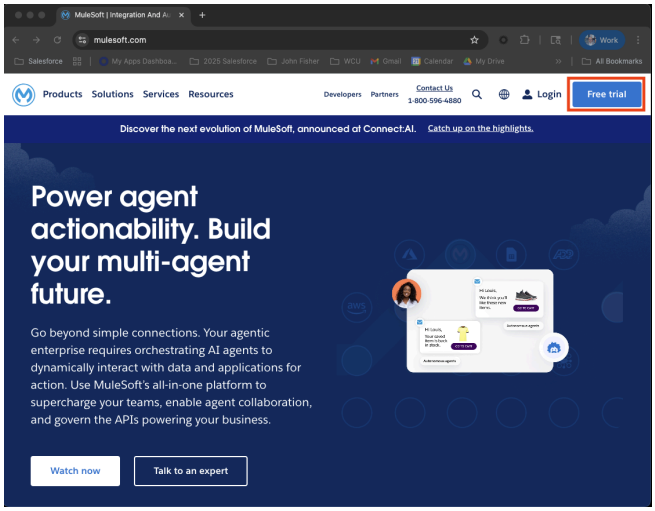
Mulesoft MAC Workshop

Table of Contents

Prerequisites

Signup for Free Trial MuleSoft Anypoint Platform

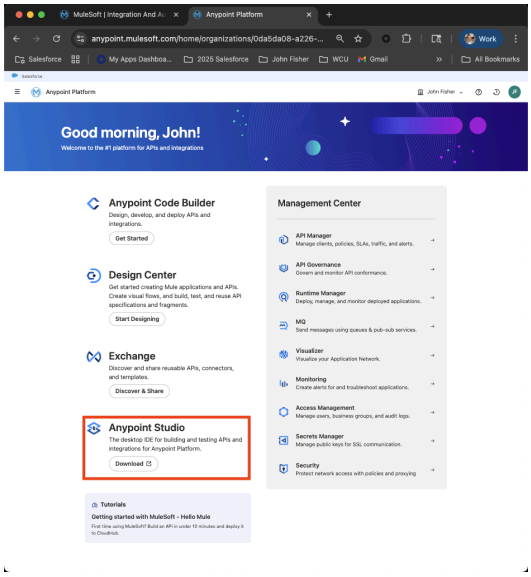
1. Sign up for a free trial of MuleSoft Anypoint Platform at www.mulesoft.com.



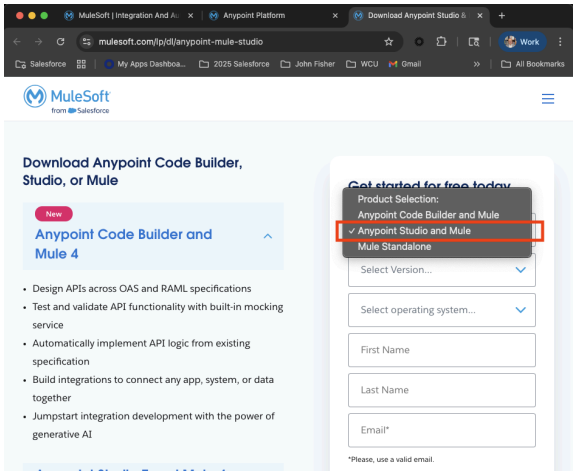
Download and Install Anypoint Studio

1. Using your credentials, Login to Anypoint Studio

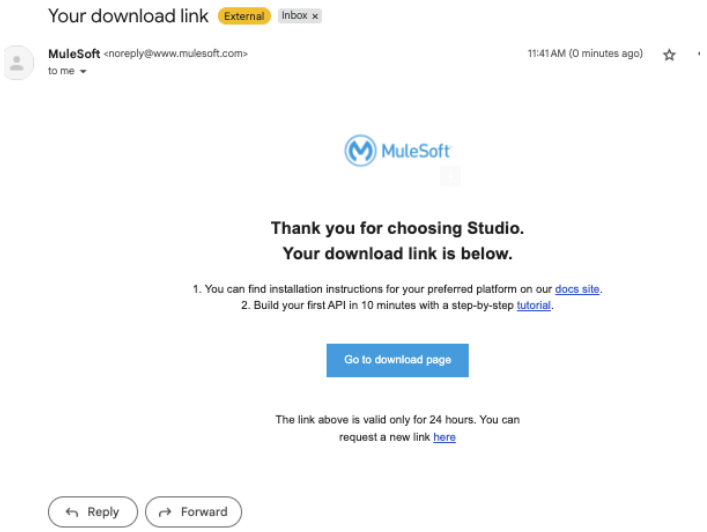
2. Click Download to open the download Anypoint Code Builder, Studio, or Mule webpage.



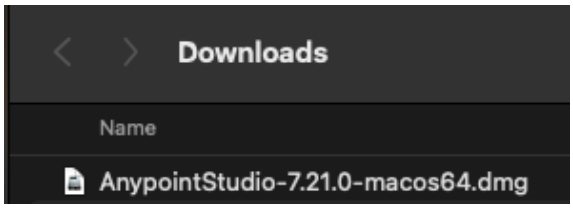
3. Select **Anypoint Studio and Mule**, fill in fields with your information and press download.



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

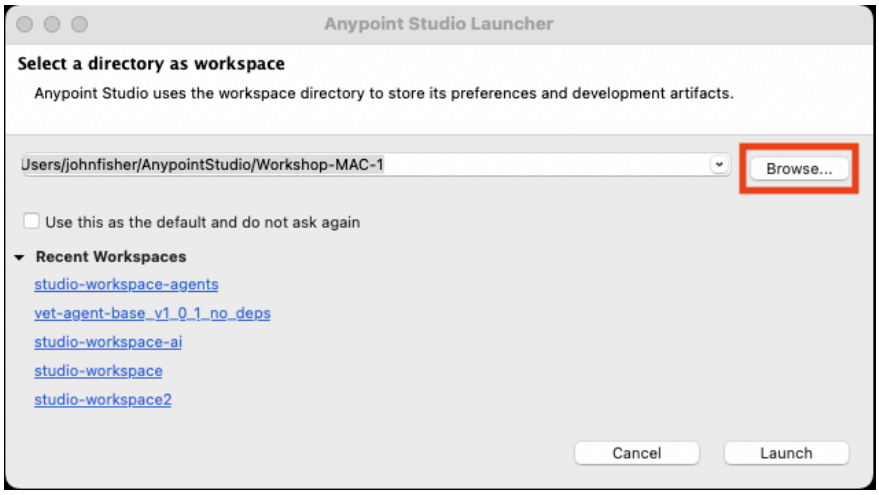


5. Click **Download .dmg** on a Mac (Or other file type specific to your OS). Once downloaded find the dmg file (Or other file type specific to your OS) and double click on it to install.

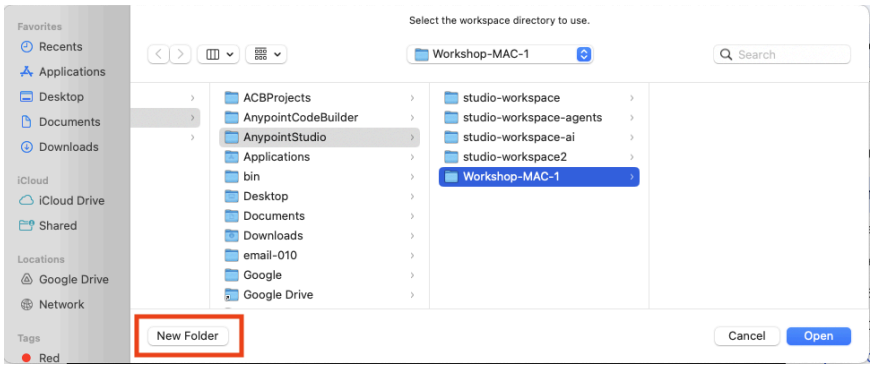


Create Anypoint Studio Workspace

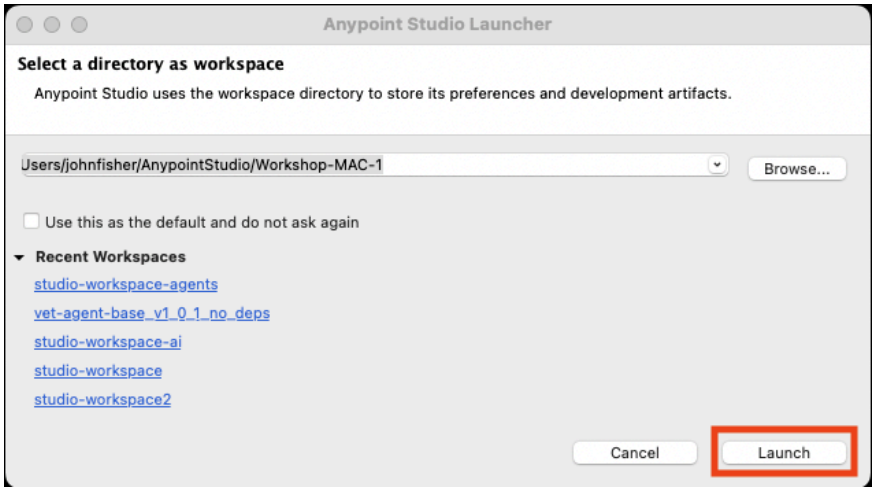
1. Open Anypoint Studio and create your workspace at directory path **<your path>/AnypointStudio/Workshop-MAC-1.** Click the **browse** button



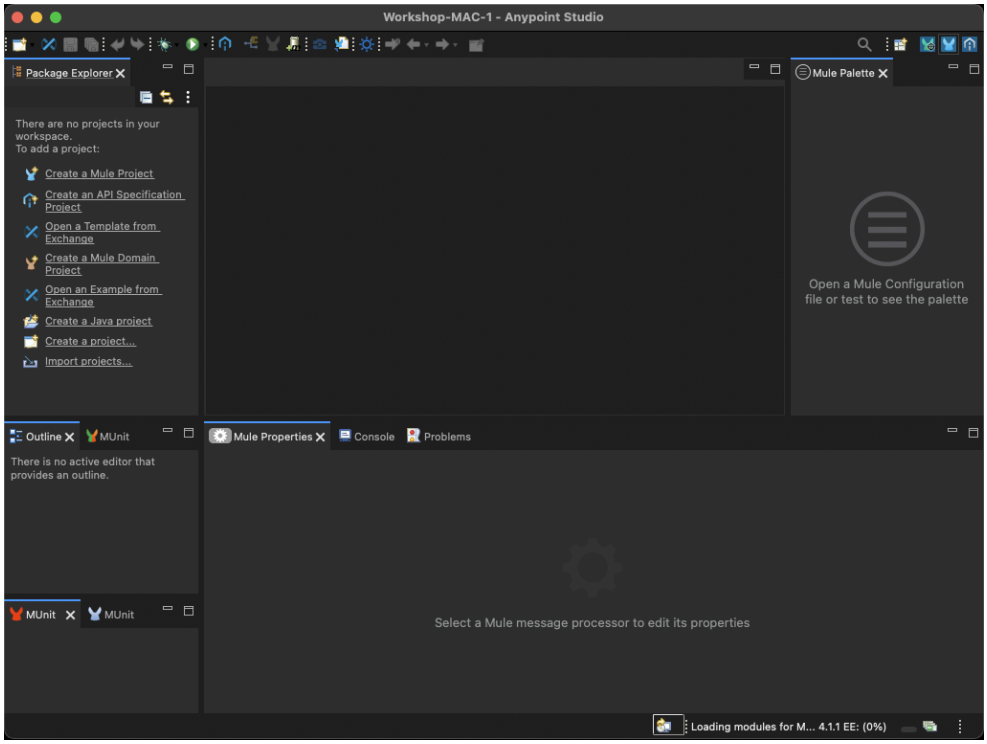
and Click **New Folder** to create a new folder in directory AnypointStudio (you may have to create this directory in your working directory first) named **Workshop-MAC-1**.



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

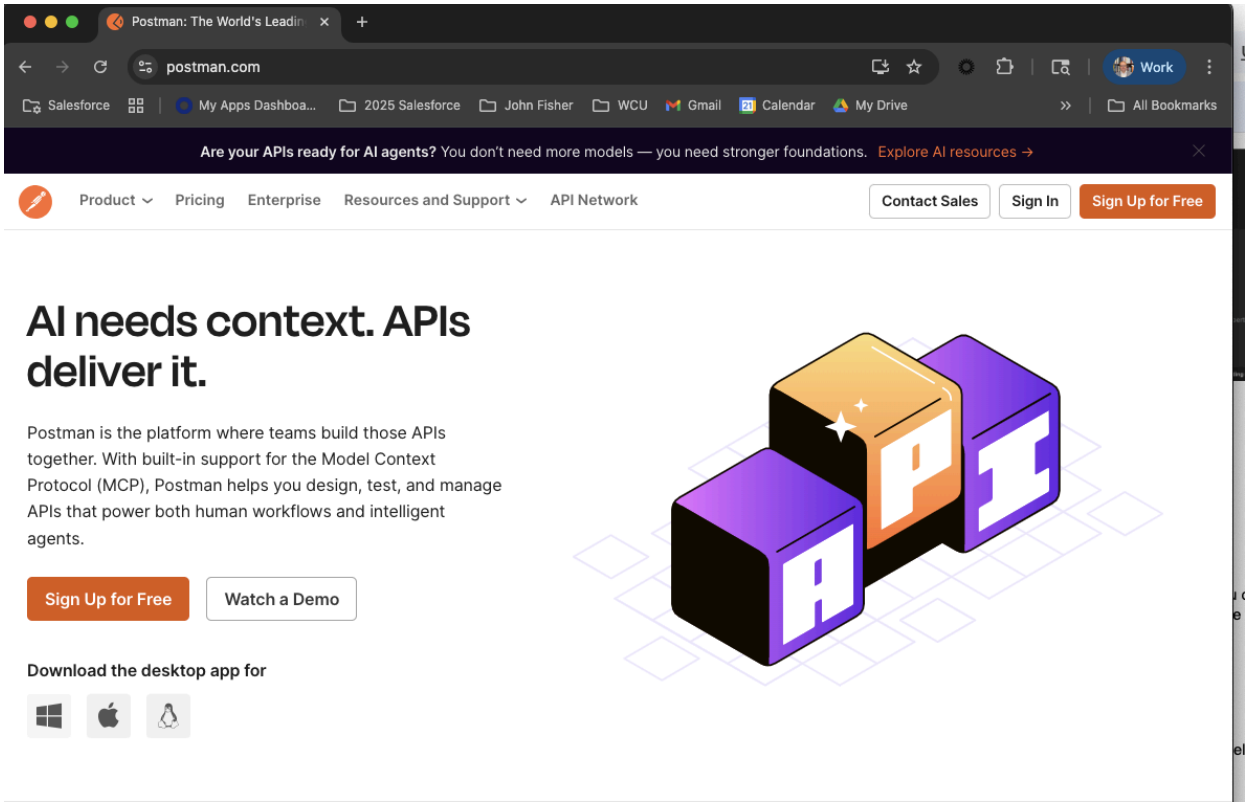


3. Anypoint Studio will open, Workshop-MAC-1 workspace will be open, with a default view as depicted below.



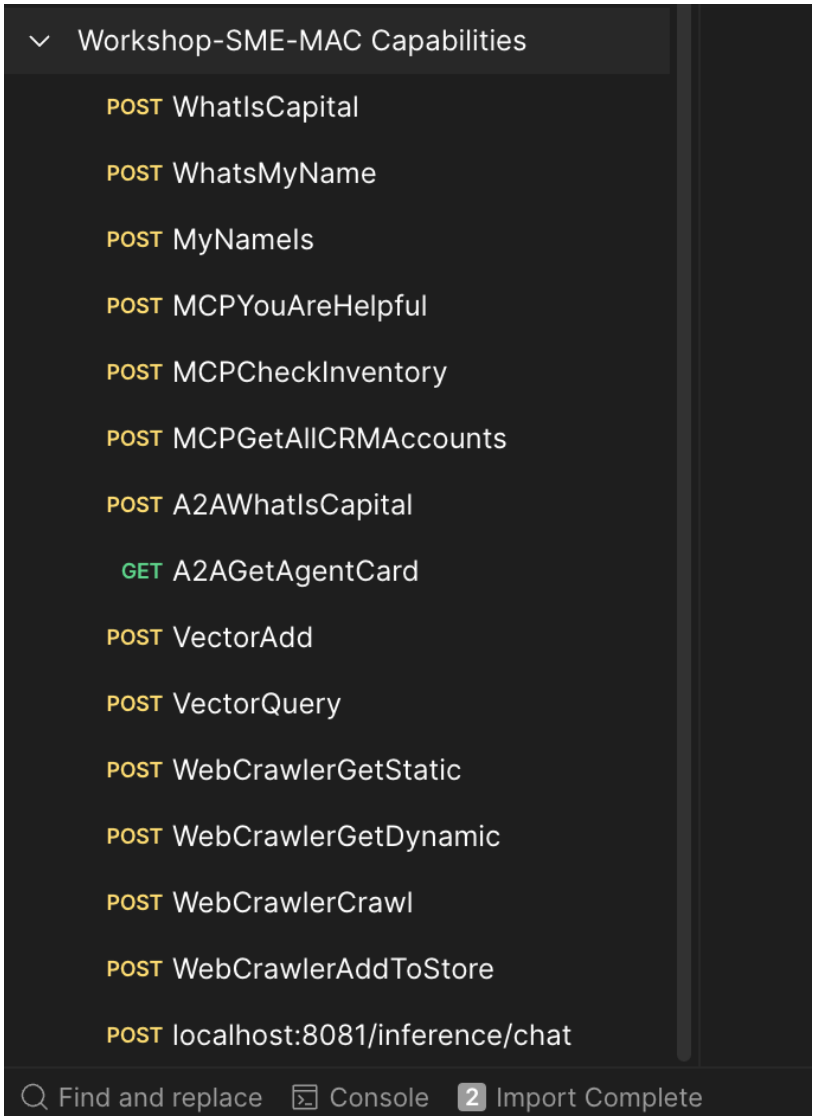
Install Postman

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



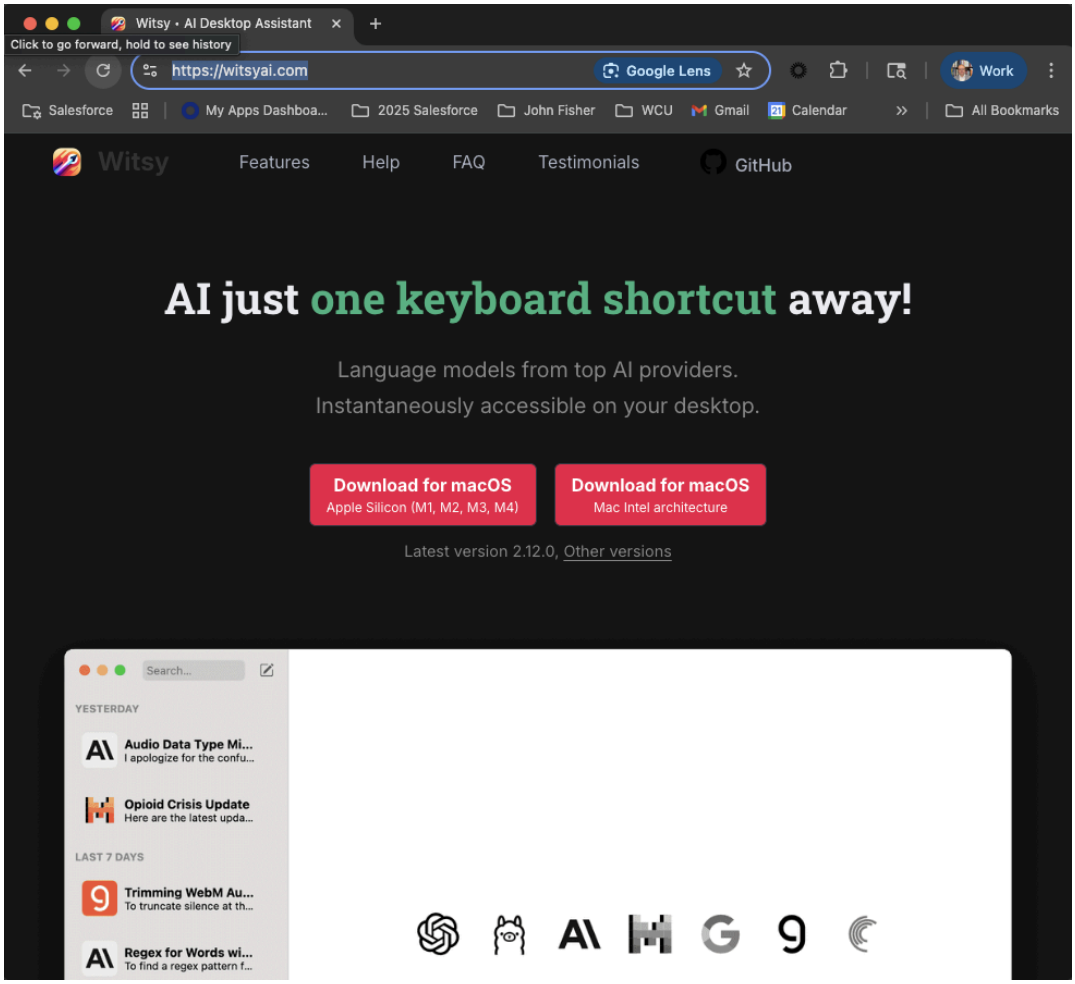
2. Download the **Workshop-SME-MAC Capabilities.json** from here and save to your desktop.

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.



Install Witsy

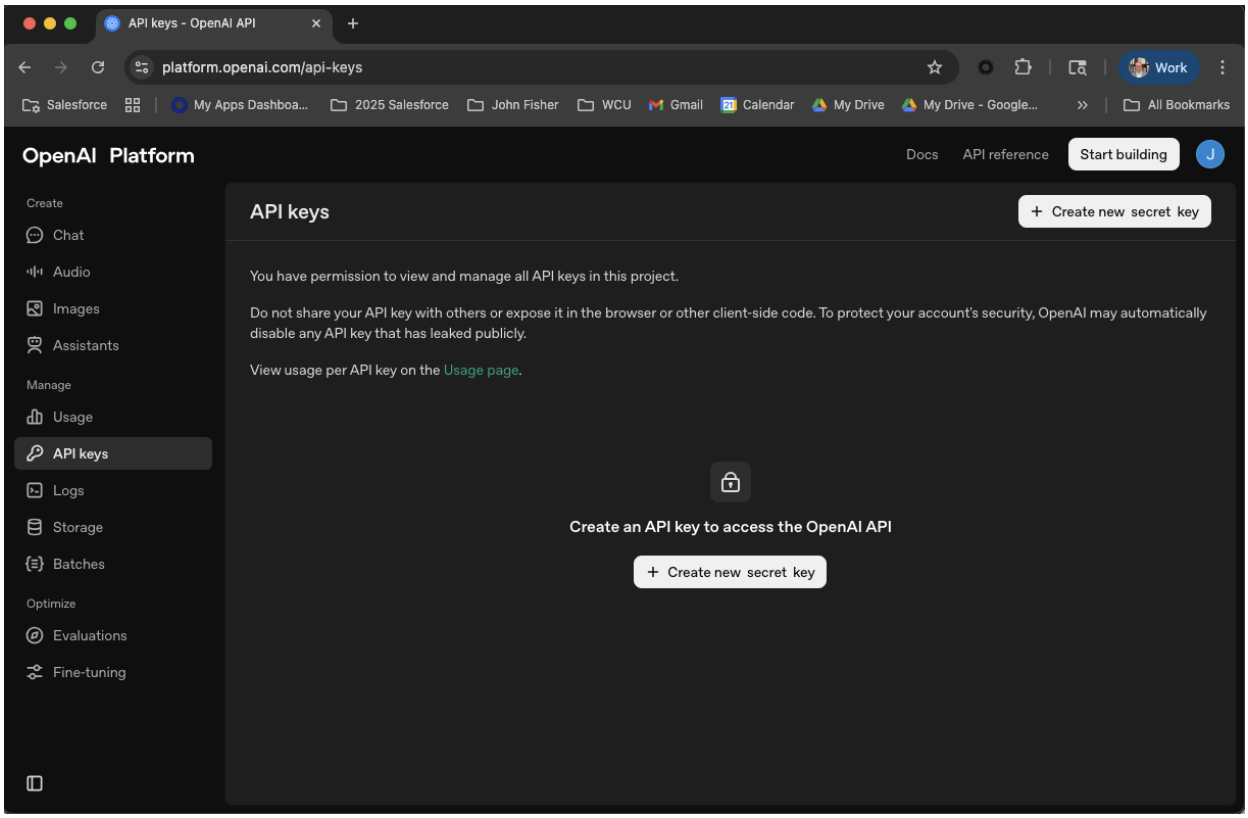
1. Install Witsy from https://witsyai.com/.



OpenAI API Key

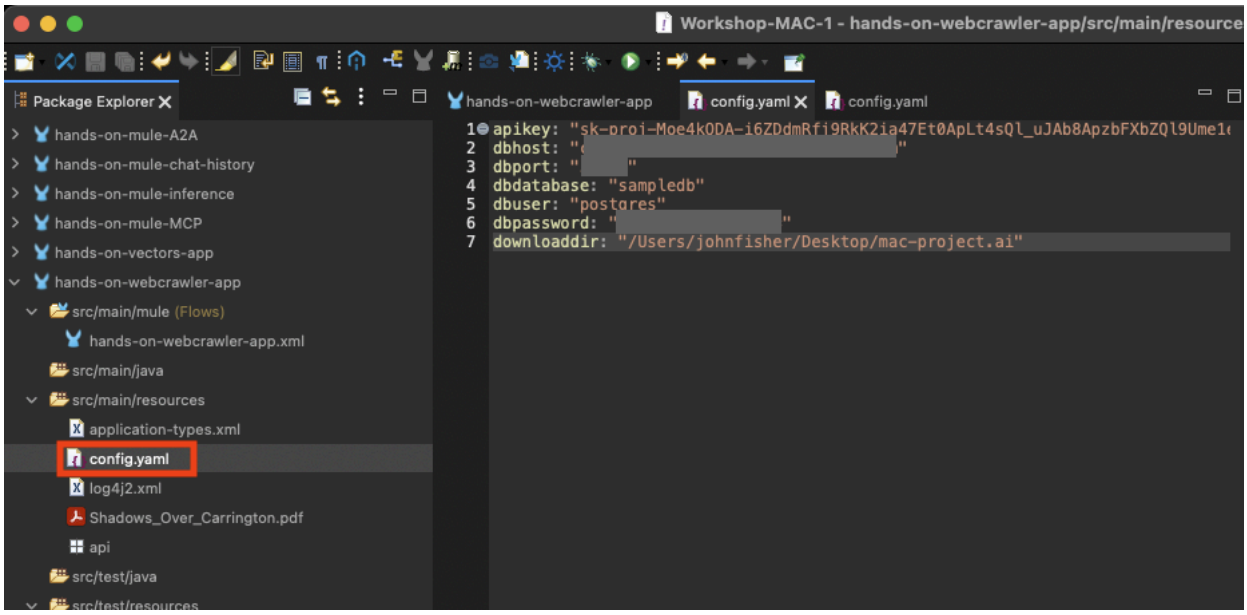
OpenAI API keys allow developers to access OpenAI's models through the API and build applications that can generate text, images, and code. To get an OpenAI API key, developers need to sign up for an OpenAI account and request an API key from the OpenAI dashboard. Developers should keep their API key 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.



