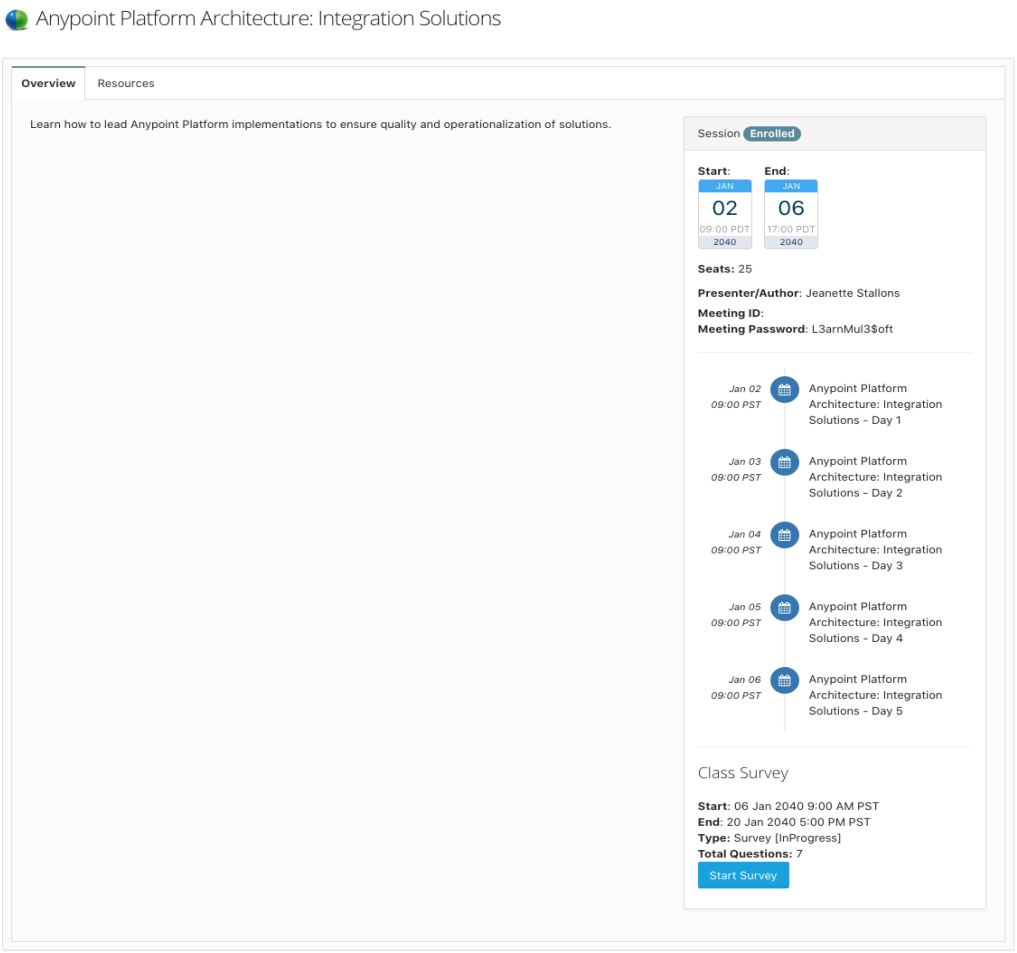
Anypoint Platform Architecture: Integration Solutions

Student Manual

Mule runtime 4.3
February 10, 2020

TABLE OF CONTENTS

Introducing the Course



In this module, you will:

- Learn about the course format.
- Download the course files.
- Make sure your computer is set up for class.
- Review the course outline.

Walkthrough: Set up your computer for class

In this walkthrough, you make sure your computer is set up correctly, so you can complete the class exercises. You will:

- Download the course files from the MuleSoft Training Learning Management System.
- Make sure Anypoint Studio starts successfully.
- Make sure you have an active trial Anypoint Platform account.
- Download the standalone Mule runtime.
- Install Advanced REST client (if you did not already).
- Download and install Maven.
- Download and install the latest version of Apache JMeter.
- Download and install VisualVM.

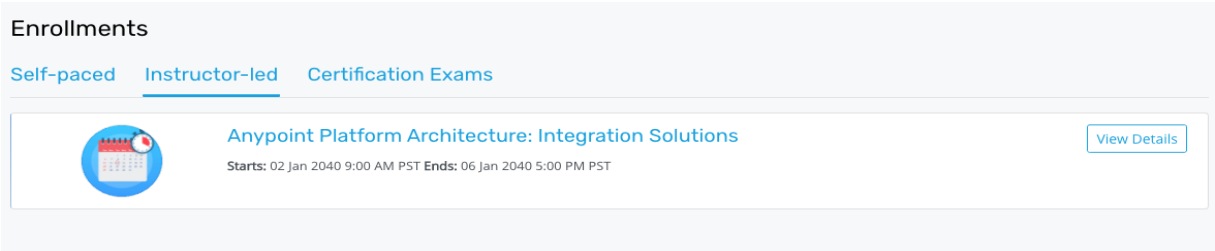
Download student files

1. In a web browser, navigate to http://training.mulesoft.com.
2. Click the Login and select Training.

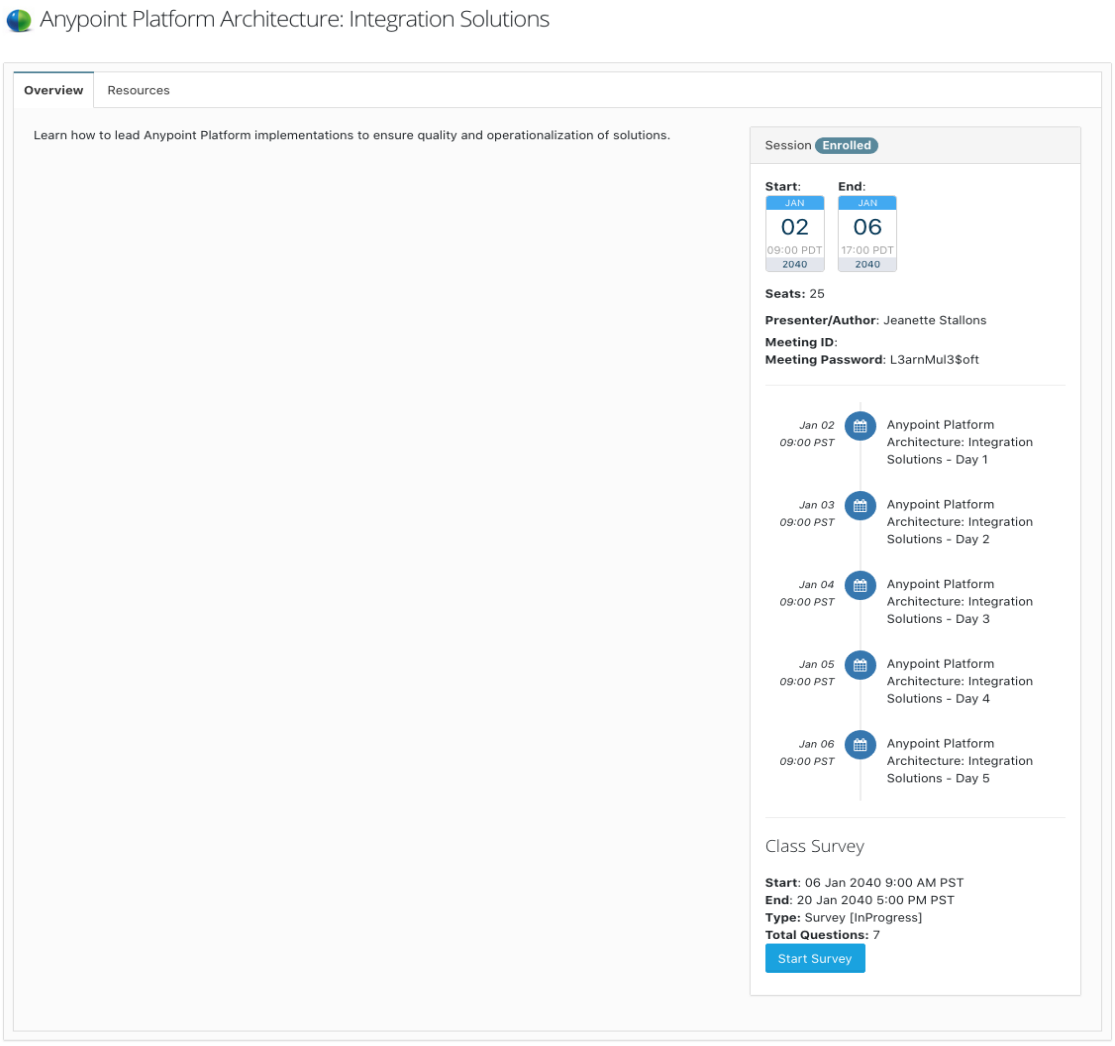
3. Log in to your MuleSoft training account using the email that was used to register you for class.

   Note: If you have never logged in before and do not have a password, click the Forgot your password link, follow the instructions to obtain a password, and then log in.

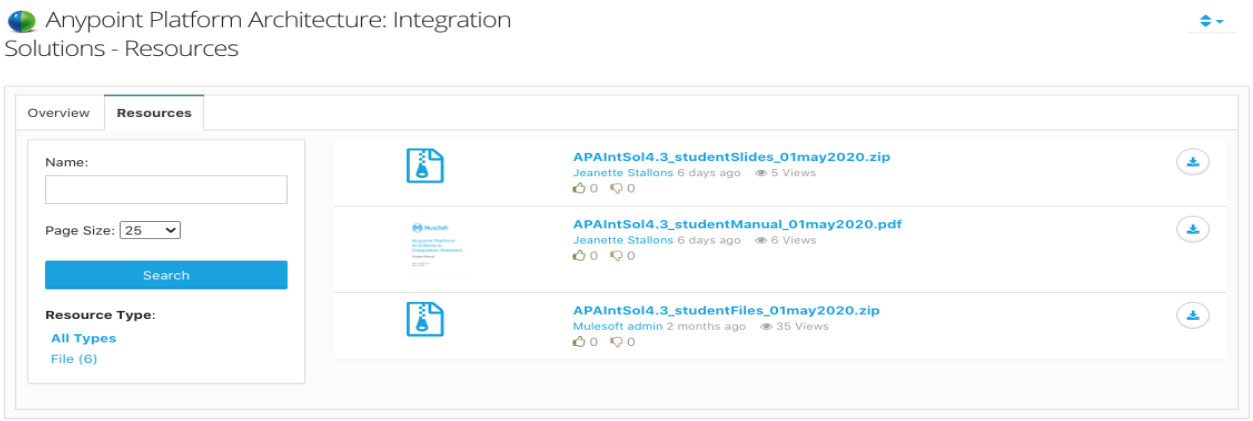
4. On the Dashboard page, select Instructor-led under Enrollments then locate the card for your class.



5. Click the course name to see the course overview page then locate the class details on the right side of the page.



6. Click the Resources tab then locate the list of course materials on the right side of the page.



7. Click the student files link to download the files to an easy to find location on your computer, such as $HOME/apaintsol; this location will be referred to as $APAINTSOL_HOME.
8. Click the student manual link to download the manual to the same $APAINTSOL_HOME location.
9. Click the student slides link to download the slides to the same $APAINTSOL_HOME location.
10. On your computer, locate the student files ZIP and expand it to an easy to find location, such as $APAINTSOL_HOME; this location will be referred to as the studentFiles location.
11. Browse to the $APAINTSOL_HOME/studentFiles/ and copy mcial1 folder and place it under .m2/com/mulesoft/training folder.
12. Open the course snippets.txt file.

    Note: Keep this file open. You will copy and paste text from it during class.

Start Anypoint Studio

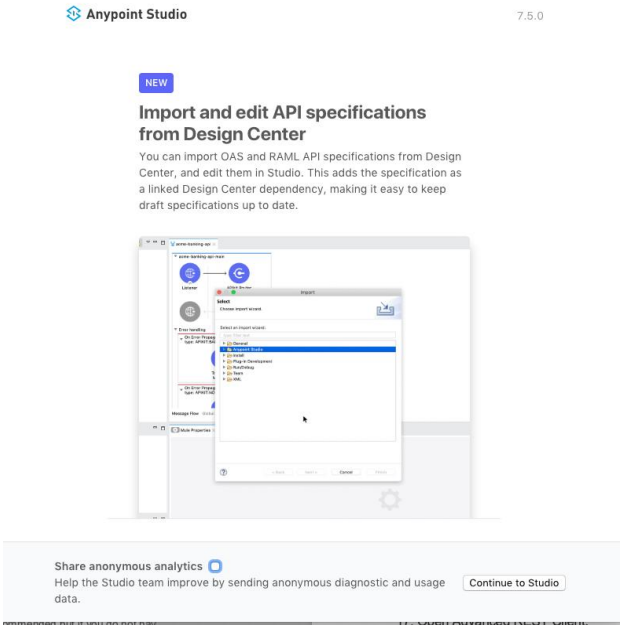
13. In your computer's file browser, navigate to where you installed Anypoint Studio and open it.

    Note: If you do not have Anypoint Studio, you can download it from https://www.mulesoft.com/lp/dl/studio. Upon starting Anypoint Studio, users on Windows may get a popup asking to allow Windows Defender Firewall access for OpenJDK; access should be allowed.

14. In the Workspace Launcher dialog box, look at the location of the default workspace; change the workspace location if you want.
15. Click OK to select the workspace; Anypoint Studio should open.

    Note: If you cannot successfully start Anypoint Studio, make sure that you have enough available memory (at least 8GB available) to run Anypoint Studio.

16. If you get a new features page, click the Continue to Studio button to close it.



17. If you get an Updates Available popup in the lower-right corner of the application, click it and install the available updates.

Make sure you have an active trial Anypoint Platform account

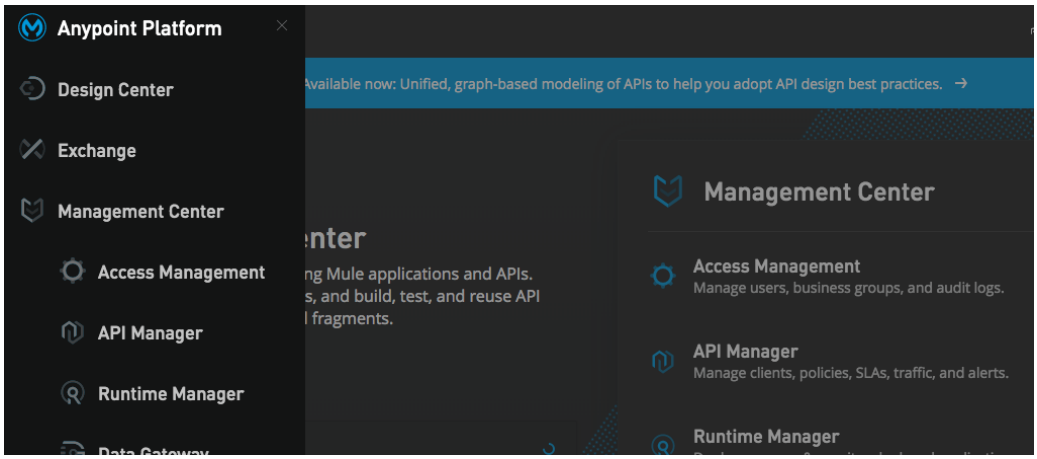
18. In a web browser, navigate to http://anypoint.mulesoft.com/ and log in.

    Note: If you do not have an account, sign up for a free, 30-day trial account now.

19. Click the menu button located in the upper left in the main menu bar.

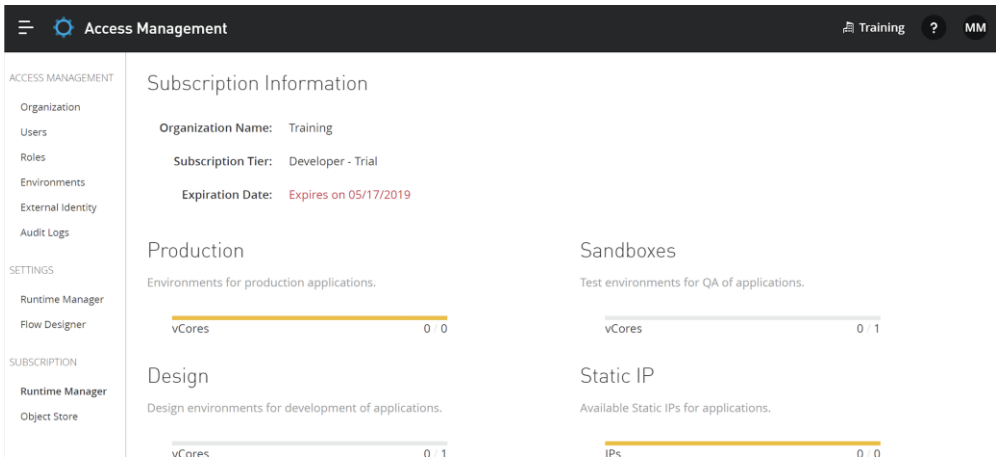
20. In the menu that appears, select Access Management.

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



21. In the left-side navigation, click the Runtime Manager link under Subscription.
22. Check your subscription level and if it is a trial account, make sure it is not expired.

    Note: If your trial is expired or will expire during class, sign out and then sign up for a new trial account now.



Locate your standalone Mule runtime installation

23. On your local computer, locate the standalone Mule 4 runtime zip file; it should begin with the prefix mule-ee-distribution-standalone.

    Note: If you did not yet download the standalone Mule 4 runtime zip file, download it now from https://www.mulesoft.com/lp/dl/mule-esb-enterprise.

24. Copy or move the Mule runtime zip file to the $APAINTSOL_HOME folder.

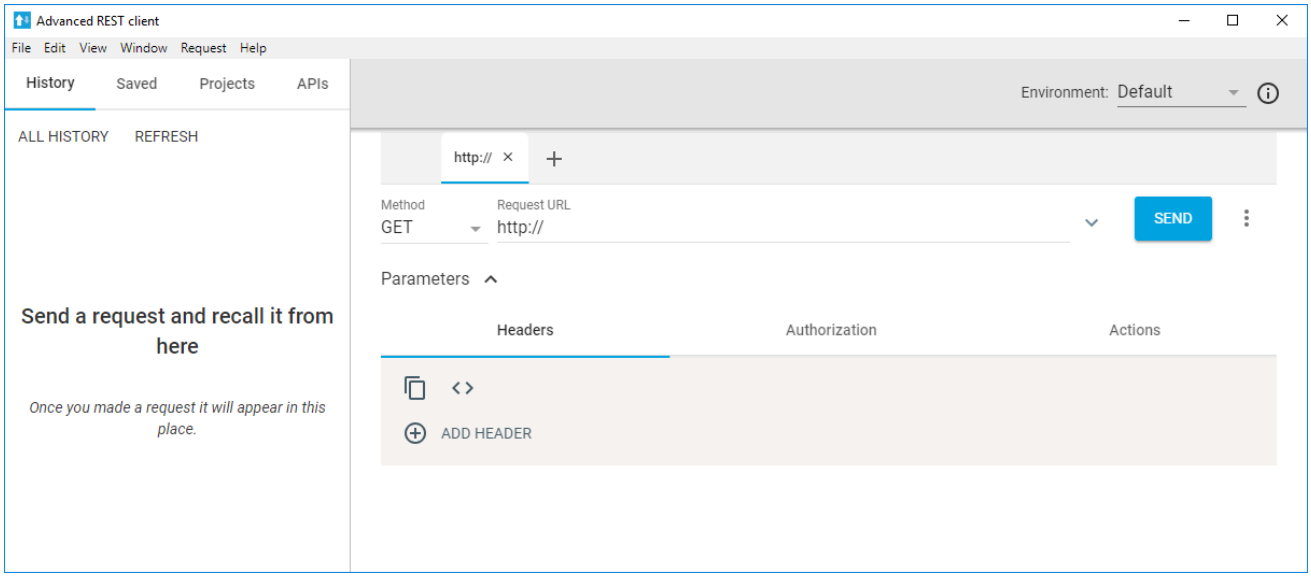
Open Advanced REST Client

25. Open Advanced REST Client.

    Note: If you do not have Advanced REST Client (or another REST API client) installed, download it now from https://install.advancedrestclient.com/ and install it.

    Note: If you are having proxy issues, try to use the chrome browser version of Advanced REST Client, or use some other REST client.

26. Leave Advanced REST client open; you will use it throughout class.

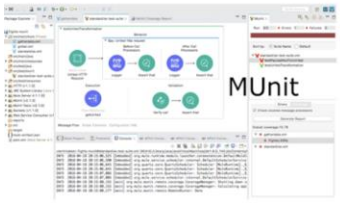


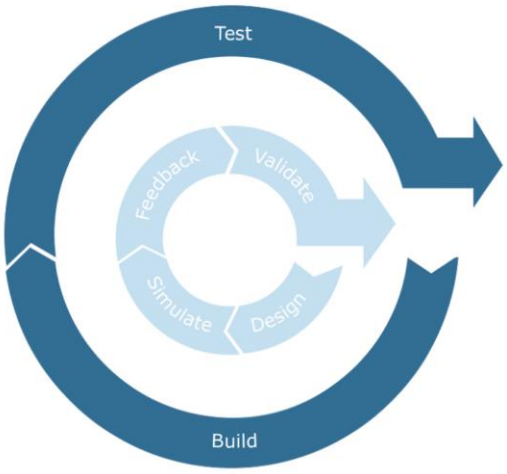
Download and install VisualVM

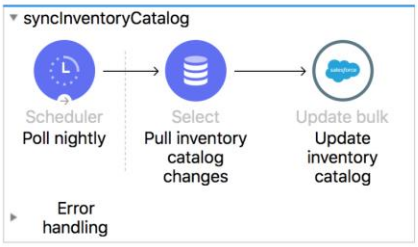
27. On your local computer, locate the visualvm.exe file and double-click it.

    Note: If you did not yet download VisualVM, download it now from https://visualvm.github.io/download.html.

PART 1: Architecting and Designing Integration Solutions



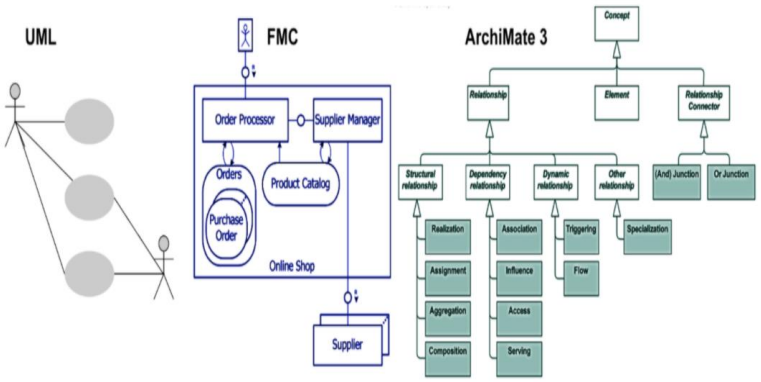




At the end of this part, you should be able to:

- Architect integration solutions.
- Identify Anypoint Platform components and capabilities.
- Design integration solutions using Mule applications.
- Choose appropriate Mule processing models.
- Choose Mule event transformation and routing patterns.
- Design testing strategies for Mule applications.

Module 1: Introducing Integration Solutions Architectures



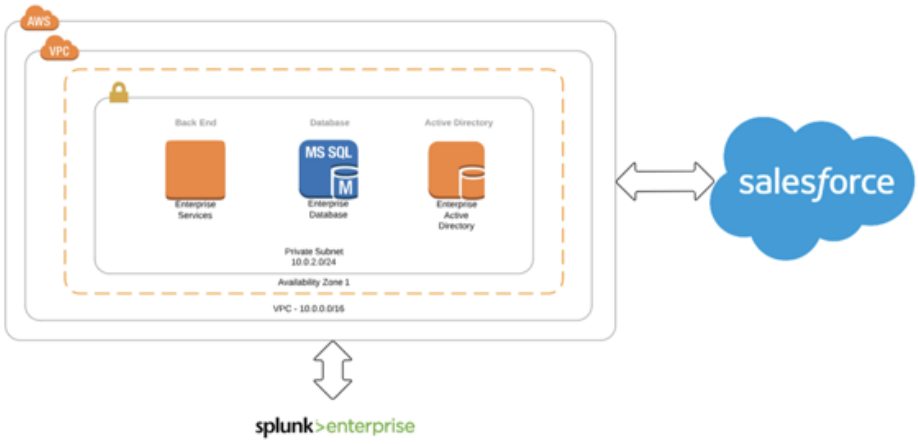
At the end of this module, you should be able to:

- Describe the objectives of enterprise integration solution.
- Summarize how to architect for customer success with Anypoint Platform.
- Describe how integration solutions using Anypoint Platform are documented.
- Understand key parts of an architecture template for integration solutions.

Exercise 1-1: Identify stakeholders

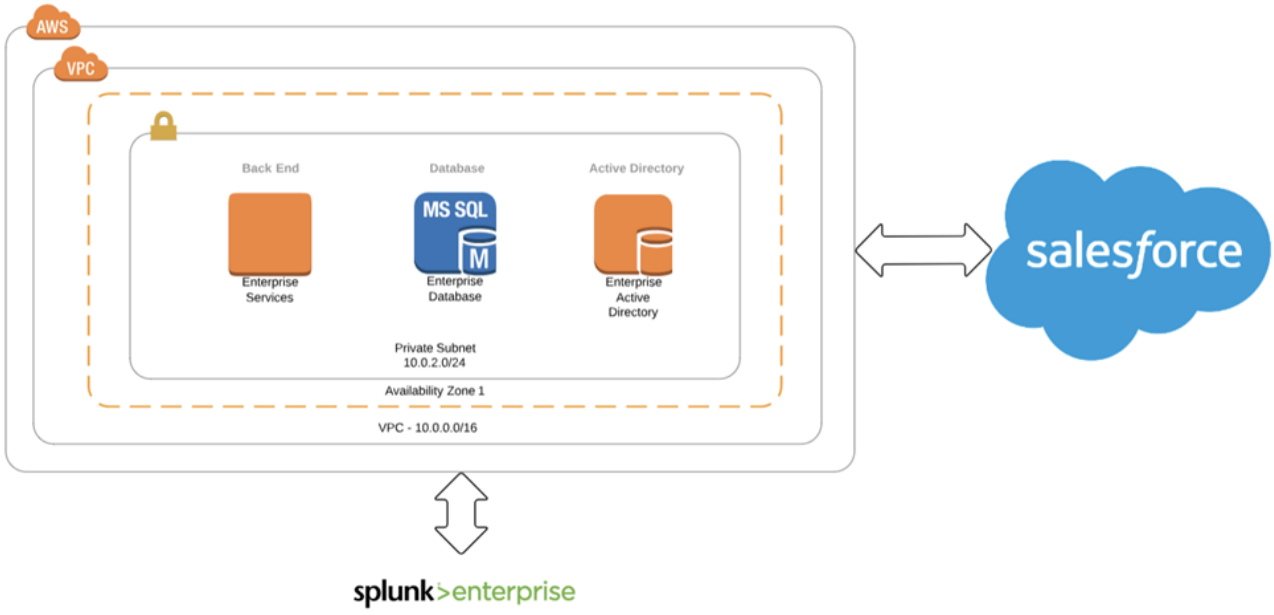
In this exercise, you look broadly at an example integration solution and speculate on the various stakeholders and their responsibilities and interests in different parts of the integration solution. You will:

- Identify project stakeholders in a typical integration solution and describe their responsibilities.
- Identify integration endpoint stakeholders in a typical integration solution and describe their responsibilities.



Review the overall integration solution architecture

1. Review the architecture diagram for the integration solution for this exercise; you will use this to answer the exercise questions.



2. List network infrastructure used by the integration solution, including systems or components not shown in this diagram.

3. List external endpoints used by this integration solution.

Describe stakeholders directly involved in the integration solution project

4. Fill in the stakeholders directly involved in the overall integration solution, and describe their roles and responsibilities.

Stakeholder	Roles and responsibilities
Project sponsor	Drives the project
Integration architects	Responsible for implementing this and other integration solutions

Describe stakeholders for the integration endpoints

5. Fill in the stakeholders for the integration endpoints, and describe their roles and responsibilities.

Stakeholder	Roles, responsibilities, primary concerns
Users	Often consume results of the integration solution
Database administrators	Control access to data Defines SLAs related to data access Control polling intervals or other more real-time options

Exercise 1-2: Identify characteristics of user stories for an integration solution

In this exercise, you expand on the sample architecture to identify various user stories that might be associated with the integration solution project. You will:

- Identify characteristics of an informal integration user story.
- Distinguish between epics and stories in integration solution projects.
- Identify additional details and acceptance criteria that might be required by user stories.

Epic	Story	Story time estimates
Single sign on	**Name:** LDAP/Anypoint Platform Integration POC **As an:** LDAP administrator **I want to:** demonstrate Anypoint Platform users can be authenticated via the corporate LDAP server **so I can:** assist in the single sign-on epic	2 weeks

Identify more general or informal integration user stories

1. Write down an informal user story related to the stakeholders you identified in the previous exercise; to do this consider what is the "minimum viable product" for the user story.

User story name:
As a:
I want to:
So that I can:

Distinguish between epics and stories in an integration solutions project

2. Describe some epics related to the exercise 1-1 solution, and then identify the types of user stories that might be part of this epic.

Epic	Story	Story time estimates
Single sign on	**Name:** LDAP/Anypoint Platform Integration POC **As an:** LDAP administrator **I want to:** demonstrate Anypoint Platform users can be authenticated via the corporate LDAP server **so I can:** assist in the single sign-on epic	2 weeks
Single sign on	**Name:** **As a:** I **want to:** **so I can:**	
Single sign on	**Name:** **As a:** I **want to:** **so I can:**	

Identify more specific integration user stories

3. Review the user stories from the last two exercise steps, and add other integration specific details or other details to the user stories.

User Story:	Acceptance criteria
As a	Ensure that _____ is able to: • • • • • •

I want/need to	
So that I can	

User Story:	Acceptance criteria
As a	Ensure that _____ is able to: • • • • • • •
I want/need to	
So that I can	

User Story:	Acceptance criteria
As a	Ensure that _____ is able to: • • • • • • •
I want/need to	
So that I can	

Answer these reflection questions

4. What might be the form of an informal integration user story?

Hint: What would be the minimum viable product delivered for a user story?

5. How might integration projects use epics vs. stories?

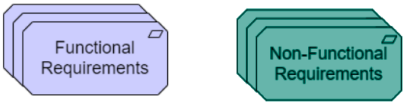
6. When might more detail be added to a user story?

7. What are some examples of integration specific detail that might be needed?

Exercise 1-3: Identify architecture documentation requirements

In this exercise, you review some example integration solution architecture documents and describe the parts of these documents. You will:

- Review an example architecture template.
- Identify the parts of an example architecture template.
- Critique an example architecture template.



Review an example integration solutions architecture template

1. Write down an informal user story related to the stakeholders you identified in the previous exercise; to do this consider what is the "minimum viable product" for the user story.

   Note: Your instructor may provide you with an already completed integration solution architecture document.

Identify section and components of the example architecture template

2. Summarize the various sections and components of the architecture template.

Architecture doc section or component	Summary and comments	How is this useful or not useful?

Evaluate and critique the example architecture template

3. In the previous table, add your own evaluation and comments about the sections and components of the architecture template.

Answer these reflection questions

4. Which sections are most helpful or essential?

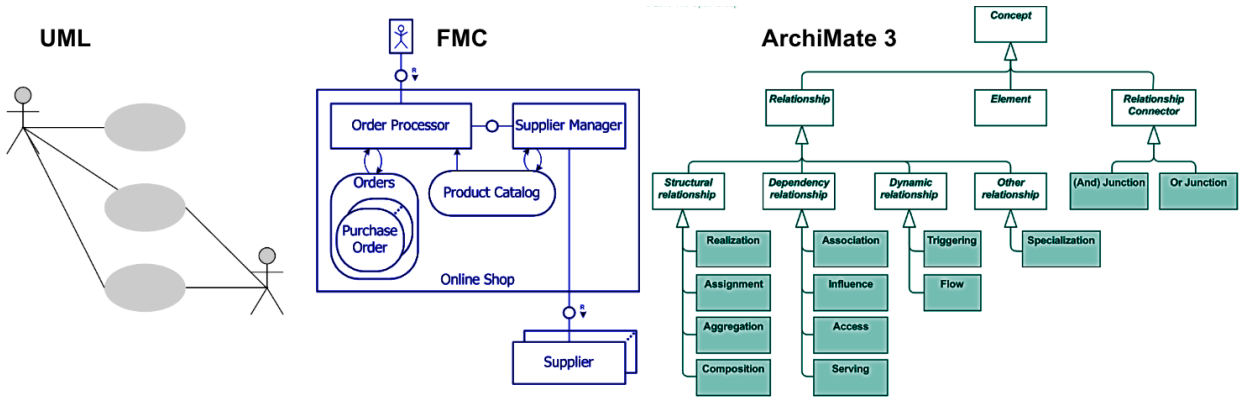
5. Are there any sections missing?

6. Which sections might you make optional/leave out?

Walkthrough 1-4: Review notation standards for use in integration solutions documentation

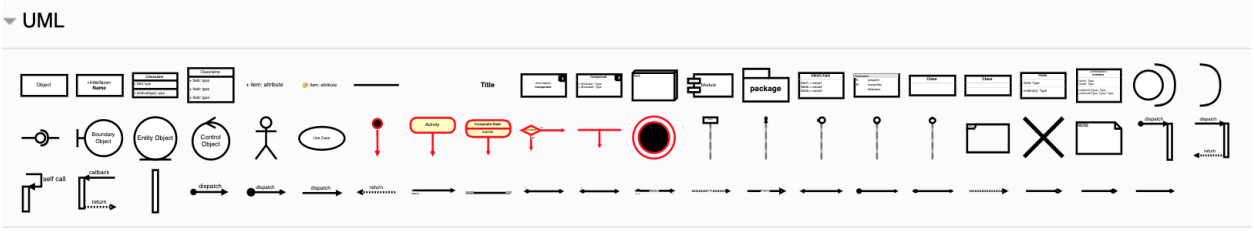
In this exercise, you connect to some online references for various architecture diagramming tools. You will:

- Compare and reflect on architecture diagramming standards.
- Explore UML, Archimate 3, FMC, C4, and BMPN 2.
- Create ad hoc “Boxes and Lines”.



Explore Unified Modeling Language (UML) diagrams

1. In a web browser, navigate to the draw.io description of UML at https://about.draw.io/uml-diagrams/.
2. In another web browser tab or window, navigate to the draw.io site with the UML shape library loaded at https://www.draw.io/?splash=0&libs=uml.
3. On the left side, expand the UML shape library.

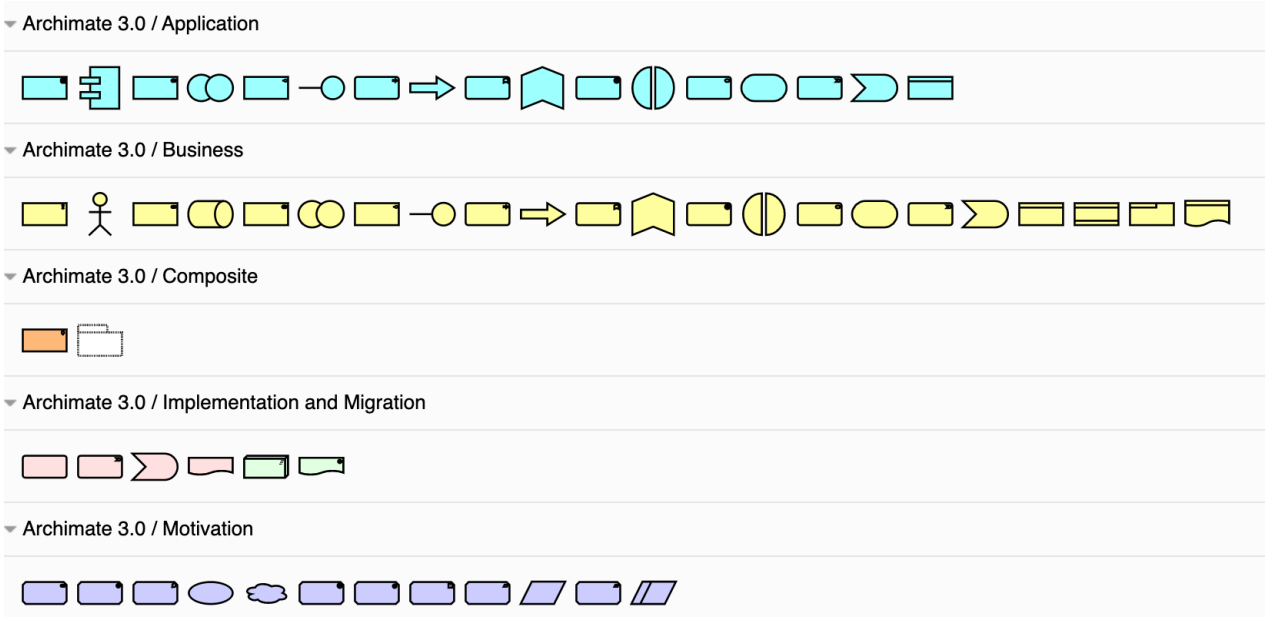


4. Click some of the shapes and connect them together in the diagram.

Explore ArchiMate 3 diagrams

5. In another web browser tab or window, navigate to the draw.io site with the Archimate3 shape library at https://www.draw.io/?splash=0&libs=archimate3.

6. In the new draw.io page, expand the Archimate 3.0 shape libraries.

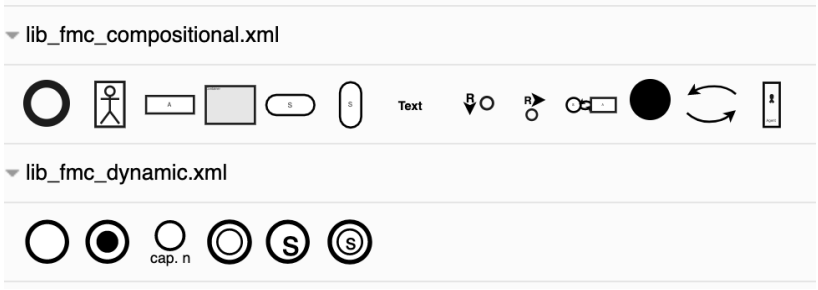


7. Click some of the shapes and connect them together in the diagram.

   Note: Notice that color is part of the modeling language. You can learn more about ArchiMate 3 usage in the Anypoint Platform Architecture: Application Networks course.

Explore Fundamental Modeling Concepts (FMC) notation diagrams

8. In another web browser tab or window, navigate to http://www.fmc-modeling.org/notation_reference.
9. Read about the Fundamental Modeling Concepts (FMC) notation.
10. In another web browser tab or window, navigate to https://gist.github.com/drozzy/1ce8fb9e2245519ceafe3c6c21867265.
11. Scroll to the bottom of the page, and in the Installation section, click each link to load the FMC libraries into draw.io.
12. In the new draw.io page that opens, expand each of the FMC shape libraries.



13. Click some of the shapes and connect them together in the diagram.

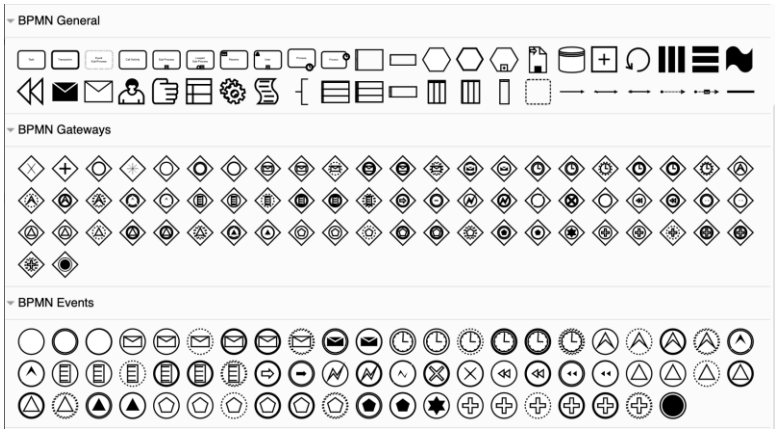
Explore the C4 model for visualizing software architecture

14. In another web browser tab or window, navigate to the C4model.com website at https://c4model.com/.
15. In another web browser tab or window, navigate to the draw.io site with the C4 shape library loaded at https://www.draw.io/?splash=0&clibs=Uhttps%3A%2F%2Ftobiashochguertel.github.io%2Fc4-draw.io%2Fc4.js.
16. Note: If this link does not automatically load the C4 shape library, follow the instructions at https://github.com/tobiashochguertel/c4-draw.io.
17. In the new draw.io page, expand the C4 Notation shape library.

18. Click some of the shapes and connect them together in the diagram.

Read about BPMN 2.0

19. In another web browser tab or window, navigate to the C4model.com https://www.omg.org/cgi-bin/doc?dtc/10-06-02.
20. Download the PDF document.
21. Open and review the PDF document.
22. In another web browser tab or window, navigate to the draw.io site with the BPMN shape library loaded at https://www.draw.io/?splash=0&libs=bpmn.
23. In the new draw.io page, expand the BPMN shape libraries.



24. Click some of the shapes and connect them together in the diagram.

Answer these reflection questions

25. Which of these diagramming options appeals to you, and why?

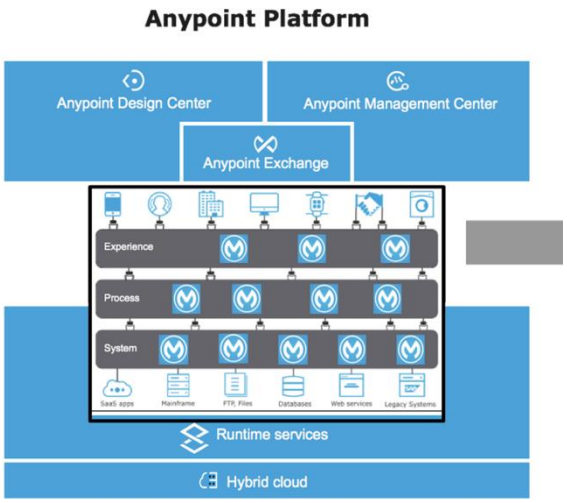
26. Which of these diagramming tools are less useful, and why?

27. What other diagramming tools or options have you used or do you prefer, and why?

28. Are these diagram/notation formats exclusive from each other, or can they be combined?

29. Which diagramming tool is beneficial over your own boxes and lines?

Module 2: Identifying Anypoint Platform Components and Capabilities



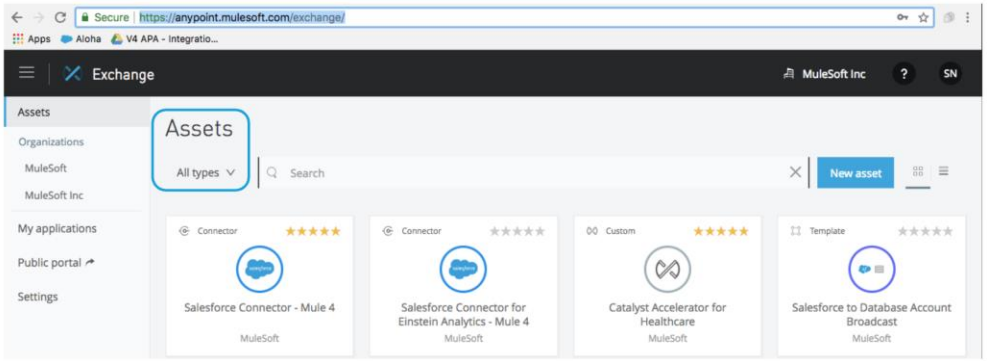
At the end of this module, you should be able to:

- Identify overall design intentions of Anypoint Platform.
- Review Anypoint Platform capabilities and high-level components.
- Review API-led development techniques and options.
- Distinguish between Anypoint Platform service and deployment models.

Walkthrough 2-1: Explore APIs in Anypoint Exchange

In this exercise, you log in to Anypoint Platform and explore Anypoint Exchange. You will:

- Review asset types supported by Anypoint Exchange.
- Review resources defined in a REST API.
- Identify API dependencies and RAML fragments.
- Identify how different versions of an API are stored in Anypoint Exchange.

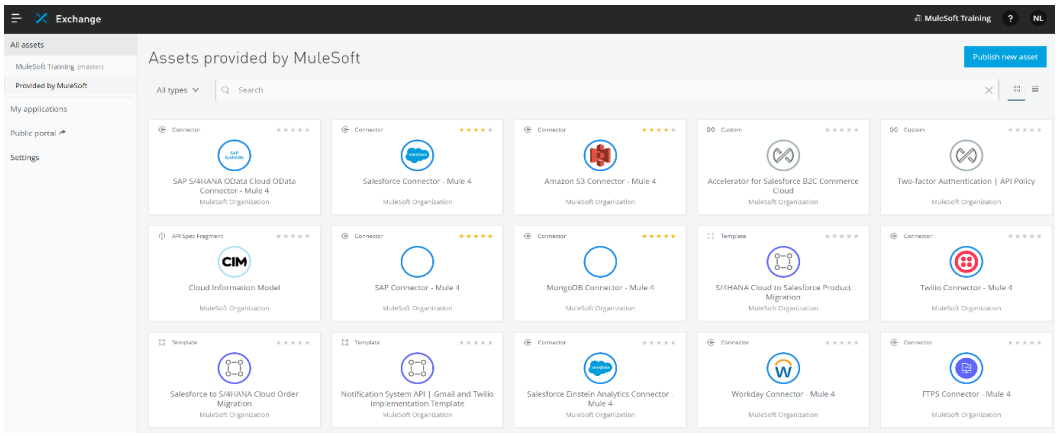


Identify Anypoint Exchange asset types

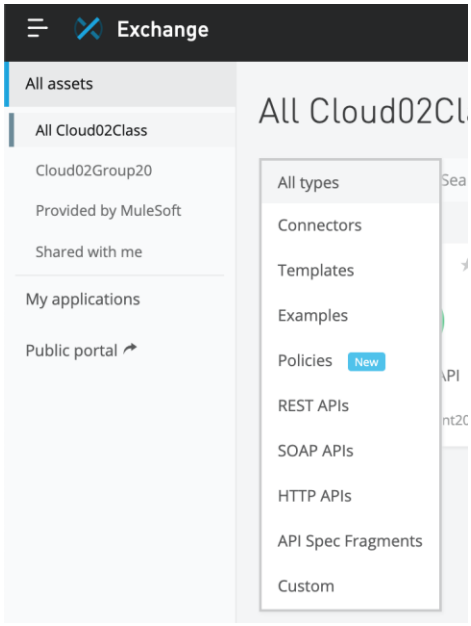
1. In a web browser, navigate to Anypoint Platform at http://anypoint.mulesoft.com and log in to your account.

   Note: If you do not have an Anypoint Platform account, create one now.

2. In Anypoint Platform, navigate to Anypoint Exchange.
3. In the upper right, make sure you select a business group with permission to see Anypoint Exchange assets.
4. In the upper left, click All assets.
5. Review the cards for the available assets.



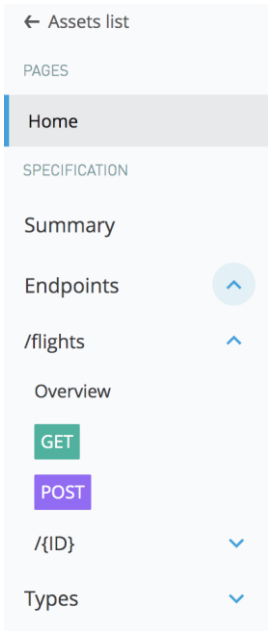
6. In the upper left, click the types drop-down menu and identify the asset types supported by Anypoint Exchange.



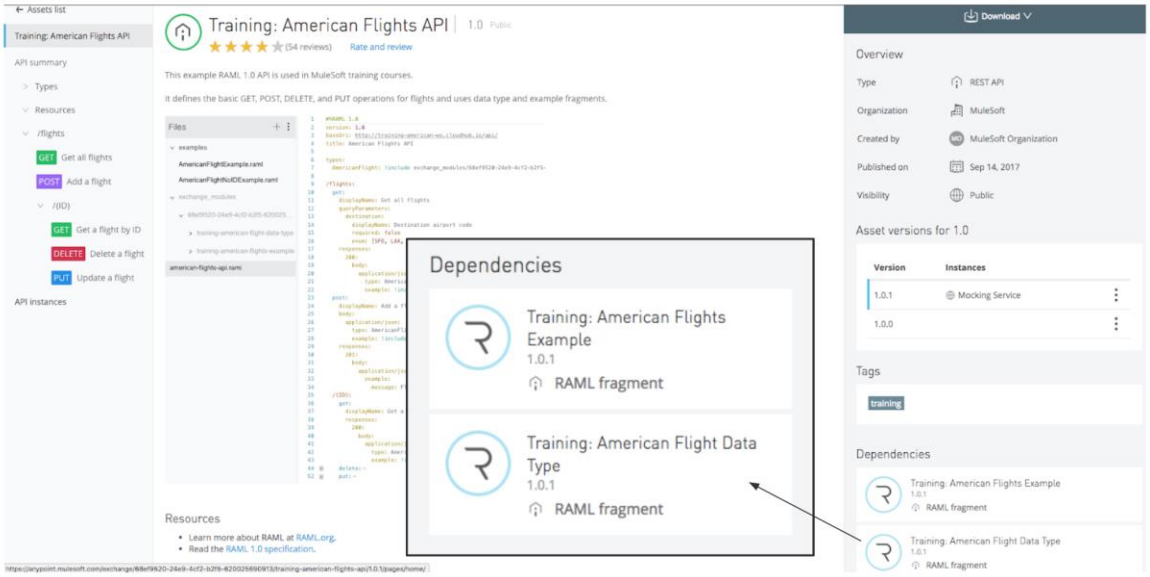
7. Search for the term American Flights; this should return assets used in the Development Fundamentals training course.

Identify how API resources and dependencies are stored in Anypoint Exchange

8. In the cards returned by the search, select the American Flights API card.
9. Look in the API summary section and observe the REST resources defined by this REST API.



