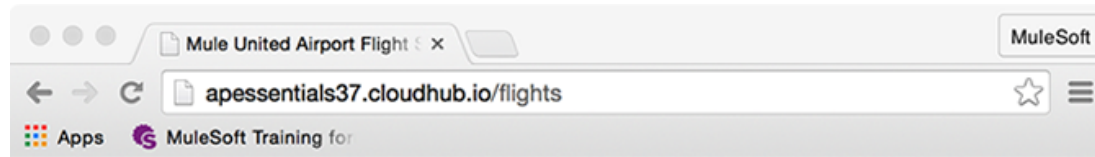




Module 11: Deploying Mule Applications



Goal



Mule United Airport

SFO - San Francisco

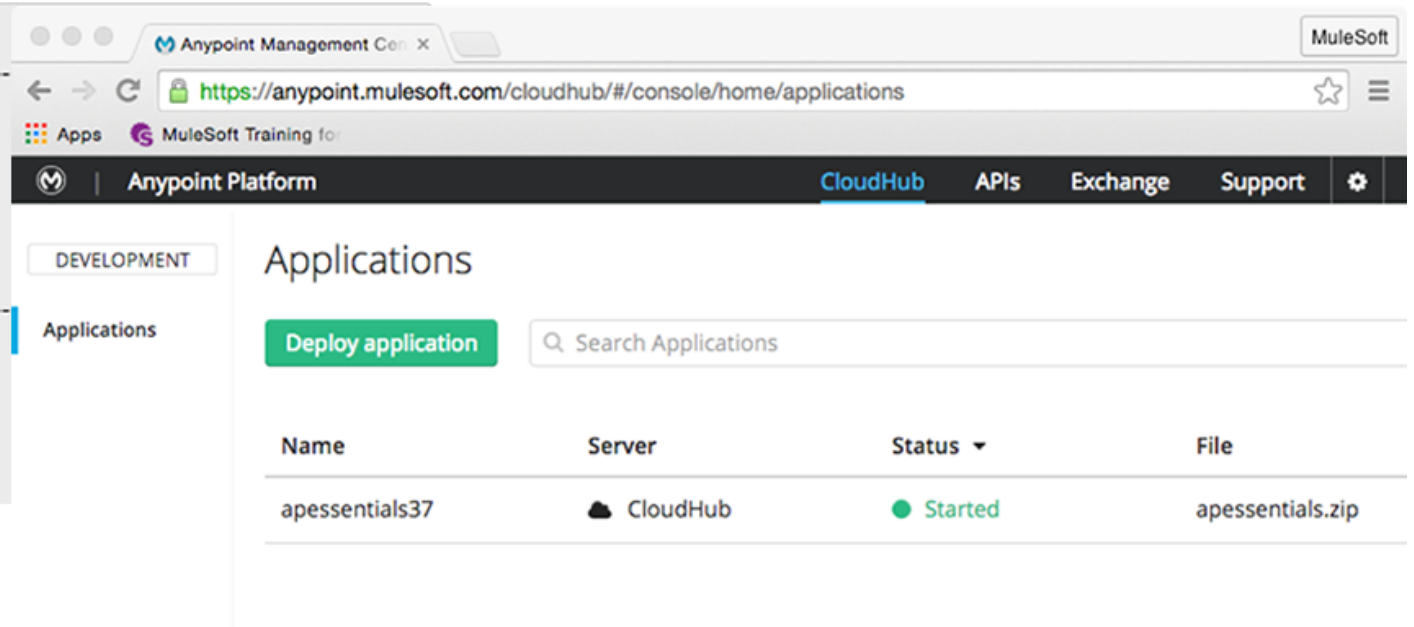
All Airlines

Find Flights

Available Flights

Flight Code: rree1093
Airline Name: American Airlines
Destination: SFO
Plane Type: Boeing 737
Price: \$142
Departure Date: 2015-02-11T00:00:00
Available Seats: 1

