

Mulesoft MAC Workshop

Table of Contents

[Mulesoft MAC Workshop](#)

[Prerequisites](#)

[Signup for Free Trial MuleSoft Anypoint Platform](#)

[Download and Install Anypoint Studio](#)

[Create Anypoint Studio Workspace](#)

[Install Postman](#)

[Install Witsy](#)

[OpenAI API Key](#)

[Database Usernames/Passwords, Directory Paths and OpenAI API Key Configuration](#)

[Inference Hands On Exercises](#)

[Exercise 1 - Simple Chat App](#)

[Create a Mule Project](#)

[Add Dependency](#)

[Build Simple Chat Interaction](#)

[Run Your Project Locally](#)

[Test Your Mule Application](#)

[Exercise 2 - Enabling Chat History](#)

[Import and Explore Project](#)

[Test hands-on-mule-chat-history](#)

[Exercise 3 - Native MCP Support](#)

[Import and Explore Project](#)

[Test hands-on-mule-MCP](#)

[Exercise 4 - Implement A2A](#)

[Import and Explore Project](#)

[Test hands-on-mule-A2A](#)

[Vectors Hands On Exercises](#)

[Exercise 1 - Data Load](#)

[Import and Explore Project](#)

[Test hands-on-mule-vectors-app](#)

[Exercise 2 - Query Vector Store](#)

[Test hands-on-mule-vectors-app](#)

[Exercise 3 - Vector Store as a MCP Server](#)

[Test hands-on-mule-vectors-app](#)

[WebCrawler Hands On Exercises](#)

Exercise 1 - Get Static Page

[Import and Explore Project](#)

[Test hands-on-mule-webcrawler-app](#)

Exercise 2 - Get Dynamic Page

[Import and Explore Project](#)

[Test hands-on-mule-webcrawler-app](#)

Exercise 3 - Crawl a web page

[Import and Explore Project](#)

[Test hands-on-mule-webcrawler-app](#)

Exercise 3 - Crawl a web page and Add to Store

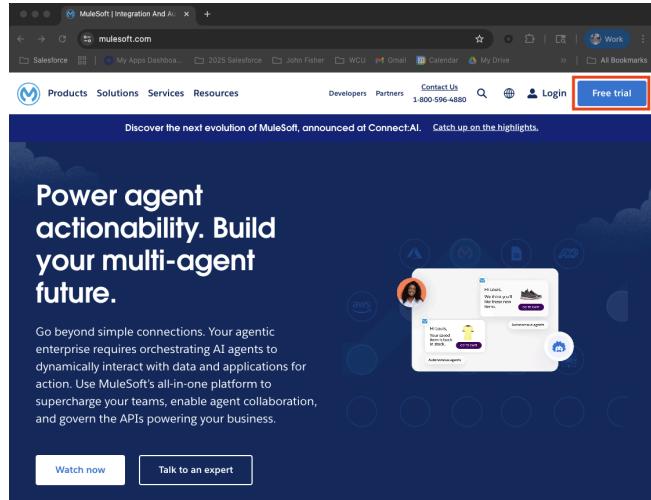
[Import and Explore Project](#)

[Test hands-on-mule-webcrawler-app](#)

Prerequisites

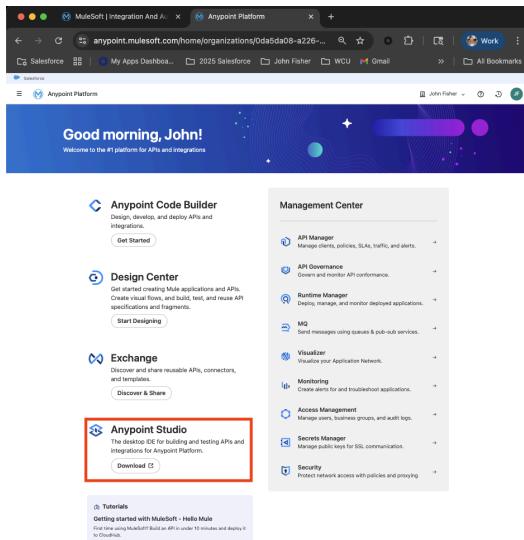
Signup for Free Trial MuleSoft Anypoint Platform

1. Sign up for a free trial of MuleSoft Anypoint Platform at www.mulesoft.com.

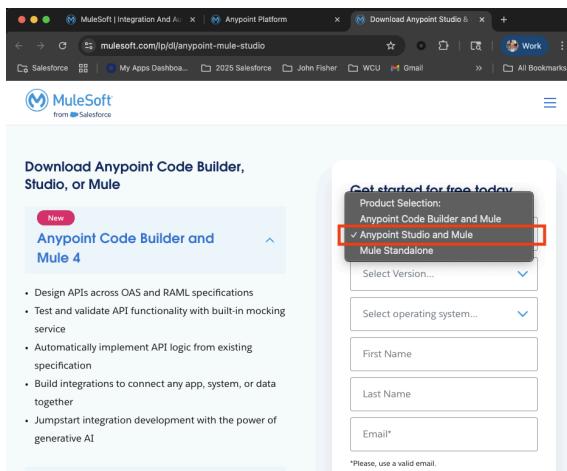


Download and Install Anypoint Studio

1. Using your credentials, Login to Anypoint Studio
2. Click Download to open the download Anypoint Code Builder, Studio, or Mule webpage.



3. Select **Anypoint Studio** and **Mule**, fill in fields with your information and press download.



4. Check your email for downloading Studio, and click **Go to download page**.

Your download link [External](#) [Inbox](#)

MuleSoft <noreply@www.mulesoft.com> 11:41 AM (0 minutes ago) [to me](#)

 MuleSoft

Thank you for choosing Studio.
Your download link is below.

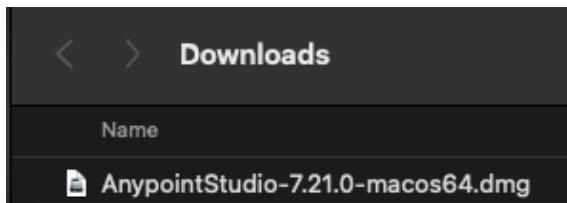
1. You can find installation instructions for your preferred platform on our [docs site](#).
2. Build your first API in 10 minutes with a step-by-step [tutorial](#).

[Go to download page](#)

The link above is valid only for 24 hours. You can request a new link [here](#)

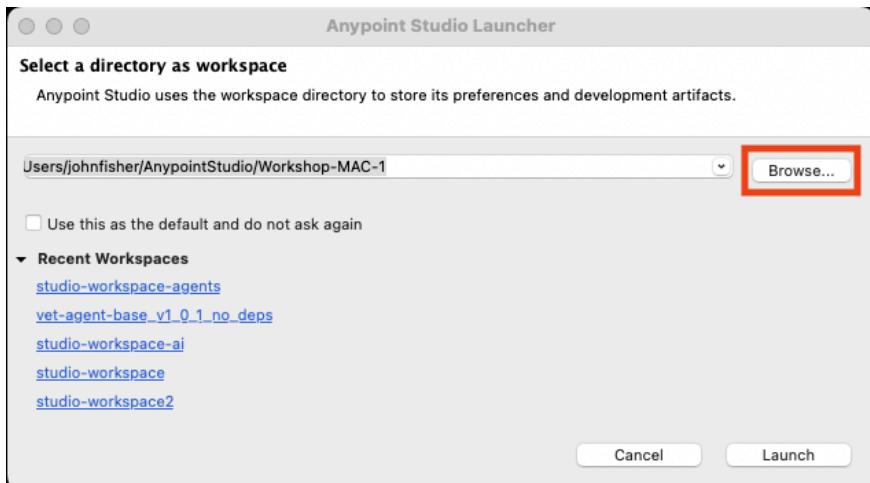
[Reply](#) [Forward](#)

5. Click **Download .dmg** on a Mac (Or other file type specific to your OS). Once downloaded find the dmg file (Or other file type specific to your OS) and double click on it to install.

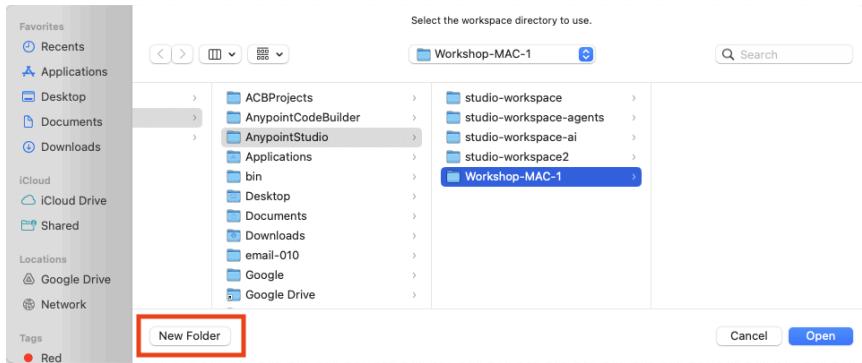


Create Anypoint Studio Workspace

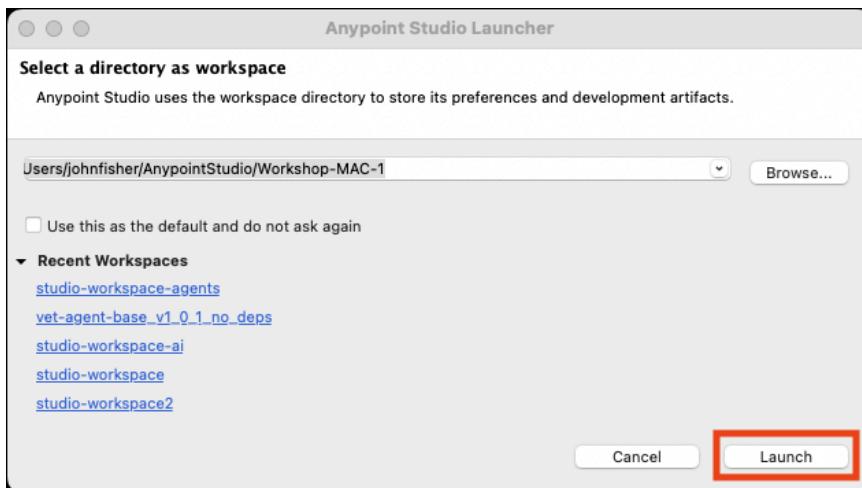
1. Open Anypoint Studio and create your workspace at directory path **<your path>/AnypointStudio/Workshop-MAC-1**. Click the **browse** button



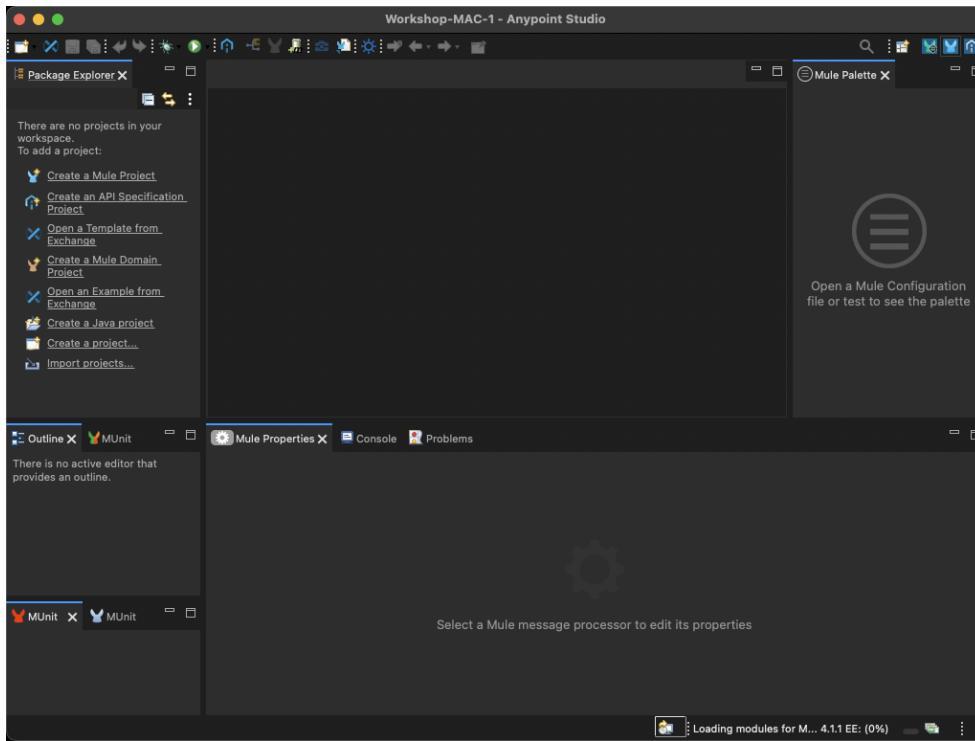
and Click **New Folder** to create a new folder in directory AnypointStudio (you may have to create this directory in your working directory first) named **Workshop-MAC-1**.



2. Click **Launch** to open Anypoint Studio.



3. Anypoint Studio will open, Workshop-MAC-1 workspace will be open, with a default view as depicted below.



Install Postman

1. Install Postman from <https://www.postman.com/>.

The screenshot shows the Postman homepage. At the top, there's a banner with the text "Are your APIs ready for AI agents? You don't need more models — you need stronger foundations. Explore AI resources →". Below the banner, there's a navigation bar with links for Product, Pricing, Enterprise, Resources and Support, API Network, Contact Sales, Sign In, and Sign Up for Free. The main headline reads "AI needs context. APIs deliver it." Below the headline, there's a paragraph about Postman's support for the Model Context Protocol (MCP). There are two buttons: "Sign Up for Free" and "Watch a Demo". Further down, there's a section for downloading the desktop app, with icons for Windows, macOS, and Linux. To the right, there's a graphic of three 3D cubes (orange, black, and purple) stacked together, with the letters "API" visible on them.

2. Download the **Workshop-SME-MAC Capabilities.json** from [here](#) and save to your desktop.
3. Import Workshop-SME-Mac Capabilities collection into Postman by dragging and dropping **Workshop-SME-MAC Capabilities.json** into Collections canvas. Postman Workspace should look something like the screenshot below.

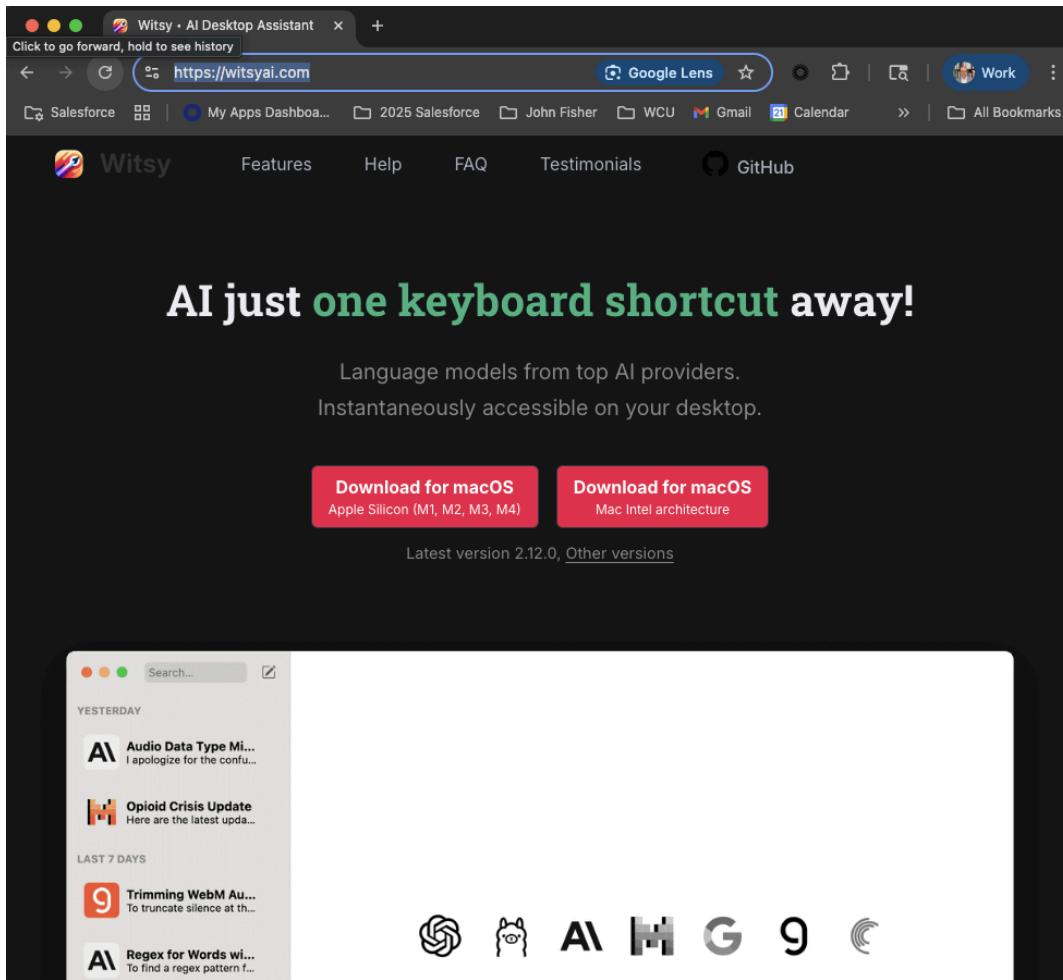
⌄ Workshop-SME-MAC Capabilities

```
POST WhatIsCapital  
POST WhatsMyName  
POST MyNamels  
POST MCPYouAreHelpful  
POST MCPCheckInventory  
POST MCPGetAllCRMAccounts  
POST A2AWhatIsCapital  
GET A2AGetAgentCard  
POST VectorAdd  
POST VectorQuery  
POST WebCrawlerGetStatic  
POST WebCrawlerGetDynamic  
POST WebCrawlerCrawl  
POST WebCrawlerAddToStore  
POST localhost:8081/inference/chat
```

Q Find and replace ⌂ Console 2 Import Complete

Install Witsy

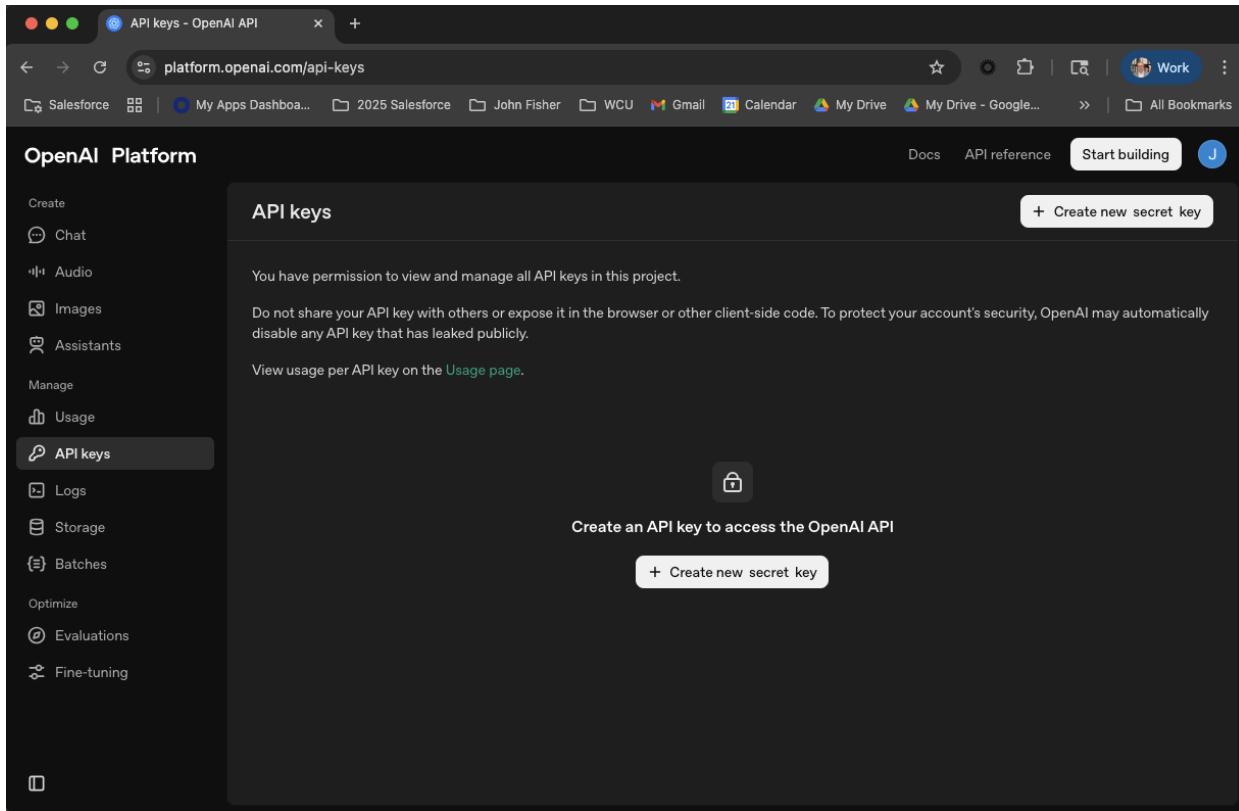
1. Install Witsy from <https://witsyai.com/>.



OpenAI API Key

OpenAI API keys allow developers to access OpenAI's models through the API and build applications that can generate text, images, and code. To get an OpenAI API key, developers need to sign up for an OpenAI account and request an API key from the OpenAI dashboard. Developers should keep their API key secret, monitor their usage, handle errors gracefully, and test their applications thoroughly before deploying them to production.

1. To get an OpenAI API key, developers need to sign up for an OpenAI account and request an API key from the OpenAI Platform dashboard.



Database Usernames/Passwords, Directory Paths and OpenAI API Key Configuration

All MuleSoft project files have stored database usernames and passwords, OpenAI API Key and directory paths in a config.yaml file located in your project path

<Project>->src/main/resources->config.yaml. Update these entries to reflect your database user name and passwords, directory paths and OpenAI API key. See screenshot below for reference.