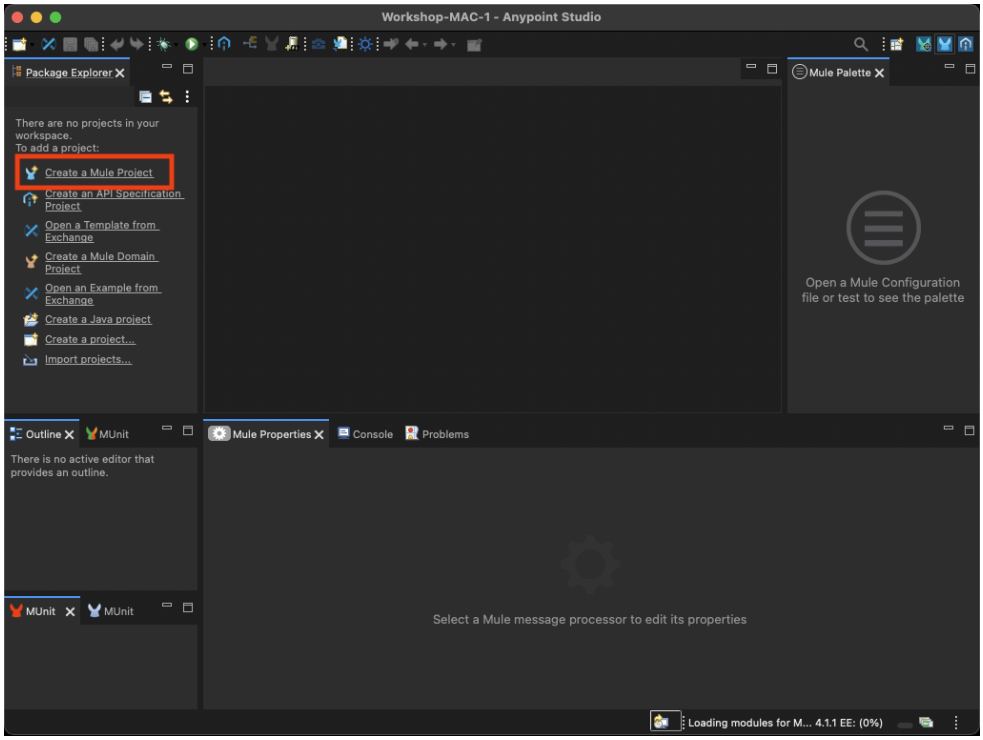
Inference Hands On Exercises

Exercise 1 - Simple Chat App

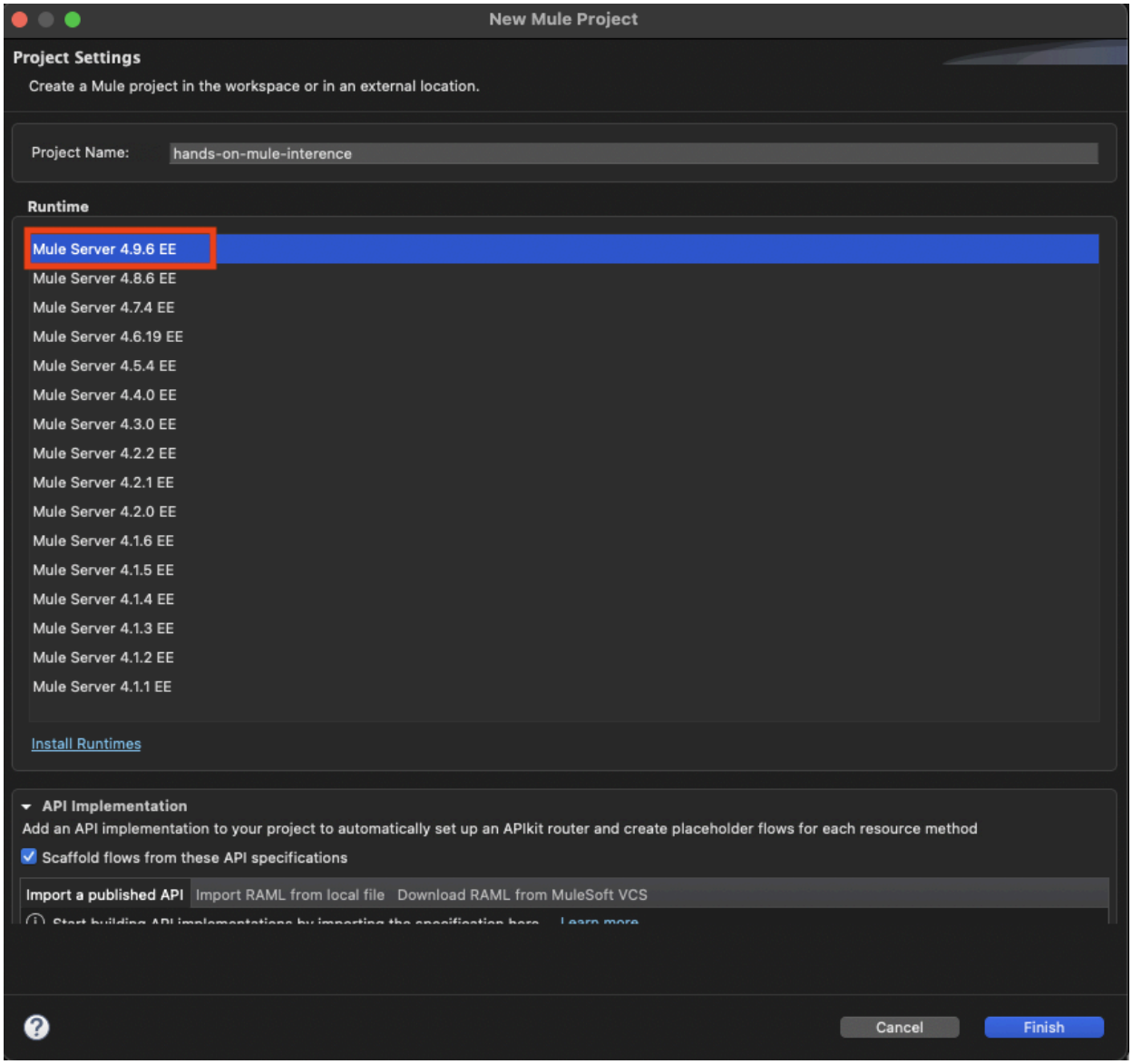
The purpose of this exercise is to demonstrate how you can use the chat completion operation of LLM to implement a workflow in MuleSoft. In this exercise we are going to build a simple headless chat app with an LLM.

Create a Mule Project

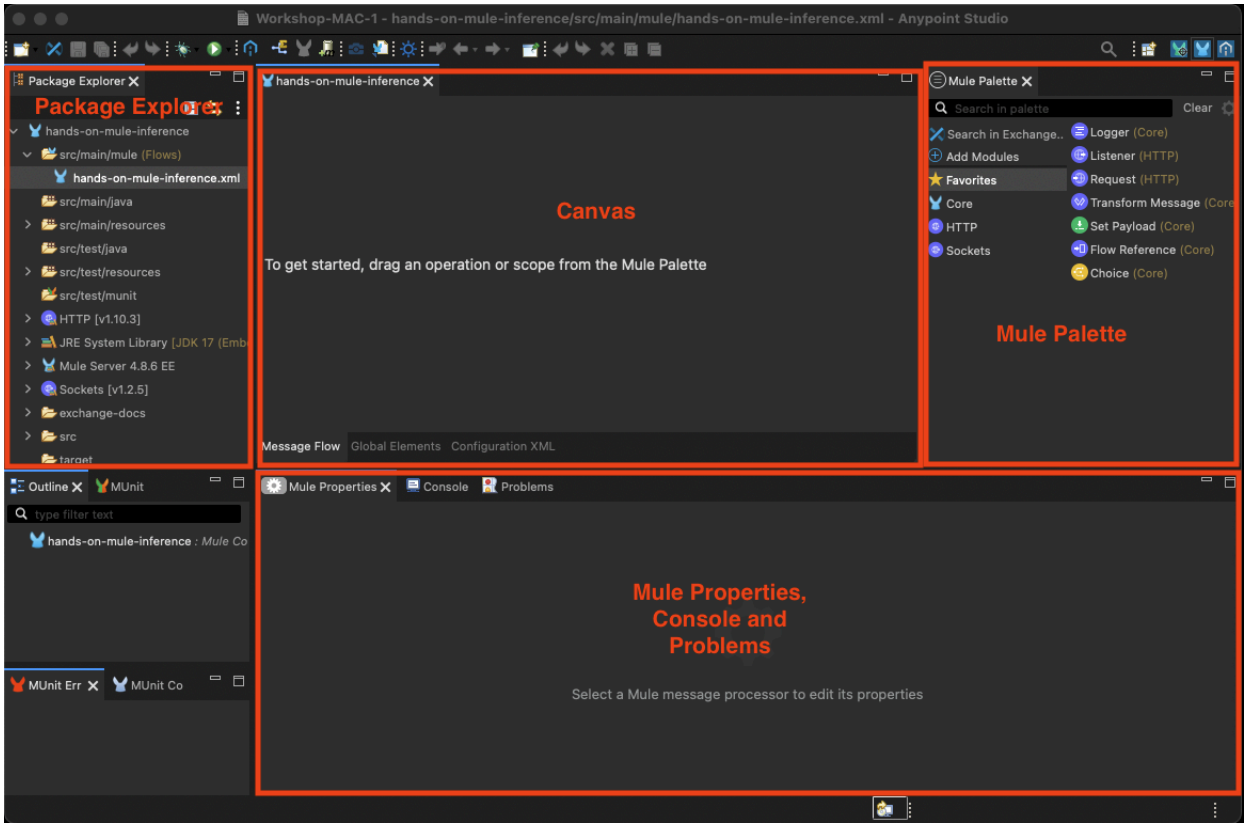
1. Create a Mule Project hands-on-mule-inference by selecting **Create a Mule Project** to add a project to your workspace.



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

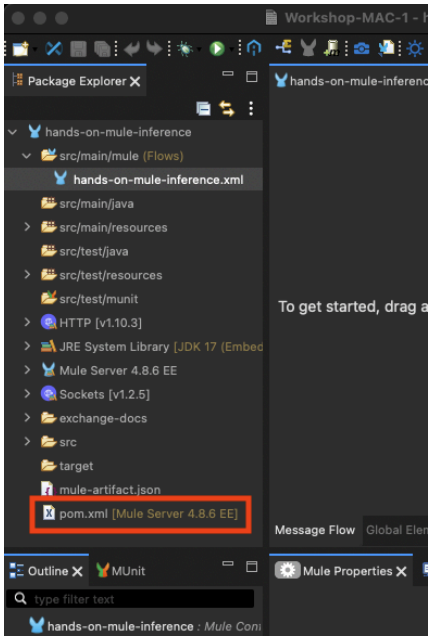


Your hands-on-mule-interface project will be created and your design canvas will be opened (see below image), package explorer on left will show your projects assets, you will see the Mule Palette on the right with components and activities, and towards the bottom is the mule properties, console and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



Add Dependency

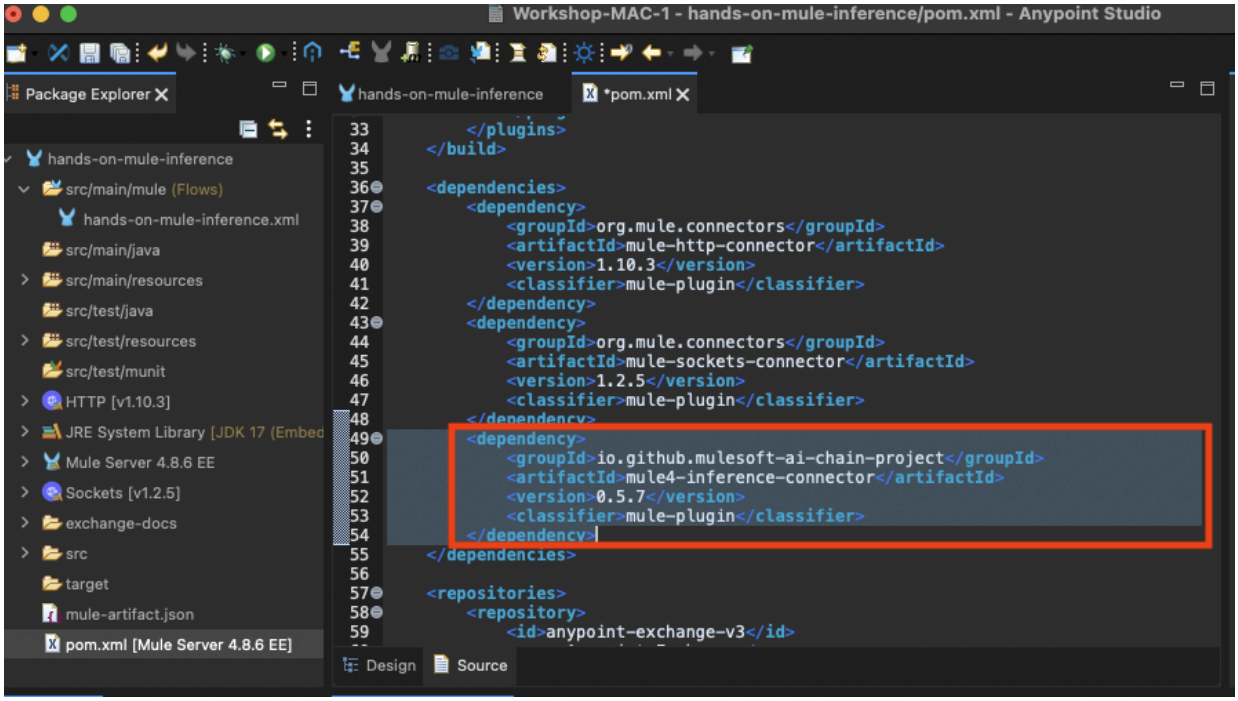
1. Double-click on **pom.xml** in the package explorer to open it.



2. Copy the following dependency XML entry.

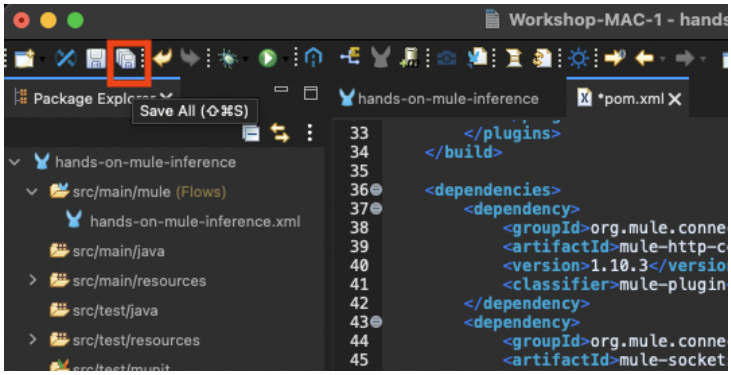
```
<dependency>
      <groupId>io.github.mulesoft-ai-chain-project</groupId>
      <artifactId>mule4-inference-connector</artifactId>
      <version>0.5.7</version>
      <classifier>mule-plugin</classifier>
</dependency>
```

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

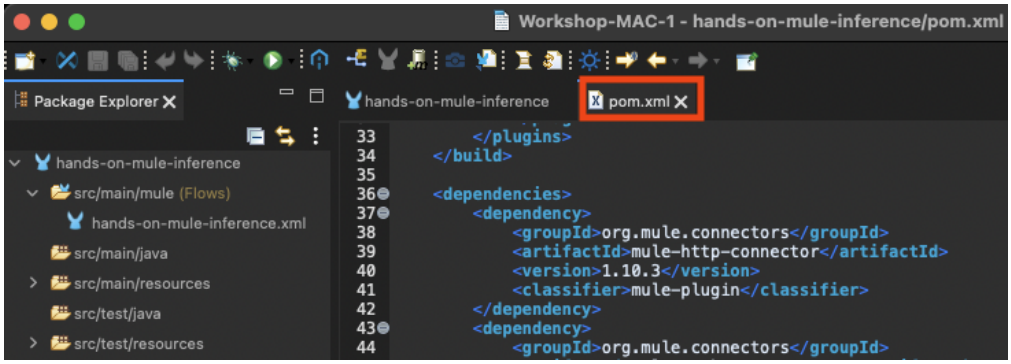


4. Click **save all icon** (highlighted in the screenshot) in the toolbar, or press ⌘S (macOS) / Ctrl+S (Windows/Linux) to save your project. This will add the inference connector to your project.

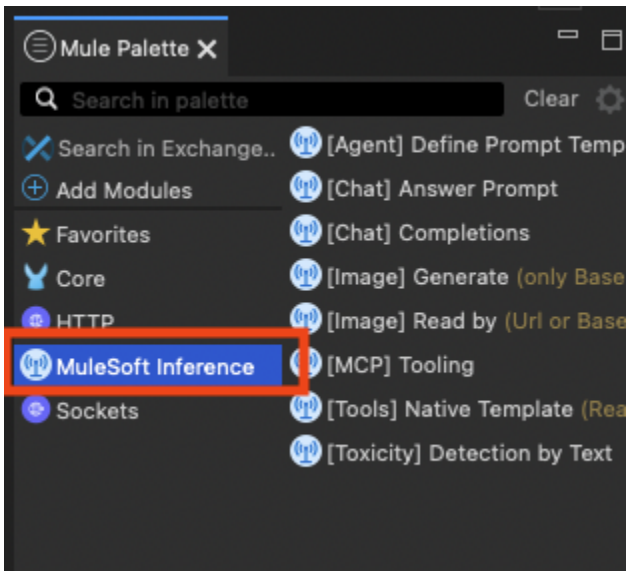
Note: Once Inference Connector is Generally Available (GA), it will be accessible via Exchange.



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

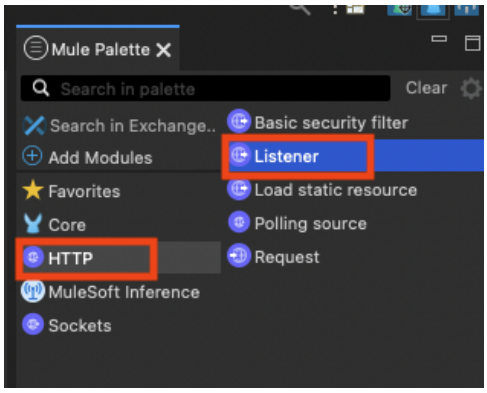


6. Verify “MuleSoft Inference” appears in the Mule Palette (highlighted in the screenshot below).

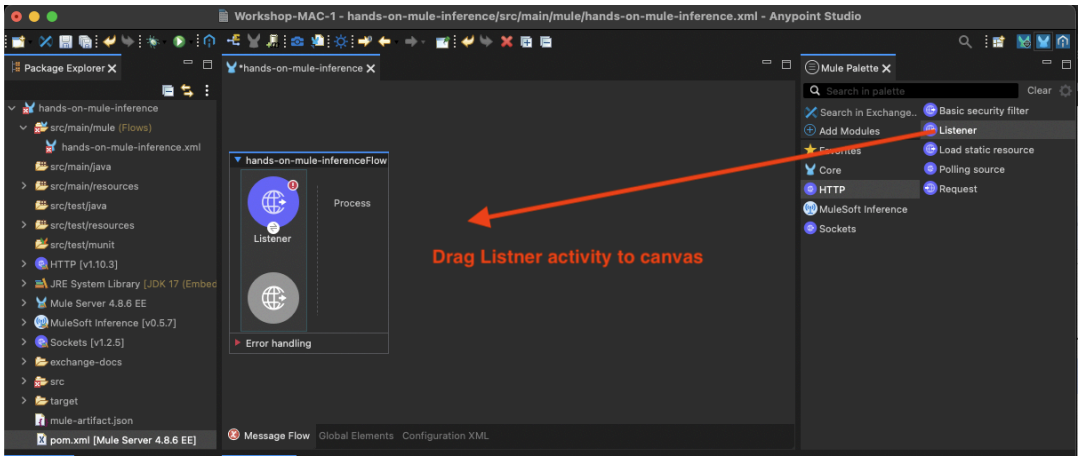


Build Simple Chat Interaction

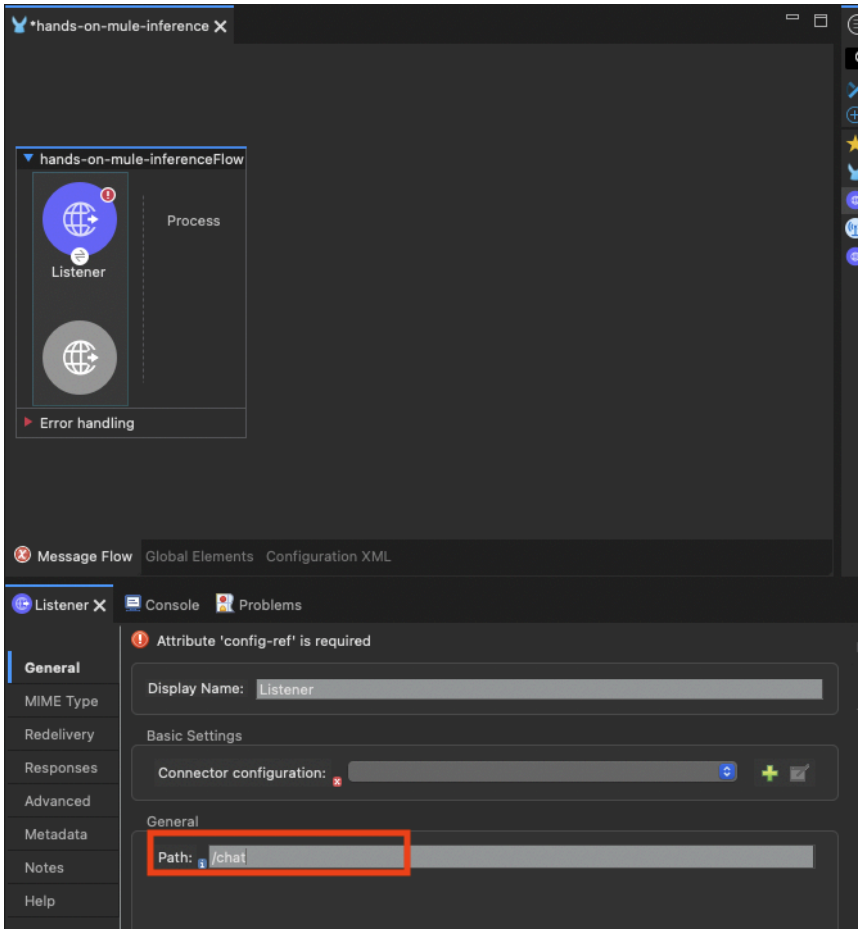
1. From the Mule Palette, select **HTTP**, then



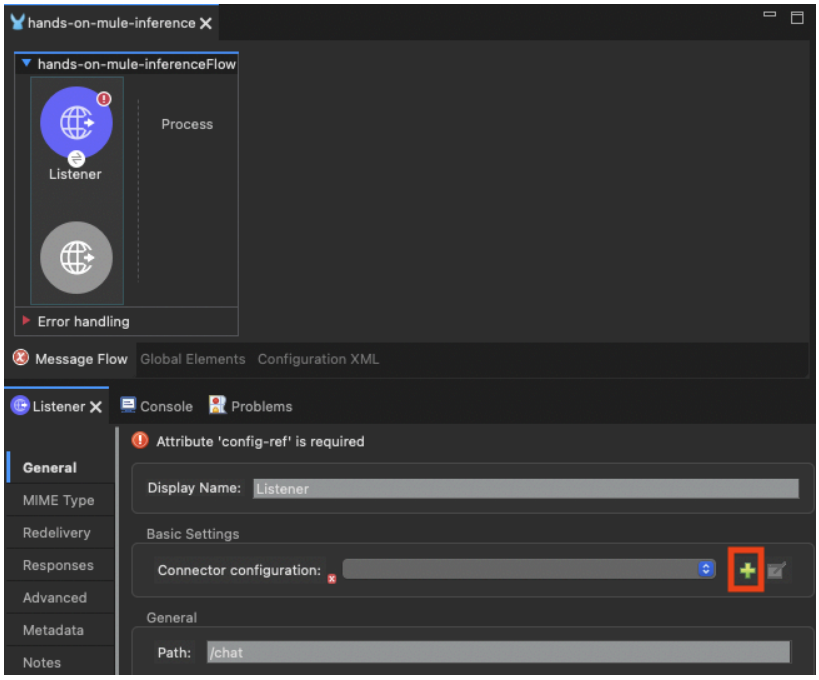
drag the **Listener** activity to your design canvas.



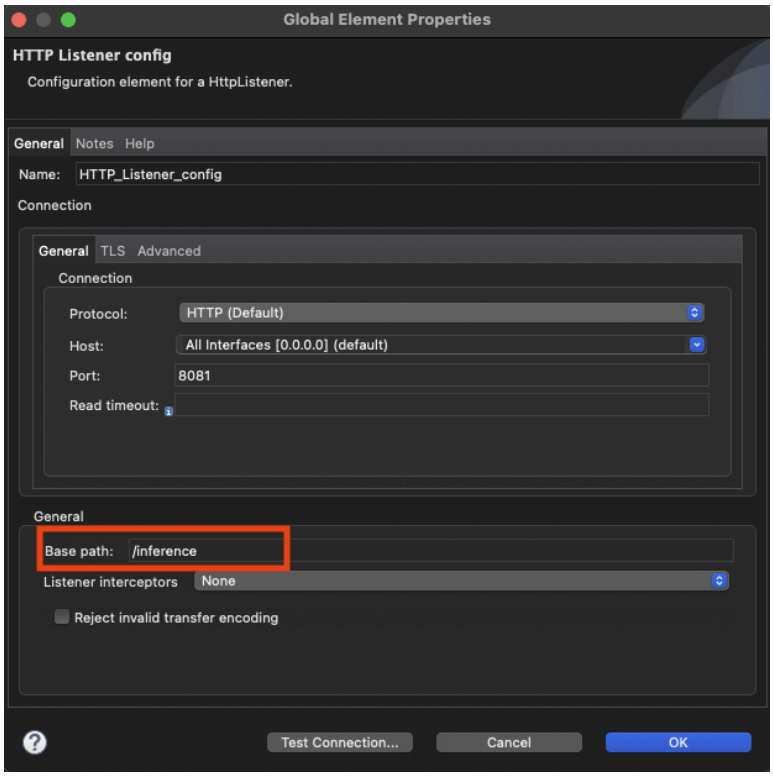
2. In the Listerner's configuration, set the **Path** to /chat (as highlighted in the screenshot).



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.