10. Look in the Dependencies section and observe the RAML fragments used by this API specification.

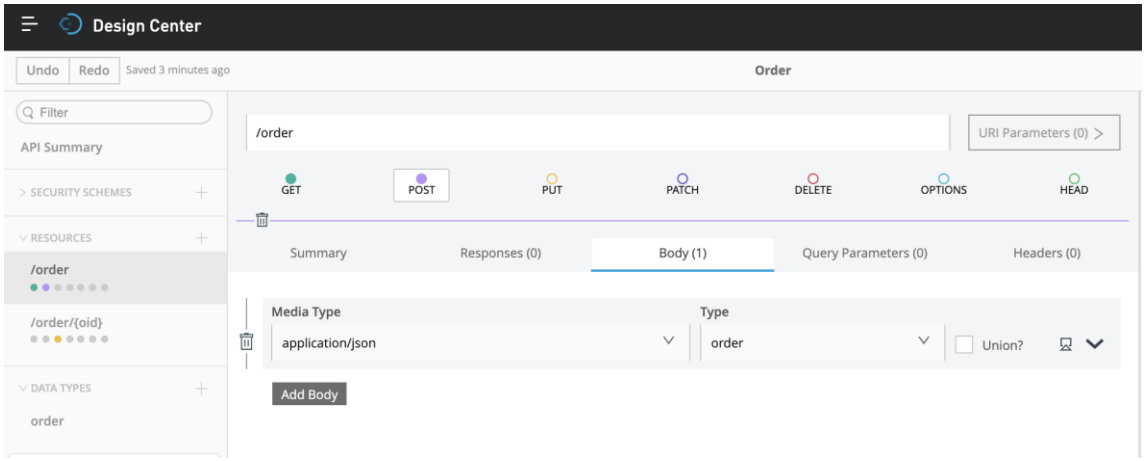


11. In the versions section, identify how different versions of this API are stored in Anypoint Exchange.

Walkthrough 2-2: Create a REST API using API Designer

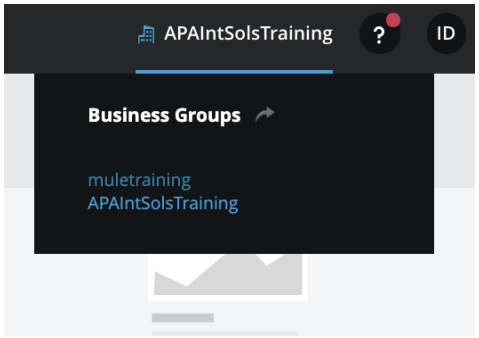
In this walkthrough, you use the visual editor in Anypoint API Designer to quickly define a new REST API specification with several nested parent-child relationships. You will define an Order object that contains an Item child that itself is composed of an array of LineItem objects. Many of the REST API calls will get or set Order objects. You will also create an Orders type that is an array of Order objects to return multiple Order objects in a GET request. You will:

- Define an API specification with several REST methods.
- Add a child URI parameter resource and its methods to the API specification.
- Define response status codes for each REST method.
- Specify which request and response parameters are required and which are optional.
- Define type and value constraints on request and response parameter values.

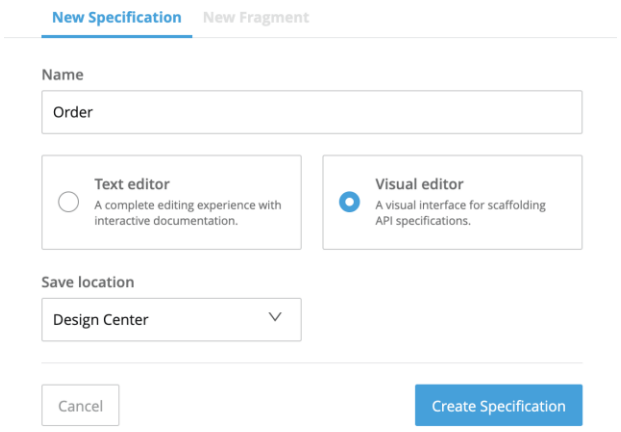


Create a new API design in API Designer

1. Return to Anypoint Platform and navigate to Design Center.
2. In the upper right, make sure to select a Business Group with access to Design Center.

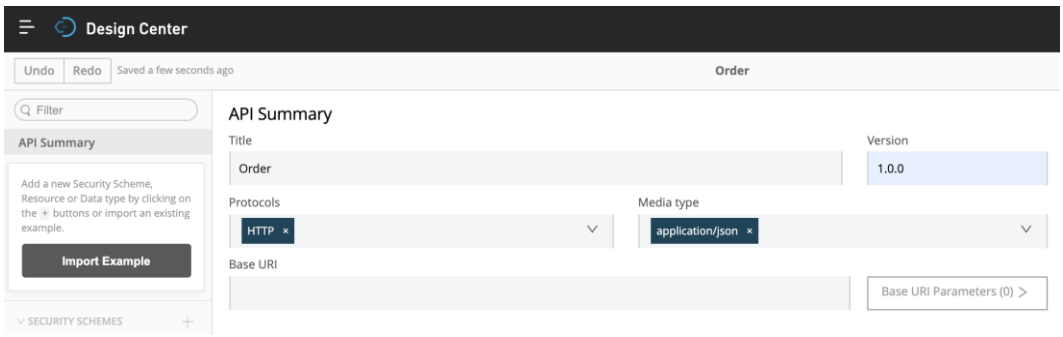


3. In the upper right, click Create new.
4. Click Create API specification.
5. In the New Specification tab, set the name to Order.
6. Click the Visual editor radio button.



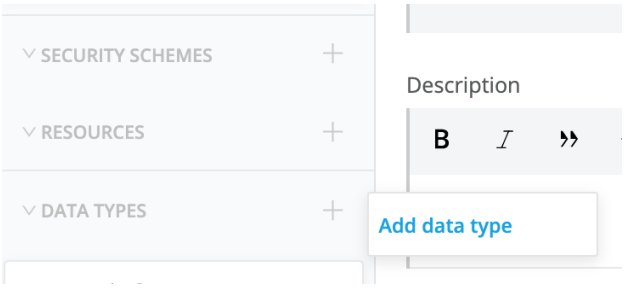
Note: The visual editor helps build REST API specifications with valid structures, such as indentation levels and correct YAML formatted key names. If you switch back to the code editor, you cannot go back to the visual editor.

7. Click Create Specification.
8. In the API Summary, set the Title to Order, the Version to 1.0.0, the Protocols to HTTP, and the Media type to application/json.

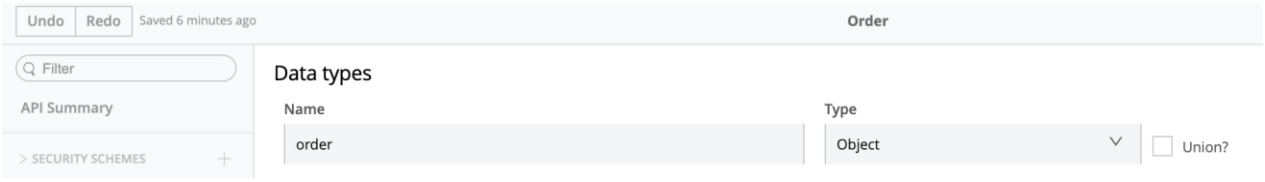


Define data types for the Order API specification

9. In the left-side navigation, click the + icon to the right of DATA TYPES and select Add data type.



10. Define a new data type named order and then click Add Property to add a new Parameter.

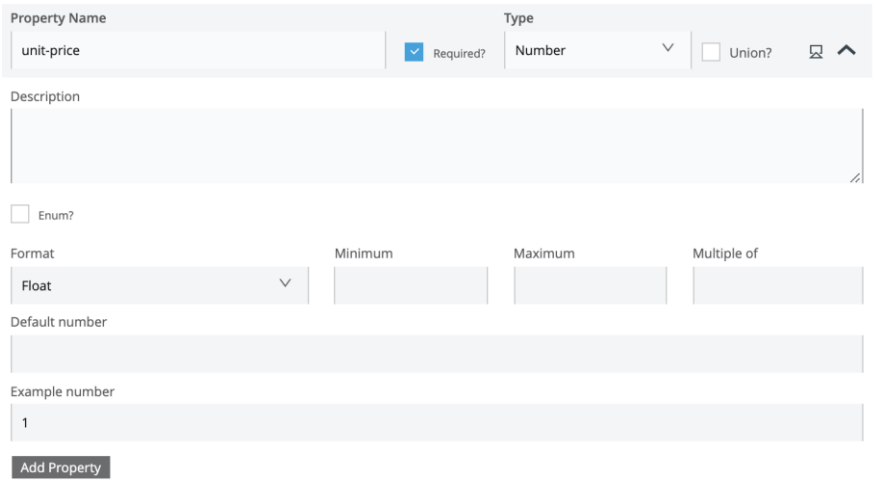


11. Add parameters to the order object according to the specification in the following table.

    Note: Edit the specification in this table with any changes you'd like to make, such as additional parameters, constraints, or enumeration of allowed values for a parameter.

Parameter	Type	Required	Constraint	Example
oid	string	Yes		
account-id	string	Yes		
itemName	string	Yes		
unit-price	float	Yes	>=0	
quantity	string	Yes		
total-price	float	Yes	>=0	

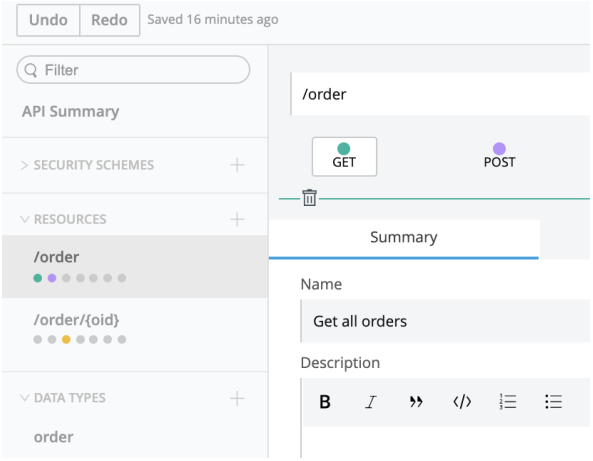
Note: To specify a float or integer type, set the Type to Number, then click the Details drop-down arrow icon to and set the Format drop-down list to Float or Integer. You can also configure constraints on allowed values in the Details area.



Define methods of the order resource

12. In the left-side navigation, click the + icon next to Resources and select Add resource.

13. Change the resource name to /order.



14. Click the Get method.
15. Define a GET method with the following response details:
    - Status: 200 - OK
    - Body media type: application/json
    - Body Type: order
16. In the upper methods list, click POST and define a POST method with the Body set to the following:
    - Media Type: application/json
    - Type: order

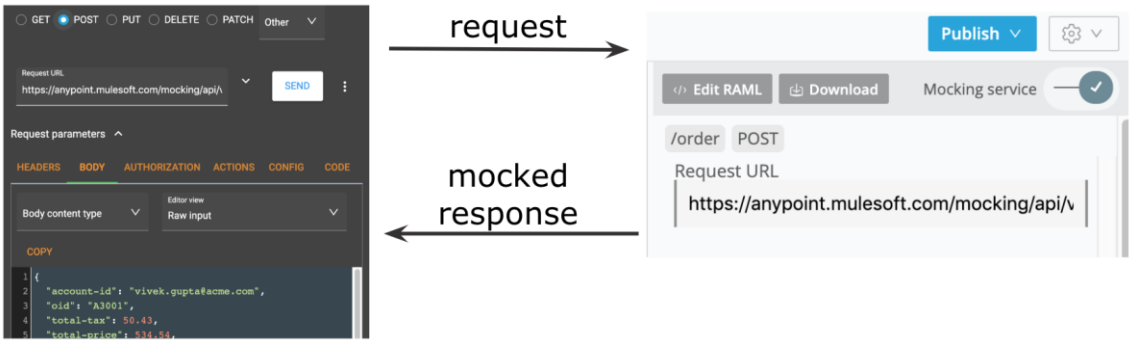
Define a nested resource under the order resource for individual order instances

17. In the left-side navigation, click the + icon next to /order, click Nested Resource, and then name the new child resource {oid}; this defines a URI parameter named oid under the /order resource.
18. With the /order/{oid} resource selected, add relevant methods to modify a specific order instance:
    - A PUT method with an application/json body set to type order; this is to completely replace an order instance.

Walkthrough 2-3: Test and mock a REST API using API Designer

In this walkthrough, you use the visual editor in Anypoint API Designer to test an API. You will:

- Try out a REST API in an API console.
- Mock a REST API specification so it can be tested by other REST clients.
- Use a mocking service to validate REST client requests against an API specification.

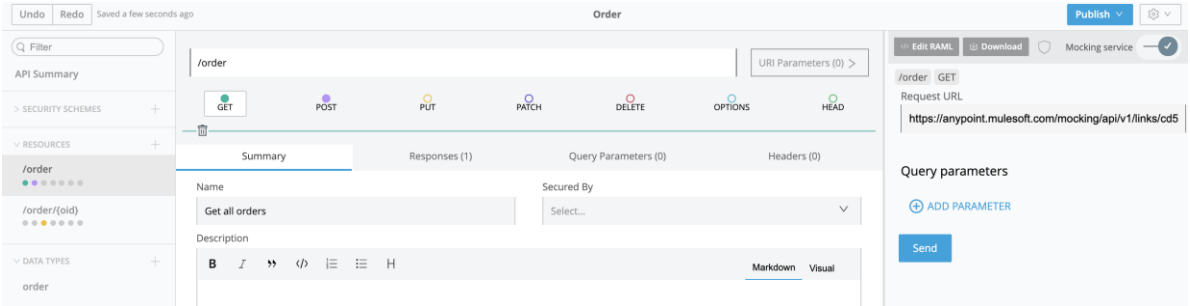


Mock the API so it can be tested by other REST clients

7. In the upper right, click Mocking service.
8. In the lower right, select the Try It tab.

Test the mocked API using the API Console embedded in API Designer

9. Select each of the methods and send requests to the mocking service; when an oid is required by a method, type in any string value.
10. Verify you get example responses as expected from each method call.
11. In the left-side navigation, select the /order resource.
12. Select the GET method.



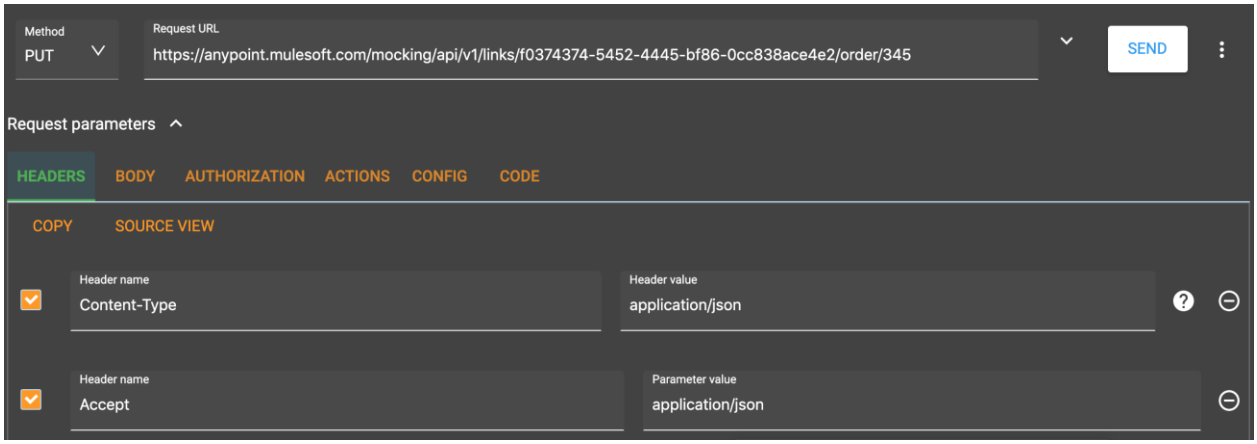
13. Copy the Request URL value to a REST client and submit a GET request; you should see the example Order array returned in the body of the response, and the status should be 200 - OK.

Test other API methods of the mocking service using a different REST client

8. In API Designer, select the /order/{oid} PUT method.
9. Type in a sample oid string, such as 345.
10. Copy the Request URL to a REST client.

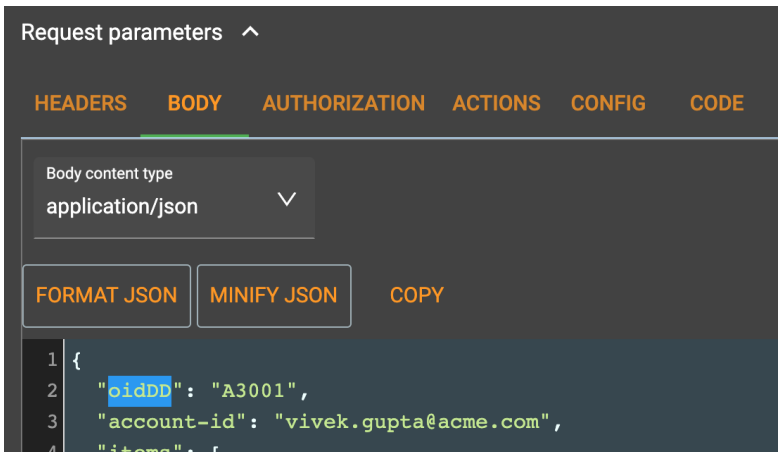
    Note: You could continue to use the API Console to test the REST API, but in this step you confirm that the API mocking service provides public URLs that can be used by non-Mule developers who do not have access to the API portal, or perhaps want to send REST requests in another programming language or tool.

11. In the REST client, configure a PUT request and paste the Request URL.
12. Copy the example Object body in API Designer and paste it in the body of the REST client request.
13. Configure the Body content type to application/json.
14. Configure an Accept header to application/json.



15. Submit the PUT request and verify a 200 - OK status is returned.

16. In the REST client's PUT request body, change the oid key to oidDD.



17. Submit the PUT request; a 400 Bad Request response should be returned because the request data validation failed.

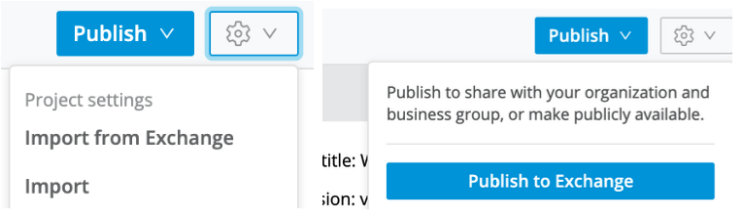
```
{
    "code": "REQUEST_VALIDATION_ERROR",
    "message": "Invalid schema for content type application/json.
    Errors: required key [oid] not found."
}
```

NOTE: This shows you how the mocking service allows REST clients to "fail fast" based on the business requirements of the API specification, before any API implementation is coded. Instead of a REST client, the mocking service can also be used by REST client's that are being coded into applications that will consume this API, allowing parallel development while APIs are being implemented.

Walkthrough 2-4: Share API specifications using Anypoint Exchange

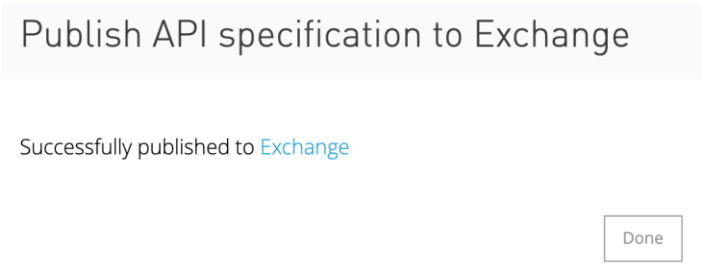
In this exercise, you publish APIs to Anypoint Exchange. You will:

- Publish an API specification defined in API Designer to Anypoint Exchange.
- Import an API into API Designer from Anypoint Exchange.



Publish the Order API to Anypoint Exchange

1. Return to API Designer and select the Order API.
2. In the upper right, in the Publish drop-down menu, select Publish to Exchange.
3. In the Publish API specification to Exchange dialog box, make sure the asset version is set to 1.0.0.
4. Click Publish.
5. Wait for the response message Successfully published to Exchange.



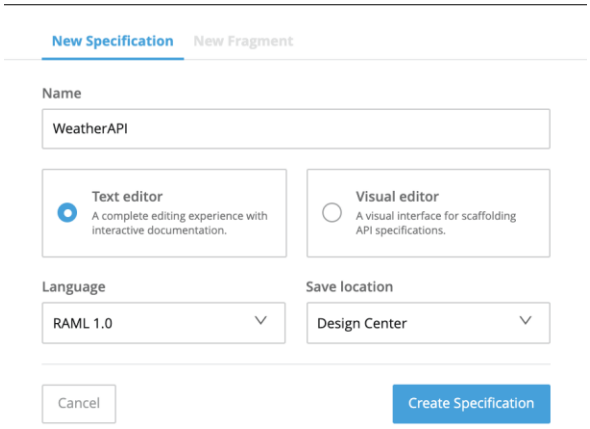
6. Click Done to close the Publish API specification to Exchange dialog box.

Import an API specification into API Designer from Anypoint Exchange

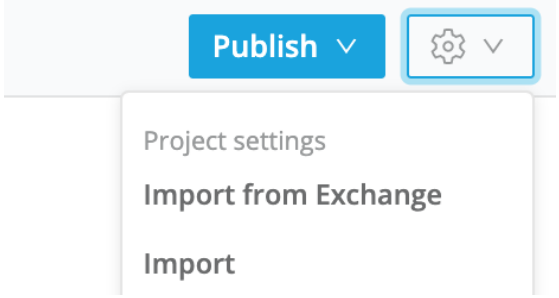
7. In the upper left, click the Design Center logo.

8. In the upper right click Create new.

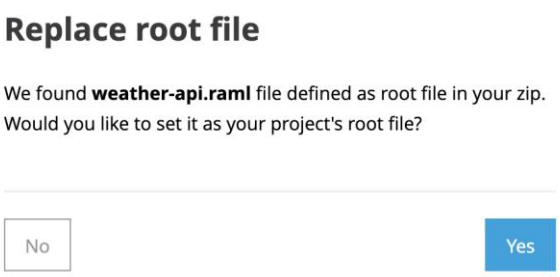
9. Select Create API specification and name the new API WeatherAPI.
10. Keep the default settings and click Create Specification.



11. In the upper right Project settings drop-down list, select Import.



12. Browse to the $APAINTSOL_HOME/studentFiles/mod02-ap-arch-components/starter_resources folder.
13. Select WeatherAPI.zip and then click Import.
14. Click Yes



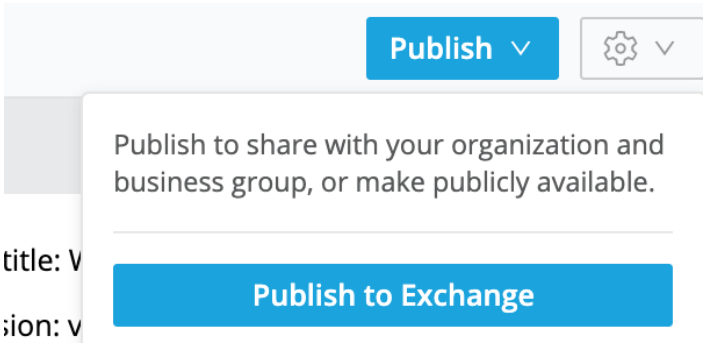
15. Review the weather-api.raml file and its associated dataTypes RAML fragment files.

    Note: You can also create your own API in Design Center, or import other APIs from Anypoint Exchange.

16. In Design Center, make some small changes to the API descriptions; this will help you verify in later steps that you are using this particular version of the API.

Publish to Exchange

17. In the upper right Publish drop-down list, select Publish to Exchange.

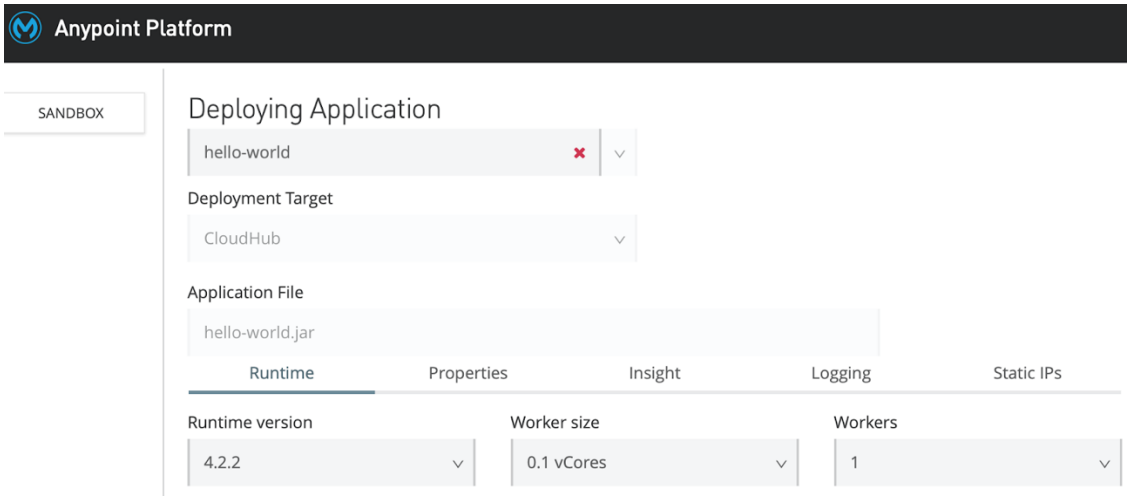


18. In the Publish API specification to Exchange dialog box, make sure asset version is set to 1.0.0.
19. Click Publish to Exchange.
20. Click View in Exchange and verify the API was successfully published to Anypoint Exchange.

Walkthrough 2-5: Deploy a Mule Application to the MuleSoft-hosted runtime plane

In this walkthrough, you deploy a Mule application to one or more CloudHub workers. You will:

- Configure and deploy a Mule application to Anypoint Runtime Manager from within Anypoint Studio.
- Manage a deployed Mule application from Runtime Manager.



Link Anypoint Studio with an Anypoint Platform account

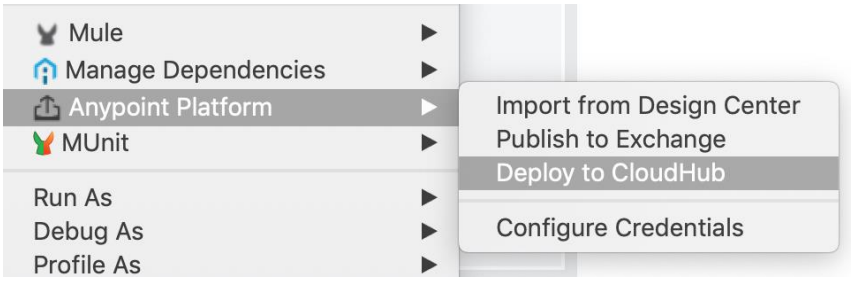
1. Open Anypoint Studio.
2. Import hello-world.jar from $APAINTSOL_HOME/studentFiles/mod02-ap-arch-components/starter_resources.
3. In the Project Explorer view, right-click the top-level hello-world project folder and select Anypoint Platform > Configure Credentials; the Preferences dialog box should open with Anypoint Studio > Authentication selected.

4. In the Preferences dialog box, in Anypoint Studio > Authentication, click Add.

5. In the User login dialog box, enter the username and password for a valid and unexpired Anypoint Platform account and then click Sign in.

   Note: Do not enter your company's official credentials.

6. In the Preferences dialog box, click Apply and Close.
7. In the Package Explorer view, right-click the upper hello-world project folder and select Anypoint Platform > Deploy to CloudHub; this opens a web page logged in to Anypoint Platform.



   Note: If you have trouble logging in to Anypoint Platform, you can open a separate browser window and log in to anypoint.mulesoft.com.

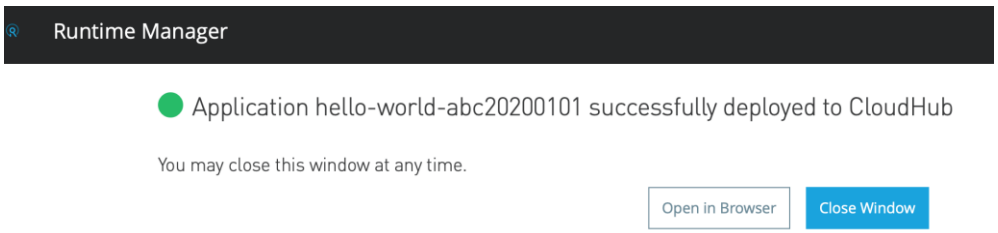
8. In the Anypoint Platform browser window, select an environment in which to deploy the Mule application; this should display an application deployment configuration page.

Configure the Mule application deployment with a globally unique name

9. Add a unique suffix to the end of the application deployment name; a good suffix is your initials and the date, such as hello-world-abc20201011.

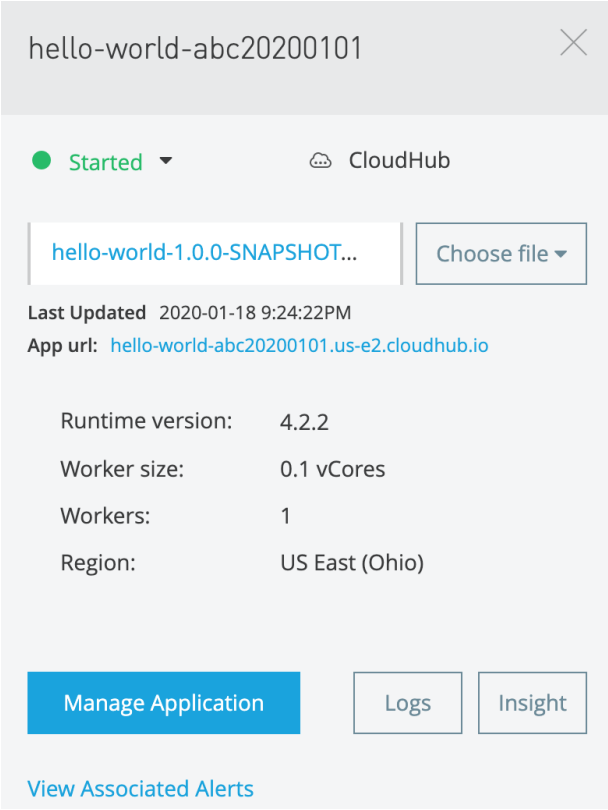
   Note: The application name must be globally unique among all Mule application deployments across every MuleSoft customer.

10. Look at all the configuration options in each tab of the deployment configuration page.
11. Click Deploy Application; Anypoint Studio should automatically package up your application (using maven) and then upload the Mule application's deployable archive JAR file to Runtime Manager using the Anypoint Platform REST APIs.
12. Wait for the hello-world Mule application to deploy successfully.



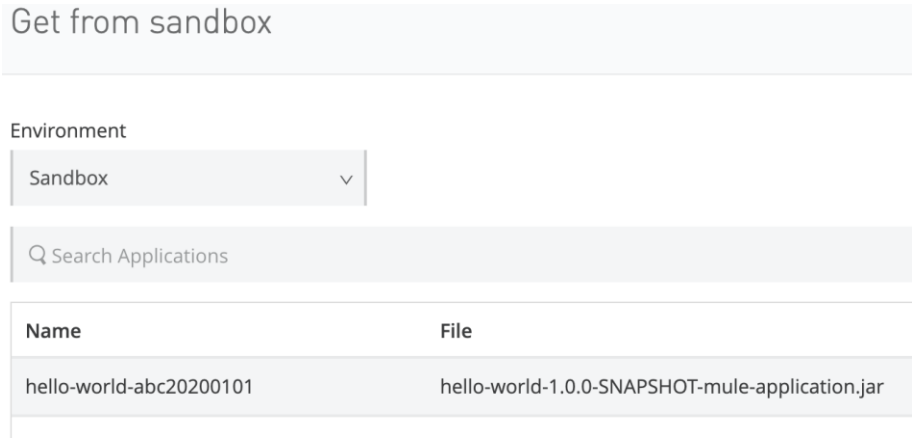
Review the Mule application deployment

13. Click Open in Browser to view the deployed Mule application in Runtime Manager.
14. If necessary, log in with the same credentials that were supplied to Anypoint Studio to configure the Anypoint Platform authorization.
15. Ensure the environment to which the hello-world Mule application was deployed is selected.
16. Verify the Mule application status is Started.
17. Select the Mule application then click Logs.



18. Read the logs and observe the sequence of steps a Mule application goes through when it starts.
19. In the left side, select Settings.
20. Click Get from sandbox.

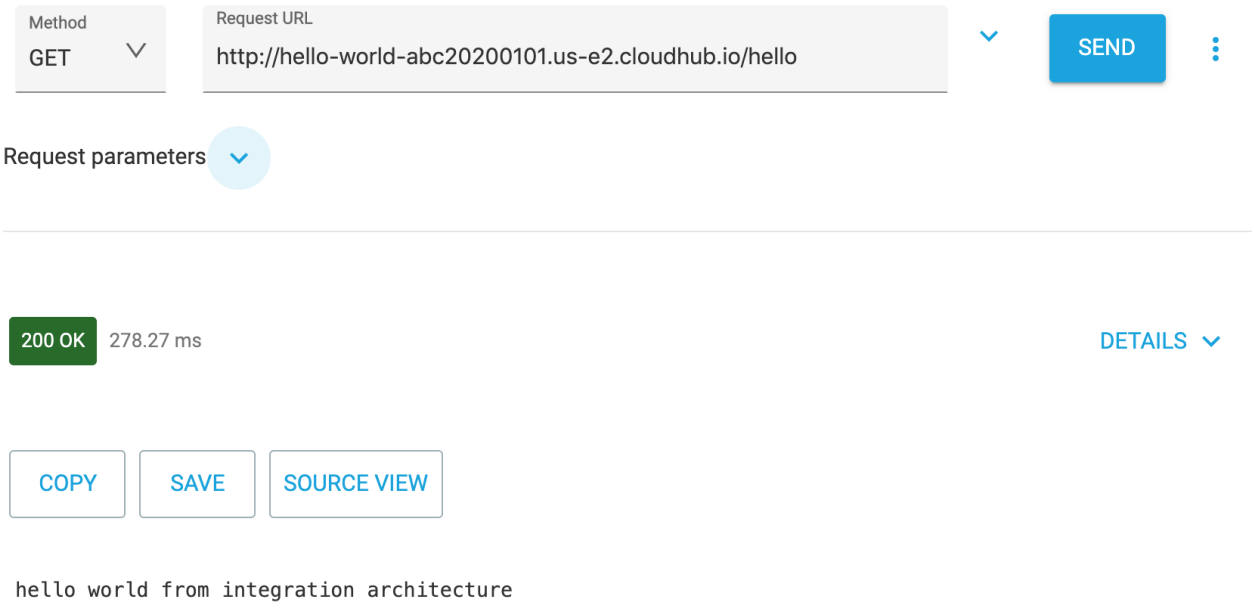
21. Verify the hello-world Mule application was deployed as a JAR file uploaded from Anypoint Studio to Runtime Manager.



22. Click Cancel.

Verify the hello-world application is accessible from your computer

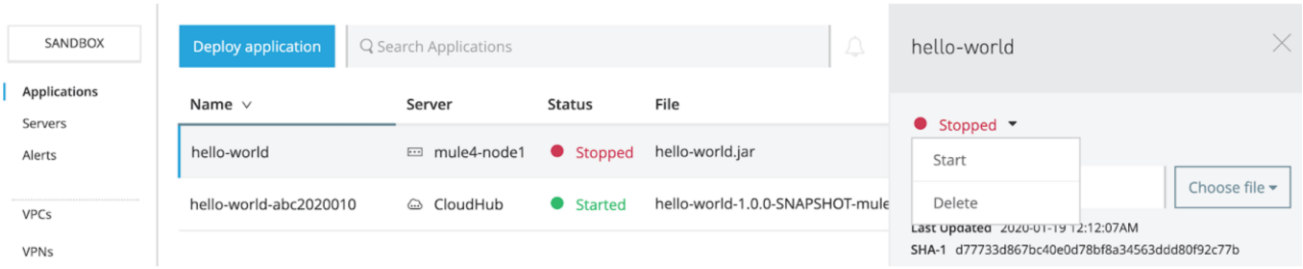
23. Copy the App url link to a REST client, then append /hello to the end of the URL.
24. Submit a GET request to the App url and verify the string result is returned in the successful response.



Walkthrough 2-6: Deploy a Mule application to the customer-hosted runtime plane

In this walkthrough, you download and install a stand-alone Mule runtime on your computer and then deploy the hello-world Mule application exported from Anypoint Studio to it. You will:

- Install and run a stand-alone Mule runtime on a local host.
- Register a stand-alone Mule runtime with Anypoint Runtime Manager using the Runtime Manager agent.
- Export a Mule application deployable archive from Anypoint Studio.
- Deploy a Mule application to a stand-alone Mule Runtime using Runtime Manager.



Prepare your computer to run a stand-alone Mule runtime

1. Open the task manager and kill any running Java processes.

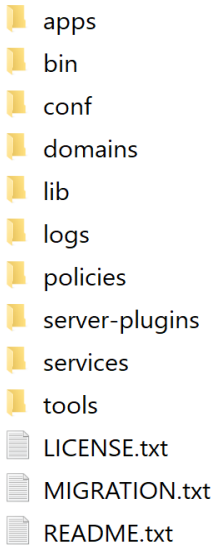
   Note: Anypoint Studio uses an embedded Mule runtime. Even after stopping the debugger, the embedded Mule runtime may continue to run and may still be binding to HTTP port 8081 from your previous walkthroughs.

Install and run a stand-alone Mule runtime on your computer

2. Locate the Mule 4 runtime zip file you downloaded at the beginning of the class.

14. Copy this zip file to your $APAINTSOL_HOME folder.
15. Unzip the Mule 4 runtime zip file, and rename the expanded folder to mule4-node1.

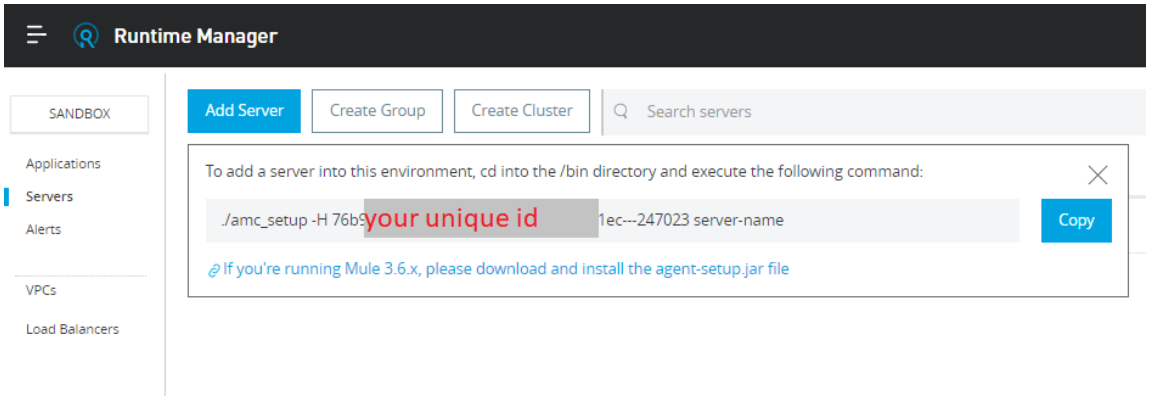
16. Review the structure of the Mule runtime.



17. Verify the apps folder is empty.

Register the stand-alone Mule runtime with Runtime Manager using the Runtime Manager agent

18. Return to Runtime Manager in your Anypoint Platform account.
19. On the left side, click Servers.
20. Click Add Server.
21. Click Copy to copy the server registration command.



Note: If you are installing on Windows, copy the registration command after the first two characters (./). The windows shell usually does not understand the ./ characters.

22. In the upper right of the dialog box, click X to close the dialog box.
23. Open a command-line interface and navigate to the $APAINTSOL_HOME/mule4-node1/bin.

24. Paste the Mule runtime registration command you copied from Runtime Manager into the command-line interface.

    Note: On Windows, replace the start of the command ./amc_setup with amc_setup.

25. Replace the server name server-name at the end of the Mule runtime registration command with mule4-node1.
26. Enter the command and wait for the registration process to complete successfully.

```
Communication between Anypoint Management Center and the Mule Agent is authorize
d by using two way SSL verification. In the following step we are
going to create a keystore and request Anypoint Management Center to sign it. Yo
ur private key will never leave your machine.


Sending Sign request to Anypoint Management Center

Mule Agent configured successfully
```

    Note: This unpacks the Runtime Manager agent plugin, generates a new TLS configuration for a secure web socket, and reaches out to the Anypoint Management Center to register itself with Runtime Manager.

    Note: If you have trouble with this step, for example if you are behind a corporate proxy, you can specify the proxy host/port in the amc_setup command. Run amc_setup –help to see the list of options. If your proxy interferes with the secure web socket (e.g. some corporate proxies inject their own TLS after terminating the request in order to spy on HTTPS traffic) then you will not be able to continue this walkthrough.