Flight Code: A14244
Airline Name: Delta
Destination: SFO
Plane Type: Boeing 787
Price: \$294
Departure Date: 2015/02/12



Objectives

- In this module, you will learn:
 - About the options for deploying your applications
 - About when and how to use application properties
 - What CloudHub is
 - (Optional) To deploy and run applications in the cloud
 - (Optional) To deploy and run applications on-prem

Introducing deployment options



Deploying applications

- During development, applications are deployed on an embedded Mule runtime in Anypoint Studio
- For everything else (testing, Q&A, and production), applications can be deployed to
 - On-prem Mule Server Runtime
 - As a standalone application to a Mule ESB (typically)
 - Simpler architecture and better performance
 - As a WAR file with an embedded Mule instance to an application server (not recommended)
 - CloudHub
 - Hosted Mule Server Runtime on AWS
 - Integration Platform as a Service (IPaaS)

On –prem Mule runtime features

- Easy to install
- Requires minimal resources
- Can run multiple applications
- Uses a Java Service Wrapper which controls the JVM from your operating system and starts Mule
- Mule Management Console for controlling applications
 - Deploying and undeploying applications
 - Starting and stopping servers
 - Managing and monitoring applications

What is CloudHub?

- A cloud-based integration platform as a service (iPaaS)
 - Eliminates the need to install or manage middleware or hardware infrastructure
 - Enables developers to integrate and orchestrate applications and services
 - Gives operations the control and visibility they require for mission-critical demands

CloudHub features

- Low maintenance
 - No hardware to maintain
 - No software to upgrade
 - Redundancy with 99.99% guaranteed uptime and support
- Additional out-of-the box capabilities
 - Infrastructure for DNS and load-balancing
- Global and scalable
 - Data centers around the world
- Secure
- Future-proof for hybrid cloud architectures
- Monitoring capabilities

Before deploying

- Think about anything in your application that might change between development and production...

```
12
13 <sfdc:config name="Salesforce" username="${sfdc.username}"
14           password="${sfdc.password}" securityToken="${sfdc.token}"
15           doc:name="Salesforce"/>
16
17 <db:mysql-config name="MySQL_Configuration" host="${db.host}"
18               port="${db.port}" user="${db.user}" password="${db.password}"
19               database="${db.database}" doc:name="MySQL Configuration"/>
20
```

Using application properties



Application properties

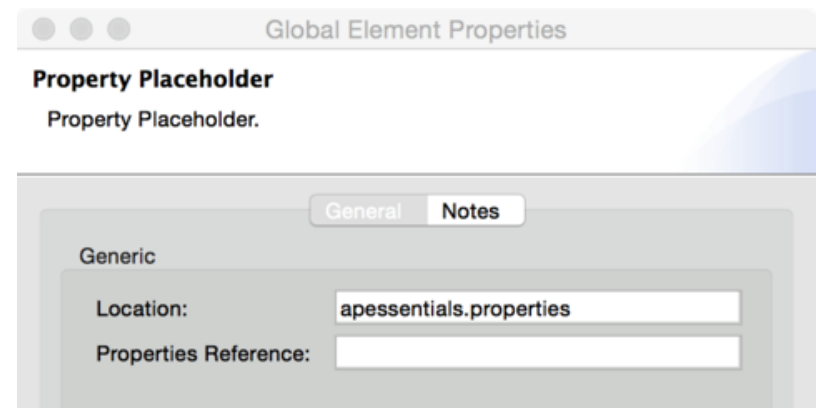
- Are an alternative to hard-coding hardcoding credentials, resources, etc.
- Are injected into the application at runtime
- Provide an easier way to manage credentials, changes, and settings
- Can be encrypted
- Are defined in .properties files
 - Separate property files can host values specific to an environment
 - app-dev.properties and app-prod.properties