The screenshot shows the Eclipse IDE interface. The Package Explorer view on the left lists several Mule projects: hands-on-mule-A2A, hands-on-mule-chat-history, hands-on-mule-inference, hands-on-mule-MCP, hands-on-vectors-app, and hands-on-webcrawler-app. The hands-on-webcrawler-app project is expanded, showing its structure: src/main/mule (Flows), src/main/java, src/main/resources (application-types.xml, config.yaml, log4j2.xml, Shadows_Over_Carrington.pdf, api), and src/test/java. The config.yaml file is selected in the resources folder and is displayed in the main editor area. The code in config.yaml is as follows:

```
1 @apikey: "sk_proj-Moe4kODA-i6ZdmRfi9RkK2ia47Et0ApLt4sQl_uJAb8ApzbFXbZQl9Ume1c"
2 dbhost: "<REDACTED>"
3 dbport: "<REDACTED>"
4 dbdatabase: "sampledb"
5 dbuser: "postgres"
6 dbpassword: "<REDACTED>"
7 downloaddir: "/Users/johnfisher/Desktop/mac-project.ai"
```

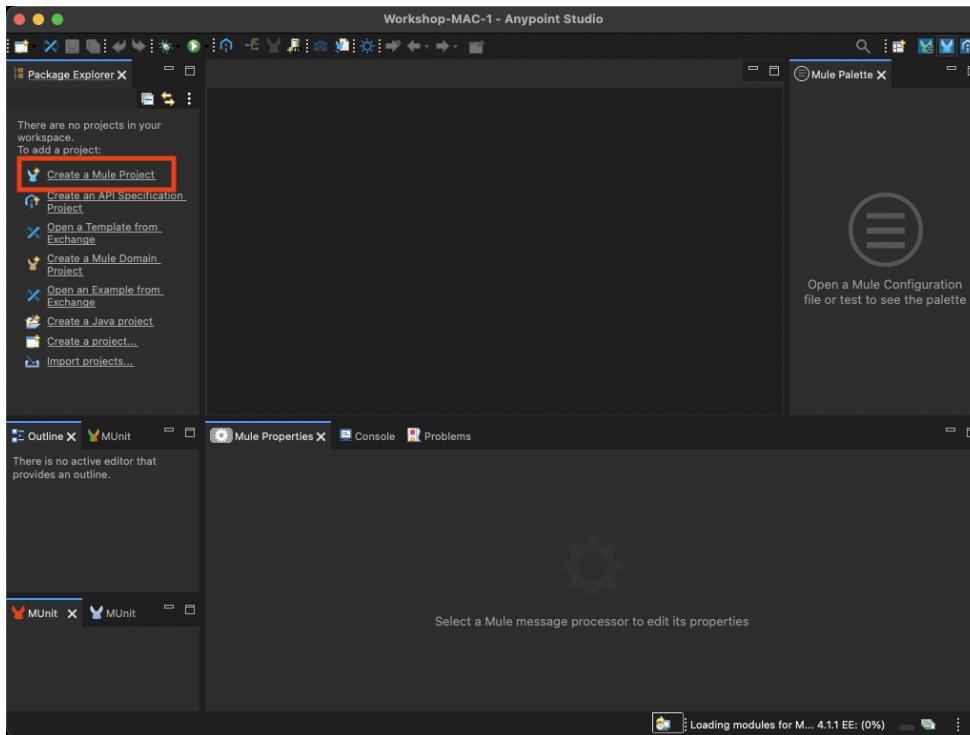
Inference Hands On Exercises

Exercise 1 - Simple Chat App

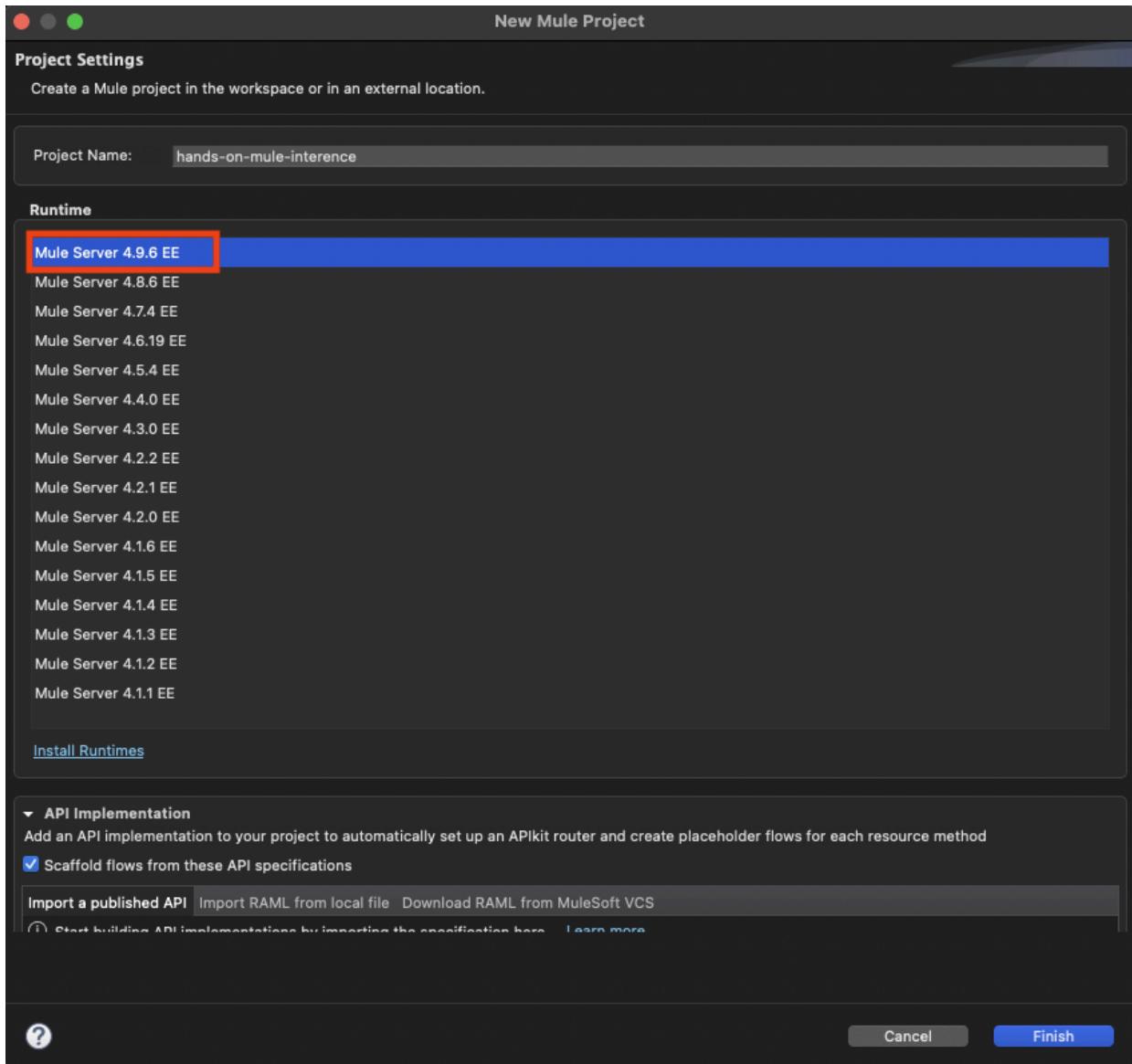
The purpose of this exercise is to demonstrate how you can use the chat completion operation of LLM to implement a workflow in MuleSoft. In this exercise we are going to build a simple headless chat app with an LLM.

Create a Mule Project

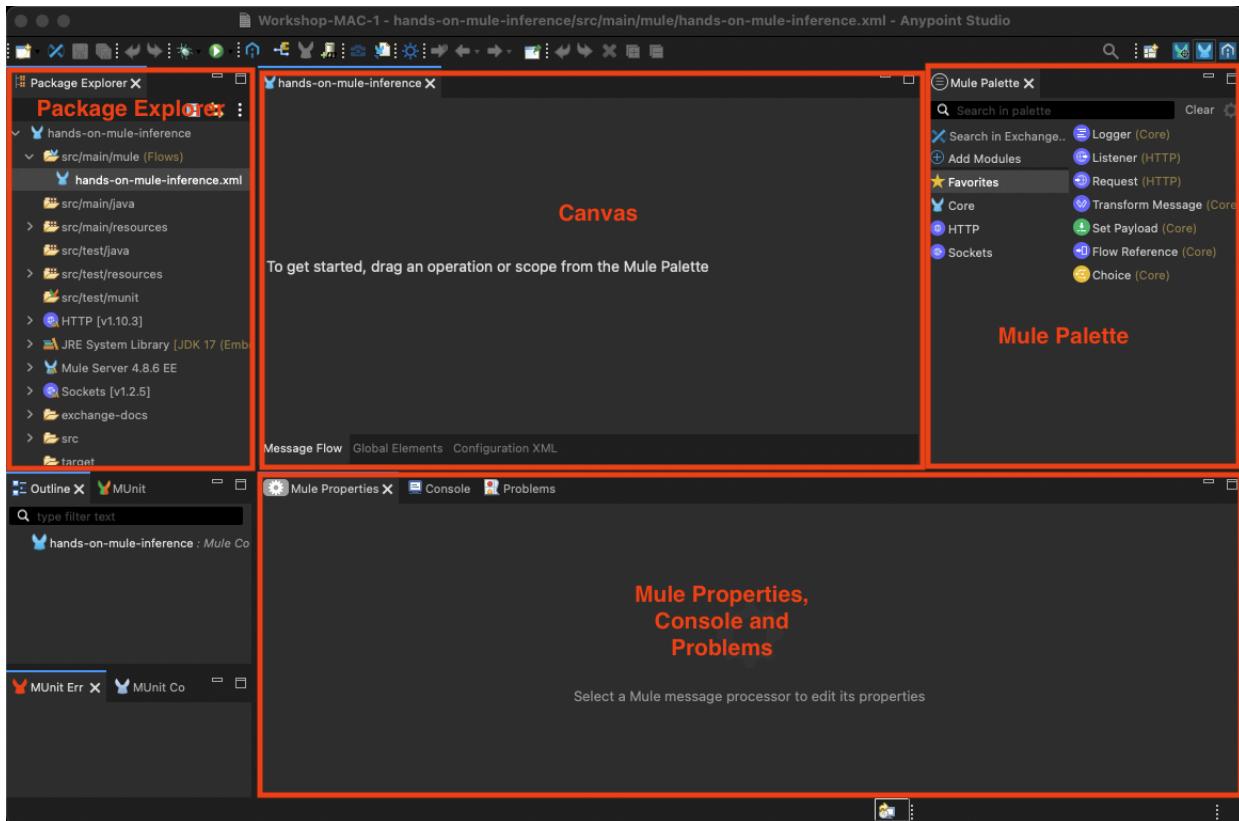
1. Create a Mule Project hands-on-mule-inference by selecting **Create a Mule Project** to add a project to your workspace.



2. Enter project name, **hands-on-mule-inference**, select **Mule Server 4.9.X**, and press **Finish**.

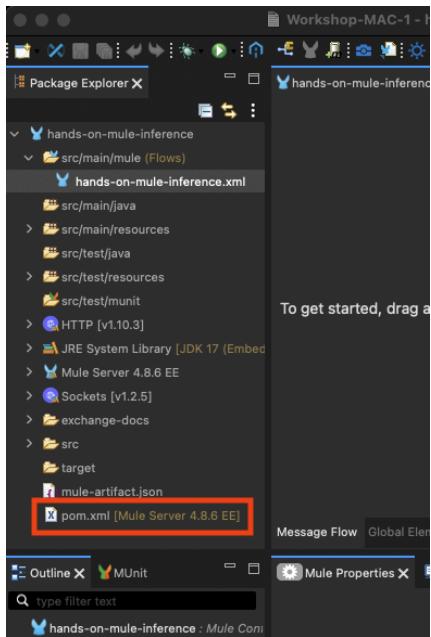


Your hands-on-mule-interface project will be created and your design canvas will be opened (see below image), package explorer on left will show your projects assets, you will see the Mule Palette on the right with components and activities, and towards the bottom is the mule properties, console and problems.



Add Dependency

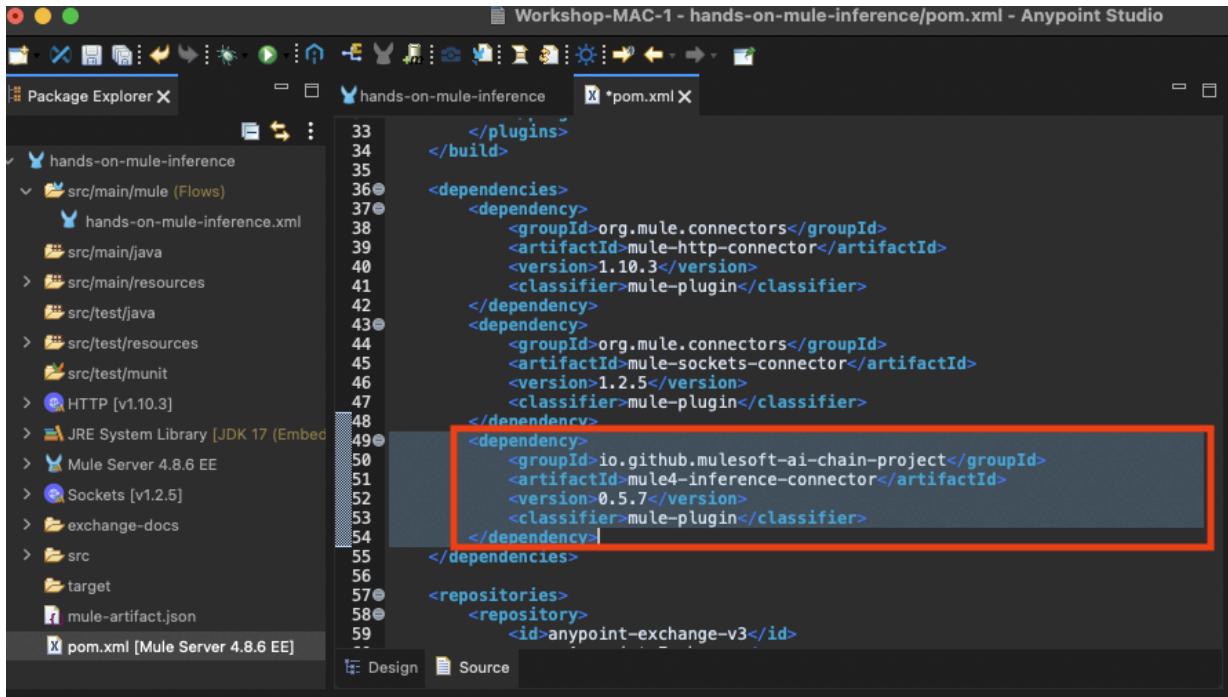
1. Double-click on **pom.xml** in the package explorer to open it.



2. Copy the following dependency XML entry.

```
<dependency>
    <groupId>io.github.mulesoft-ai-chain-project</groupId>
    <artifactId>mule4-inference-connector</artifactId>
    <version>0.5.7</version>
    <classifier>mule-plugin</classifier>
</dependency>
```

3. Paste the copied dependency XML entry immediately after the last </dependency> tag with the <dependencies> block (lines 49-54 in the screenshot below).

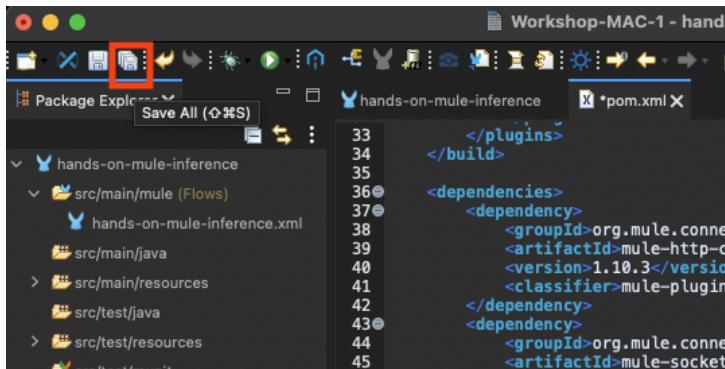


```
</plugins>
</build>
<dependencies>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-http-connector</artifactId>
        <version>1.10.3</version>
        <classifier>mule-plugin</classifier>
    </dependency>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-sockets-connector</artifactId>
        <version>1.2.5</version>
        <classifier>mule-plugin</classifier>
    </dependency>
    <dependency>
        <groupId>io.github.mulesoft-ai-chain-project</groupId>
        <artifactId>mule4-inference-connector</artifactId>
        <version>0.5.7</version>
        <classifier>mule-plugin</classifier>
    </dependency>
</dependencies>
<repositories>
    <repository>
        <id>anypoint-exchange-v3</id>

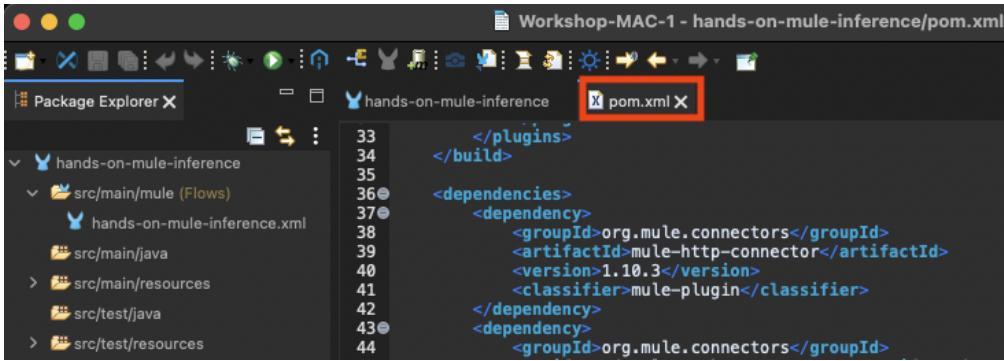
```

4. Click **save all icon** (highlighted in the screenshot) in the toolbar, or press ⌘S (macOS) / Ctrl+S (Windows/Linux) to save your project. This will add the inference connector to your project.

Note: Once Inference Connector is Generally Available (GA), it will be accessible via Exchange.



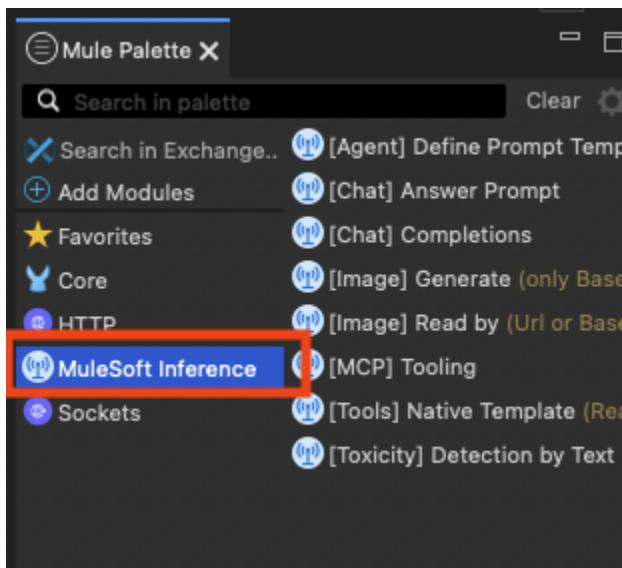
5. Close pom.xml by clicking on the 'X' icon on its tab (highlighted in the screenshot below).



