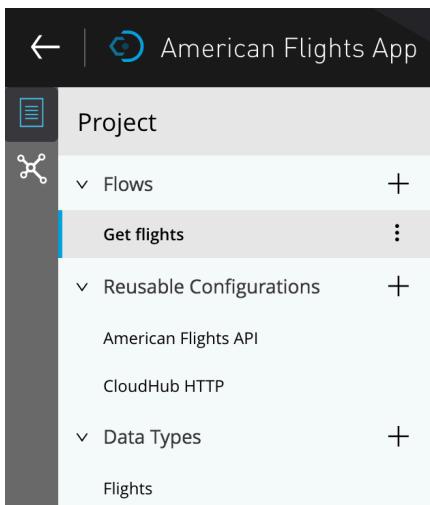
47. Look at the preview section, you should see a sample response for the transformation.

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' is shown. It shows the status as 'Started', 'CloudHub' as the server, and 'null' as the file. Below this, runtime details are listed: 'Last Updated' (2017-07-26 9:17:36AM), 'App url' (americanflightsapp-tngx.cloudhub.io), 'Runtime version' (4.0.0-BETA.4), 'Worker size' (0.2 vCores), 'Workers' (1), and 'Region' (US West (N. California)). 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 is identical to the previous one, but the application status in the table has been changed to 'Undeployed' from 'Started'. The rest of the interface and application details remain the same.

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 buttons for 'Deploy' and more options. The main workspace is visible below the header.

58. Return to Design Center.