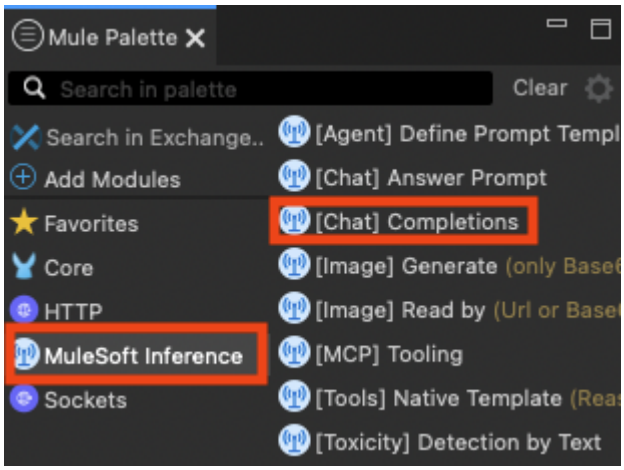


4. In HTTP Listener config, set the **Base path** to /inference (as highlighted in the screenshot). Click on **OK** to close the HTTP Listener config dialog.

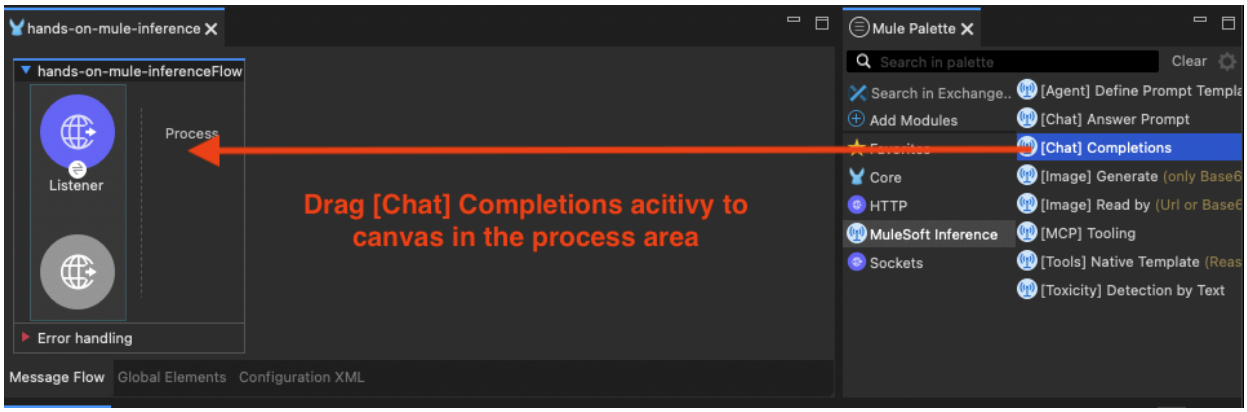


5. Save your project.

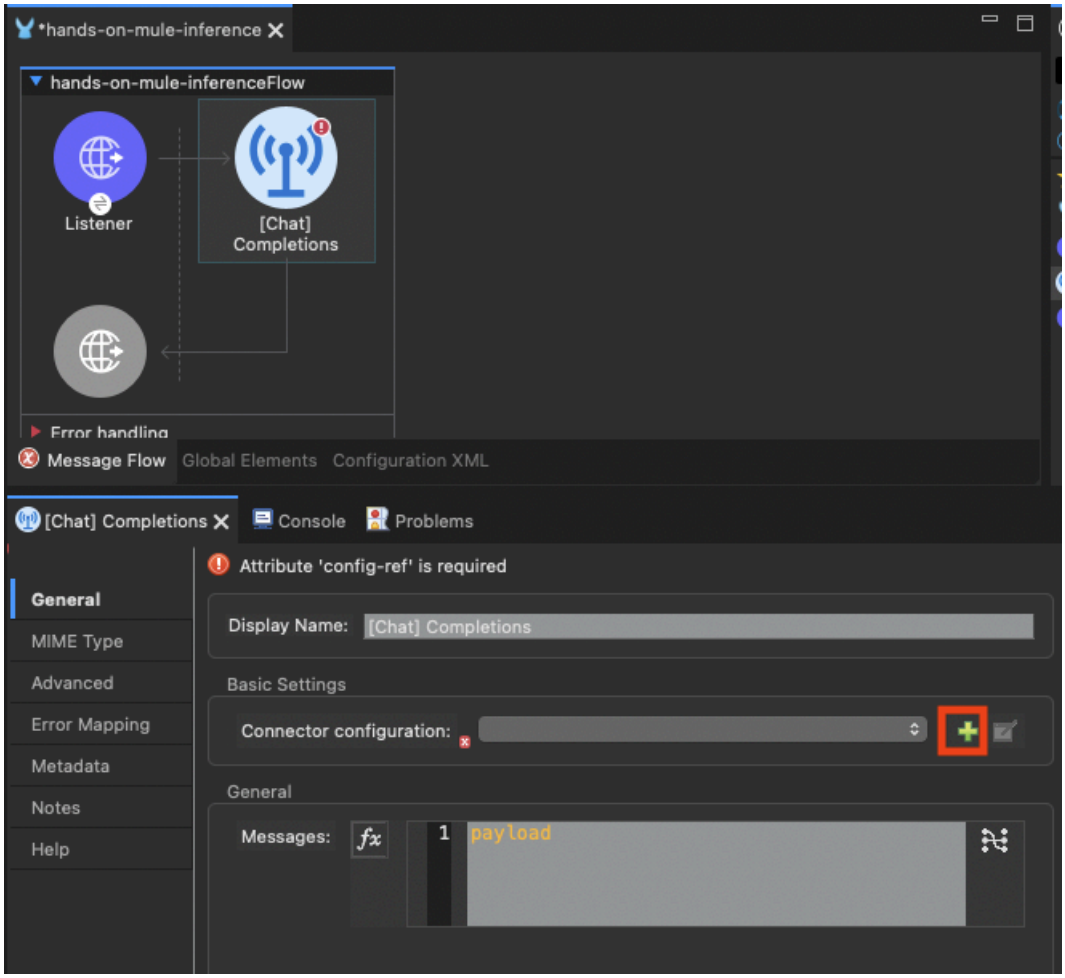
6. From the Mule Palette, select **MuleSoft Inference**,



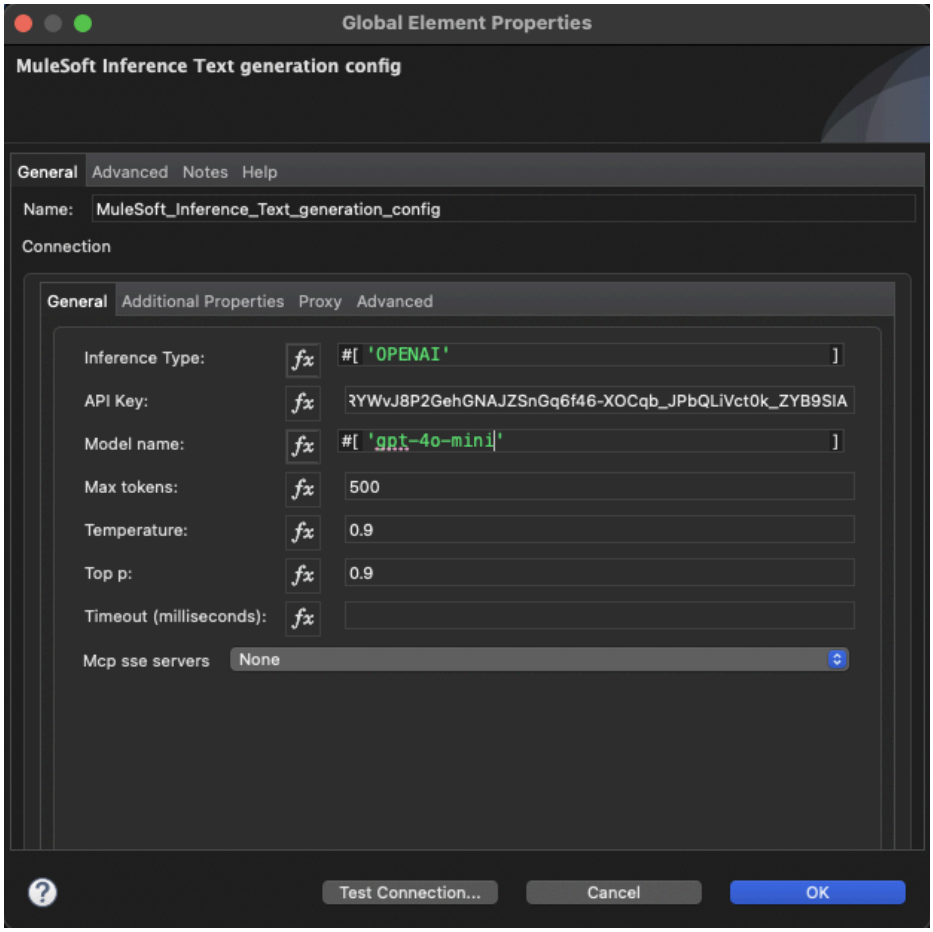
then drag the **[Chat] Completions** activity to your design canvas and drop in the **Process** area.



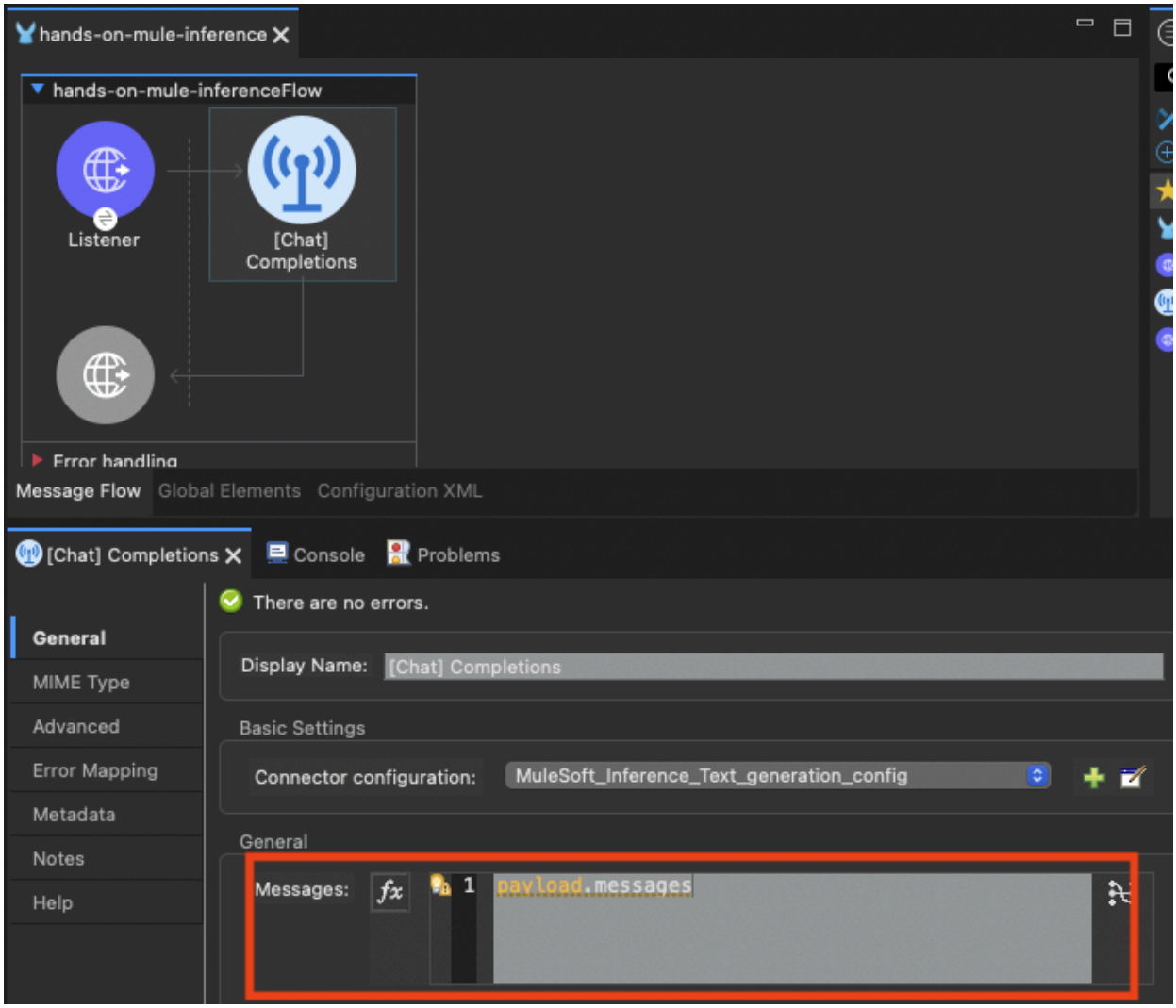
7. To create a connector configuration, click the green **'+'** icon next to “Connector configuration” (highlighted in the screenshot). This will open the MuleSoft Inference Test generation config.



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.

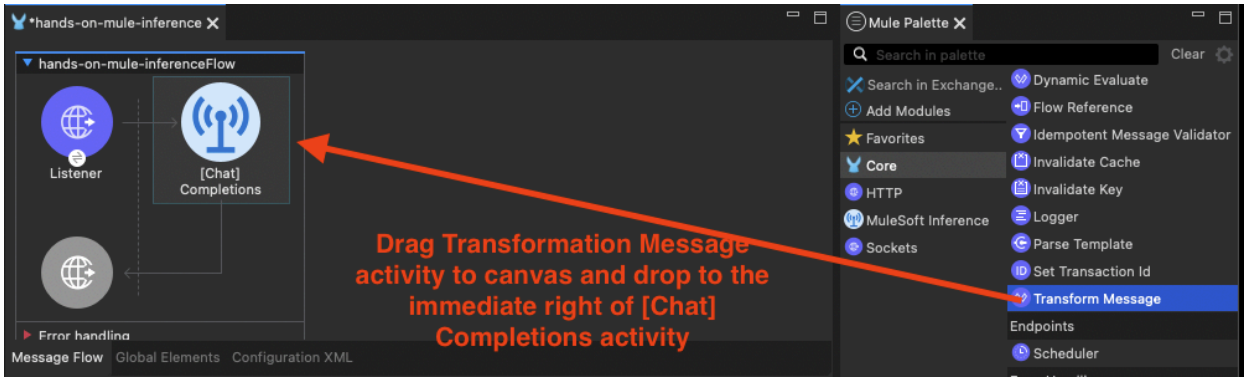


9. Update **Messages** to payload.messages as shown in the screenshot below.



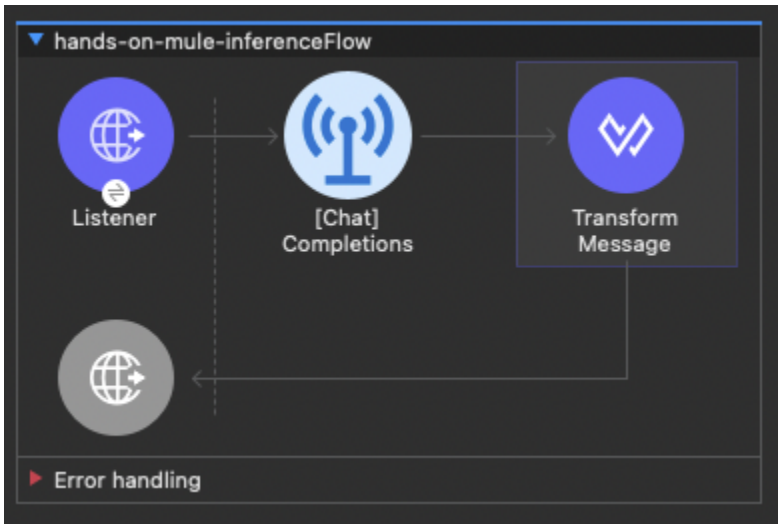
10. Save your project.

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.

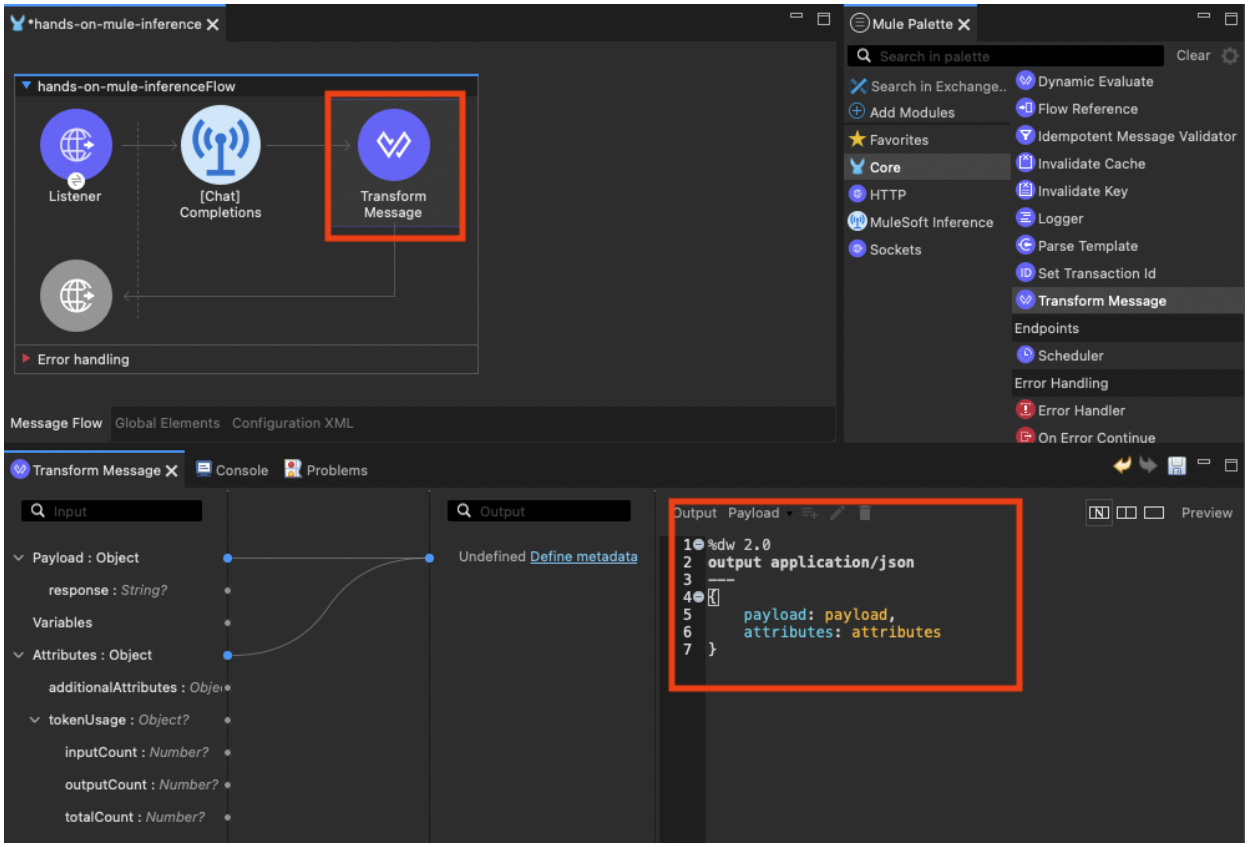
Your flow should look like this.



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

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

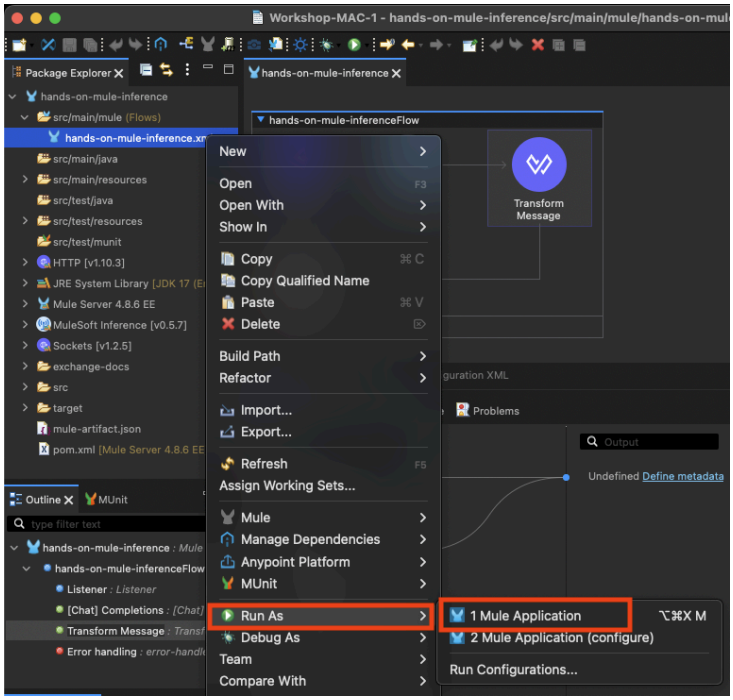
and paste the expression into the **output payload** as shown in the screenshot below.



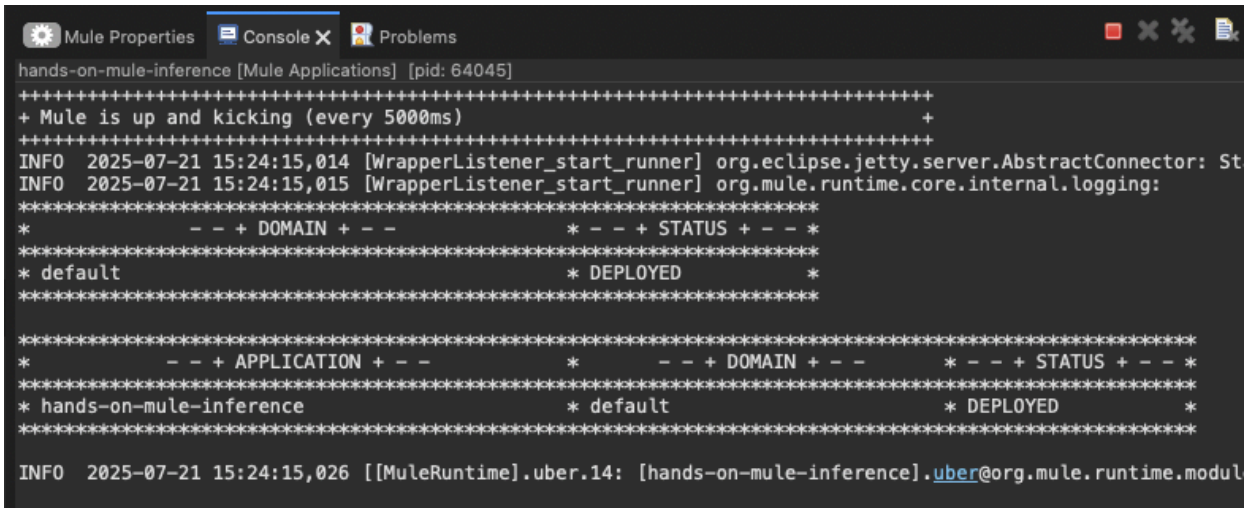
13. Save your project.

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.

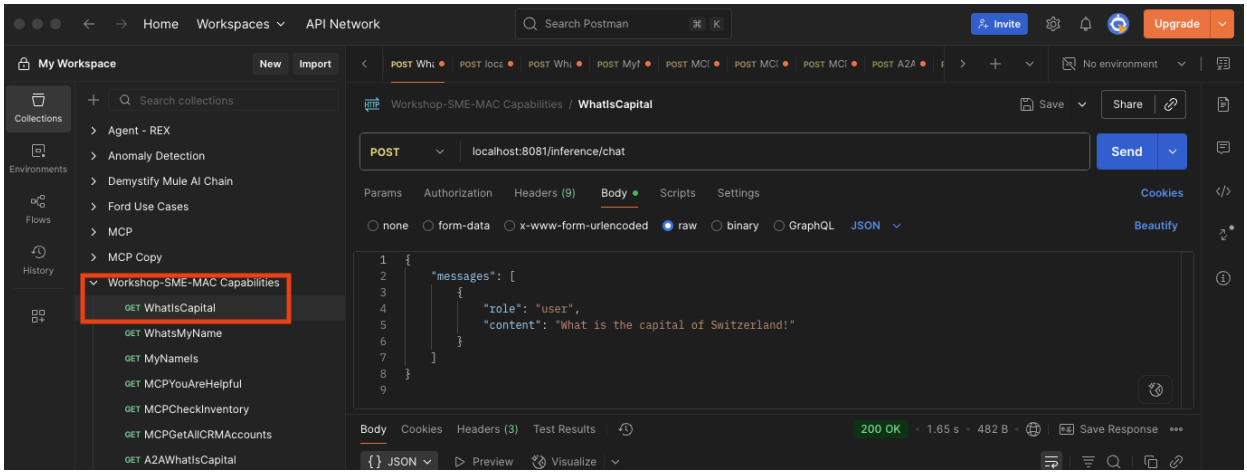


2. Your mule project will be started. Please wait for application status deployed in the console window as shown in screenshot below.

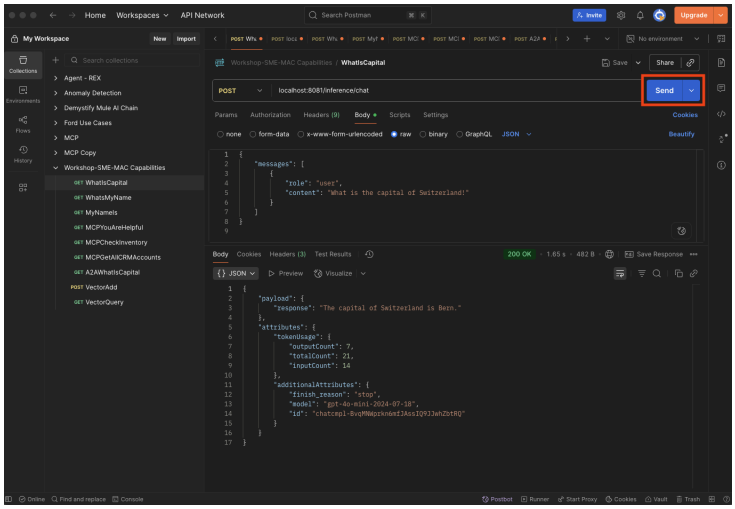


Test Your Mule Application

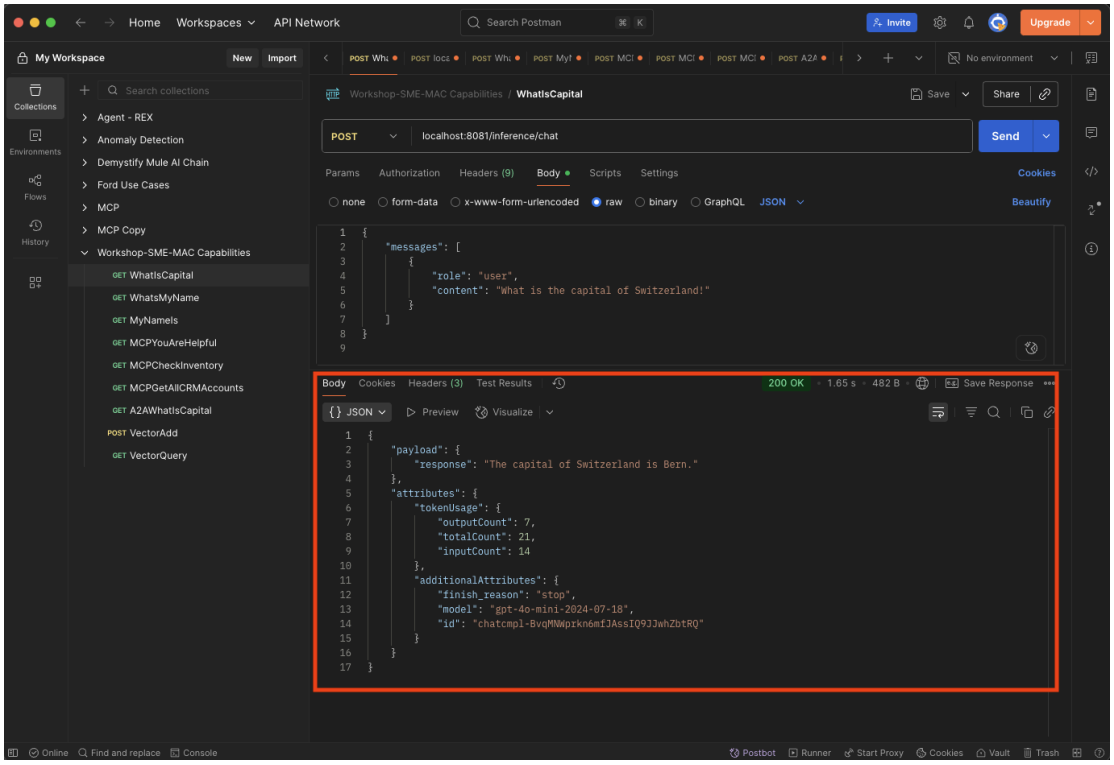
1. Open Postman and select **WhatIsCapital** request.