The screenshot shows the Eclipse IDE interface with the title bar "Workshop-MAC-1 - hands-on-mule-inference/pom.xml". The left sidebar is the "Package Explorer" showing a project named "hands-on-mule-inference" with several source folders like "src/main/mule (Flows)" and "src/main/java". The main editor area displays the "pom.xml" file with the following code:

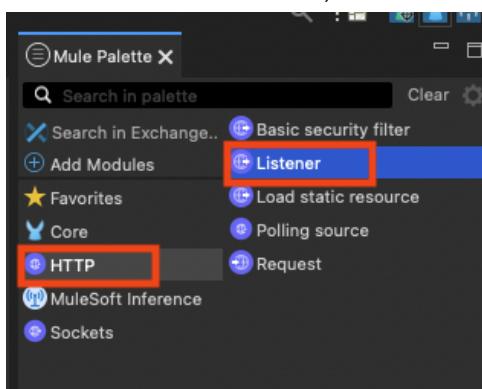
```
</build>
<dependencies>
    <dependency>
        <groupId>org.mule.connectors</groupId>
        <artifactId>mule-http-connector</artifactId>
        <version>1.10.3</version>
        <classifier>mule-plugin</classifier>
    </dependency>
    <dependency>
        <groupId>org.mule.connectors</groupId>
```

6. Verify “MuleSoft Inference” appears in the Mule Palette (highlighted in the screenshot below).

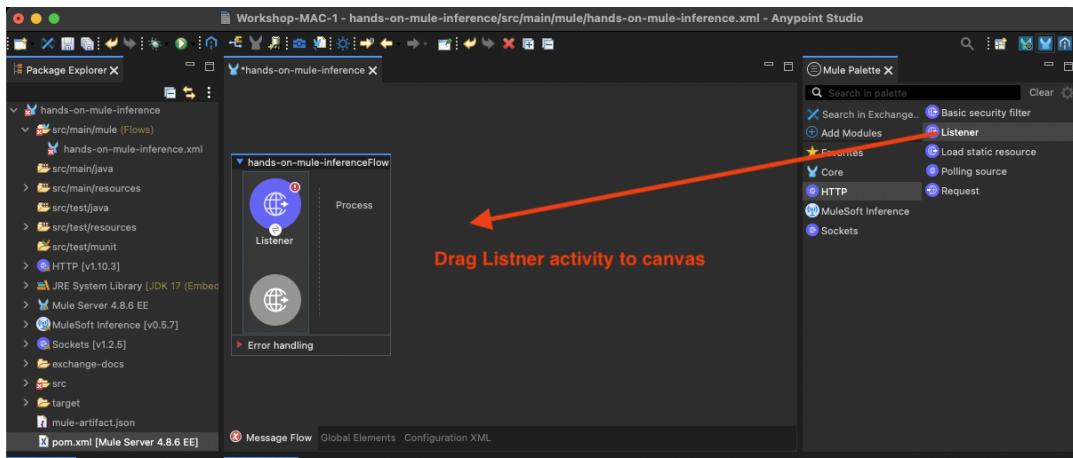


Build Simple Chat Interaction

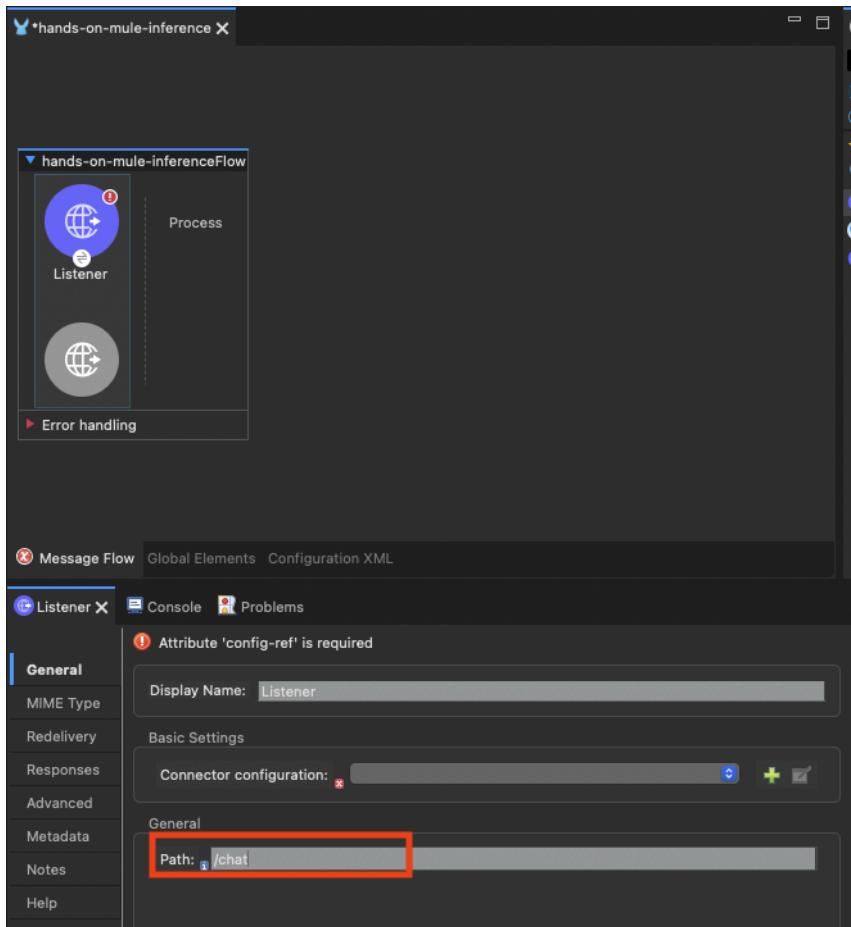
1. From the Mule Palette, select **HTTP**, then



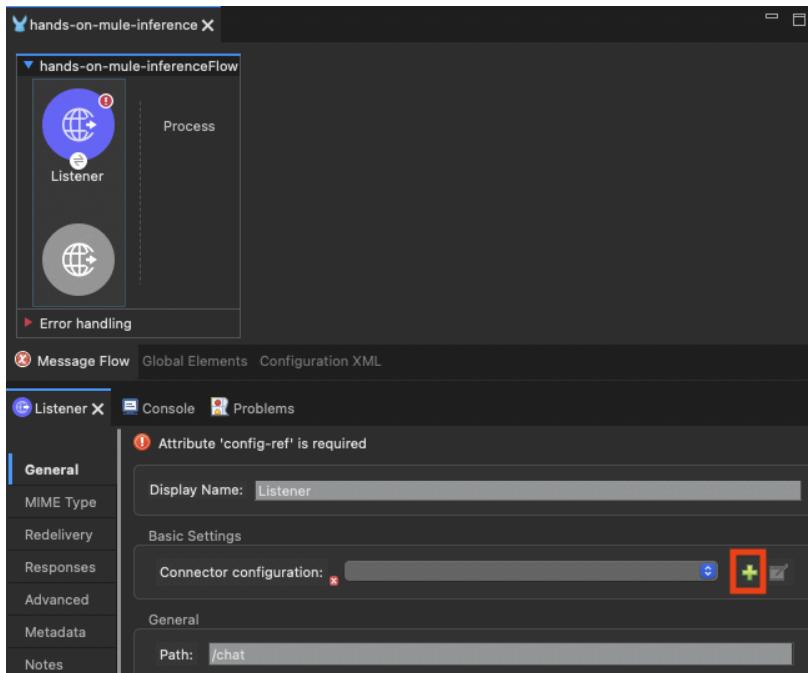
drag the **Listener** activity to your design canvas.



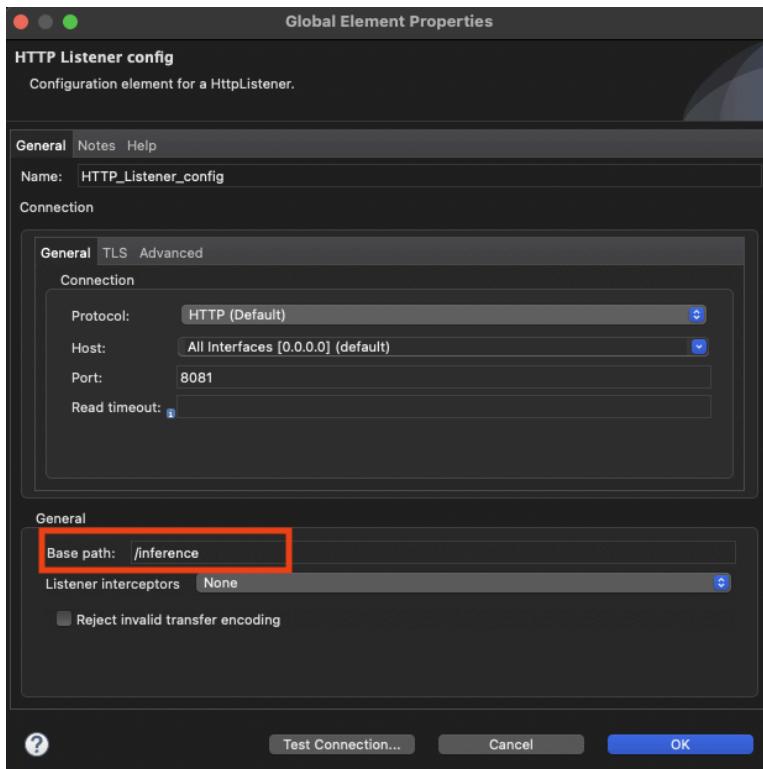
2. In the Listener's configuration, set the **Path** to /chat (as highlighted in the screenshot).



3. To create a connector configuration, click the green '+' icon next to "Connector configuration" (highlighted in the screenshot). This will open the HTTP Listener config.

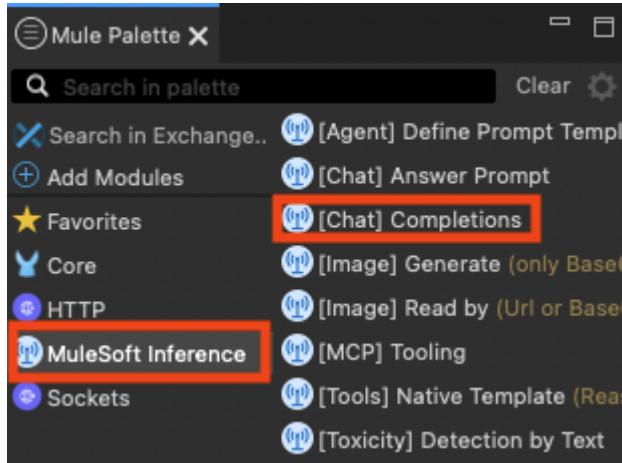


4. In HTTP Listener config, set the **Base path** to /inference (as highlighted in the screenshot). Click on **OK** to close the HTTP Listener config dialog.

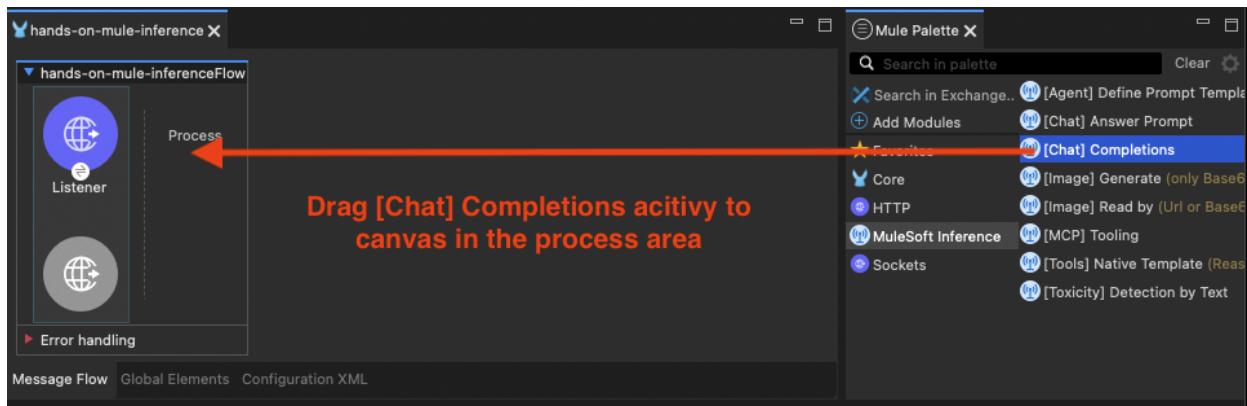


5. Save your project.

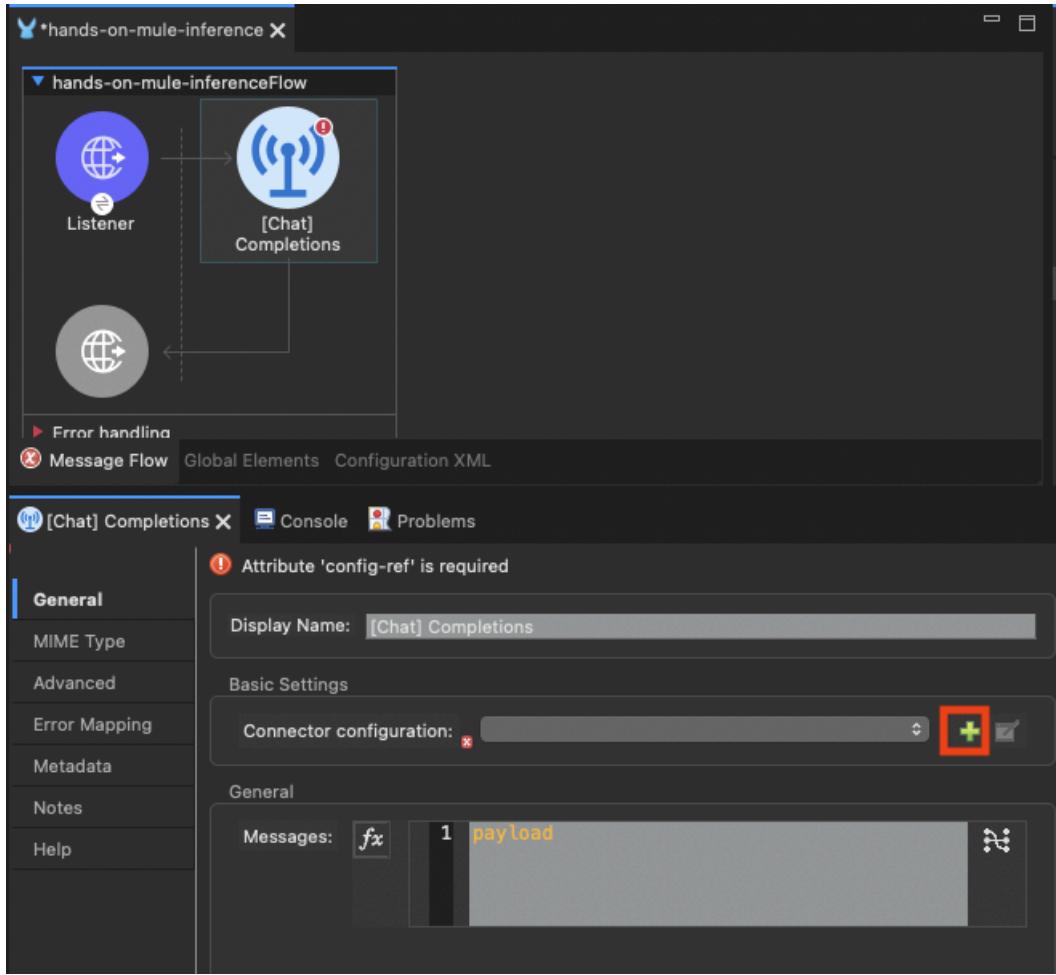
6. From the Mule Palette, select **MuleSoft Inference**,



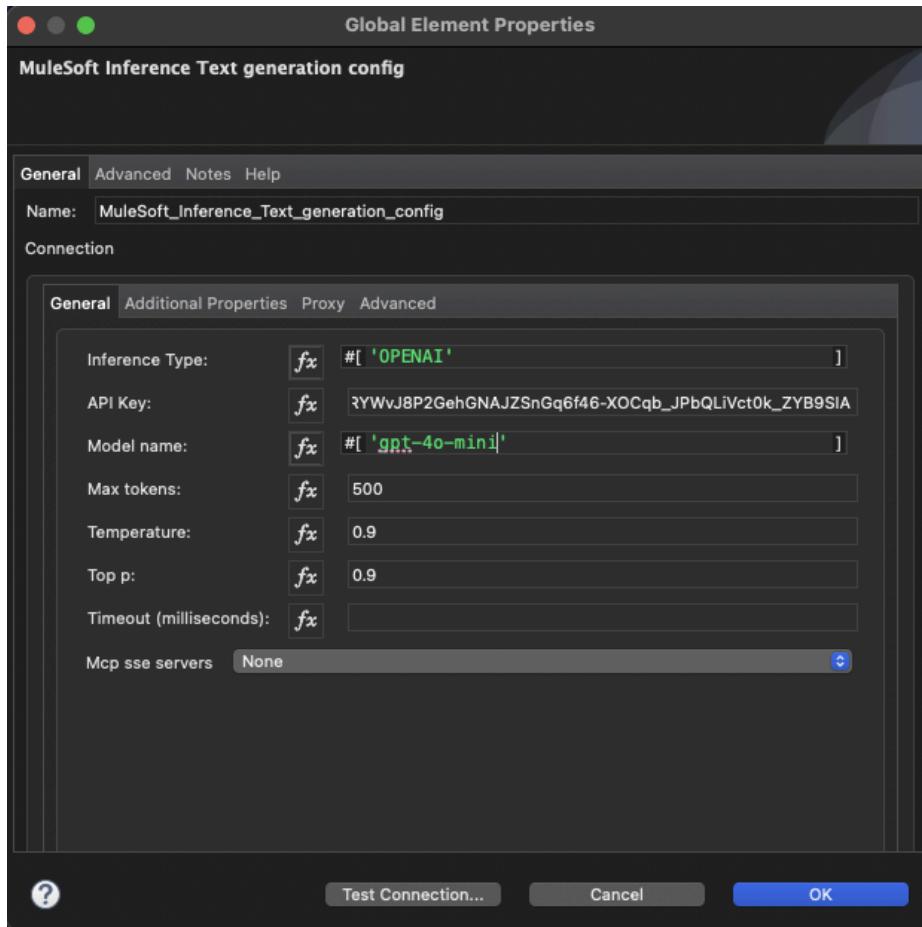
then drag the **[Chat] Completions** activity to your design canvas and drop in the **Process** area.



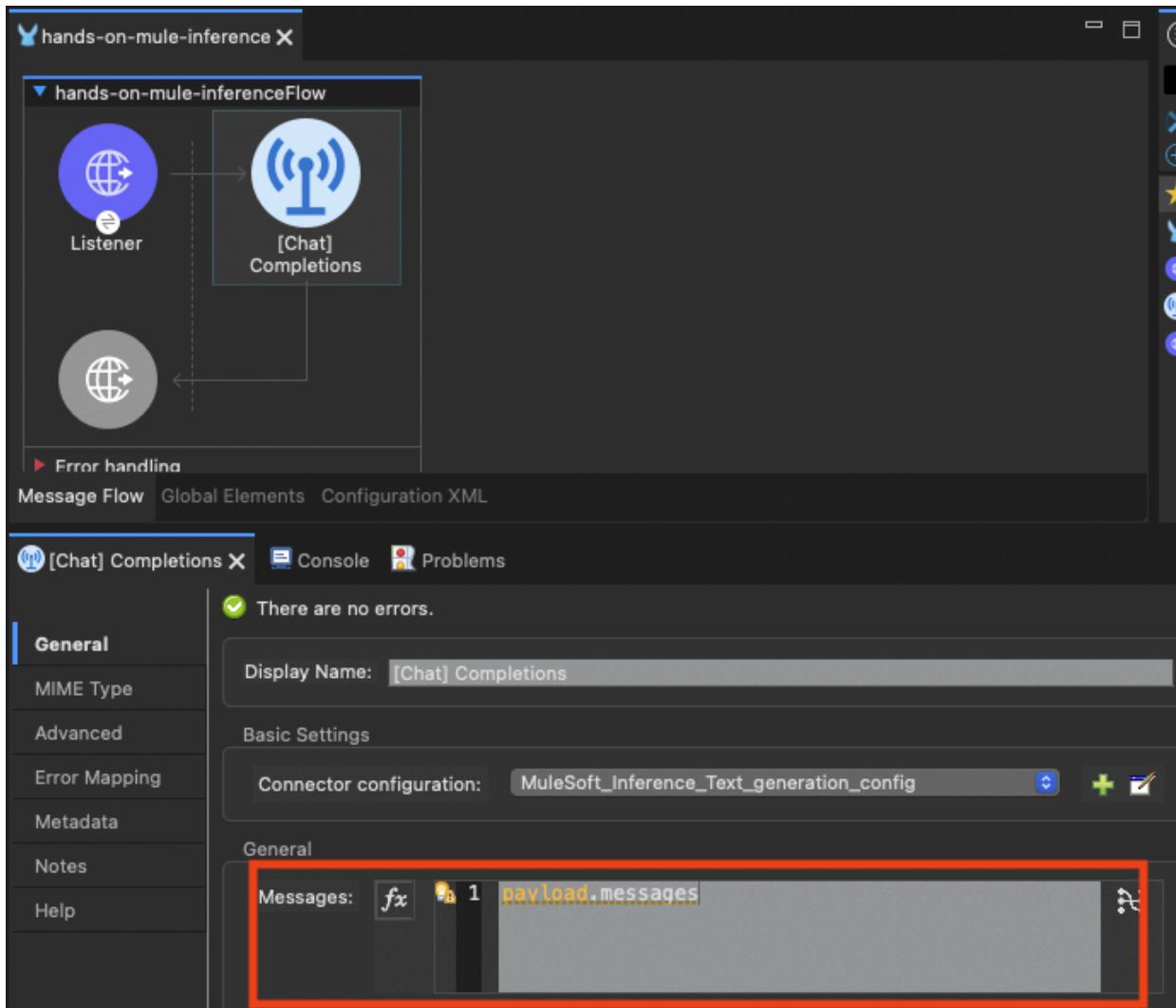
7. To create a connector configuration, click the green '+' icon next to "Connector configuration" (highlighted in the screenshot). This will open the MuleSoft Inference Test generation config.



8. In MuleSoft Inference Test generation config, set the **Inference Type, API Key and Model name** as shown in the screenshot below. To enter Inference Type and Model name press 'fx', to open expression mode, next to each entry. Click **OK** to close.

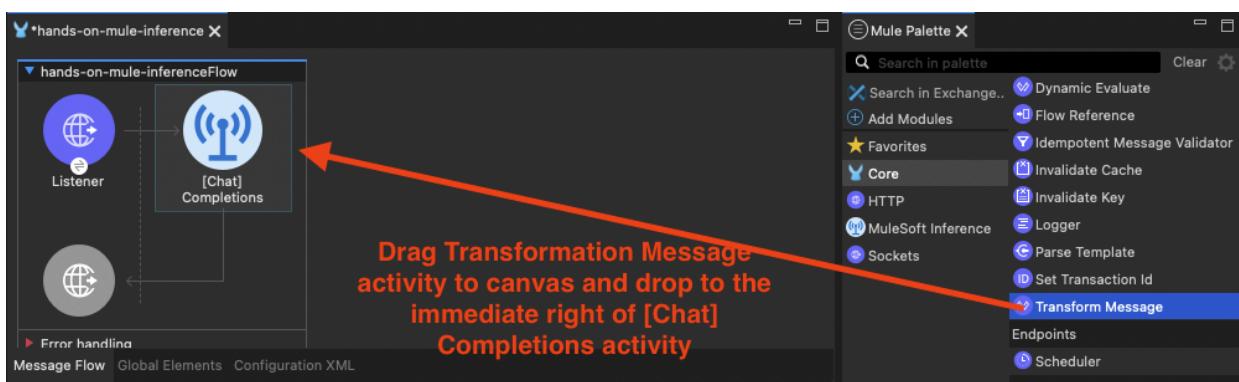


9. Update **Messages** to payload.messages as shown in the screenshot below.



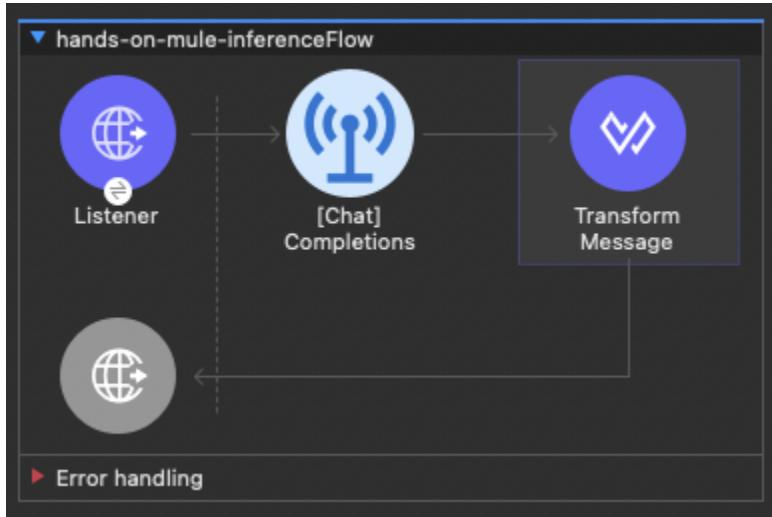
10. Save your project.