Existing property files

- Mule Projects contain two property files by default
 - `src/main/app`
 - `mule-app.properties`
 - `mule-deploy.properties`
- mule-deploy is the deployment descriptor
 - Describes how the application should be deployed
- mule-app
 - Initially blank and is for custom application properties
 - Inherently loaded into CloudHub as environment variables when deploying from Anypoint Studio
 - For Mule standalone, must be passed to Mule runtime when it starts

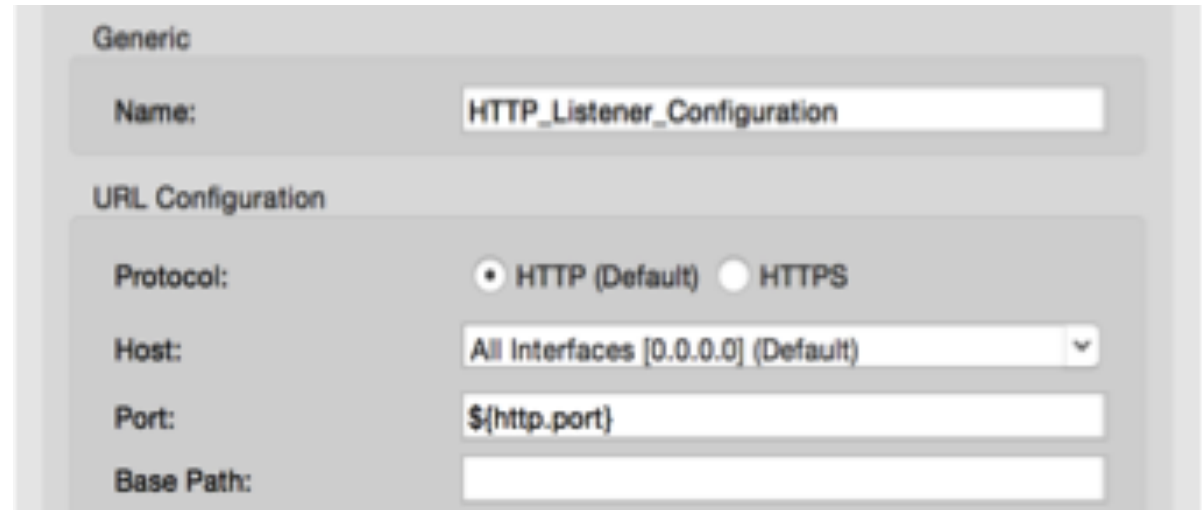
Defining application properties

- Create a custom properties file anywhere in the project
`apessentials.properties`
- Define properties in the properties file
`db.account = ReaderAccount`
- Create a Properties Placeholder global element
- Use the properties in the application
`${db.account}`



Parameterizing the HTTP Listener port

- `http.port=8081`



The screenshot shows the 'Generic' configuration section for an HTTP listener. The 'Name' field is set to 'HTTP_Listener_Configuration'. Below this is the 'URL Configuration' section. The 'Protocol' is set to 'HTTP (Default)' with a radio button. The 'Host' is set to 'All Interfaces [0.0.0.0] (Default)' with a dropdown arrow. The 'Port' is set to '\$(http.port)' in a text field. The 'Base Path' is an empty text field.

- If deploying to CloudHub, you must name this application property `http.port`
 - `http.port` is a reserved CloudHub property
 - Traffic on port 80 to a CloudHub application domain URL will be routed to the port set by this property
 - By default, `http.port` is 8081

Walkthrough 11-1: Use application properties

- Create a properties file for your application
- Create a Properties Placeholder global element
- Parameterize the HTTP Listener connector port
- Define and use Database connector properties