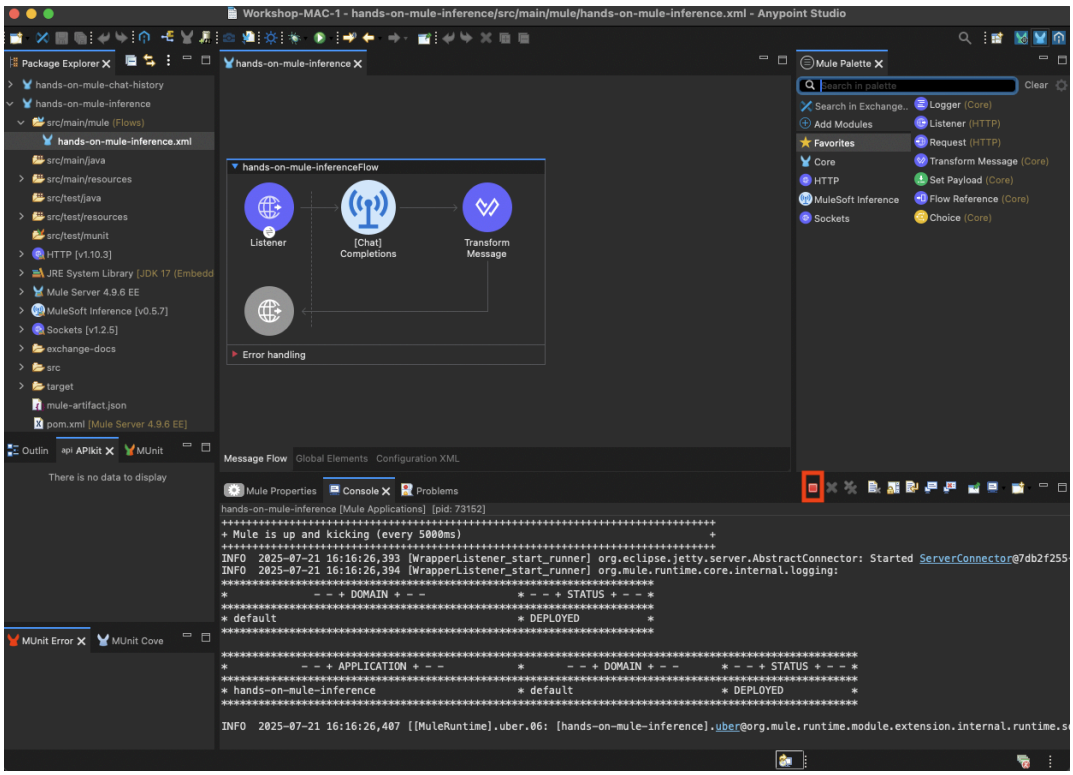
2. Click **Send** to Post.



3. MuleSoft will respond with the following response.



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

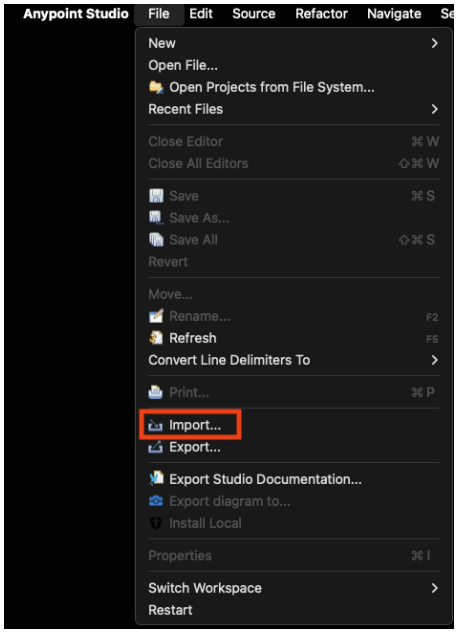


Exercise 2 - Enabling Chat History

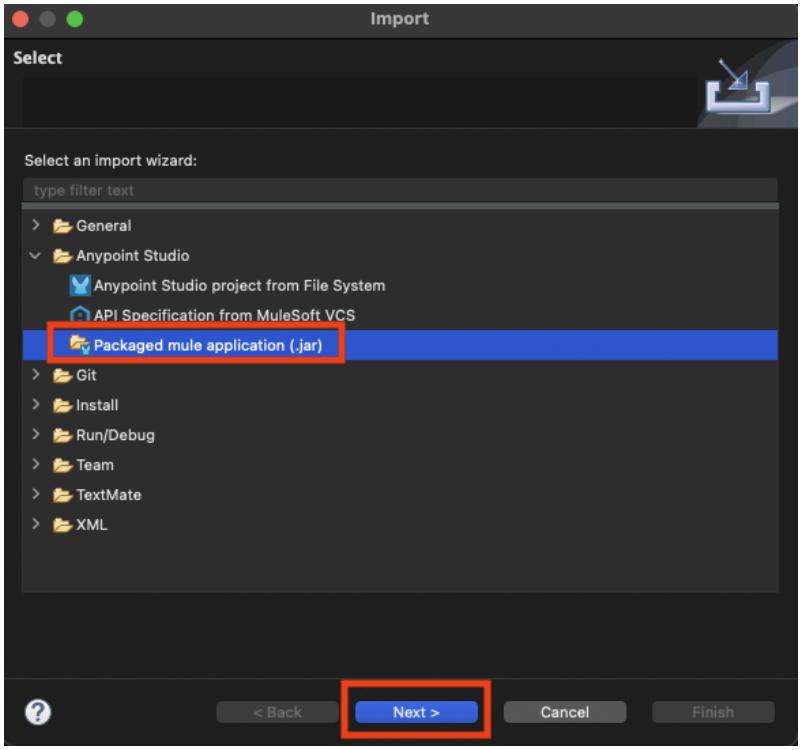
When using Chat Completions activity, providing chat history is crucial for maintaining conversational context. You have control over which past interactions to include to optimize performance and relevance.

Import and Explore Project

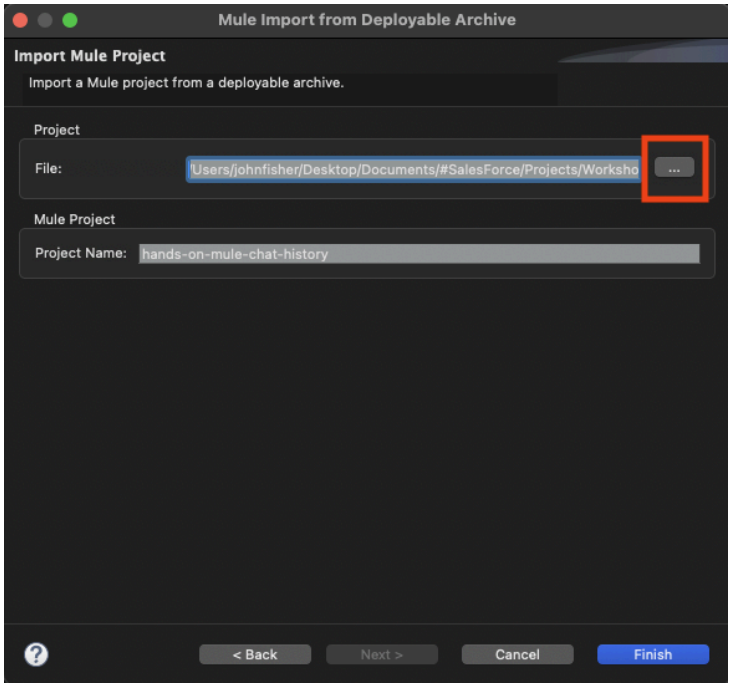
1. Import **hands-on-mule-chat-history** into your workspace. Select File->Import…



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

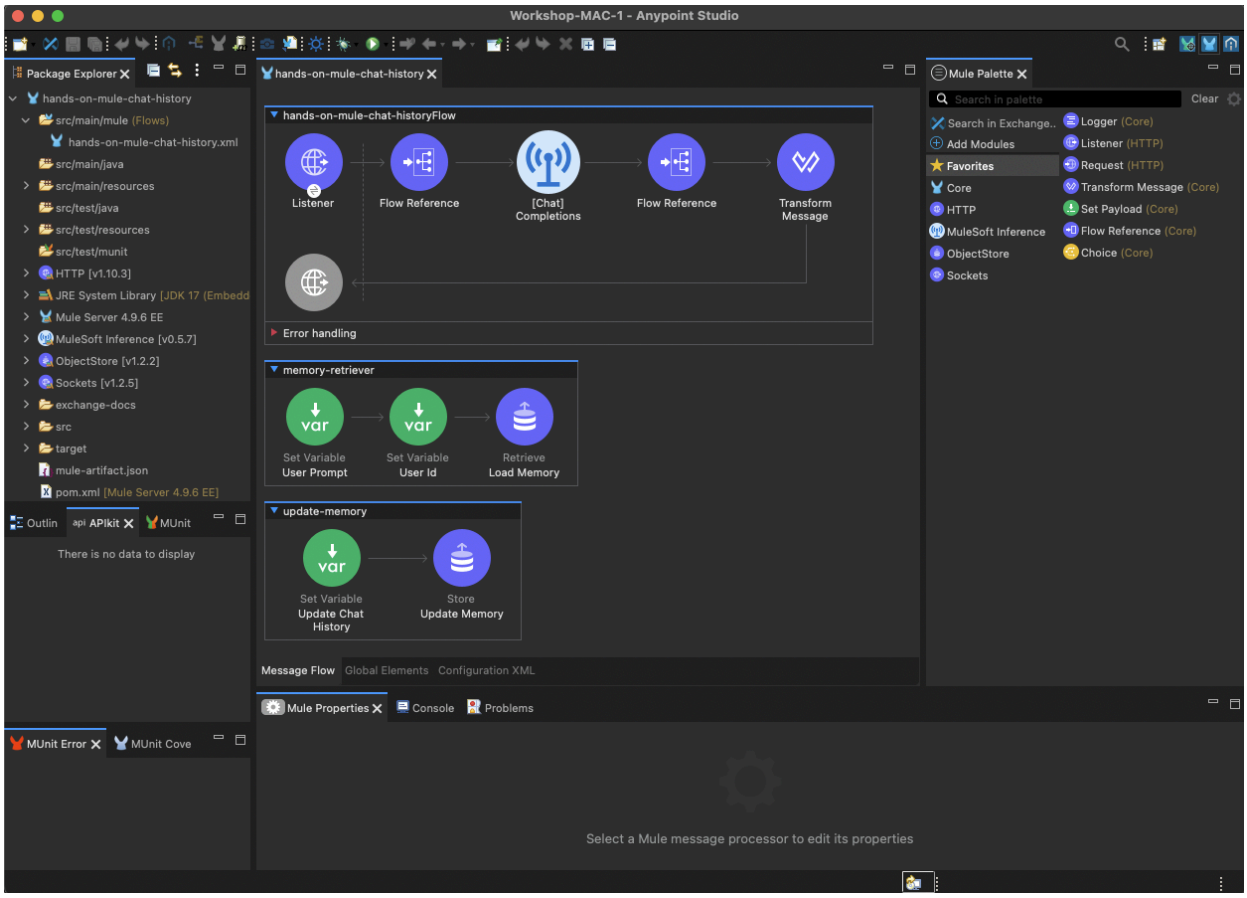


3. Click the "..." icon and browse to the directory of **hands-on-mule-chat-history.jar**, select hands-on-mule-chat-history.jar and click **open**.

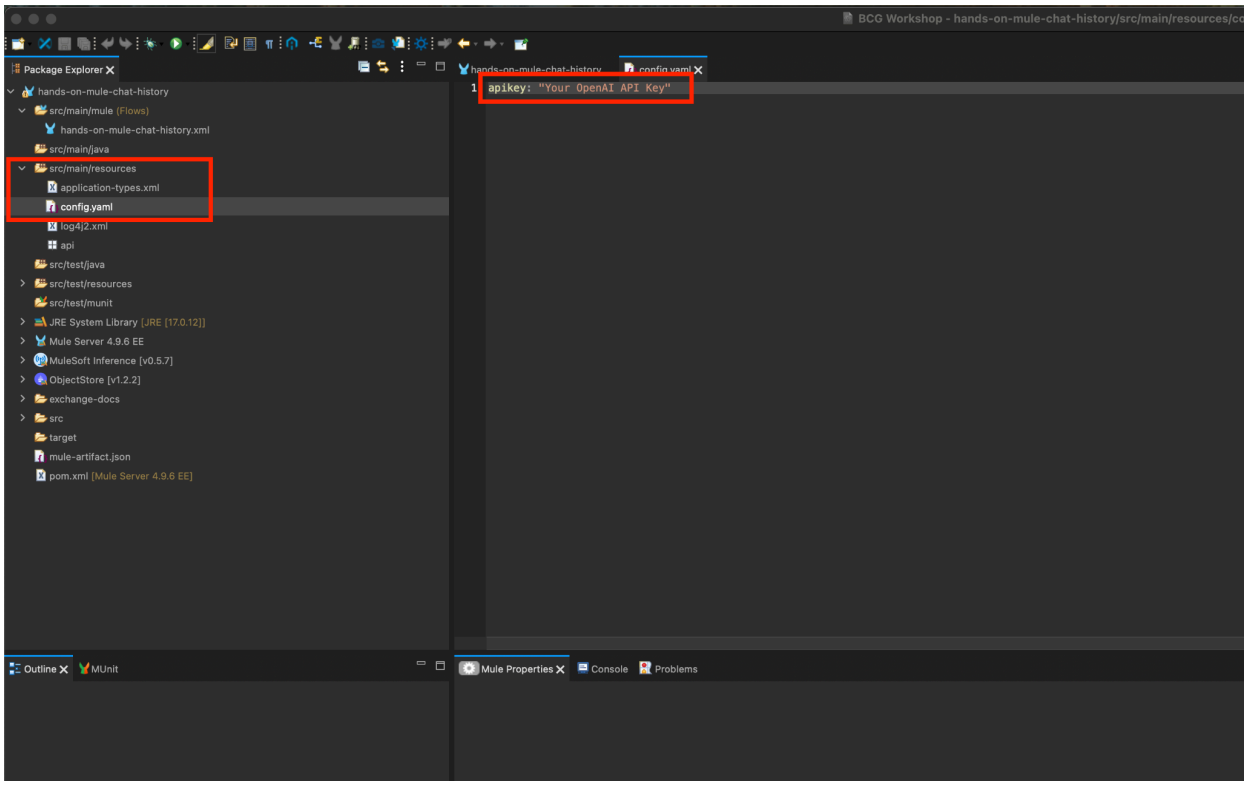


4. Click Finish.

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.

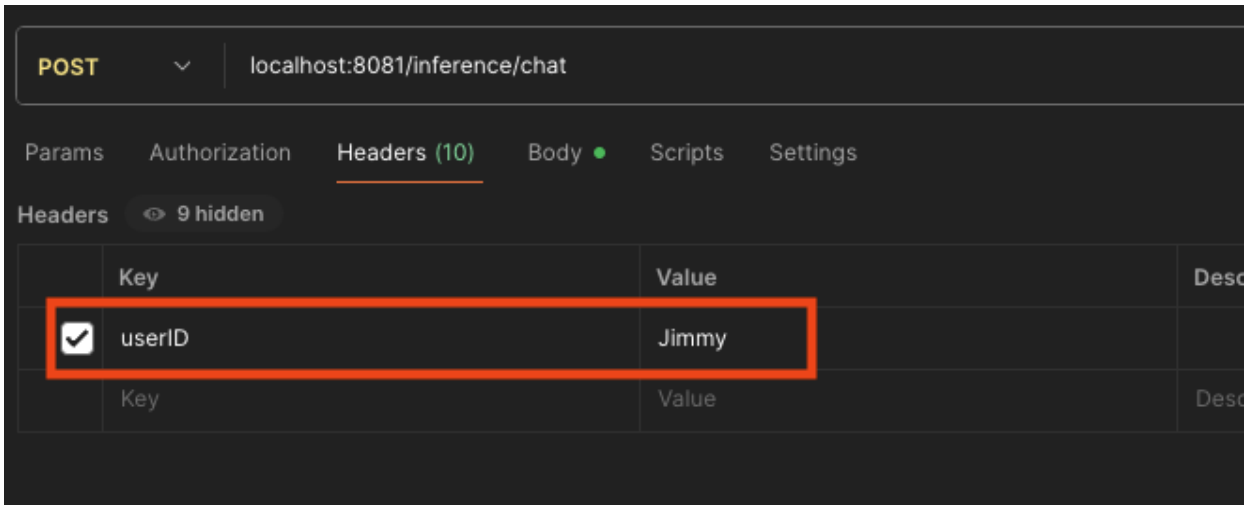


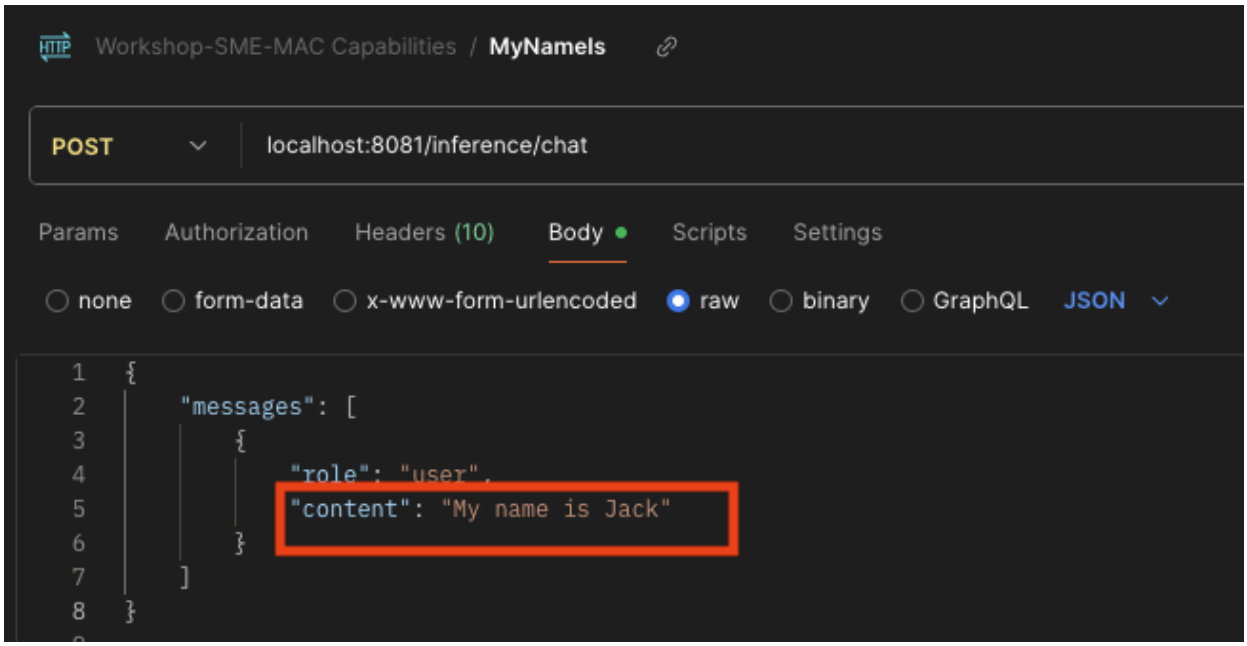
4. Run hands-on-mule-chat-history locally.

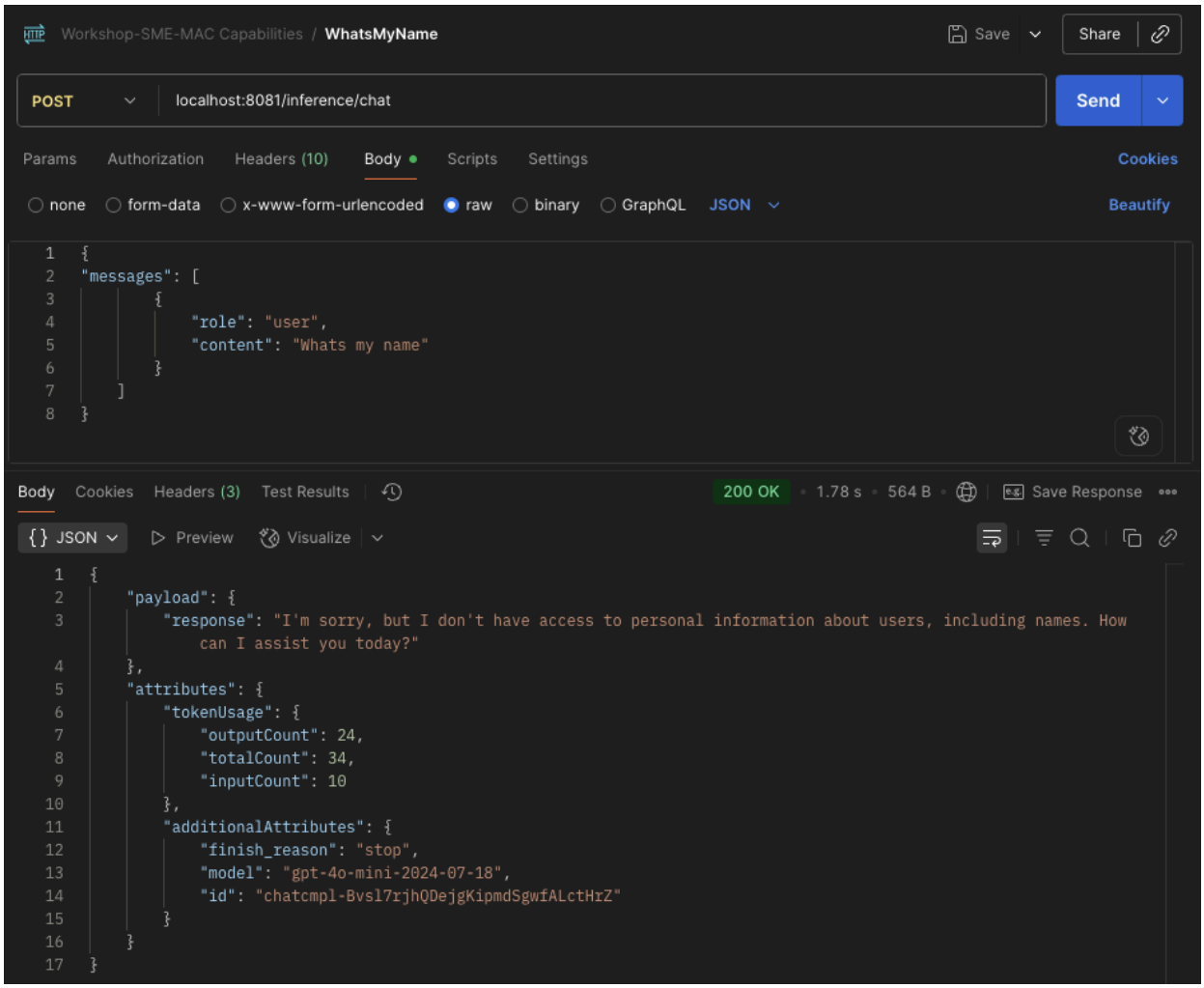
Test hands-on-mule-chat-history

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.

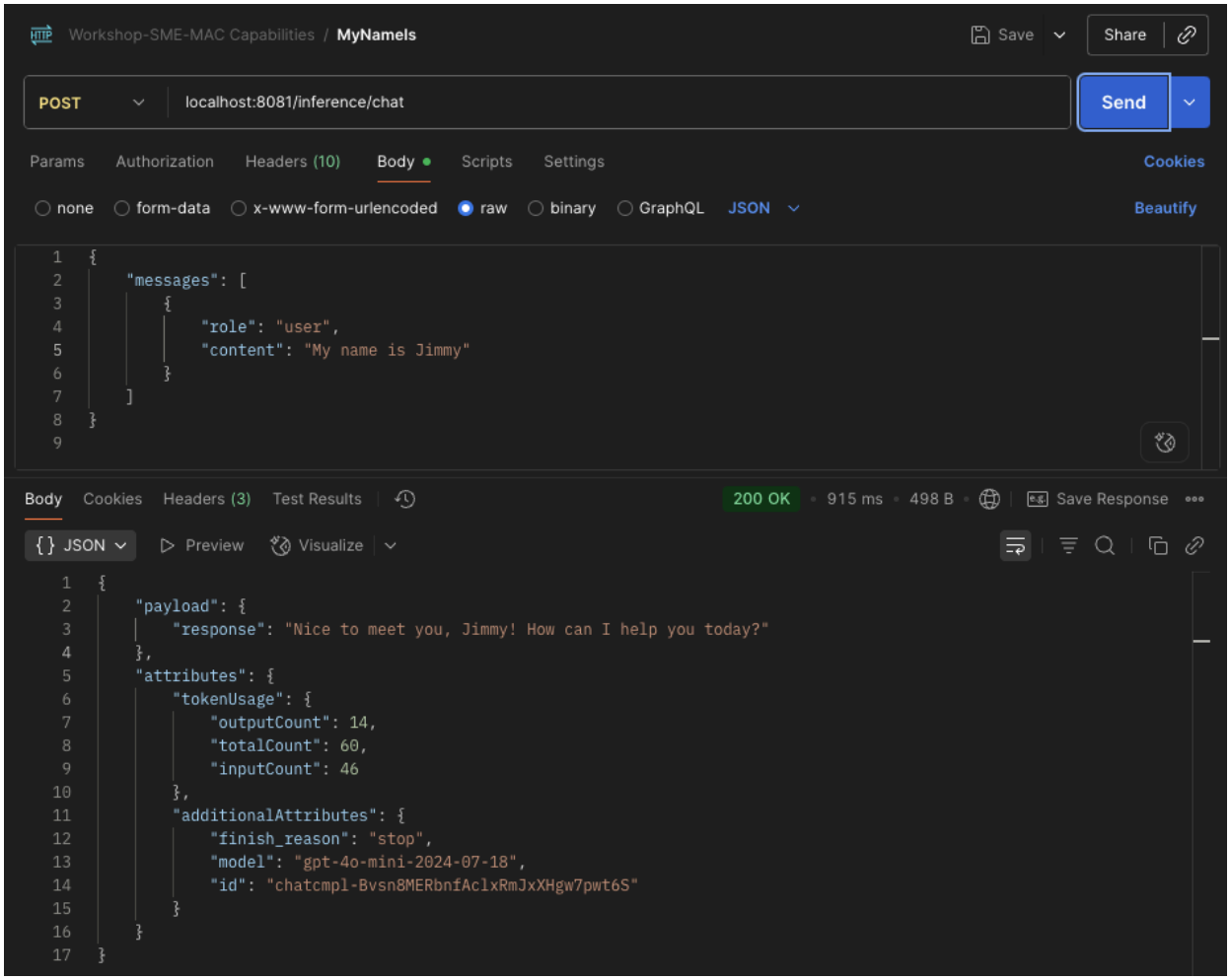
Note: Each time your test WhatsMyName and MyNameIs change the **userID** found in Headers to a different name and update the content.