11. From the Mule Palette, select **Core**,



then drag the **Transform Message** activity to your design canvas and drop it to the right of the [Chat] Completions activity.

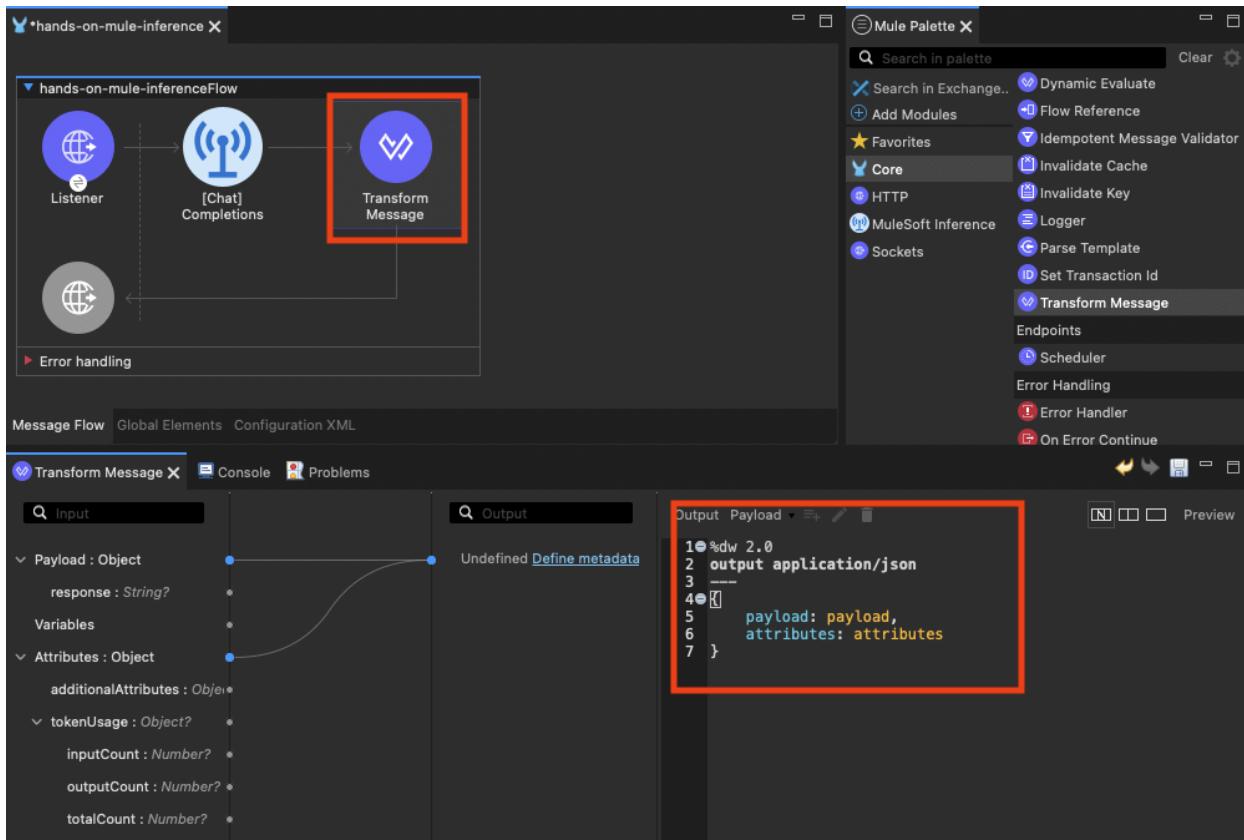
Your flow should look like this.



12. Select **Transform Message** activity that you just added, copy following dataweave expression

```
%dw 2.0
output application/json
---
{
    payload: payload,
    attributes: attributes
}
```

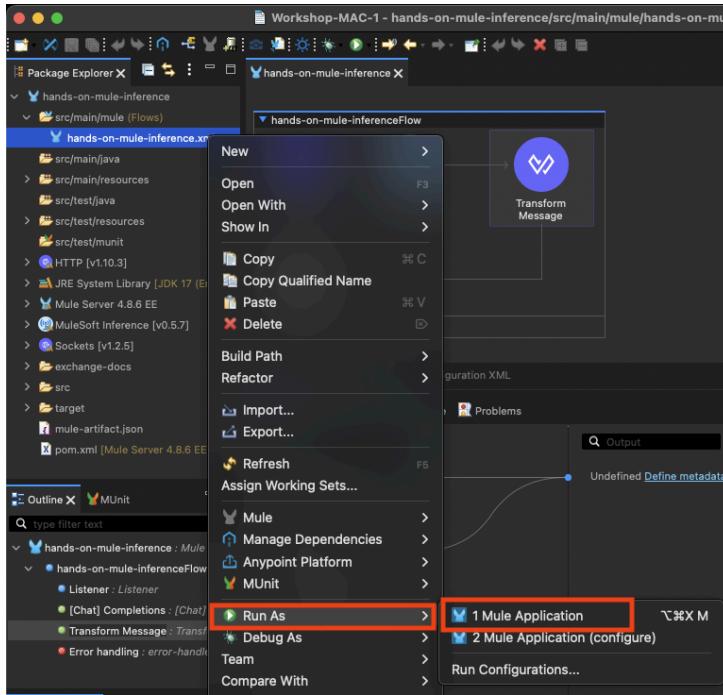
and paste the expression into the **output payload** as shown in the screenshot below.



13. Save your project.

Run Your Project Locally

1. To run the project locally, from the project explorer, right-click on `hands-on-mule-inference.xml`, select Run-As, select Mule Application as shown in the screenshot below.



2. Your mule project will be started. Please wait for application status deployed in the console window as shown in screenshot below.

```

Mule Properties  Console X  Problems
hands-on-mule-inference [Mule Applications] [pid: 64045]
+-----+
+ Mule is up and kicking (every 5000ms) +
+-----+
INFO 2025-07-21 15:24:15,014 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: St
INFO 2025-07-21 15:24:15,015 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging:
*-----*
* - + DOMAIN + - - * - + STATUS + - - *
*-----*
* default * DEPLOYED *
*-----*

*-----*
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*-----*
* hands-on-mule-inference * default * DEPLOYED *
*-----*

INFO 2025-07-21 15:24:15,026 [[MuleRuntime].uber.14: [hands-on-mule-inference].uber@org.mule.runtime.modul

```

Test Your Mule Application

1. Open Postman and select **WhatIsCapital** request.

The screenshot shows the Postman interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Agent - REX', 'Anomaly Detection', 'Demystify Mule AI Chain', 'Ford Use Cases', 'MCP', 'MCP Copy', and 'Workshop-SME-MAC Capabilities'. Under 'Workshop-SME-MAC Capabilities', several items are listed: 'GET WhatsCapital', 'GET WhatsMyName', 'GET MyNames', 'GET MCPYouAreHelpful', 'GET MCPCheckInventory', 'GET MCPGetAllCRMAccounts', and 'GET A2AWhatsCapital'. The 'GET WhatsCapital' item is highlighted with a red box. The main workspace shows a POST request to 'localhost:8081/inference/chat'. The 'Body' tab is selected, showing the following JSON payload:

```
1 {
2     "messages": [
3         {
4             "role": "user",
5             "content": "What is the capital of Switzerland?"
6         }
7     ]
8 }
```

The response status is '200 OK' with a time of '1.65 s' and a size of '482 B'. Below the body, there are tabs for Body, Cookies, Headers (3), Test Results, and a preview of the JSON response.

2. Click **Send** to Post.

This screenshot is similar to the previous one but focuses on the 'Send' button. The 'Send' button is highlighted with a red box. The rest of the interface and the JSON payload remain the same as in the first screenshot.

3. MuleSoft will respond with the following response.

The screenshot shows the Postman interface with a successful API call. The URL is `localhost:8081/inference/chat`. The request method is POST. The Body tab shows the following JSON payload:

```

1 {
2   "messages": [
3     {
4       "role": "user",
5       "content": "What is the capital of Switzerland?"
6     }
7   ]
8 }
9

```

The response status is 200 OK, with a response time of 1.65 s and a size of 482 B. The response body is:

```

1 {
2   "payload": {
3     "response": "The capital of Switzerland is Bern."
4   },
5   "attributes": {
6     "tokenUsage": {
7       "outputCount": 7,
8       "totalCount": 21,
9       "inputCount": 14
10    },
11    "additionalAttributes": {
12      "finish_reason": "stop",
13      "model": "gpt-4o-mini-2024-07-18",
14      "id": "chatcmpl-BvqHNPirkn6mfJAssI09JJwhZbtRQ"
15    }
16  }
17 }

```

4. Stop your mule application by clicking on ‘RED’ terminate icon.

The screenshot shows the Anypoint Studio interface with the Mule flow editor open. The flow is named `hands-on-mule-inferenceFlow` and consists of three components: Listener, [Chat] Completions, and Transform Message. The Mule Palette on the right lists various Mule components like Listener (HTTP), Request (HTTP), Transform Message (Core), and MuleSoft Inference.

The bottom pane shows the deployment log for the application `hands-on-mule-inference`:

```

+ Mule is up and kicking (every 5000ms)
INFO 2025-07-21 16:16:26,394 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector@7db2f255
INFO 2025-07-21 16:16:26,394 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging:
* -- + DOMAIN + - - * -- + STATUS + - - *
* hands-on-mule-inference           * default          * DEPLOYED      *
* -- + APPLICATION + - - * -- + DOMAIN + - - * -- + STATUS + - - *
* hands-on-mule-inference           * default          * DEPLOYED      *
INFO 2025-07-21 16:16:407 [[MuleRuntime].uber.06: [hands-on-mule-inference].uber@org.mule.runtime.module.extension.internal.runtime.s

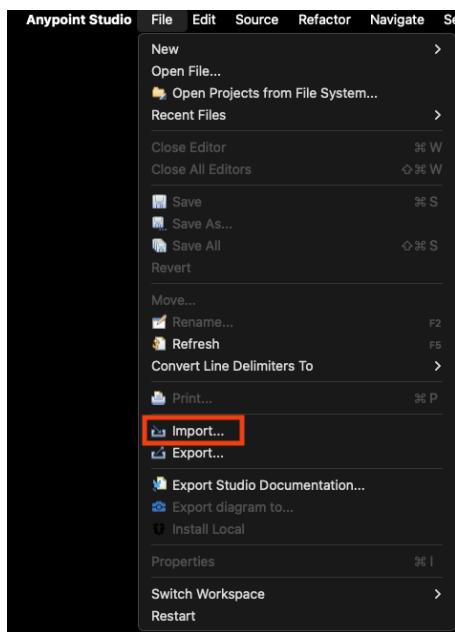
```

Exercise 2 - Enabling Chat History

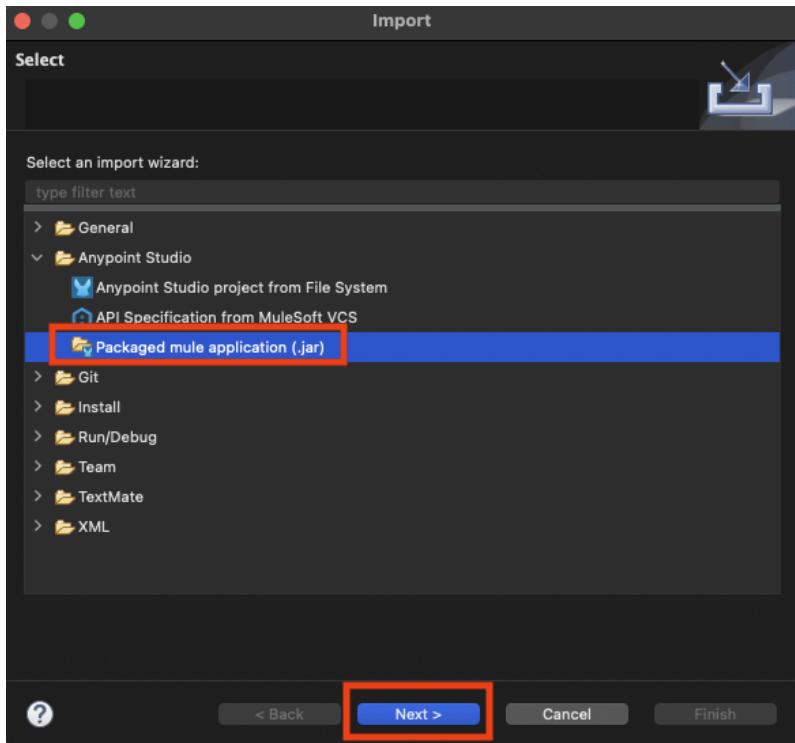
When using Chat Completions activity, providing chat history is crucial for maintaining conversational context. You have control over which past interactions to include to optimize performance and relevance.

Import and Explore Project

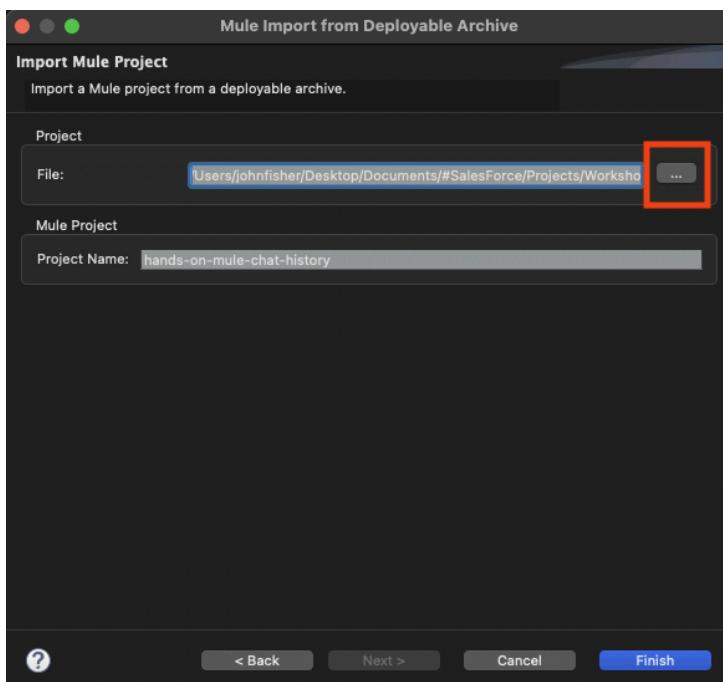
1. Import **hands-on-mule-chat-history** into your workspace. Select File->Import...



2. Expand **Anypoint Studio**, select **Packaged mule application (.jar)**, click **next**.

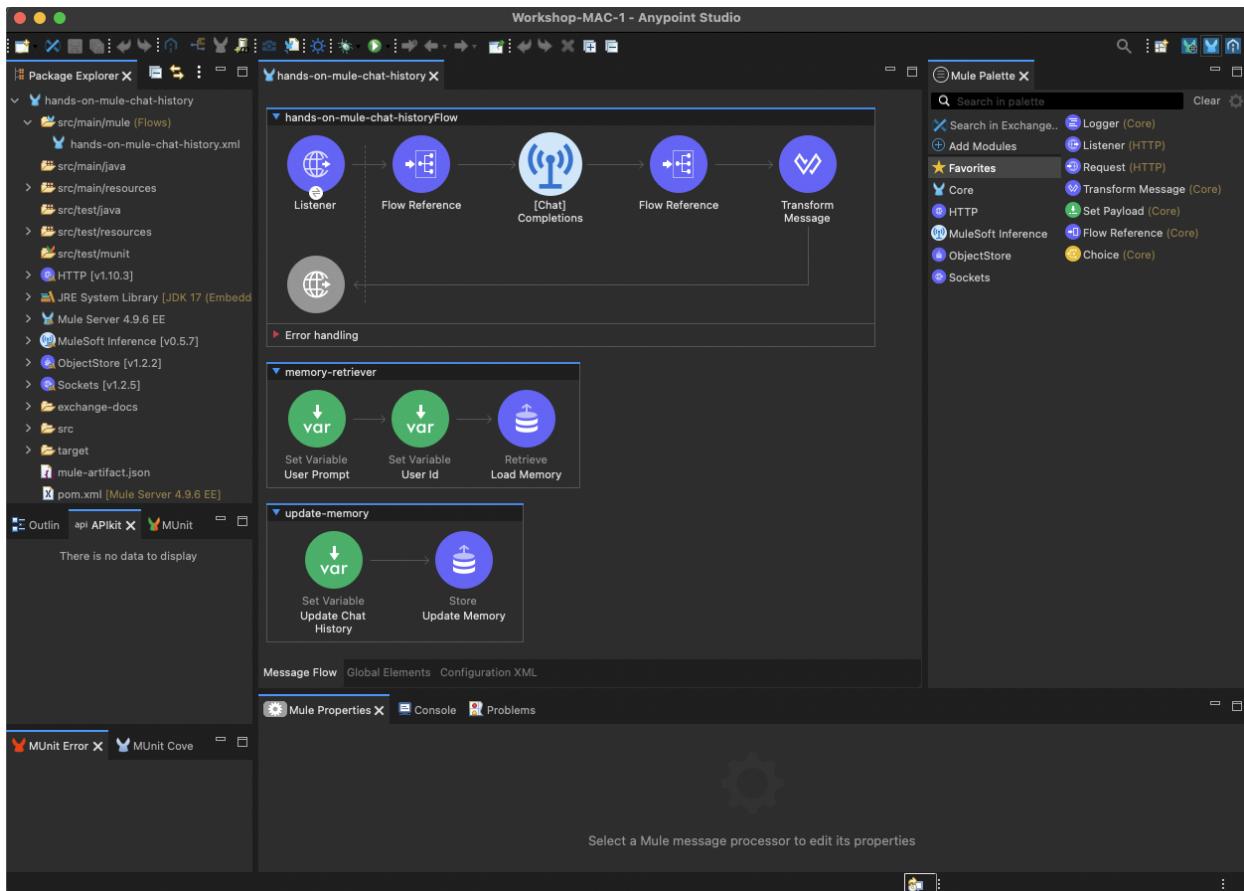


3. Click the “...” icon and browse to the directory of **hands-on-mule-chat-history.jar**, select hands-on-mule-chat-history.jar and click **open**.

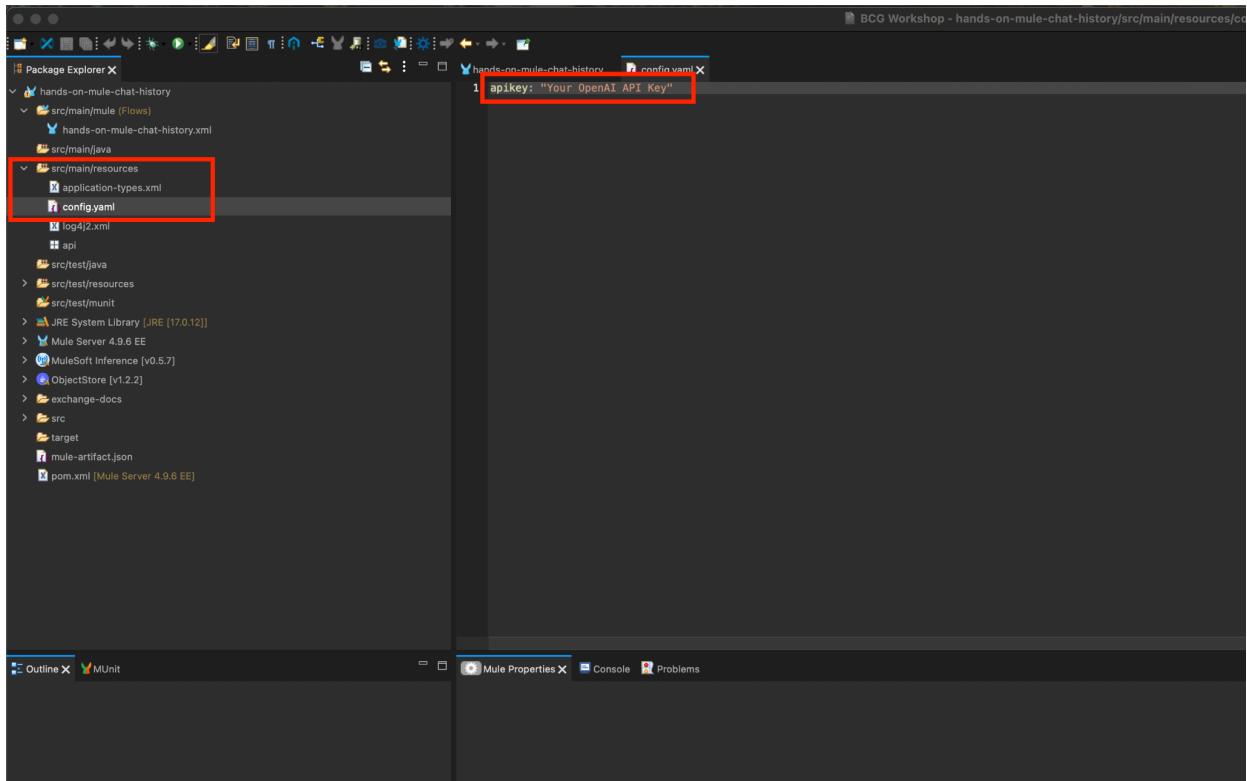


4. Click Finish.

5. Explore the project.



3. Replace the “Your OpenAI API Key” with your OpenAI API Key in the config.json. Put your API key between the quotes.



4. Run hands-on-mule-chat-history locally.

Test hands-on-mule-chat-history

1. Open Postman and select **WhatsMyName** request and click **Send**. The Agent doesn't know your name so you get a "I'm sorry" response as shown below.