The screenshot displays two windows from an IDE. The left window, titled 'apessentials-dev.properties', shows a list of properties: `1 http.port=8081`, `2`, `3 db.host=localhost`, `4 db.port=4406`, `5 db.user=mule`, `6 db.password=mule`, `7 db.database=training`, `8`, `9 http.host=localhost`, `10 http.portnum=8112`, `11 http.path_united=/essentials/united/flights`, `12 http.path_bank=/build/banking/raml`, `13`, `14 sfdc.username=your@email.com`, `15 sfdc.password=your_password`, and `16 sfdc.token=your_token`. The right window, titled 'Global Element Properties', shows the 'MySQL Configuration' dialog. It has tabs for 'General', 'Advanced', and 'Notes'. The 'General' tab is active, showing a 'Name' field with 'Training_MySQL_Configuration'. Below, under 'Database configuration parameters', there are fields for 'Host' (containing '\${db.host}'), 'Port' (containing '\${db.port}'), 'User' (containing '\${db.user}'), 'Password' (containing '\${db.password}' with a 'Show password' checkbox checked), and 'Database' (containing '\${db.database}'). At the bottom are buttons for '?', 'Test Connection...', 'Cancel', and 'OK'.

```
apessentials-dev.properties x global
1 http.port=8081
2
3 db.host=localhost
4 db.port=4406
5 db.user=mule
6 db.password=mule
7 db.database=training
8
9 http.host=localhost
10 http.portnum=8112
11 http.path_united=/essentials/united/flights
12 http.path_bank=/build/banking/raml
13
14 sfdc.username=your@email.com
15 sfdc.password=your_password
16 sfdc.token=your_token
```

Global Element Properties

MySQL Configuration

MySQL configuration information.

General Advanced Notes

Generic

Name: Training_MySQL_Configuration

General

Database configuration parameters

Host: \${db.host}

Port: \${db.port}

User: \${db.user}

Password: \${db.password} ☒ Show password

Database: \${db.database}

? Test Connection... Cancel OK

Dynamically specify property files

- Resources and credentials often vary from development to production environments
- You can use a property for the location value in the Property Placeholder



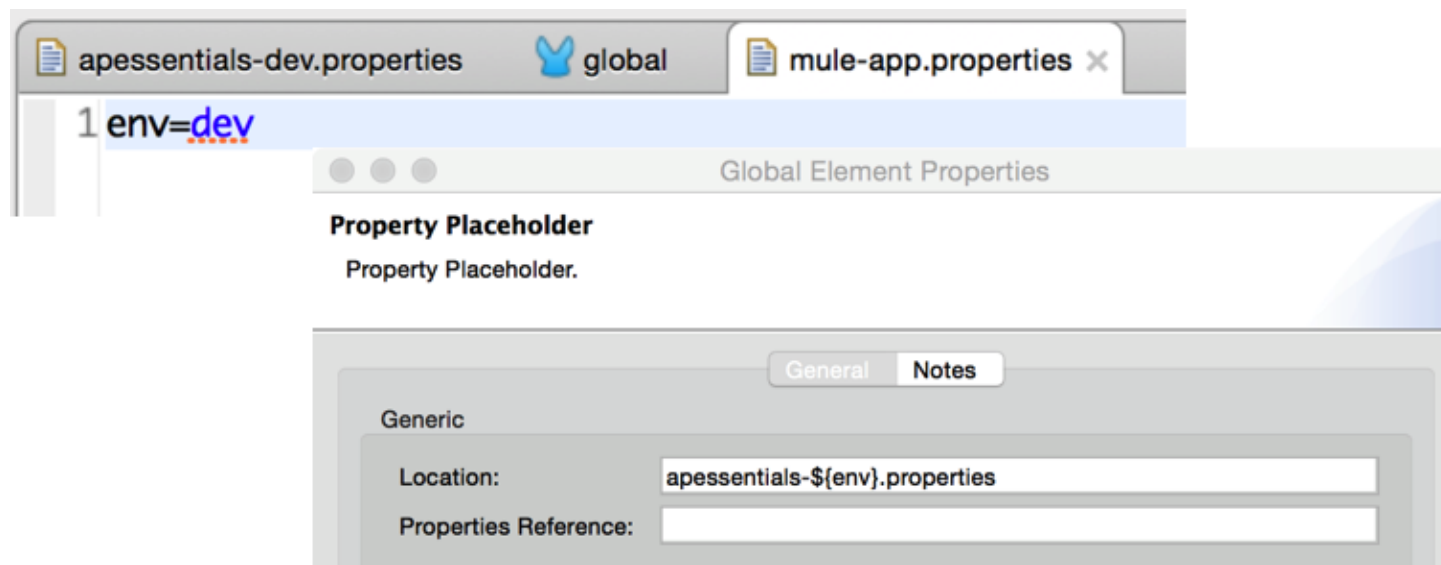
Property
Placeholder
(Global Element)

```
<context:property-placeholder  
location="appName-${env}.properties" />
```