Manage the Mule runtime from Runtime Manager

27. On Windows, type mule.bat, or on a Unix machine, type ./mule.
28. Verify the Mule runtime starts without errors.

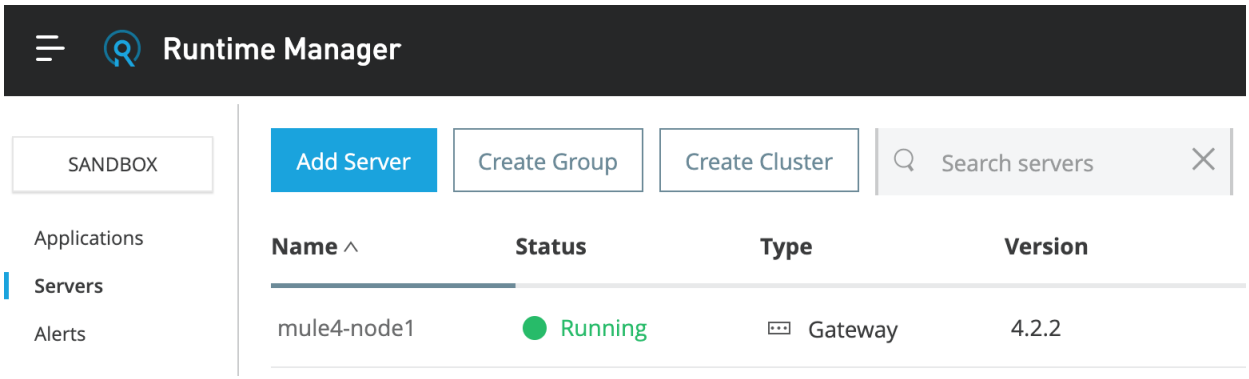
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                                          +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**********************************************************************
*               - - + DOMAIN + - -               * - - + STATUS + - - *
**********************************************************************
* default                                        * DEPLOYED           *
**********************************************************************
```

    Note: If you see errors, verify the JDK 8 is installed correctly on your computer.

29. Verify you see console messages about the Mule runtime using the Mule Agent to connect to Runtime Manager over a web socket.
30. Return to Runtime Manager in your Anypoint Platform account.

31. In the Servers section, verify you see the mule4-node1 server status is Running.



Deploy the Mule runtime using Runtime Manager

32. Return to the Runtime Manager window.
33. Select Applications then click Deploy Application.
34. Set the application name to hello-world.

    Note: Unlike the CloudHub deployment, deployments to a stand-alone Mule runtime do not need to be globally unique between all MuleSoft customer deployments.

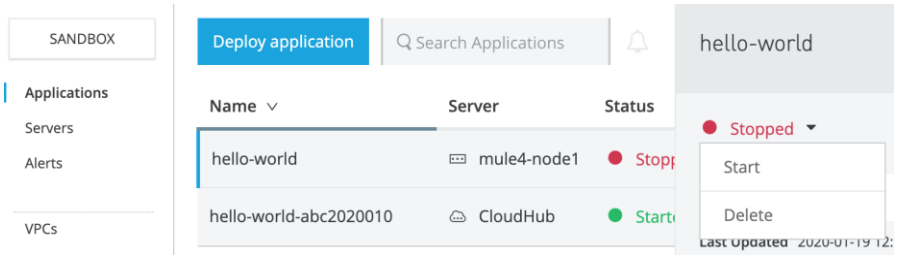
35. Set the deployment target to mule4-node1.
36. In the application file section, click Choose file, then Upload file.
37. Browse to $APAINTSOL_HOME/studentFiles/mod02-ap-arch-components/starter_resources and select the hello-world.jar deployable archive file.
38. Click each deployment option tab; notice you can set properties and logging options.

    Note: Unlike a CloudHub deployment, you cannot set the vCore size or number of workers.

39. Click Deploy Application.
40. Wait for the hello-world application status to change to Started.

Manage and validate the local Mule application deployment

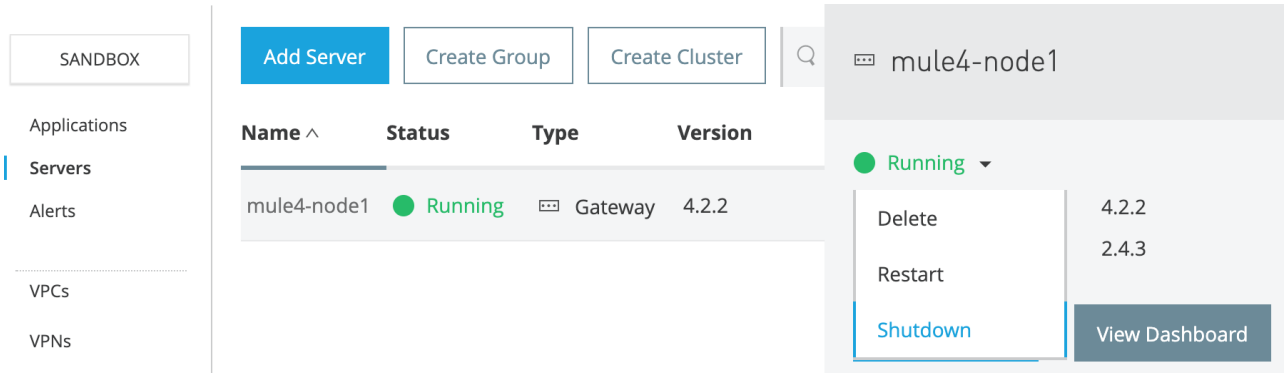
41. Select the hello-world application row (not the hello-world name link).
42. Click Started, then in the drop-down list select Stop; the status should change to Stopped.
43. After the status changes to Stopped, click Stopped and from the drop-down list select Start; the status should change to Started.



44. Return to the command-line terminal and verify the Mule 4 runtime logs messages about the hello-world Mule application stopping and starting.
45. Return to your REST client.
46. Submit a GET request to http://localhost:8081/hello.
47. Verify a 200 OK status is returned with the string response.

Manage the Mule runtime using Runtime Manager

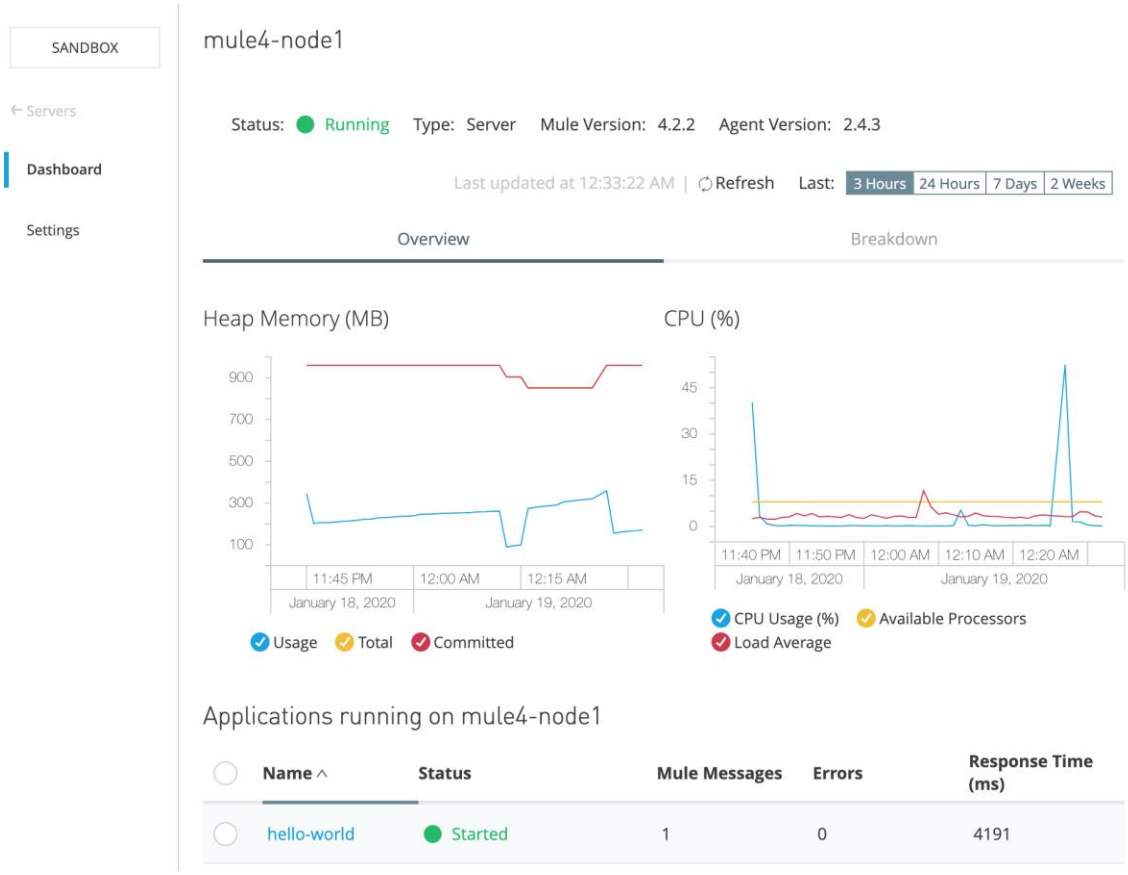
48. In Runtime Manager, navigate to the Servers section and select the mule4-node1 server row and then click Running.
49. In the drop-down list, select Shutdown and then confirm the shutdown in the dialog box; the status should change to Disconnected.



Note: Once a stand-alone Mule runtime is stopped, the Runtime Manager agent plugin also stops running, so there is no longer a connection between Runtime Manager and the stand-alone Mule runtime. The Mule runtime must be manually started on the Mule runtime's host.

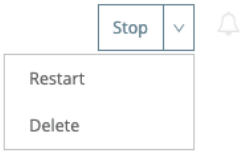
50. Return to the command-line interface.
51. On Windows type mule.bat, otherwise type ./mule.
52. Verify the Mule runtime starts up again successfully and automatically deploys the hello-world Mule application.
53. Return to Runtime Manager and verify the mule4-node1 server status is Running and the hello-world Mule application status is Started.
54. Select Servers and click the mule4-node1 link.

55. Verify you see monitoring data for the stand-alone Mule runtime.



Stop the Mule runtime

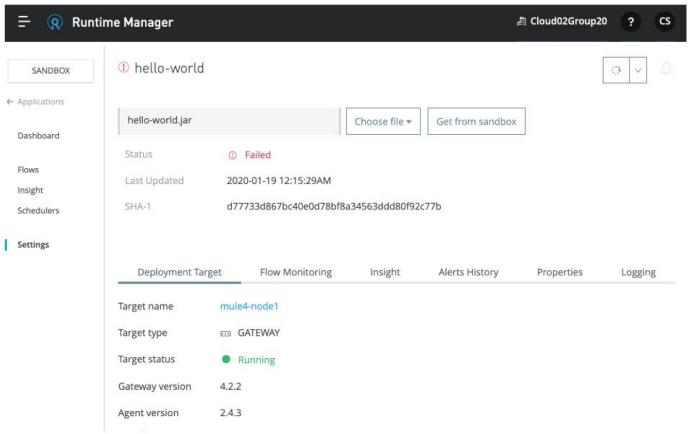
56. In the left-side navigation, select Settings.
57. In the upper-right drop-down list, select Stop.

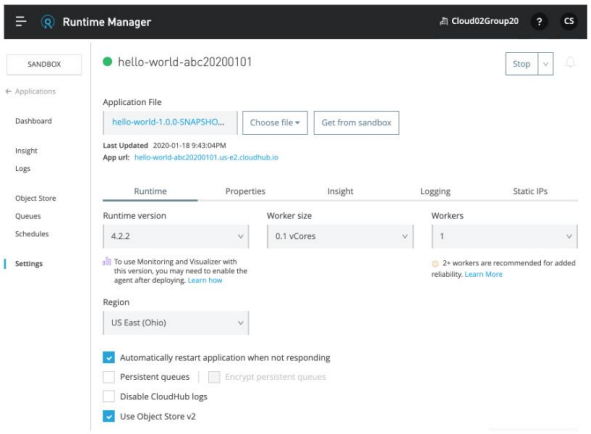


Exercise 2-7: Compare Mule application deployment options for various deployment targets

In this exercise, you compare and contrast deployment options for a Mule application deployed to CloudHub workers vs. stand-alone Mule runtimes. You will:

- List similar options when deploying to any type of deployment target.
- Compare options for CloudHub-specific vs. customer-hosted-specific deployment target.





List common deployment options for CloudHub vs. stand-alone Mule runtime deployment targets

1. Open two browser windows next to each other.
2. In one window, navigate to the Settings for the hello-world Mule application deployment to the mule4-node1 Mule runtime and then select the Properties tab.
3. In the other browser window, navigate to the Settings for the hello-world Mule application deployment to CloudHub and then select the Properties tab.

List options for each deployment target

4. Compare the hello-world application deployments in both deployment targets and fill in the table with differences and similarities.

Deployment Target Runtime Manager tab	Features	CloudHub	Stand-alone Mule runtime	Differences & similarities
Runtime	Runtime version, worker size, workers, region, persistent queues, auto-restart applications when not responding, disable CloudHub logs, Use Object Store v2	x		both have an application file, that can be uploaded or imported from Exchange. Both can import an already deployed Mule application deployable archive, but CloudHub deployments can only access Mule applications deployed to CloudHub, and all the stand-alone Mule runtime servers can only access Mule applications already deployed to a stand-alone Mule runtime server.
Properties			x	
Properties		x		
Logging			x	
Logging		x		
Insight			x	
Insight		x		
Alert History			x	
Flow Monitoring			x	
Static IP		x		

5. What features are common to all deployment targets?

6. What features are specific to CloudHub deployments?

7. What additional infrastructure or features are not available to stand-alone Mule runtime targets?

Answer these reflection questions

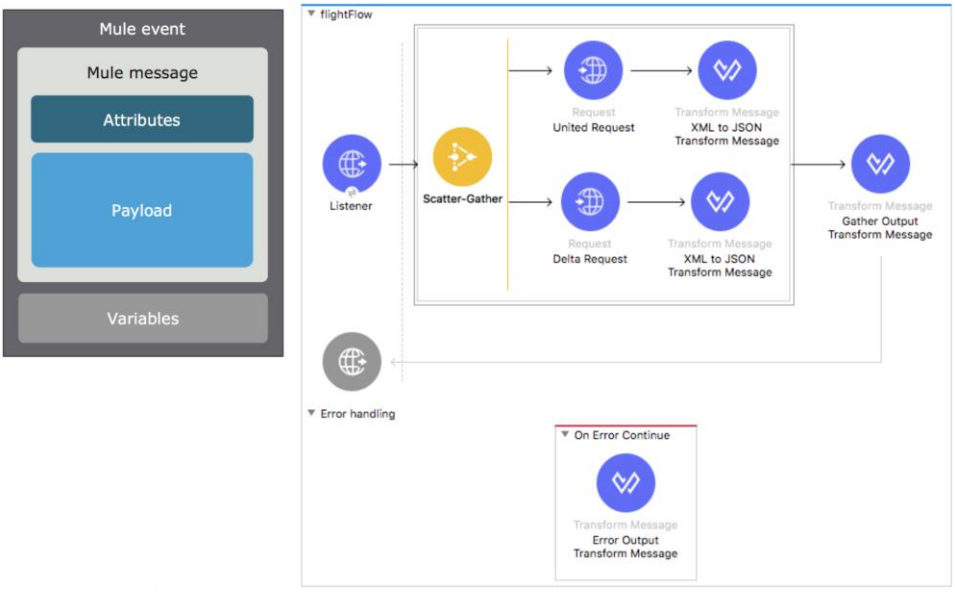
8. What features are specific to CloudHub deployments?

9. What features are you responsible for if you run a stand-alone Mule runtime?

10. Does a Mule application continue to run even if it is not connected to Runtime Manager?

11. What happens to the deployed Mule applications if the Mule runtime is restarted while it is disconnected from Runtime Manager?

Module 3: Designing Integration Solutions using Mule Applications



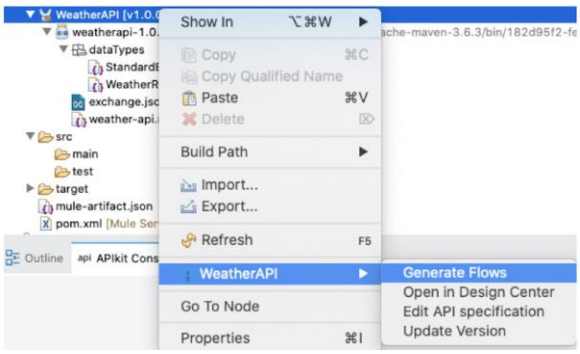
At the end of this module, you should be able to:

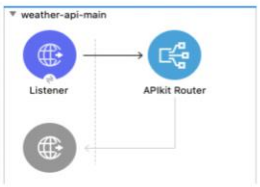
- Identify how Mule application components (including connectors, transformers, routers, and error handlers) are typically used to design integration solutions.
- Apply Mule application components to an integration use case.
- Summarize how class loading isolation is implemented in Mule runtimes.

Walkthrough 3-1: Implement APIs using APIkit

In this walkthrough, you carry out API-led development using APIkit. You will:

- Import REST APIs into Anypoint Studio from Anypoint Design Center.
- Auto-generate scaffolding flows in a Mule project from REST APIs.
- Test API implementations using an auto-generated API Console process.
- Test API implementations using an external REST client.



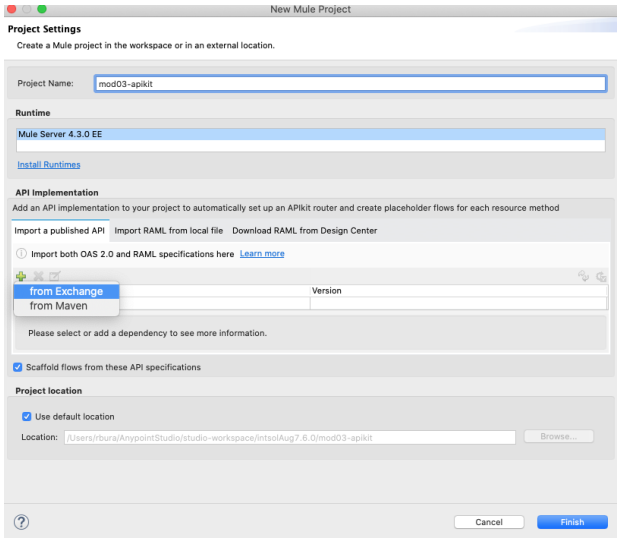


Make sure you have stopped your local Mule runtime

1. Navigate to the command-line interface for the mule4-node1 Mule runtime.
2. If the Mule runtime is running, type Ctrl+C to stop it.

Import API definition files from Design Center into Anypoint Studio

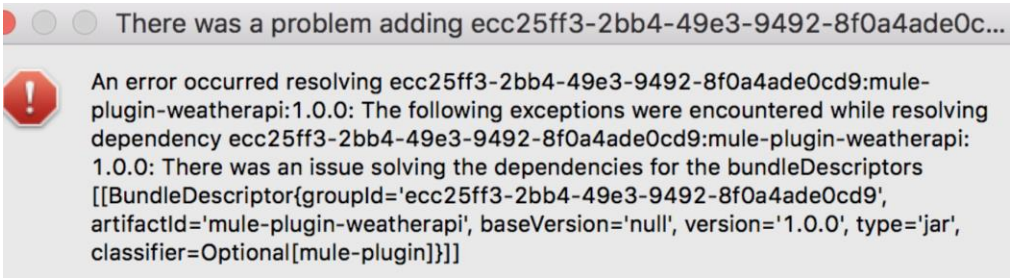
3. Return to Anypoint Studio and create a new project named mod03-apikit.



4. In the API Implementation section, Click the Add icon and select the from Exchange

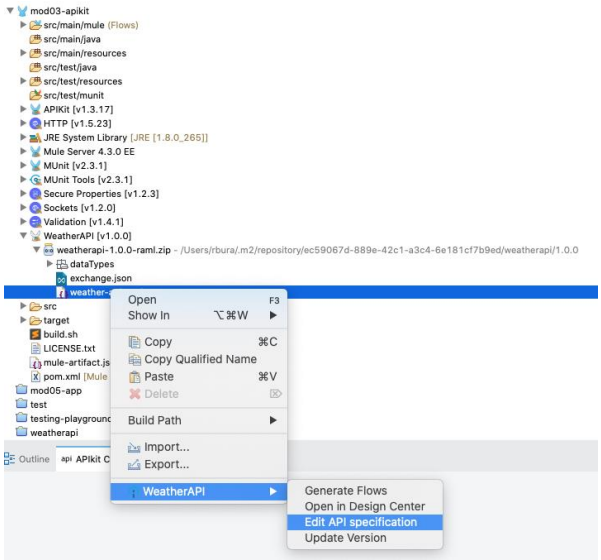
5. Search for the WeatherAPI and click Add.
6. Click Finish.
7. Click Apply and Close.

   Note: If you see a problem dialog at this point, look on your computer for a folder named $HOME/.m2. If this folder is there, shut down Anypoint Studio, rename the .m2 folder to .m2-bak, then restart Anypoint Studio. You can move .m2-bak back to .m2 after you finish this class.



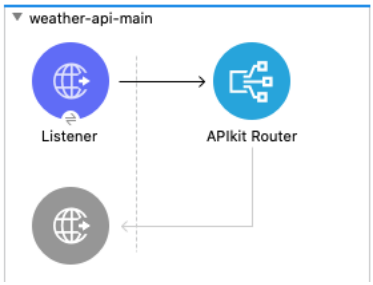
8. Wait for the Progress information dialog box to close.
9. Expand the weatherapi-1.0.0-raml.zip and dataTypes folders and then open and review the files.

10. Imported API Specification can be edited in a new API project; Right-click weather-api.raml in the Package Explorer and select WeatherAPI > Edit API specification.



Note: This opens up a new API project which can be edited and synched/published to Design center/Exchange

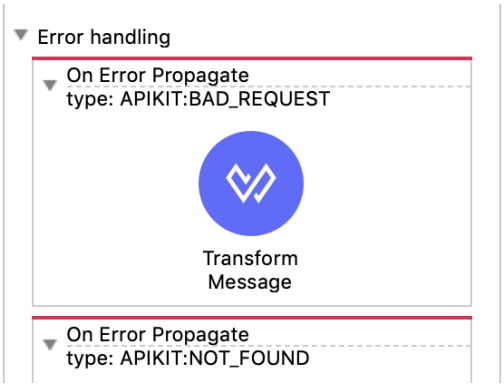
11. In the src/main/mule folder, open the weather-api.xml file; this file was automatically generated from the imported weather-api.raml file.
12. Verify an HTTP Listener and APIkit Router were added to the weather-api-main flow.



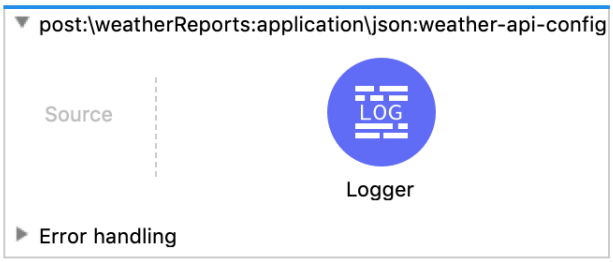
13. Select the Listener and verify the path is set to /api/*.

    Note: This indicates requests to this REST endpoint must start with the /api suffix before accessing resources defined in the Weather API.

14. Scroll down and review the error handlers that were automatically added to the weather-api-main flow.

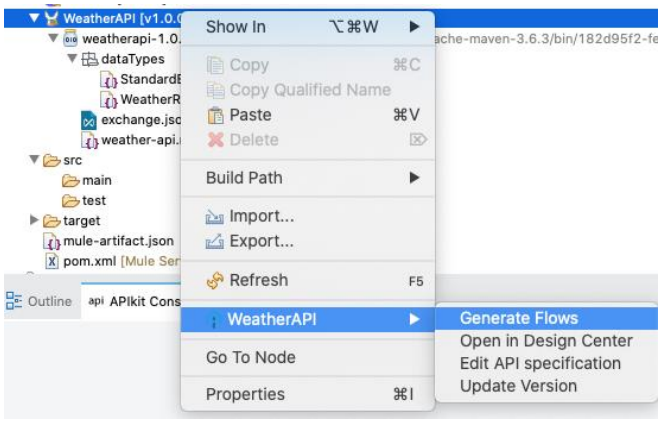


15. Scroll to the bottom and verify a post:\weatherReports flow was added to handle POST requests from the APIkit Router; this is an empty scaffolding flow (also called a skeleton flow) you can use to implement the behavior of this API operation.



Regenerate the APIkit scaffolding flows from the API specification

16. Delete the src/main/mule/weather-api.xml file; this deletes the APIkit scaffolding flows from the Mule project that were auto-generated when the API was imported from Anypoint Exchange.
17. Right-click WeatherAPI and select WeatherAPI > Generate Flows.



18. Verify the weather-api.xml file is recreated with the same flows and error handlers.

    Note: Instead of importing an API specification from Exchange, you can create the specification in Anypoint Studio, or you can import API files as a zip archive from your local computer.

Run the Weather API skeleton implementation

19. Right-click the canvas and select Run project mod03-apikit.
20. Wait for the project to deploy without error.
21. Open a web browser and navigate to http://localhost:8081/console.
22. Under the /weatherReports resource click Post.
23. In the upper right, verify the Request URL is set to http://localhost:8081/api/weatherReports; otherwise enter this URL now.
24. Just below the Request URL, select Body.

25. Copy the POST request's JSON body to your snippets file.

```
{
    "postalOrZipCode": "2000",
    "locationName": "Sydney",
    "dateTimeOfReport": "2019-12-09T22:29:54.396Z",
    "rainfallInLastHour": 12,
    "fireDangerLevel": "high",
    "humidity": ""
}
```

26. Click SEND.
27. Verify a 201 Created status is returned.

    Note: This response is returned from the mod03-apikit Mule application that is running in Anypoint Studio. The API console is also running in this Mule application.

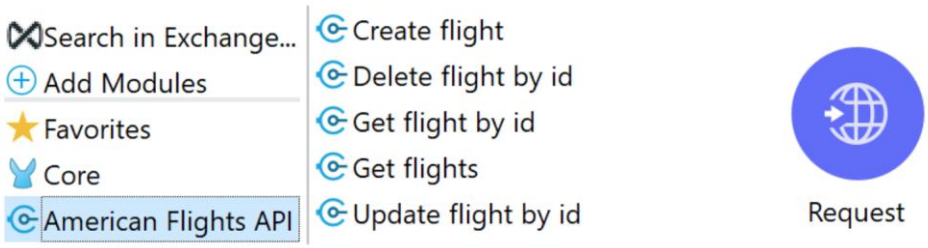
Answer these reflection questions

28. How much data validation logic did you write in this API implementation?
29. How much error handling logic did you have to write/add to this API implementation?

Walkthrough 3-2: Call a REST API from a Mule application using a REST Connect connector

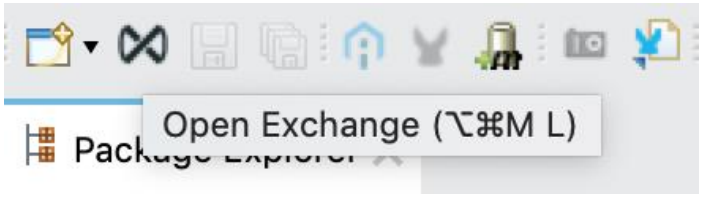
In this walkthrough, you carry out API-led development using APIkit and auto-generated REST Connect API connectors. You will:

- Load a REST Connect connector from Anypoint Exchange into a Mule application.
- Develop a Mule flow that uses a REST Connect connector.
- Use a REST Connect connector to consume from a corresponding REST API implementation.
- Identify ways that REST Connect connectors speed up REST API client development.

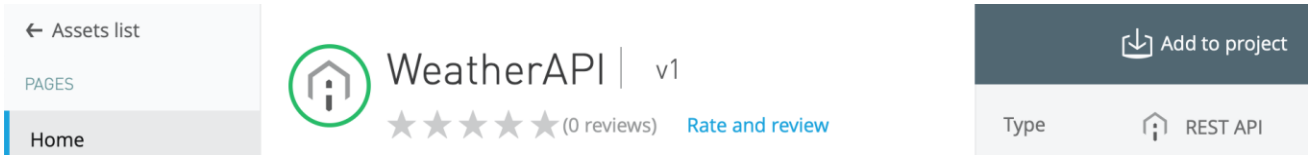


Import the Weather API's Anypoint REST Connector into the Mule project

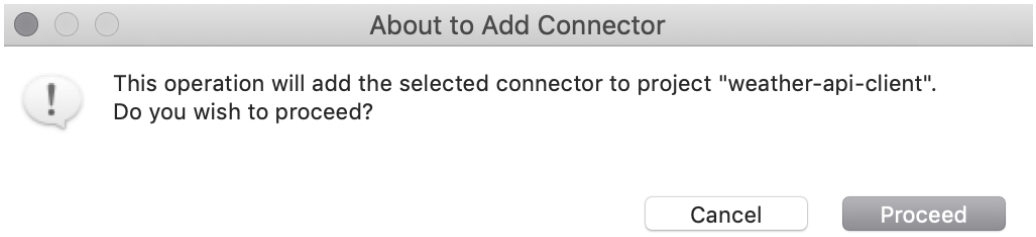
1. In Anypoint Studio, create a new Mule project named weather-api-client.
2. In the main menu, click the Open Exchange icon; a browser window should open to your private Anypoint Exchange.



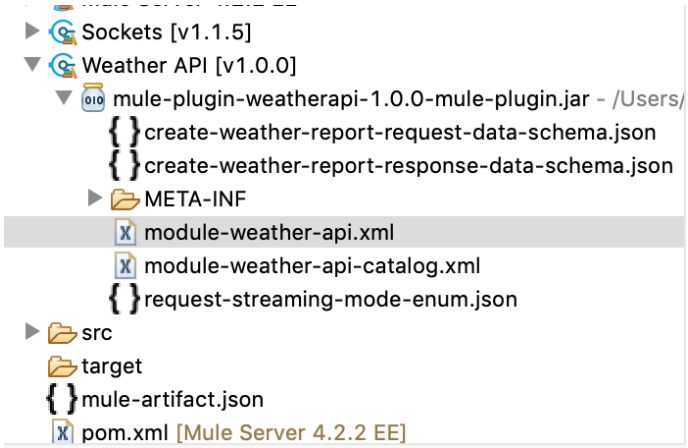
3. Search for your WeatherAPI card, then click the WeatherAPI card; the WeatherAPI page should load.
4. In the upper right of the Weather API page, click Add to project.



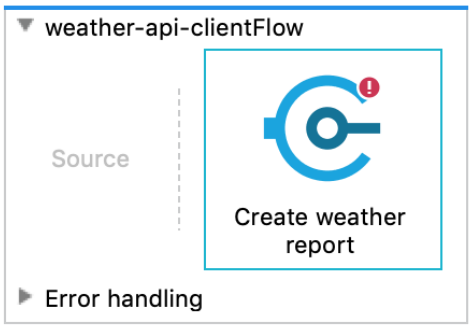
5. In the About to Add Connector dialog box, verify the operation will add the WeatherAPI connector to the weather-api-client Mule project and then click Proceed.



6. In the bottom of the weather-api-client Mule project, verify the Weather API REST connector was added as a plugin.



7. Open the src/main/mule/weather-api-client.xml file.
8. In the Mule Palette, select the Weather API module and then drag out a Create weather report operation and drop it in the canvas.



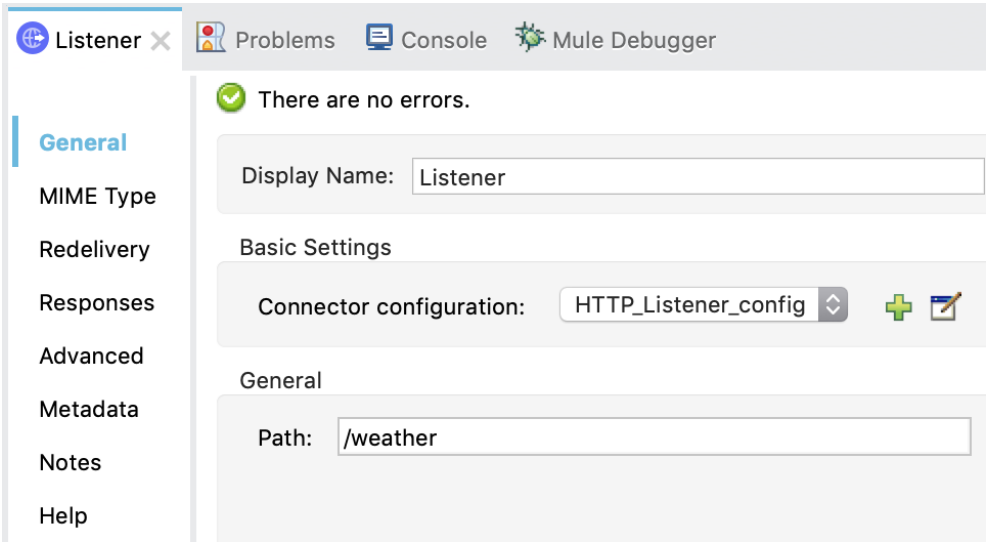
9. Select the Create weather report component.

10. In the Create weather report request data text area, replace the string payload with the POST request JSON you copied from the API Console.

    Note: The API only requires the postalOrZipCode and dateTimeOfReport keys.

```
{
   "postalOrZipCode": "2000",
   "dateTimeOfReport": "2019-12-09T22:29:54.396Z"
}
```

11. In the Create weather report configuration view, add a new connector configuration with the following values:
    - host: localhost
    - port: 8081
    - protocol: HTTP
    - basePath: /api/
12. Click Test Connection; a Test connection successful message should appear in the Test connection dialog box.
13. Add an HTTP Listener to weather-api-clientflow and configure it to listen on port 8082 and path /weather.

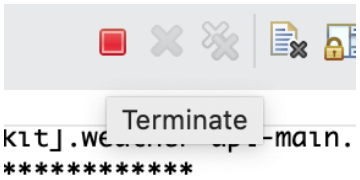


14. Add a Set Payload transformer to the end of the flow and change the value to return a status message plus the request payload.

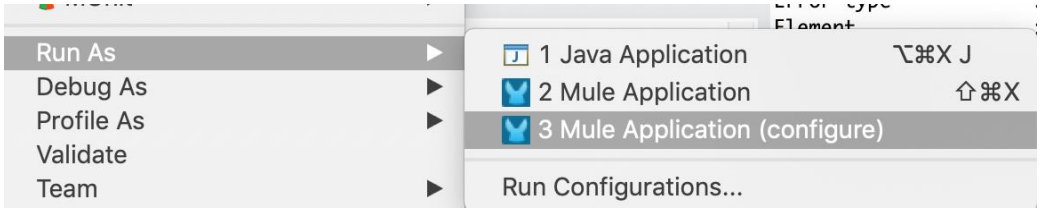
```
{
  result: "Weather report submitted",
  request: payload
}
```

Test the REST Connect client with the Weather API implementation

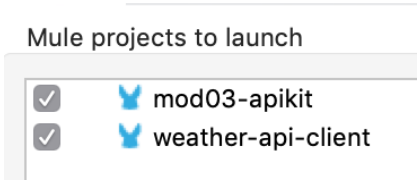
15. In the Console view, click the Terminate icon.



16. Right-click the weather-api-client folder and select Run As > Mule Appplication (configure).



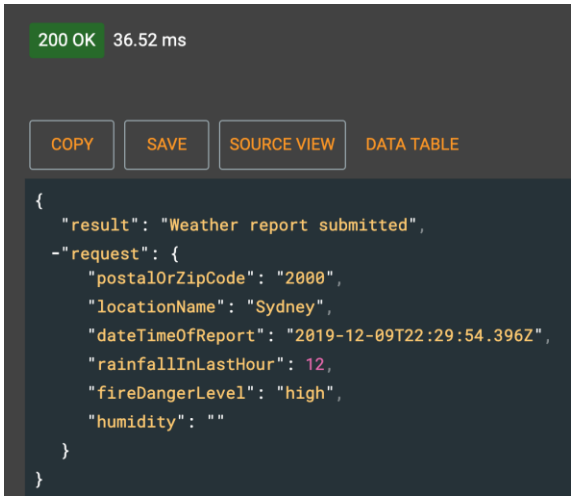
17. Check both the mod03-apikit and weather-api-client Mule projects.



18. Click Run and wait for both Mule applications to deploy.

```
**************************************************************************************************
*            - - + APPLICATION + - -            *       - - + DOMAIN + - -       * - - + STATUS + - - *
**************************************************************************************************
* mod03-apikit                                  * default                        * DEPLOYED           *
* weather-api-client                            * default                        * DEPLOYED           *
**************************************************************************************************
```

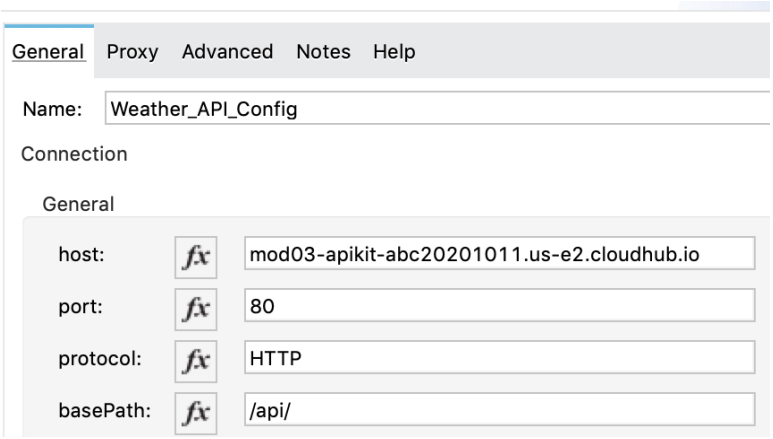
19. In a web client, send a GET request to http://localhost:8082/weather; you should get a 200 OK response with the response message.



Deploy the weather API implementation to CloudHub and test with your weather-api-client Mule application

20. Return to Anypoint Studio.
21. Right-click the mod03-apikit Mule project folder and select Anypoint Platform > Deploy to CloudHub.
22. Select a Runtime Manager environment.
23. Append your initials and the date to the deployment name.
24. Click Deploy Application.
25. Log in to Runtime Manager and select the mod03-apikit deployment.
26. Copy the App url.
27. Return to Anypoint Studio.
28. In the weather-api-client Mule project, select the weather-api-client.xml tab, then select the Global Elements tab.

29. Edit the Weather API Config with the CloudHub App url, and change the port to 80.



Note: Do not include http:// in the host name, and make sure HTTP is all uppercase.

30. Click Test Connection and verify the CloudHub endpoint can be accessed.

    Note: If you are behind a corporate firewall or proxy, you may not be able to complete this part of the walkthrough.

31. Click OK and save your changes.
32. Return to the REST client and submit another GET request to http://localhost:8082/weather.
33. Verify a 200 OK status is returned with the same JSON response.
34. Look in the Anypoint Studio Console view and verify the mod03-apikit Mule application did not log any messages; this is because it is no longer used by the weather-api-client's Weather API connector.

    Note: The Weather API connector is acting as a proxy to the Weather API implementation that is now deployed to CloudHub.

Compare the same code using an HTTP Request operation

35. Add another flow to the weather-api-client and construct the same POST request using an HTTP Request operation.
36. From the Weather API plugin folder, open the create-weather-report-request-data-schema.json file.
37. Verify the schema only requires the postalOrZipCode and dateTimeOfReport properties.
38. Verify the constraints on these two required properties; postalOrZipCode must be a string of length at least 2 characters, and dateTimeOfReport must be a string in date-time format.
39. Open the module-weather-api.xml file.
40. Look at all the error mappings that are automatically created.

41. Look where the headers and response payload are specified as application/json.
42. Add the JSON module to the Mule project.
43. From the JSON module, drag out a Validate schema component and add it to a new flow.
44. Verify there is no JSON schema in the Mule project to be used.

    Note: You can copy the text from create-weather-report-request-data-schema.json to a new file in src/main/resources and then the schema is available to configure the JSON Validate schema component.

Answer these reflection questions

45. What are some benefits of using Connectors over writing Java code to talk to a system?

46. Why would a REST Connect connector be more desirable compared to using an HTTP request operation?

47. Considering APIkit:

    - How do you extend the skeleton?

    - What role can the API Console play in testing?

48. Considering the REST Connect connector:

    - Why would you use an HTTP Request over the REST Connector?

    - What security schemes are supported?

    - How does the REST Connector support custom TLS configurations?

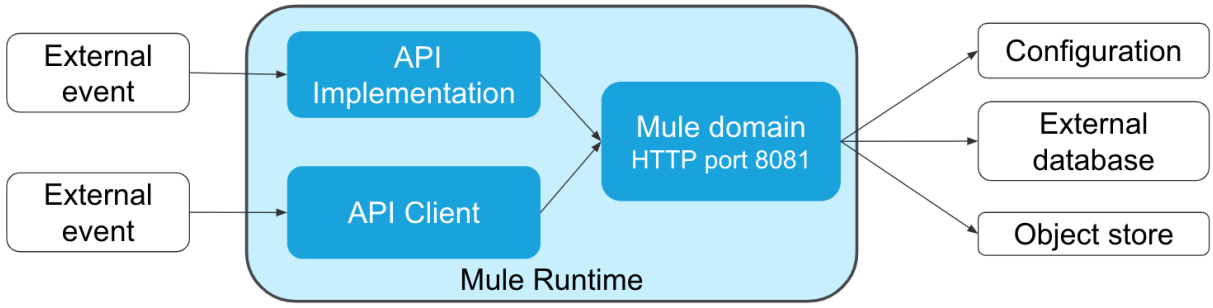
49. How is the REST Connector different from the APIkit?

50. Comparing APIkit and the REST Connector, how is the API lifecycle (create, run, test) impacted?

Walkthrough 3-3: Create a Mule domain project to share configs in a customer-hosted Mule runtime

In this walkthrough, you deploy a customer-hosted Mule domain containing some shared global Mule resources on a single Mule runtime. Then you deploy several Mule applications configured to share the Mule domain's global resources. This will allow several Mule applications to share the same properties and listen on the same HTTP port without causing TCP bind errors. You will:

- Create a Mule domain project.
- Add modules/connectors to the Mule domain.
- Configure the Mule domain with properties.
- Rewrite Mule applications to participate in the Mule domain.
- Deploy a Mule domain and its Mule applications to the same Mule runtime.



Create a Mule domain project

1. In Anypoint Studio, from the File menu select New > Mule Domain Project named intsol-domain.
2. Open the mule-domain-config.xml file and select the Configuration XML tab.
3. Verify the outer XML element is named <domain:mule-domain>.

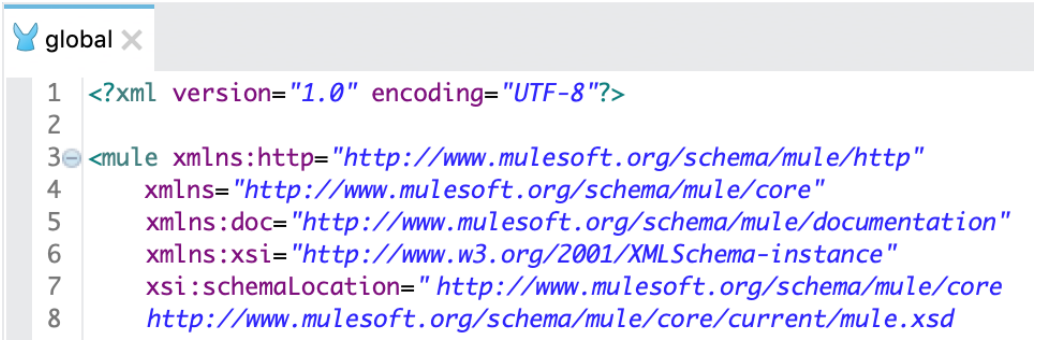
mule-domain-config

```
<?xml version="1.0" encoding="UTF-8"?>
<domain:mule-domain
        xmlns="http://www.mulesoft.org/schema/mule/core"
        xmlns:domain="http://www.mulesoft.org/schema/mule/ee/domain"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
        xsi:schemaLocation="
               http://www.mulesoft.org/schema/mule/core
               http://www.mulesoft.org/schema/mule/core/current/mule.xsd
               http://www.mulesoft.org/schema/mule/ee/domain
               http://www.mulesoft.org/schema/mule/ee/domain/current/mule-domain-ee.xsd">

    <!-- configure here resource to be shared within the domain -->

</domain:mule-domain>
```

4. Locate the xmlns:domain namespace.
5. Select an XML configuration file from another Mule project and select the Configuration XML tab.
6. Verify the outer XML element is named <mule>.



7. Compare the other namespace elements with a Mule project XML configuration file; you should see the default namespace xmlns and the xmlns:doc and xmlns:xsi namespaces have the same values in both XML configuration files, with the same xsi:schemaLocation URLs.

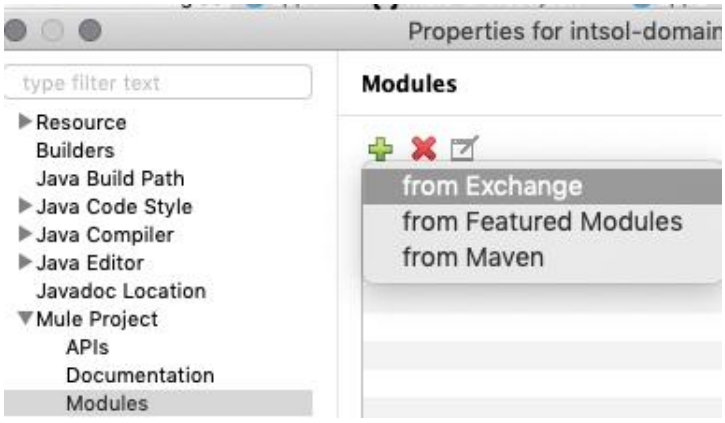
Compare a Mule domain's pom.xml file with a Mule application's pom.xml file

8. In the intsol-domain project, open the pom.xml file.
9. In the top of the pom.xml file, locate the packaging type for the Maven coordinates; it should be mule-domain.
10. View the pom.xml file for a Mule application; the packaging type for the Maven coordinates should say mule-application.
11. View any dependencies or shared libraries; unlike a Mule project, there should be no dependencies nor shared libraries configured for a new Mule domain project.
12. Compare the Mule application and Mule domain pom.xml files and verify they both set the Maven coordinates groupId, artifactId, and version.

Add required modules/connectors to the domain's dependencies

13. Select the mule-domain-config tab and select the Global Elements tab.
14. In the upper right, click Manage Modules.

15. In the Properties for intsol-domain dialog box, click the Add modules icon then select from Exchange.



16. Log in to Anypoint Platform.
17. In the Add Dependencies to Project dialog-box, search for http.
18. Select HTTP Connector - Mule 4 and click Add.
19. Click Finish.

    Note: Not all types of connector/module are supported for domains.

20. Click Apply and Close.
21. In the upper right, click Create.
22. Search for HTTP.
23. Select HTTP Listener config.
24. Click OK.
25. Accept the defaults and click OK.

Create a Mule application and link to the domain

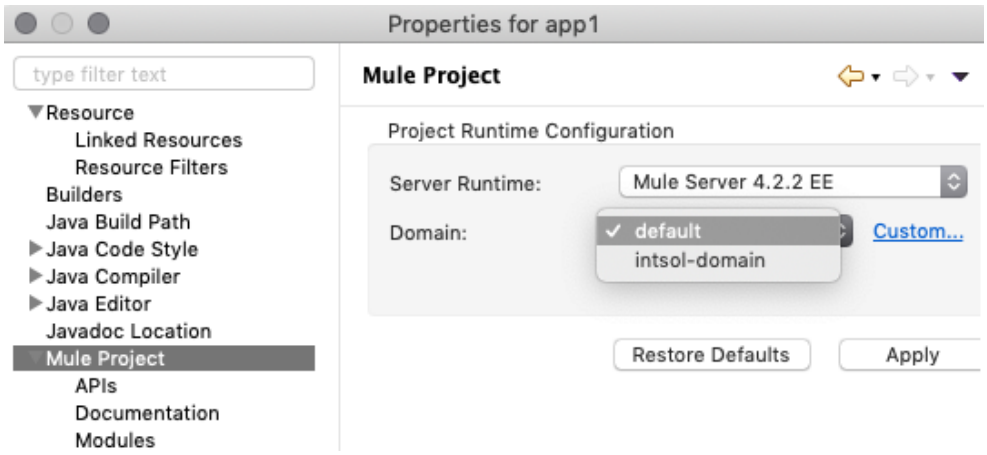
26. Create a new Mule project named app1.
27. Add an HTTP Listener to the start of the app1Flow flow.
28. Set the HTTP Listener's path to app1.

    Note: Do not create a new connector configuration.

29. Add a Set Payload component to the flow and set the message to "app1".
30. In the Package Explorer view, right-click the app1 Mule project folder and then select Properties; you should see the Modules folder selected with the HTTP and Sockets modules loaded.

31. Select the parent Mule project folder.

32. Change the Domain drop-down list from default to intsol-domain.



33. Click Apply and Close.

Resolve dependencies between app and domain

34. Return to app1 tab and select the HTTP Listener.
35. In the Connector configuration drop-down list, select HTTP_Listener_config then save.
36. In the Problems tab, right-click the warning and use the quick fix to resolve the multiple definitions the repeat in the Mule domain and Mule application pom.xml files.

Create a second application to use the domain

37. Create a new Mule project named app2.
38. Click Apply and Close.
39. Open app2.xml and add an HTTP Listener to the canvas; the Connector configuration should automatically be set to the Mule domain's HTTP_Listener_config, which is shared with app1.

40. Set the path to app2.
41. Add a Set Payload component to the flow and set the message to "app2".
42. Select Mule Project and change the Domain drop-down list to intsol-domain.

Export the Mule domain and Mule applications from Anypoint Studio

43. Export the intsol-domain project as a deployable archive JAR file; make sure to include project modules and dependencies.
44. Export the app1 and app2 Mule projects.

Deploy the Mule domain project to a Mule runtime

45. In a file browser, navigate to the mule4-node1 folder.
46. Delete any application token files from the apps folder.
47. Copy the intsol-domain-1.0.0-SNAPSHOT-mule-domain.jar file to the mule4-node1 Mule runtime's domains folder.
48. In the mule4-node1 command-line interface, verify INFO messages report the intsol-domain Mule domain deployed successfully.
49. In the mule4-node1/domains folder, verify an anchor.txt file is created for the intsol-domain Mule domain.

Deploy Mule applications to the Mule domain

50. From Runtime Manager, deploy app1 and app2 to the mule4-node1 target.
51. Verify both Mule applications start without error; this means both applications are sharing the Mule domain's HTTP Listener global configuration element that listens on TCP 8081.
52. In a web browser, make requests to http://localhost:8081/app1 and http://locahost:8081/app2 and verify each URL returns a different response; this shows that both HTTP Listeners are sharing the one Mule domain's HTTP Listener configuration across both Mule applications.

Answer these reflection questions

53. When are Mule domains useful, and what are the trade-offs of coupling Mule applications with a Mule domain?

54. Why are Mule domains not needed in CloudHub?

55. Can a Mule domain be deployed in Runtime Fabric, and if so, how and when would they be helpful?

56. How are these shared resources implemented and how do they behave between Mule applications in a Mule domain?

- HTTP Listener
- HTTP Request
- VM Listener
- Object Store connector
- File Listener (On New or Updated File operation)
- Database Listener (On Table Row operation)

Walkthrough 3-4: Explore basic DataWeave expressions

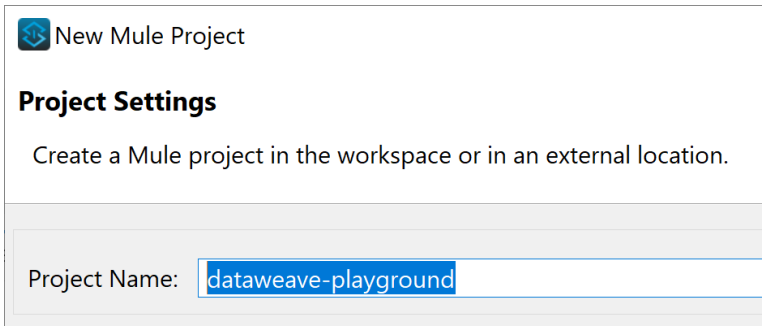
In this walkthrough, you write some simple DataWeave expressions and preview the results in Anypoint Studio. You will:

- Write some DataWeave expressions in a component in a Mule application.
- Preview the result of DataWeave expressions in the Preview pane of the DataWeave editor.
- Explore DataWeave types.
- Explore DataWeave output types.

