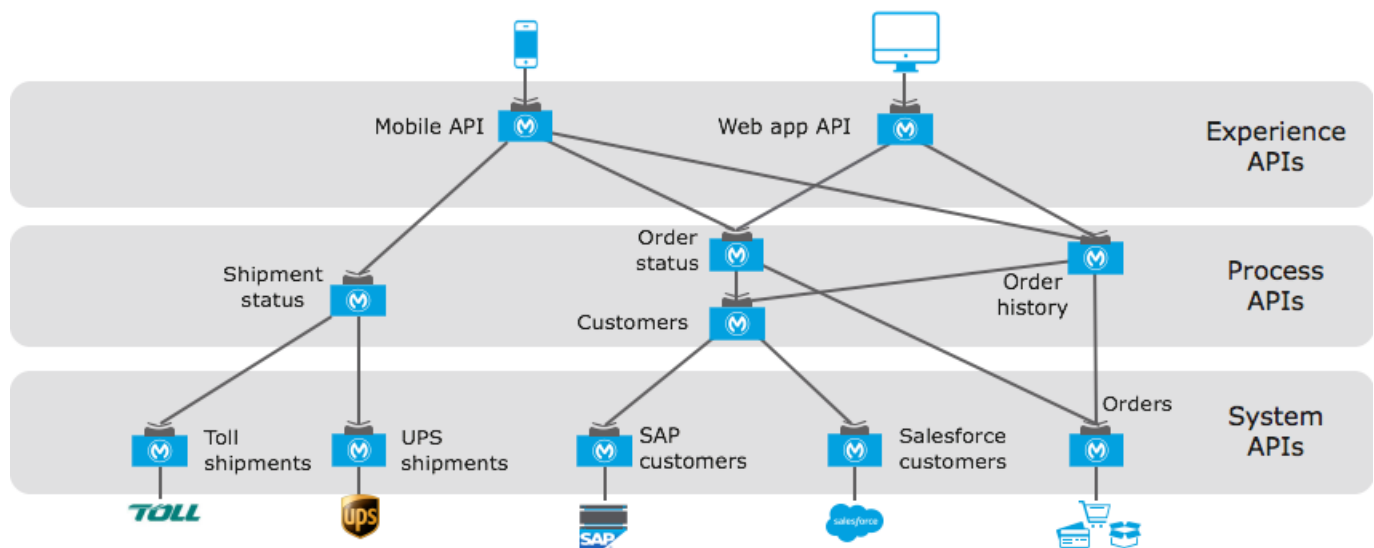


Module 1: Introducing Application Networks and API-Led Connectivity



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.

The screenshot shows the Twitter API reference page. The top navigation bar is purple with links for Developer, Use cases, Products, Docs, and More. A search bar and a 'Sign In' button are on the right. The main heading is 'Post, retrieve and engage with Tweets'. Below it, there are tabs for Overview, Guides, and API Reference. The left sidebar lists various API endpoints under the heading 'Tweets'. The main content area lists the following API endpoints:


Tweets	Retweets	Likes (formerly favorites)
<ul style="list-style-type: none">• POST statuses/update• POST statuses/destroy/:id• GET statuses/show/:id	<ul style="list-style-type: none">• POST statuses/retweet/:id• POST statuses/unretweet/:id• GET statuses/retweets/:id• GET statuses/retweets_of_me• GET statuses/retweeters/ids	<ul style="list-style-type: none">• POST favorites/create/:id• POST favorites/destroy/:id• GET favorites/list

Browse the ProgrammableWeb API directory

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

The screenshot shows the ProgrammableWeb API directory homepage. The top navigation bar includes links for API NEWS, API DIRECTORY, and a search bar. The main content area features a grid of API categories: API UNIVERSITY, RESEARCH, SPORTS, SECURITY, TRAVEL, and DESIGN. A prominent advertisement for MuleSoft is displayed on the right, with the text 'Build a better API strategy' and 'A 7 step blueprint for success'.