Note: Each time your test WhatsMyName and MyName1s change the **userID** found in Headers to a different name and update the content.

POST localhost:8081/inference/chat

Params Authorization Headers (10) Body Scripts Settings

Headers (1) 9 hidden

Key	Value	Description
<input checked="" type="checkbox"/> userID	Jimmy	
Key	Value	Description

HTTP Workshop-SME-MAC Capabilities / MyNames

POST localhost:8081/inference/chat

Params Authorization Headers (10) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL **JSON**

```
1 {  
2   "messages": [  
3     {  
4       "role": "user",  
5       "content": "My name is Jack"  
6     }  
7   ]  
8 }
```

The screenshot shows the Postman interface with the following details:

- Request URL:** `localhost:8081/inference/chat`
- Method:** `POST`
- Body Content:**

```
1 {
2   "messages": [
3     {
4       "role": "user",
5       "content": "Whats my name"
6     }
7   ]
8 }
```
- Response Status:** `200 OK`
- Response Headers:** `1.78 s • 564 B`
- Response Body (JSON):**

```
1 {
2   "payload": {
3     "response": "I'm sorry, but I don't have access to personal information about users, including names. How can I assist you today?"
4   },
5   "attributes": {
6     "tokenUsage": {
7       "outputCount": 24,
8       "totalCount": 34,
9       "inputCount": 10
10    },
11    "additionalAttributes": {
12      "finish_reason": "stop",
13      "model": "gpt-4o-mini-2024-07-18",
14      "id": "chatmpl-Bvs17rjhQDejgKipmdSgwfALcthRz"
15    }
16  }
17 }
```

2. In Postman select **MyNameIs** request and click **Send**. Now that you provided your name, the user name will be stored and remembered.

The screenshot shows the Postman interface with the following details:

- Request URL:** POST localhost:8081/inference/chat
- Body Content (Raw JSON):**

```
1 {
2     "messages": [
3         {
4             "role": "user",
5             "content": "My name is Jimmy"
6         }
7     ]
8 }
```

- Response Status:** 200 OK
- Response Headers:** 915 ms, 498 B
- Response Body (Raw JSON):**

```
1 {
2     "payload": {
3         "response": "Nice to meet you, Jimmy! How can I help you today?"
4     },
5     "attributes": {
6         "tokenUsage": {
7             "outputCount": 14,
8             "totalCount": 60,
9             "inputCount": 46
10        },
11        "additionalAttributes": {
12            "finish_reason": "stop",
13            "model": "gpt-4o-mini-2024-07-18",
14            "id": "chatcmpl-Bvsn8MERbnfAclxRmJxXHgw7pwt6S"
15        }
16    }
17 }
```

3. In Postman select **WhatsMyName** (for a second time) request and click **Send**. Since you provided your name in step 2 above, your name has been remembered.

The screenshot shows the Postman interface with the following details:

- Request URL:** `localhost:8081/inference/chat`
- Method:** POST
- Body Content:**

```

1 {
2   "messages": [
3     {
4       "role": "user",
5       "content": "Whats my name"
6     }
7   ]
8 }
```
- Response Status:** 200 OK
- Response Headers:** 565 ms, 497 B
- Response Body (JSON):**

```

1 {
2   "payload": {
3     "response": "Your name is Jimmy. How can I assist you further?"
4   },
5   "attributes": {
6     "tokenUsage": {
7       "outputCount": 12,
8       "totalCount": 83,
9       "inputCount": 71
10    },
11    "additionalAttributes": {
12      "finish_reason": "stop",
13      "model": "gpt-4o-mini-2024-07-18",
14      "id": "chatcmpl-Bvsnhw3upDNEYGvyHN0E3EA5Llib"
15    }
16  }
17 }
```

Exercise 3 - Native MCP Support

For this exercise, we are going to explore MCP server usage. Mule Inference comes with an out-of-the-box MCP client implementation, which allows users to configure one or more MCP servers to discover and execute tools.

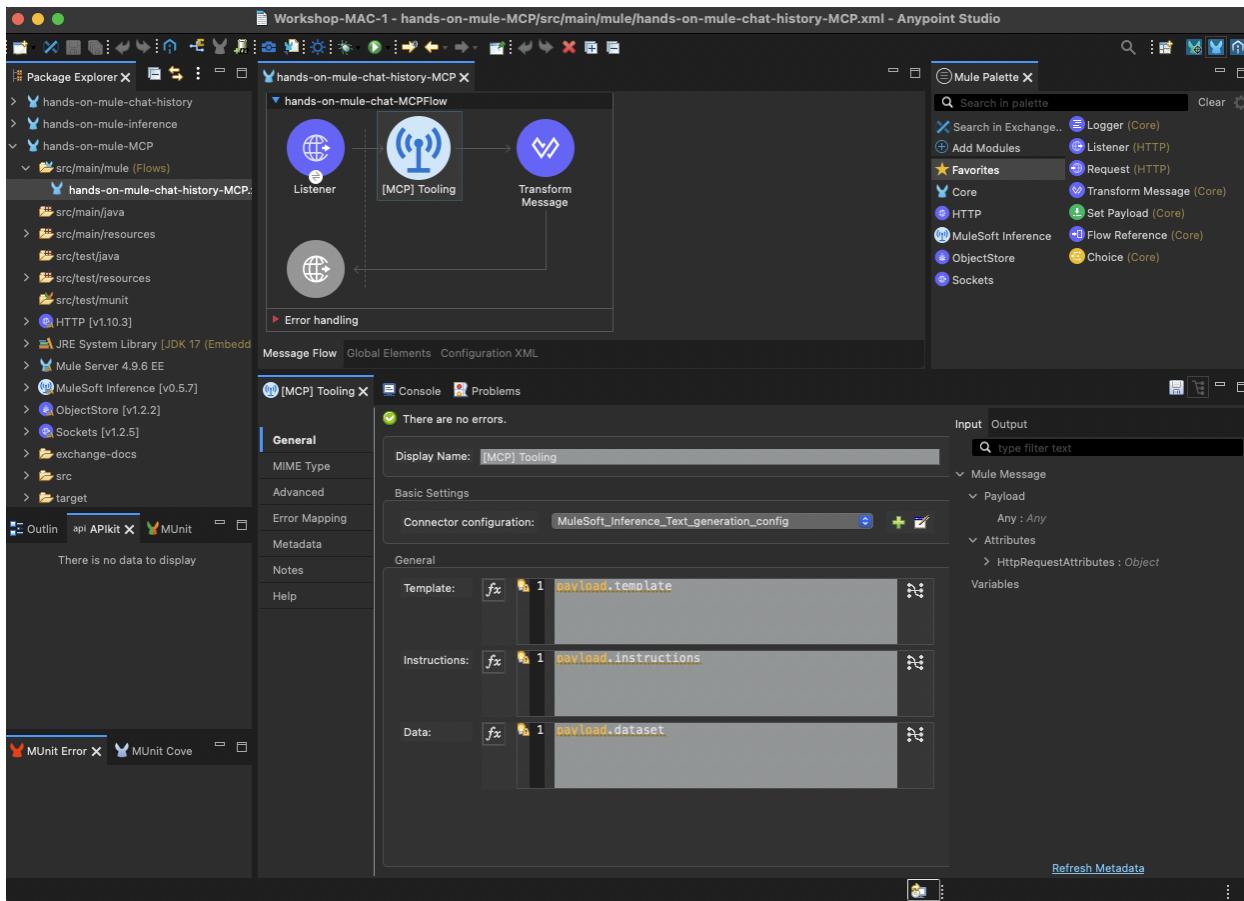
We are going to use the following 2 MCP Servers deployed on Cloudbus using the MuleSoft MCP Connector:

- ERP: <http://mcp-server-demo-b2jb0y.1d6nel.usa-e1.cloudbus.io>
- CRM: <http://mcp-server-demo-crm-b2jb0y.1d6nel.usa-e1.cloudbus.io>

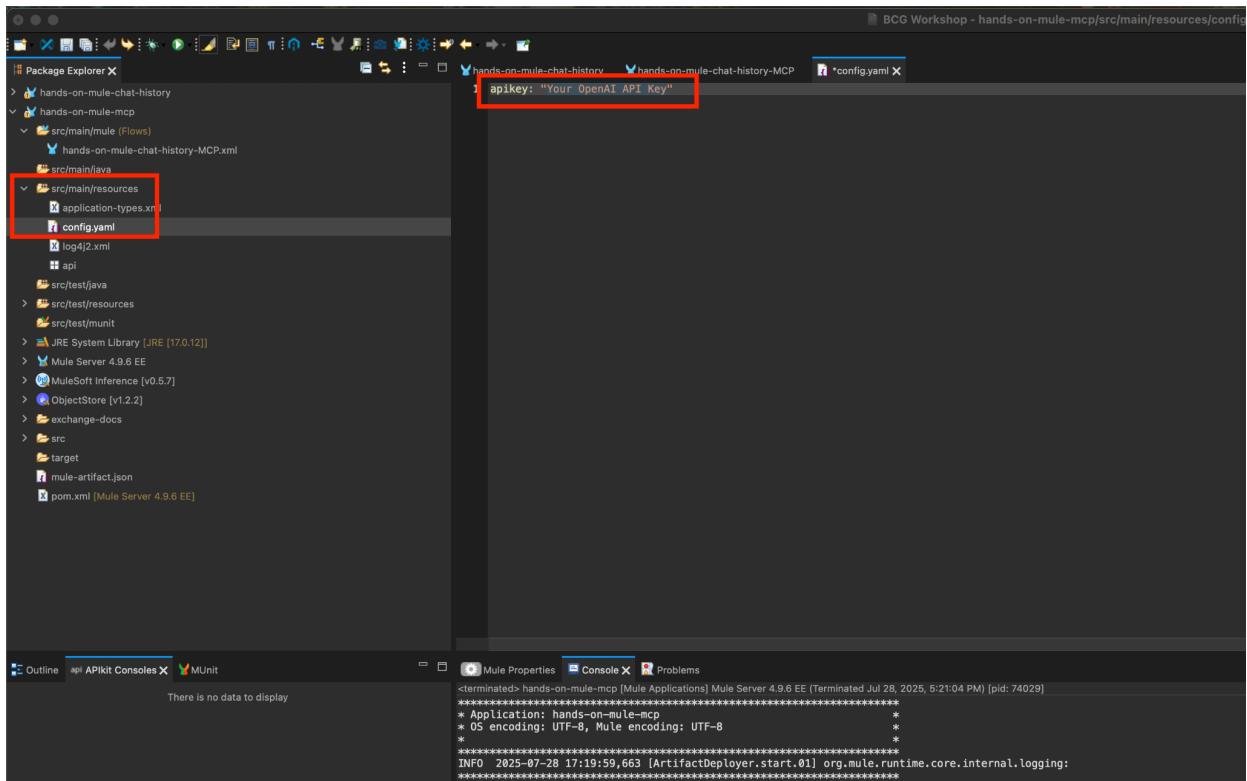
Import and Explore Project

1. Import **hands-on-mule-MCP** into your workspace.

2. Explore the project.



3. Replace the “Your OpenAI API Key” with your OpenAI API Key in the config.json. Put your API key between the quotes.



4. Run hands-on-mule-chat-history-MCP locally.

Test hands-on-mule-MCP

1. Open Postman and select **MCPYouAreHelpful** request and click **Send**.

The screenshot shows the Postman interface with the following details:

- Request URL:** POST <localhost:8081/inference/mcp>
- Body Content:**

```
1 {
2     "template": "You are an helpful assistant",
3     "instructions": "Answer the request with politeness.",
4     "dataset": "What is the capital of Switzerland"
5 }
```
- Response Status:** 200 OK
- Response Headers:** 2.07 s, 513 B
- Response Body (JSON):**

```
1 {
2     "payload": {
3         "response": "The capital of Switzerland is Bern.",
4         "tools": []
5     },
6     "attributes": {
7         "tokenUsage": {
8             "outputCount": 8,
9             "totalCount": 330,
10            "inputCount": 322
11        },
12        "additionalAttributes": {
13            "finish_reason": "stop",
14            "model": "gpt-4o-mini-2024-07-18",
15            "id": "chatcmpl-BvtBBAkqljabeJ5dItJL3gfqs6Fv3"
16        }
17    }
18 }
```

2. In Postman select **MCPCheckInventory** request and click **Send**.

The screenshot shows the Postman interface with the following details:

- URL:** `localhost:8081/inference/mcp`
- Method:** `POST`
- Body Type:** `raw`, `JSON`
- Request Body:**

```

1  {
2   "template": "You are an helpful assistant",
3   "instructions": "Answer the request with politeness.",
4   "dataset": "Check Inventory for MULETEST0"
5 }
6

```
- Response Status:** `200 OK`
- Response Headers:** `Content-Type: application/json; charset=UTF-8`
- Response Body (JSON):**

```

1 {
2   "payload": {
3     "toolsExecutionReport": [
4       {
5         "result": {
6           "productId": "MULETEST0",
7           "availableQuantity": 1650502
8         },
9         "serverUrl": "http://mcp-server-demo-exp-ch1-application.us-e2.cloudhub.io",
10        "serverName": "ERP Sever",
11        "tool": "get_inventory",
12        "timestamp": "2025-07-21T22:23:15.991007Z"
13      }
14    ],
15    "tools": [
16      {
17        "function": {
18          "name": "get_inventory",
19          "arguments": {
20            "materialNo": "MULETEST0"
21          }
22        },
23        "id": "call_6f0WsLradZlIe5kwZmfWncSd",
24        "type": "function"
25      }
26    ]
27  }
28

```

Exercise 4 - Implement A2A

The MuleSoft A2A Connector is a specialized connector within the MuleSoft Anypoint Platform that facilitates secure and discoverable communication between AI agents, known as Agent-to-Agent (A2A) communication.

A2A Connector Can be added through Anypoint Exchange

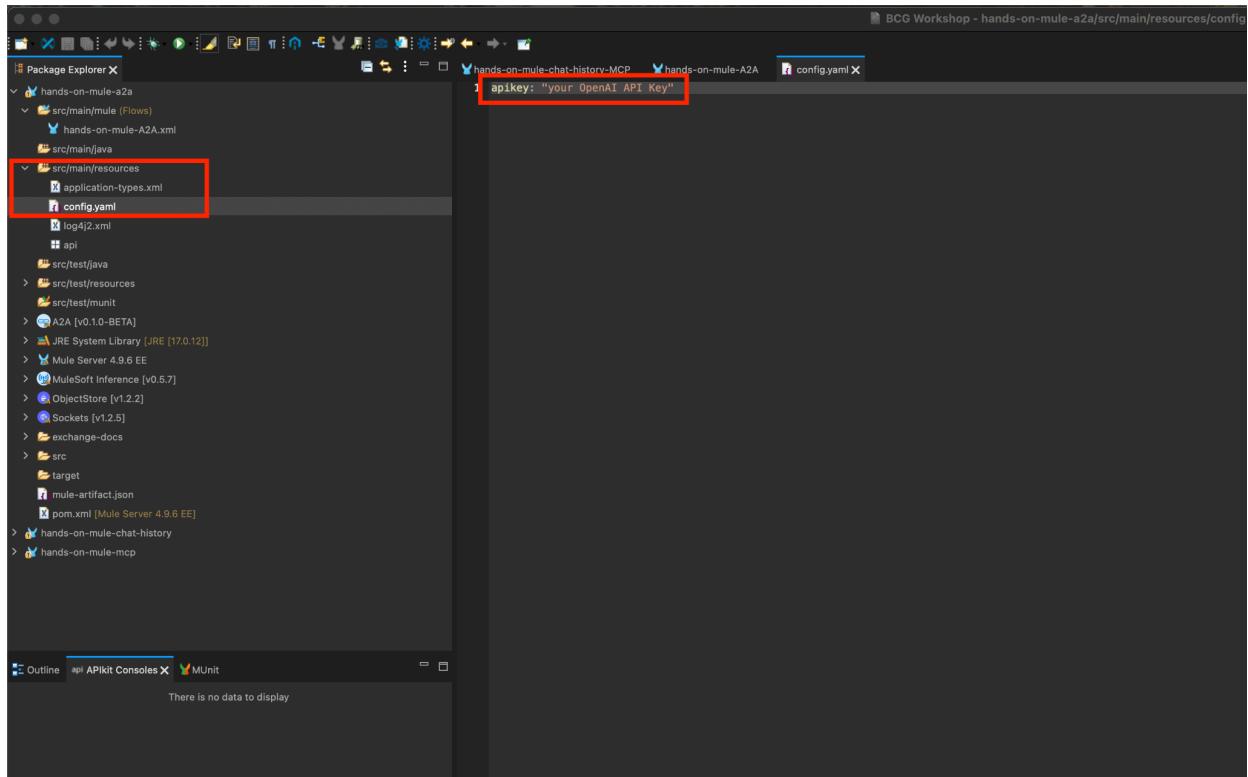
Import and Explore Project

1. Import **hands-on-mule-A2A** into your workspace.

2. Explore the project.



3. 3. Replace the “Your OpenAI API Key” with your OpenAI API Key in the config.json. Put your API key between the quotes.



4. Run hands-on-mule-chat-history-A2A locally.

Test hands-on-mule-A2A

1. Open Postman and select **A2AWhatIsCapital** request and click **Send**.

HTTP Workshop-SME-MAC Capabilities / A2AWhatIsCapital

POST localhost:8081/inference/a2a-client

Params Authorization Headers (9) Body Scripts Settings Cookies

Body Authorization Headers (3) Test Results

200 OK 1.49 s 691 B Save Response

{} JSON ▾ Preview Visualize

```
1 {
2   "prompt": [
3     {
4       "role": "user",
5       "content": "What is the capital of Switzerland"
6     }
7   ]
8 }
```

```
1 {
2   "id": "bc56fc18-e00c-422e-94ae-73504938fe31",
3   "sessionId": "943ed092-60fa-46bd-a440-ca1259d298d0",
4   "status": {
5     "state": "completed",
6     "message": {
7       "role": "agent",
8       "parts": [
9         {
10           "type": "text",
11           "text": "The capital of Switzerland is Bern."
12         }
13       ],
14     },
15     "timestamp": "2025-07-22T01:15:33Z"
16   },
17   "artifacts": [
18     {
19       "name": "Answer",
20       "index": 0,
21       "parts": [
22         {
23           "type": "text",
24           "text": "The capital of Switzerland is Bern."
25         }
26       ],
27     }
28   ]
29 }
```

2. In Postman select **A2AGetAgentCard** request and click **Send**.

The screenshot shows a Postman interface with the following details:

- HTTP Method:** GET
- URL:** localhost:9081/a2a-agent/well-known/agent.json
- Body:** None (selected)
- Headers:** (7) (selected)
- Body Content:** This request does not have a body.
- Response Status:** 200 OK
- Response Headers:** 3 ms, 611 B
- Response Body:**

```
1  {
2    "capabilities": {
3      "streaming": false,
4      "pushNotifications": false,
5      "stateTransitionHistory": false
6    },
7    "defaultOutputModes": [
8      "text/plain"
9    ],
10   "description": "This is a simple chat agent",
11   "defaultInputModes": [
12     "text/plain"
13   ],
14   "version": "1.0.0",
15   "url": "http://localhost:9081/a2a-agent",
16   "skills": [
17     {
18       "description": "This agent has the ability to remember conversation",
19       "name": "ChatMemory",
20       "id": "1"
21     }
22   ],
23   "name": "A2A Agent"
24 }
```

Vectors Hands On Exercises

MAC Vectors provides access to a broad number of external Vector Stores and Databases. It is built to be leveraged by the MAC Projects AI Connector (Amazon Bedrock, MuleSoft AI Chain, MuleSoft Inference & Einstein AI).

As the Vectors Connector is being certified as we deliver this training, it is not available on the Anypoint Exchange Portal. Will will be using the latest open source release 0.6.20

Exercise 5 - Data Load

In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's [Document] Load Single operation is designed to parse a single document and, optionally, split it into chunks of a specified size,

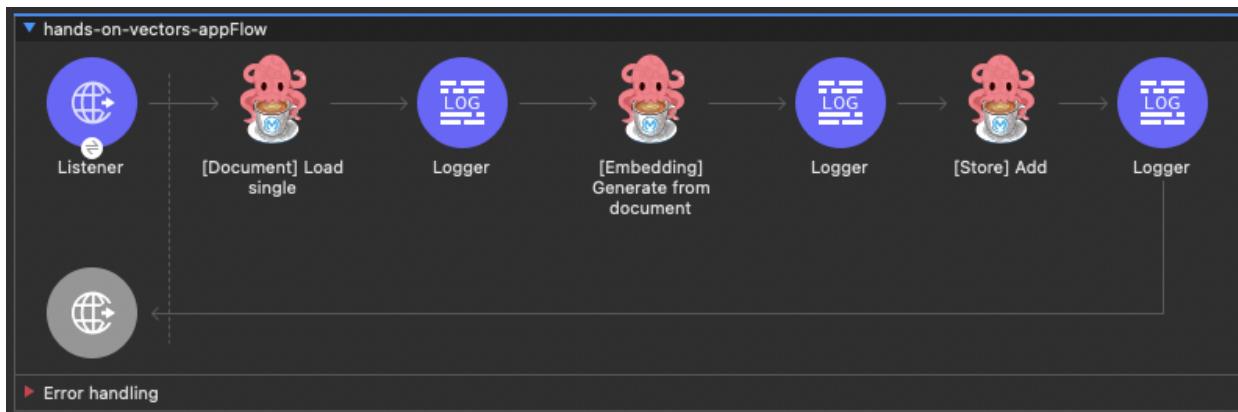
[Embedding] Generate from document operation, allowing you to create searchable embeddings from your documents,

[Store] Add operation adds a document or text into an embedding store.