```
%dw 2.0
output application/xml
---
outerTag: {
    hello: "world",
    aNumber: 42,
    aBoolean: true,
    aNull: null,
    anArray: [1,2,3],
    anObject: {}
}
```

Create a dataweave-playground project

1. In Anypoint Studio, create a project named dataweave-playground.

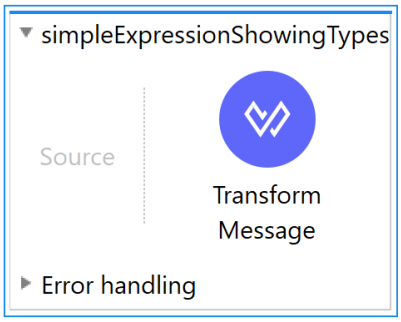


Note: This project will be a place to explore DataWeave syntax and functions in small digestible chunks.

Note: Try to leave this project in a working state when you finish experiments. You may wish to put this into source control and share with others who are learning DataWeave syntax.

Write a hello world expression using transform message to explore data types

2. Drag a Transform Message component onto the canvas and rename the created flow to simpleExpressionShowingTypes.

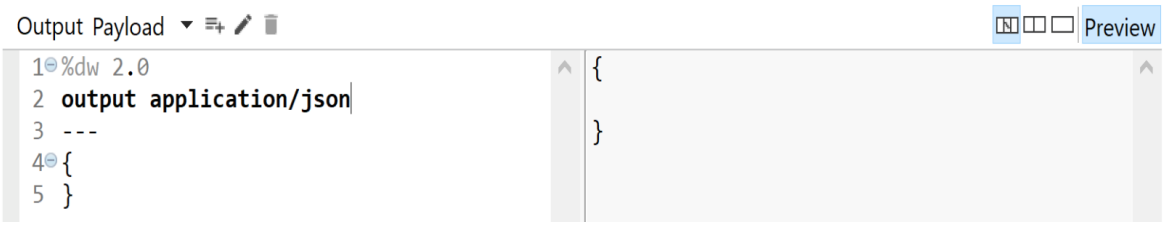


Note: This will be a "hello world" of sorts to show the different types that make up whatever output format/structure you are creating. For other experiments, you can divide up your examples into different Mule configuration XML files based on the broad category (e.g. xml-examples or strings-module-functions) to avoid one massive XML file and to aid finding them later. The flow names can be used to describe what it is this example is about.

3. Click Preview; the Preview pane should toggle on and display an empty Java object using the default output type application/java.

Explore different output formats DataWeave can produce

4. Change the output format from application/java to application/json; you should see different output display in the Preview pane.



Note: DataWeave allows quick conversion between popular output formats. DataWeave expressions in the Output pane use DataWeave format to describe data structures and

formatting, which is a format compatible with all the different output types, such as JSON, XML, CSV, flat files, and Java objects. The Preview pane displays the output in the configured output format. You never code DataWeave in other formats besides the native DataWeave format.

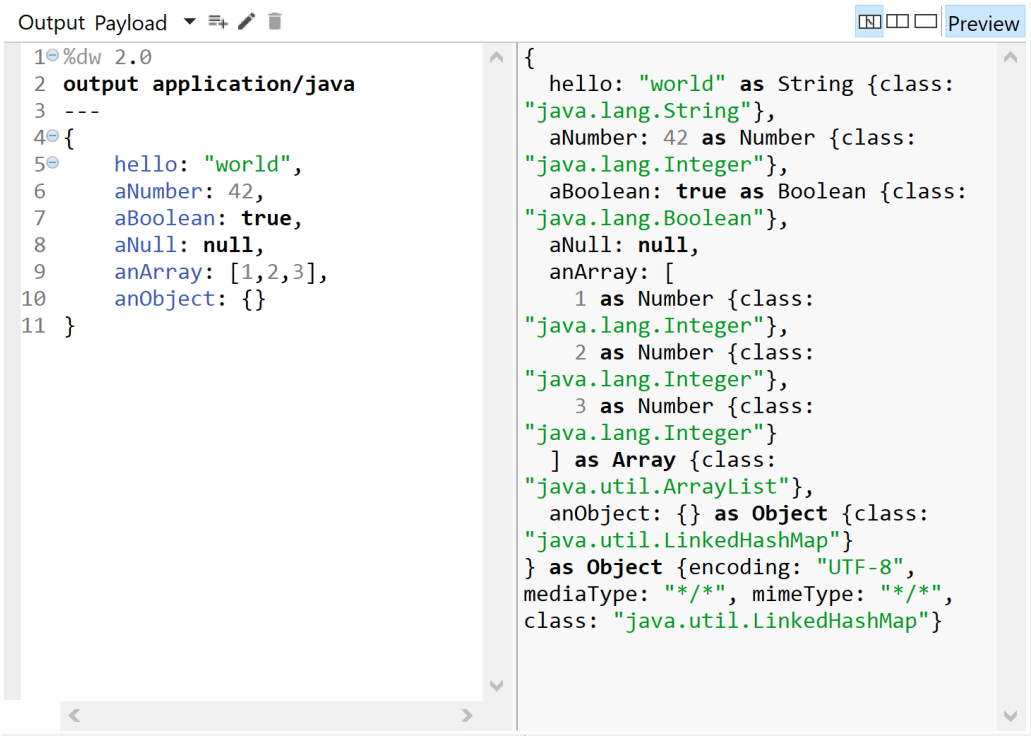
5. In the Output pane, in the body expression, place the cursor between the object constructor curly braces { } and add a key/value pair hello: "world".

Note: Just like JSON, the key:value pairs need to be separated by commas.

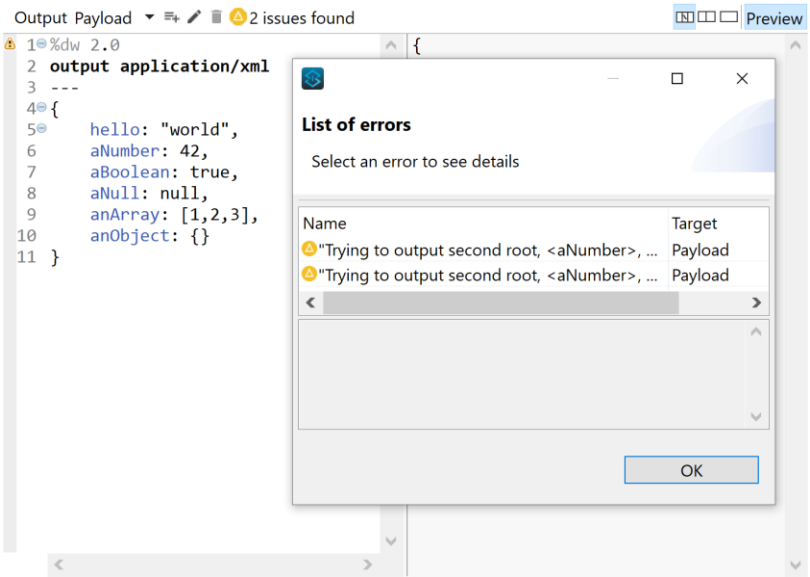
Note: You can see the similarities between DW structure and JSON – although DW is able to omit the quotes around the keys (unless the key uses special characters that must be escaped, or a non alpha-numeric character such as - or _, or has spaces between words).

6. In a web browser, view the list of DataWeave types in the documentation: https://docs.mulesoft.com/mule-runtime/4.3/dataweave-formats
7. Return to Anypoint Studio and use the documentation to try some examples of various DataWeave types to see how they produce different output in JSON.

8. Change the output type to application/java.
9. In the Preview pane, observe how the output is now represented in Java instead of JSON format.



10. Change the output directive to application/xml output; errors should appear.



Note: The errors are because XML demands that structures be underneath a top level tag. DataWeave is not magically able to produce invalid/non-conforming data structures for a given output.

11. Click Preview to toggle off the Preview pane; the errors should go away.

    Note: The error goes away because the script is valid DataWeave syntax, but the sample input provided cannot be transformed to valid XML by the output writer once you toggle on the Preview pane. To see the difference, you can remove a comma to create a broken DataWeave syntax error without Preview toggled.

12. To fix the XML parsing error, add a root-level key (outer-level key) and add the current DataWeave expression as the child value of this root-level key.

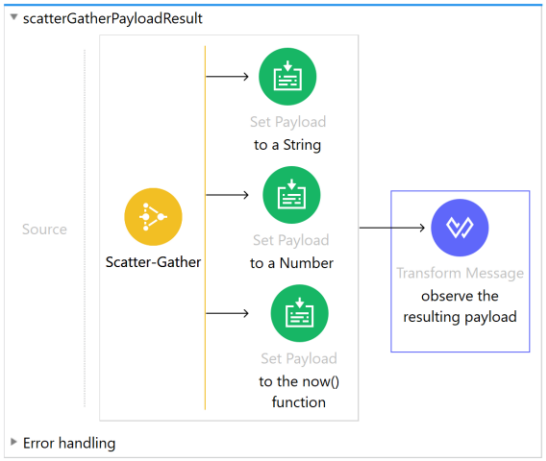
    ```
    %dw 2.0
    output application/xml
    ---
    outerTag: {
        hello: "world",
        aNumber: 42,
        aBoolean: true,
        aNull: null,
        anArray: [1,2,3],
        anObject: {}
    }
    ```

13. Verify the XML parsing error is fixed and valid XML is displayed in the Preview pane, starting with the new outerTag element.

Walkthrough 3-5: Explore Scatter-Gather behavior in a Mule application

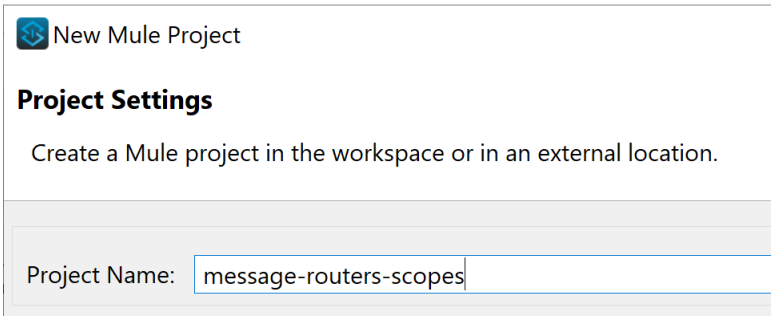
In this walkthrough, you examine how a Scatter-Gather combines results from multiple routes that return results in parallel and in any order. You will:

- Add a Scatter-Gather to a Mule application flow.
- Populate various Scatter-Gather routes with different types.
- Observe how a Scatter-Gather merges payloads from all its routes.



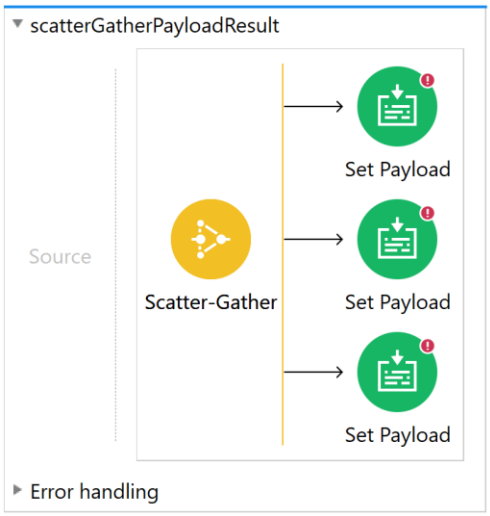
Create a Mule application that uses a Scatter-Gather

1. Drop In Anypoint Studio, create a new project named message-routers-scopes.



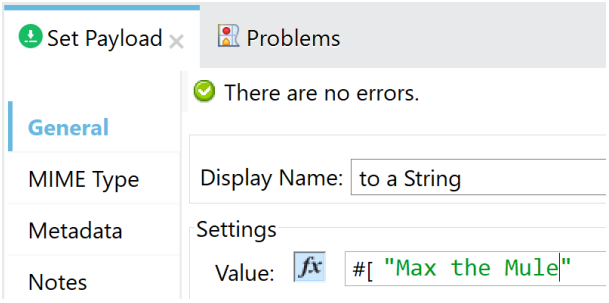
2. Drop a Scatter-Gather onto the canvas and rename the flow to scatterGatherPayloadResult.

3. Add three Set Payload components to the Scatter-Gather; carefully drop each component in the Scatter-Gather where you see the black vertical landing line appear, so that each component creates a new route in the Scatter-Gather.

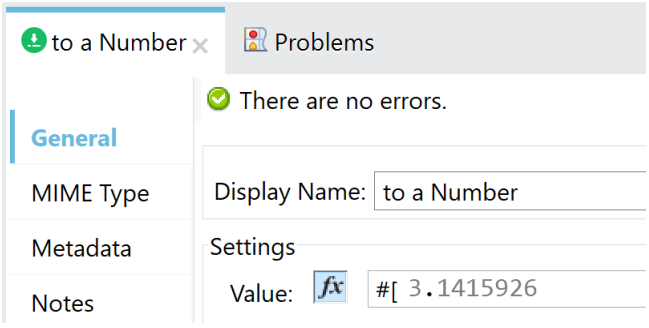


Set each route to create a different payload to see how DataSense works in each route

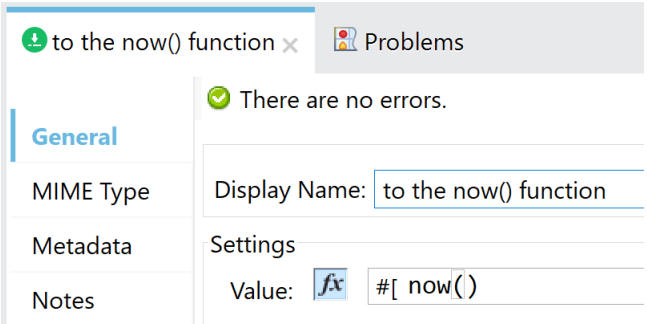
4. In the top route, set the Set Payload component's value to "Max the Mule" to output a DataWeave String type.



5. In the middle route, set the Set Payload component's value to output a DataWeave Number type.



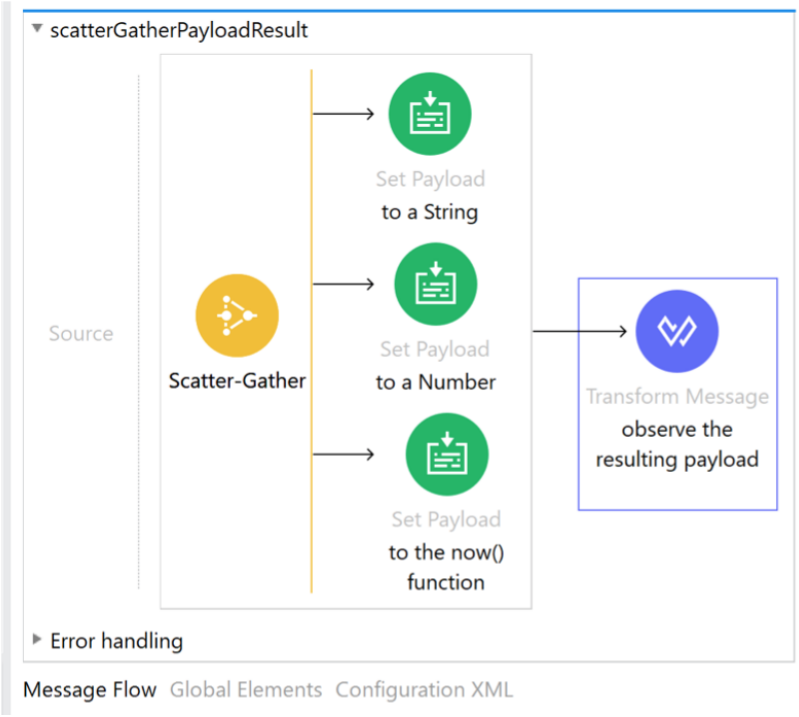
6. In the bottom route, set the Set Payload component's value now() to return a DataWeave DateTime type.



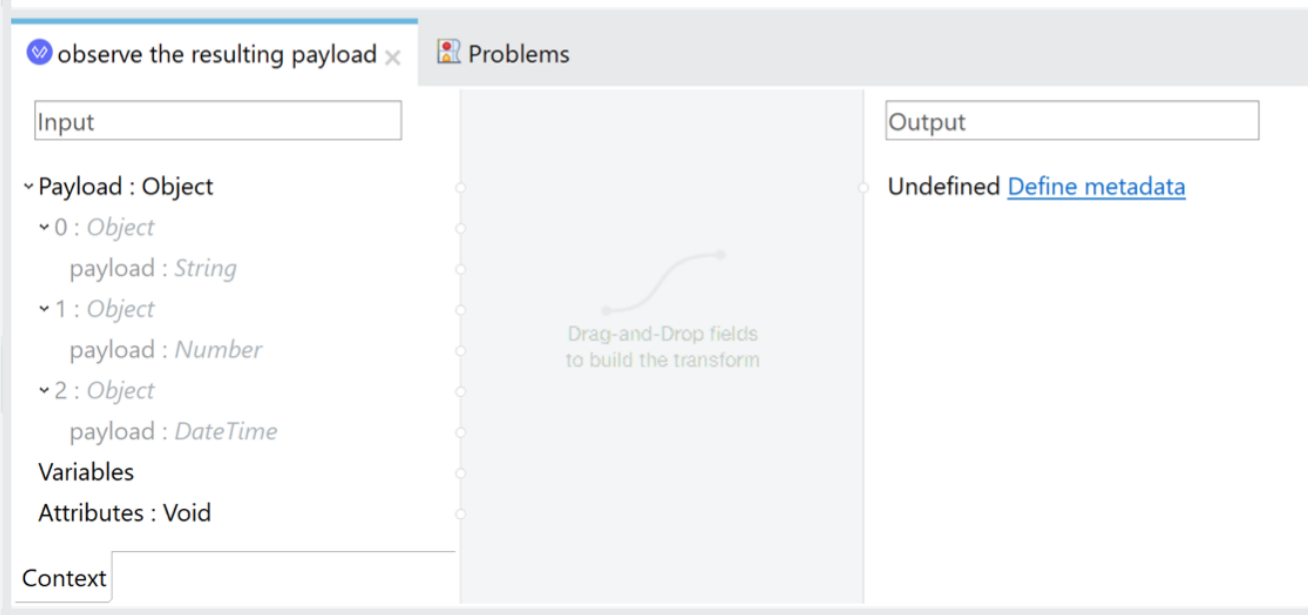
Note: The three routes execute concurrently in different threads, each returning different types that must finally be combined (or "gathered") in the final Mule event outputted from the Scatter-Gather component. DataSense will keep a track of the type of each route's payload and show how they are transformed into a single resulting combined structure.

Observe the Mule event structure that results from the Scatter-Gather

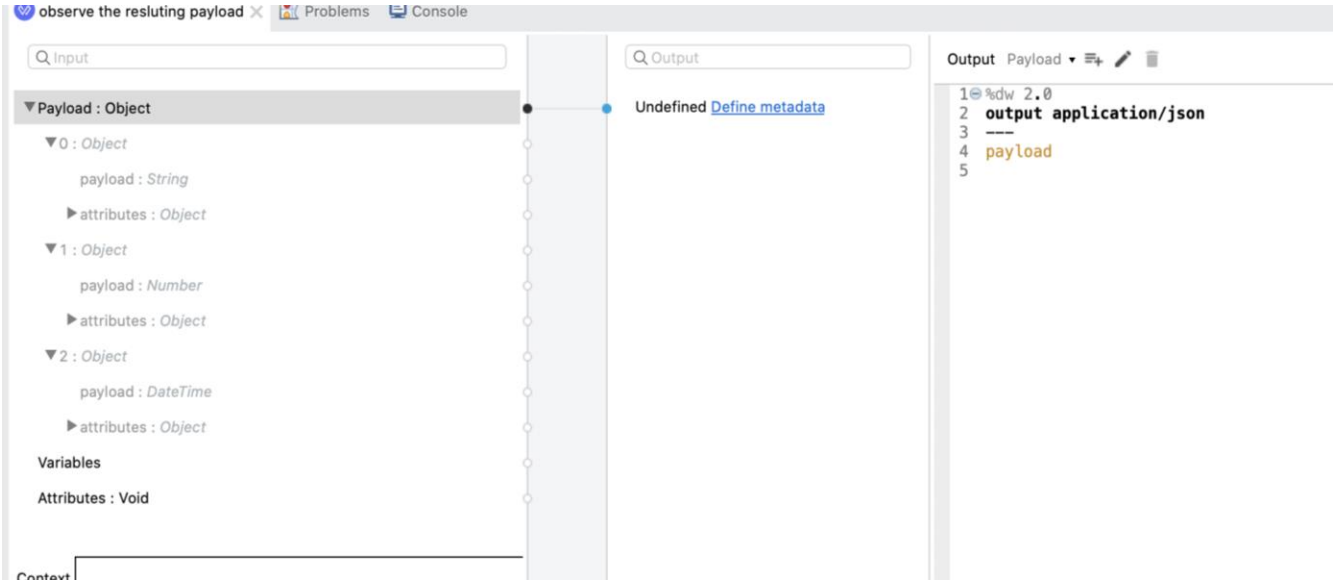
7. Add a Transform Message component after the Scatter-Gather.



8. Below, in the Transform Message view, look at the left side Input pane; you should see the DataSense preview of the Scatter-Gather result with an outer payload object organizing the three different payload objects returned from each of the three routes.



9. In the Output pane, change the DataWeave output header to "application/json" and add “payload”.



10. Add an HTTP Listener as a source of the flow.
11. Select HTTP Listener, add an HTTP connector configuration with default values and also set the Path to “/SG”
12. Run the Mule application.

13. After the Mule application starts, in a web client, send a request to http://localhost:8081/SG.
14. Verify the response to the web client is JSON with a data structure similar to what you saw in the Input pane of the Transform Message component.

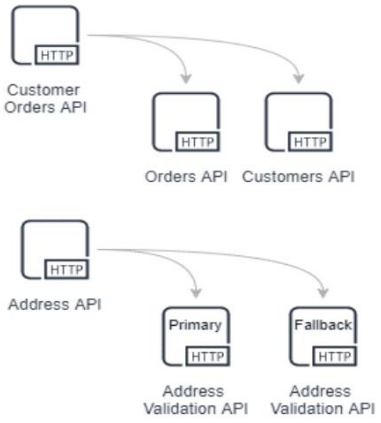
Answer this reflection question

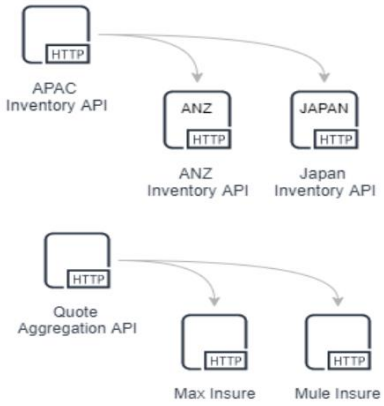
15. When might a Scatter-Gather be used vs. a Choice router?

Exercise 3-6: Design an integration application that consumes and combines two external APIs

In this walkthrough, you design a Mule flow to retrieve data from two separate APIs based on some requirements. You will create a different flow for each requirement and consider what routers are to be used and how to deal with errors. You will:

- Design Mule flows to call multiple APIs in various common scenarios.
- Pick the most idiomatic (used for its intended purposes) options for each multiple-API scenario.





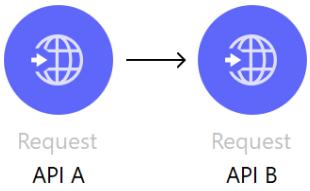
Create a new Mule configuration file in which to work

1. Return to Anypoint Studio.
2. Add a new Mule configuration file named combiningTwoAPIs to the message-routers-scopes Mule project's src/main/mule folder.
3. Select the combiningTwoAPIs tab, then drag out and drop an HTTP listener to the canvas.

   Note: Do not be too concerned about working code – we are exploring concepts.

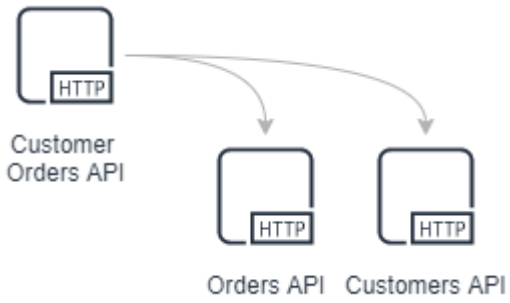
Create a serial invocation example where APIs are called synchronously in a sequence

4. Create a prototype flow where the output of first API is needed by second API



Case 1: Create an example of joining two different APIs together

5. Create a flow that calls an Orders API and Customers API and then joins the results together.



6. What scopes or other options are available to call both APIs and merge the results?

7. Compare the pros/cons of these options.

	Option 1: Scatter-Gather	**Option 2: Async scope**	**Option 3:**
Parallel execution	Yes	Yes	
When and how are the two API results merged?		They cannot be merged because the flow completes before the async routes completes	
Possible performance improvement options			
Where could message queues like JMS be used?			
What happens if one API fails?			

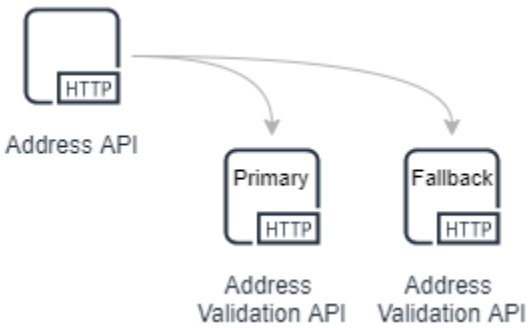
8. Consider any other options.

	Option 1:	Option 2:	Option 3:
Parallel execution			
When and how are the two API results merged?			
Possible performance improvement options			
Where could message queues like JMS be used?			

9. Prototype solutions in Anypoint Studio and paste images of the flows here:

Case 2: Create an example of a fall-back API

10. Create a prototype flow where two Address Validation API endpoints can be called to perform identical functions, but one is to be a fall-back option.



11. What scopes or other options are available to call both APIs and merge the results?

12. Compare the pros/cons of these options.

	Option 1: Scatter-Gather	**Option 2: Async scope**	**Option 3:**
Parallel execution	Yes	Yes	
When and how are the two API results merged?		They cannot be merged because the flow completes before the async routes completes	
Possible performance improvement options			
Where could message queues like JMS be used?			
What happens if one API fails?			

13. Consider any other options.

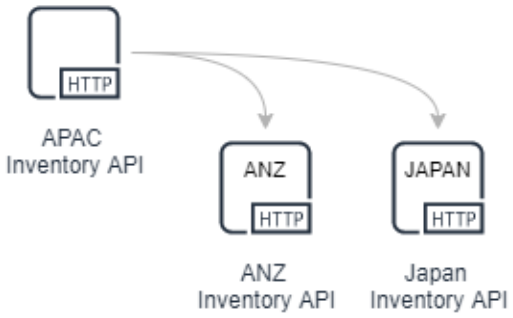
	Option 1:	**Option 2:**	**Option 3:**
Parallel execution			
When and how are the two API results merged?			
Possible performance improvement options			

Where could message queues like JMS be used?			

14. Prototype solutions in Anypoint Studio and paste images of the flows here.

Case 3: Create an example of calling one of two available APIs based on the region of the request

15. Create a flow that calls an Inventory API endpoint in one of two regions(ANZ, JAPAN).



16. What scopes or other options are available to call both APIs and merge the results?

17. Compare the pros/cons of these options.

	Option 1: Scatter-Gather	Option 2: Async scope	Option 3:
Parallel execution	Yes	Yes	
When and how are the two API results merged?		They cannot be merged because the flow completes before the async routes completes	

Possible performance improvement options			
Where could message queues like JMS be used?			
What happens if one API fails?			

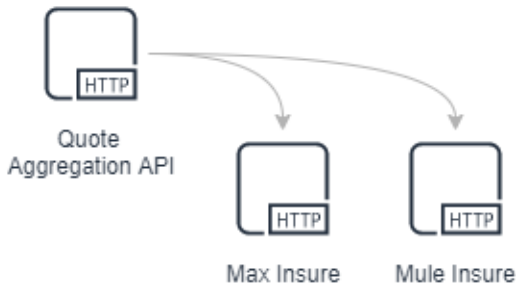
18. Consider any other options.

	Option 1:	**Option 2:**	**Option 3:**
Parallel execution			
When and how are the two API results merged?			
Possible performance improvement options			
Where could message queues like JMS be used?			

19. Prototype solutions in Anypoint Studio and paste images of the flows here.

Case 4: Create a global comparison example that selects the best result from several similar API calls to different providers/vendors

20. Create a flow to call two different Insurance Quote APIs from two insurers and them combine and sort the results to find the best option.



21. What scopes or other options are available to call both APIs and merge the results?

22. Compare the pros/cons of these options.

	Option 1: Scatter-Gather	**Option 2: Async scope**	**Option 3:**
Parallel execution	Yes	Yes	
When and how are the two API results merged?		They cannot be merged because the flow completes before the async routes completes	
Possible performance improvement options			
Where could message queues like JMS be used?			
What happens if one API fails?			

23. Consider any other options.

	Option 1:	Option 2:	Option 3:
Parallel execution			
When and how are the two API results merged?			
Possible performance improvement options			
Where could message queues like JMS be used?			

24. Prototype solutions in Anypoint Studio and paste images of the flows here.

Identify how to transform and combine data

25. Review the examples and look for opportunities to refactor processing into reusable flows.

Identify error handling for the solution

26. Add error handling strategies and decisions to your prototype flows.
27. Decide on error handling options and response codes.

Identify scopes to use in the integration solution

28. Use Anypoint Studio and your design documents to identify interdependencies between components the prototyped solutions.

User story	Related/dependent components	Independent components	Ways to improve the design	Impact of the design change
Case 1				

Case 2				
Case 3				
Case 4				

29. Review your design and see if there are places where the Mule application can process independent components in a different construct, such as by using a Scatter-Gather.

Answer these reflection questions

30. Give some examples of components you feel should be:
    - CPU_INTENSIVE, and why?
    - CPU_LITE, and why?
    - IO_INTENSTIVE (non-blocking IO), and why?
31. What would need to be done if the Mule runtime could not self-tune these component execution profiles?
32. Why would you develop a new SOAP service instead of a REST service?
33. What are the trade-offs of using SOAP vs. REST?

34. How are data schemas validated and enforced in SOAP vs. REST?

35. Does REST support XML payloads?

36. What types of data formats are supported by SOAP vs. REST?

37. What are some other common synchronous data processing models, and what are the trade-offs compared with REST or SOAP?

38. Compare how errors are communicated in REST vs. SOAP vs. other event processing models?

39. How are error status messages returned to clients in SOAP vs. REST, and what are the trade-offs between these two styles?

40. What are some popular SOAP extensions (aka the WS-* standards)?

41. What are the trade-offs between using a file stored repeatable stream (that uses Kryo) vs. an in-memory repeatable stream?

42. What type of use cases would benefit from repeatable streams?

43. When would a non-repeatable stream be helpful or necessary?

44. What are the trade-offs between using a file stored repeatable stream (that uses Kryo) vs. an in-memory repeatable stream?

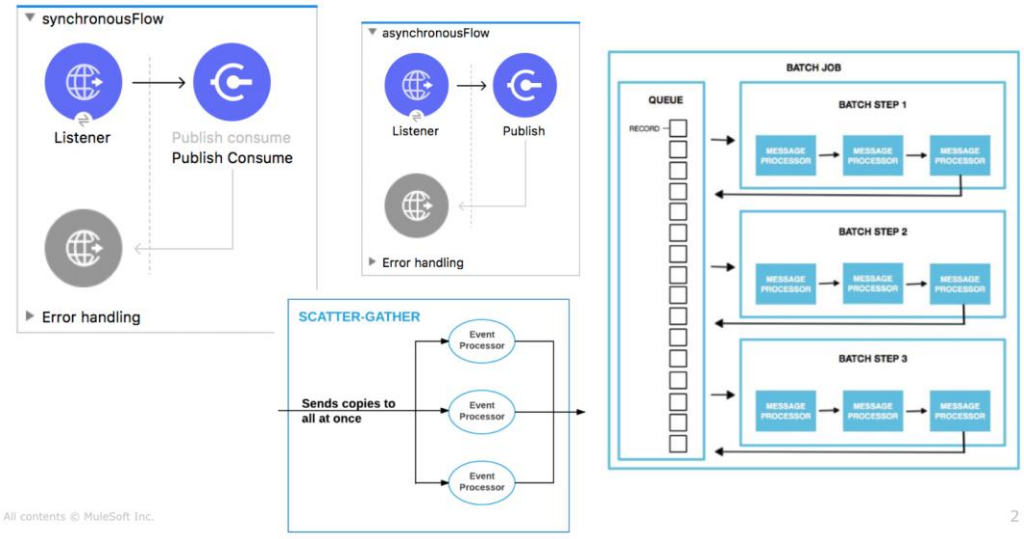
45. What type of use cases would benefit from repeatable streams?

46. When would a non-repeatable stream be helpful or necessary?

Additional References

- Salesforce event processing: https://developer.salesforce.com/docs/atlas.enus.platform_events.meta/platform_events/platform_events_intro.htm
- WS-ReliableMessaging: https://en.wikipedia.org/wiki/WS-ReliableMessaging
- XML Signature: https://en.wikipedia.org/wiki/XML_Signature
- XML Encryption: https://en.wikipedia.org/wiki/XML_Encryption

Module 4: Choosing Appropriate Mule 4 Event Processing Models



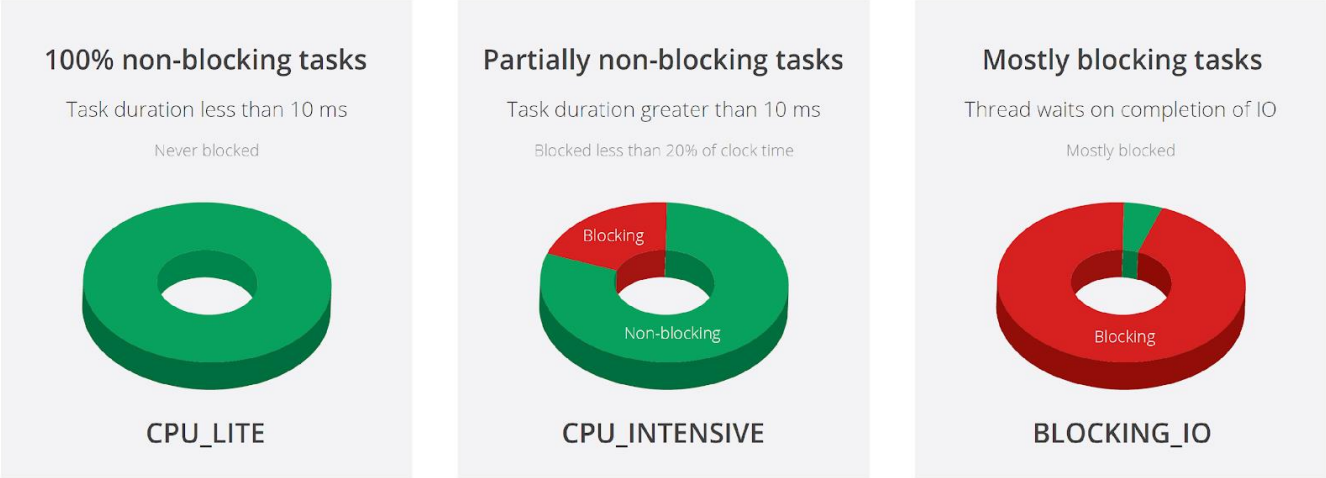
At the end of this module, you should be able to:

- Distinguish between Mule 4 blocking, non-blocking, parallel, and reactive event processing options.
- Identify the event processing models used in various Mule 4 scopes and components.
- Identify Mule 4 streaming options and behaviors.
- Describe event processing options with JMS and VM connectors.
- Select appropriate event processing for an integration use case.
- Design batch, near real-time, and real-time data synchronization integration use cases.

Walkthrough 4-1: View and understand thread pool implementation of different Mule runtimes

In this walkthrough, you run VisualVM on your computer and then deploy the threadtest Mule application to stand-alone Mule runtime to understand thread pool implementation differences between Mule 4.3+ and previous versions. You will:

- Run VisualVM on computer.
- Deploy Mule application to the stand-alone Mule Runtime.
- View and understand thread pools.



Run VisualVM on computer

1. Locate and open VisualVM you downloaded and installed at the beginning of the class.

   Note: JConsole can be used to view the threads alternatively. From command line type jconsole to open Java Monitoring & Management Console.

Deploy Mule application to the Mule Runtime

2. Open a command-line interface and navigate to the $APAINTSOL_HOME/mule4-node1/bin.
3. On Windows, type mule.bat, or on a Unix machine, type ./mule.

4. Verify the Mule runtime starts without errors.

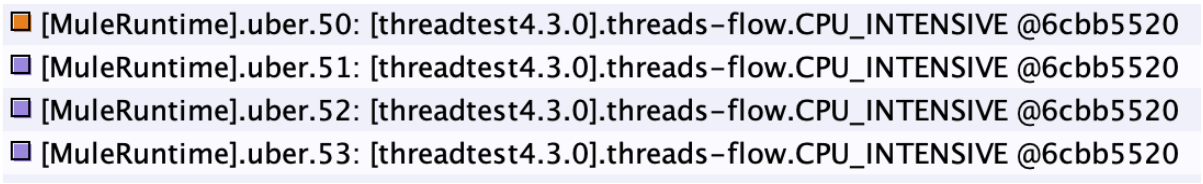
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+ Mule is up and kicking (every 5000ms)                                          +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**************************************************************************
*               - - + DOMAIN + - -               * - - + STATUS + - - *
**************************************************************************
* default                                        * DEPLOYED           *
**************************************************************************
```

5. Verify you see console messages about the Mule runtime using the Mule Agent to connect to Runtime Manager over a web socket.
6. Delete the existing applications from $APAINTSOL_HOME/mule4-node1/apps.
7. Locate and copy $APAINTSOL_HOME/studentFiles/mod04-choose-processing-models/starter_resources/threadtest4.3.0.jar to $APAINTSOL_HOME/mule4-node1/apps.
8. Verify the Mule 4 runtime logs messages about the threadtest4.3.0 Mule application stopping and starting.
9. Return to your REST client.
10. Submit a GET request to http://localhost:8081/threadTest.
11. Verify a 200 OK status is returned with the string response.

View and understand thread pools

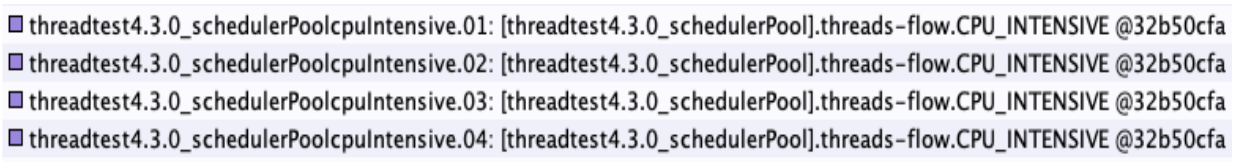
12. Return to VisualVM and locate the Mule runtime then click on Threads.
13. Verify UBER thread pool is used for all the three execution types.



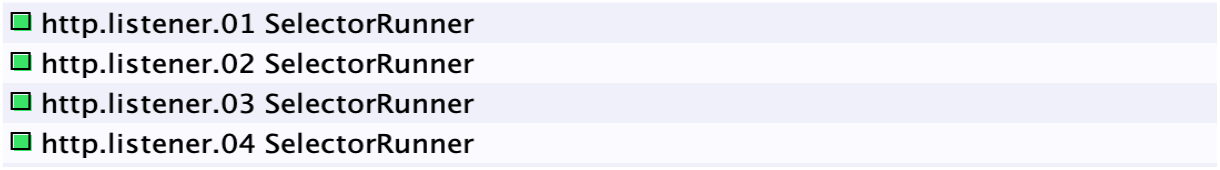
14. Locate and copy $APAINTSOL_HOME/studentFiles/mod04-choose-processing-models/starter_resources/threadtest4.3.0_schedulerPool.jar to $APAINTSOL_HOME/mule4-node1/apps.

    Note: This Mule application is built on Mule Runtime 4.3.0 and configured with scheduler pools property which overrides the thread pool behavior of stand-alone Runtime.

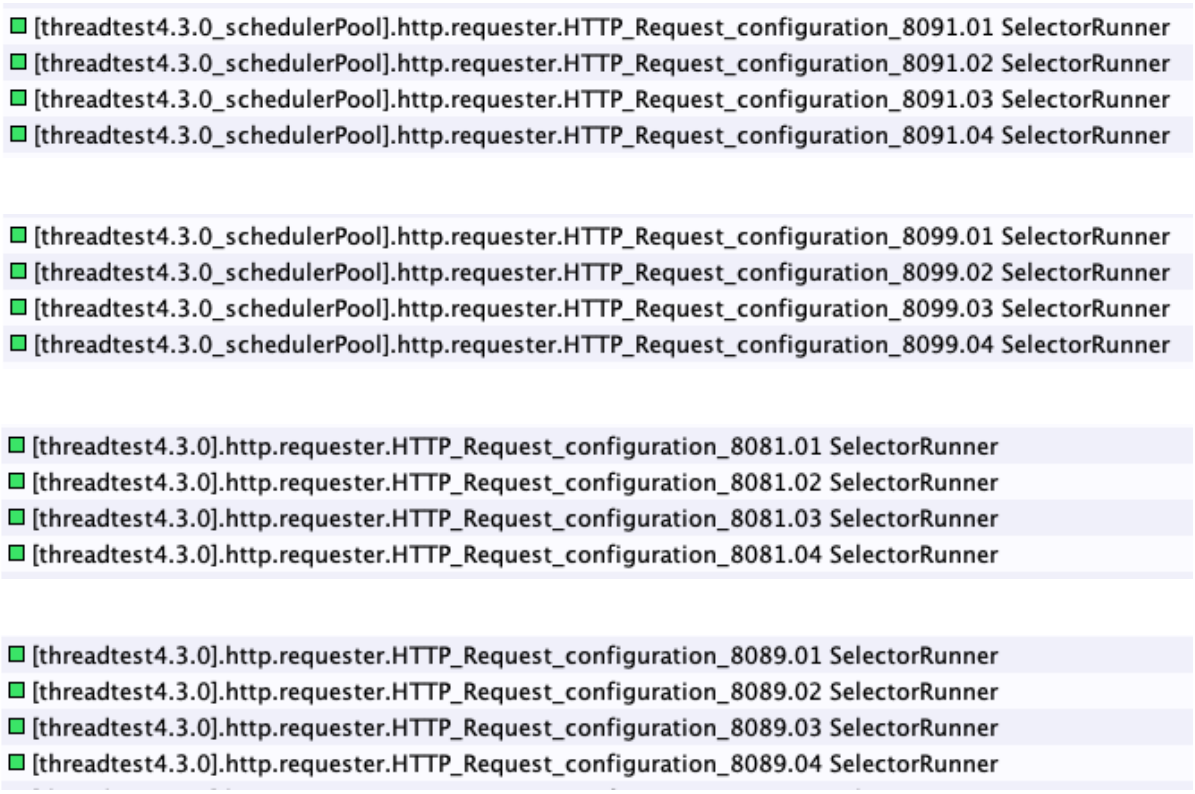
15. Observe DEDICATED thread pool is used for the threadtest4.2.2 Mule application.



16. Verify HTTP Listener uses a shared selector pool provided by the Mule runtime for both Mule applications.



17. Observe HTTP Requester has a dedicated selector pool for each HTTP connection

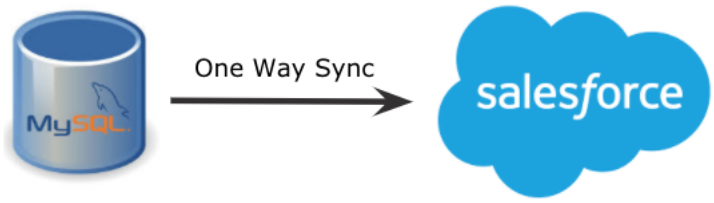


Note: Both Mule applications are using different HTTP connections with different ports. Observe that different selector pools are allocated for each HTTP request configuration for 8091, 8099, 8081 and 8089.

Exercise 4-2: Model Mule event processing for a SaaS data sync use case

In this exercise, you design a one-directional synchronization integration solution between two enterprise systems: from a MySQL database to a Salesforce CRM system. You evaluate Mule event processing options, decisions, and their trade-offs. You will:

- Choose the most appropriate integration style to design a one-way data synchronization integration solution that meets an organization's current requirement
- Analyze the impact of competing requirements on integration style
- Modify the integration style (Mule event processing models) to balance data throughput vs. data processing latency requirements



Review the context and requirements for the database to SaaS use case

1. Review the background context associated this one-way data synchronization use case.
    - Randomly, a few times a day, approximately a million rows of customer data are added to the MySQL database from another integration application.
    - The Mule application must quickly forward these changed database rows to Salesforce.
    - Only new or modified database entries should be processed.
    - Records must be transformed to match the Salesforce data structures.
    - Salesforce bulk operations should be used.
    - Auditing and traceability of each record is critical.
    - Acknowledgements must be sent to other endpoints and logs when records or files are received and when data is successful written to the target database.
    - Human intervention must be allowed to post process any failed database rows.
    - The Mule application has been budgeted to deploy to one 0.2 vCore CloudHub worker (with 1 GB of heap memory).

Identify Mule event processing options to read in changes to a table in a source database

2. Identity options to read in changes from a source database.
    - How can the database trigger events to external endpoints when a row or rows change in the database?

    - What MuleSoft components can monitor a dataset for changes and read in those changes to a Mule application?

Identify Mule event processing options to write changes to Salesforce

3. Identify options to write changes to Salesforce.
    - How can changed records be written to Salesforce?
    - What MuleSoft components can write changes to Salesforce?
    - Can records be written in bulk to Salesforce, or must they be written one record at a time?
    - What Salesforce SLAs must be considered?

Use Anypoint Studio and the MuleSoft documentation to continue to design and analyze the integration solution

4. Create a Mule project and add connectors and components to represent the options described in the last step; continue to use Anypoint Studio and the MuleSoft documentation to complete the exercise.

Analyze options to build the integration solution

5. Analyze options to read in database changes.
    - How can you trigger a Mule application when a row or rows are changed in the database?

    - What are options to manage, monitor, and restart connections to the source database?

    - How can you trigger a Mule application when a row or rows are changed in the database?

- Identify where schedulers or batch jobs might be applied, and in which components.

- Identify the types of errors that might occur, and how they could be handled.

6. Document the data model for the source database system.

   Note: Use the results of the previous exercise to document the data model of records from the database; invent any additional database schema that might be required to properly update the Salesforce system.

   Note: For this exercise you can quickly make something up now, then return to this section later as you refine your design.

   - Document tables and views that should be used to synchronize with Salesforce.

   - Document elements and attributes to be transformed, and if they are required or not.

   - Document any extension element beyond industry standards that must be considered.

7. Document the schema for the target Salesforce system.

   Note: For this exercise you can quickly make something up now, then return to this section later as you refine your design.

   - Document the structure of Salesforce objects that will store each database record or records.

   - Document elements and attributes to be written to, including the data type, max/min ranges, or other data formatting restrictions, and if they are required or not.

   - Document any extension element beyond industry standards, such as custom headers or fault schemas for web services interfaces.

   - Document allowed values restrictions for the documented elements and attributes.

- Document file system operations and their allowed values.

- Provide any sample documents to support this design.

Decide and document assumptions regarding SLAs for the source system in the use case

8. Document SLAs related to reading files from the file system.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Processing latency to process each file record and the total file?		
What file system SLAs restrict how fast or how much data can be written to the database?		
What should be done in the source file system with processed files, and when should the source files be modified (before, during, or after the file records are processed or written to the database)?		

9. Document SLAs related to reading changes from the source database.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Must data be read from the database system in real time or near real time?		
If not, at what rate are file system reads scheduled?		
Will scheduled data processing be at a fixed rate, or on specific dates (cron jobs)?	Will schedule data processing be at a fixed rate, or on specific dates (cron jobs)?	

What is the average number of records and recrod size per scheduling cycle?		
What is the worst case (upper bound) number of rows and record size per scheduling cycle?		
If data processing is not scheduled, what is the maximum burst of database changes and record sizes the solution should support?		
If data processing is not scheduled, are there long periods (months?) that the peak number of database changes or record sizes is much smaller than the "busy" periods?		

10. Document record processing SLAs.

SLA related question	Assumptions, agreements, and requirements	Event processing model changes
Processing latency to process new records from the database?		
What databasse SLAs restrict how fast or how much data can be read from the database?		

Analyze data processing requirements and trade-offs for the use case

11. Decide how the processing model can change based on changes to these competing data processing SLAs.

SLA related question	Assumptions, agreements, and requirements	Event processing model changes
What are the average and maximum throughput rates and how do they affect the processing model?		

What are the data processing latency SLAs and how to they affect the processing model?		
How is data written to the Salesforce?		
How is flow processing triggered?		
Must the database read option change to support real time or near real time latency SLAs?		
Does the number or size of records read impact how records are processed?		
How can the data synchronization between the system be limited to only new or modified records?		

12. How is database state synchronized with Salesforce?

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
How does Salesforce signal the Mule application that records were successfully written?		

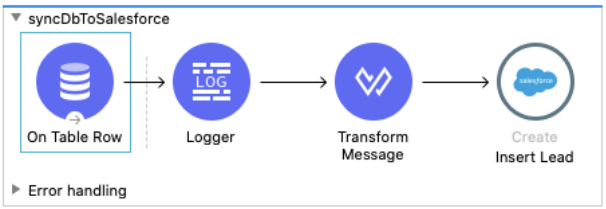
How does Salesforce signal the Mule application that records were not successfully written?		
If transactions are used, when should the transaction be committed?		
How are watermarks synchronized with writes to Salesforce?		
If transactions are used, what is the rollback behavior?		
How are watermarks synchronized with any rollback behavior?		
What is the impact of duplicate records being written to Salesforce?		
What is the impact of missing a record in Salesforce due to an incorrect watermark state?		

Design Mule event processing models

13. Use the previous analysis to decide the Mule event processing models to synchronize the database with Salesforce.

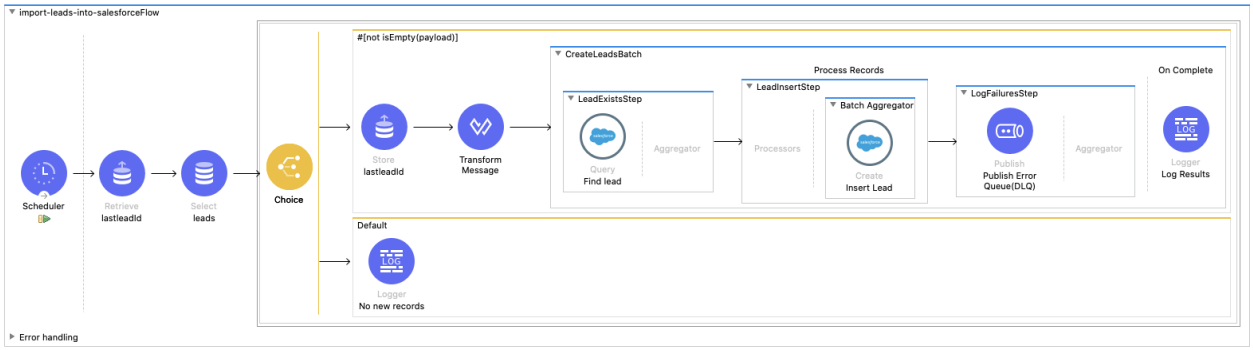
- Propose a solution that uses a For-Each scope to process the database results one record at a time.

- Prototype the solution in Anypoint Studio.



- Propose another solution using a Batch Job scope to process the database results in batches.

- Prototype the solution in Anypoint Studio.



- Is there any advantage to using a Database On Table Row source instead of a Scheduler?

- How can the memory footprint be reduced while reading from the database?

- How can the memory footprint be reduced while writing to Salesforce?

- How are database connections handled between writes to the database, between file reads, and between processing each file record?

- How are Salesforce connections handled between writes to the database, between file reads, and between processing each file record?

- How are failed records managed?

- Are transactions used?

Compare the two designs

14. Use this table to compare the designs.

Evaluation criteria	For-Each scope	Batch scope	Other option
Near real time processing			
Records (rows) processed per cycle			
Can batch together writes to Salesfore			
Types of possible performance improvements			
Trade-offs of bulk vs. individual writes			

15. Answer additional questions related to design choices and trade-offs.

- What is the trade-off of using On Table Row vs. a Scheduler and a Select operation?

- Which one of these triggers can set streaming data?

16. Design a solution using a Scheduler to trigger a database select inside a Batch Step scope.
17. What are the trade-offs to using bulk operations vs. processing individual records?

Evaluate streaming options

18. Which components should have streaming configured?

19. What are the trade-offs to enabling streaming?

20. What are the design and performance trade-offs between

- Using streaming (in-memory or in a file) and no Batch Aggregator or Bulk operation
- Not using streaming, with a Batch Aggregator and a Bulk operation

Evaluation criteria	Streaming	Not streaming	References
Performance throughput			
Performance latency			
Salesforce SLA considerations			

Identify scopes to use in the integration solution

21. Use the sketch of the integration solution to help identify interdependencies between components in the Mule application, and suggestions to improve/change the design.

Related components	Independent components	Ways to improve/change the design	Impact of the design change

Identify error handling for the solution

22. In Anypoint Studio, add error handling scopes and components to your flows.
23. Iterate on the previous design analysis to decide on error handling options.

Identify and explain every factor that helps evaluate and agree upon the best processing model

24. Document the factors specific to your use case that are involved in choosing between Mule event processing models.

Justify each Mule event processing model decision based on the identified factors

25. Give reasons for your choice of Mule event processing models based on the identified factors.

Verify requirements

26. Review the beginning of the exercise against your design and verify all the requirements are met.

Address data latency requirements, options, and trade-offs

27. How would you address a requirement for "real-time or near-real-time" integration between the database and Salesforce CRM?

Plan for future scaling requirements

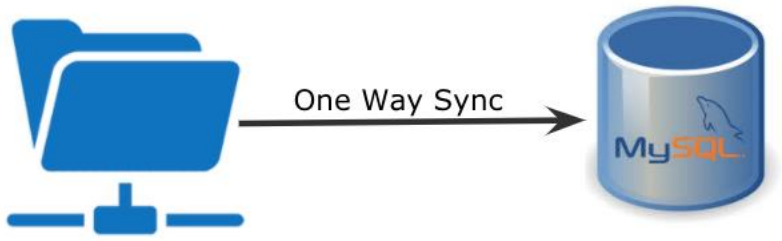
28. How would a dramatic increase in insert volume impact your solution?

29. What other options could address future scaling requirements, and what are the trade-offs?

Exercise 4-3: Design Mule event processing for a one-way file transfer use case

In this exercise, you design the Mule event processing options for the file transfer use case. You will:

- Identify and recommend the Mule event processing models for a file transfer use case.
- Identify and explain every factor that helps evaluate the best processing model.
- Justify each Mule event processing model decision based on the identified factors.
- Document flows that can implement the selected Mule event processing models.



Review the context and requirements for the file transfer use case

1. For this exercise, assume the project has the following context and requirements:

   - Randomly, a few times a day, a new source file containing financial data for customers are uploaded to a specific directory on the file server.
   - The Mule application must quickly process the file and then send results to a target MySQL database.
   - Each file has about 1 million records of customer data.
   - Auditing and traceability of each record is critical.
   - Acknowledgements must be sent to other endpoints and logs when records or files are received and when data is successful written to the target database.
   - Human intervention must be allowed to post process any failed records.
   - The Mule application has been budgeted to deploy to one 0.2 vCore CloudHub worker (with 1 GB of heap memory).

   Note: You will likely need to invent some additional assumptions to make decisions in this exercise. If you are working with a group, discuss the assumptions and document them.

Identify Mule event processing options to read in and process the file

2. Document recommended Mule event processing models for a one-way file transfer to database use case, by addressing at least these options:

   - What are the options to read a file into an integration application?

   - What are the options to read a file into a Mule application, including what type of connectors that are available?

   - After reading in a file, how are records in files processed?
   - How can records be processed sequentially, parallelly, or in batch?
   - How can the memory footprint be reduced while processing?
   - How can failed records be managed?

Prototype and analyze how to read in files from the file system

3. Create a Mule project and add connectors and components to prototype a solution to read in files from the file system.

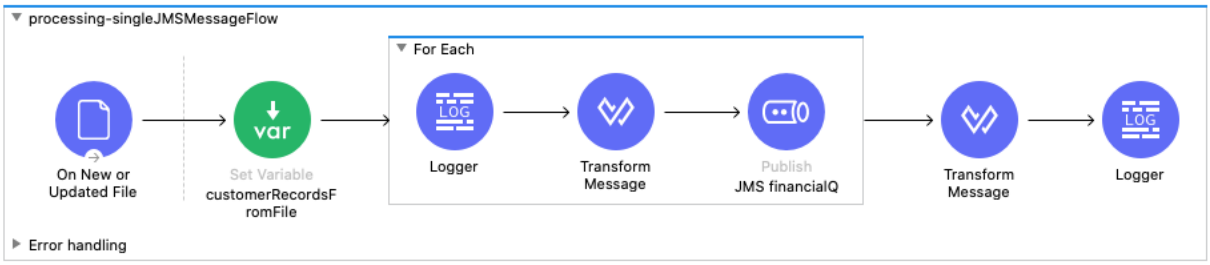
   - After reading in a file into a Mule application, what are the options for how process each file?

   - What are the options to move or delete a file in the file system after it is processed?
   - Identify how the source data will appear in a Mule application; consider the payload, headers, and attributes of the received Mule events.
   - How can DataWeave help transform the files, and what type of DataWeave patterns could be used?

   - What type of errors might occur when reading and processing a file, and how could they be handled?

Use Anypoint Studio and the MuleSoft documentation to continue to design and analyze the integration solution

4. Add components to the prototype Mule project to use a For-Each scope to process file records.



Note: The prototype Mule application can have errors and does not need to run.

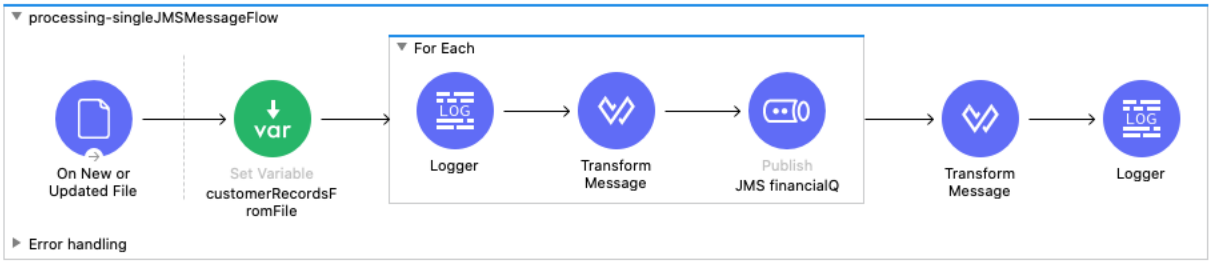
Note: Consider using a File connector, For-Each scope, Set Payload transform, Batch job, Async scope, Try scope, and error handlers.