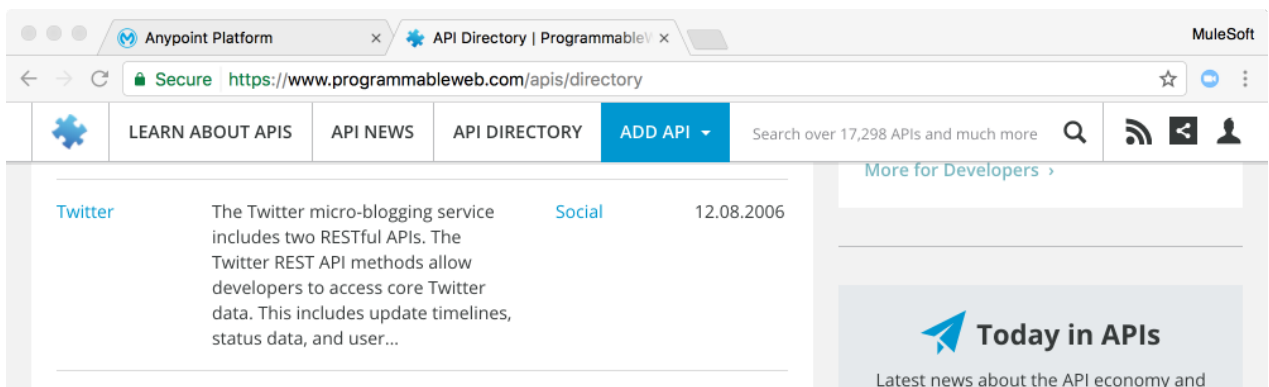
3. Browse the list of popular APIs.

	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 URL <https://www.programmableweb.com/apis/directory>. The page features a navigation bar with tabs: "LEARN ABOUT APIS", "API NEWS", "API DIRECTORY" (selected), and "ADD API ▾". A search bar indicates "Search over 17,298 APIs and much more". The main content area displays a list of APIs, with "Twitter" at the top. The Twitter entry includes a description: "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...". To the right of the list, there is a sidebar with a "More for Developers >" link and a section titled "Today in APIs" with the subtitle "Latest news about the API economy and".

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

SPECS

API Endpoint	http://twitter.com/statuses/
API Portal / Home Page	https://dev.twitter.com/rest/public
Primary Category	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.

The screenshot shows the Twitter API documentation page. The top navigation bar is purple with links for Developer, Use cases, Products, Docs, and More. On the right, there are links for Apply, a search icon, and a Sign In button. The main content area has a search bar and a title 'Post, retrieve and engage with Tweets'. Below the title are tabs for Overview, Guides, and API Reference. The left sidebar contains a list of navigation items: Basics, Accounts and users, and Tweets. Under Tweets, there is a link for 'Post, retrieve and engage with Tweets'. The main content area lists the following API endpoints that can be used to programmatically create, retrieve and delete Tweets, Retweets and Likes:

Tweets	Retweets	Likes (formerly favorites)
<ul style="list-style-type: none">• POST statuses/update• POST statuses/destroy/:id• GET statuses/show/:id	<ul style="list-style-type: none">• POST statuses/retweet/:id• POST statuses/unretweet/:id• GET statuses/retweets/:id• GET statuses/retweets_of_me• GET statuses/retweeters/ids	<ul style="list-style-type: none">• 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.