- For development, set env in mule-app.properties

Walkthrough 11-2: Dynamically specify property files

- Create a Define an environment property value in mule-app.properties
- Use the environment property in the Property Placeholder



Setting environment variables

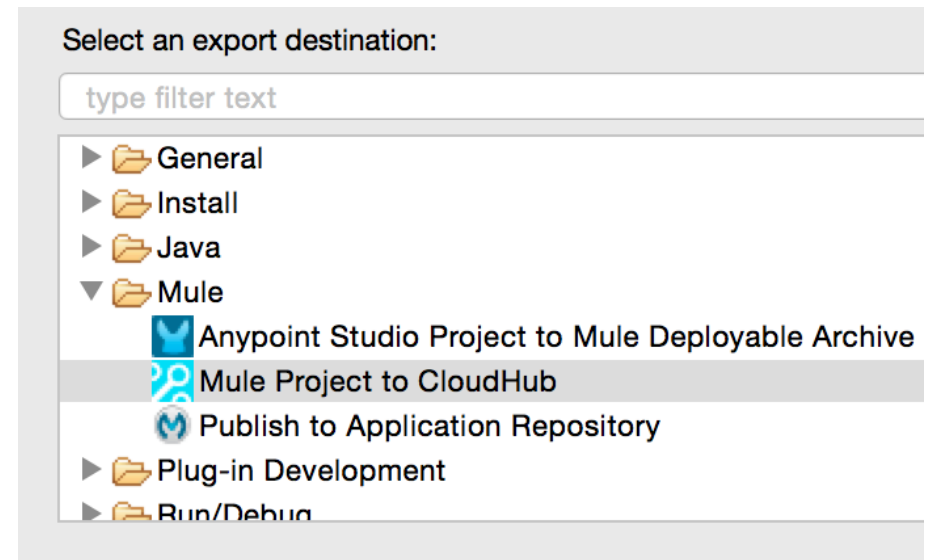
- For development, set env in mule-app.properties
- For deployment to CloudHub
 - Automatically loaded into CloudHub as environment variables
 - Can be modified in the CloudHub management console
- For Mule standalone
 - Must be passed to Mule runtime when it starts
 - Set in wrapper.conf file before starting Mule

Deploying applications to the cloud



Deploying applications to CloudHub

- From Anypoint Studio
 - Export Mule Project directly to CloudHub
 - Enter Anypoint Platform credentials
- From CloudHub
 - In Anypoint Studio, create a Mule Deployable Archive
 - On CloudHub, add an application and then upload a Mule Deployable Archive