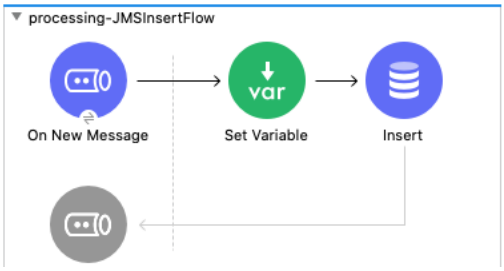
Answer questions about options to process files in a Mule appplication

5. What are options to manage, monitor, and restart connections to the source file system?

6. How could message queues decouple the integration logic and steps?

7. Prototype a solution that uses message queues to decouple the file reads from the database writes.

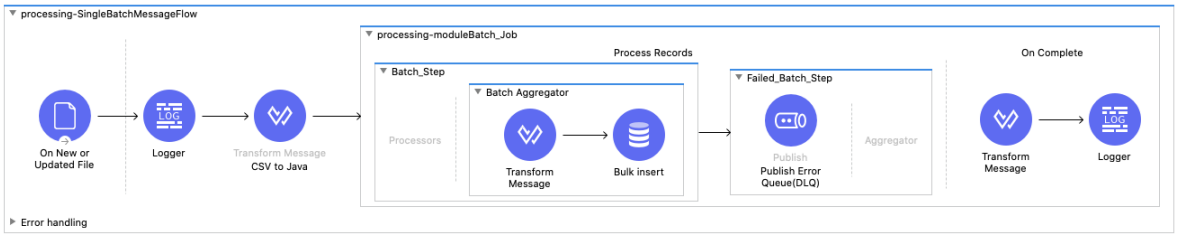




8. Where and how could schedulers or batch jobs be used?

Prototype solutions to process files

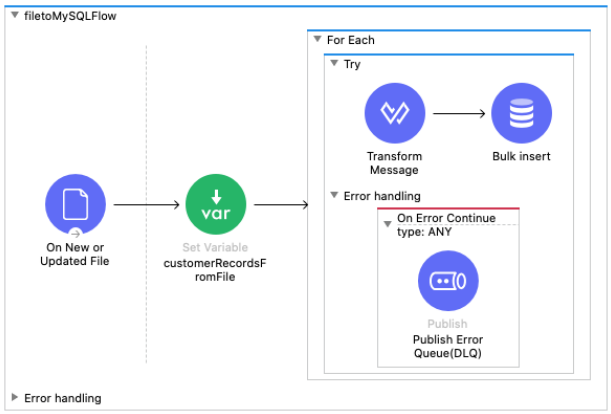
9. Create a flow that uses a Batch Job scope instead of a For-Each scope, and sends failed database write attempts to a dead-letter message queue; you can paste screenshots here.



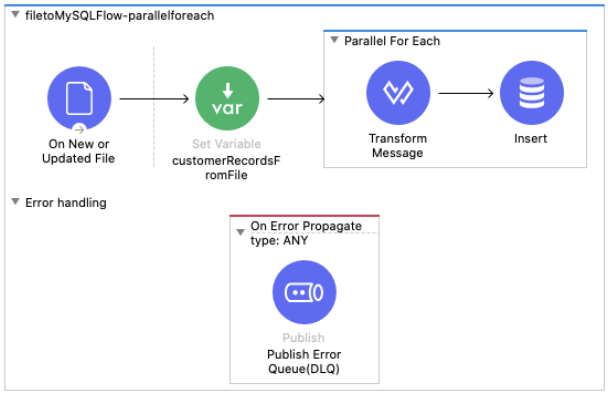
10. Document how many records at a time should be batched in each Batch Aggregator step, and configure the prototype to match this assumption.

11. For each connector that might be used, document allowed operations and possible errors.

12. Prototype a processing model that uses a For-Each scope with a batch size and a JMS dead letter queue for failed inserts; you can paste screenshots here.



13. Create a flow using a Parallel For Each scope configured to process the source file as streaming data



14. Add error handling to the all prototypes.

Decide and document assumptions regarding SLAs for the source system in the use case

15. Document SLAs related to reading files from the file system.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
What process latency is allowed or required to process each file record and the total file?		
What file system SLAs restrict how fast or how much data can be written to the database?		
What should be done in the source file system with processed files, and when should the source files be modified (before, during, or after the file records are processed or written to the database)?		

16. Document SLAs related to reading files from the file system.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Must data be read from the file system in real time or near real time?		
If not, at what rate are file system reads scheduled?		
Will scheduled data processing be at a fixed rate, or on specific dates (cron jobs)?	Will schedule data processing be at a fixed rate, or on specific dates (cron jobs)?	
What is the average number of files and file size per scheduling cycle?		
What is the worst case (upper bound) number of files and file size per scheduling cycle?		
If data processing is not scheduled, what is the maximum burst of files and file sizes the solution should support?		
If data processing is not scheduled, are there long periods (months?) that the peak number of files and file sizes is much smaller than the "busy" periods?		

17. When should the file system be modified?

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
What should be done in the source file system with processed files, and when should the source files be modified (before, during, or after the file records are processed or written to the database) ?		
Must file system modifications be performed only after the success of database writes?		

Can files be deleted or moved from the source folder?		
Do any other processes share the files from the file system?		

18. Document record processing SLAs

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Processing latency to process each file record and the total file?		
What file system SLAs restrict how fast or how much data can be read from the file system?		

Document the data model for records read from the input files

19. Document the data structure of source files (such as CVS, flat files, hierarchical objects).

    Note: For this exercise you can quickly make something up now, then return to this section later as you refine your design.

    - Document the structure of the files that are being read in.
    - Document elements and attributes from the source files, including the data type, max/min ranges, or other data formatting restrictions, and if they are required or not.

    - Document any extension element beyond industry standards, such as custom headers or fault schemas for web services interfaces.

    - Document allowed values restrictions for the documented elements and attributes.

    - Document file system operations and their allowed values.

- Provide any sample documents to support this design.

Decide and document assumptions regarding how the processed records are integrated with the database

20. Document how tightly or loosely coupled the integration is between read files and changes to the database.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Will new, modified, or deleted records be immediately synchronized with the database, or will there be a middle layer?		
Will processed records be shared with other applications or endpoints?		

21. Document logging and auditing SLAs.

SLA and requirements	Assumptions, agreements, and requirements	Comments
What needs to be logged for each processed record?		
How are failures logged?		
How are retry attempts logged?		

Decide and document assumptions regarding SLAs for the target (database) system in the use case

22. Document latency SLAs affecting writing to the database.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Latency from after a file is processed until it is written to the database		

What database SLAs restrict how fast or how much data can be written to the database?		

23. Document latency SLAs affecting the rate and volume of data written to the database.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
Must data be written to the database in real time or near real time?		
If not, at what rate are database updates scheduled		

24. Document scheduling SLAs (if any).

SLA related question	**Assumptions, agreements, and requirements**	**Trade-offs and other comments**
Will schedule data processing be at a fixed rate, or on specific dates (cron jobs)?	Will schedule data processing be at a fixed rate, or on specific dates (cron jobs)?	
What is the average number of files and file size per scheduling cycle?		
What is the worst case (upper bound) number of files and file size per scheduling cycle?		
If data processing is not scheduled, what is the maximum burst of files and file sizes the solution should support?		

Analyze data processing requirements and trade-offs for the use case

25. Decide how the processing model can change based on changes to these competing data processing SLAs.

SLA related question	Assumptions, agreements, and requirements	Trade-offs and other comments
What are the average and maximum throughput rates and how do they affect the processing model?		
What are the data processing latency SLAs and how to they affect the processing model?		
How is data written to the DB?		
How is flow processing triggered?		
Must the File reading option change to support real time or near real time latency SLAs?		
Does the number or size of records read impact how records are processed?		

How can the data synchronization between the system be limited to only new or modified records?		

Compare trade-offs between the two design prototypes

26. Assess pros and cons of each solution design.

Batch Scope	Batch scope	For Each	For Each with a JMS dead letter queue
Audit and tracing per record	Yes (Pro)		
Auditing and tracing is easier and more open with a JMS topic	N/A		Yes (Pro)
Easier and isolated error handling of failed inserts to the target DB	No		Pro
Uses internal queues (may cause out of memory errors with large payloads and high throughput)	Con		

Design a Mule event processing models for a file transfer use case based upon your analysis

27. Use the previous analysis to decide the Mule event processing models for a file transfer use case.

- How are files read into the Mule app, and what options should be configured for the Mule flows and Mule components?

- After reading in a file, how are records from each file processed?

- Will records be processed sequentially, parallel, or in batch?

- How can the memory footprint be reduced while processing files?

- How can the memory footprint be reduced while writing to the target database?

- How are database connections handled between writes to the database, between file reads, and between processing each file record?

- How are failed records managed?

- Are transactions used?

Prototype the integration solution

28. In Anypoint Studio, add connectors and components to the Mule project to write to the database, based on the decisions made in the previous steps.
29. Identify scopes to use in the integration solution.
30. Use the sketch of the integration solution to help identify interdependencies between components in the Mule application, and suggestions to improve/change the design.

Related components	Independent components	Ways to improve/change the design	Impact of the design change

Identify error handling for the solution

31. In Anypoint Studio, add error handling scopes and components to your flows.
32. Which scopes and components did you add, and why?
33. Iterate on the previous design analysis to decide on error handling options; document your decisions here:

Identify and explain every factor that helps evaluate and agree upon the best processing model

34. Document the factors specific to your use case that are involved in choosing between Mule event processing models.

Justify each Mule event processing model decision based on the identified factors

35. Give reasons for your choice of Mule event processing models based on the identified factors.

Verify requirements

36. Review the beginning of the exercise against your design and verify all the requirements are met.

Answer these reflection questions

37. What is the difference between on error continue vs. on error propagate?
38. Why might first successful be more suitable compared to a try scope?
39. What are the trade-offs to process collections of elements/records with a For Each scope vs. a Batch Job vs. all at once with a single bulk operation?

40. What are the trade-offs of using bulk operations inside a Batch Job vs. all at once?

41. What are the trade-offs of using a single bulk operation vs. processing each record in a For Each scope?

42. Is parallel processing (such as in a Scatter-Gather) always faster?

43. What external factors might limit parallel processing?

44. What other trade-offs need to be considered to decide between sync, async, and parallel processing?

45. Which use cases are well suited to bulk operations, and what external factors might affect this decision?

46. When is DataWeave a good fit for data transformation?

47. Is DataWeave more or less useful when used for CDM mapping?

48. What other data transformation options are useful to your Mule applications, and what are the trade-offs?

49. What are the pros and cons of using XSLT vs. DataWeave, and what use cases are especially well suited for XSLT?

50. What are the pros and cons of using custom Java code vs. DataWeave, and what use cases are especially well suited for custom Java code?

51. What are the pros and cons of using an external service vs. DataWeave, and what use cases are especially well suited for custom Java code?

52. What caching considerations would relate to using external mapping services?

53. What evidence can you gather to validate your assumptions about these pros and cons?

54. What are the trade-offs of using a validation module vs. Choice routers vs. DataWeave code vs. custom code?

55. How do validation modules relate to error handling components?

56. What are the trade-offs of throwing and handling an error in the same flow?

57. How does APIkit handle invalid requests?

58. What are the trade-offs of using well defined schema in your flows to "fail fast"?

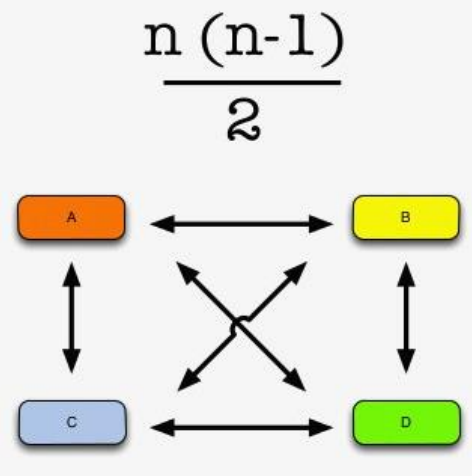
59. Where specifically would well defined schema be helpful in flows, and what are the trade-offs?

60. How do these trade-offs relate to decisions to use common data models?

References

- Enterprise Integration Patterns, Hohpe & Woolf, 2003
    - Describes 65 patterns for Enterprise Application Integration (EAI) and Message Oriented Middleware (MOM)
    - The Mule runtime has implemented many of these patterns: http://www.eaipatterns.com

Module 5: Choosing Appropriate Message Transformation and Routing Patterns



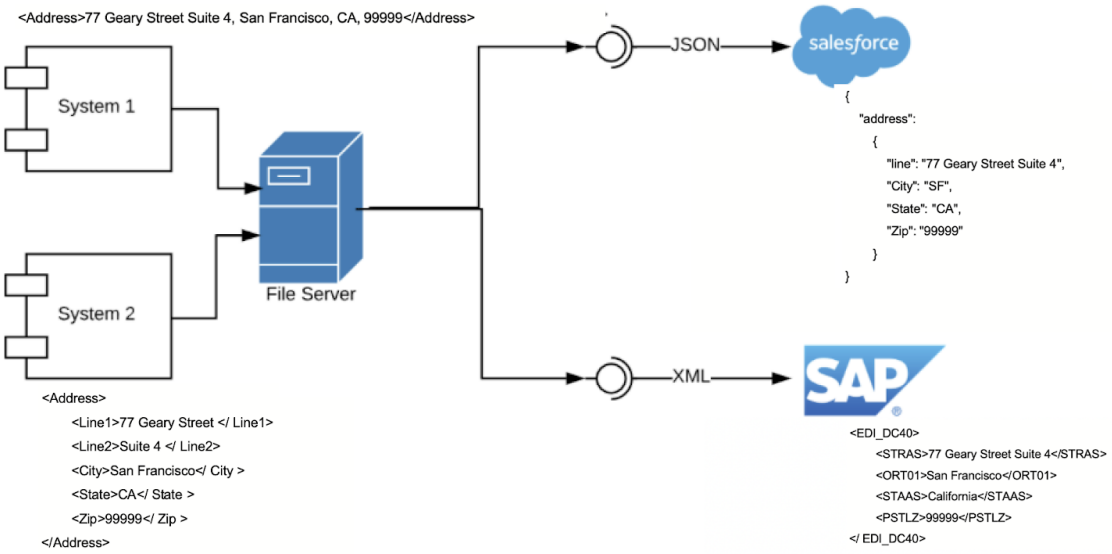
At the end of this module, you should be able to:

- Identify reusable ways to transform and process events.
- Simplify and decouple complex data mappings using common data models.
- Transform between data models.
- Apply the best event transformation, data validation, and event routing patterns to an integration use case.

Exercise 5-1: Apply message transformation and routing patterns to a systems integration use case

In this exercise, you design message transformation and routing between inbound and outbound systems. You will:

- Design integration between multiple inbound and outbound systems.
- Apply enterprise integration design patterns efficiently and effectively to meet requirements of a specific scenario.



Review the exercise context

1. Refer to the architecture diagram in the exercise introduction and read the requirements for the enterprise integration use case:
    - Two source systems System 1 and System 2 send files with the same kind of information to a shared file server.
    - System 1 and System 2 both write XML files, but each system uses a different data model to represent an Address object.
    - Data from both systems must also be sent to two target systems: an SAP and a Salesforce system.
    - The target systems each use different data models for Address objects.

Decide which data transformation processes are required or not

2. Document the source and target data schema differences.

Transformation Details	System 1 -> Salesforce target	System 2 -> SAP target
Schema type transformation	XML to JSON	XML to XML
Schema key transformations	<address> -> {AddressLine, City, State, Zip}	Line1 + Line2 -> STRAS City -> ORT01 State (CA) -> STAAS (California) Zip -> PSTLZ
Data transformation	State formatting converts from the state abbreviation to the full name: CA -> California	State formatting converts from the state abbreviation to the full name: State: CA -> STAAS: California

3. Document all required data transformations.

Target Source	System 1	System 2	Salesforce	SAP
System 1			XML1 -> JSON	XML1 -> XML
System 2			XML2 -> JSON	XML2 -> XML

Analyze the impact on the data model of future changes to the enterprise integration solution

4. What type of transformations are needed to add an additional target database?

Target Source	System 1	System 2	Salesforce	SAP	Database
System 1			XML1 -> JSON	XML1 -> XML	XML1 -> DB
System 2			XML2 -> JSON	XML2 -> XML	XML2 -> DB

5. How many transformations are needed, and how many are added to add this one new target?

6. What type of transformations are needed to then add an additional source System 3 that uses a different Address schema than the other two system.

Target Source	System 1	System 2	Salesforce	SAP
System 1			XML1 -> JSON	XML1 -> XML
System 2			XML2 -> JSON	XML2 -> XML
System 3				

7. How many transformations are required between three input source systems System 1, System 2, and System 3 and the three output target systems Salesforce and SAP, and how many additional transformations must be created to add the one new source system?

8. From your analysis, if there are S source systems and T target systems, how many data transformations are needed?

9. How many data transformations need to be changed if System 1 needs to change its schema?

10. How many data transformations need to be changed if the SAP system needs to change its schema?

Analyze how a CDM across systems could be implemented.

11. Fill in the table to see the transformations needed to add a common CDM to the original enterprise integration solution; a common CDM means that all the source and target systems are compatible with the CDM.

Target Source	System 1	System 2	CDM	Salesforce	SAP
System 1			XML1 -> CDM		
System 2			XML2 -> CDM		
CDM				CDM -> JSON	CDM -> XML

12. How many different data transformations are needed to support the CDM?

13. How does this number of data transformations compare with the tightly coupled solution?

14. Fill in the table to see the how many new transformations are needed to add a new source System 3 with a different XML schema from the other two source systems.

Target Source	System 1	System 2	CDM	Salesforce	SAP
System 1			XML1 -> CDM		
System 2			XML2 -> CDM		
System 3					
				CDM -> JSON	CDM -> XML

15. How many different data transformations are needed to support the CDM?

16. How does this number of data transformations compare with the tightly coupled solution?

17. Fill in the table to see the how many new transformations are needed to add a new target database.

Target Source	System 1	System 2	CDM	Salesforce	SAP	Database
System 1			XML1 -> CDM			
System 2			XML2 -> CDM			
System 3			XML3 -> CDM			
CDM				CDM -> JSON	CDM -> XML	

18. How many different data transformations are needed to support the CDM?

19. How does this number of data transformations compare with the tightly coupled solution?

20. From your analysis, how many different data transformations are needed using a CDM if there are S source systems, T target systems, and one CDM for all the systems?

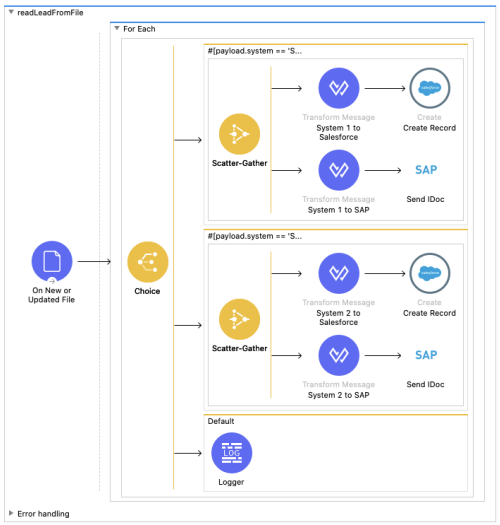
Decide ways to reduce multiple transformations

21. Design a reasonable CDM that combines all the source and target data transformations.

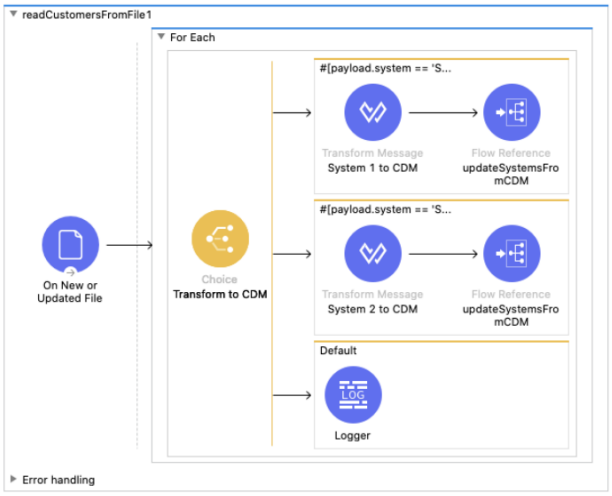
- Using a CDM, how many systems are affected if System 1 needs to change it's schema?
- Using a CDM, how many systems are affected if the SAP schema needs to change it's schema?
- What additional planning is required to make the CDM stable as new source and target systems are added?
- What is the impact on the original systems if the CDM needs to be updated?
- Are there additional layers that might make the existing CDM more stable as future systems are added?

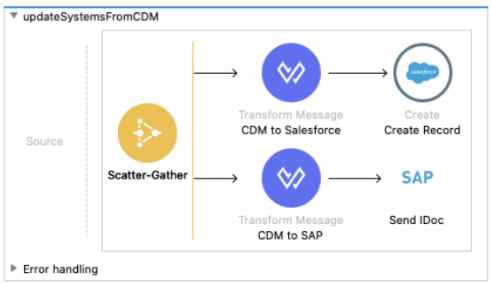
Design Mule integration flow(s) that use a CDM for this scenario

22. In Anypoint Studio, create skeleton flow to further refine your sketches of the Mule flow without a CDM.



23. In Anypoint Studio, create skeleton flows to further refine your sketches of the Mule flows with a CDM.





24. How might this be implemented in a 3 layered API-led connectivity approach? (E.g. What would be in the Experience, Process and System API layers?)

Answer these reflection questions:

25. How would the Mule 4 target Variable concept help with aggregation of data made by multiple calls?

26. How can data be joined between previous event processors, and what are the trade-offs?

27. How does data volume, request volume, or latency concerns affect these trade-offs and choices?

28. What are the overall pros and cons of creating a CDM?

29. What are some scenarios where a common data model (CDM) is an obvious fit?

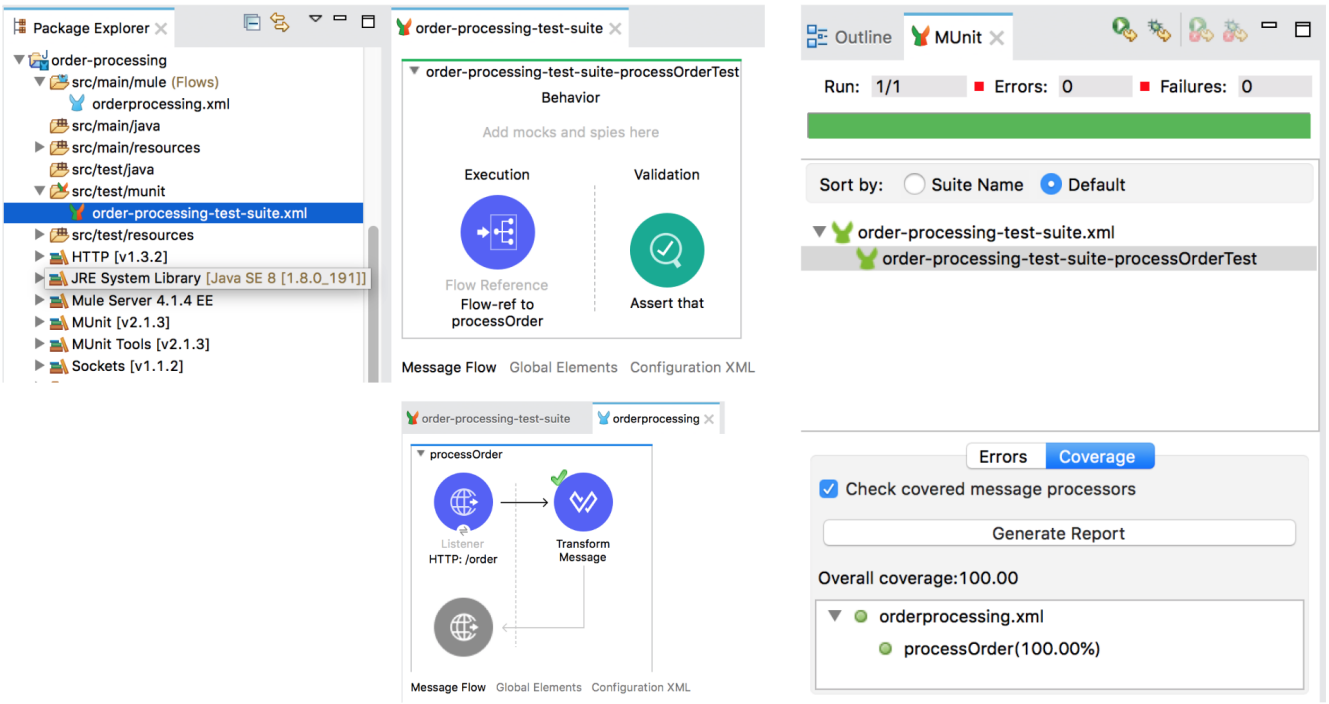
30. What are some scenarios where a CDM is not a good fit?

31. As what project or organizational scope(s) are CDMs typically successfully applied?

32. At what project or organizational scope(s) are CDMs often a hinderance or liability?

33. What are the trade-offs to building one unified CDM for the entire enterprise (such as a Master Data Management effort)?

Module 6: Designing Mule Application Testing Strategies



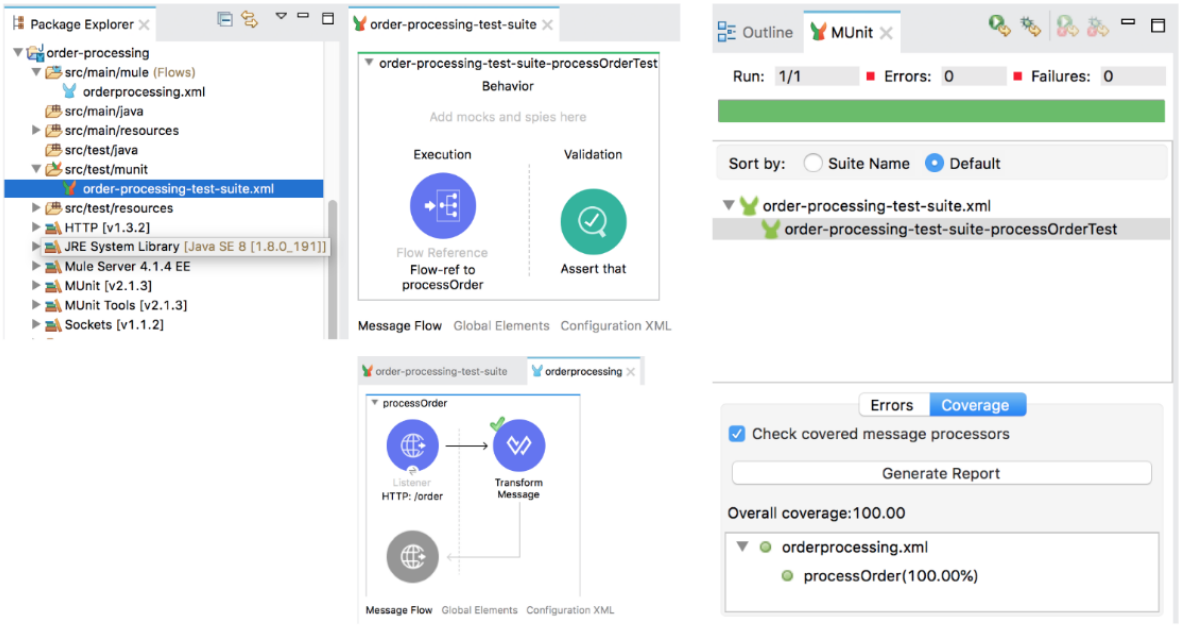
At the end of this module, you should be able to:

- Understand why testing is important in an integration project.
- Identify the MuleSoft provided testing framework and tools for testing during the software development lifecycle.
- Design testing strategies for a Mule application.
- Define test coverage for flows in a Mule application.

Walkthrough 6-1: Define a testing strategy for a use case

In this walkthrough, you add testing and code coverage to some Mule application flows. You will:

- Define MUnit test suites for Mule applications.



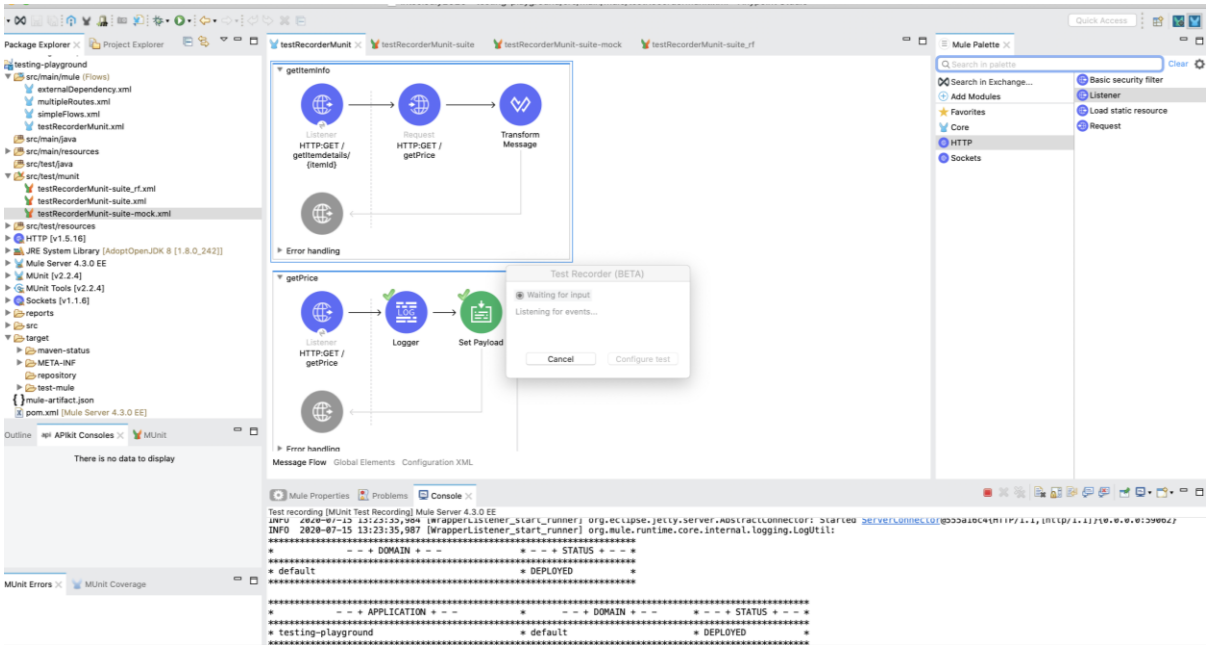
Define MUnit tests and code coverage

1. Add MUnit tests to some of your prototype Mule projects.
2. Develop tests for various scenarios:
    - Test for equality.
    - Test for errors.
    - Test when there are multiple routes or paths.
    - Test or mock external calls into a flow.
    - Test error handling.

Walkthrough 6-2: Generate MUnit tests with MUnit Test Recorder

In this walkthrough, you automatically generate MUnit tests using the MUnit Test Recorder for some Mule application flows. You will:

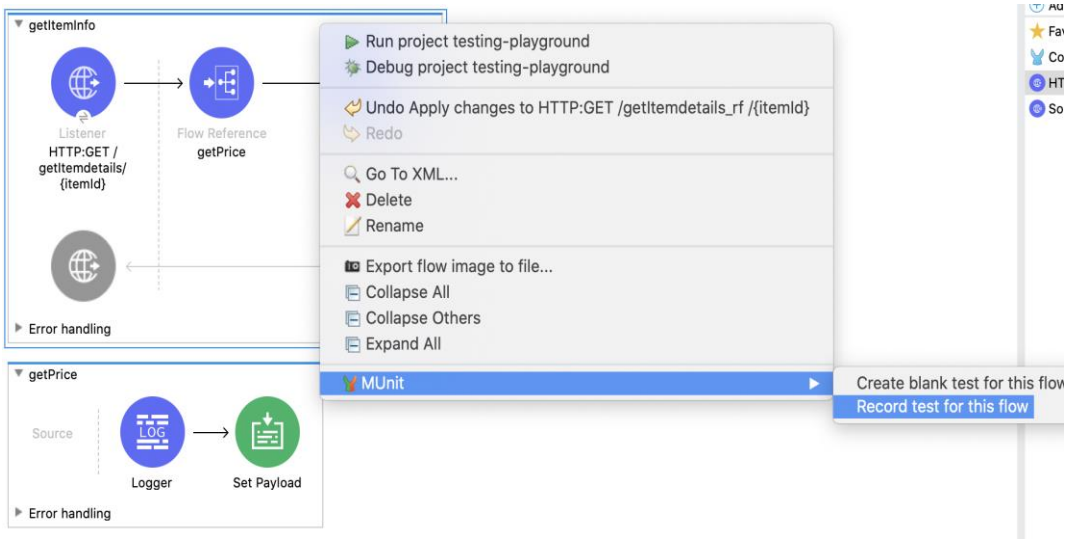
- Generate an MUnit test using MUnit Test Recorder.
- Run a MUnit test.
- Mock flow invocations that depend on other APIs or flows.



Generate an MUnit test using MUnit Test Recorder

1. In Anypoint Studio, import the Mule project testing-playground from the following location: $APAINTSOL_HOME/studentFiles/mod06-design-testing/starter_resources/testing-playground.jar
2. Wait for the application to fully load and then open testRecorderMUnit.xml.

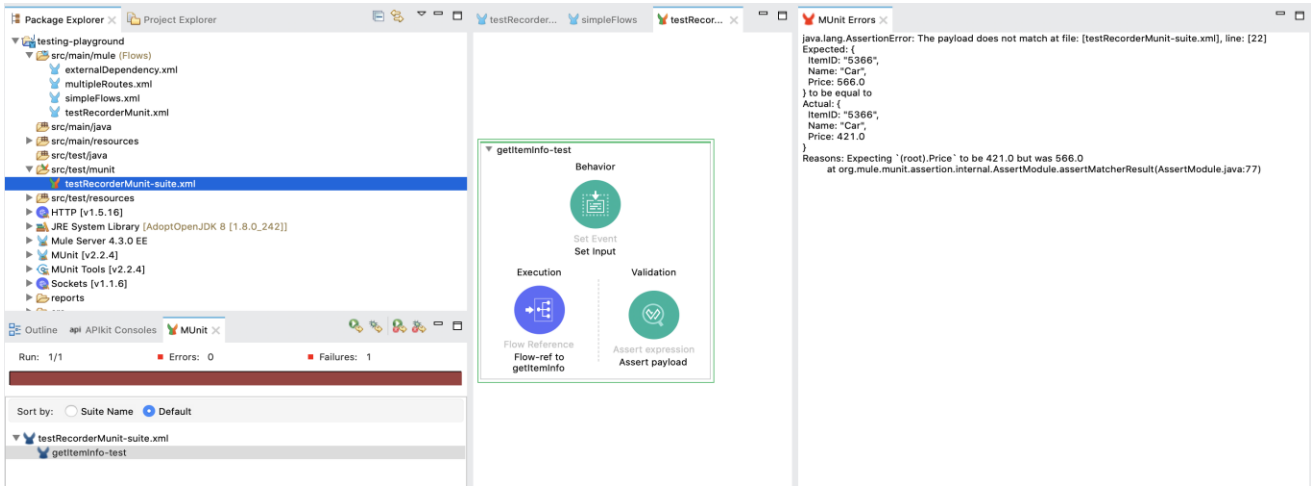
3. Select the getItemInfo flow from testRecorderMunit.xml, then right-click and select MUnit > Record test for this flow.



4. In a web client such as Advanced REST Client or any web browser, send a GET request to: http://localhost:8081/getItemdetails/5366
5. Return to Anypoint Studio and in the Test Recorder pop-up, select configure test and do the following:
    - Provide a file name and a test name, and then click next.
    - Observe the configure test window, do not change any of the configurations, and click next.
    - Observe the summary of the test that will be automatically generated for the flow, and click finish.
6. Inspect the generated MUnit test added to testRecorderMunit-suite.xml, and review the Behavior, Execution, and Validation scopes of the test; be sure to pay close attention to the type of assertion and the input/output test data configuration.
7. Review the src/test/resources directory and locate the newly created package named after the generated test.

Run the MUnit test

8. Right-click the getItemInfo-test MUnit flow and run MUnit suite; MUnit should fail with an AssertionError that the Payload does not match.

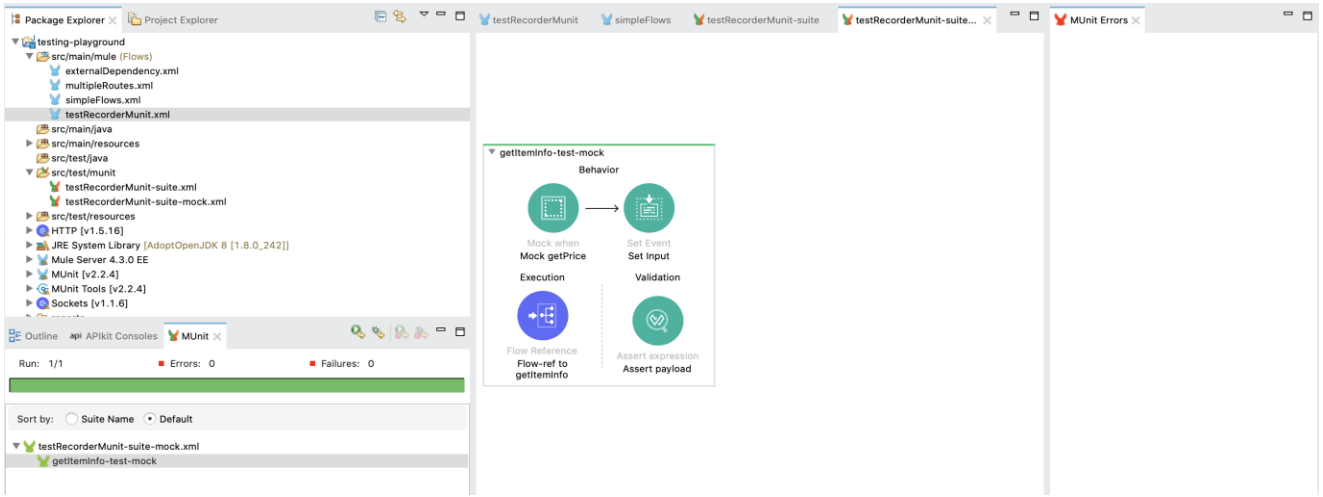


Note: The test fails because of the difference in the response for the getPrice flow; the price value is changed every time the flow is invoked. In order to fix this issue, you can mock the response of the getPrice flow.

Mock flow invocations that depend on other APIs or flows

9. Select the getItemInfo flow, then right-click and select MUnit > Record test for this flow.
10. In a web client such as Advanced REST Client or any web browser, send a GET request to: http://localhost:8081/getItemdetails/5366
11. Return to Anypoint Studio and in the Test Recorder pop-up, select configure test and do the following:
    - Set the file name to testRecorderMunit-suite-mock.xml, the test name to getItemInfo-test-mock, and then click next.
    - In the configure test window, select the getPrice flow reference, select mock this processor. and click next.
    - Observe the summary of the test that will be automatically generated for the flow, and click finish.

12. Right-click the getItemInfo-test-mock MUnit flow in testRecorderMunit.xml and run MUnit suite; the MUnit test should now run successfully without any errors.



Answer these reflection questions

13. What are some examples where unit testing is helpful, and how would you implement unit tests?

14. What are the general features of a unit test of the following scenarios:

    - An HTTP Request?
    - A database select vs. a database insert, update, or delete operation?
    - A DataWeave transformation?
    - A Scatter-Gather component?
    - A For Each scope and its components?

15. Custom Java code?

16. What network considerations might affect performance testing?

17. What choice of tools might affect performance testing?

18. What are the trade-offs of meeting various performance goals, and what other goals?

19. What might conflict with performance goals?

20. What are some examples where unit testing is helpful, and how would you implement unit tests?

21. How are integration tests of a flow different from unit tests?

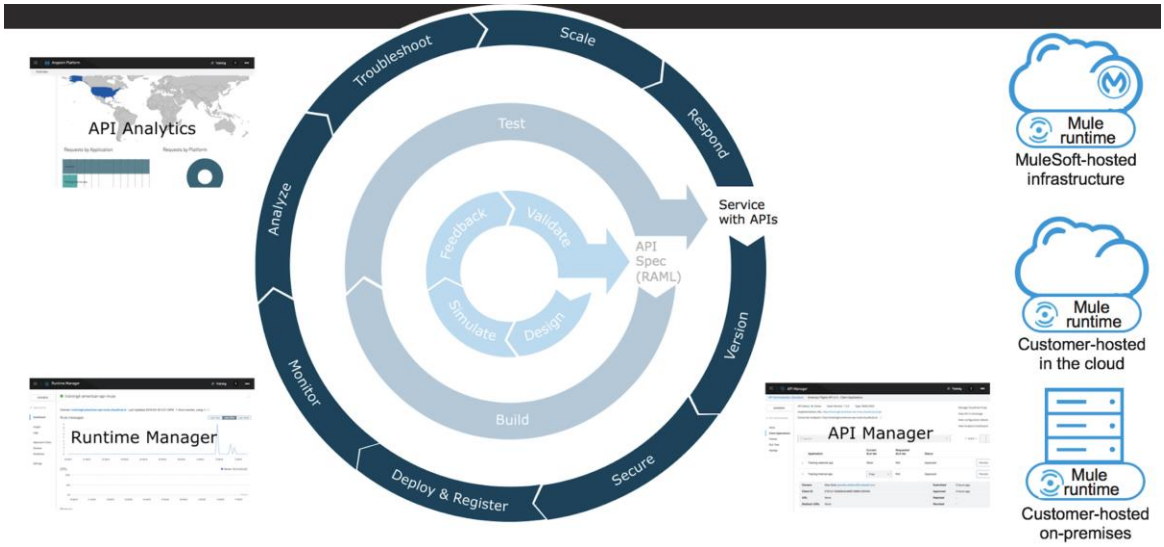
22. How is source and target data retrieved vs. mocked for these types of tests?

23. What factors affect or limit the effectiveness of an integration test?

24. What types of performance tests are common for Mule flows?

25. Should testing use average load, worst case load, or some other model?

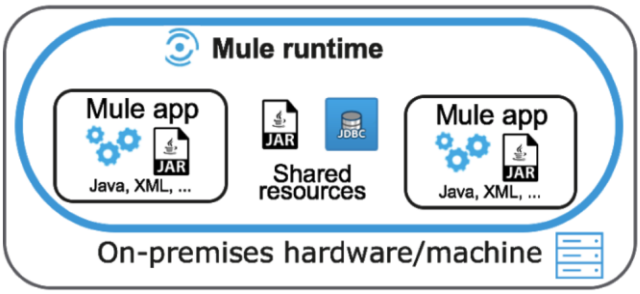
PART 2: Operationalizing Integration Solutions

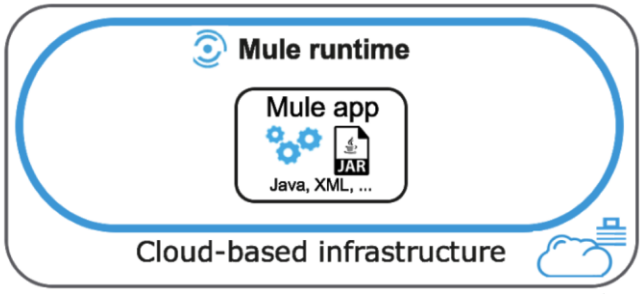


At the end of this part, you should be able to:

- Decide and develop appropriate and optimal deployment strategies.
- Decide the best ways to preserve and manage Mule application state.
- Design effective and sufficient logging and monitoring strategies.
- Create an efficient and automated software development lifecycle (SDLC).

Module 7: Deciding upon and Developing a Deployment Strategy





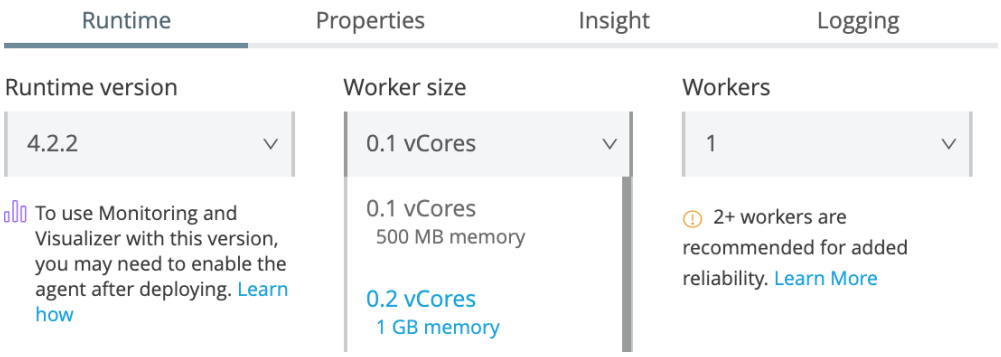
At the end of this module, you should be able to:

- Identify the service and deployment models supported by Anypoint Platform.
- Decide deployment options for various scenarios.
- Design containerized deployments for Mule runtimes.

Walkthrough 7-1: Scale a Mule application deployed to the MuleSoft-hosted runtime plane

In this walkthrough exercise, you deploy a Mule application to several CloudHub workers. You will:

- Scale a Mule application vertically by changing the CloudHub worker size of a CloudHub-deployed Mule application.
- Scale a Mule application horizontally by changing the number of CloudHub workers assigned to CloudHub-deployed Mule application.
- Re-deploy a CloudHub-deployed Mule application with zero downtime.



Use Runtime Manager to scale a Mule application deployed to CloudHub

1. Return to Runtime Manager and navigate a CloudHub deployed Mule application such as hello-world, mod03-apikit, or the order-proxy deployment.

   Note: You deployed this Mule application earlier in the class.

2. Leave the worker size as 0.1vCores.
3. If you are using a training account, change the workers from 1 to 2.
4. View the logs while the new worker is starting.
5. Wait for the status to change to Started.

Update the deployed Mule application and verify it redeploys with zero downtime

6. Restart the Mule application again.
7. From a web client, submit requests to the Mule application while it is restarting and verify you get responses.
8. View the log files and verify the Mule application updates with zero downtime.

9. Look in the log files and verify web client requests are load balanced across both CloudHub workers.

Answer these reflection questions:

10. What is the difference between a control plane and a runtime plane?

11. What are the components of the MuleSoft-hosted control plane?

12. What use cases would prohibit using a MuleSoft-hosted control plane?

13. What use cases would use a combination of control planes, and what would be the trade-offs and benefits?

14. For what use cases is the MuleSoft-hosted control plane the best option?

15. For what use cases is a customer-hosted control plane the best option?

16. What components make up the MuleSoft-hosted runtime planes?

17. What components make up various customer-hosted runtime planes?

18. What use cases would prohibit using a MuleSoft-hosted runtime plan

19. What are the trade-offs of these runtime plane options?

20. What use cases would benefit from a hybrid collection of runtime planes?

21. What are the trade-offs in using a hybrid model?

22. How could a hybrid model help an enterprise move to the Cloud?

23. How do customer-hosted Mule runtimes compare with an Anypoint Runtime Fabric runtime plane?

24. Which scenarios are better suited for Anypoint Runtime Fabric, and which are better suited for customer-hosted Mule runtimes?

25. Which control planes can control customer-hosted Mule runtimes vs. Anypoint Runtime Fabric?

26. What control plane is used by Runtime Fabric?

27. How do Runtime Fabric's runtime plane features and behaviors compare with the MuleSoft-hosted runtime plane?

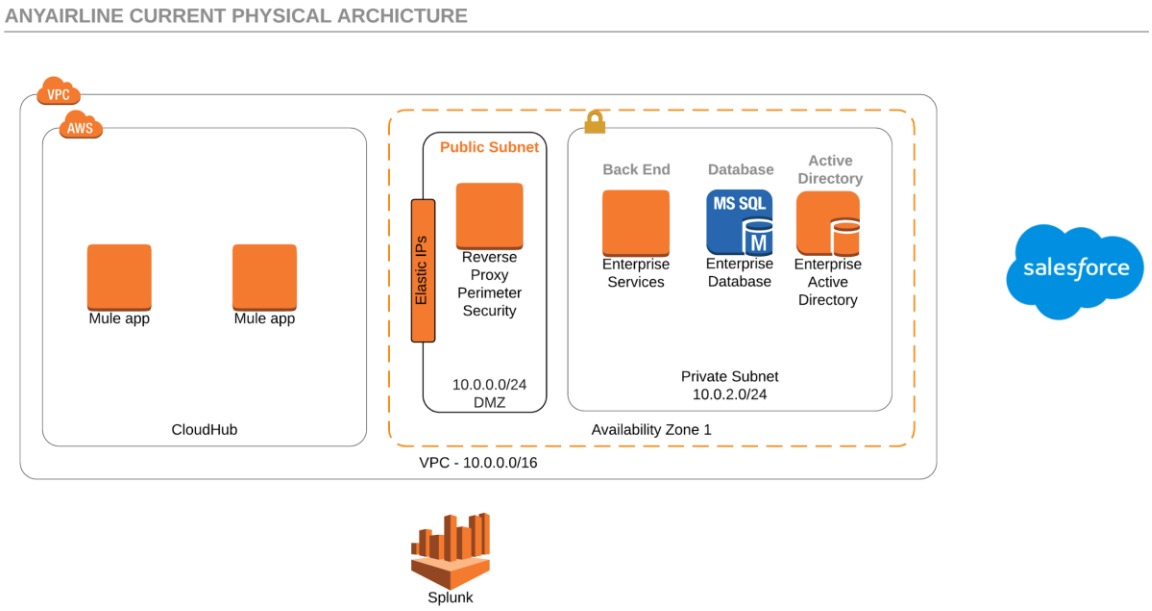
28. How do Runtime Fabric's runtime plane features and behaviors compare with other customer-hosted runtime plane options?

29. For which use cases is Runtime Fabric the best choice of runtime plane?

Exercise 7-2: Identify the best deployment model for a specific scenario

In this exercise, you analyze and compare deployment options for a scenario, then decide the best options. You will:

- Analyze deployment options for a specific scenario.
- Design an optimal service model and deployment model for a specific scenario based on functional and non-functional requirements.