 Developer Use cases Products Docs More Apply  Sign in

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

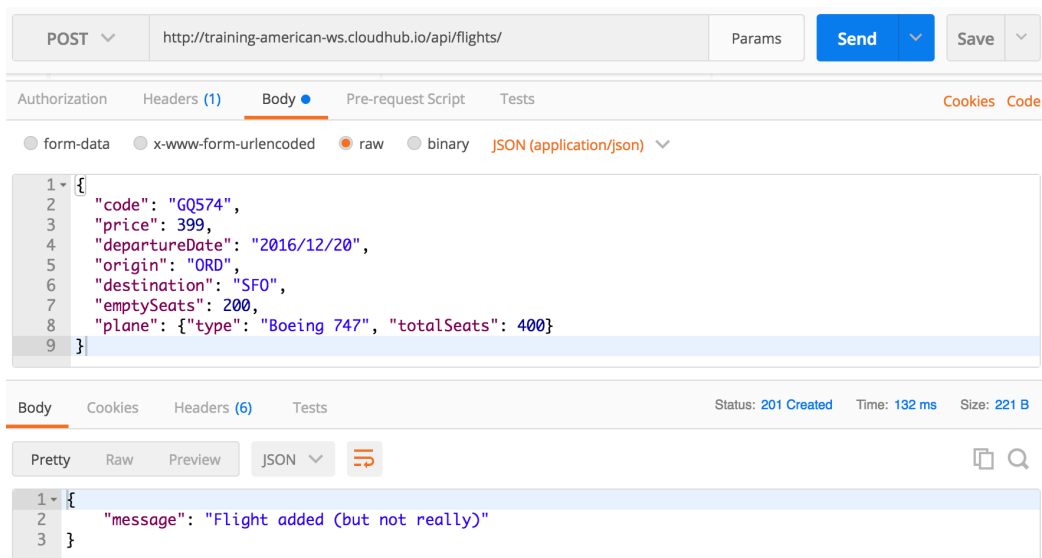
```
{
  "coordinates": null,
  "favorited": false,
  "created_at": "Wed Sep 05 00:37:15 +0000 2012",
  "truncated": false,
  "id_str": "243145735212777472",
  "entities": {
    "urls": [
```

11. Close the two browser tabs.

Walkthrough 1-2: Make calls to an API

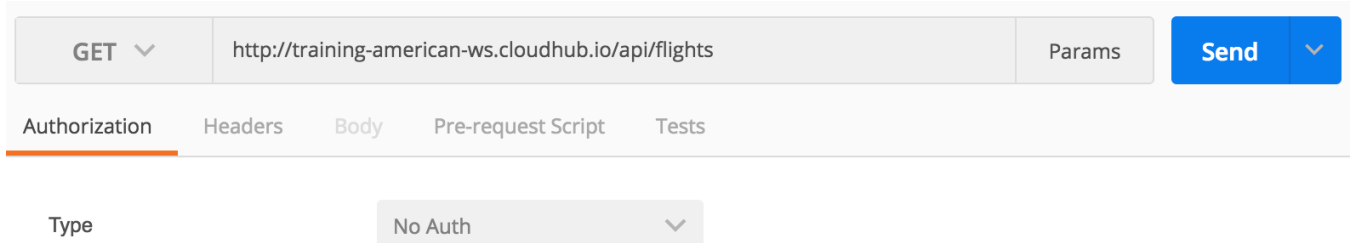
In this walkthrough, you make calls to a RESTful API. You will:

- 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).