POST
localhost:8081/inference/chat
Params
Authorization
Headers (10)
Body
Scripts
Settings
Headers
9 hidden
Key
Value
Desc
userID
Jimmy
Key
Value
Desc

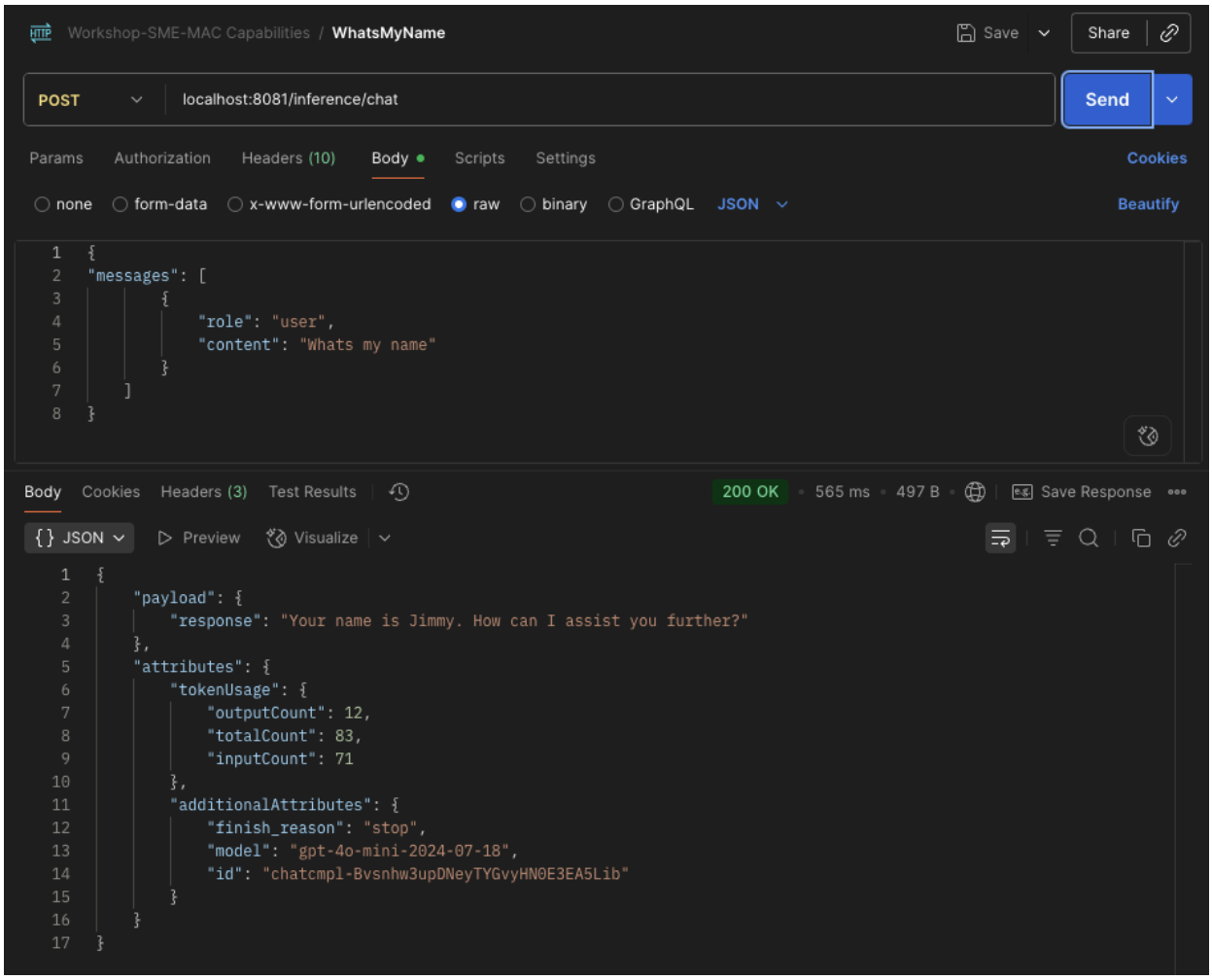

Workshop-SME-MAC Capabilities / MyNamels
POST
localhost:8081/inference/chat
Params
Authorization
Headers (10)
Body
Scripts
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
{
"messages": [
{
"role": "user",
"content": "My name is Jack"
}
]
}



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



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.



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.

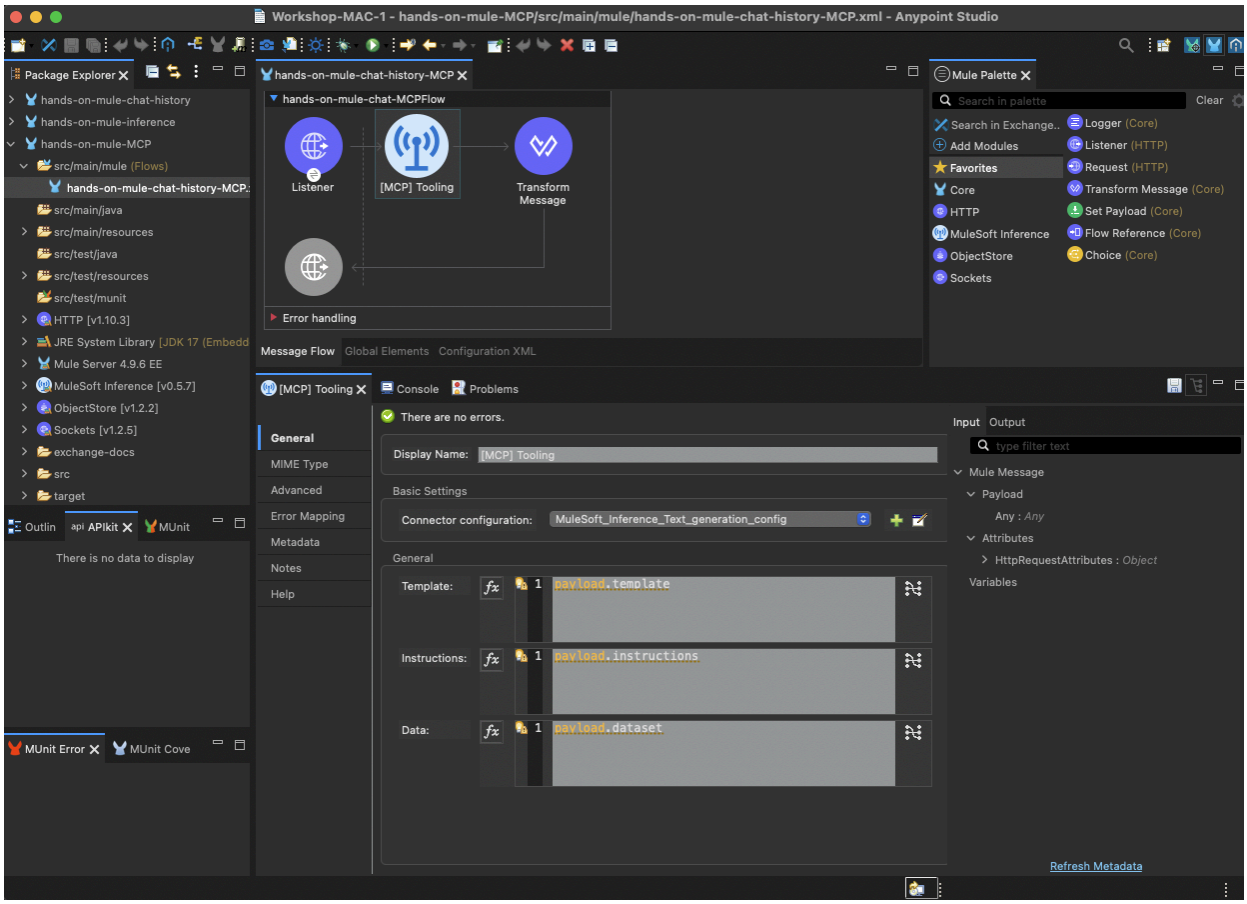
We are going to use the following 2 MCP Servers deployed on Cloudhub using the MuleSoft MCP Connector:

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

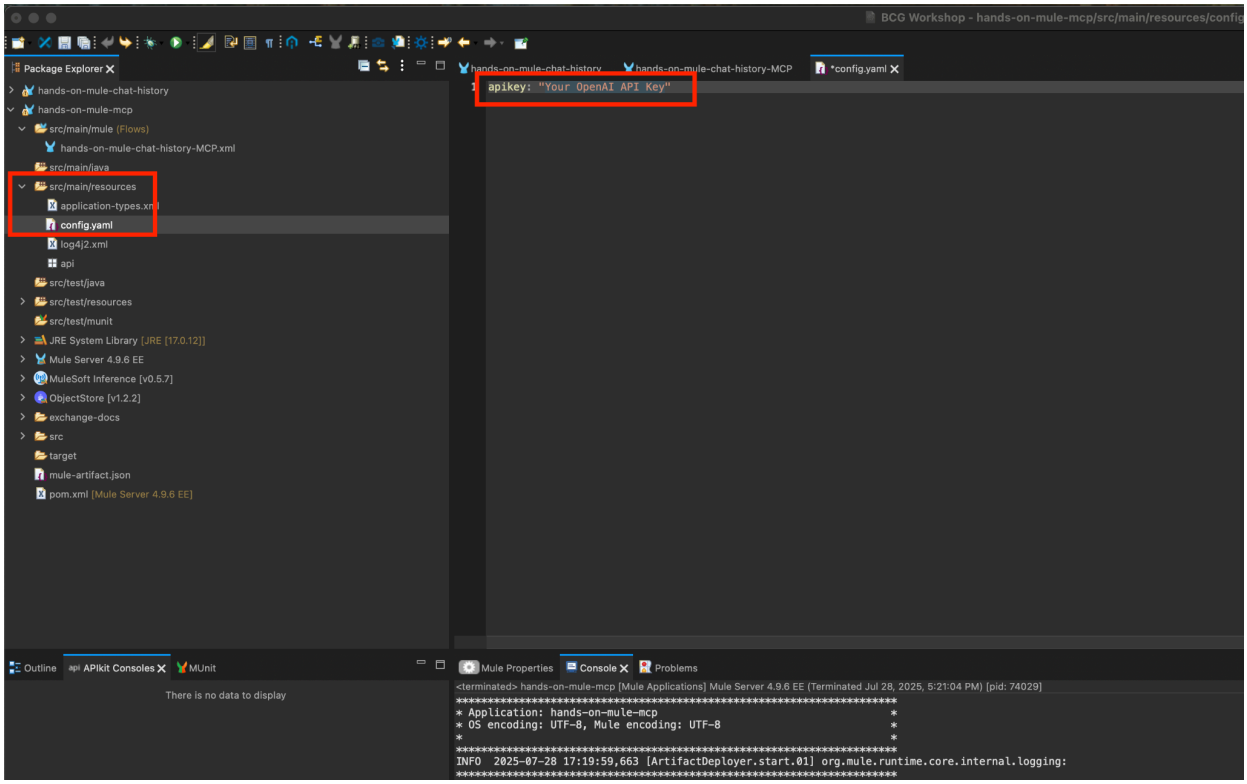
Import and Explore Project

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

2. Explore the project.



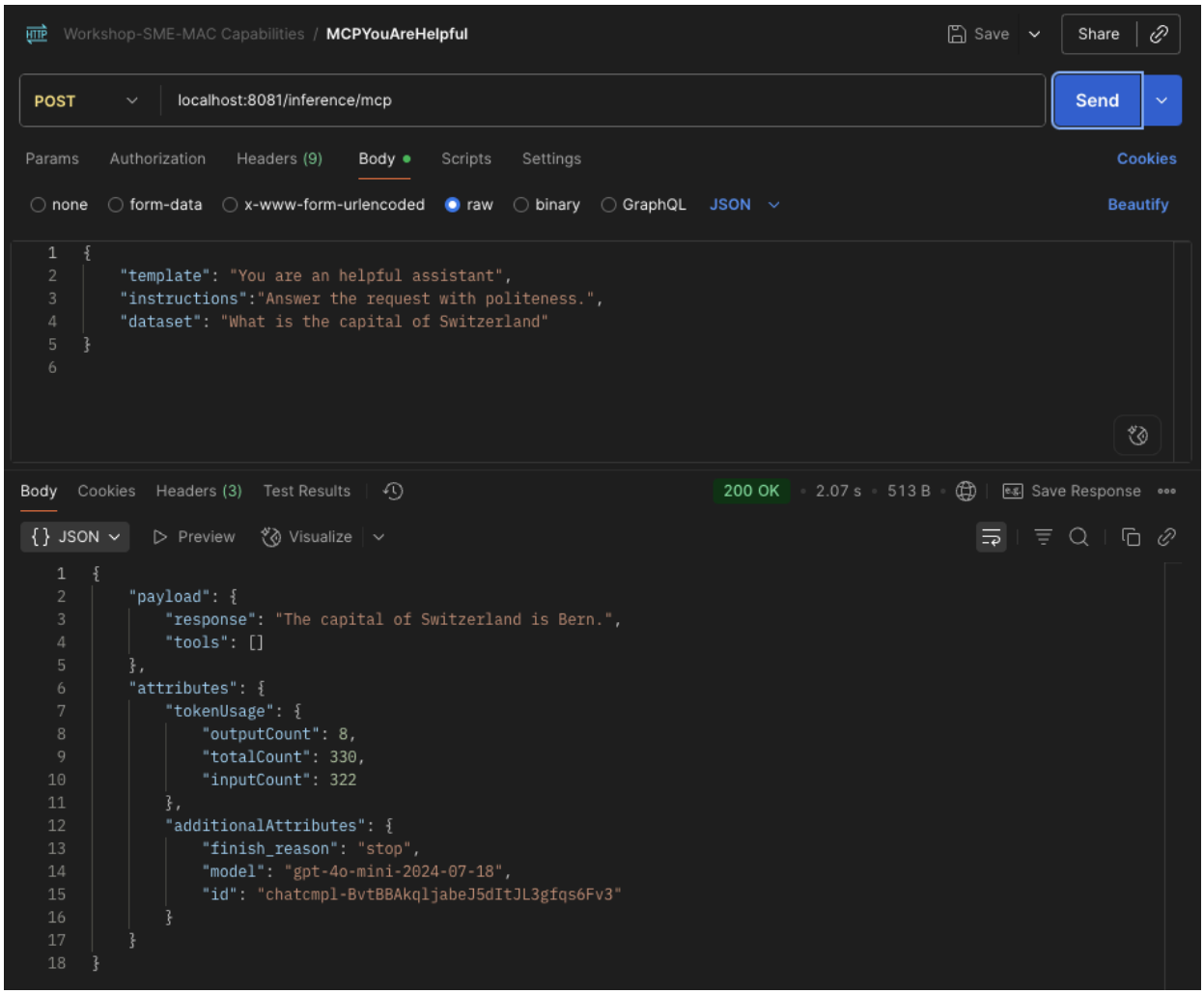
3. Replace the “Your OpenAI API Key” with your OpenAI API Key in the config.json. Put your API key between the quotes.



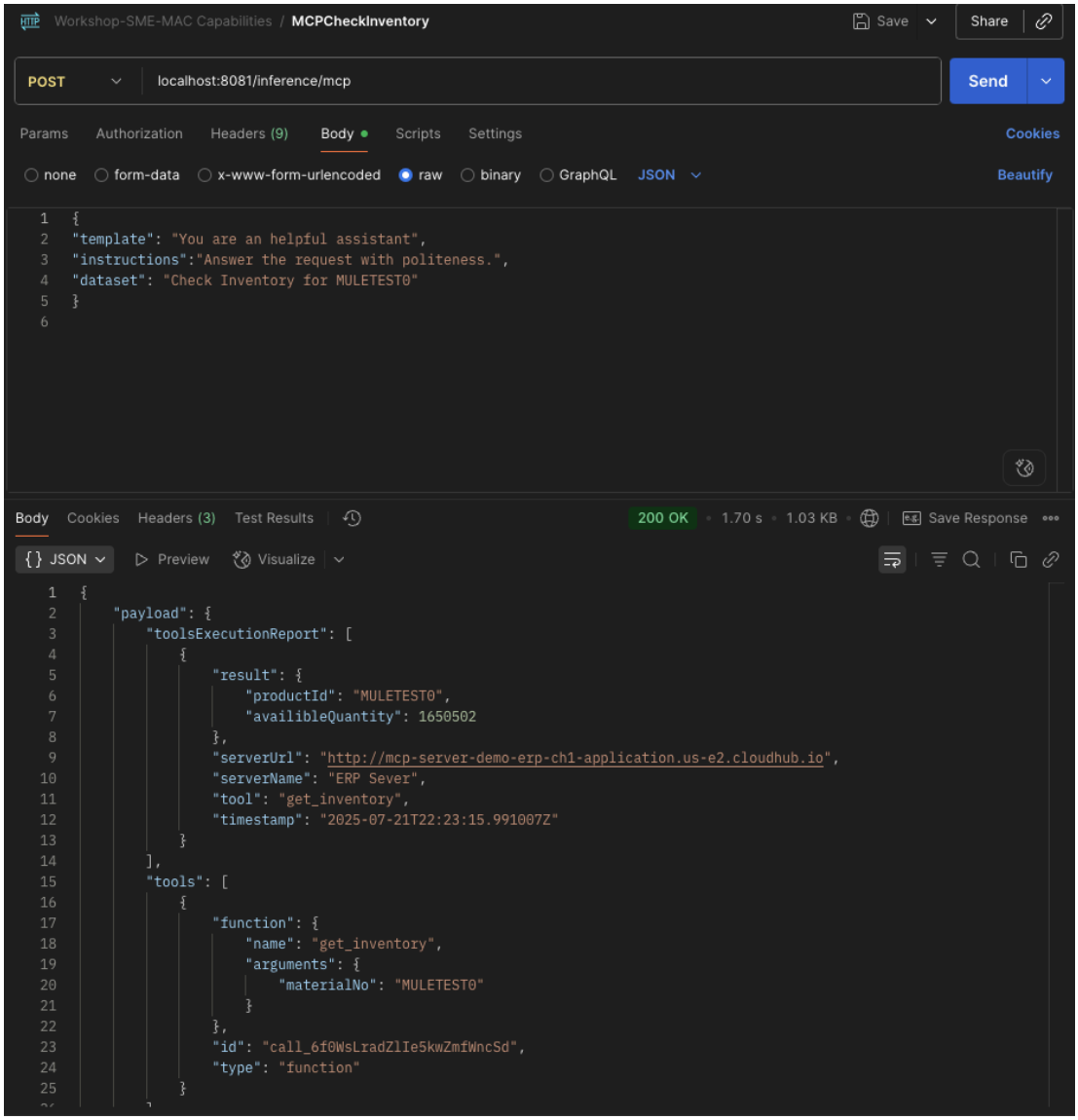
4. Run hands-on-mule-chat-history-MCP locally.

Test hands-on-mule-MCP

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



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



Exercise 4 - Implement A2A

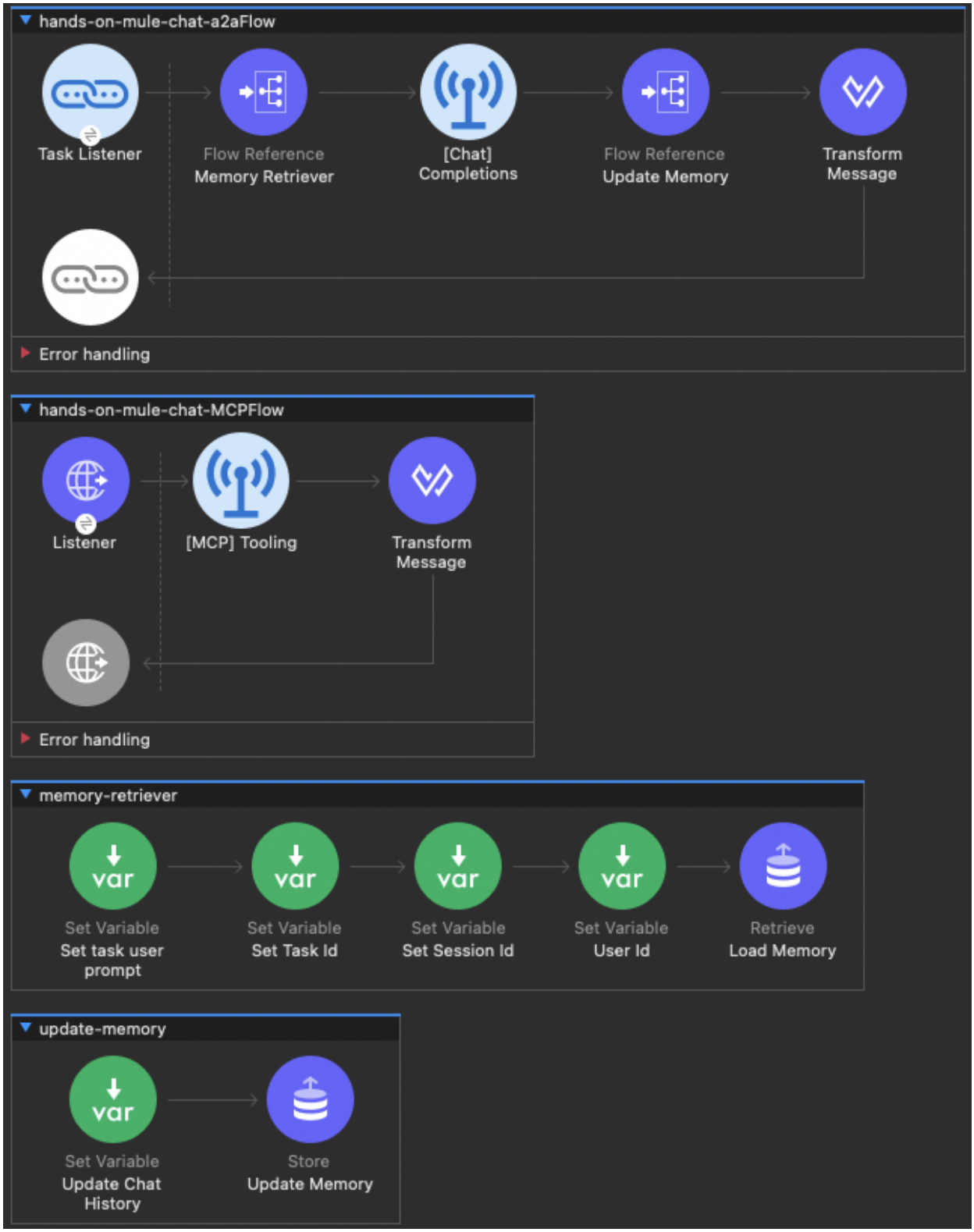
The MuleSoft A2A Connector is a specialized connector within the MuleSoft Anypoint Platform that facilitates secure and discoverable communication between AI agents, known as Agent-to-Agent (A2A) communication.