Identify the exercise scenario and context

1. Read the exercise context and requirements:
    - Data integration should have minimal network latency.
    - Existing monitoring capabilities of Splunk should be utilized.
    - Existing HTTP-based services should be utilized for cloud-based Salesforce and Splunk systems.
    - The organization has strict data residency requirements and the organization infrastructure is PCI compliant.
    - Organization's DevOps has limited expertise in containerization.
    - The organization has decided to use MuleSoft Anypoint Platform as the integration platform for data integration between various systems.

- The Mule runtimes can use any available runtime plane that best meets the scenario's requirements.
- High availability and scaling goals need to be supported.
- Mule applications should be able to be scaled "horizontally" across multiple Mule runtimes, so the services are highly available.
- Should be possible to scale the Mule runtime's host infrastructure "vertically" without service interruption.

1. Review the competing functional and non-functional requirements of the organization:

- High availability (HA) and scalability options and SLAs.
- Accounting for network latency.
- Monitoring is important.
- Effective and safe networking and load balancing for endpoints.
- Secure data at rest and in transit.
- DevOps currently has limited containerization capability.

Compare and analyze deployment options across runtime planes

2. Compare deployment options for different runtime planes.

Control plane **Deployment option**	**CloudHub**	**Anypoint Runtime Fabric**	**Customer-hosted or Private Cloud Edition**	**Pivotal Cloud Foundry**
HA and Scalability				
Network latency				
Monitoring				
Networking				
Data security				
DevOps containerization capability				

Decide the best deployment model that might involve AWS autoscaling

3. Answer questions to decide the best deployment model based on the fact that AWS autoscaling can be achieved using deployment on EC2 instances under ELB control.

   - Can CloudHub, PCF, or Runtime Fabric be used to create these types of deployments
   - If not, what runtime plane choices remain?

   - What are the network latency requirements for this scenario?
   - What is the fastest way to connect an on-premises network to Mule runtimes running in EC2 virtual images?

Compare and analyze deployment options across control plane

4. Compare how various deployment related features are implemented in different control planes.

Control plane **Deployment option**	**CloudHub**	**Anypoint Runtime Fabric**	**Customer-hosted or Private Cloud Edition**	**Pivotal Cloud Foundry**
Scheduling				
CH enhanced logging				
Dashboard				
Insights				
Alerts				

Compare and analyze deployment options in various runtime planes

5. Compare what and how additional deployment features are implemented in different control planes.

Control plane Deployment option	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry	Control plane Deployment option
Auto scaling	Yes	Manual scaling	unless managed	unless managed	unless managed
Anypoint Monitoring					
Anypoint Visualizer					
Anypoint Edge Security					
Anypoint Tokenization					
Load balancing					
DLB support					
Anypoint MQ					
File persistence					

Analyze features and capabilities available to the scenario for various deployment options

6. Compare other deployment options and behaviors of various runtimes.

Control plane Deployment option	CloudHub	Anypoint Runtime Fabric	Customer-hosted or Private Cloud Edition	Pivotal Cloud Foundry
HA and Scalability	Preferred	Preferred	Preferred	Preferred
Network latency				
Monitoring				
Networking				
Data security				
DevOps containerization capability				

Discuss the options with the class and decide together on the best deployment option for the scenario

7. Which runtime and control planes can support AWS auto-scaling in EC2 instances under ELB control?

8. What is the best way to minimize network latency between on-premises systems and the Mule runtimes running in EC2 under ELB control?

Analyze ways to select the best runtime and control planes to meet the requirements of the scenario

9. Which runtime and control planes can support AWS auto-scaling in EC2 instances under ELB control?

10. What is the best way to minimize network latency between on-premises systems and the Mule runtimes running in EC2 under ELB control?

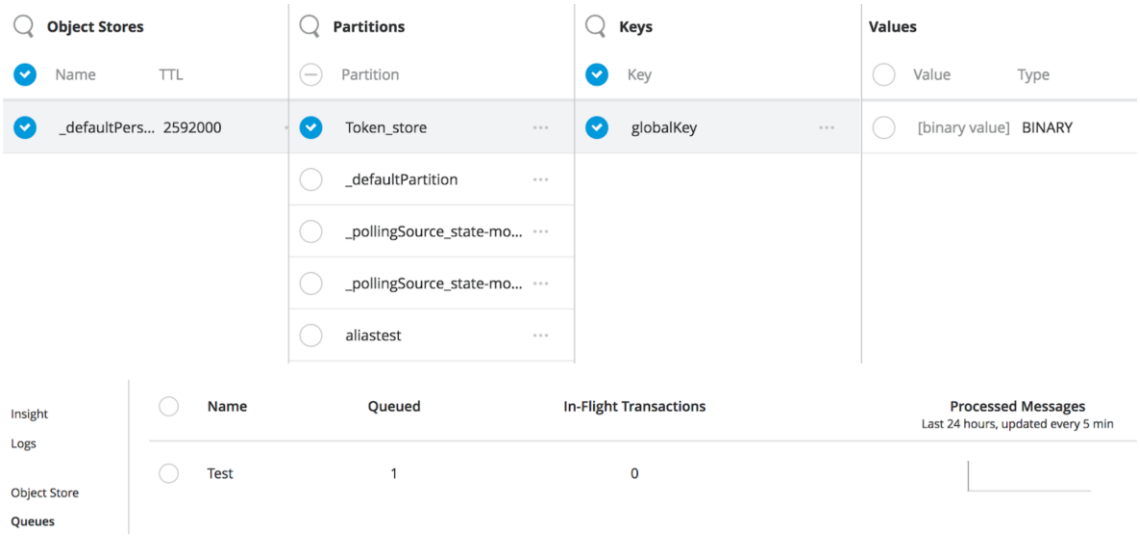
Decide the top two deployment options

11. Decide the best two deployment options for the organization and summarize how each deployment model best serves organization requirement.

12. After some discussion, apply your analysis to identify the best deployment model for the scenario which meets the requirements.

13. Add our deployment decisions to the GetAways architecture document.

Module 8: Designing with Appropriate State Preservation and Management Options



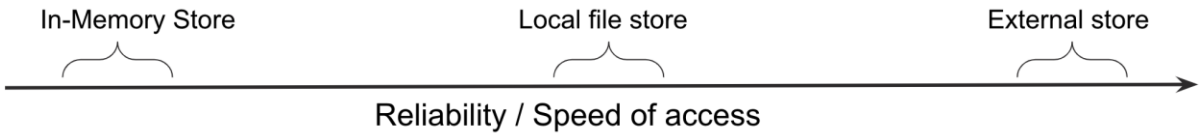
At the end of this module, you should be able to:

- Decide the best way to store Mule application state in persistent or non-persistent storage.
- Identify how to store Mule application state using the Object Store v2.
- Decide the best way to manage storage of Mule application state using persistent queues.
- Configure Mule application provided caches to store Mule application state.
- Avoid duplicate processing of previous records using Mule connector watermarks.

Exercise 8-1: Identify why and when Mule applications need to maintain state

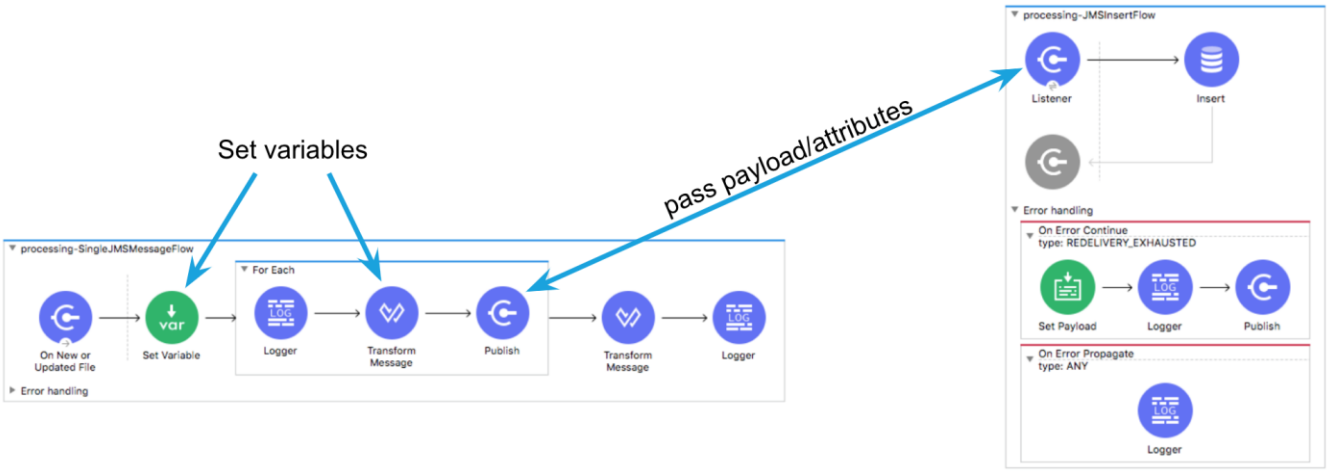
In this exercise, you analyze which scenarios can benefit from storing state and the associated trade-offs. You will:

- Identify some scenarios where a Mule application needs to maintain state.
- Analyze the trade-offs of various state storage options for various scenarios.



Review the exercise context slide for a polling scenario

1. Review the context of the polling scenario that might require state management options.



2. In which components might state management be needed?

Provide some example scenarios where state need to be maintained, and how is it maintained

3. Provide examples for the following scenarios and suggest state management options.

- Between Mule applications and non-Mule applications

- Between Mule applications

- Between flows

- Between HTTP requests

- Using schedulers

- Processing collections of records sequentially in order

- Processing collections of records out of order in a batch job

4. Analyze each of your previous scenarios to compare the trade-offs for each state management option.

Scenario	State storage option	Pros and other trade-offs	Cons and other trade-offs	Ranked preference for the usecase
Between Mule and non-Mule apps	Database	transactional is more reliable	transactional is slower	
	File	Persists data	Not transactional. Requires networked file share.	
	JMS server	Persistence vs. performance can be tuned by admins	Extra layer of infrastructure to support and buy	
	In memory data grid	Faster	Not as persistent. Data lost if entire grid goes down	
Between Mule apps				
Between flows	Object store			
	(See between apps options)			
	VM queues			
Between HTTP requests				
Using schedulers				
Sequential record processing				
Batch record processing				

Exercise 8-2: Design state management for a polling use case

In this exercise, you design state management options for a polling use case. You will:

- Identify ways to improve performance by storing and managing the state of Mule events in a flow.
- Identify ways to avoid duplicate data processing by storing and managing the state of Mule events in a flow.
- Decide when the state of response data should be stored.
- Decide when stored state data should be encrypted.
- Identify the best integration style for a scenario.

Runtime Plane	Object store v1	Object store v2	Persistent queue	File store	Cache	Watermark
Access Control						
Persistence lifespan						
Cluster Support						
Has REST API						

Iterate on the analysis of the last exercise to add in other state management options provided by MuleSoft

1. What are some additional state management options covered in the class after the last exercise?

   Watermarks, caches, VM queues, private object stores, Anypoint Object Store v2 , file persistence, in-memory data grid persistence, database backed persistence

2. Provide examples for the following scenarios and suggest state management options; to get started you can copy your answers from the previous exercise.

   - Between Mule applications and non-Mule applications

   - Between Mule applications

- Between flows

- Between HTTP requests

- Using schedulers

- Processing collections of records sequentially in order

- Processing collections of records out of order in a batch job

3. Analyze each of your previous scenarios to compare the trade-offs for each state management option, including additional state management options you listed in the previous step; you might want to copy your answers from the previous exercise to get started.

Scenario	State storage option	Pros and other trade-offs	Cons and other trade-offs	Ranked preference for the usecase
Between Mule and non-Mule apps	Database	transactional is more reliable	transactional is slower	
	File	Persists data	Not transactional. Requires networked file share.	
	JMS server	Persistence vs. performance can be tuned by admins	Extra layer of infrastructure to support and buy	
	In memory data grid	Faster	Not as persistent. Data lost if entire grid goes down	
Between Mule apps				
Between flows	Object store			
	(See between apps options)			
	VM queues			

Between HTTP requests				
Using schedulers				
Sequential record processing				
Batch record processing				

Effects of runtime plane choices on scenario benefits

4. Fill in the table to compare runtime plane choices for various scenarios and which choice is better.

Scenario	Runtime plane choice	Effects (benefits/trade-offs)
Between Mule applications		
Between flows		
Between HTTP requests		
Using schedulers		
Processing collections of records sequentially in order		
Processing collections of records out of order in a batch job		

Design state management for a polling use case

5. Compare state management options based on performance and reliability goals in the context of particular deployment model

State Management option Runtime plane deployment option	Object store v1	Object store v2	Persistent queue	File store	Cache	Watermark
Access Control						
Persistence lifespan						
Cluster Support						
Has REST API						

Decide the best state management option based on performance and reliability goals

6. Fill in this table to compare the use of MuleSoft-provided state management options with various runtime plane performance, reliability, or security options.

State management option Runtime plane options	Object Store v1	Object Store v2	Persistent Queue	File Store	Try scope	Watermark
Automated failover						Stores state after a failover
HA (multiple workers)						
Transactional						
Encryption						
Performance	For on-prem can be in memory or file/db backed	Slower for on-prem		Slower		
Replication						

Answer these reflection questions

7. How can you improve the delayed response time cause by the API taking more than 30 seconds to respond?

8. Is there a way to avoid calling the same API repeatedly, and what are the benefits of avoiding these repeated API calls?

9. Is state management needed, and if so, why?

10. How can state management help this use case?

11. If state management is needed, what Mule application state or objects should be stored (Mule event, request and response)

12. If you design to avoid repetitive API calls, then how should the Mule application send back its response?

Mock a flow to meet state management requirements

13. In Anypoint Studio, create flows to meet the state management requirements; describe the flows here.

14. Paste screenshots of your solution flows and configurations here.

Answer these reflection questions

15. What use cases should use a local cache?

16. What use cases should use a cache backed by an external store?

17. What types of external stores are possible, and how do you decide the best choice?

18. What use case would benefit from an intermediary API between caching operations in the Mule application and an external store?

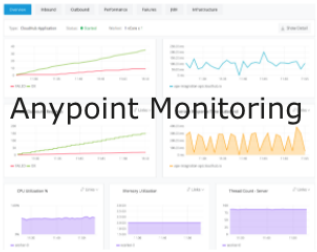
19. How does an On Table Row operation's watermark work?

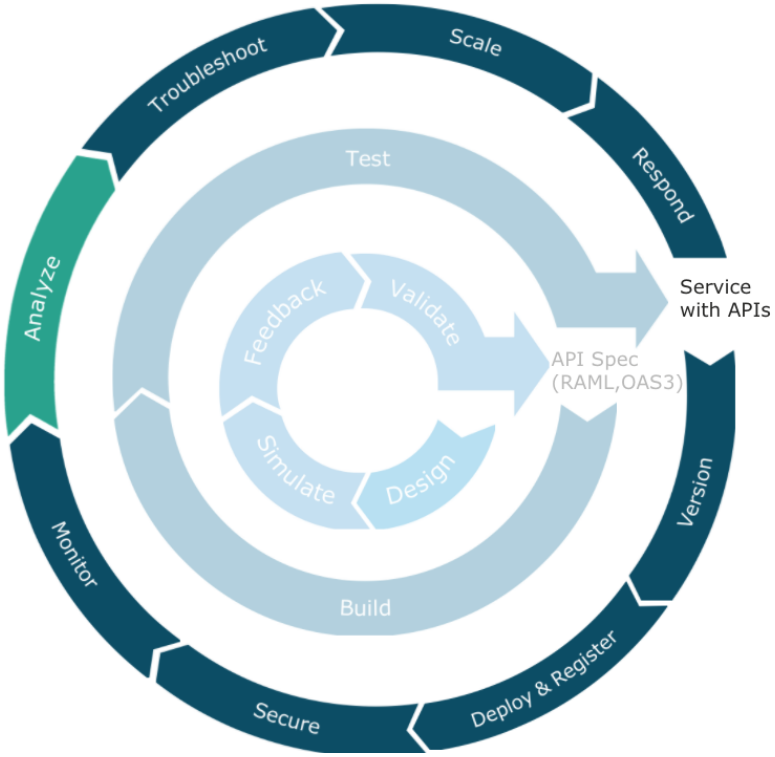
20. How does this feature compare with using a Scheduler and any other database operation?

21. How does an On New or Updated File operation's watermark work?

22. How does this feature compare with using a Scheduler and any other database operation?

Module 9: Designing Effective Logging and Monitoring





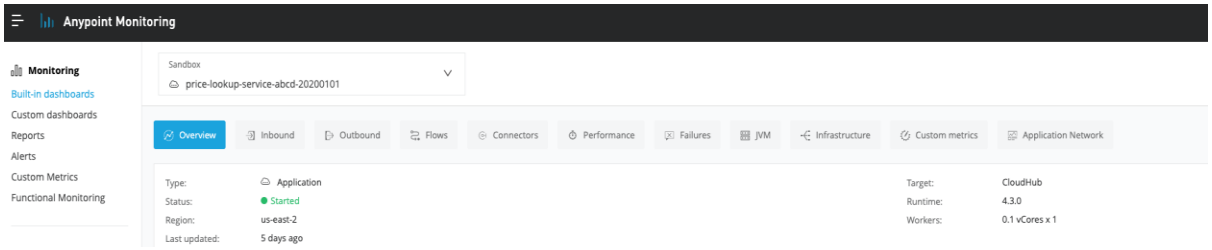
At the end of this module, you should be able to:

- Describe auditing and logging options for Mule application.
- Design logging strategies for Mule applications.
- Choose logging policies for Mule application log files.
- Describe integration options with third party log management system.
- Specify Anypoint Platform monitoring, alerting, notification, visualization, and reporting options for APIs and integration solutions.

Walkthrough 9-1: Explore the Anypoint Monitoring dashboard

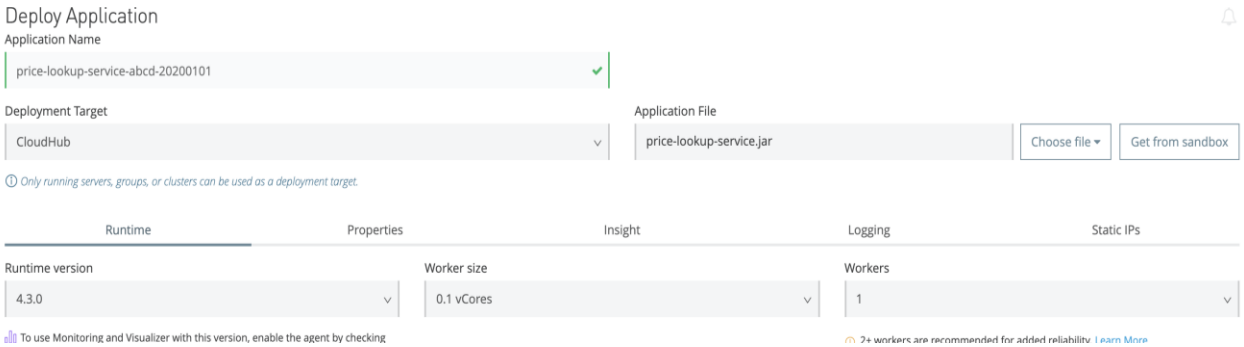
In this walkthrough, you use Anypoint Monitoring for monitoring a Mule application. You will:

- Configure Anypoint Monitoring to monitor the Mule application.
- Run a stress test against the Mule application.
- Observe the information produced by Anypoint Monitoring.



Deploy a Mule application to CloudHub

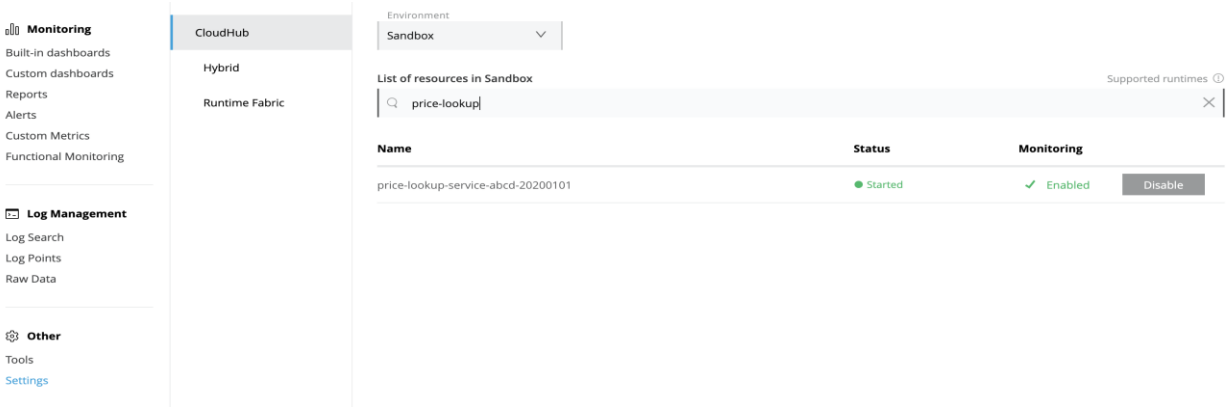
1. Return to Runtime Manager.
2. Switch to Sandbox environment.
3. Click the Deploy Application button.
4. Write down a unique suffix to use in the deployment application name (also called the domain) such as <<your initials>>-<<today's date>>, for example *abcd-20200101*.
5. In the Deploy Application page, enter the following information:
    - Application Name: price-lookup-service-*abcd-20200101*
    - Deployment Target: CloudHub
    - Application File: Select Choose File > Upload file and then browse to $APAINTSOL_HOME/studentFiles/mod09-design-logging-monitoring/starter_resources folder in your studentFiles and select the price-lookup-service.jar deployable archive.
    - Runtime version: Choose the latest Mule 4.3.0 Runtime version
    - Worker size: 0.1 vCores
    - Workers: 1



6. Click Deploy Application.

Use Anypoint Monitor to gather application performance statistics

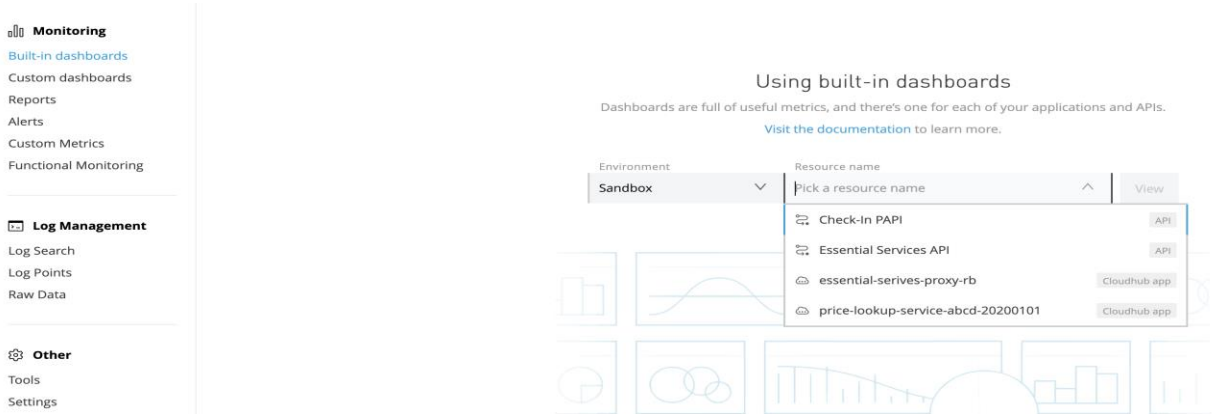
7. In the main menu, select Anypoint Monitoring.
8. In the left-side navigation, select Settings under Other.
9. In the center panel, select Sandbox environment.
10. Enable the price-lookup-service-*abcd-20200101.*
11. Wait for the application to be redeployed.



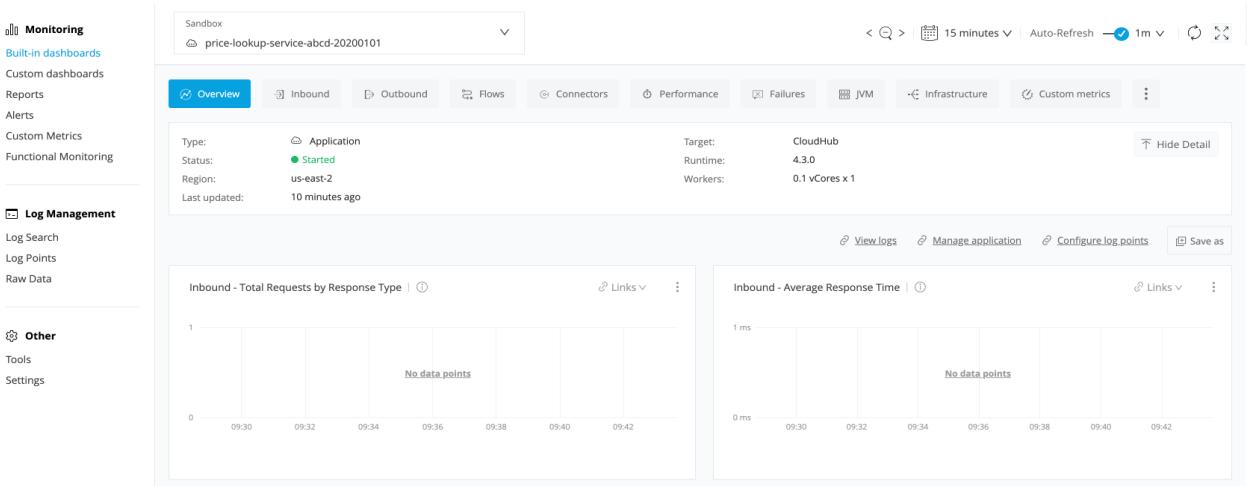
12. In the left-side navigation, select Built-in Dashboards.

    Note: To complete this walkthrough, you will need an Anypoint Platform account with Anypoint Monitoring enabled. If you do not have one, watch the instructor's demonstration of this walkthrough.

13. In the center panel, select Sandbox environment.



14. Select the price-lookup-service-*abcd-20200101* CloudHub application.
15. Observe the information displayed in the built-in dashboard.
16. Explore the dashboard by clicking various metrics tab - for example, Overview, Inbound, Outbound, Performance, Failures, JVM, Infrastructure.

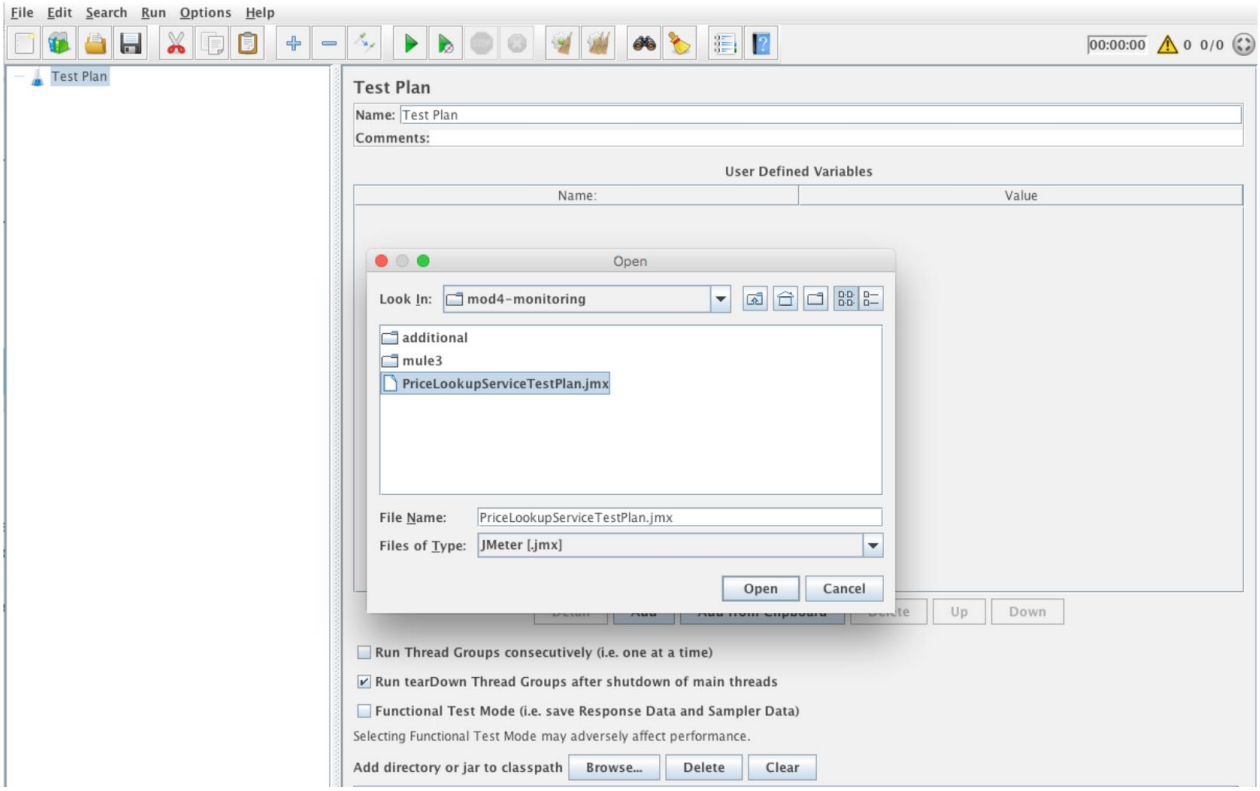


Use JMeter to generate load and observe the results

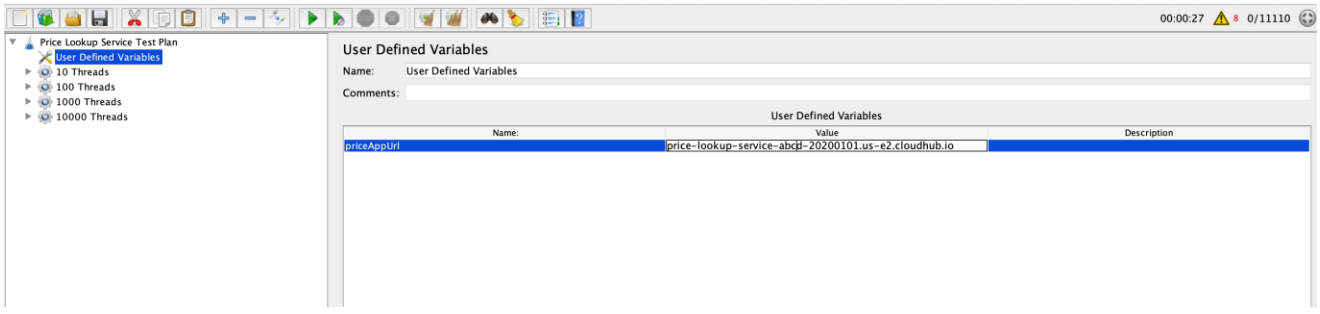
17. On your computer, navigate to the resources folder in your studentFiles.
18. Unzip apache-jmeter-x.x.zip file.
19. Navigate to the apache-jmeter-x.x\bin directory.
20. Start Apache JMeter using one of the following options:

    - Windows: From command line by executing apache-jmeter-x.x\bin\jmeter.bat
    - Linux/Unix: apache-jmeter-x.x/bin/jmeter

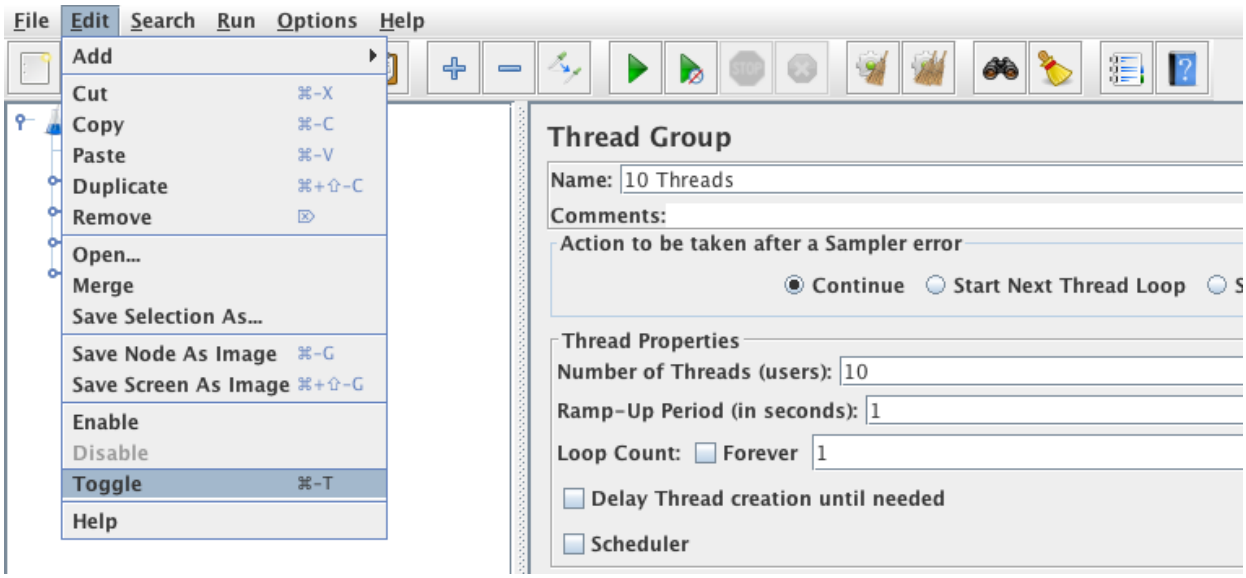
21. In JMeter, select the File menu and then click Open.
22. In the file chooser, navigate to the mod09-design-logging-monitoring folder in your studentFiles and select the PriceLookupServiceTestPlan.jmx file.



23. In the left-side navigation, select User Defined Variables.
24. In the center panel, set the value of the priceAppUrl to the app URL of the CloudHub deployment, for example price-lookup-service-*abcd20200101*.us-e2.cloudhub.io.



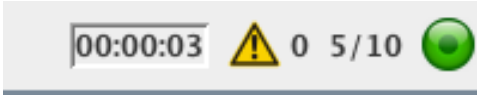
25. From the File menu, select Save.
26. In the left side, select 10 Threads.
27. In the Edit menu click Toggle.



Note: By default, all thread groups in this particular test plan are disabled to avoid any accidental loads on your application. Thread groups define the number of HTTP requests that will be sent to your application's endpoint. By enabling or disabling a thread group, you can send a given number of HTTP requests.

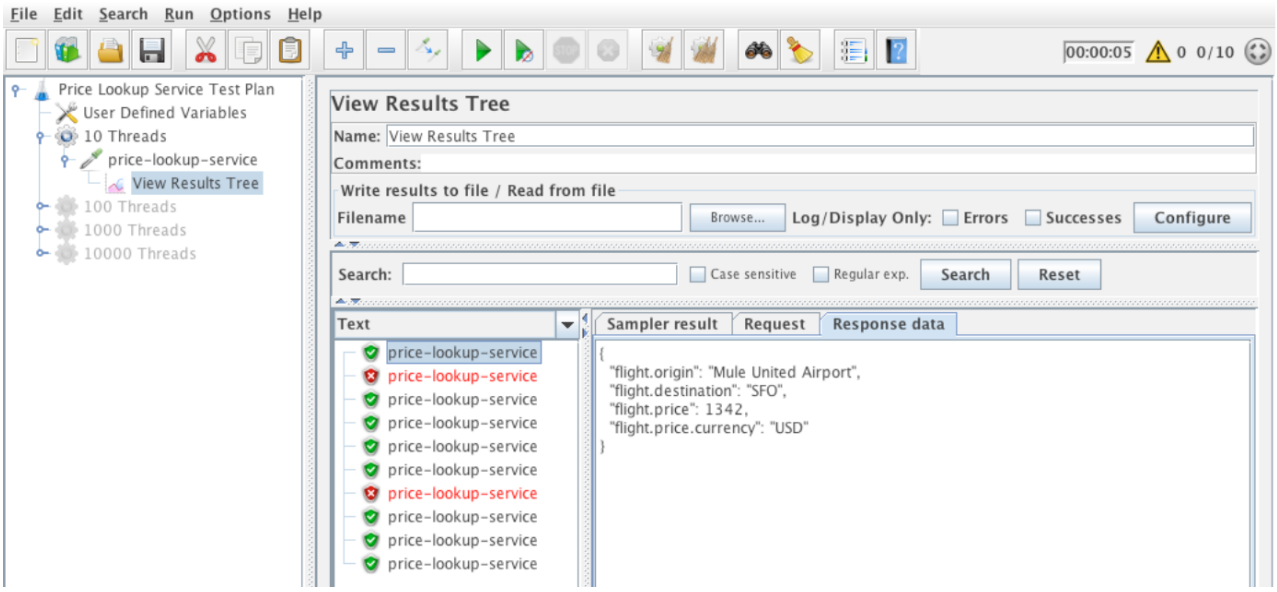
28. From the Run menu, select Start; this will execute the test plan with 10 threads.

    Note: The icon bar will display the progress of the test plan.



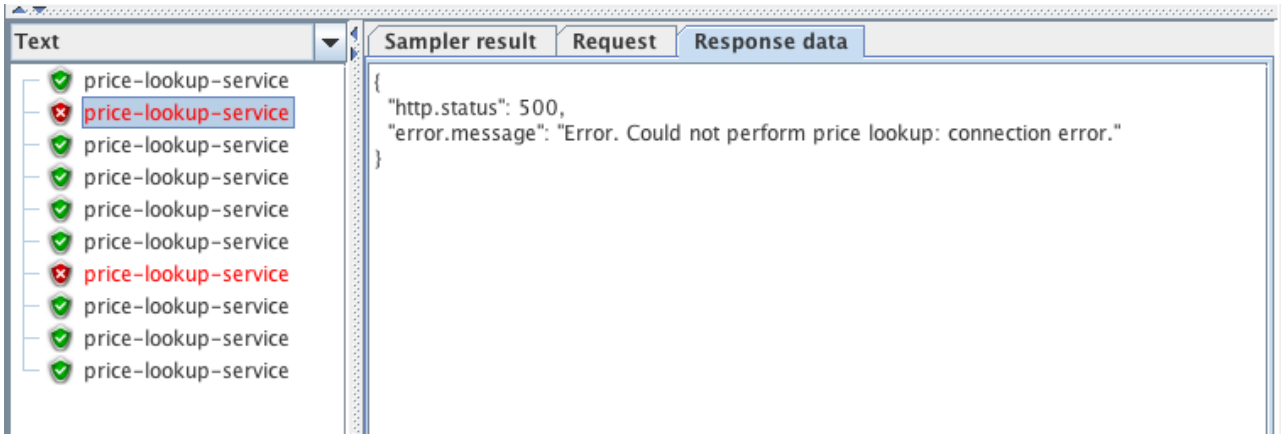
29. In the left-side navigation, under 10 Threads, expand the price-lookup-service, and select View Results Tree.
30. In the center section, select any one of the test results.

31. Click the Response data tab to view the actual response of the HTTP request.



Note: If you do not see any results, look at the logs in Runtime Manager for the price-lookup-service deployment and verify the GET requests from JMeter were received by the Mule application in CloudHub.

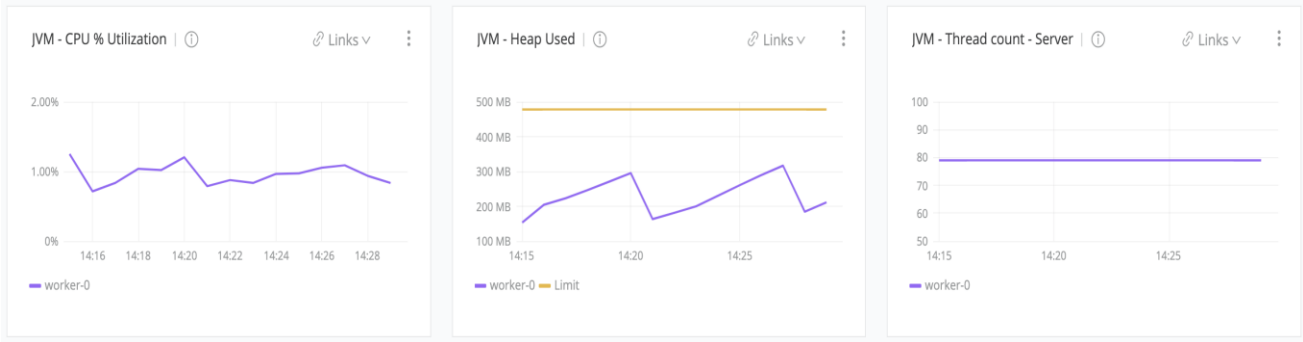
32. If there are any failed tests in red font, click the tab Response data to view the response of the failed HTTP request.



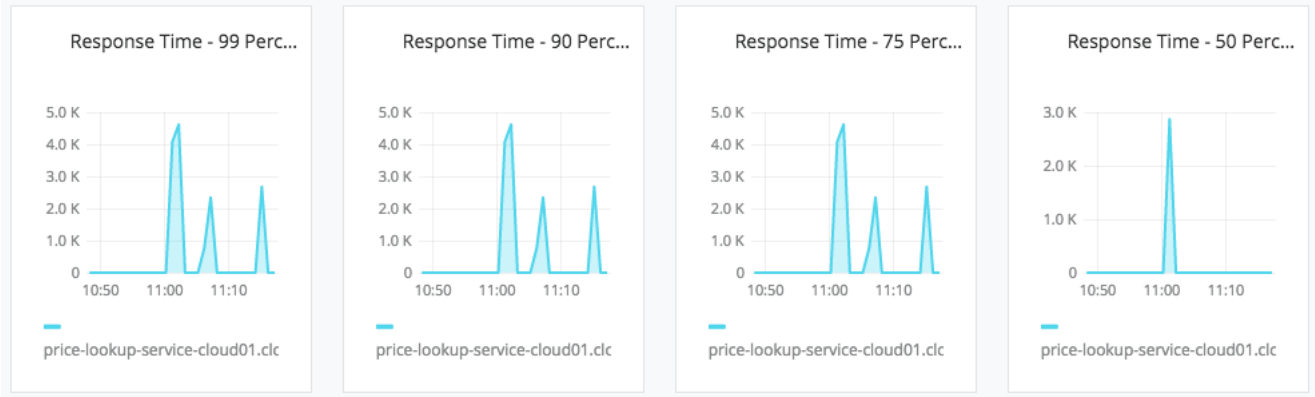
Observe the results of the load in Anypoint Monitoring

33. Return to Anypoint Monitoring and reload the page to get the latest statistics.
34. In the center section, select Overview.

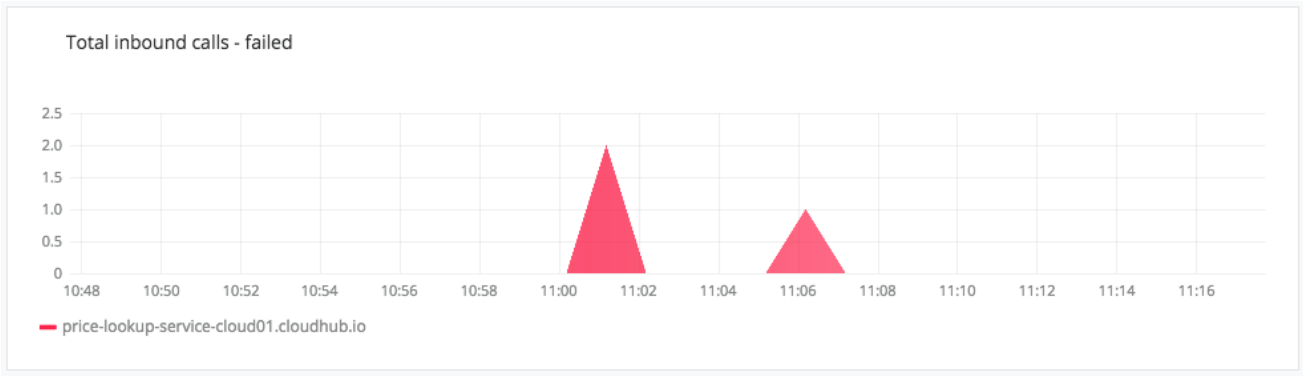
35. Locate the graph named Thread count – server.



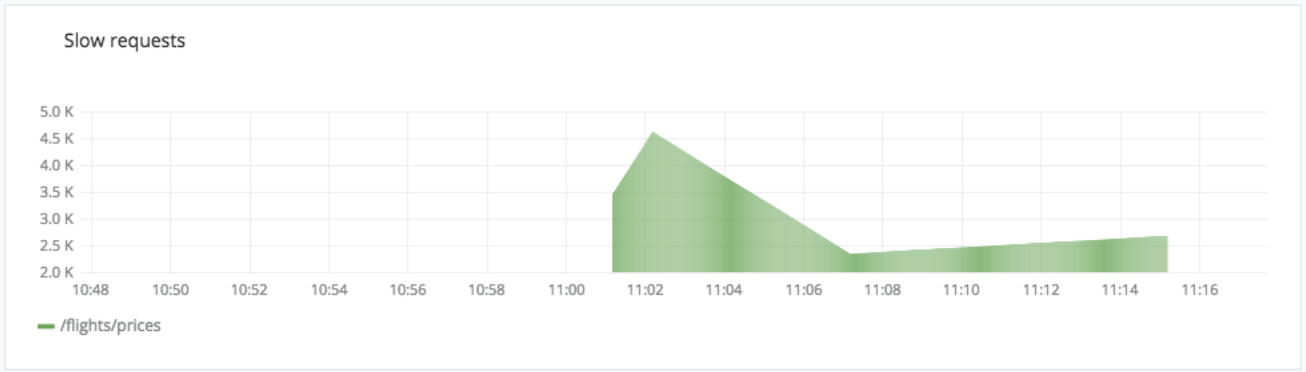
36. In the center section, select the Inbound tab.
37. Scroll down and locate the graphs named Response Time – xx Percentile inbound.
38. Observes the spikes caused by executing the load test.



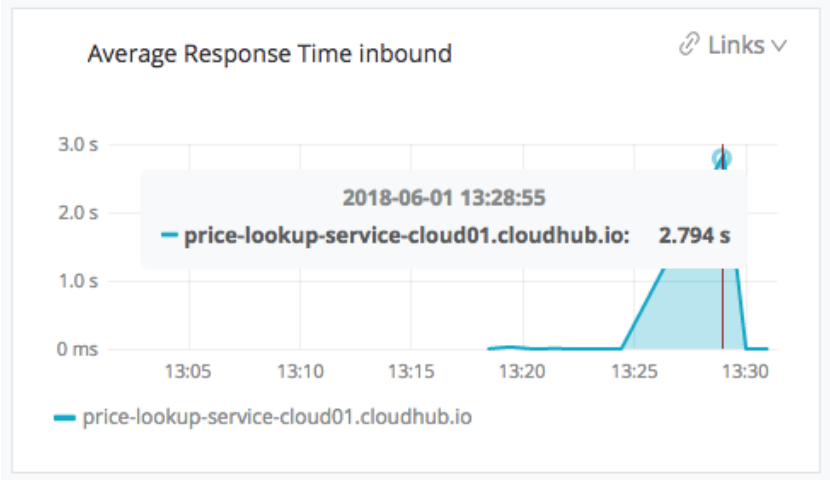
39. Locate the graph named Total inbound calls – failed.
40. If there are any failed tests, verify that the graph indicates the failure.



41. Locate the graph named Slow requests and observe the results.



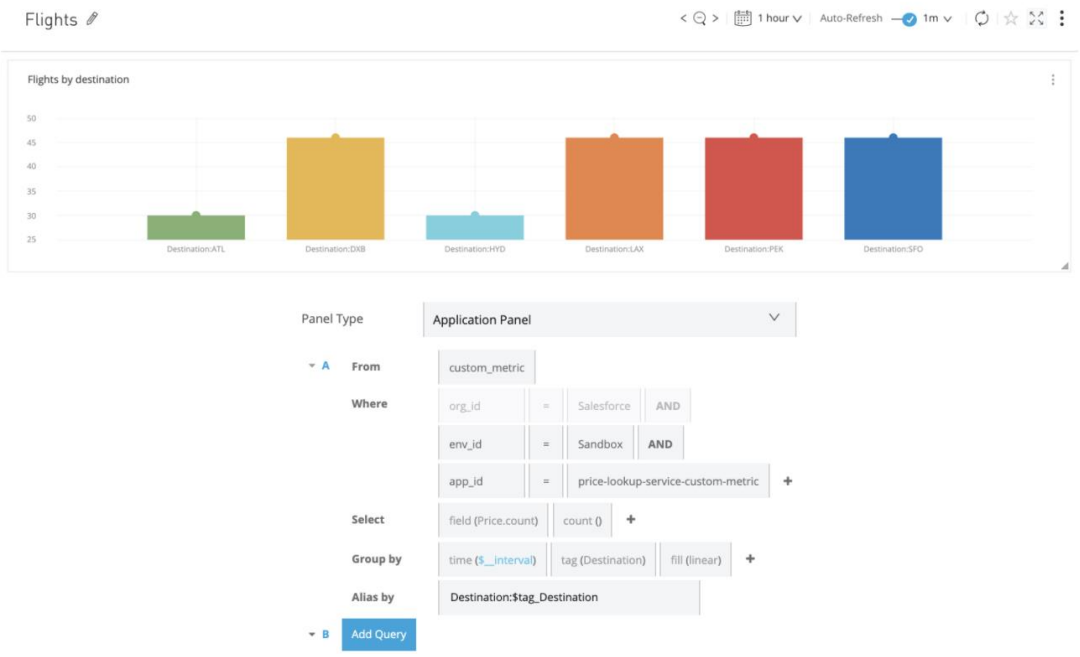
42. In the center section, select the Overview tab, and locate the average response time of the 10 HTTP requests.



Walkthrough 9-2: Monitor Mule application using custom metrics

In this walkthrough, you deploy a Mule application that generates custom metrics, and you view these custom metrics in the Anypoint Monitoring. You will:

- Deploy a Mule application that generates custom events.
- View custom metrics in built-in dashboards of Anypoint Monitoring.
- Create custom dashboards using custom metrics generated from Mule application.