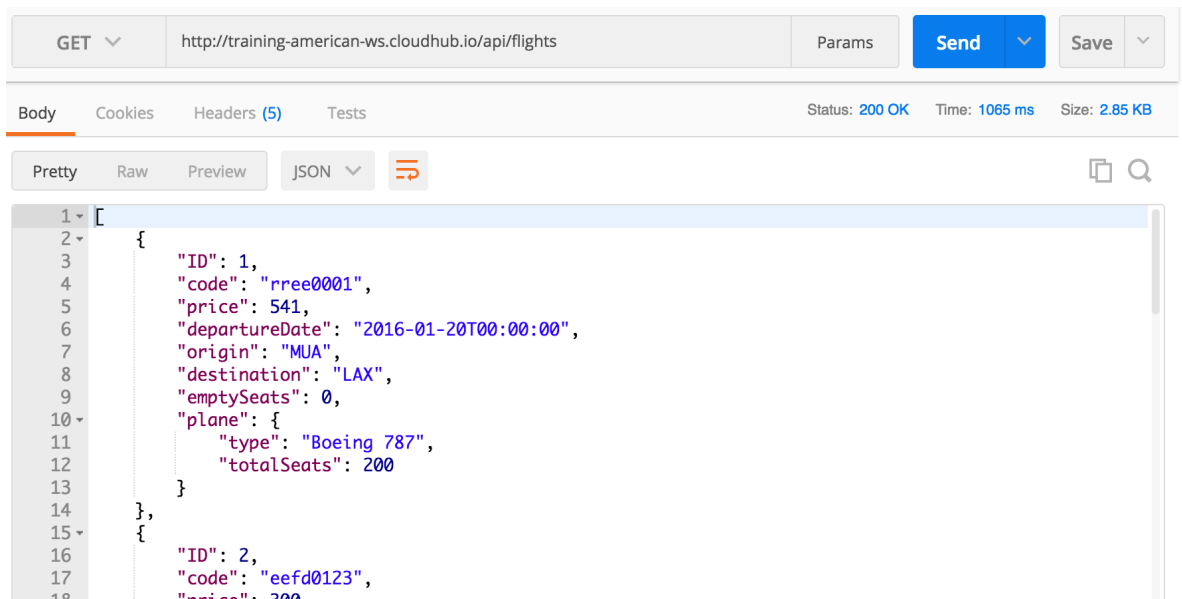


Make GET requests to retrieve data

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.



- 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.



GET ▼ http://training-american-ws.cloudhub.io/api/flights Params Send ▼ Save ▼

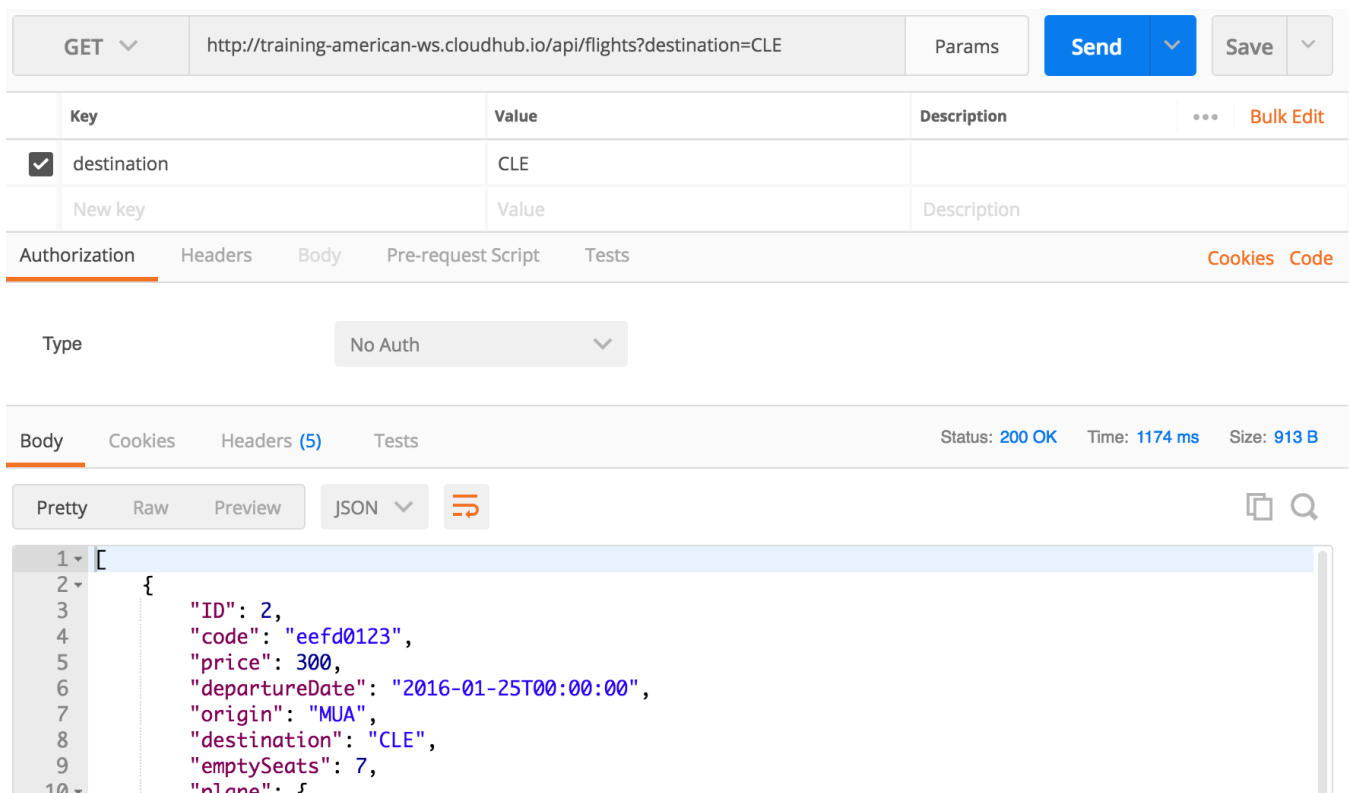
Body Cookies Headers (5) Tests Status: 200 OK Time: 1065 ms Size: 2.85 KB