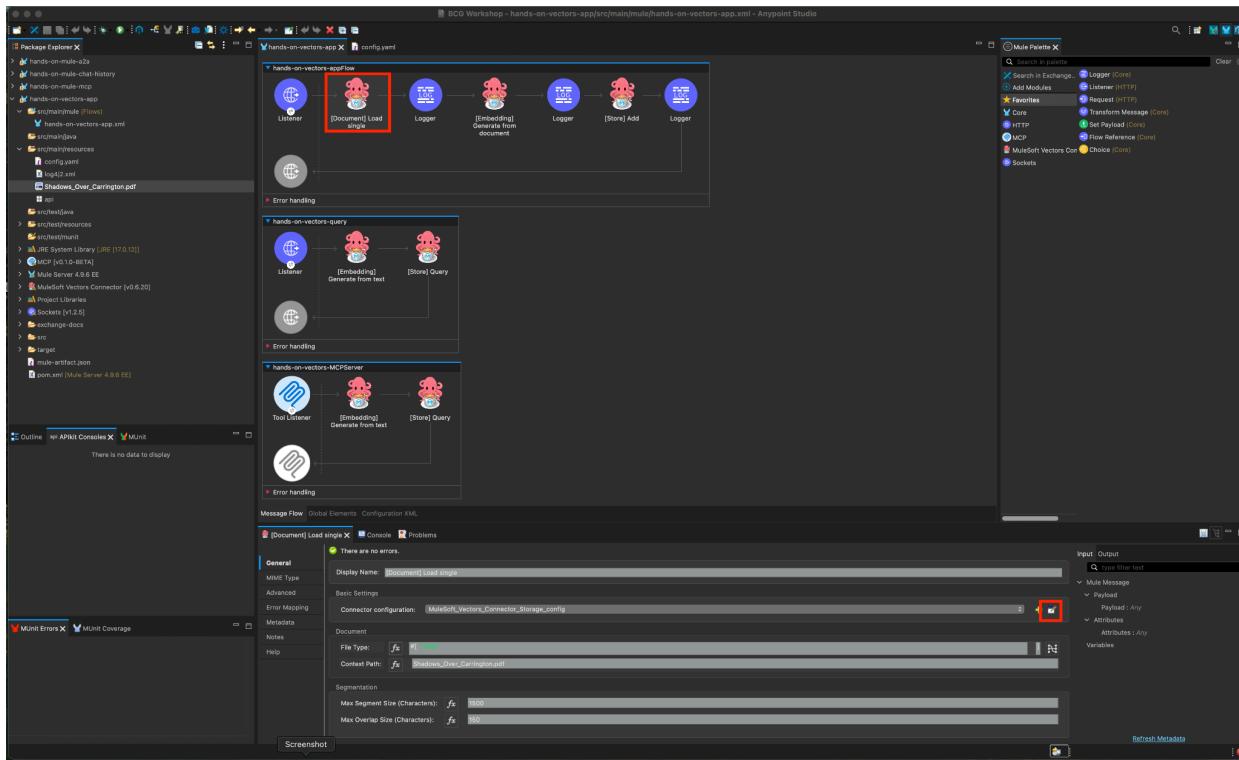
Import and Explore Project

1. Import **hands-on-vectors-app** into your workspace.

2. Explore the project.



3. Click on the first vector operation, the “[Document] Load Single”, then click on the pencil icon to modify the configuration file.

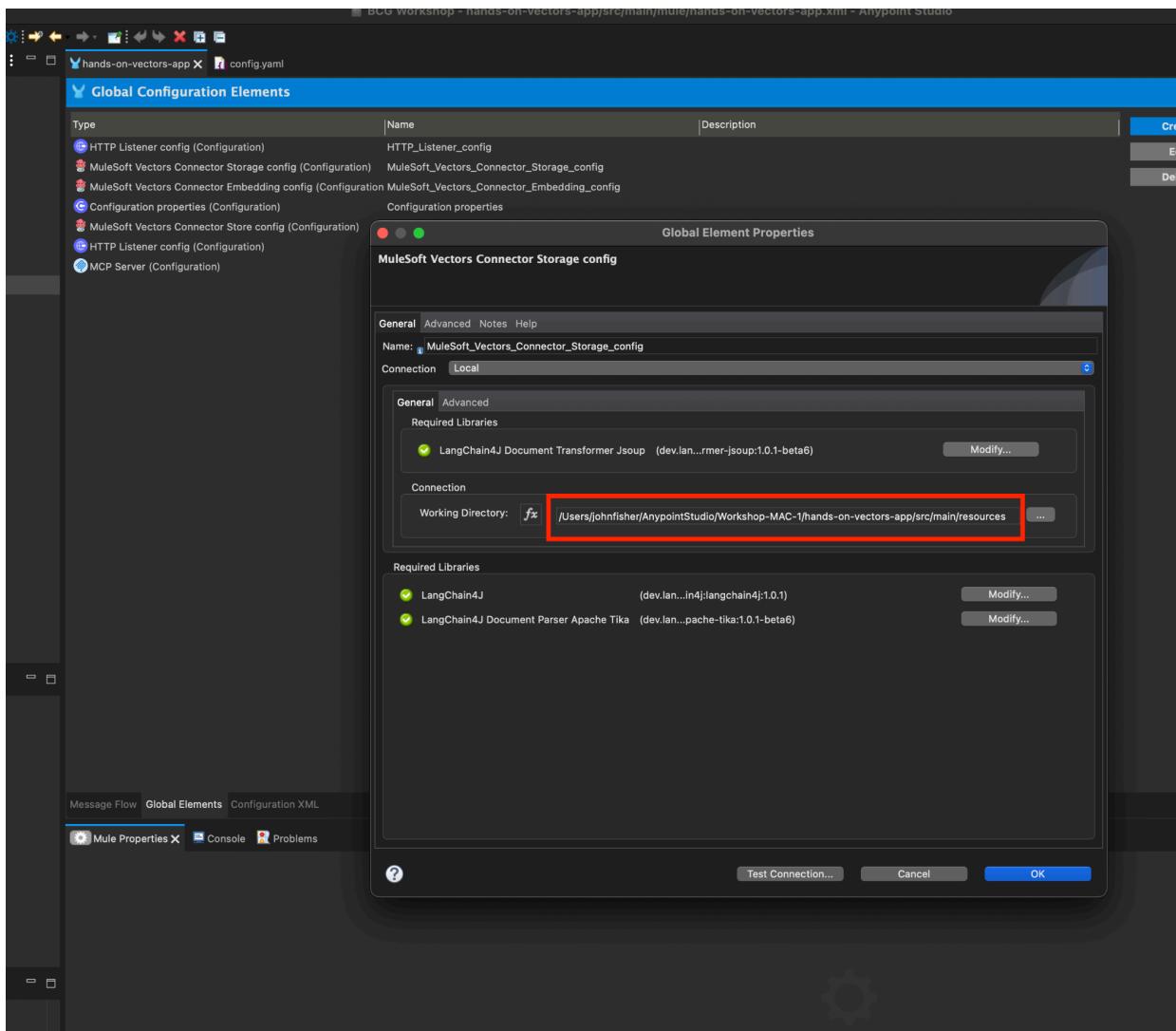


4. Modify the “Working Directory” and replace it with your own path the “Shadows_Over_Carrington.pdf”. The file is located in the Mule App under the ./src/main/resources folder. For example:

Mac: /Users/bpittman/AnypointStudio/BCG Workshop/hands-on-vectors-app/src/main/resources

Windows: C:\Users\bpittman\AnypointStudio\BCG

Workshop\hands-on-vectors-app\src\main\resources



Test hands-on-mule-vectors-app

1. Open Postman and select **VectorAdd** request and click **Send**.

The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:8081/vectors/add`. Below the URL, there are tabs for `Params`, `Authorization`, `Headers (8)`, `Body`, `Scripts`, and `Settings`. The `Headers (8)` tab is currently selected. In the `Body` section, the `JSON` tab is selected, showing the following JSON response:

```
1  {
2   "sourceId": "aecc9560-594c-49c6-a8e3-dfa5e9154285",
3   "embeddingIds": [
4     "2135467a-8339-49d4-ac3b-c7114eaf2ef3",
5     "12540b33-bfe4-4cdd-be62-d1817c0e5eaa",
6     "4dd15f11-e9d1-4b90-aa82-f909c5f1dc28",
7     "c08f2e38-a551-4d29-9f4b-9fdfd3c71134"
8   ],
9   "status": "updated"
10 }
```

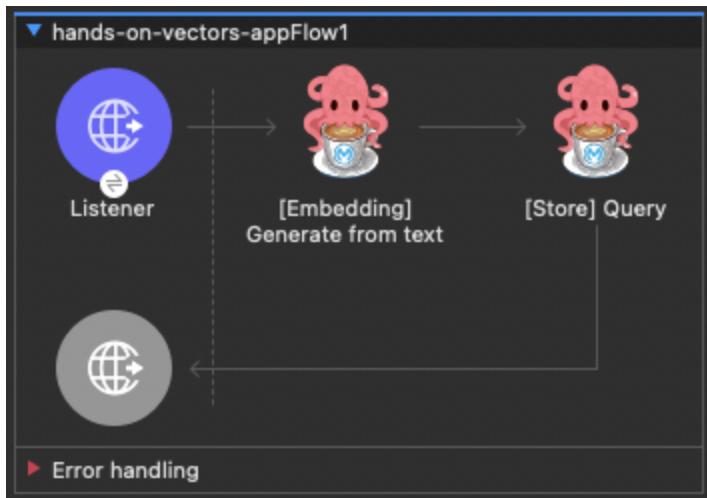
Exercise 6 - Query Vector Store

In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's [Embedding] Generate from text operation optionally splits the text into chunks of the provided size and creates numeric vectors for each text chunk.

[Store] Query operation retrieves information from the embedding store based on an embedding (previously generated from a text prompt) and optionally a filter on metadata.

1. Import **hands-on-vectors-app** into your workspace (if you havent done so already).
2. Explore the project.



Test hands-on-mule-vectors-app

1. Open Postman and select **VectorQuery** request and click **Send**.

```

1  {
2    "prompt": "Shadows Over Carrington"
3  }

```

```

1  {
2    "minScore": 0.5,
3    "question": "Shadows Over Carrington",
4    "sources": [
5      {
6        "score": 0.8284866912074875,
7        "metadata": {
8          "absolute_directory_path": "/Users/johnfisher/AnypointStudio/
9            Workshop-MAC-1/hands-on-vectors-app/src/main/resources",
10         "ingestion_timestamp": "1753155638982",
11         "file_name": "Shadows_Over_Carrington.pdf",
12         "file_type": "any",
13         "index": "0",
14         "source_id": "aecc9560-594c-49c6-a8e3-dfa5e9154285",
15         "ingestion_datetime": "2025-07-22T03:40:38.981511Z"
16       },
17       "embeddingId": "2135467a-8339-49d4-ac3b-c7114eaf2ef3",
18       "text": "Shadows Over Carrington \n Carrington was the kind of town where
nothing ever happened-until it did. On a chilled autumn \n morning, the
body of Victor Hensley, a retired judge, was found in his study, slumped
over a \n desk of legal tomes. A single bullet to the chest and an

```

Exercise 7 - Vector Store as a MCP Server

In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's [Embedding] Generate from text operation optionally splits the text into chunks of the provided size and creates numeric vectors for each text chunk and

[Store] Query operation retrieves information from the embedding store based on an embedding (previously generated from a text prompt) and optionally a filter on metadata.

The MCP protocol:

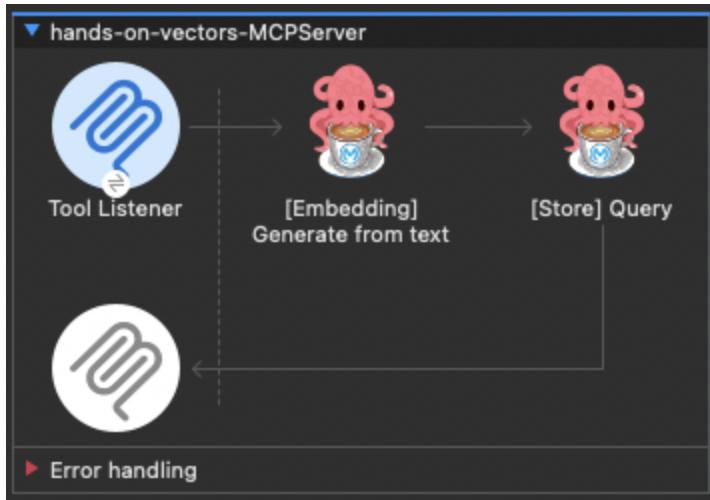
- Enhances AI precision by supplying AI with relevant information, leading to more accurate responses.
- Empowers AI to manage intricate tasks that require access to external data or services.
- Promotes interoperability between AI models and external systems, simplifying integration into workflows.

MCP Connector:

- Enables you to expose your Mule apps, connectors, and custom APIs using the MCP protocol to further enable agentic experiences from agents in Claude, Cursor, Windsurf, and your own agents.
- Enables you to write Mule apps that can act as both an MCP server and a client. The resulting MCP server or client runs inside of Mule, which means you don't need to configure your own hosting environment.
 - When acting as an MCP server, Mule allows AI clients to use existing API-led investments, legacy systems, and SaaS applications that don't natively support MCP.
 - When acting as an MCP client, Mule helps you create connections and orchestrations where AI agents are added as just another system in the workflow.
- Allows you to incorporate AI agents into your existing Enterprise ecosystem, using Mule as the orchestration layer.
- Supports Streamable HTTP for both server and client (recommended) and SSE (for both server and client) transport methods.
- Supports custom response headers on every exposed endpoint, including /sse, /messages, and /mcp. Use custom response headers for security, operational, integration, and governance benefits.

1. Import **hands-on-vectors-app** into your workspace (if you haven't done so already).

2. Explore the project.



Test hands-on-mule-vectors-app

1. Open Postman and select **VectorQuery** request and click **Send**. We want to check that the Vector store is working.

The screenshot shows a Postman interface with the following details:

- HTTP Method:** POST
- URL:** http://localhost:8081/vectors/query
- Body Content:**

```

1  {
2    "prompt": "Shadows Over Carrington"
3  }

```

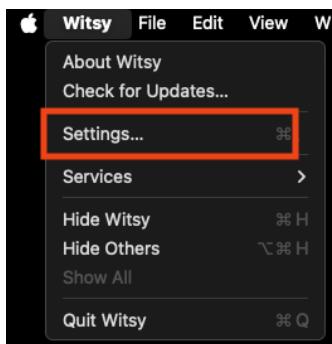
- Response Status:** 200 OK
- Response Time:** 1.58 s
- Response Size:** 17.32 KB
- Response Body (JSON):**

```

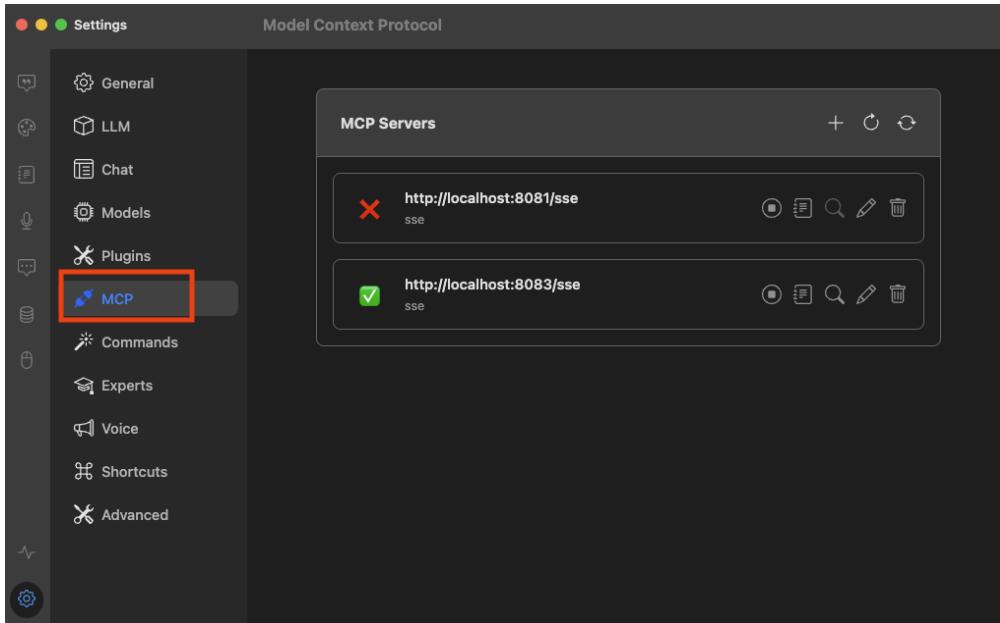
1  {
2    "minScore": 0.5,
3    "question": "Shadows Over Carrington",
4    "sources": [
5      {
6        "score": 0.8284746417150548,
7        "metadata": {
8          "absolute_directory_path": "/Users/johnfisher/AnypointStudio/
Workshop-MAC-1/hands-on-vectors-app/src/main/resources",
9          "ingestion_timestamp": "1753155638982",
10         "file_name": "Shadows_Over_Carrington.pdf",
11         "file_type": "any",
12         "index": "0",
13         "source_id": "aecc9560-594c-49c6-a8e3-dfa5e9154285",
14         "ingestion_datetime": "2025-07-22T03:40:38.981511Z"
15       },
16       "embeddingId": "2135467a-8339-49d4-ac3b-c7114eaf2ef3",
17       "text": "Shadows Over Carrington \n Carrington was the kind of town where
nothing ever happened—until it did. On a chilled autumn \n morning, the
body of Victor Hensley, a retired judge, was found in his study, slumped
over a \n desk of legal tomes. A single bullet to the chest and an

```

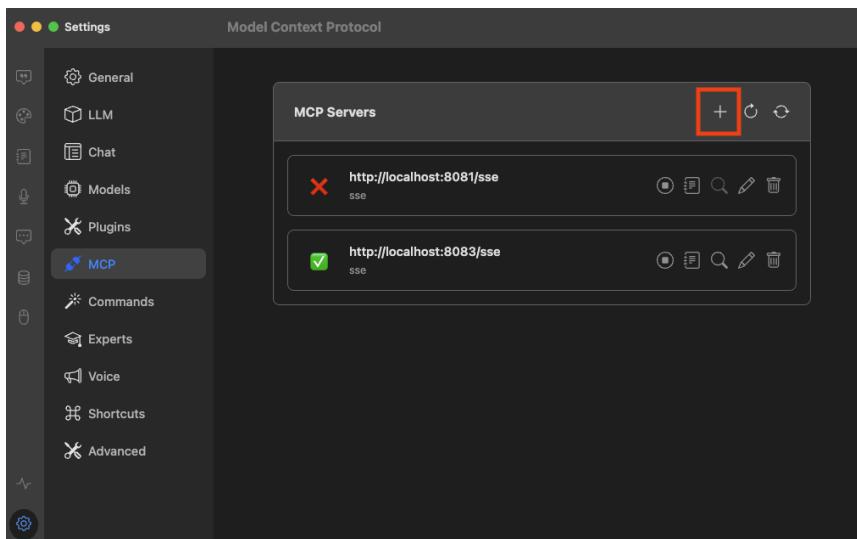
2. Open Witsy and from the Witsy Menu select **settings**.



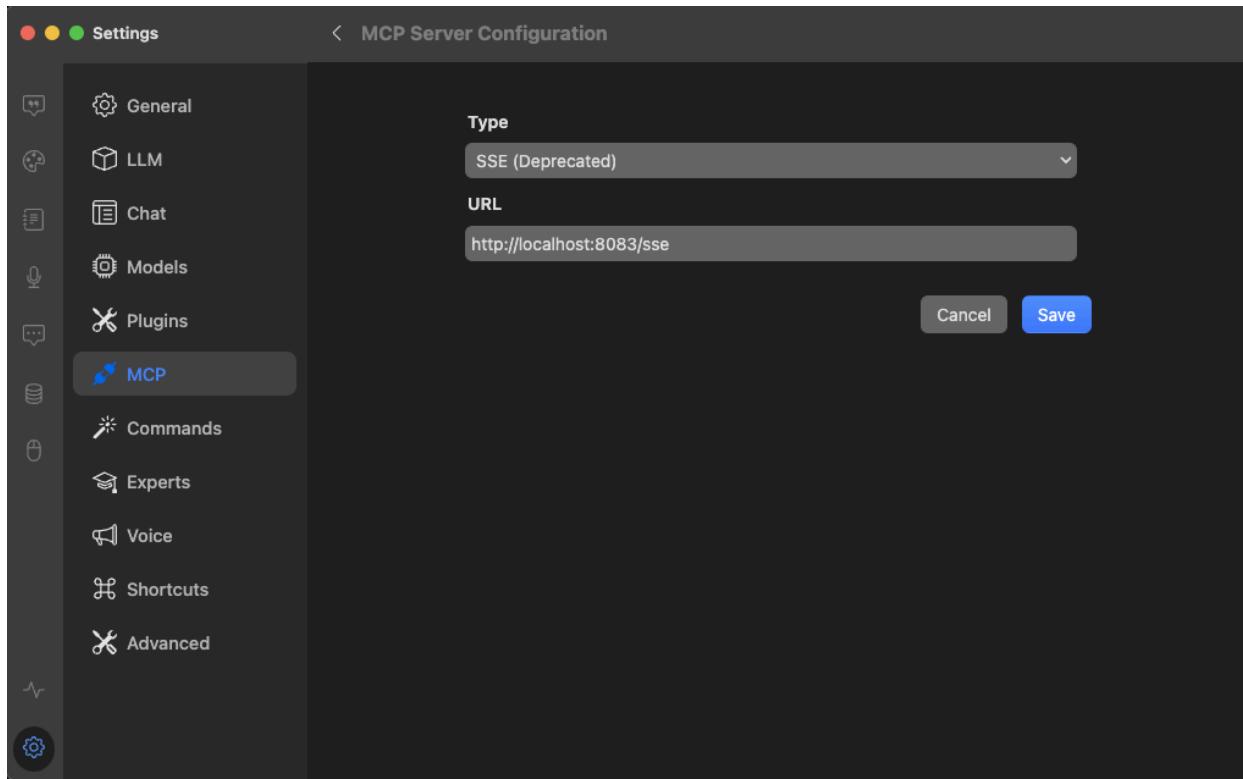
3. Click on MCP to open the MCP Servers configuration.