Deploy the custom metrics Mule application to the CloudHub

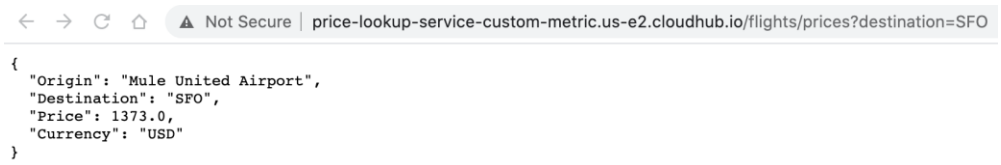
1. Return to Runtime Manager.
2. In the left-side, select Applications.
3. Click Deploy application.
4. Configure the custom metrics deployment.
    - Application Name: price-lookup-service-custom-metric
    - Deployment Target: Sandbox

- Application File: Browse for the $APAINTSOL_HOME/studentFiles/mod09-design-logging-monitoring/starter_resources folder in your studentFiles and select the price-lookup-service-custom-metric.jar deployable archive.
- Runtime version: Choose the latest Mule 4.3.0 Runtime version
- Worker size: 0.1 vCores
- Workers: 1
- Select Enable Monitoring and Visualizer

5. Click Deploy Application.
6. In a web browser, navigate to the App url and add the resource URI /flights/prices/?destination=SFO

   http://price-lookup-service-custom-metric.us-Xe2.cloudhub.io/flights/prices?destination=SFO

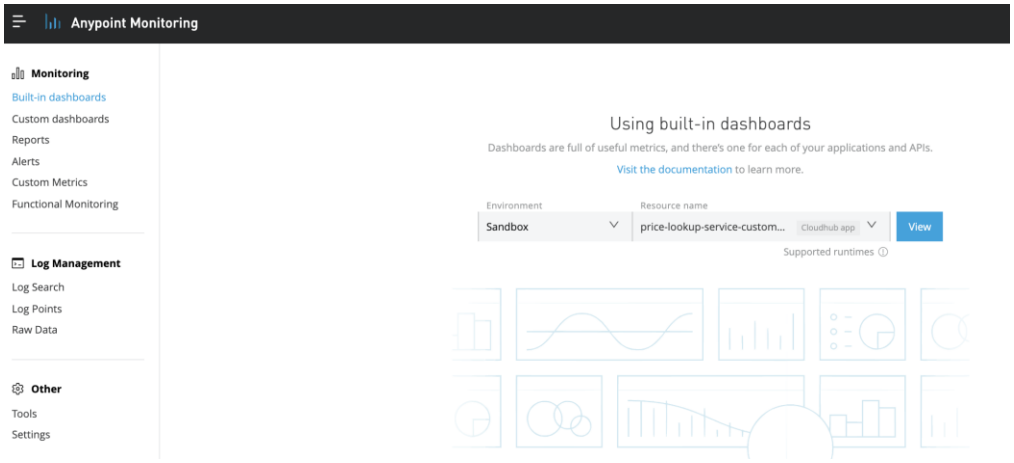
7. Observe the results.



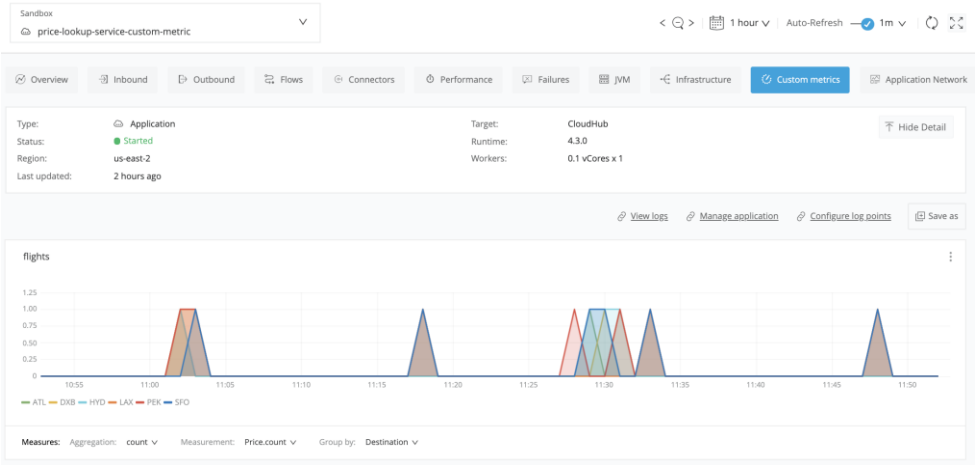
8. Change the destination to get different results; the price should change in every response.

View Mule application custom metrics using Anypoint Monitoring

9. In the main menu, select Anypoint Monitoring.
10. In the left-side navigation, select Built-in dashboards under Monitoring.
11. Select Sandbox as environment and price-lookup-service-custom-metric as resource.



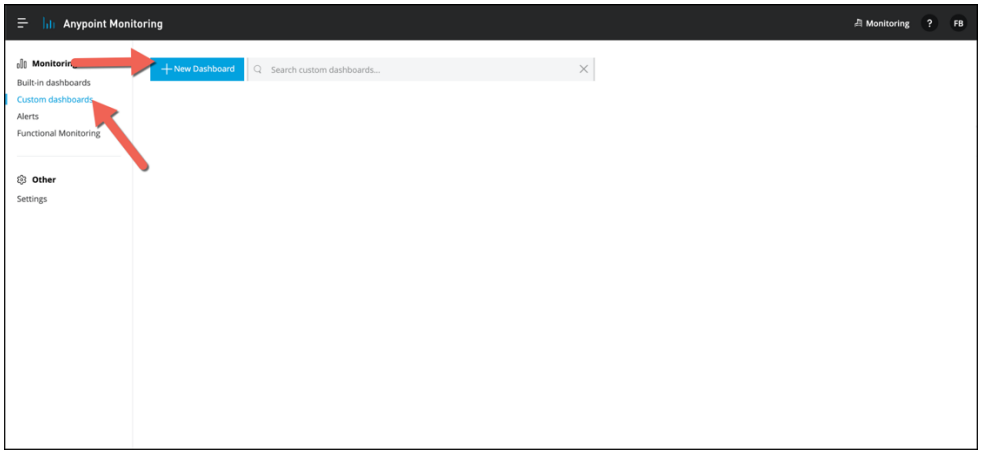
12. Click on view.
13. In the right-side panel select custom metrics.



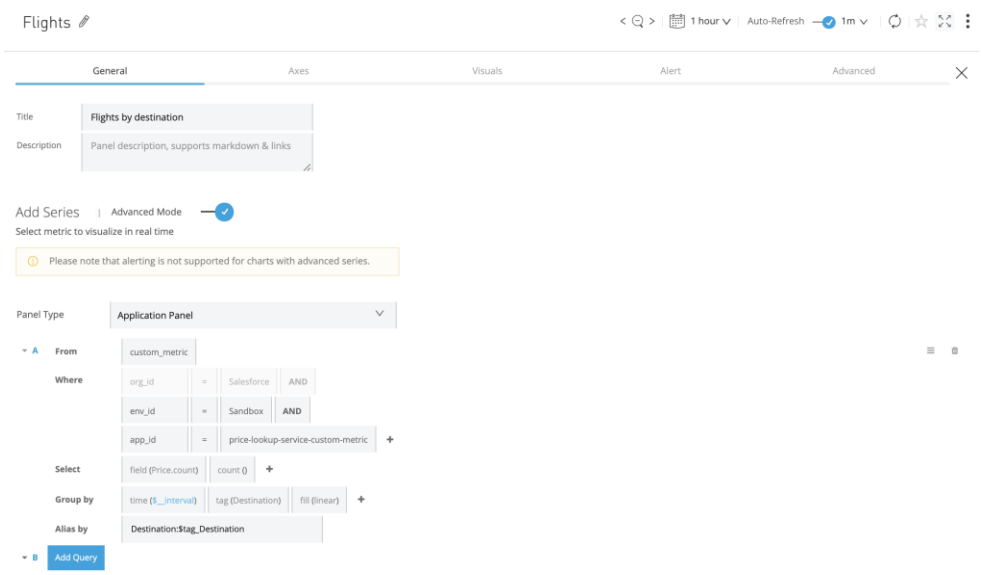
14. Select different measures to explore different graphs created automatically by Anypoint Monitoring.

Create custom dashboard

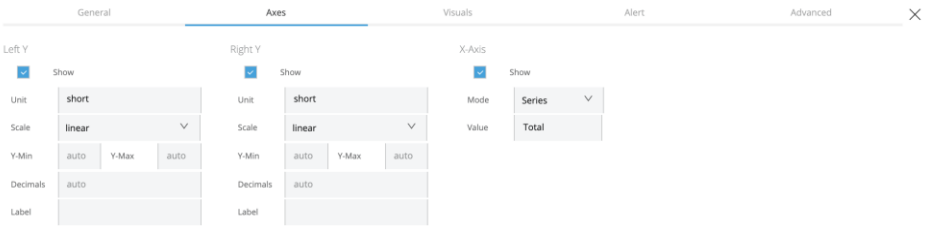
15. On the left-hand side, select Custom dashboards.



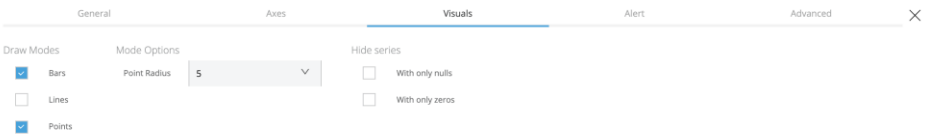
16. Click on the pencil icon to edit the dashboard properties of New dashboard.
17. Give the name as Flights to dashboard.
18. Press Add row button and then the Add Graph button.
19. Press three dots on the right side of the graphic and select configure.
20. In the General tab enable Advanced Mode and configure query as per the below screen shot.



21. Click on Axes tab, In the X-Axis change the mode to series.



22. Click on Visuals tab and select the Points checkbox.



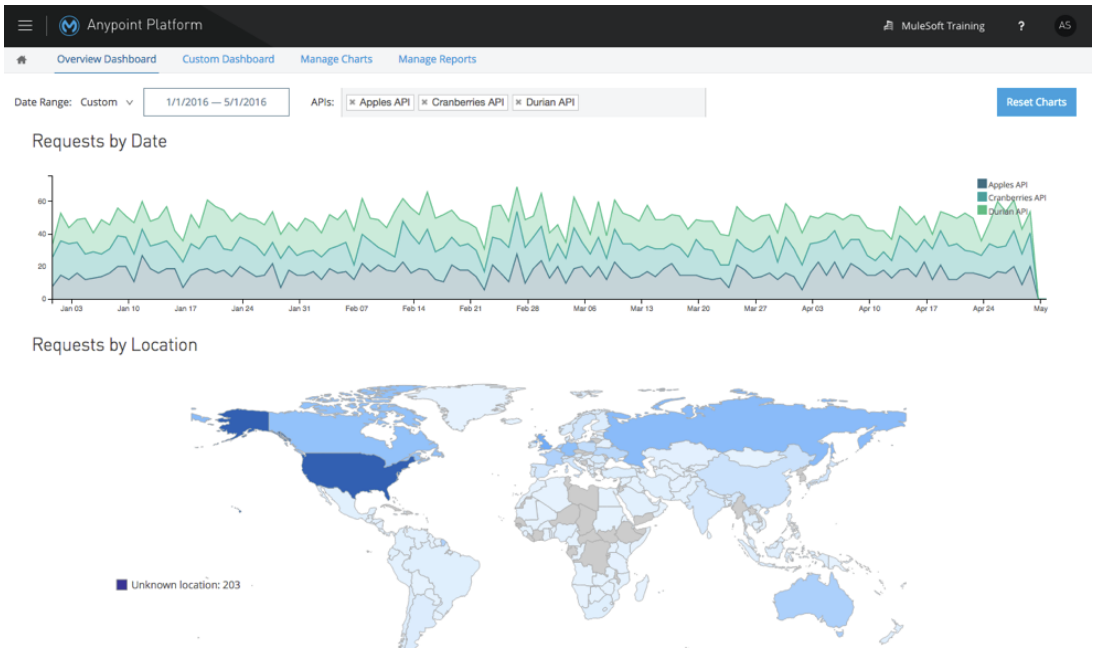
23. Click on Save Changes button.
24. Custom Flights dashboard appears like below.

25. Explore different features to create customized dashboards.

Walkthrough 9-3: Explore the Anypoint Analytics Dashboard

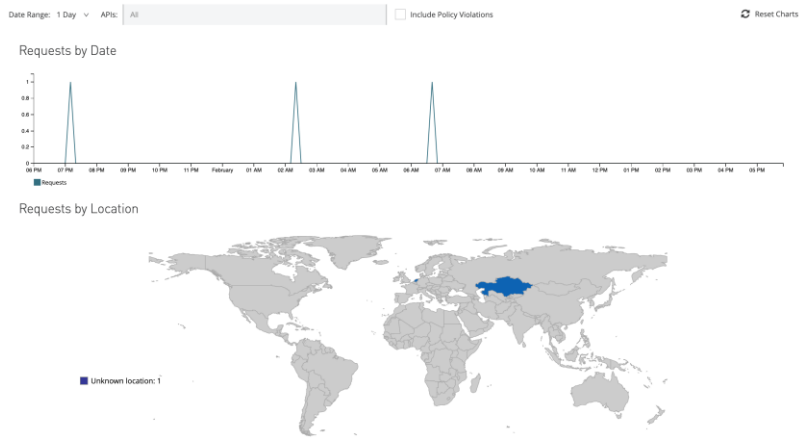
In this walkthrough, you explore the Anypoint Analytics Dashboard for an API. You will:

- Identify aspects and components of Anypoint Analytics dashboard.



Explore the Anypoint Analytics Dashboard

1. In Anypoint Platform, navigate to API Manager.
2. In the left-side navigation, select Analytics.
3. Review an API's usage and performance using the API Analytics Dashboard.



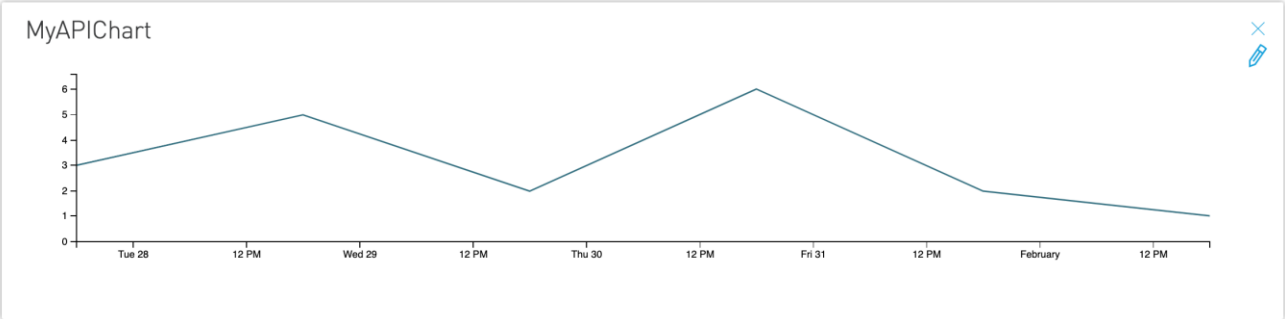
4. Explore the Overview Dashboard.

Create a custom chart

5. If the left-side navigation, select Manage Charts.
6. Create a new chart to view the number of requests to all APIs.
7. Add the custom chart to the dashboard.
8. Click Save Dashboard.

Create a custom dashboard

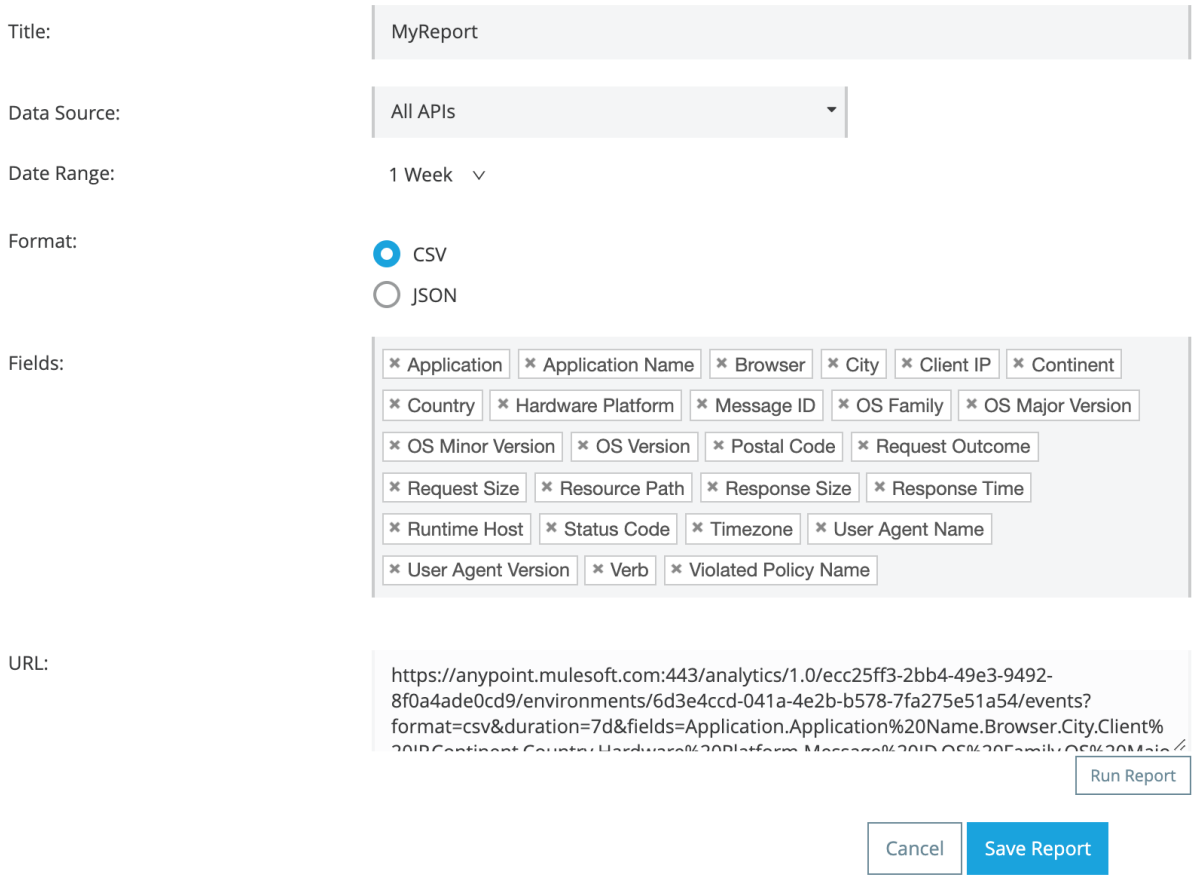
9. In the left-side navigation, select Custom Dashboards.
10. Drag and drop your new chart into the dashboard.
11. View the metrics reported by the new chart.



Create a report

12. In the left-side navigation, select Manage Reports.
13. Create a new report.

14. Select all fields.

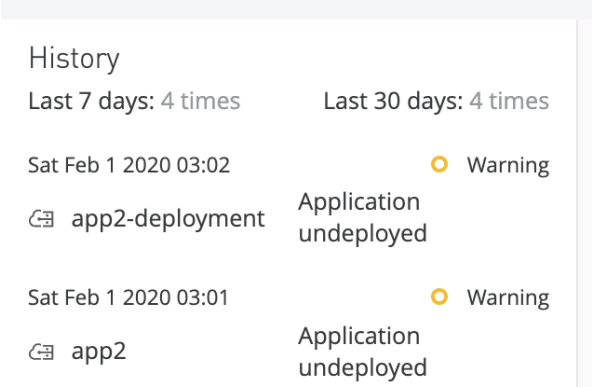


15. Run the report and download the result report file.
16. View the report data.

Walkthrough 9-4: Configure and view alerts for a Mule application

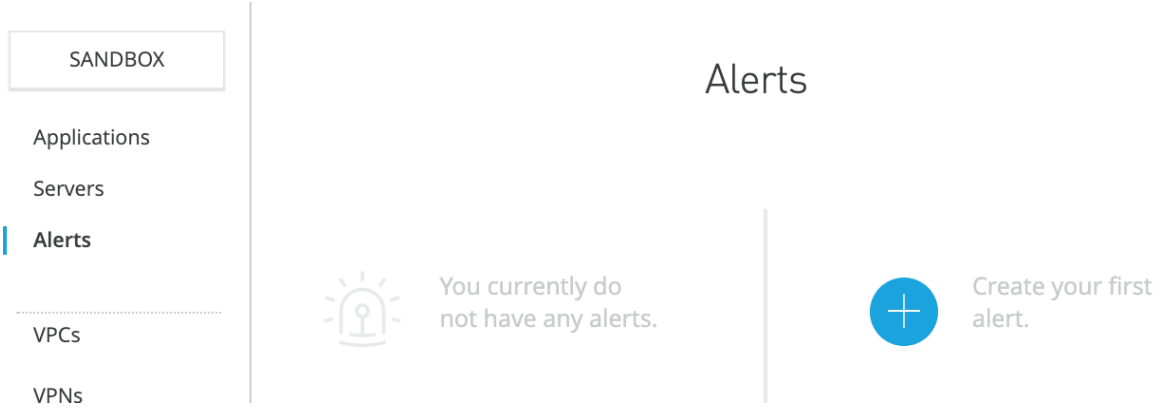
In this walkthrough, you configure alerts for a deployed Mule application. You will:

- Configure alerts for a Mule application using Runtime Manager.
- Generate alerts in a Mule application.
- View alerts in Runtime Manager.



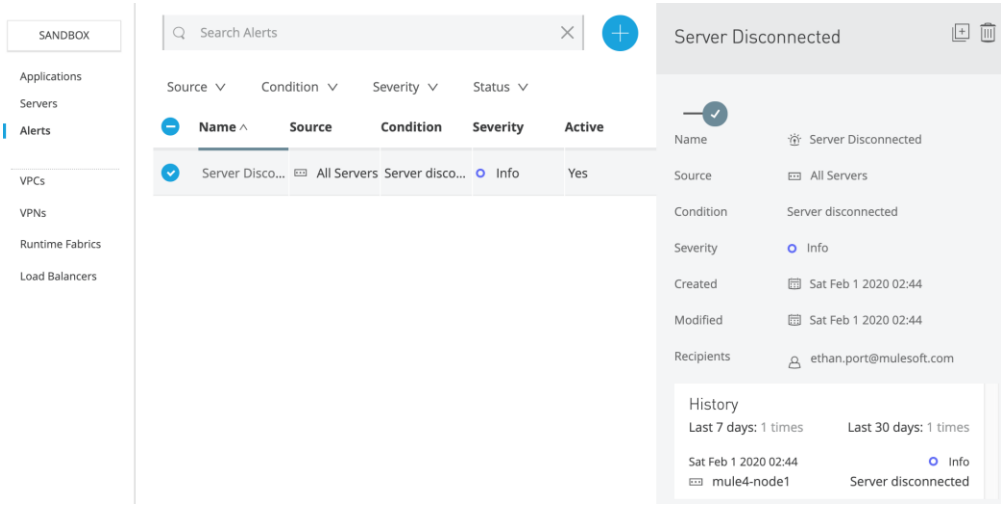
Configure alerts for registered customer-hosted Mule runtimes

1. Return to Runtime Manager.
2. Navigate back to the Alerts section.



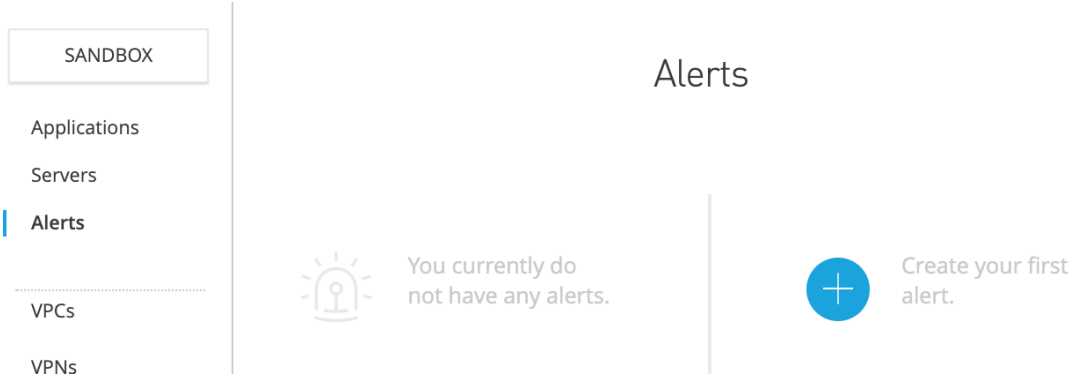
3. Configure an alert named Server Disconnected configured with:
    - Source: Servers
    - Server type: Servers
    - Condition: Server disconnected
4. Configure an email address to get alerts.

5. Click Submit.
6. Select the radio button for the new Server Disconnected alert.
7. In the mule4-node1 command-line interface, type Ctrl+C to stop the Mule runtime.
8. Start the Mule runtime again.
9. Return to Runtime Manager and verify a Server Disconnected alert is reported.



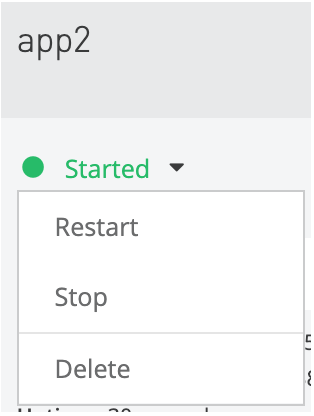
Configure alerts for Mule applications

10. Click the + Create alert button.

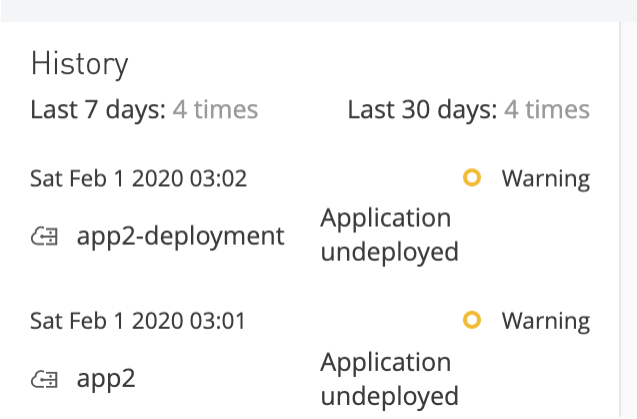


11. Configure an alert named Server Disconnected configured with:
    - Source: Applications
    - Applications: Hybrid Applications
    - Condition: Application undeployed
12. Configure an email address to get alerts.
13. Click Submit.

14. Navigate to the Application section and delete one of the Mule applications that is deployed to the mule4-node1 Mule runtime.



15. Deploy the Mule application back to the mule4-node1 Mule runtime.
16. Navigate back to the Alerts section and select the Application Undeployed alert.
17. In the left side History section, verify you see an alert reported with the Mule application deployment name.



Answer these reflection questions

18. What scenarios would require some or all Mule application logs to be externalized?

19. What scenarios would not allow storing Mule application logs in CloudHub, and what are the trade-offs resulting from this decision?

20. What is the trade-off when Mule application logs are only stored in CloudHub?

21. What type of information is not available in the Anypoint Monitoring dashboard?

22. What are the use cases and requirements for this missing information?

23. What are the differences and similarities between the API Manager dashboard and the

24. Runtime Manager dashboard?

25. What information is missing in either of these dashboards, and what scenarios would require this missing information?

26. How are alerts defined and what types of alerts are possible?

27. What types of alerts are missing, and which scenarios require these missing alert types?

28. How can you add new alerts to a Mule application in various runtime planes?

29. How are event notifications exchanged and handled?

30. How can alert notifications be shared with other monitoring systems?

31. What use cases would require using Business Events and Anypoint Insight?

32. What are the trade-offs of using Insight in a production environment?

33. What are some alternatives to using Business Events, and what are the trade-offs?

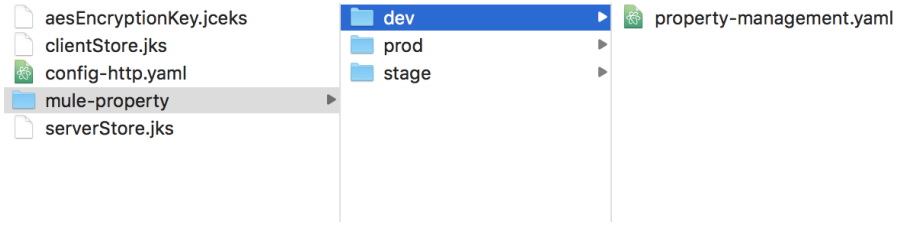
34. Can Business Events be exchanged with another log management system such as Splunk or ELK?

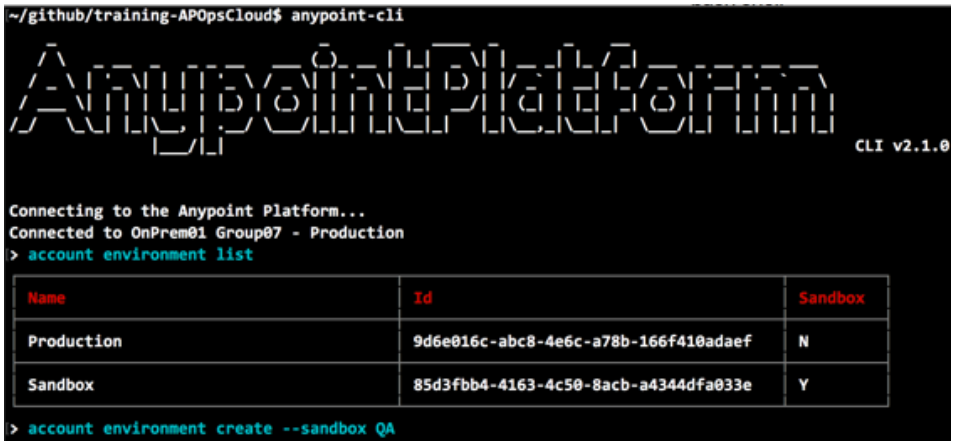
35. Can Business Events be written to a log file?

References

- CloudHub Custom Log Appender: https://docs.mulesoft.com/runtime-manager/custom-log-appender
- Splunk (Universal Forwarder): http://docs.splunk.com/Documentation/Forwarder/7.1.1/Forwarder/Abouttheuniversalforwarder
- Log4j specific-system appenders:
  - Splunk Logging Java: http://dev.splunk.com/view/splunk-logging-java/SP-CAAAE3P#_Maven
  - TCP/Socket appender: http://dev.splunk.com/view/splunk-logging-java/SP-CAAAE3R

Module 10: Designing an Efficient and Automated Software Development Lifecycle





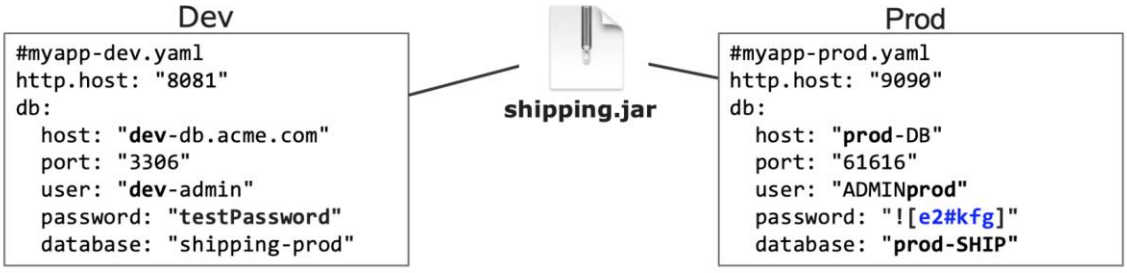
At the end of this module, you should be able to:

- Manage properties files for Mule applications across different environments.
- Manage Anypoint Platform environments for Mule application deployments.
- Design testing strategies for Mule applications.
- Implement continuous integration and continuous delivery (CI/CD) for an organization.
- Automate deployment and management with Anypoint Platform.

Exercise 10-1: Define an effective strategy to manage properties files

In this exercise, you configure properties files for a Mule application. You will:

- Analyze strategies to customize properties for different environments.
- Design security policies for external directories to store properties files.
- Analyze which properties should be encrypted.
- Design application logic to package or override Mule application properties for different environments.



Locate the Mule application for which to define properties

1. Navigate to the mod10-automatic-sdlc/solutions folder in your studentFiles folder.
2. Import the excercise10-1-solution Mule application into Anypoint Studio and explore how properties are used.

3. Explore how safely-hidden properties are set and used in CloudHub deployments.

4. What is the difference between safely-hidden properties and encrypted properties?

5. Do safely hidden properties in a Mule application deployed to CloudHub ever also need to be encrypted?

6. Which environments should the Mule applications be deployed to?

7. What properties and values are set in the Mule applications for each environment?

8. What are some things that should be agreed upon with stakeholders about how to use an external directory to store Mule application properties?

9. Define a naming strategy for property files for various environments.

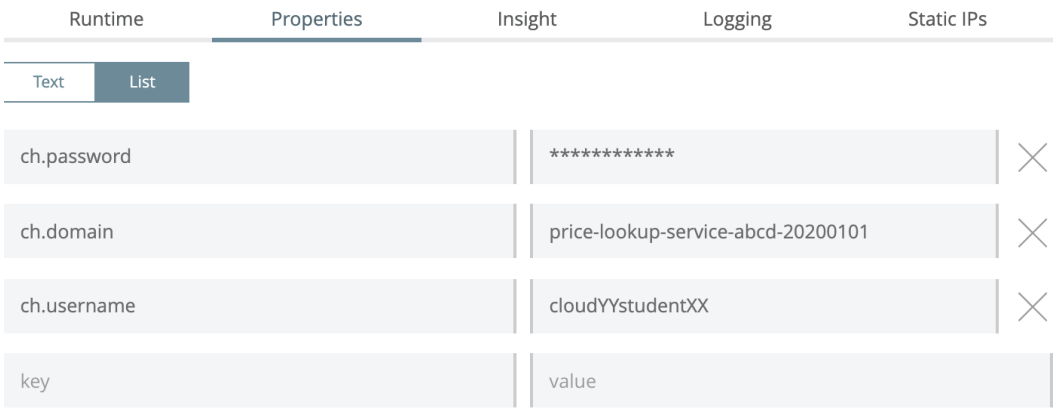
10. Define security policies for the external directory.

11. Define application logic to dynamically read in property files for each environment, without changes in application code.

Walkthrough 10-2: Set properties for a deployed Mule application by using Runtime Manager

In this walkthrough, you deploy a Mule application that uses property placeholders to CloudHub and to a customer-hosted Mule runtime target. You then change property values for the deployed Mule applications by using the Runtime Manager Properties tab for each Mule application deployment.

- Deploy a Mule application that uses property placeholders.
- Change property values for a Mule application using the Properties tab in Runtime Manager.



Deploy a Mule application that uses property placeholders to a customer-hosted Mule runtime

1. Return to Runtime Manager.
2. Create a new Mule application deployment named flights-prod.
3. Browse to the mod10-automate-sdlc/starter_resources folder in your studentFiles and select the flight-reservations.jar deployable archive.
4. Wait for the deployable archive to fully upload to Runtime Manager, then deploy the Mule application to the mule4-node1 target.
5. Select the Properties tab and set the following properties:

    - http.port: 9000
    - flight.reservations.url: prod-flights
    - reservations.sabre.response.timeout: 60000
    - reservations.sabre.port: 9000

    Note: You need to configure a different HTTP port so this deployment does not cause a bind error with other Mule applications you deployed earlier in the class.

6. Deploy the Mule application.
7. Wait for the deployment to start.

Test the Mule application and verify the property values are used

8. In a web browser, navigate to http://localhost:9000; a help page should display.

   Note: This confirms the http.port property was ready into the Mule application.

9. Return to Runtime Manager and change the http.port property to 9050 and reservations.sabre.port to 9050.
10. Wait for the Mule application to redeploy.
11. Return to the web browser and try navigating to http://localhost:9000; a connection refused error should display.
12. Navigate to http://localhost:9050; the same help page should display.

    Note: You will use this deployment in a later walkthrough to control flows and view flow metrics inside Runtime Manager.

Test the Mule application deployed to CloudHub and verify the property values are used

13. Return to Runtime Manager.
14. Switch to Sandbox environment.
15. Click the price-lookup-service-*abcd-20200101*
16. Click the Properties tab and enter the following properties:

    - ch.username: <<Your Anypoint Platform username>>
    - ch.password: <<Your Anypoint Platform password>>
    - ch.domain: <<The application name>> in the Application Name textbox, such as price-lookup-service- *abcd-20200101*

17. Click Deploy Application.

Verify the price-lookup-service application is started correctly

18. In a web browser, navigate to the App url and add the resource URI /flights/prices/?destination=LAX

    `http://price-lookup-service-abcd-20200101.cloudhub.io/flights/prices?destination=LAX`

19. Observe the results.

20. Refresh the page several times to get different results; the price should change in every response.

Walkthrough 10-3: Build, package, and deploy a Mule application by using the Mule Maven Plugin

In this walkthrough, you use Maven to manage Mule application development and deployment. You will:

- Install Maven on your computer.
- Use Maven commands to build and package a Mule deployable archive JAR file from Mule application source files.

Download Maven

1. Download Maven from https://maven.apache.org/download.cgi and install it on your computer.
2. Change into the directory of a Mule project in your Anypoint Studio workspace; make sure you are in the outer project folder with the project's pom.xml file.

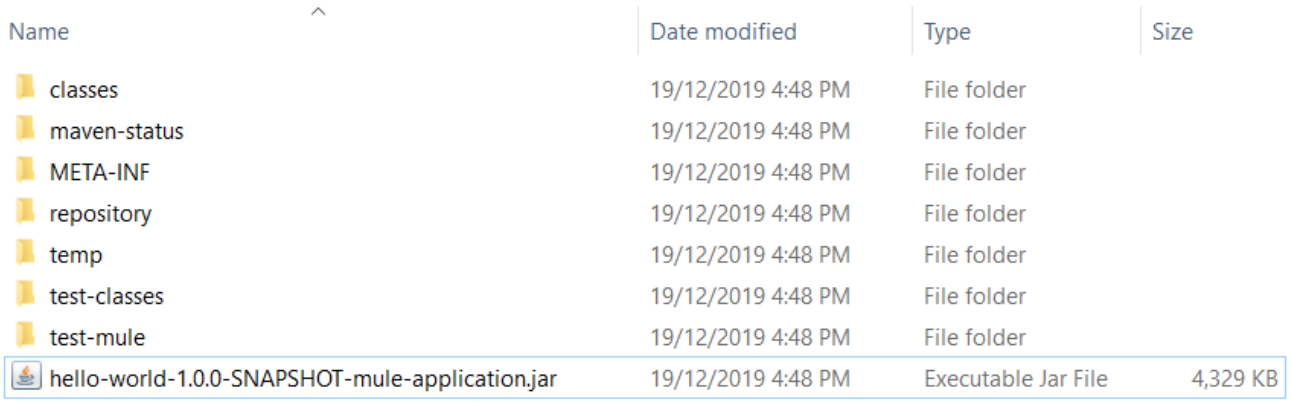
Create an archive for deployment with Maven

3. Run the command:

```
mvn clean package
```

4. Wait for the command to complete with a BUILD SUCCESS message
5. Locate the resulting package underneath the target directory.

6. Observe the resulting packaged jar (under the target directory); this file is suitable for deployment as it contains the dependencies necessary to run.



Create a package suitable for importing back into Anypoint Studio

7. Add additional Maven options to build a Mule deployable archive with the source files structure to that the resulting Mule deployable archive JAR file can be re-imported back into Anypoint Studio.

```
mvn clean package –DattachMuleSources
```

Create the smallest Mule package JAR file that can be successfully imported back into Anypoint Studio

8. Combine the last –DattachMuleSources Maven option with –DlightweightPackage to avoid bundling up the deployment dependencies.

```
mvn clean package -DattachMuleSources -DlightweightPackage
```

Note: This lightweight JAR archive is only useful for importing back into Anypoint Studio. This combination does not include enough dependency information and resources to successfully deploy the Mule application to a Mule runtime.

Other Maven commands

9. Try out some other Maven commands to see the dependencies of your project:

```
mvn dependency:tree
mvn dependency:build-classpath
```

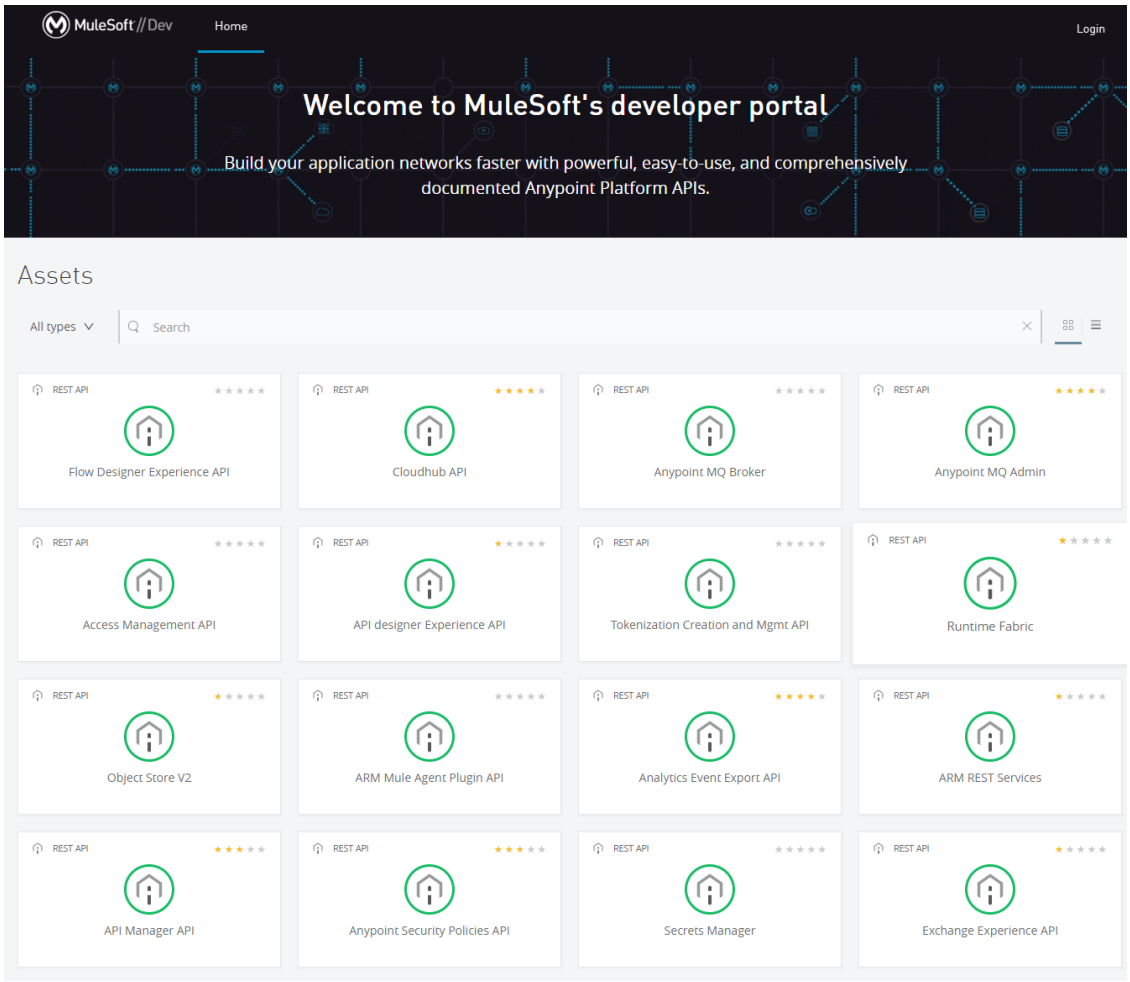
Walkthrough 10-4: Call Anypoint Platform REST APIs

In this walkthrough, you explore the Anypoint Platform Developer portal and the various REST APIs. You will:

- Explore Anypoint Platform REST APIs in the Anypoint Platform Developer portal.

Explore the Anypoint Platform Developer portal

1. In a web browser, navigate to the Anypoint Platform Developer portal at https://anypoint.mulesoft.com/exchange/portals/anypoint-platform.



10. Examine the list of APIs for any useful areas of automation.

Use the Access Management API

11. Explore the Access Management API; this API is used to authenticate with Anypoint Platform in order to securely call the other APIs.
12. Read and explore the Access Management API at:
13. https://anypoint.mulesoft.com/exchange/portals/anypoint-platform/f1e97bc6-315a-4490-82a7-23abe036327a.anypoint-platform/access-management-api/version/v1/pages/home/

Create a token to call other APIs

14. In the Access Management API, locate the REST API call to generate an access token from an Anypoint Platform username and password.
15. In a REST client, make a POST request to:

    https://anypoint.mulesoft.com/accounts/login

    - Set the body of the POST request to your Anypoint Platform username and password, as a JSON object

```
{
    "username" : "Max"
    "password" : "p@$$Wurd123"
}
```

    Note: Replace the username and password values with the values for your Anypoint Platform account.

Answer these reflection questions

16. When and how would you use the Anypoint CLI?

17. When would you use the Anypoint Platform REST APIs instead, or in conjunction with the Anypoint CLI?

18. What are some ways you can build automated scripts with Anypoint CLI and the Anypoint Platform REST APIs?

19. What is the difference between a source control repository (e.g. Git repository) and asset repository system (e.g. Nexus rep)

References

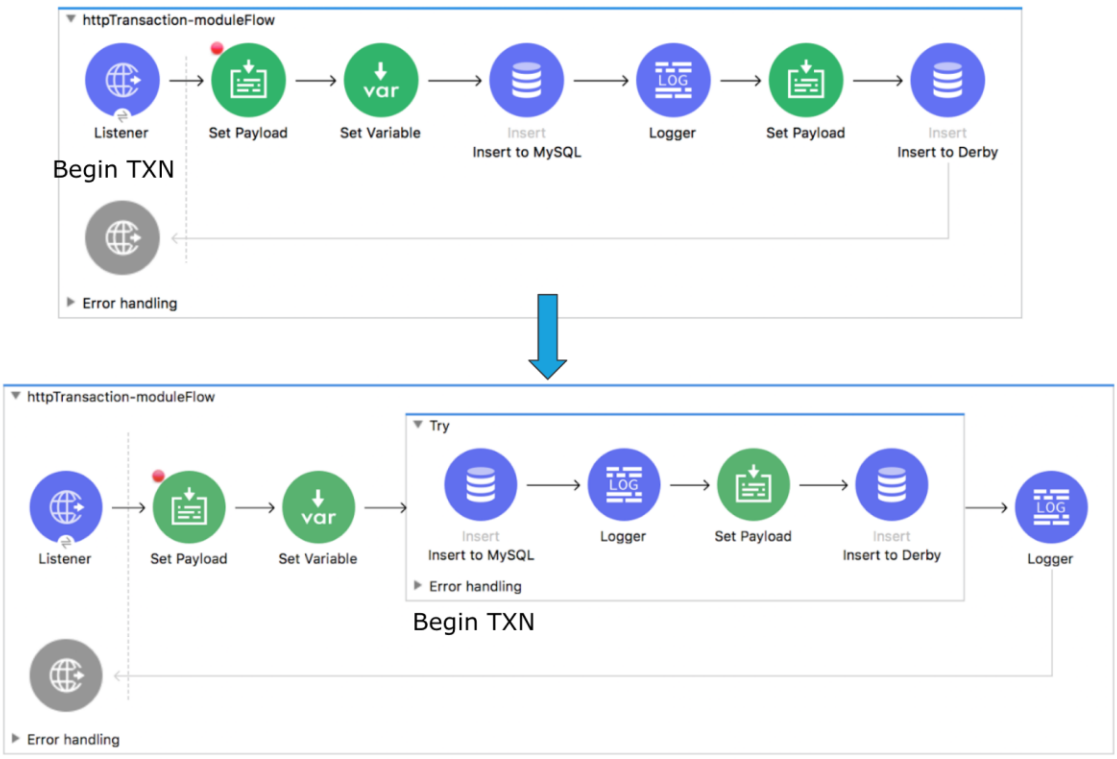
- Continuous Integration – MuleSoft Documentation: https://docs.mulesoft.com/mule-runtime/4.2/continuous-integration
- Mule Maven plugin: https://docs.mulesoft.com/mule-runtime/4.2/mmp-concept
- Anypoint REST APIs in the Developer Portal: https://anypoint.mulesoft.com/exchange/portals/anypoint-platform

PART 3: Using Strategies to Meet Non-Functional Requirements

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

- Decide when and how to design for transactional requirements.
- Clarify and balance competing reliability, high availability, and performance goals.
- Design to effectively balance competing reliability, high availability, and performance goals.
- Design secure Mule applications, their network communications, and their deployments.
- Summarize the course with a completed architecture.

Module 11: Designing Transaction Management in Mule Applications



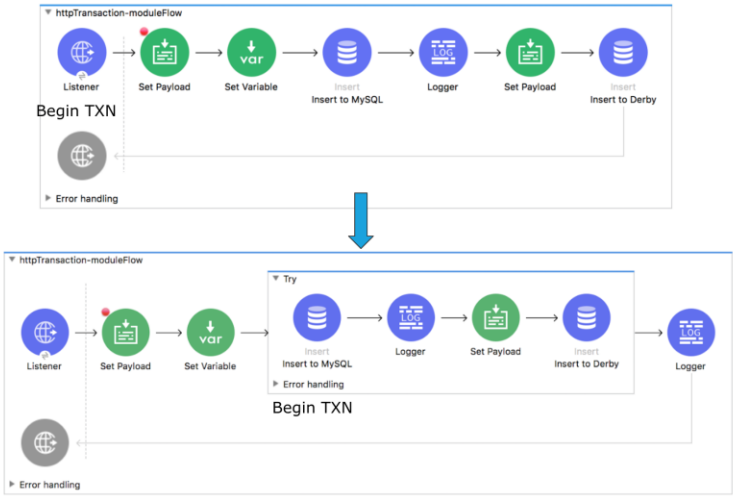
At the end of this module, you should be able to:

- Identify why and when transactions are supported in Mule applications.
- Identify resources that participate in transactions in Mule applications.
- Demarcate transaction boundaries in Mule applications.
- Choose the transaction type based on the participating resources.
- Manage a transaction using the Saga pattern.

Exercise 11-1: Identify when and why a Mule application should support transactions

In this exercise, you explore transaction options in Mule applications and their trade-offs. You will:

- Identify how to configure transactions in Mule flows.
- Analyze trade-offs and edge cases when using transactions in Mule applications.
- Prototype transactional Mule flows.