Pretty Raw Preview JSON ▼ ≡

```

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
  }
]
```

- 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.



GET ▼ http://training-american-ws.cloudhub.io/api/flights?destination=CLE Params Send ▼ Save ▼

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	destination	CLE			
	New key	Value	Description		

Authorization Headers Body Pre-request Script Tests Cookies Code

Type No Auth ▼

Body Cookies Headers (5) Tests Status: 200 OK Time: 1174 ms Size: 913 B

Pretty Raw Preview JSON ▼ ≡

```

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": {
  
```

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.

GET <http://training-american-ws.cloudhub.io/api/flights/3> Params Send Save

Body Cookies Headers (5) Tests Status: 200 OK Time: 972 ms Size: 413 B

Pretty Raw Preview JSON

```
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 ]
```

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.

DELETE <http://training-american-ws.cloudhub.io/api/flights/3> Params Send Save

Body Cookies Headers (5) Tests Status: 200 OK Time: 185 ms Size: 196 B

Pretty Raw Preview JSON

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

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.

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

Body Cookies Headers (5) Tests Status: 405 Method Not Allowed Time: 106 ms Size: 198 B

Pretty Raw Preview JSON

```
1 {
2   "message": "Method not allowed"
3 }
```

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.

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

Body Cookies Headers (5) Tests Status: 415 Unsupported Media Type Time: 103 ms Size: 206 B

Pretty Raw Preview JSON

```
1 {
2   "message": "Unsupported media type"
3 }
```

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.

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

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

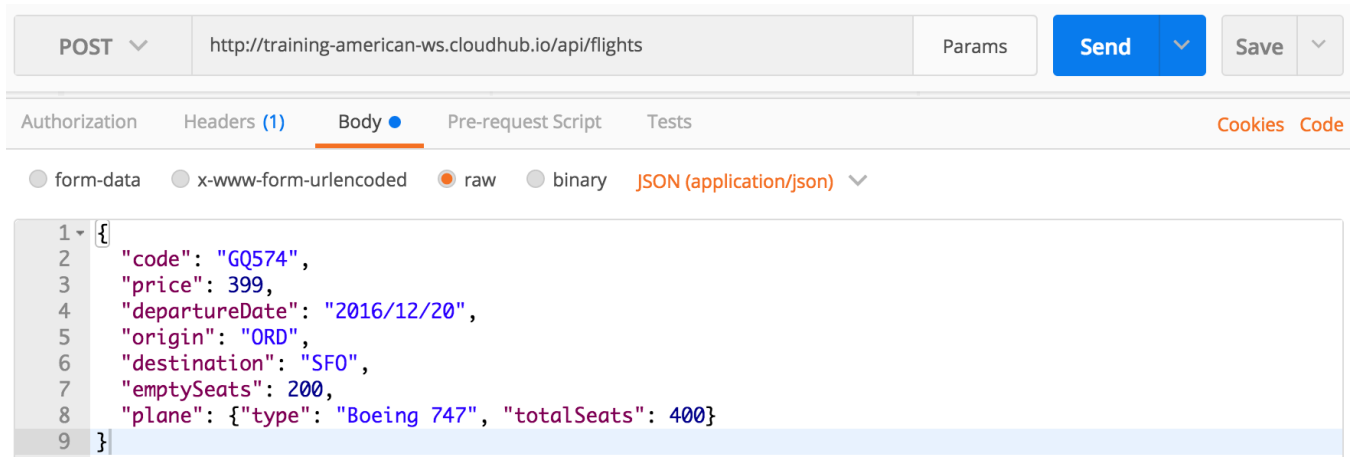
Key	Value	Description	...	Bulk Edit	Presets <input type="button" value="v"/>
<input checked="" type="checkbox"/> Content-Type	application/json				
New key	Value	Description			