A2A Connector Can be added through Anypoint Exchange

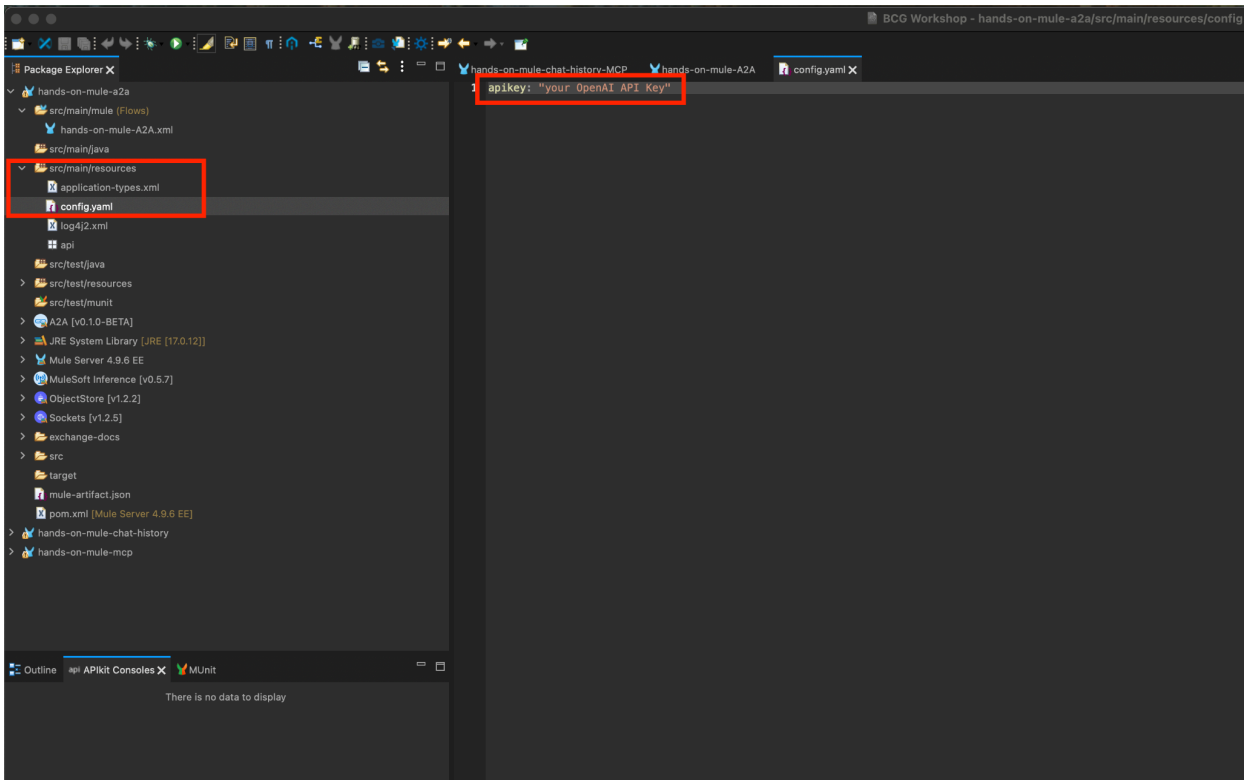
Import and Explore Project

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

2. Explore the project.



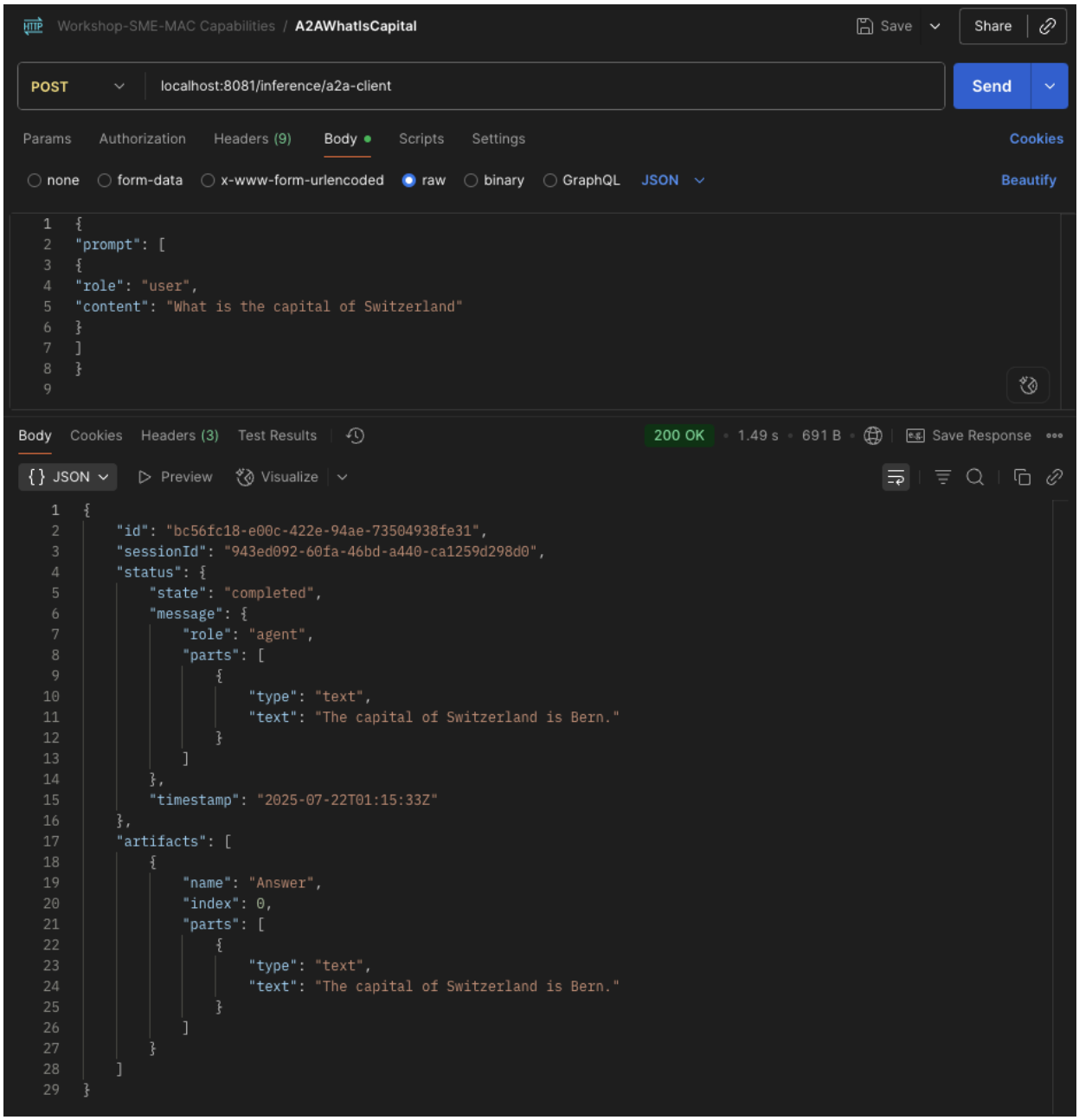
3. 3. Replace the “Your OpenAI API Key” with your OpenAI API Key in the config.json. Put your API key between the quotes.



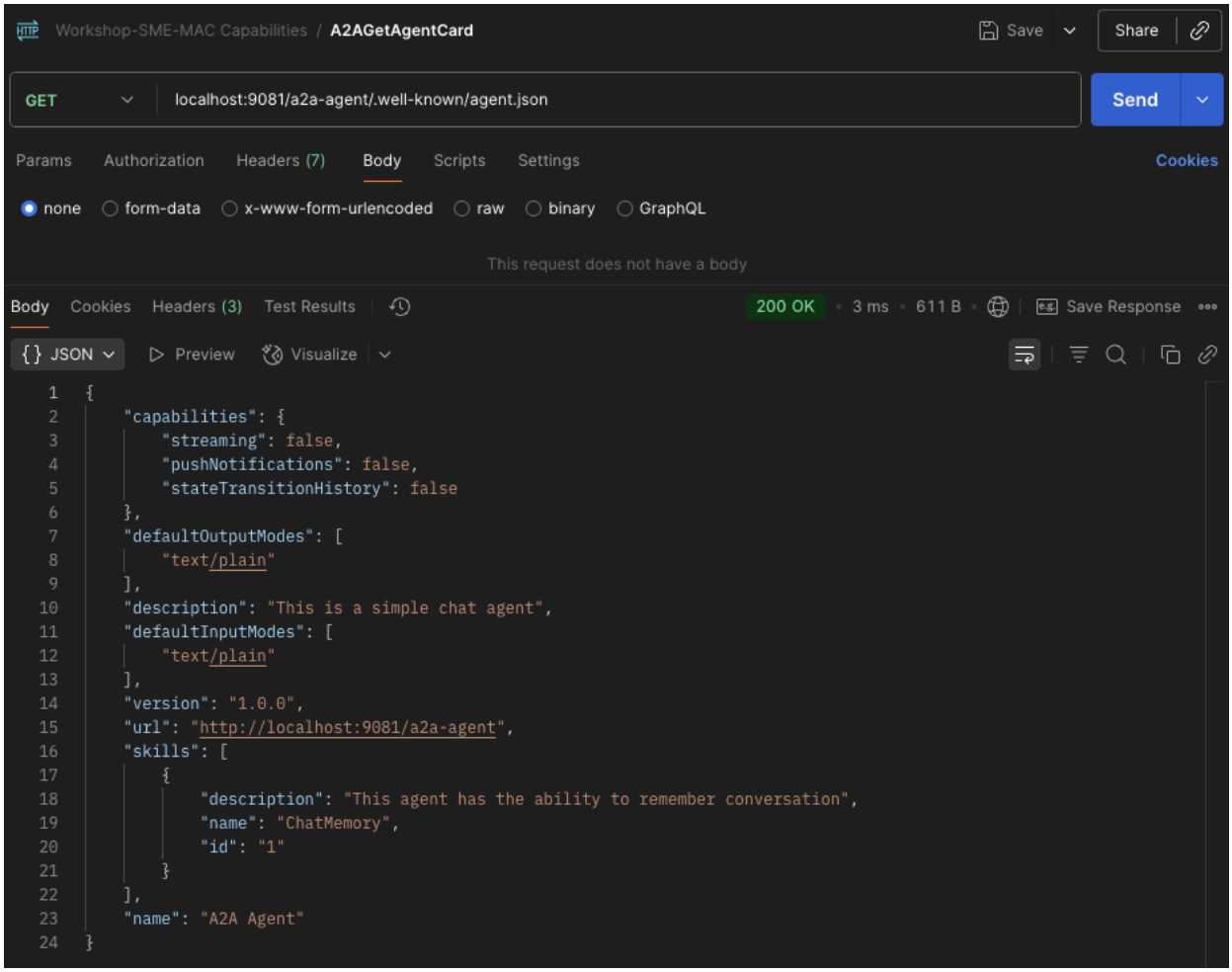
4. Run hands-on-mule-chat-history-A2A locally.

Test hands-on-mule-A2A

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



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



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,

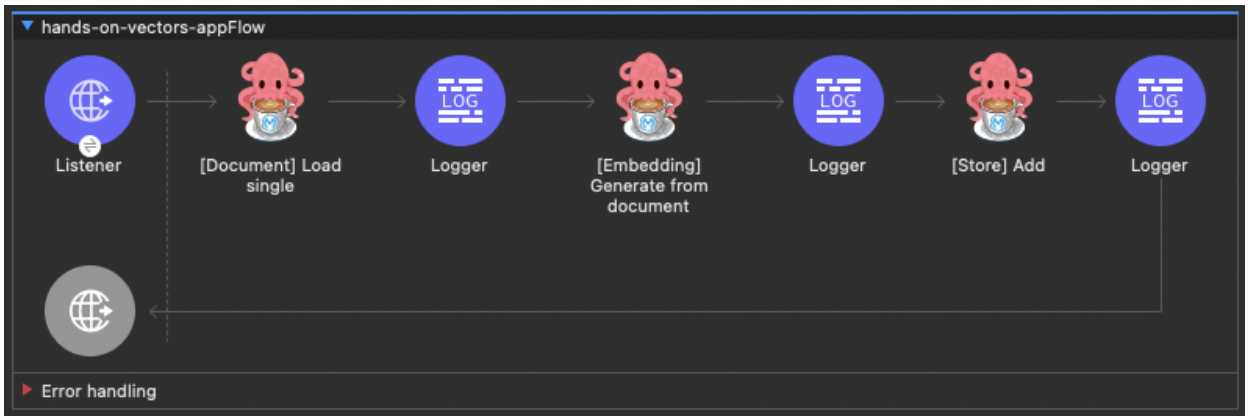
[Embedding] Generate from document operation, allowing you to create searchable embeddings from your documents,

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

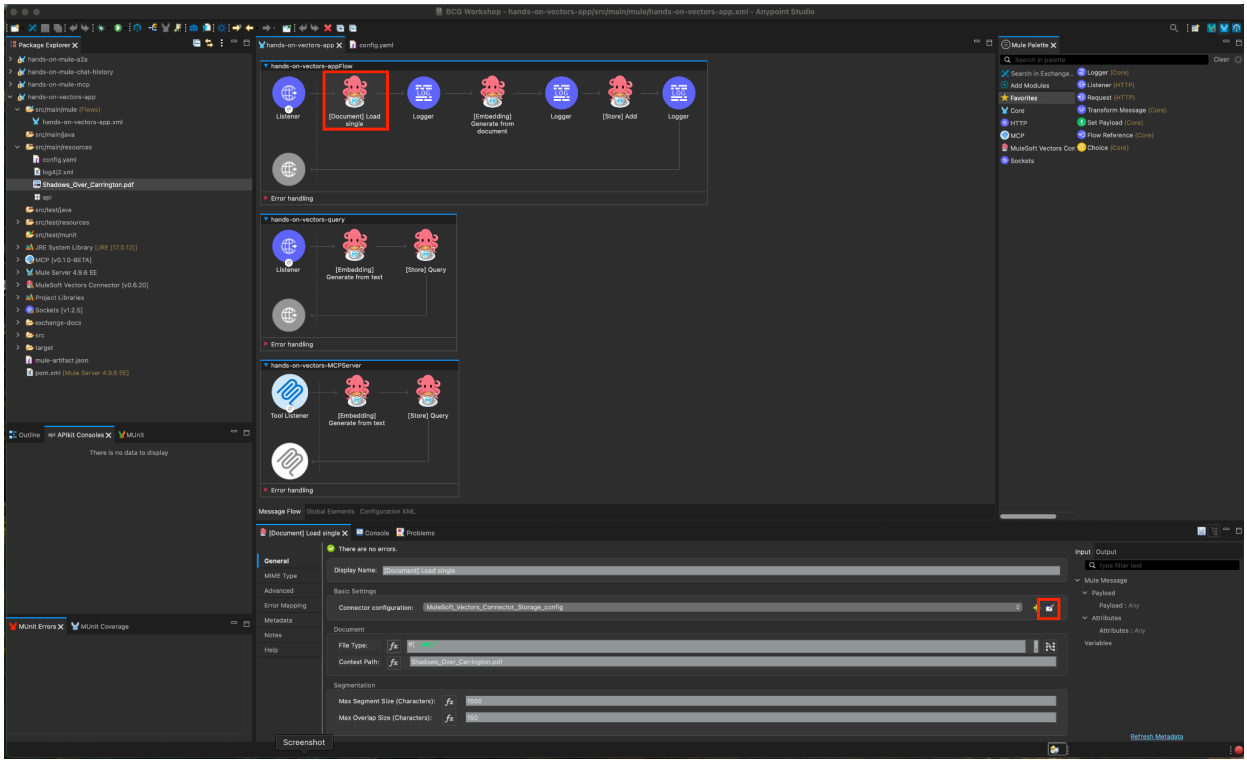
Import and Explore Project

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

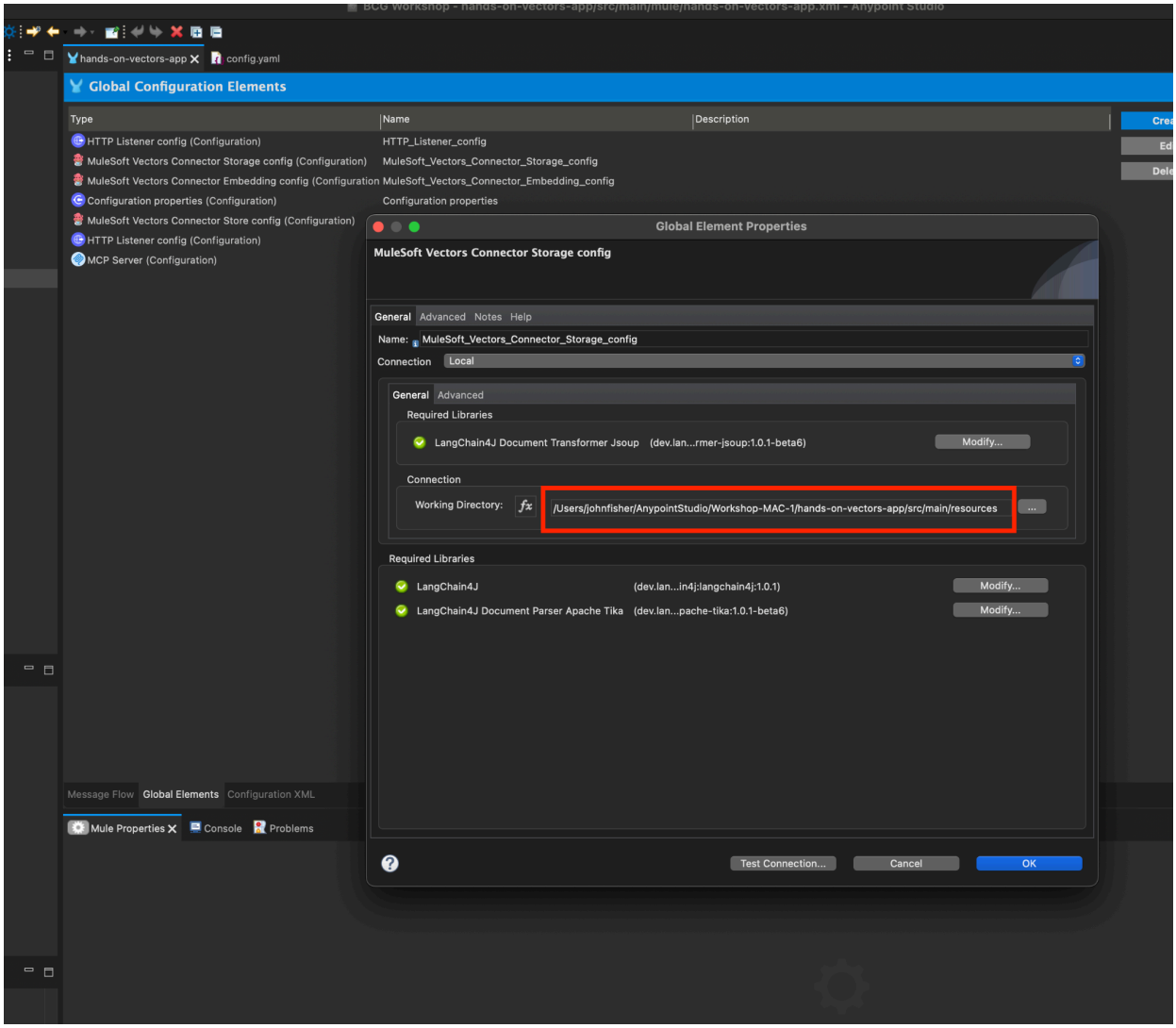
2. Explore the project.



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

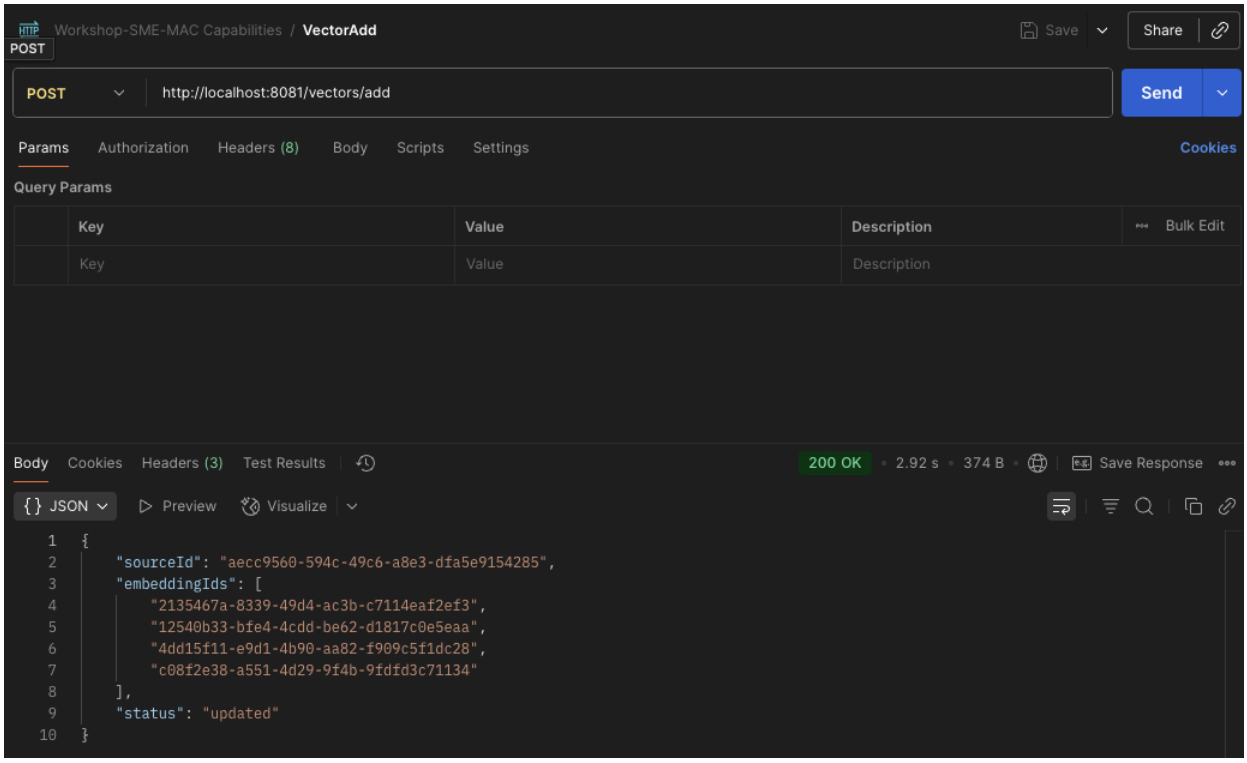


4. Modify the “Working Directory” and replace it with your own path the “Shadows_Over_Carrington.pdf”. The file is located in the Mule App under the ./src/main/resources folder. For example:
Mac: /Users/bpittman/AnypointStudio/BCG Workshop/hands-on-vectors-app/src/main/resources
Windows: C:\Users\bpittman\AnypointStudio\BCG Workshop\hands-on-vectors-app\src\main\resources



Test hands-on-mule-vectors-app

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



Exercise 6 - Query Vector Store

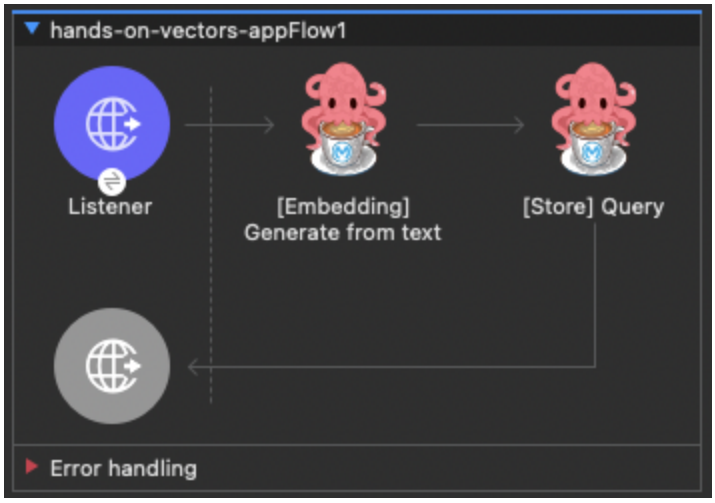
In this exercise we are going to explore:

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.

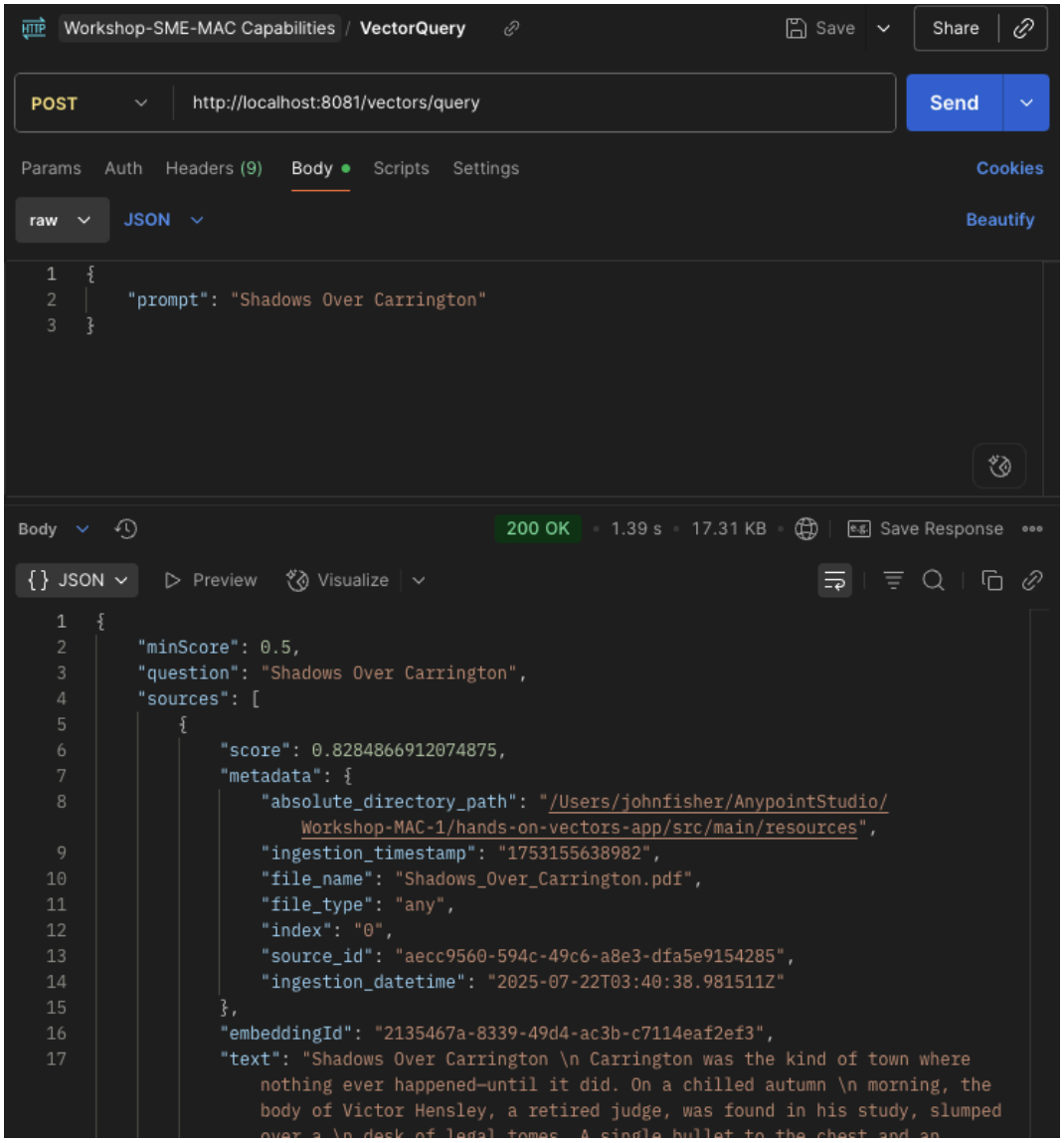
1. Import **hands-on-vectors-app** into your workspace (if you havent done so already).

2. Explore the project.



Test hands-on-mule-vectors-app

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



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:

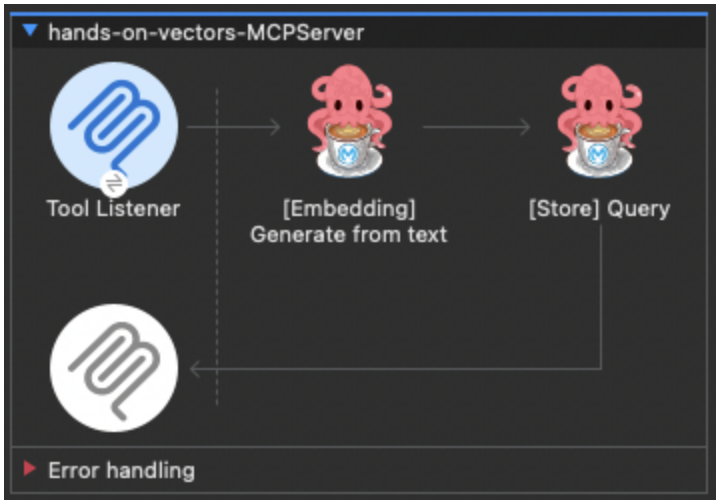
- Enhances AI precision by supplying AI with relevant information, leading to more accurate responses.
- Empowers AI to manage intricate tasks that require access to external data or services.
- Promotes interoperability between AI models and external systems, simplifying integration into workflows.

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.