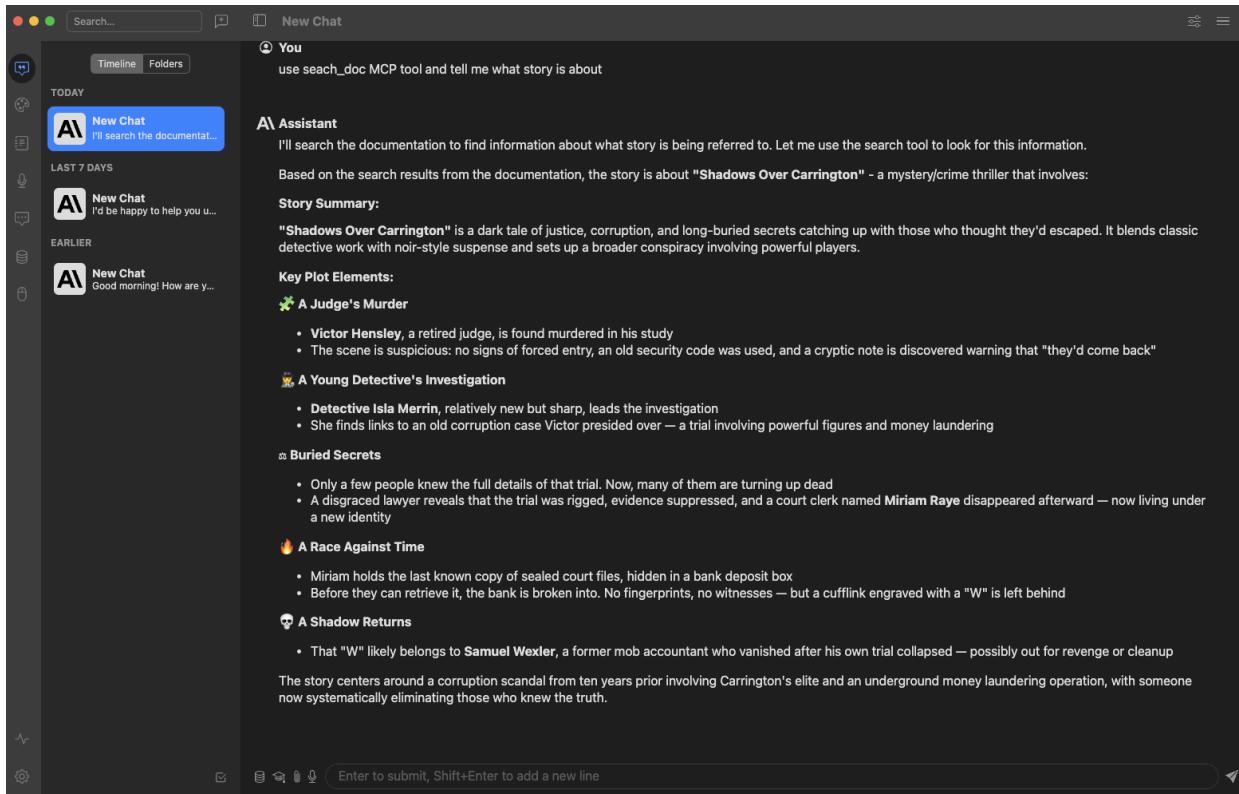
4. Click on the “+” icon to add a new MCP Server.



5. Select **Type** SSE, enter **URL** <http://localhost:8083/sse> as shown below and click **Save**.



6. Test your MCP server by chatting with Witsy. Enter “use search_doc MCP tool and tell me what the story is about”.



WebCrawler Hands On Exercises

MuleSoft AI Connector WebCrawler offers the ability to crawl websites and retrieve valuable content, empowering you to seamlessly organize data into vectors for structured knowledge extraction. By integrating this connector, you can automate web content retrieval and create exciting AI applications using up-to-date data, thus enhancing the AI-experiences of your users.

Features:

- Comprehensive Web Crawling: Crawl entire websites or target specific content, enabling large-scale data extraction with ease.
- Customizable Crawls: Tailor the crawling process with options to control depth and content filters to meet specific business needs.
- Content Extraction: Retrieve various types of content, including images, textual data, and metadata, to feed into downstream systems.
- Page Insights: Gain valuable insights into website structures, link distributions, and content wordcount, and use these insights to customize your crawl.
- Seamless Integration: Designed to fit effortlessly within MuleSoft workflows, the MuleSoft WebCrawler connector ensures your web data is transformed into actionable insights for optimized decision-making.

Exercise 8 - Get Static Page

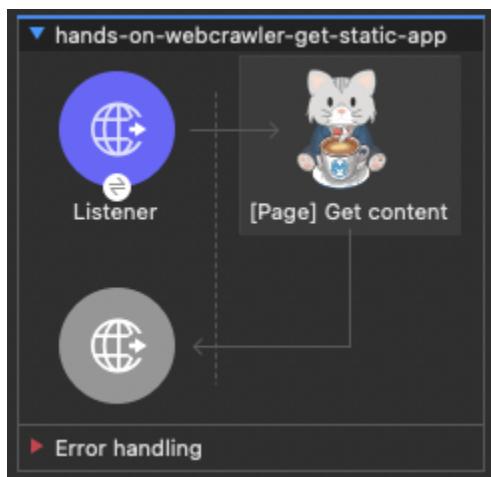
In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's:

[Page] Get content: Allows you to retrieve the contents of a specified webpage.

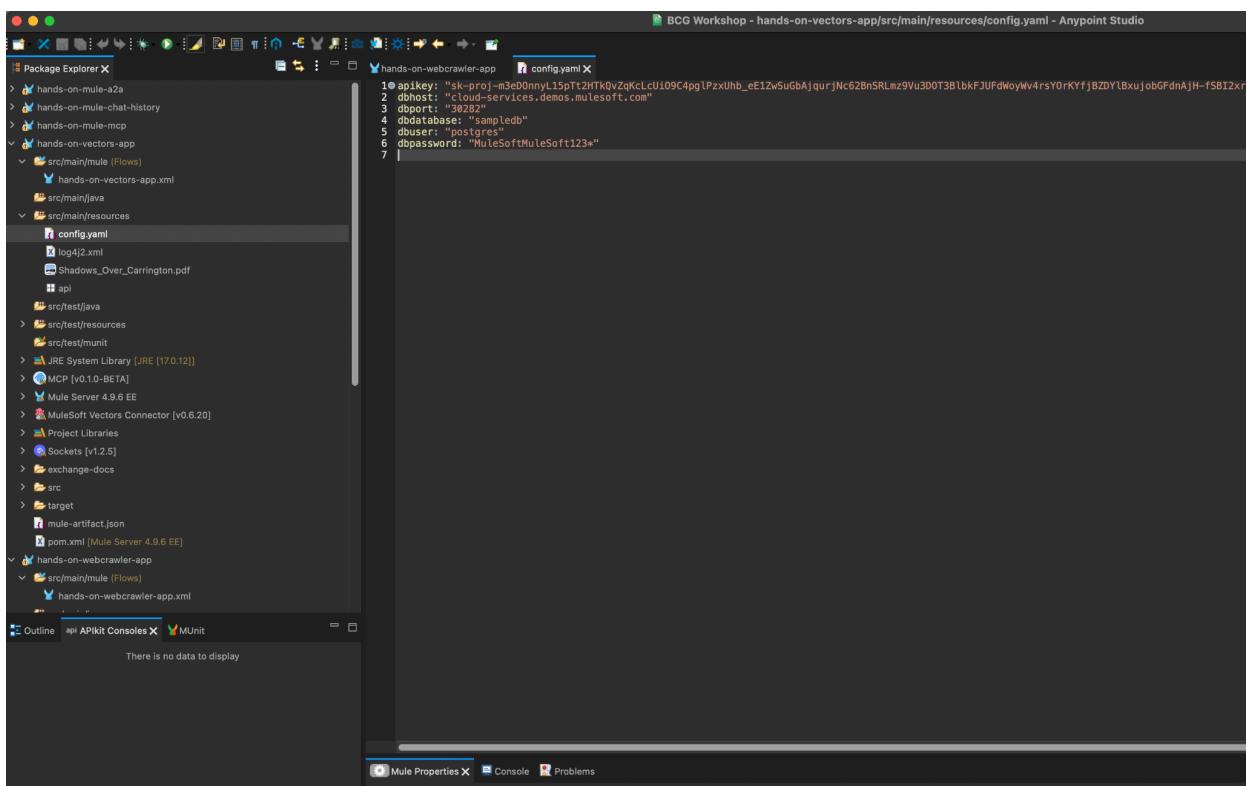
Import and Explore Project

1. Import **hands-on-webcrawler-app** into your workspace.
2. Explore the project.



3. Open the config.yaml and update the values with values provided by the teacher.

Save the file.

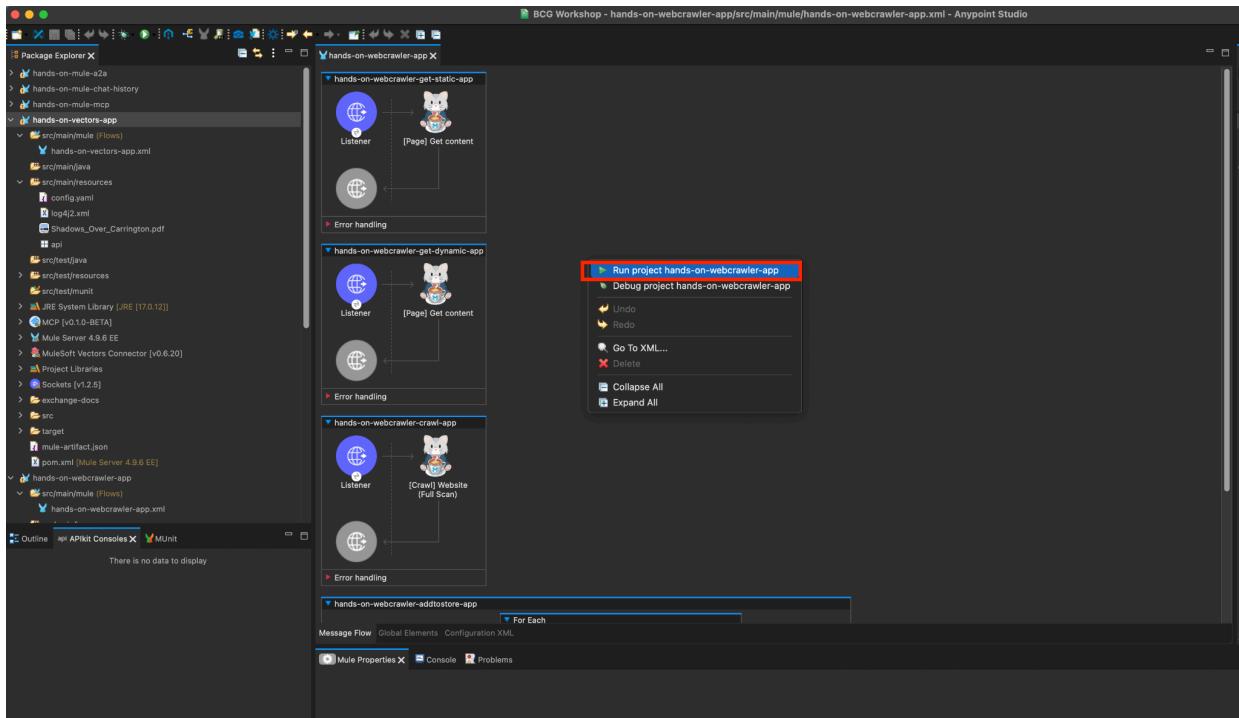


The screenshot shows the Anypoint Studio interface with the 'hands-on-webcrawler-app' project selected in the Package Explorer. The 'config.yaml' file is open in the central editor area. The code content is as follows:

```
1 apiKey: "sk-proj-m3g0DnnvL15pTz2HTk0vZqKcLcU109C4pglPzxUhb_eE1ZwSuGbAjqurjNc62BnSRLmz9Vu3D0T38lbkFJUFdWoyW4rsYOrKYfjBZDY1BxujobGFdnA|H-fSBt2xr"
2 dbhost: "cloud-services.demos.mulesoft.com"
3 dbport: "3028"
4 dbdatabase: "sampledb"
5 dbuser: "postgres"
6 dbpassword: "MuleSoftMuleSoft123"
7
```

The bottom status bar indicates 'There is no data to display'.

4. Right click on the empty space in the Canvas, and click “Run project hands-on-webcrawler-app”



5. You should see it start successfully in the “Console” Window

Mule Properties Console X Problems

hands-on-webcrawler-app [Mule Applications] [pid: 53656]

```
+-----+
+ Mule is up and kicking (every 5000ms) +
+-----+
INFO 2025-07-29 09:59:44,376 [WrapperListener_start_runner] org.eclipse.jetty.server.AbstractConnector: Started ServerConnector@0.0.0.0:8080\[http://0.0.0.0:8080\]
INFO 2025-07-29 09:59:44,377 [WrapperListener_start_runner] org.mule.runtime.core.internal.logging:
*****-* - + DOMAIN + - - * - + STATUS + - - *-
*****-* default * DEPLOYED *-
*****-* - - + APPLICATION + - - * - + DOMAIN + - - * - + STATUS + - - *-
*****-* hands-on-webcrawler-app * default * DEPLOYED *-
*****-* - - + SelectorRunner + - - * - + SelectorRunner + - - *-
INFO 2025-07-29 09:59:44,385 [[MuleRuntime].uber.05: [hands-on-webcrawler-app].uber@org.mule.runtime.module.extension.internal]
INFO 2025-07-29 09:59:44,385 [[MuleRuntime].uber.02: [hands-on-webcrawler-app].uber@org.mule.runtime.module.extension.internal]
INFO 2025-07-29 09:59:44,385 [[MuleRuntime].uber.04: [hands-on-webcrawler-app].uber@org.mule.runtime.module.extension.internal]
INFO 2025-07-29 09:59:44,385 [[MuleRuntime].uber.06: [hands-on-webcrawler-app].uber@org.mule.runtime.module.extension.internal]
INFO 2025-07-29 09:59:45,080 [http.listener.14 SelectorRunner] [processor: ; event: ] com.mulesoft.connectors.mcp.internal.s
```

Test hands-on-mule-webcrawler-app

1. Open Postman and select **WebCrawlerGetStatic** request and click **Send**.

The screenshot shows the Postman interface with the following details:

- Request URL:** `http://localhost:8081/webcrawler/get-static`
- Method:** POST
- Body (JSON):**

```

1  {
2    "url": "https://mac-project.ai/",
3    "outputFormat": "MARKDOWN"
4  }
      
```
- Response Status:** 200 OK
- Response Body (JSON):**

```

1  {
2    "title": "The MuleSoft AI Chain (MAC) Project",
3    "url": "https://mac-project.ai/",
4    "content": "The MuleSoft AI Chain (MAC) Project\nhttps://mac-project.ai/\n[](/docs)[About](/about)\nGitHub GitHub (opens in a new tab)\nLight\n# Build powerful AI Agents\nwith MuleSoft AI Chain\nSeamlessly integrate cutting-edge AI capabilities into your MuleSoft ecosystem,\nenabling smarter automation and enhanced decision-making.\n[Go to Docs](/docs)\n![Background](_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcard-1.7bf2b1df.png&w=3840&q=75)\n[Background (Dark)](_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcard-1.dark.b34c0349.png&w=3840&q=75)\nFull-powered AI Agents\nwith MuleSoft\nDevelopment\nMuleSoft AI Chain enables MuleSoft developers to leverage powerful AI capabilities with minimal coding. Easily configure and manage AI agents directly within Anypoint Studio, streamlining the development process.\n![Background](_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcard-operations.114cee09.png&w=1920&q=75)\n[Background (Dark)](_next/image?url=%2F_next%2Fstatic%2Fmedia%2Fcard-operations.dark.d99fd531.png&w=1920&q=75)\n[docs/mulechain-ai/getting-started]\nMuleSoft AI Chain (MAC)\nProject\nOpenSource\nMAC Project is an open-source project empowering developers to integrate advanced AI capabilities into the MuleSoft ecosystem.\n[Join our community](https://www.linkedin.com/groups/13047000/) to collaborate, innovate, and drive the future of technology.\n[docs/contribute]\n"
      
```

Exercise 9 - Get Dynamic Page

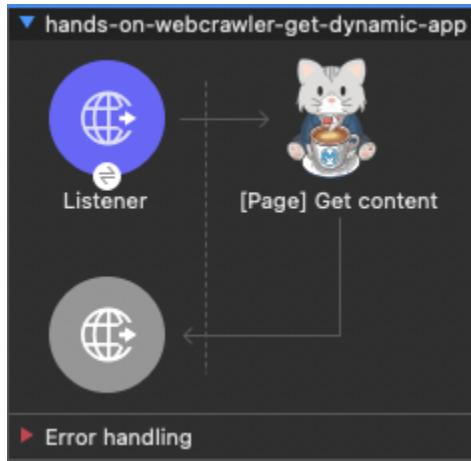
In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's:

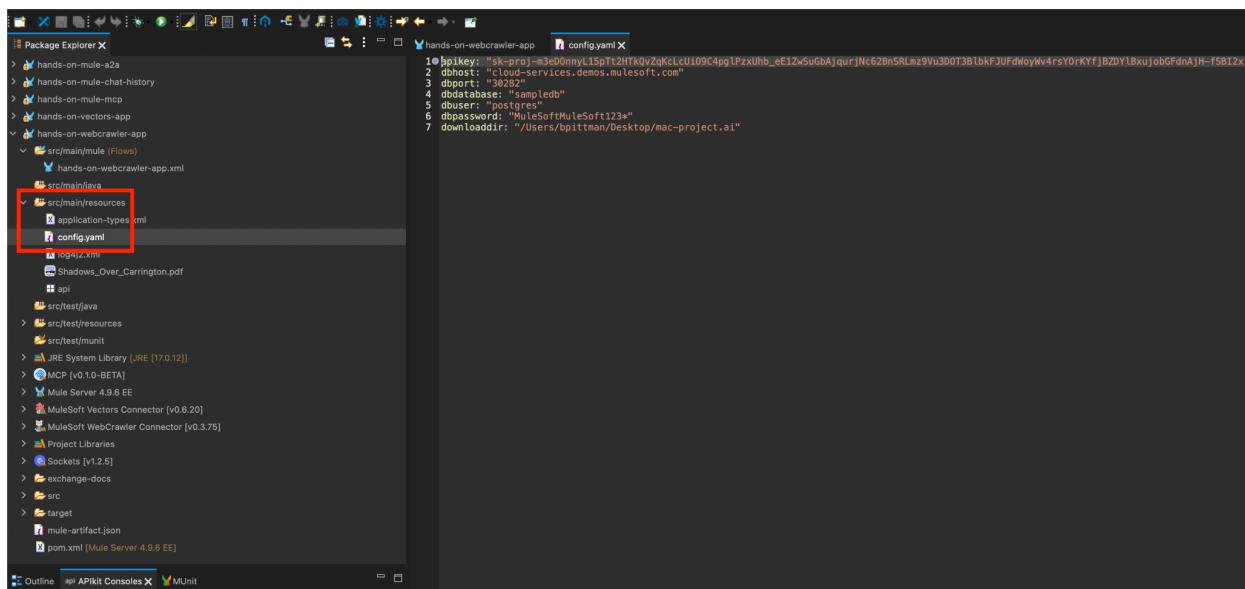
[Page] Get content (this time we are going to crawl a dynamic web page)

Import and Explore Project

1. Import **hands-on-webcrawler-app** into your workspace (if you haven't done so already).
2. Explore the project.

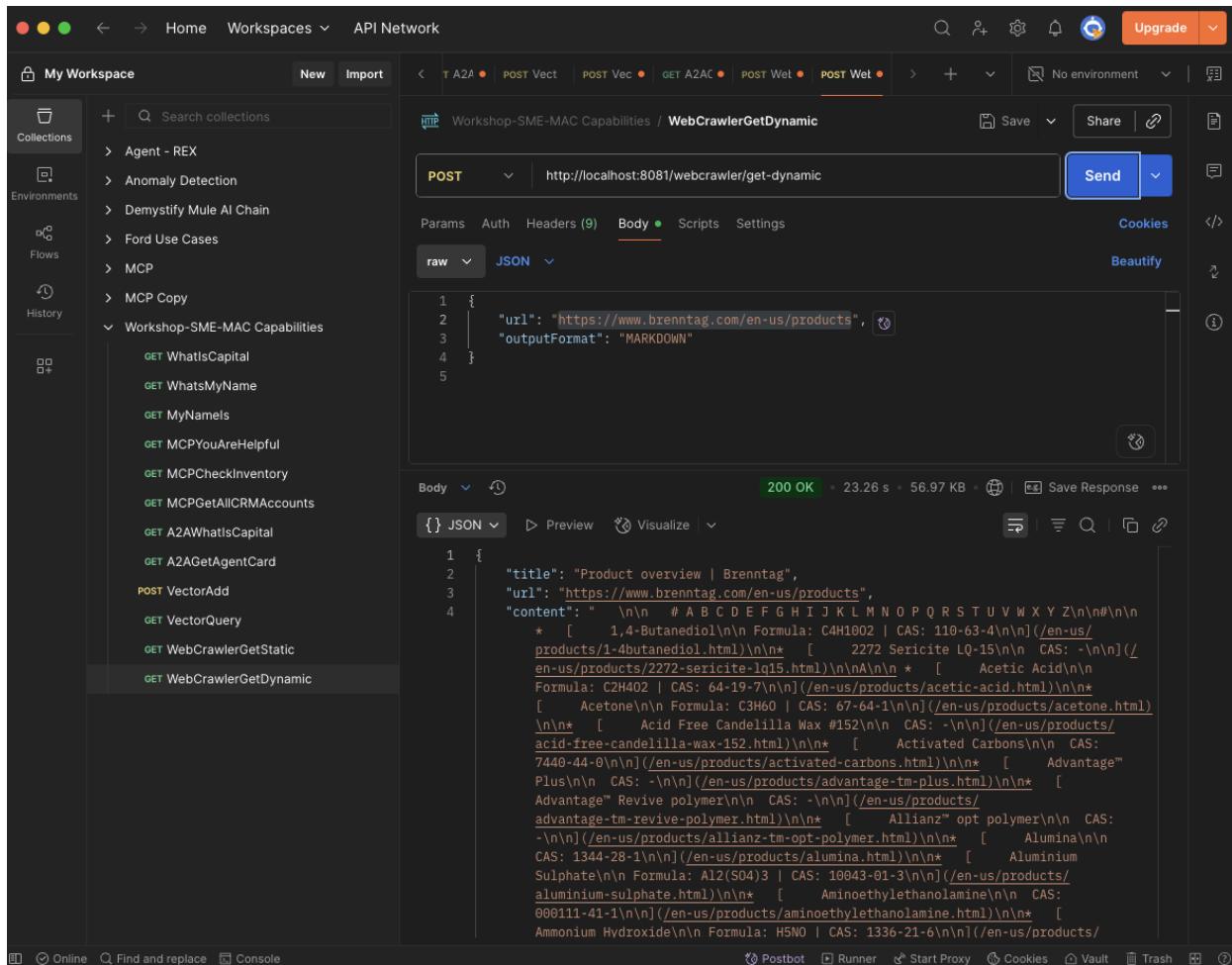


3. Open the config.yaml and update the values with values provided by the teacher.
Save the file.



Test hands-on-mule-webcrawler-app

1. Open Postman and select **WebCrawlerGetDynamic** request and click **Send**.



Exercise 10 - Crawl a web page

In this exercise we are going to explore:

The MuleSoft AI Chain (MAC) Connector's:

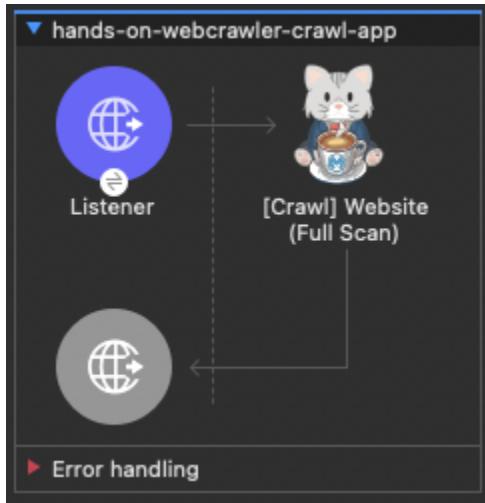
[Crawl] Website (Full Scan): operation allows you to easily crawl for website content, at a specified depth. This operation allows you to additionally:

- set a crawl delay so that you are not overloading the webserver with requests
- download images from the crawled web pages during the crawl

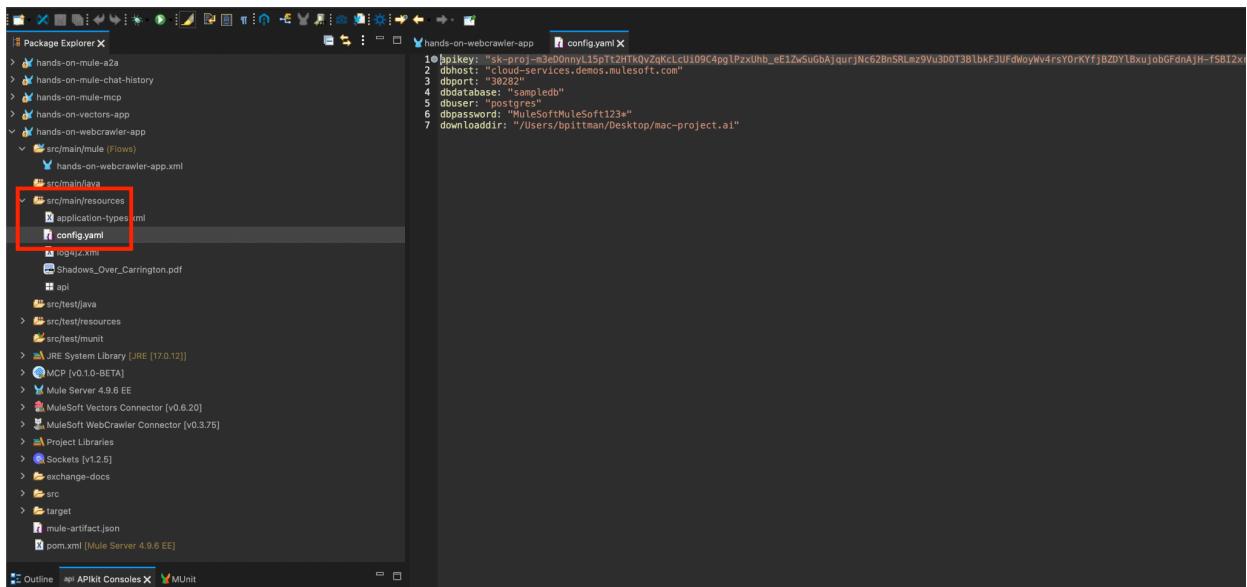
Import and Explore Project

1. Import **hands-on-webcrawler-app** into your workspace (if you haven't done so already).

2. Explore the project.

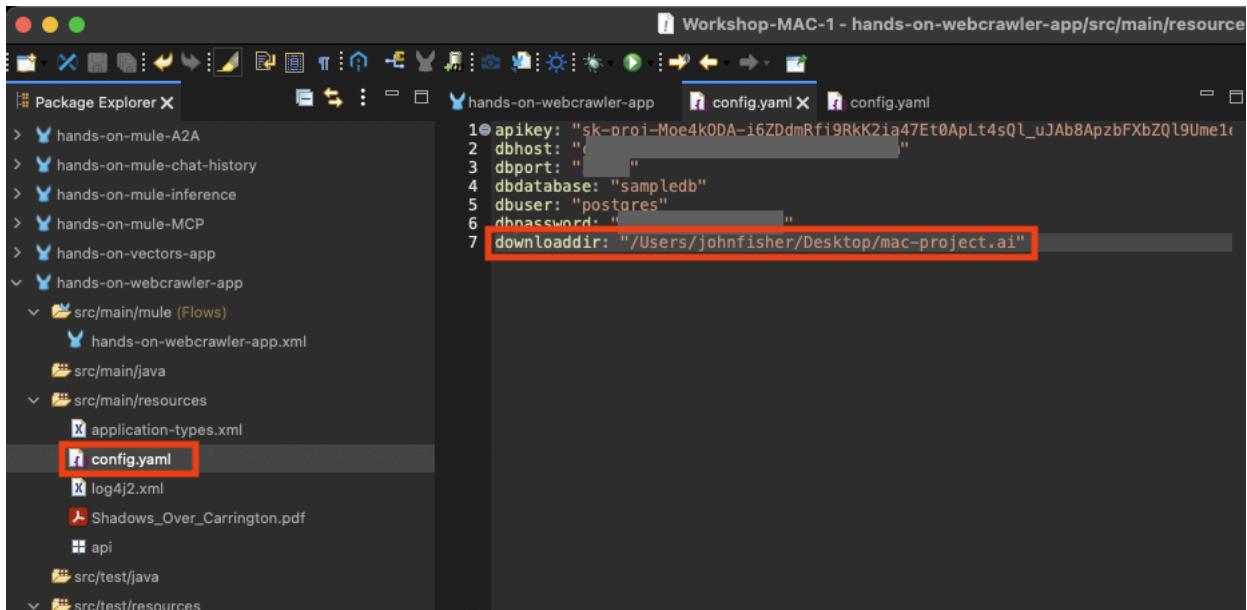


3. Open the config.yaml and update the values with values provided by the teacher.
Save the file.



4. Create a folder on your desktop and name it mac-project.ai. Hands-on-webcrawler-app will download the web site content to this directory.

5. Update **config.yaml downloadadir** to reflect your computer path to folder mac-project.ai. See screenshot below.



```
1 apikey: "sk-proj-Moe4kODA-i6ZdmRfi9RkK2ia47Et0ApLt4sQl_uJAb8ApzbFXbZQl9Ume1c"
2 dbhost: "127.0.0.1"
3 dbport: "5432"
4 dbdatabase: "sampledb"
5 dbuser: "postgres"
6 dbpassword: "password"
7 downloadadir: "/Users/johnfisher/Desktop/mac-project.ai"
```

Test hands-on-mule-webcrawler-app

1. Open Postman and select **WebCrawlerCrawl** request and click **Send**.

HTTP Demystify Mule AI Chain / WebCrawlerCrawl

Save Share

POST http://localhost:8081/webcrawler/crawl Send

Params Auth Headers (9) Body Scripts Settings Cookies

raw JSON Beautify

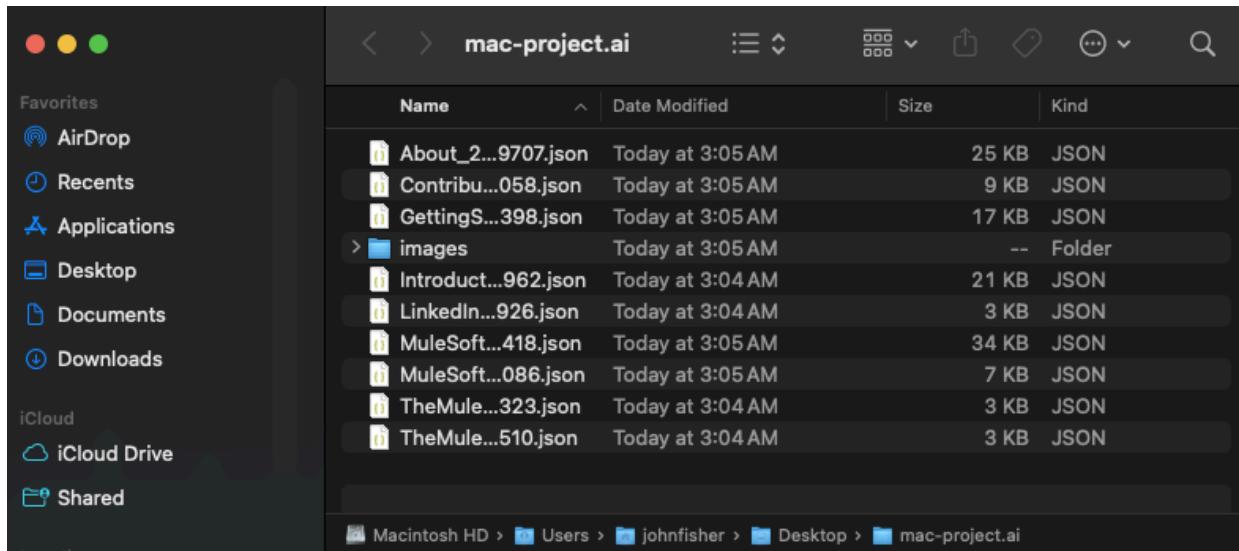
```
1  {}
```

Body 200 OK 54.41 s 1.09 KB Save Response

{ } JSON Preview Visualize

```
1 {
2   "url": "https://mac-project.ai",
3   "filename": "TheMuleSoftAIChain(MAC)Project_20250722020923076.json",
4   "children": [
5     {
6       "url": "https://www.linkedin.com/groups/13047000/",
7       "filename": "LinkedInLogin_Signin_LinkedIn_20250722020923731.json"
8     },
9     {
10       "url": "https://mac-project.ai/docs",
11       "filename": "Introduction_20250722020936823.json"
12     },
13     {
14       "url": "https://mac-project.ai/",
15       "filename": "TheMuleSoftAIChain(MAC)Project_20250722020939065.json"
16     },
17     {
18       "url": "https://mac-project.ai/docs/mulechain-ai/getting-started",
19       "filename": "GettingStarted_20250722020948949.json"
20     },
21   ]
}
```

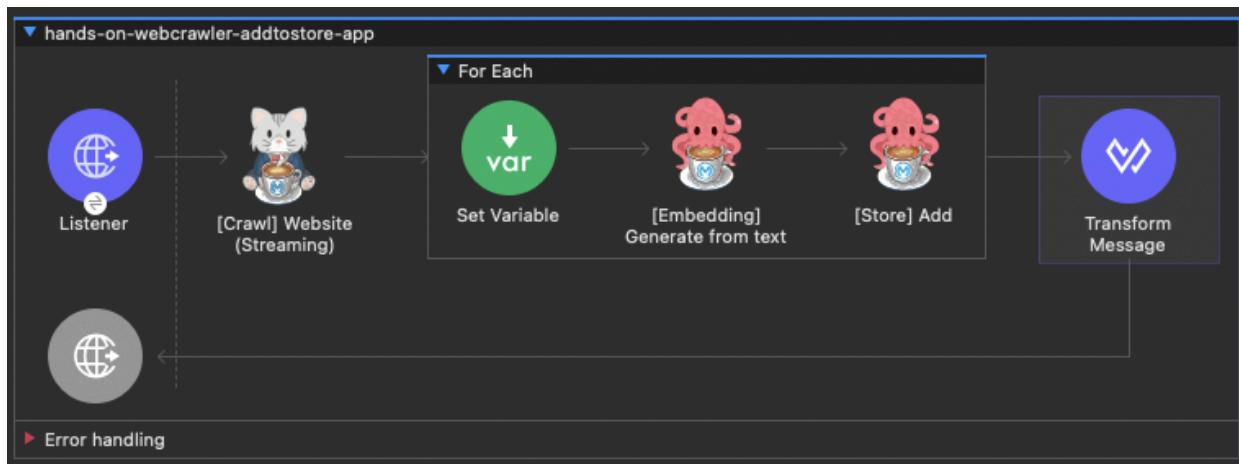
2. Open folder **mac-project.ai** and review its contents.



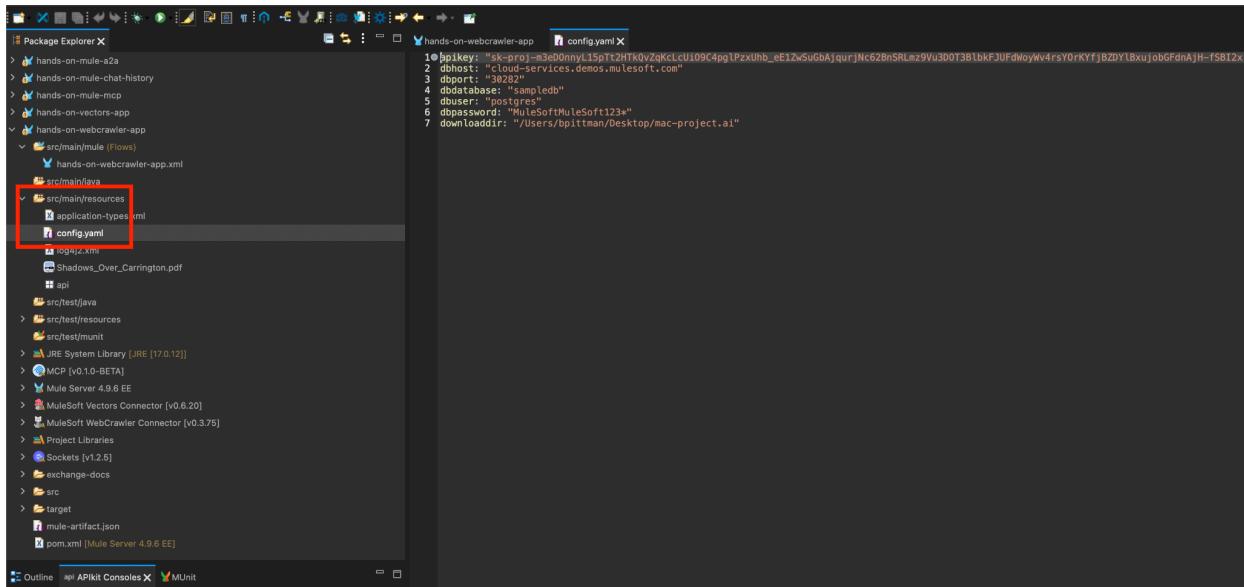
Exercise 11 - Crawl a web page and Add to Store

Import and Explore Project

1. Import **hands-on-webcrawler-app** into your workspace (if you haven't done so already).
2. Explore the project.



3. Open the config.yaml and update the values with values provided by the teacher. Save the file.



Test hands-on-mule-webcrawler-app

1. Open Postman and select **WebCrawlerAddToStore** request and click **Send**.

The screenshot shows the Postman application interface. At the top, it displays the URL `http://localhost:8081/webcrawler/add-to-store`. Below the URL, there are tabs for **POST**, **Auth**, **Headers (8)**, **Body**, **Scripts**, and **Settings**. The **Headers (8)** tab is currently selected. On the right side, there are buttons for **Save**, **Share**, and **Send**.

In the main area, under the **Query Params** section, there is a table with columns: **Key**, **Value**, **Description**, and **Bulk Edit**. The table is currently empty.

At the bottom, the **Body** section shows the response content:

```
{ } JSON ▾ ▶ Preview ⚡ Visualize ▾
```

```
1 {  
2 |   "status": "completed"  
3 }
```

The response status is shown as **200 OK** with a duration of **181 ms** and a size of **150 B**. There are also buttons for **Save Response** and other options.

2. Using a database explorer like DBeaver, review the `hands_on_vector` table.

DBeaver 25.1.2 - sampledb

sampledb public@sampledb

sampledb hands_on_vectors

Show SQL Enter a SQL expression to filter results (use Ctrl+Space)

Grid

	embedding_id	embedding	text	metadata
35	be9798d-dcce-4cc0-bb1t	[-0.012641472,0.01016923]	[!]{_.next}/image?url=%2F{("ingestion_timestamp": "1753166215855", "index": "0", "source_id": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")}	
36	f1d5a09-d601-43e6-baa1	[-0.014285213,0.0092795]	[Einstein AI]/docs/einst ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
37	b572132e-7b9f-4ec5-807e	[-0.0125570465,0.0151014]	[MuleSoft Inference]*[("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
38	263259d7-9f05-4aca-a721	[-0.034080684,-0.006960]	* [Use the Connector in Yo! ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/use-the-connector-in-yo")]	
39	59fe30c-1312-4421-a296	[-0.0037175387,0.033356]	Scroll to top¶Docs¶Multi ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
40	93e3b982-a43a-4ca6-89t	[0.0069607007,0.0176352]	* bash!mvn clean install - ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
41	93e3b982-a43a-4ca6-89t	[0.0069607007,0.0176352]	* bash!mvn clean install - ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
42	6c06f1fb-561f-4974-9be5	[-0.005143225,0.0404523]	* [Environment Variable ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
43	4d3b8505-12d9-4d6f-ba5	[0.05584264,0.00881689]	### A) Configuration JSON ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
44	83c795e8-f6ff-48ec-a0bb	[-0.029761804,0.02227005]	The file can be stored exter ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
45	6771dd0d-a154-4bc9-b241	[0.019300854,-0.023278]	** Temperature** is a numb ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")	
46	ac192c3-514e-45d3-932	[-0.012612994,0.01016945]	[!]{_.next}/image?url=%2F{("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")}]	
47	61e24637-7757-4b63-9c2	[-0.012539216,0.01509108]	[MuleSoft Inference]*[("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
48	ec2285a1-b6e8-4c6b-84c	[0.028084686,0.01982528]	* [MuleSoft Whisperer]/c ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-whisperer")	
49	767b6017-6d0a-4bde-8a4	[0.035063054,0.01303834]	Scroll to top¶Docs¶How ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-how")	
50	lee12815-211c-4eb4-a75d	[0.013991088,0.00824343]	* Go to your forked reposi ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-repository")	
51	9d4424c2-8356-42e8-b91	[0.037174392,0.003771410]	Various trademarks held by ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-trademarks")	
52	6d52c021-e038-4a4b-823	[-0.012612994,0.01016945]	[!]{_.next}/image?url=%2F{("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")}]	
53	7573a57d-7263-4f4b-a3fb	[-0.014329235,-0.009272]	* [Einstein AI]/docs/einst ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
54	6d06a581-77a9-4b9b-b45	[-0.012539216,0.01509108]	[MuleSoft Inference]*[("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
55	5b88fac9-b5af-4450-a12e	[-0.0004422022,0.022326]	* [MuleSoft Whisperer]/c ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-whisperer")]	
56	b235fac9-a2e3-4ca0-889	[0.005793553,0.02657445]	[Agent]/docs/mulechain- ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulechain")	
57	2e422e63-dfd1-4847-8e8t	[0.0030316864,0.0145959]	[!]{_.next}/image?url=%2F{("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")}]	
58	f8d2b309-b610-4749-9cd	[0.009054127,-0.0276573C]	* [Alex Martinez]/_next/ma ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
59	cb871284-88f9-4aff-aea0	[0.007769802,-0.0294620]	!! [Ettore Giallaurito]/_next/f ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
60	bcc5d7d-c06d-43fe-8f3b	[0.020680309,-0.00854192]	Salesforce Expert Demo fe ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
61	ed1b600e-9394-4e0e-8a9	[-0.007831838,0.0037384]	* [Vibhor Sharma]/_next/im ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
62	e4ba9220-1315-4118-b345	[-0.03544872,0.00480376]	## Amir Khan]/initiatorAc ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
63	6c13b674-57d6-48d9-be3	[0.015712345,0.02407750S]	## Dan Porat]/ambassad ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
64	619f4a17-d32b-43dd-994t	[0.020796172,-0.0330382]	These icons symbolize vari ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
65	ffd3c1e8-c599-40cd-a521	[0.040621076,-0.01241199]	These icons symbolize varí ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	
66	1c2661913-a895-4fd4-835	[0.031105466,-0.04014622]	Einstein Logo Represents ir ("ingestion_timestamp": "0696738d-b586-4cb3-a1b8-692f, "title": "Getting Started", "ingestion_datetime": "2025-07-22T06:36:15", "url": "https://mac-project.ai/docs/mulesoft-inference/getting-started")]	

66 row(s) fetched - 0.754s (0.002s fetch), on 2025-07-22 at 02:37:05

Refresh Save Cancel Export data 200 66