24. Click the Body link under the request URL.

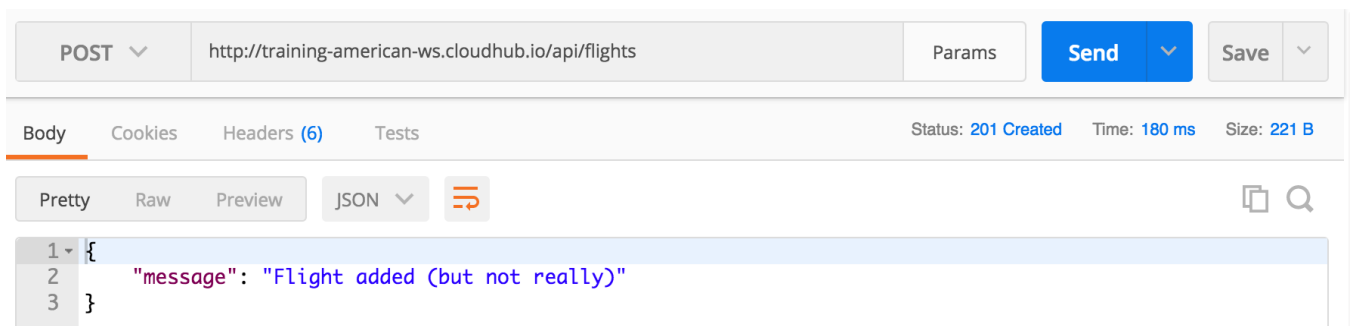
25. Select the raw radio button.

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.