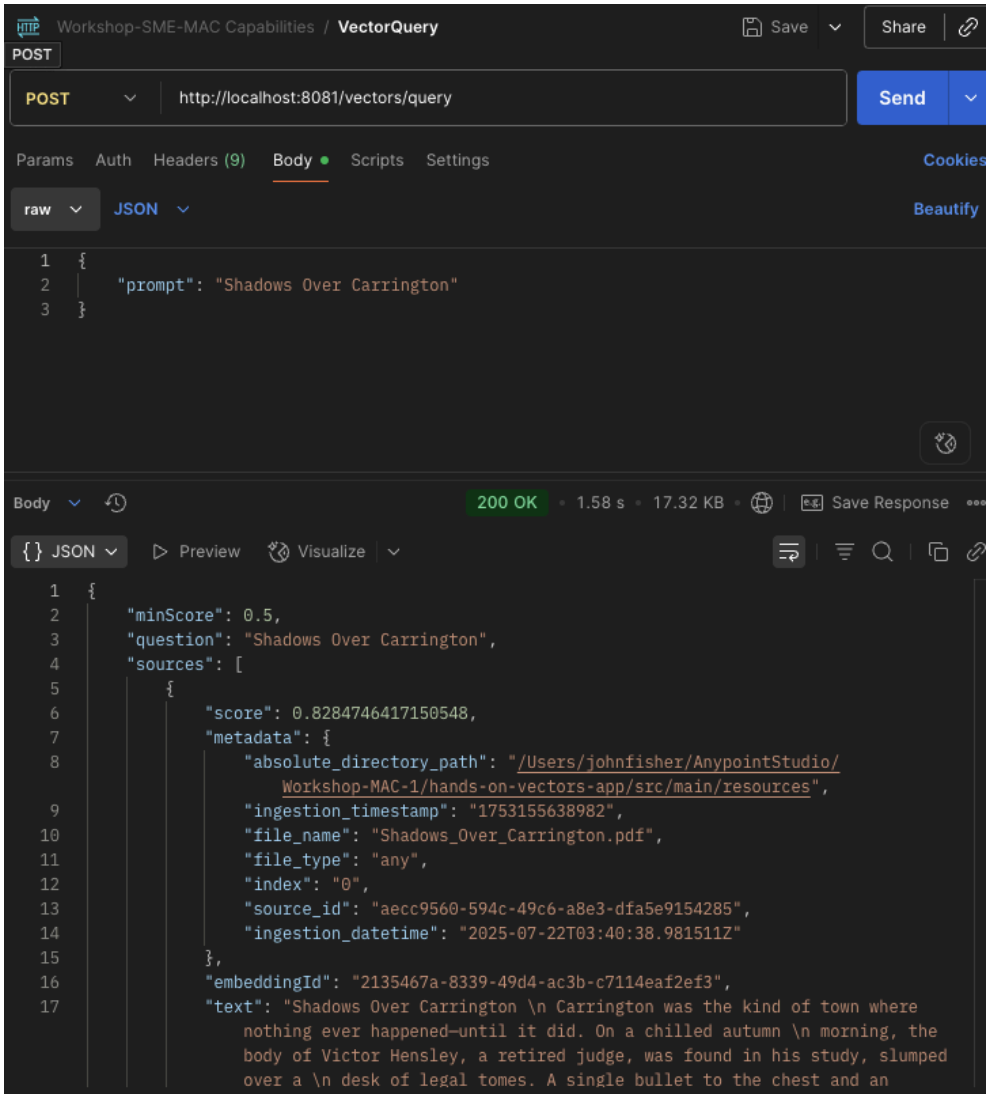
1. Import **hands-on-vectors-app** into your workspace (if you haven't done so already).

2. Explore the project.

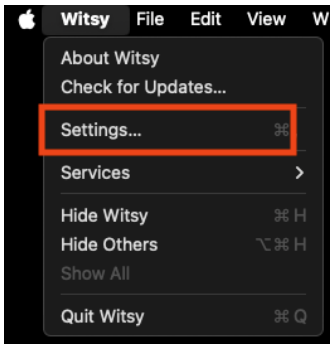


Test hands-on-mule-vectors-app

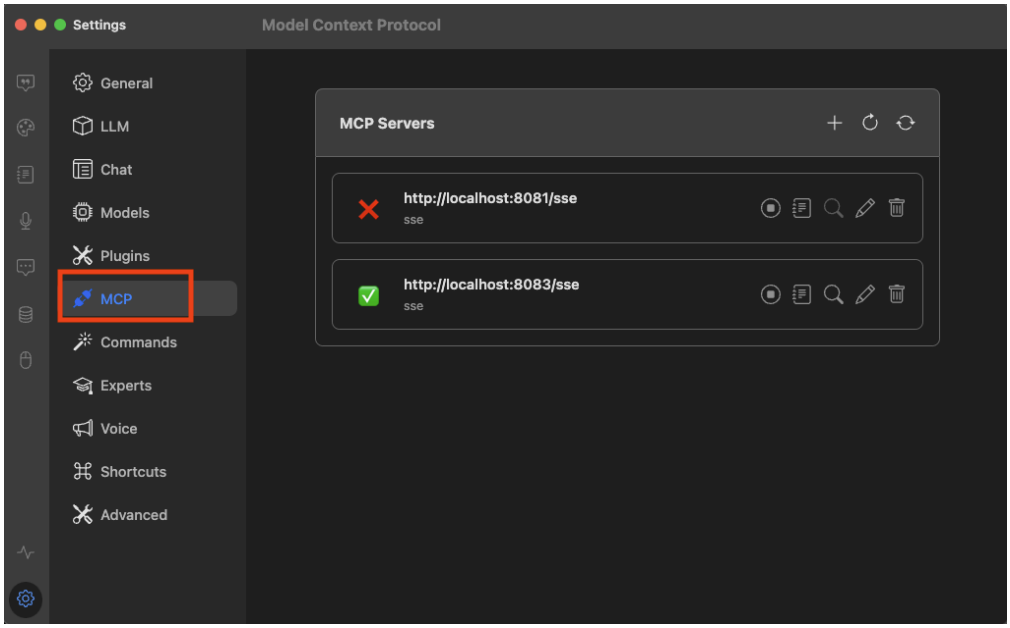
1. Open Postman and select **VectorQuery** request and click **Send.** We want to check that the Vector store is working.



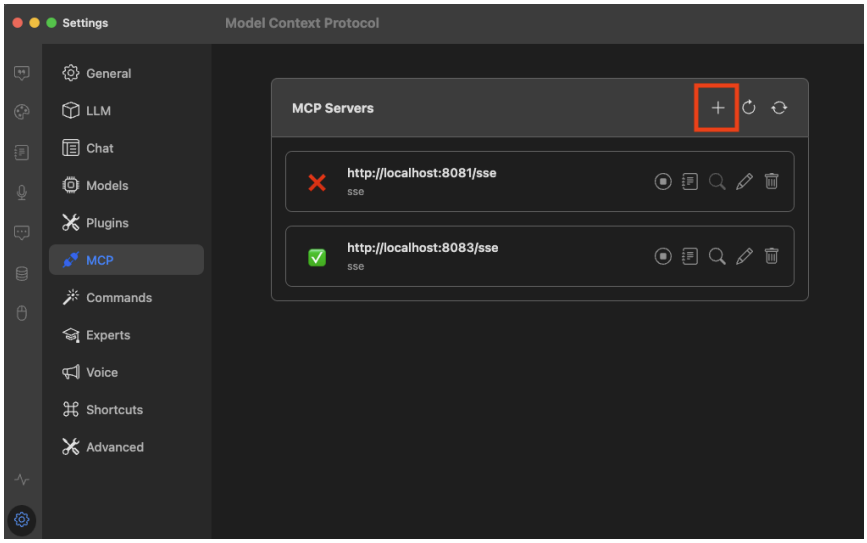
2. Open Witsy and from the Witsy Menu select **settings**.



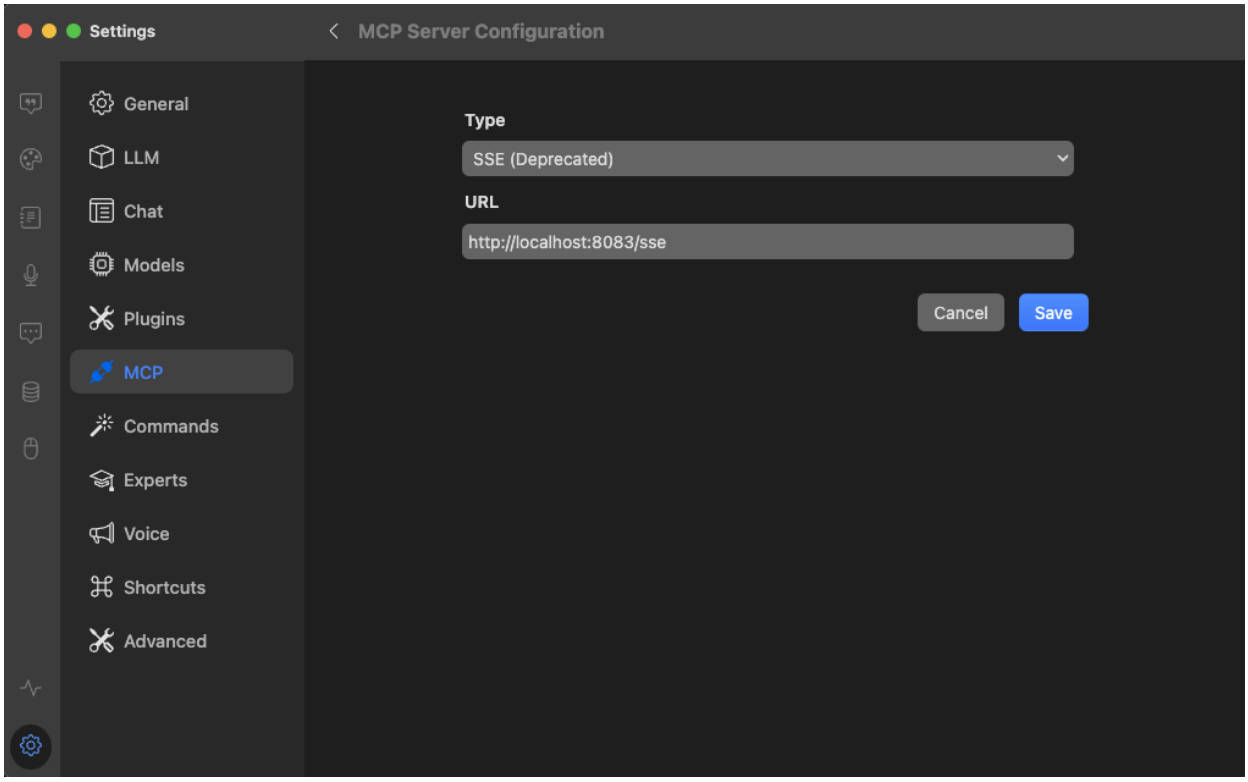
3. Click on MCP to open the MCP Servers configuration.



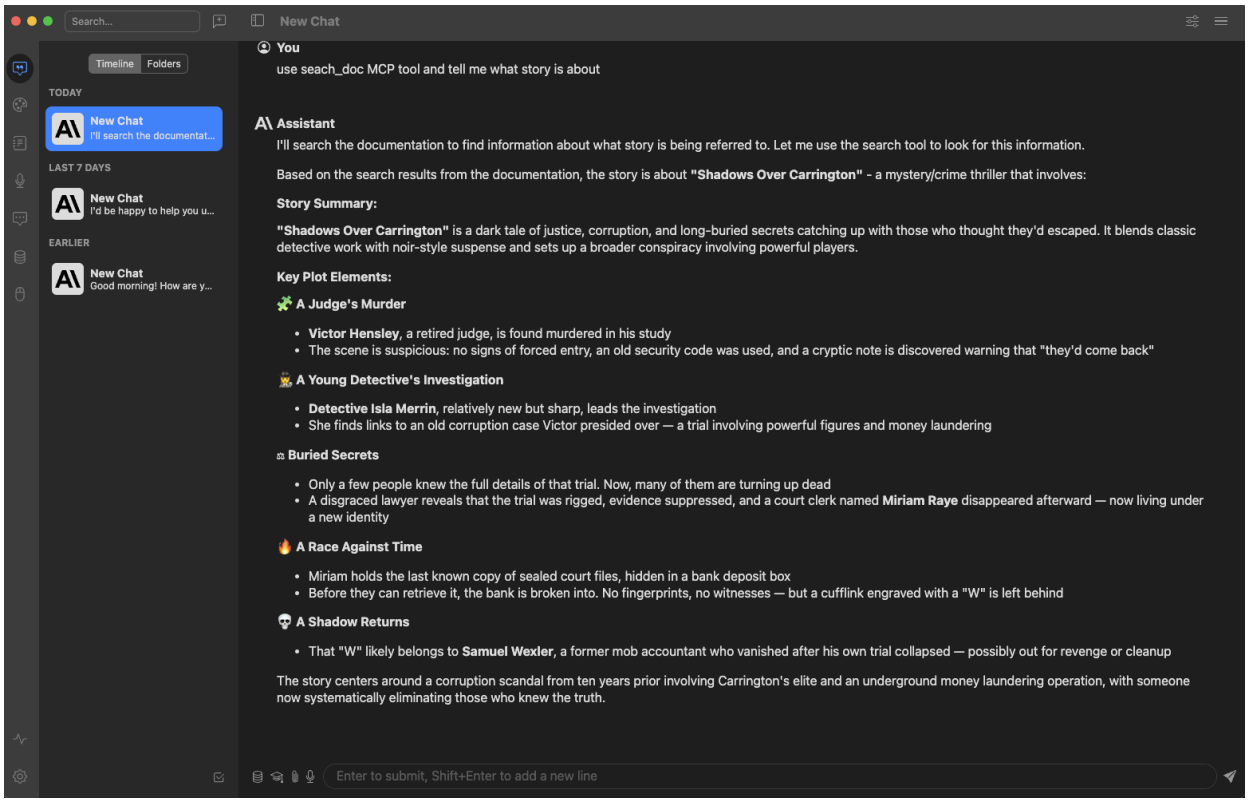
4. Click on the “+” icon to add a new MCP Server.



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:

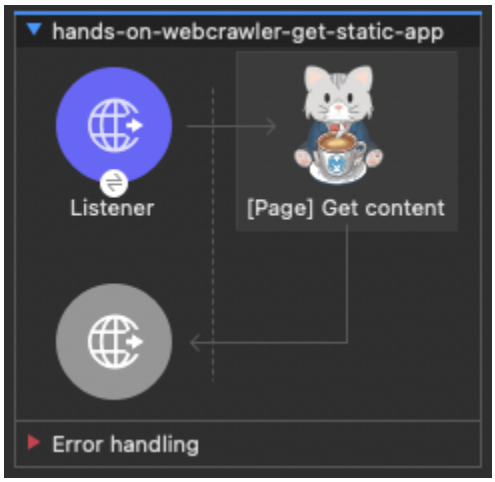
The MuleSoft AI Chain (MAC) Connector's:

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

Import and Explore Project

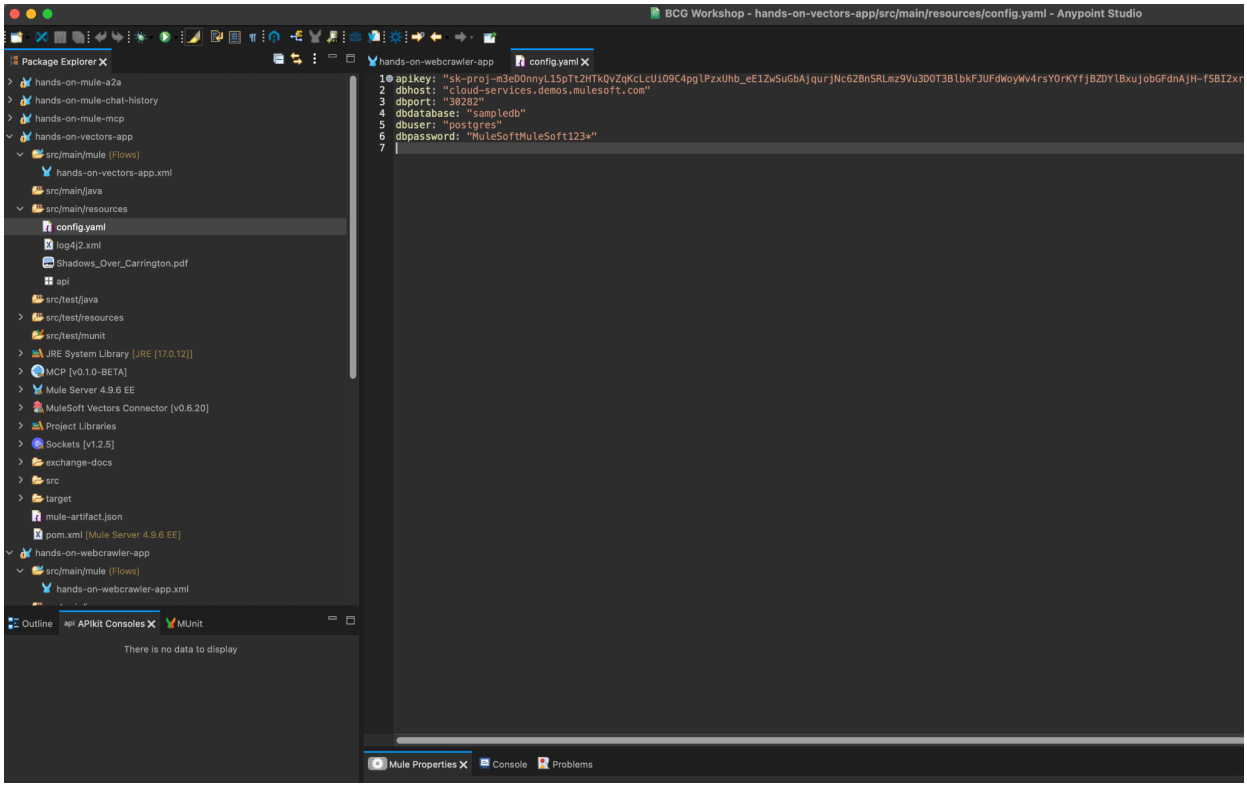
1. Import **hands-on-webcrawler-app** into your workspace.

2. Explore the project.

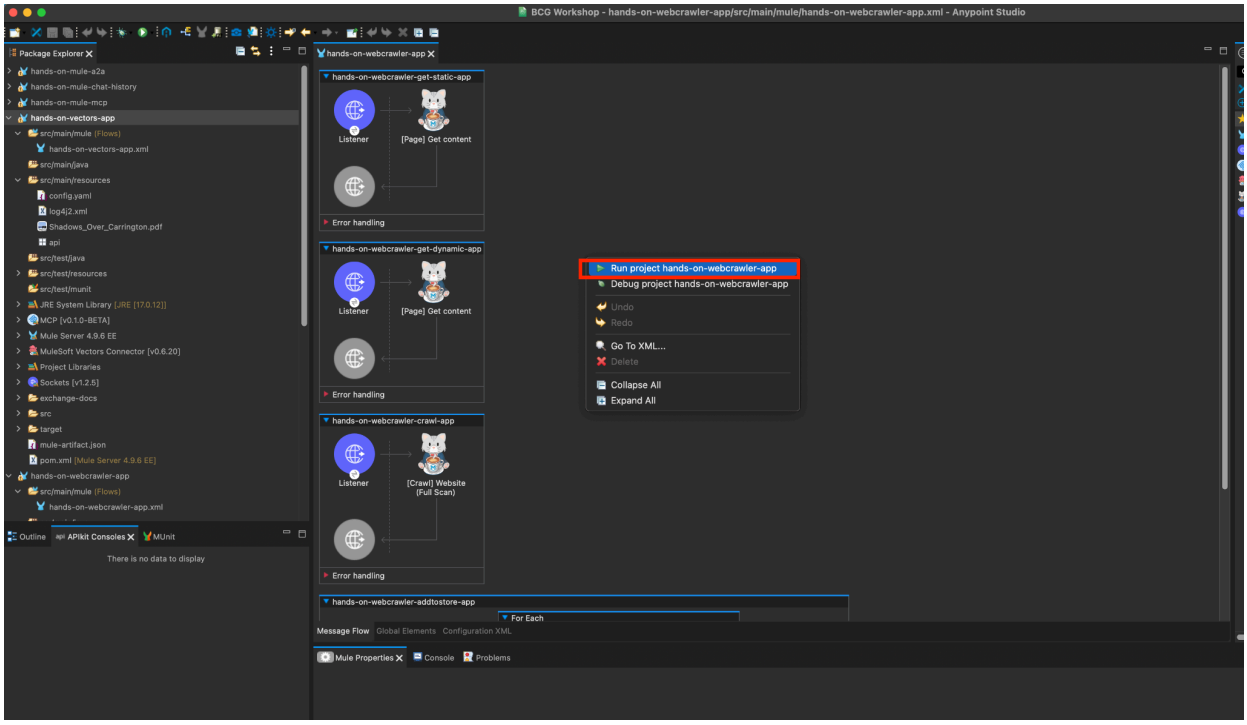


3. Open the config.yaml and update the values with values provided by the teacher.

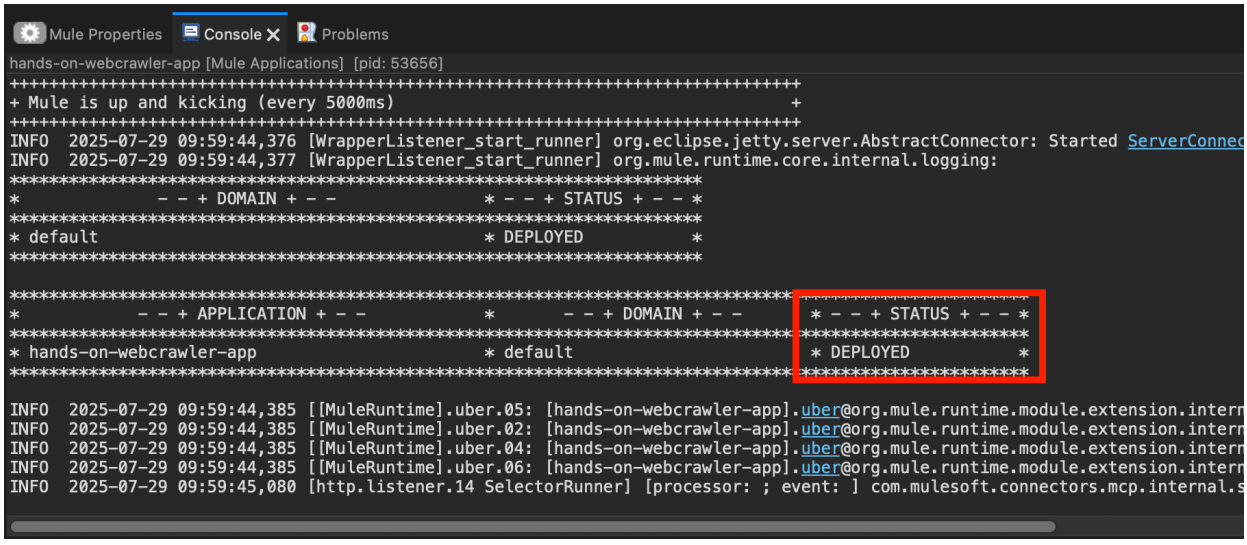
Save the file.