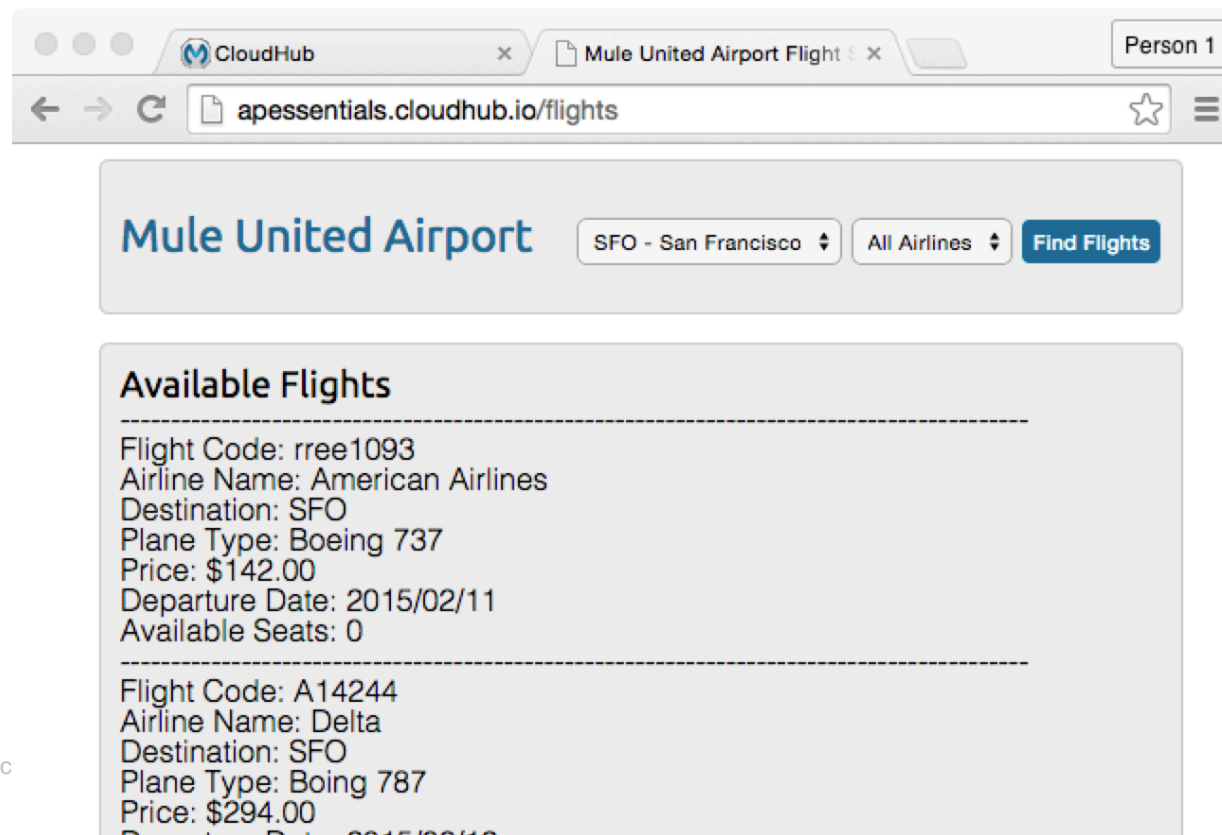
CloudHub environment variables

- Environment variables can be set from the CloudHub console after deployment
- Variables set here will override values set in .properties file

Dashboard	Environment variables	
Application Data	Name	Value
Deployment	<input type="text" value="env"/>	<input type="text" value="dev"/> ×
Insight	<input type="text" value="keyProp"/>	<input type="text" value="abc"/> ×
Logs		
Notifications		
Schedules		
Settings		
Queues		
	<input type="button" value="Add Variable"/>	

Walkthrough 11-3: (Optional) Deploy an application to the cloud

- Deploy an application to CloudHub from Anypoint Studio
- Run the application on its new, hosted domain
- View application data in CloudHub