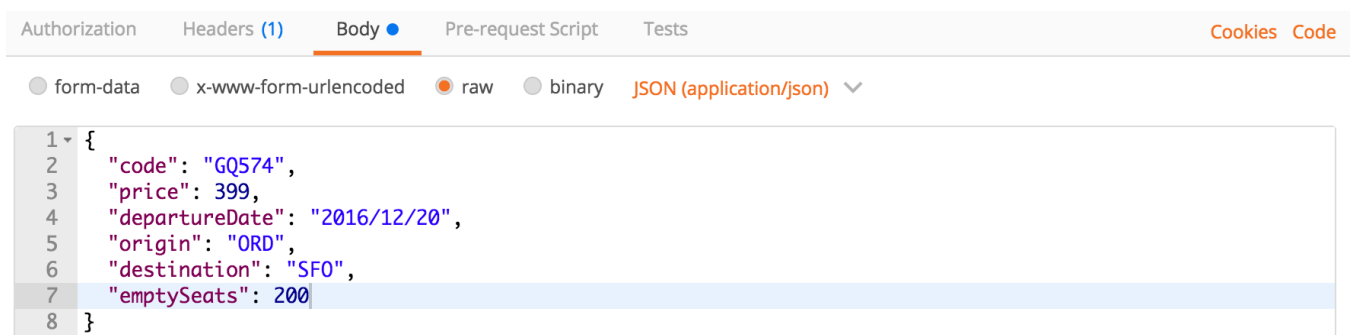


28. Click the Send button; you should see a 201 response with the message Flight added (but not really).



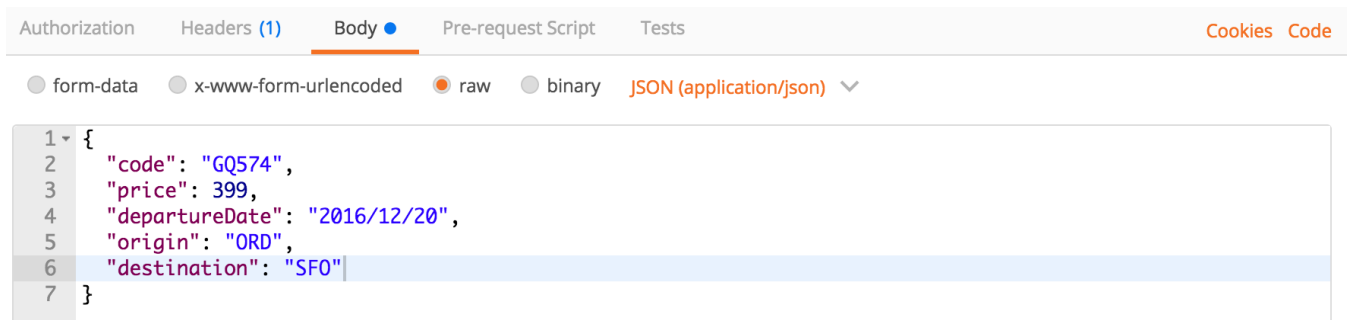
29. Return to the request body and remove the plane field and value from the request body.

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



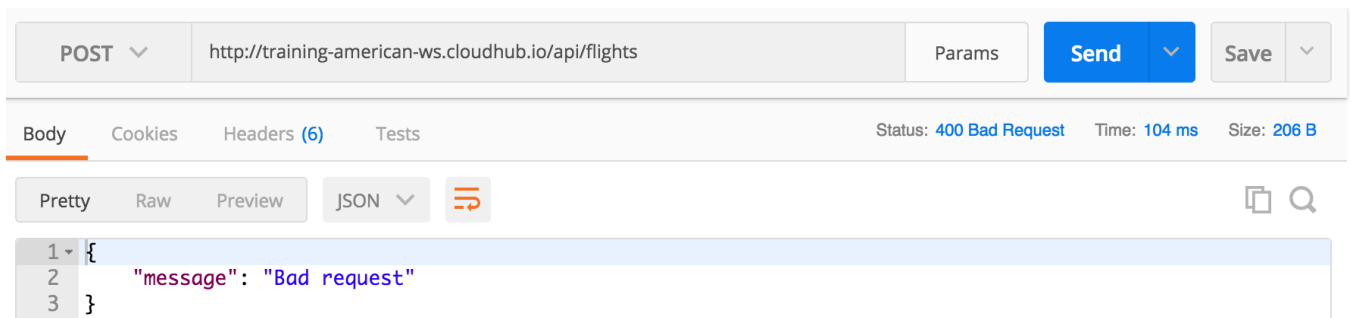
31. Send the request; the message should still post successfully.

32. In the request body, remove the emptySeats key/value pair.
33. Delete the comma after the destination key/value pair.



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

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



```
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 image shows a Postman interface for a PUT request. The URL is `http://training-american-ws.cloudhub.io/api/flights/3`. The request body is a JSON object: `{ "code": "GQ574", "price": 399, "departureDate": "2016/12/20", "origin": "ORD", "destination": "SFO", "emptySeats": 200 }`. The response status is 200 OK, with a time of 112 ms and a size of 218 B. The response body is a JSON object: `{ "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 image shows a Postman interface for a GET request. The URL is `http://training-american-api.cloudhub.io/flights`. The response status is 401 Unauthorized, with a time of 106 ms and a size of 192 B. The response body is a text 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 REST Client interface with a GET request to `http://training-american-api.cloudhub.io/flights/3?client_id=d1374b15c6864c3...`. The request is configured with two headers: `client_id` and `client_secret`. The response status is `200 OK` with a time of `1223 ms` and a size of `497 B`. The response body is displayed in JSON format:

```
{
  "ID": 3,
  "code": "ffee0192",
  "price": 300,
  "departureDate": "2016-01-20T00:00:00",
  "origin": "MUA",
  "destination": "LAX",
  "emptySeats": 0,
  "plane": {
    "type": "Boeing 777",
    "totalSeats": 300
  }
}
```

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 REST Client interface with a response status of `429 Too Many Requests`, a time of `100 ms`, and a size of `230 B`. The response body is displayed in Text format:

```
1 API calls exceeded
```