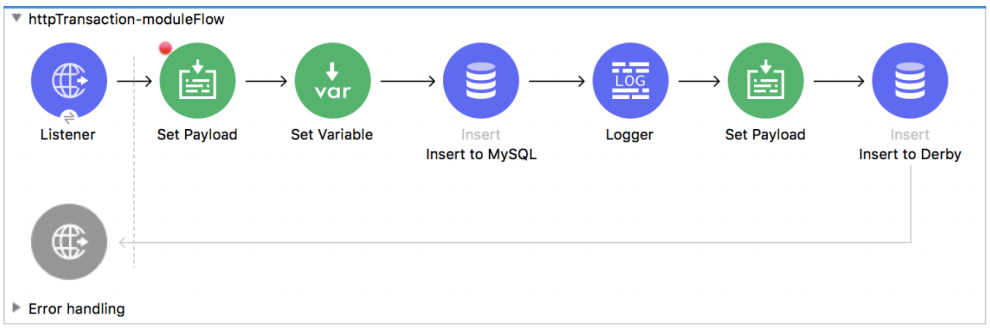


Review the scenario for this exercise

1. Read the following scenario:
    - A Mule application makes multiple related modifications, and at least one modification fails.

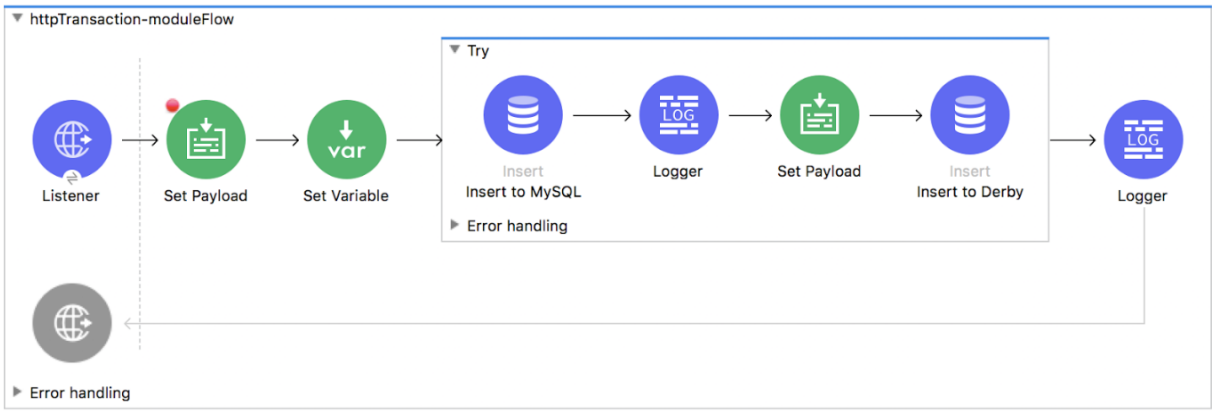
Prototype transaction options

2. Create a new Mule project in Anypoint Studio.
3. Prototype a flow with some database operations.



4. Look at configuration options for the flow and for every component in the flow and identify transactional resource(s), transactional boundary transactional type (XA or Local).

5. Add transaction options to the flow.



Analyze transaction management requirements and trade-offs

6. How can the consistency of the entire system be guaranteed?

7. What does consistency mean?

8. Discuss the different aspects of this scenario, including edge cases and performance considerations.

Analyze the transaction options for the Mule flow

9. Identify Mule connectors and components that participate in transactions.

Define the TX boundary and TX type for this use case

10. Should the flow use local/single resource or XA TX?

11. Should the TX start with the listener (flow source) or have transaction in in a Try scope or in some other scope?

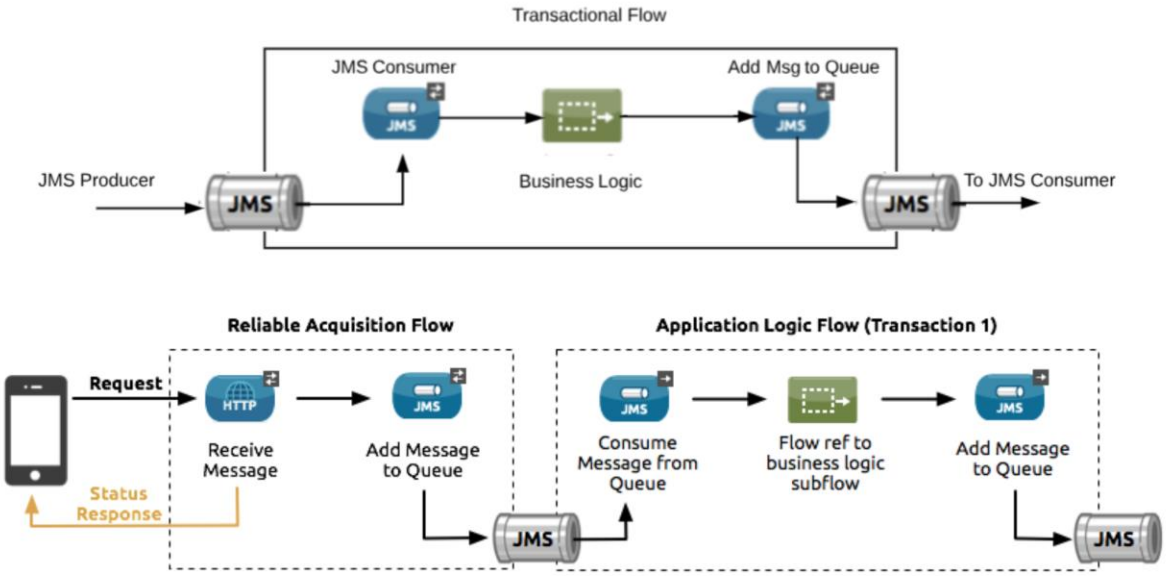
Answer these reflection questions

12. What are some reasons to use transactions?

13. For JMS, when might Acknowledgements be more suitable than Transactions?

14. For what is the Saga pattern used?

Module 12: Designing for Reliability Goals



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

- Distinguish between competing non-functional requirements.
- Clarify and validate reliability goals for a scenario.
- Design Mule applications and their deployments to meet reliability goals.
- Identify reliability pattern for mule application and their deployments.

Exercise 12-1: Reflect on reliability patterns

In this exercise, you reflect on reliability patterns and options. You will:

- Reflect on reliability patterns.

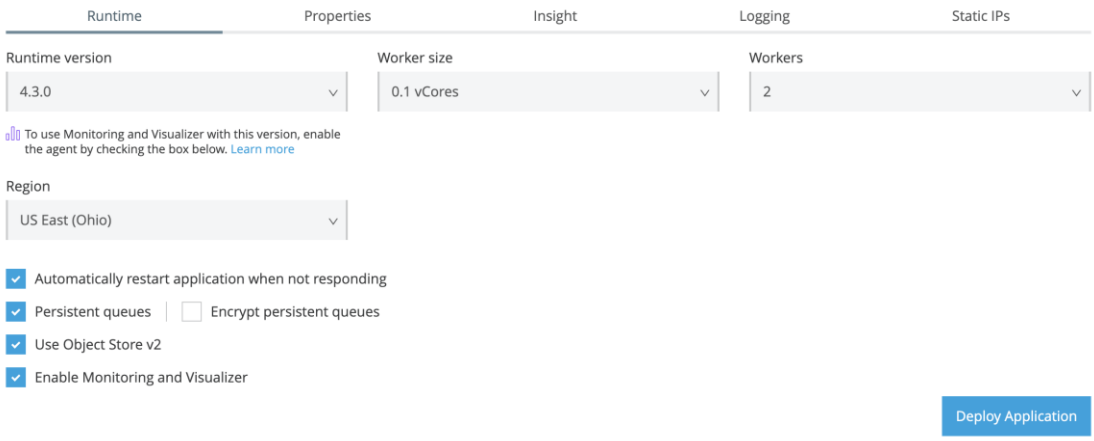
Answer these reflection questions

1. What do the Mule reliability patterns involve?

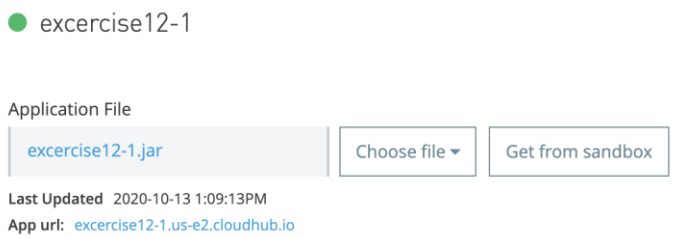
2. With what other non-functional requirements do reliability pattern compete or conflict?

Deploy and test a Mule reliability pattern

3. In the mod12-reliability/starter_resources folder, deploy the excercise12-1.jar Mule application to multiple CloudHub workers with Persistent queues checked.



4. After the Mule application deployment starts, click the App url link.



5. In the web page that opens, click the Submit Batch Orders button.

6. Return to Runtime Manager and view the excercise12-1 application's log files.
7. Click on each worker and verify records are being processed by both workers.

8. Select Deployment and search for different integers like 50, 60, 70, and 80.
9. Verify each step is processed by one worker, but different workers process different

Note: The application submits the large orders file to a for-each loop, which sends batches of 5 orders at a time to a VM queue for processing. All the CloudHub workers read from the same VM queue, so different batch items are load balanced among all the CloudHub workers for processing.

10. Return to the web page that opened and Submit another batch order.
11. Return to Runtime manager and restart the Mule application while it is still processing batch records.

Answer these reflection questions

12. Does the restarted Mule application continue processing records from the old batch job?

13. Open the Mule application in Anypoint Studio and explore how VM queues are used to provide some reliability patterns.
14. How does this queuing pattern provide reliability?

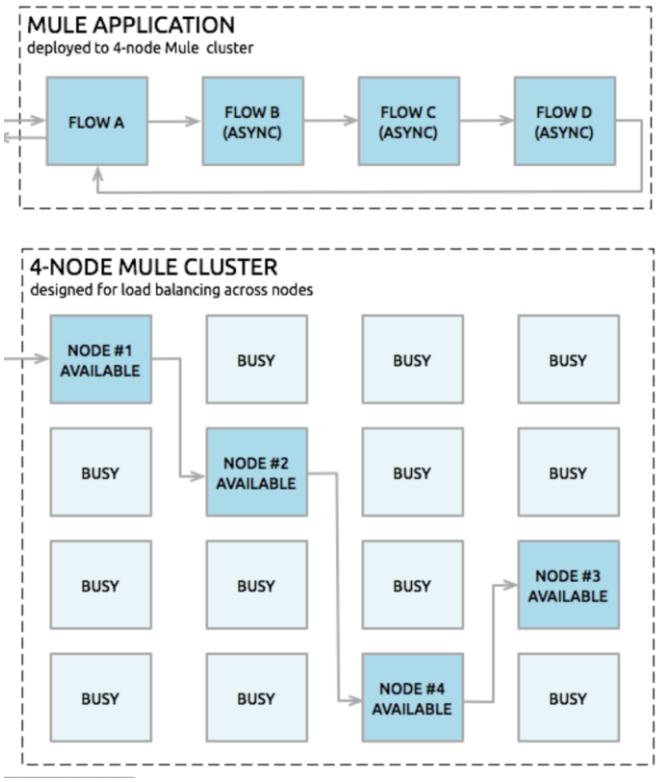
15. How does this queuing pattern handle higher volumes of incoming requests?

16. Does this queuing pattern help with very large individual messages?

17. What other options can be used besides VM queues to distribute work to multiple workers?

18. What are the trade-offs between these other options?

Module 13: Designing for High Availability Goals



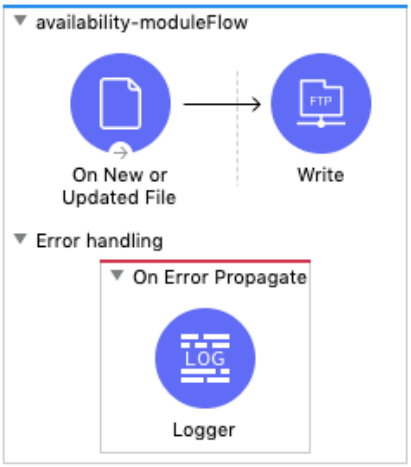
At the end of this module, you should be able to:

- Clarify HA goals for Mule applications.
- Balance HA goals with reliability and performance goals.
- Identify ways to achieve high availability (HA) using Anypoint Platform, in CloudHub and on-premises.
- Describe how clustering and load balancing works.
- Identify HA aware connectors and their design trade-offs.

Exercise 13-1: Design an HA and reliable application for file transfer

In this exercise, you analyze HA and reliability goals for a file transfer use case. You will:

- Design a file transfer use case that meets agreed HA goals, including minimum uptime and minimum allowed Mule application failures.
- Design flow/s for the Mule application.
- Identify scaling options for the Mule application.
- Identify the number of workers for the Mule application.
- Decide Mule application persistence options to meet the Mule application's reliability goals.
- Design a Mule application to meet reliability goals related to Mule application and system crashes.



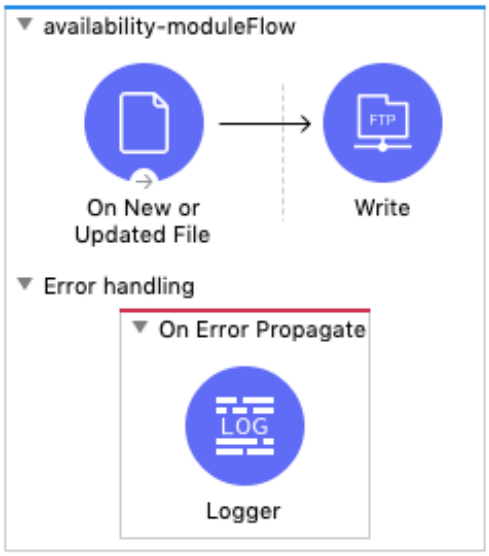
Review the Mule application requirements

1. Read the following requirements:
   - The Mule application has to transfer a file from a local directory to an online SFTP server
   - The Mule application has to meet an HA availability SLA of 99.99% per year and must be highly reliable against Mule application failures
2. How would you design one flow to transfer a file to the SFTP server?

   Note: This is less reliable, but easier and more direct to implement

Prototype a file transfer flow

3. Create a new Mule project in Anypoint Studio.
4. Add components to sketch a file transfer flow using the FTP connector and File connector.



5. What happens if the FTP write fails and the file is already deleted?

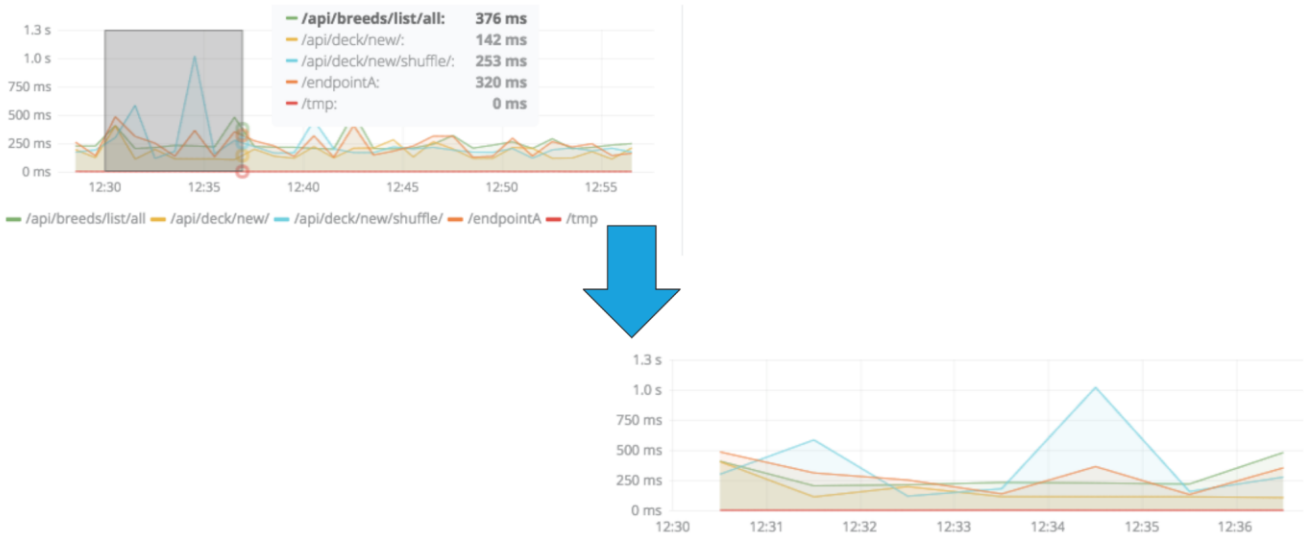
Answer these reflection questions:

6. What failures points can reduce reliability of the system?
7. If an error happens after reading the file, but before the file is completely processed, how can you assure the file is not deleted?
8. What should be done if file reading fails a few times?
9. What should be done if the FTP write process fails a few times?
10. Can persistence help to enhance reliability?
11. What are the options to maintain persistence?
12. When should the file be deleted from the inbound file server?
13. What horizontal or vertical scaling options can help?

Redesign the Mule application to meet reliability and HA requirements and goals

14. Redesign the Mule application, perhaps including messaging queues to decouple processing steps.
15. Prototype the new design in Anypoint Studio and paste a screenshot here.

Module 14: Optimizing the Performance of Deployed Mule Applications



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

- Clarify performance goals for Mule applications.
- Balance performance goals with reliability and HA goals.
- Identify the need for performance optimization and associated trade-offs.
- Describe ways to search for and locate performance bottlenecks.
- Describe how to design, architect, design, and implement for performance.
- Describe ways to measure performance.
- Describe methods and best practices to performance tune Mule applications and Mule runtimes.

Exercise 14-1: Identify and choose between performance optimization options and trade-offs

In this exercise, you evaluate and analyze performance goals for Mule applications, and you balance performance goals with other reliability and security goals. You also identify metrics to measure performance objectives, and then you apply this analysis to design Mule applications for particular use cases. You will:

- Select performance goals that may be important to particular types of Mule applications.
- Compare how performance goals balance with other security and reliability goals.
- Identify key performance indicators to measure Mule application performance objectives.
- Decide which types of performance optimizations are required for a particular type of Mule application and its associated business use cases.

Think about performance objectives and requirements

1. Why do Mule applications need to optimize performance?

2. What are the key performance indicators for Mule application?

3. Is optimizing performance mandatory for all Mule applications

Identify performance goals that may be important to particular types of Mule applications

4. You cannot tune for all performance goals simultaneously – so identify different performance goals even if they compete with each other.

   Note: Replace any of the example goals in the table.

Performance goal	Consideration	Comments
Less than 100 ms end-to-end latency		
1Gb payloads		
100k different client requests per hour		

Identify how performance goals balance with other reliability goals

5. Identify other reliability and security goals, or other types of goals.

Reliability/security goal	Consideration	Completing performance goals
All or nothing transactional groups of operations		
External API calls with replay tracking		
Idempotent message tracking		
At least once processing		

6. Brainstorm ways to balance these goals and identify SLAs and requirements that may need to be relaxed.

Identify key performance indicators to measure Mule application performance objectives

7. What are some measures of the performance goals?

   Note: These metrics are something you would need tools or scripting to measure.

Reliability/security goal	KPI	Measuring tools or techniques

Decide which types of performance optimizations are required for a particular type of Mule application and its associated business use cases

8. For the following scenarios, identify likely performance goals and optimization priorities.

scenario	Performance and optimization goals	Ranked priority
A "type ahead" address lookup API called from a mobile app		
End of Financial year financial transactions report extract		
Managed file transfer mechanism for security video files		

Exercise 14-2: Identify the best processing model for optimizing performance

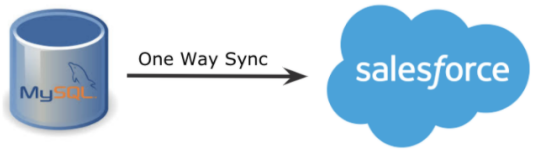
In this exercise, you analyze options to optimize performance of the processing models for various use cases with different orders of magnitude throughput and latency requirements. You will:

- Design a flow for 3 different scenarios with different order of magnitude throughput and latency requirements.
- Compare the trade-offs and reasons for each design.

Scenario 1 - Sync 500 records a day with periodic sync

Scenario 2 - Sync more than 10,000 records within 5 minutes

Scenario 3 - Real time sync between MySQL and salesforce
(less than 10 records a minute)



Scenario 1: Sync 500 records a day with periodic sync

1. How should the Mule application be triggered and what components and connectors come next?

Identify expected and required workload for the scenario

2. Does the scenario need real time processing or can periodic processing suffice?
3. Identify how much latency is allowed before new database records must be synchronized to Salesforce
4. Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.).

5. Evaluate various processing options for scenarios, such async message queues, batch, or schedulers.

6. Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model.

7. Prototype Mule flows to meet these requirements and paste screenshots here:

Scenario 2: Sync more than 10,000 records within 5 minutes

8. How should the Mule application be triggered and what components and connectors come next?

   Note: Assume there is a 250ms roundtrip + processing time latency for each call to Salesforce.

9. Identify expected and required workloads for the scenario.

10. Does the scenario need real time processing or can periodic processing suffice?

11. Identify how much latency is allowed before new database records must be synchronized to Salesforce.

12. Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.).

13. Evaluate various processing options for scenarios such as async message queues, batch, and schedulers.

14. Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model.

15. Prototype Mule flows to meet these requirements and paste screenshots here:

Scenario 3: Real time sync between MySQL and salesforce (less than 10 records a minute)

16. How should the Mule application be triggered and what components and connectors come next?

    Note: Assume there is a 250ms roundtrip + processing time latency for each call to Salesforce.

17. Identify expected and required workloads for the scenario.

18. Does the scenario need real time processing or can periodic processing suffice?

19. Identify how much latency is allowed before new database records must be synchronized to Salesforce.

20. Identify message source(s) for flow(s) that best meet workload and processing requirement (real time, periodic, etc.).

21. Evaluate various processing options for scenarios such as async message queues, batch, and schedulers.

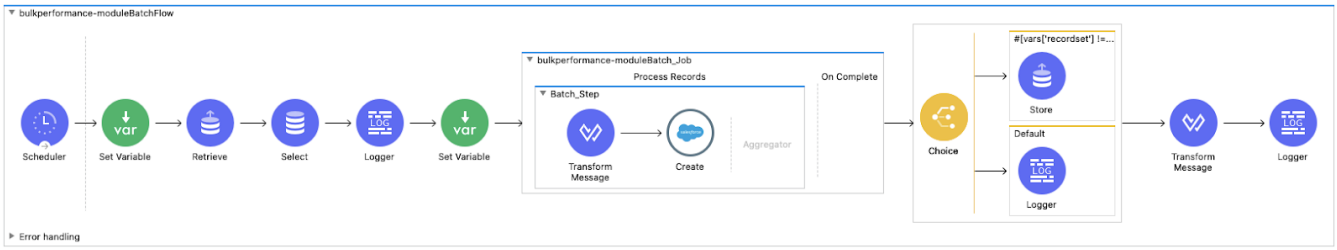
22. Evaluate and analyze if the expected and required workloads justify the use of the proposed processing model.

23. Prototype Mule flows to meet these requirements and paste screenshots here:

Exercise 14-3: Tune a Mule application for performance

In this exercise, you performance tune a Mule application that uses a Batch Job, and Object Store, and a database connector. You will:

- List possible points of tuning of an example Mule application.
- Rank the most significant performance tuning goals for an example Mule application.



Evaluate the flow

1. Look at the flow in the exercise objectives.
2. List possible points of tuning.

3. Rank the most significant as far as performance (consider a large number of records).

Exercise 14-4: Tune a high-volume Mule application for performance

In this exercise, you design a flow to stream a high volume of database records, then transform the records to JSON format and then write the records to a file system. You will:

- Design and performance tune a flow for writing 1M records from an enterprise MySQL database system into a JSON file.
- Identify component capable of performance tuning.
- Identify tuning parameters for components.
- Understand trade-off of each tuning parameter.

Design a flow to meet the requirements

1. Design a flow for writing 1M records from an enterprise MySQL database system into a JSON file.
2. Identify what how the flow will be triggered.

3. Which options are available to build the database selects and file writing, and which one is most suitable for this scenario?

4. How will records be transformed and processed?

Tune the flow for optimum performance

5. Identify components capable of performance tuning.

6. Identify tuning parameters for components.

7. Document trade-offs between each tuning parameter.

8. Prototype the design in Anypoint Studio.

Answer these reflection questions

9. Does every application need to undergo optimization?

10. Where are the likely big gains to be had for performance optimization?

11. What are the downsides to caching?

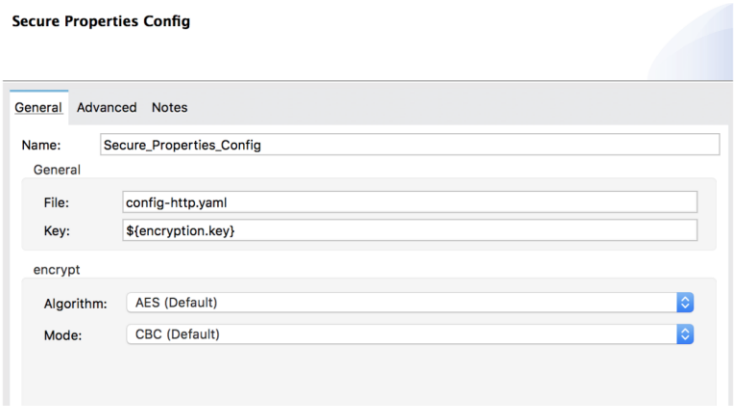
12. Can streaming help improve performance while still achieving reliability goals?

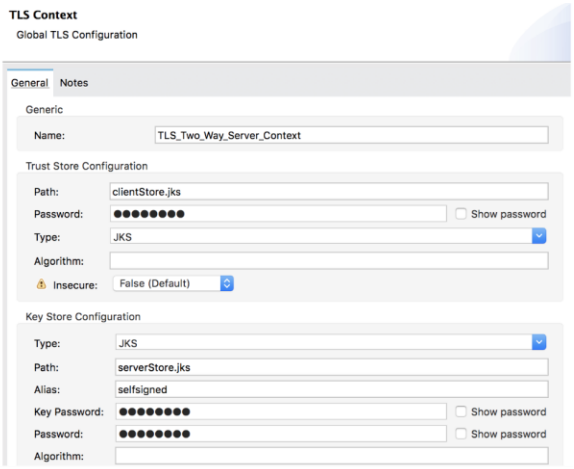
13. What are the performance trade-offs of batch vs. For Each vs. streaming?

References

- Mule runtime performance/tuning:
  - https://www.mulesoft.com/lp/whitepaper/api/reactive-programming
  - https://www.slideshare.net/mulesoft/mule-runtime-performance-tuning
  - https://blogs.mulesoft.com/biz/mule/thread-management-auto-tuning-mule-4/
- Java Hotspot Tuning:
  - "The Performance Engineer's Guide To Java HotSpot" by Monica Beckwith: https://www.youtube.com/watch?v=6a4Id3lj7Sw
  - Oracle Java tuning guides/presentations
  - https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/
  - https://docs.oracle.com/cd/E21764_01/web.1111/e13814/jvm_tuning.htm#PERFM151

Module 15: Designing Secure Mule Applications and Deployments





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

- Identify Anypoint Platform security concepts and options.
- Describe how to secure APIs on Anypoint Platform.
- Describe the security needs addressed by Anypoint Platform Edge security.
- Differentiate between MuleSoft and customer responsibilities related to Anypoint Platform security.
- Evaluate security risks for Mule applications.
- Describe how to secure Mule application properties and data in transit.

Walkthrough 15-1: Explore API policies

In this walkthrough, you will explore policies in API Manager and Anypoint Exchange. You will:

- Apply API policies to managed API instances using API Manager.
- Auto-generate and deploy an API proxy Mule application from API Manager.
- Configure an API Portal in Anypoint Exchange to use an API proxy.
- Locate the autodiscovery component in an auto-generated API proxy's deployable archive.
- View policy types and categories available in Anypoint Exchange.

View policy options for an API in API Manager

1. If it has not already been deployed, deploy the mod03-apikit Mule project to CloudHub; you can do this from Anypoint Studio or manually in Runtime Manager after exporting the Mule deployable archive from Anypoint Studio.
2. Remember to give the deployment a globally unique name such as with your initials and the current date:

   `mod03-apikit-abc20200101`

3. Wait for the deployment status to change to Started.
4. Copy the deployed App url to the clipboard and paste it here (this will be referred to as <<API CloudHub URL>>:

   `http://mod03-apikit-abc20200101.us-e2.cloudhub.io/api/weatherReports`

   Note:: You can also use the URL /api to access the Weather API, /console to see the Weather API console.

Manage the API instance with API Manager

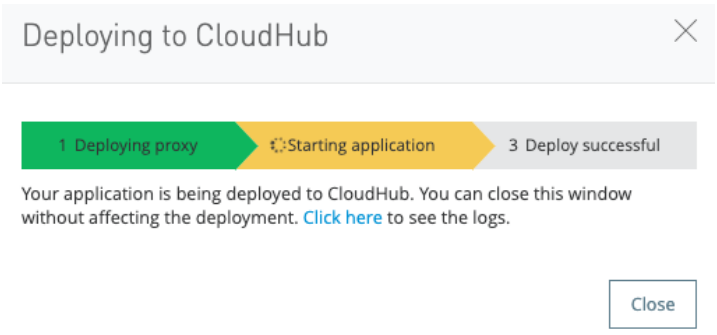
5. Navigate to API Manager.
6. Click Manage API and select Manage API from Exchange.
7. In the Manage API from Exchange form, fill in the following values:
   - API name: Browse for Weather API
   - Mule version: Check the check box to specify this is a Mule 4 deployment
   - Implementation URL: <<API CloudHub URL>>
8. Accept all the other default values and click Save; the API management page should display showing graphs of metrics.
9. Change the Managing type from Basic Endpoint to Endpoint with Proxy.
10. Click Save.

Create and deploy and API proxy for the API implementation.

11. In API Manager select Settings.
12. In the top write down the API Instance and Autodiscovery IDs.

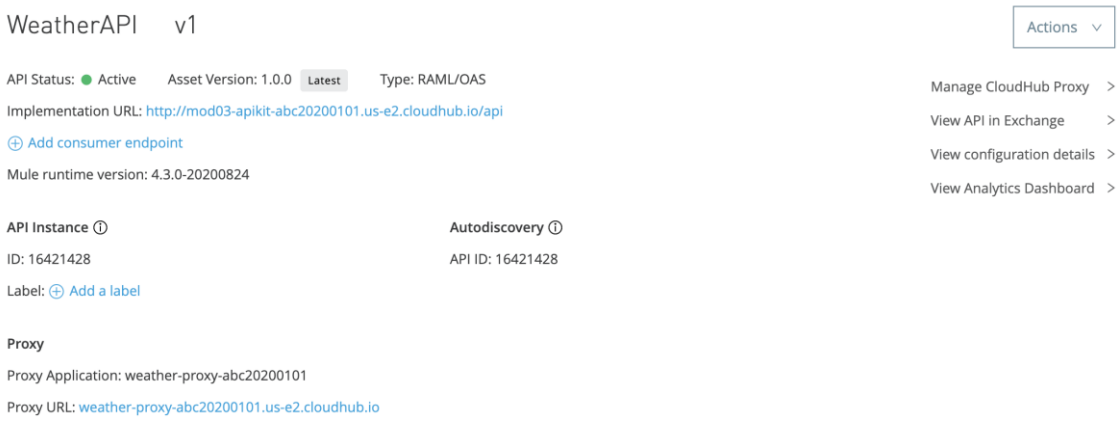
Note: Later you will observe that these IDs are used in an autogenerated API proxy Mule application.

13. Set the Deployment Configuration to the latest Mule runtime version.
14. Set the Proxy application name to a unique name such as weather-proxy-*abc20200101*.
15. Click Deploy.
16. Wait for the API proxy to deploy to CloudHub.

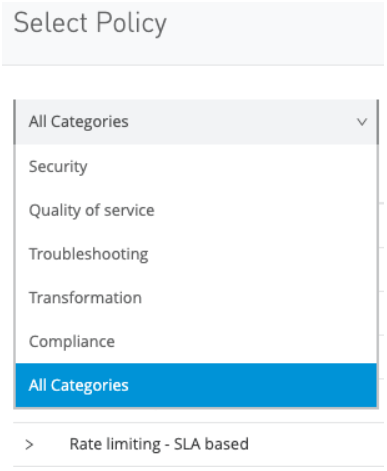


Apply policies to a managed API

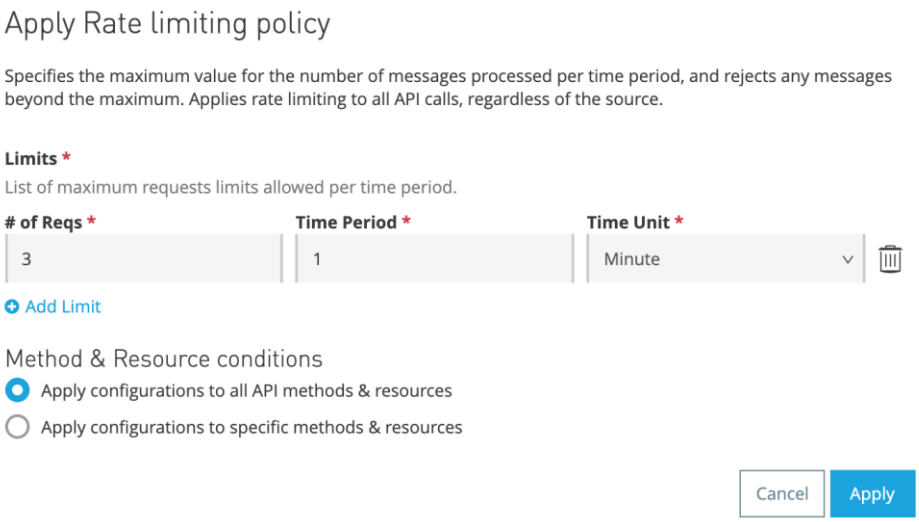
17. In the left side, select Policies.
18. Click Apply New Policy.



19. Explore all the policy options.
20. Select the All Categories drop-down menu and filter by each policy category type.



21. Select the Quality of service policy category, then select the latest version of Rate Limiting and configure the policy to 3 requests per minute.



Test the rate limiting policy

22. Make five to ten POST requests to the /order resource of the API proxy URL; after a few requests you should see an error status returned due to the rate limiting policy.

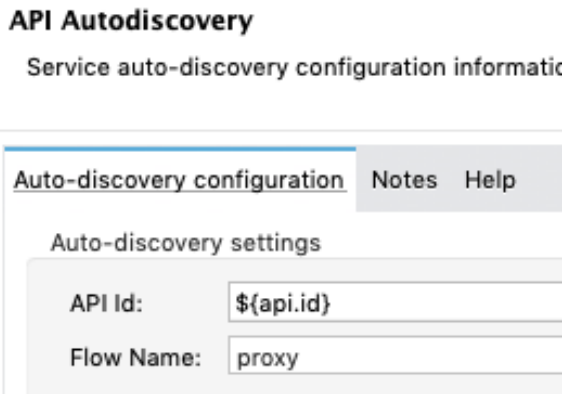
```
{
  "error": "Quota has been exceeded"
}
```

23. After a minute try again and verify you can send more API GET requests.

Inspect the API proxy Mule application source code

24. In the upper right Actions drop-down list, select Download Proxy.
25. Import the downloaded API proxy into Anypoint Studio.
26. Open the http-proxy.xml file and select Global Elements.

27. Look at the API Autodiscovery global element configuration; it should be configured with a property placeholder for the api.id, which is the API ID in API Manager for the Order API.



Note: The API Autodiscovery element communicates with API Manager to download policies from API Manager.

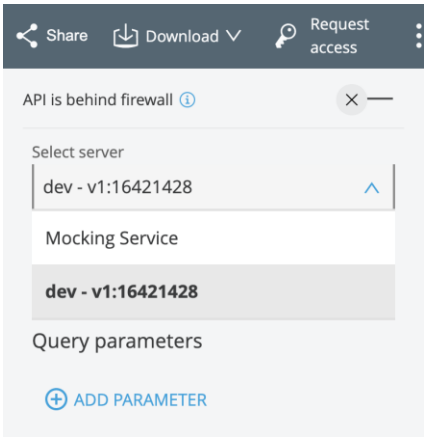
28. Open src/main/resources/config.properties; you should see the configuration for the API implementation to which the API Proxy forwards REST requests.

```
api.id=16421428
proxy.path=/*
proxy.port=8081
proxy.responseTimeout=10000
implementation.host=mod03-apikit-abc20200101.us-e2.cloudhub.io
implementation.port=80
implementation.path=/api
implementation.protocol=HTTP
implementation.api.parser=AUTO
validation.disable=false
validation.strict.headers=false
validation.strict.queryParams=false
implementation.api.spec=com/mulesoft/anypoint/gw/weather-api.raml
```

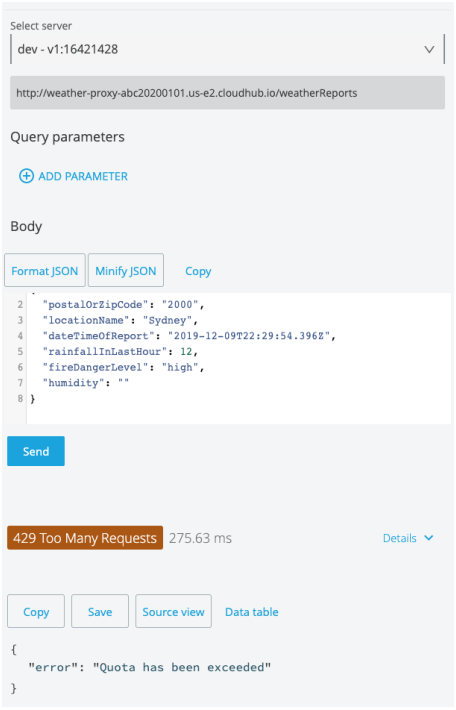
Expose an API endpoint in Anypoint Exchange

29. Copy the API proxy URL.
30. Navigate to Anypoint Exchange.
31. Search for the WeatherAPI then select its card.
32. Select the /weatherReports POST method.

33. In the upper-right drop-down list under Select server, select the API proxy deployment link.

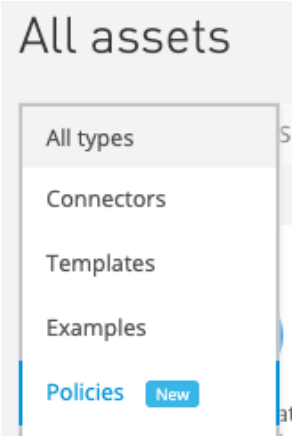


34. Click Send a few times; after a few GET requests you should see a 429 Too Many Requests response status.



View policies available in Anypoint Exchange

35. Navigate to Anypoint Exchange.
36. In the Assets drop-down menu, select Policies.

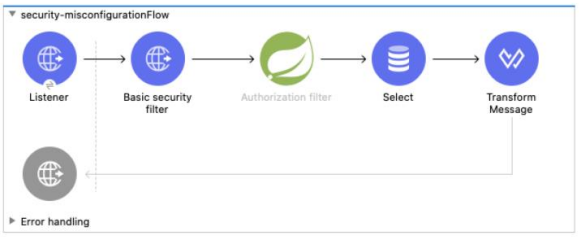


37. In the left-side navigation, select All assets.
38. Explore some of the policies.

Exercise 15-2: Identify security threats exposed by a sample application

In this exercise, you document security threats in Mule applications. You will:

- Review and identify security threats exposed in a Mule application
- Suggest fixes or improvements to security threats.

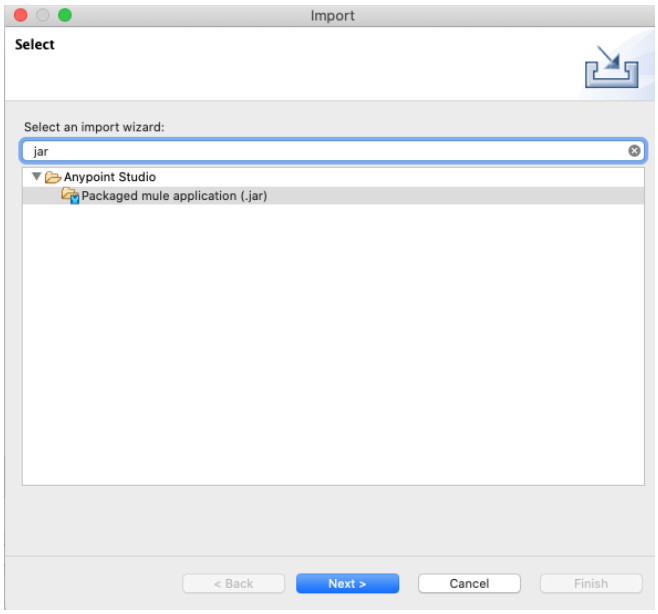



```
<ss:authentication-manager alias="authenticationManager">
  <ss:authentication-provider>
    <ss:user-service id="userService">
      <ss:user name="admin" password="admin" authorities="ROLE_ADMIN" />
      <ss:user name="joe" password="secret" authorities="ROLE_ADMIN" />
      <ss:user name="anon" password="anon" authorities="ROLE_ANON" />
      <ss:user name="user" password="password" authorities="ROLE_ANON" />
      <ss:user name="ross" password="ross" authorities="ROLE_ANON" />
      <ss:user name="marie" password="marie" authorities="ROLE_ANON" />
    </ss:user-service>
  </ss:authentication-provider>
</ss:authentication-manager>
```

```
http:
  port: "8081"
db:
  host: "localhost"
  port: "3306"
  user: "root"
  url: "training?useUnicode=true&useJDBCCompliantTimezoneShift
  =true&useLegacyDatetimeCode=false&serverTimezone=UTC&useSSL=false"
  password: "password"
file:
  securefile: "config-http.yaml"
```

Import and example Mule application with security vulnerabilities

1. Return to Anypoint Studio.
2. From the File menu select Import.
3. In the Import dialog box, type jar in the search field.
4. Select Anypoint Studio > Packaged mule application (.jar).



5. Click Next and browse for to the mod15-securing-apps/starter_resources/exercise15-2.jar file in your studentFiles.
6. Click Finish and wait for the Mule project to load into Anypoint Studio; if you see a dialog box asking to update the workspace, click Perform update.
7. After the exercise15-2 project loads into Anypoint Studio, open the security-misconfiguration.xml file.
8. In the flow, view all the components and property files.
9. Check for these security threat categories:

    - Security misconfiguration
    - Sensitive data exposure
    - Broken authentication and session management

Identify threats exposed by flows in Mule applications

10. Locate the following security flaws in the Mule application.

Security threat	Mule components	Threat category	Comments/solutions
Password, security key are in clear text			
Unused/vulnerable properties in property file			
Misconfiguration of properties			
Sensitive data exposure			
Data stored in database in clear text			
Data transmitted in clear text			
Broken authentication and session management			
User authentication credentials are not protected			
Credential can be guessed			
Session did not timeout			
Credential sent over unencrypted communication			

Answer these reflection question

11. What could be improved to better secure this Mule application?

12. How would you secure the Mule application in CloudHub vs. in a stand-alone Mule runtime?

Walkthrough 15-3: Prevent risk via secure properties for a Mule application

In this exercise, you download and use the MuleSoft provided secure properties jar tool to encrypt and decrypt values that can be used in secure properties placeholder files in Mule applications. You will:

- Download and install the secure properties jar tool.
- Use the secure properties jar tool to encrypt values for secure properties placeholders files.
- Use secure properties in a Mule application.

Download the secure properties jar tool

1. Download the secure properties jar tool from: https://docs.mulesoft.com/mule-runtime/4.3/secure-configuration-properties#secure_props_tool
2. Refer to the information on how to run the secure properties jar tool then run it in a command-line interface using values such as:

```
java -cp secure-properties-tool.jar com.mulesoft.tools.SecurePropertiesTool \
file \
encrypt \
AES \
CBC \
<<encryptionKey>> \
dev-properties.yaml\
dev-secure-properties.yaml
```

3. dev-secure-properties.yaml file looks like below:

```
db:
  host: "![KnX4buOif9KqD8oAGdpwwQ==]"
  port: "![K/nWkATLSyyYzohPijD/RA==]"
  user: "![6UtsvH/CsyVnY+VqkRE+0w==]"
  password: "![6UtsvH/CsyVnY+VqkRE+0w==]"
  database: "![k+fgMyRfKm9kq8vKOB6qBQkmTJ3LVHmxmnlOLSC4qewSIh1D+GKJJF2F8a4YGxjlgVWDzE+Nbh0VvfmiaTHiwZBTnhasi+2OvGVXHgueRYrvJNqzrP0NWGcrf+v5jZEMChe19lD/BhRgc0Cc2cukA5K+fY5R4hgMKP7o9qwReVY=]"
  batchSize: "![i974lYUqqgdLEvBU6lhz6w==]"
```

Create a Mule project that uses secure properties

4. In Anypoint Studio, import the mod15-securing-apps/starter_resources/exercise-15-3.jar file from your studentFiles as a new Mule project.
5. Once the Mule project loads into Anypoint Studio, open the src/main/resources/dev-secure-properties.yaml file.
6. Replace encrypted values from #2 in to src/main/resources/dev-secure-properties.yaml file.

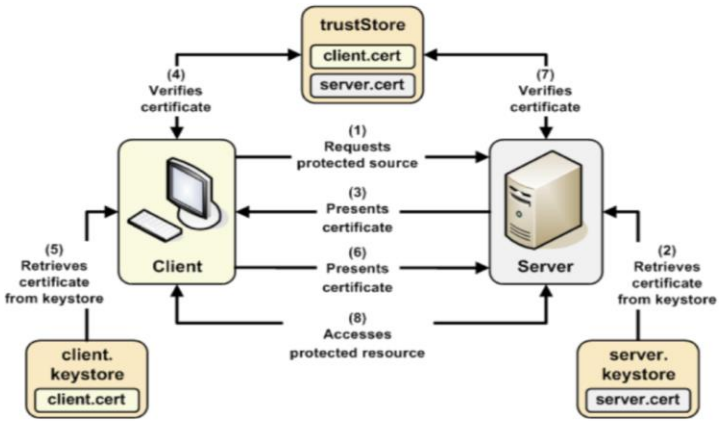
Update Run configuration with the secret key

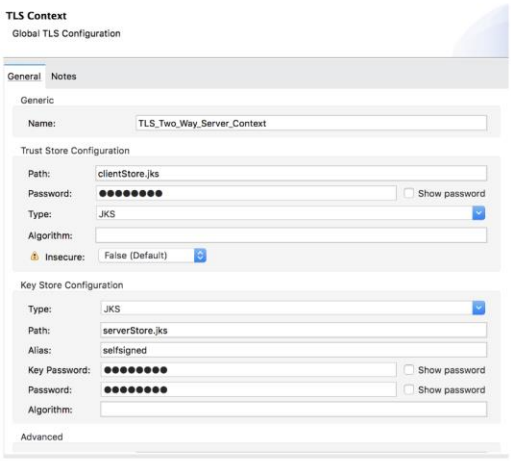
7. Update the Run configurations for the Mule project by adding the following parameter:
    - `-Dencrypt.key=<<encryptionKey>>`

Answer this reflection question

8. How can the security misconfiguration be fixed?

Module 16: Securing Network Communications between Mule Applications





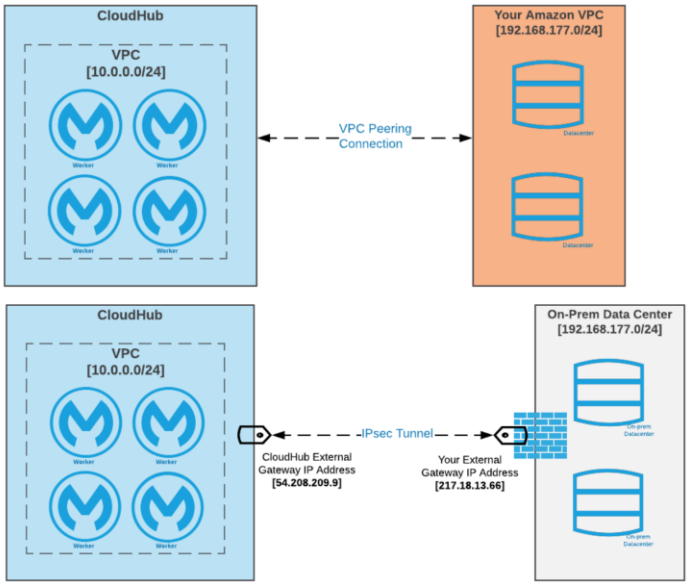
At the end of this module, you should be able to:

- Describe Anypoint Platform network security options and architectures.
- Identify MuleSoft-owned and customer-owned roles and responsibilities related to Anypoint Platform network security.
- Describe how to secure Mule applications using Java key stores.
- Design TLS communication and other network security options for an integration use case.
- Properly size an Anypoint VPC to support deployment of all expected Mule applications.

Walkthrough 16-1: Size a VPC

In this walkthrough, you decide an appropriate network subnet mask to properly size a VPC.
You will:

- Plan for two VPCs to split networking between production and non-production environments.
- Calculate the minimum CIDR block required to meet current requirements.
- Properly size VPCs to anticipate future growth requirements.



Understand the sizing exercise requirements

1. Read the following project requirements:

    - The organization has **four** different environments: Dev, Staging, Performance, and Production.
    - The organization is planning to deploy **50** applications in each environment.
    - In the **performance** and **production** environments, each Mule application will be deployed to **two** workers.
    - In other environments, the Mule application is only deployed to one worker.
    - The Production VPC should be properly sized to account for possible future growth so that the Production VPC does not need to be taken offline to recreate the VPC.

Calculate the number of IP addresses needed in the VPC

2. Based on the requirements, calculate the number of IPs per environment in each VPC

Environment and calculations	Production VPC	Non-production VPC
Dev	-	
Staging	-	
Performance	-	
Production		-
Total		
Additional IP for zero downtime deployment(50%)		
Total IPs		

Calculate a minimum CIDR network subnet that will meet the requirements

3. Go to https://www.ipaddressguide.com/cidr and find the CIDR network subnet to support the minimum required total number of IPs.
4. Write the CIDR block here.

5. Redo the calculations in the table to factor in suitable multiples of worst-case growth for the total number of future IPs, and write down the total number of IP addresses needed for future planning.

Environment and calculations	Production VPC	Non-production VPC
Dev	-	_____ x *100* =
Staging	-	
Performance	-	
Production	_____ x *100*	-
Total		
Additional IP for zero downtime deployment(50%)		
Total IPs		

6. Lookup and write down the CIDR block for all budgeted future growth.

Note: Remember that there is no charge for bigger CIDR sizes. Refer to https://docs.mulesoft.com/runtime-manager/vpc-provisioning-concept#faq_how_to_size_vpc to see one such recommendation.

Answer these reflection questions

7. Do you use TLS in your Mule apps? If so, what issues did you have with TLS?
8. Does your CI/CD tool provide any mechanism for managing TLS configuration?
9. How might secure properties be used in conjunction with keystores and truststores?
10. What are some alternatives to two-way TLS/HTTPS?