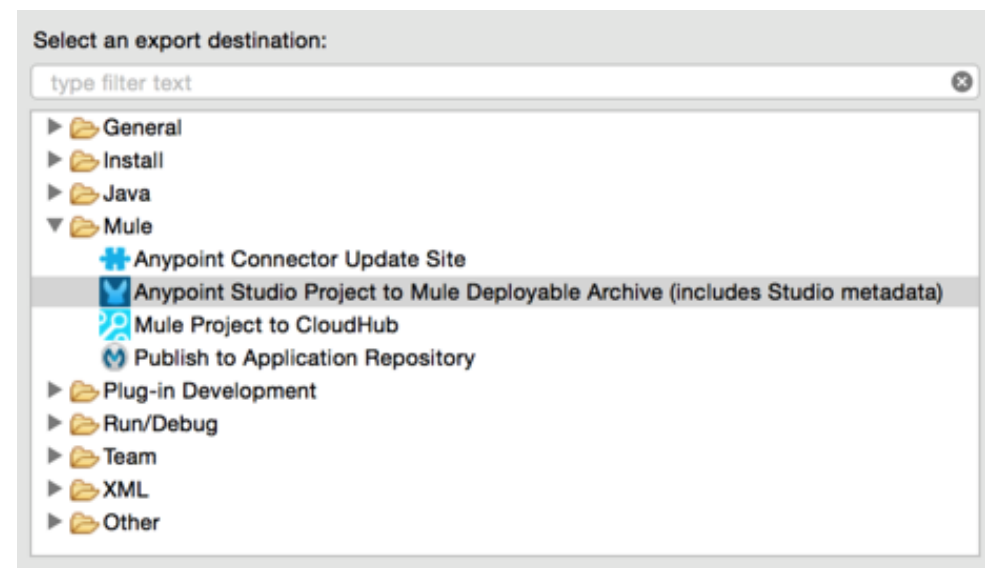


Deploying applications on-prem



Deploying applications to an on-prem Mule runtime

- In Anypoint Studio, create a Mule Deployable Archive
- Install standalone Mule runtime
- Modify wrapper.conf (to pass in environment variables)
- Start Mule
- Start Mule Management Console (MMC)
- Use MMC to deploy the application



Walkthrough 11-4: (Optional) Deploy an application on-prem

- Package an application as a Mule deployable archive
- Start Mule runtime and MMC
- Deploy an application to an on-prem Mule runtime
- Run the application

The screenshot shows the MuleSoft Mule ESB Enterprise console. The 'All Deployments' page is active, displaying a table with one deployment: 'apessentials' targeting 'Mule-3.6.0@10'. Below the table, a JSON response is shown, detailing flight information for American Airlines, Delta, and United flights from MUA to SFO.

```
{
  "departureDate": "2015/02/11",
  "airlineName": "American Airlines",
  "destination": "SFO",
  "price": 142.0,
  "planeType": "Boeing 737",
  "code": "rree1093",
  "origin": "MUA",
  "emptySeats": 10,
  "departureDate": "2015/02/12",
  "airlineName": "Delta",
  "destination": "SFO",
  "price": 294.0,
  "planeType": "Boeing 787",
  "code": "Al4244",
  "origin": "MUA",
  "emptySeats": 10,
  "departureDate": "2015/02/20",
  "airlineName": "American Airlines",
  "destination": "SFO",
  "price": 300.0,
  "planeType": "Boeing 737",
  "code": "rree2000",
  "origin": "MUA",
  "emptySeats": 10,
  "departureDate": "2015/03/20",
  "airlineName": "United",
  "destination": "SFO",
  "price": 400.0,
  "planeType": "Boeing 737",
  "code": "ER38sd",
  "origin": "MUA",
  "emptySeats": 10,
  "departureDate": "2015/03/20",
  "airlineName": "Delta",
  "destination": "SFO",
  "price": 400.0,
  "planeType": "Boeing 737",
  "code": "AlB2C3",
  "origin": "MUA",
  "emptySeats": 10
}
```

Summary



Summary

- In this module, you learned to deploy Mule applications
- Use application properties to avoid hard-coding endpoint properties, credentials, resources and so on
- Define application properties in a .properties file whose location is specified in a Properties Placeholder global element
- Dynamically specify a properties file when the application starts by parameterizing its name and setting the variable
 - As an application property with the CloudHub console
 - As an argument in the Mule standalone wrapper.conf file

Summary

- Deploy an application to the cloud
 - Directly from Anypoint Studio by exporting a project and entering your Anypoint Platform credentials
 - From the CloudHub console by creating a deployable archive in Anypoint Studio and then uploading the archive to a CloudHub application
- Deploy an application on-prem
 - By creating a deployable archive in Anypoint Studio and then uploading the archive using the Mule Management Console (MMC)