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

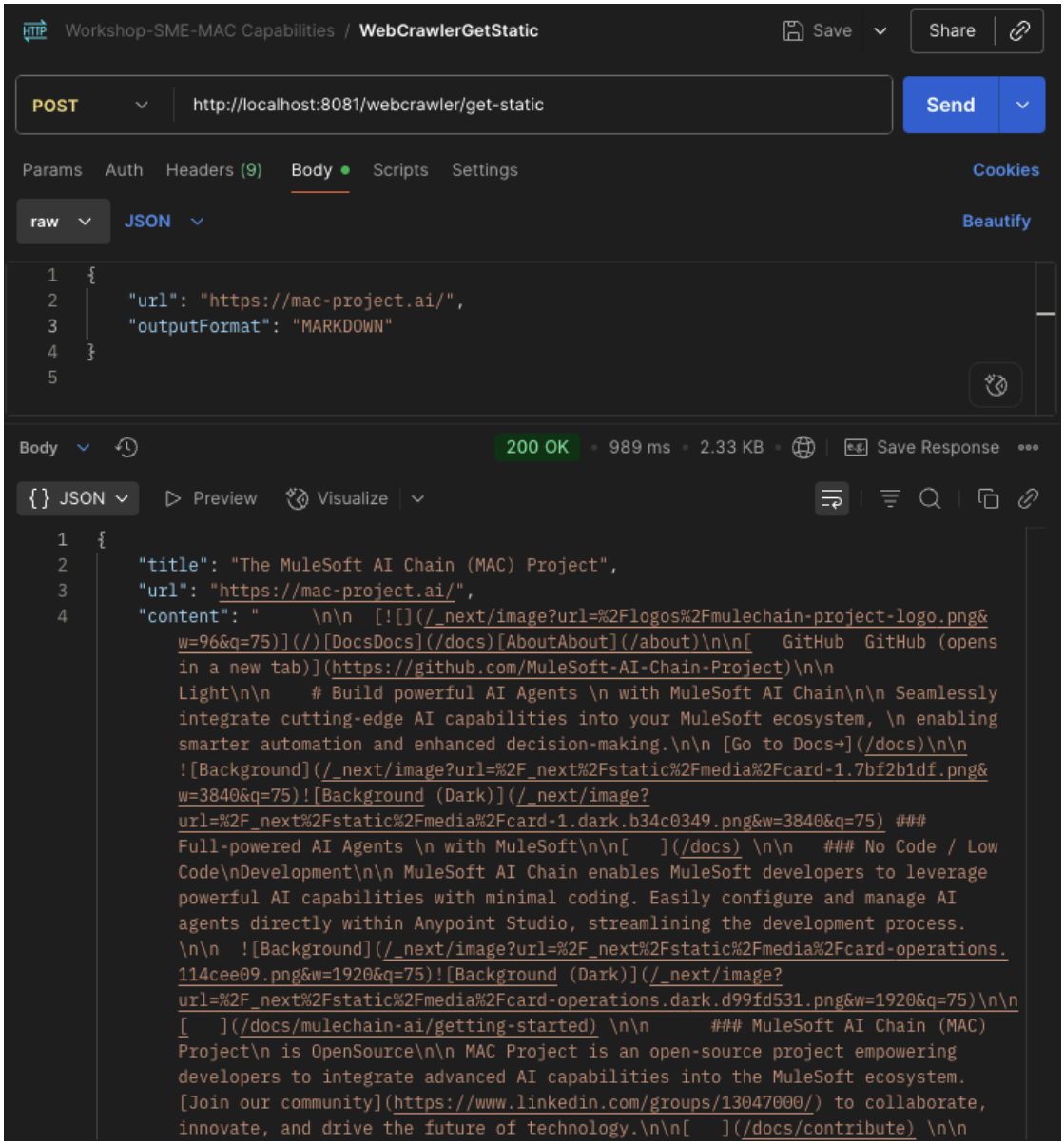


5. You should see it start successfully in the “Console” Window



Test hands-on-mule-webcrawler-app

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



Exercise 9 - Get Dynamic Page

In this exercise we are going to explore:

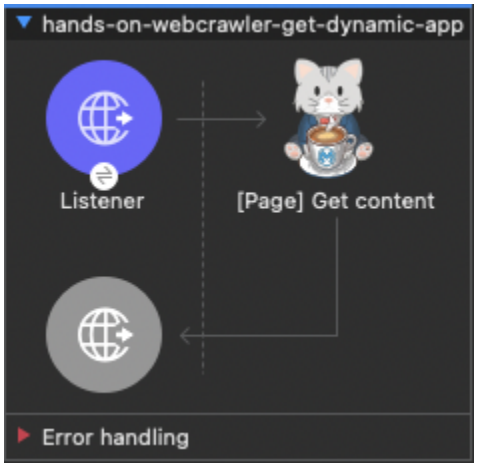
The MuleSoft AI Chain (MAC) Connector's:

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

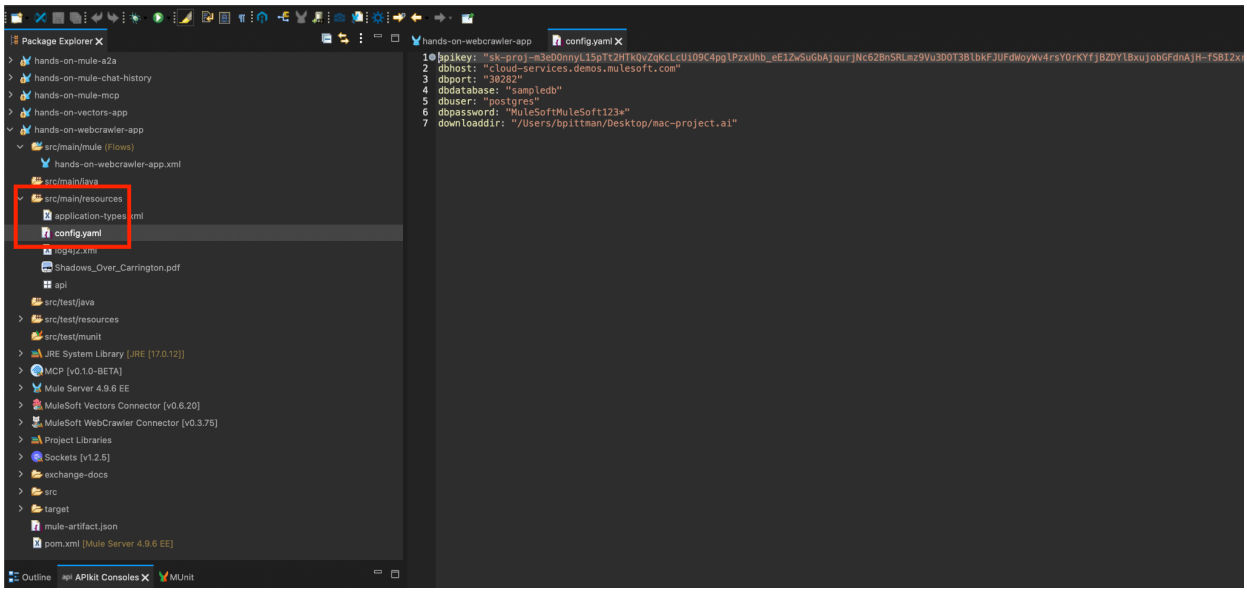
Import and Explore Project

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

2. Explore the project.

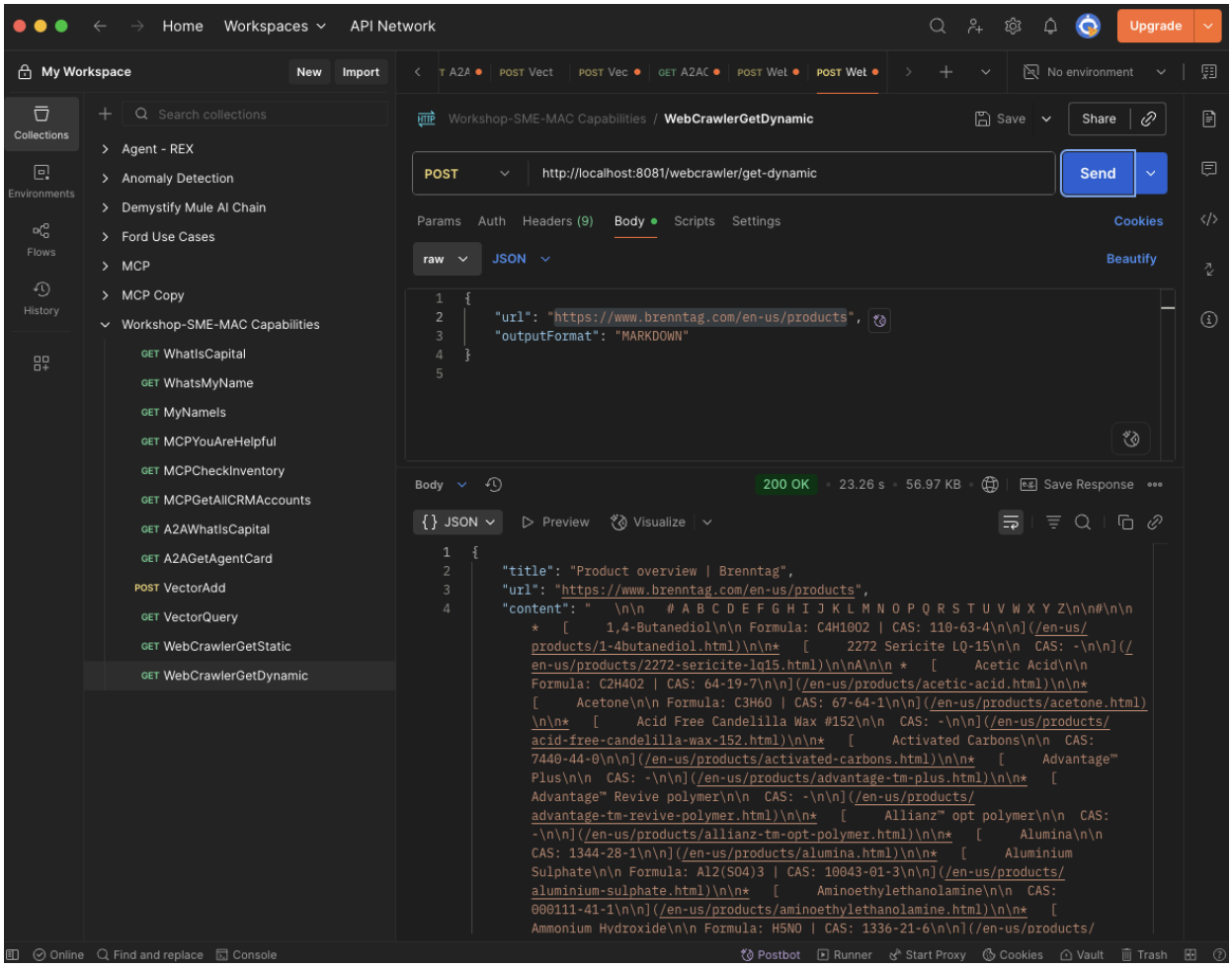


3. Open the config.yaml and update the values with values provided by the teacher.
Save the file.



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:

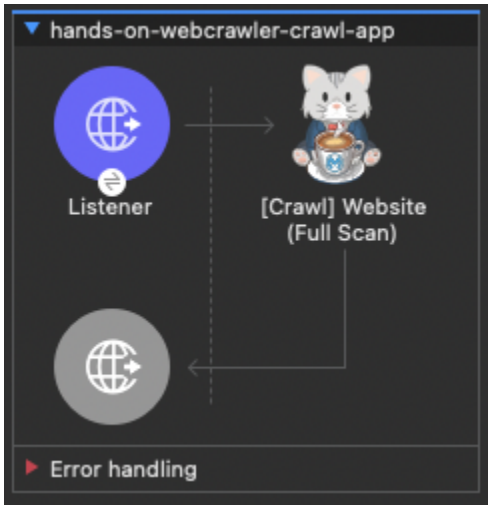
[Crawl] Website (Full Scan): operation allows you to easily crawl for website content, at a specified depth. This operation allows you to additionally:

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

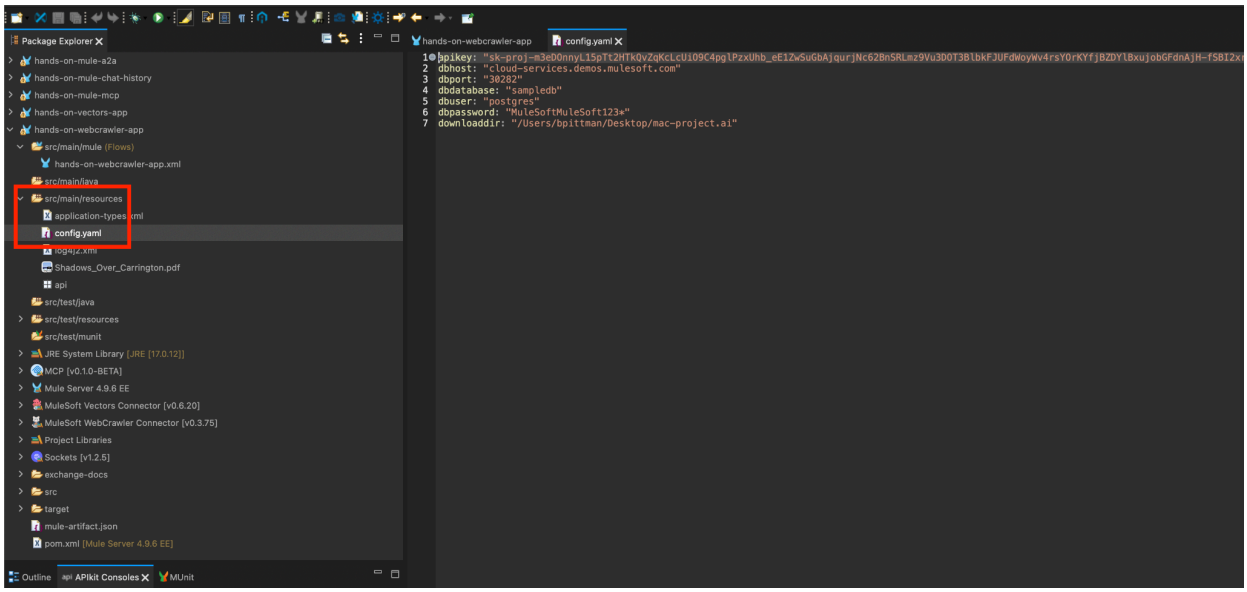
Import and Explore Project

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

2. Explore the project.

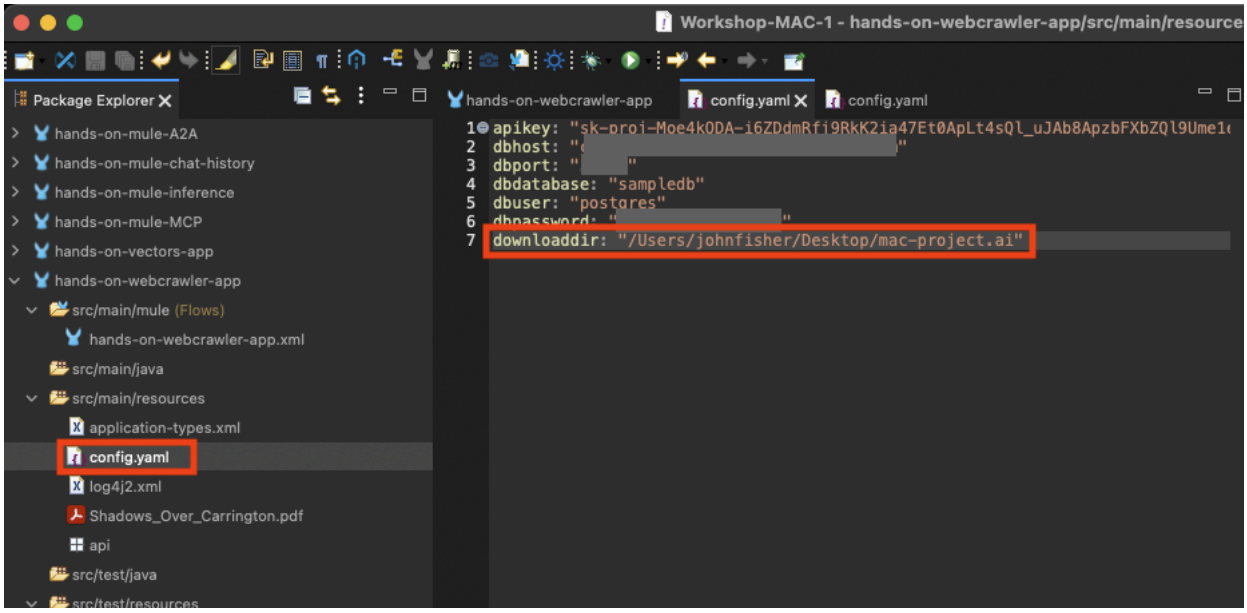


3. Open the config.yaml and update the values with values provided by the teacher.
Save the file.



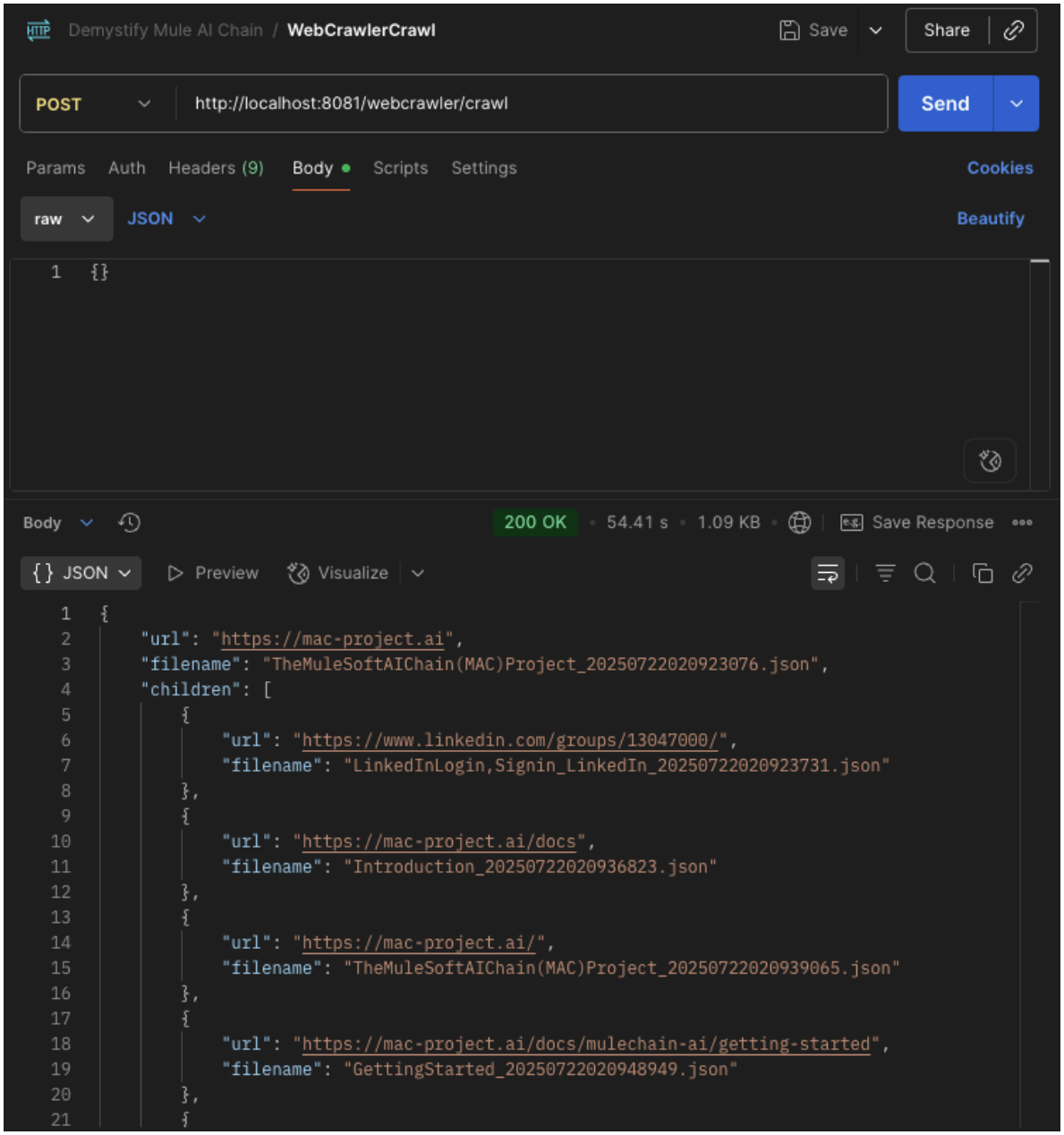
4. Create a folder on your desktop and name it mac-project.ai. Hands-on-webcrawler-app will download the web site content to this directory.

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

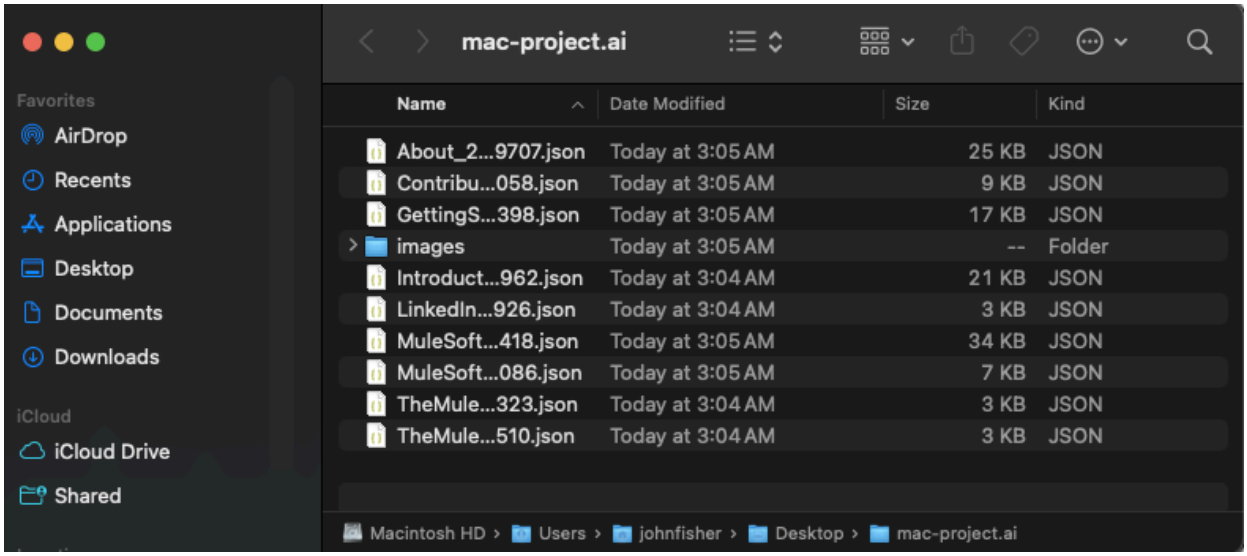


Test hands-on-mule-webcrawler-app

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



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

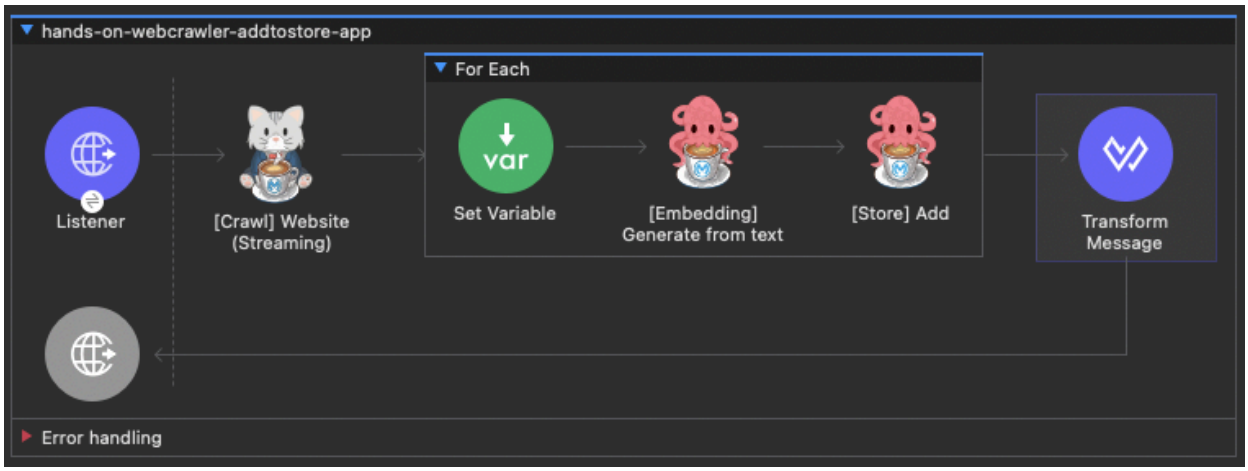


Exercise 11 - Crawl a web page and Add to Store

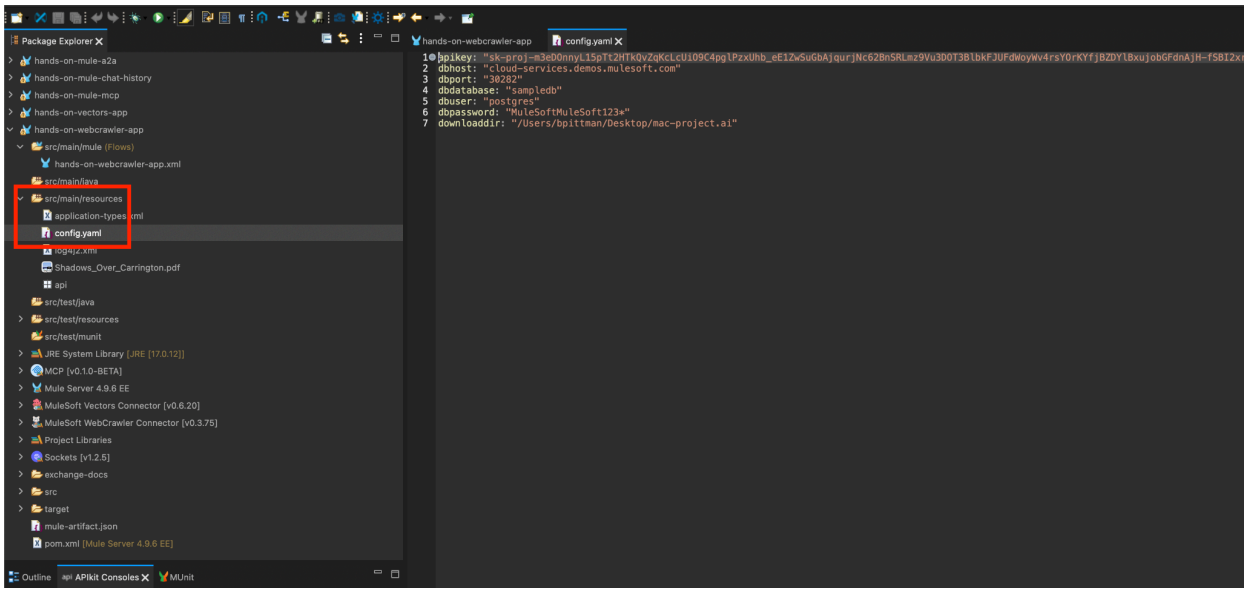
Import and Explore Project

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

2. Explore the project.

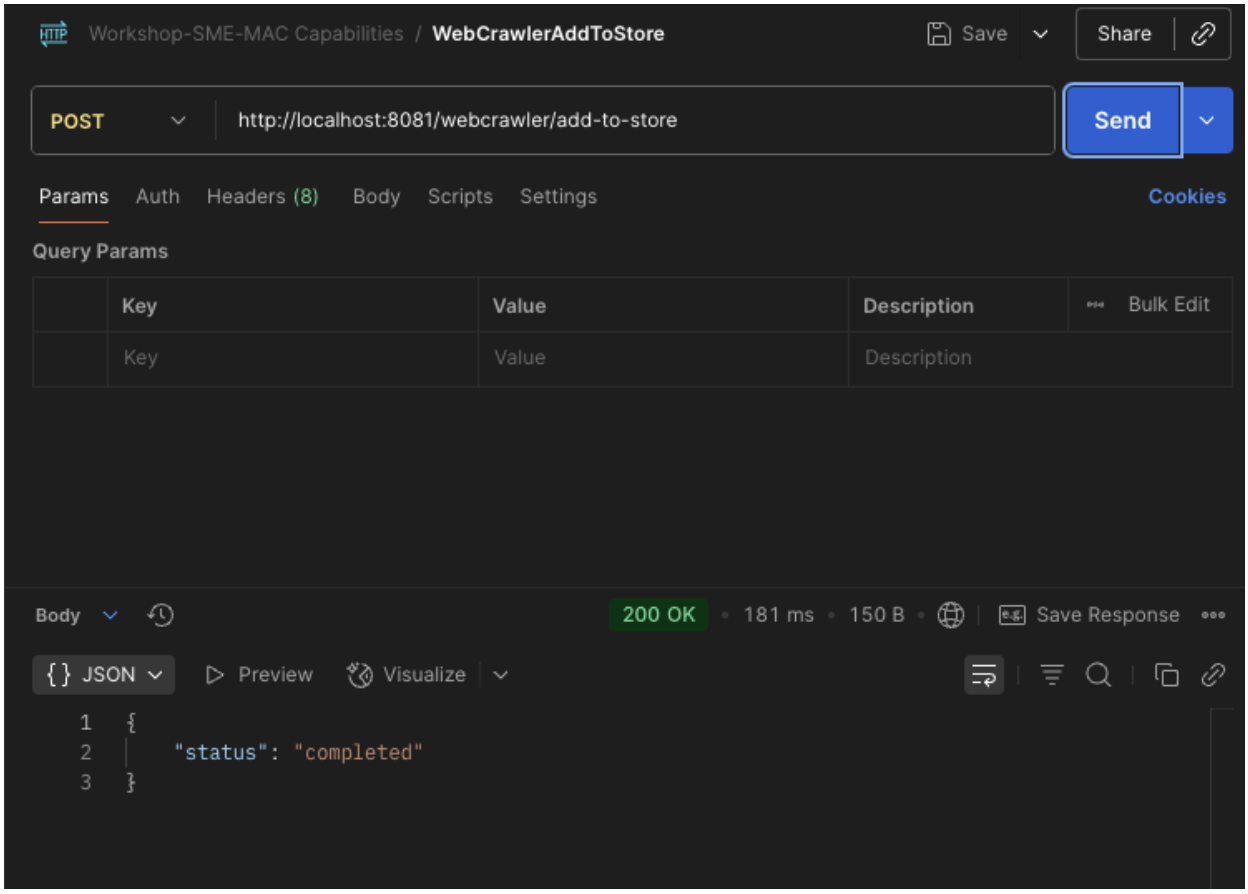


3. Open the config.yaml and update the values with values provided by the teacher.
Save the file.

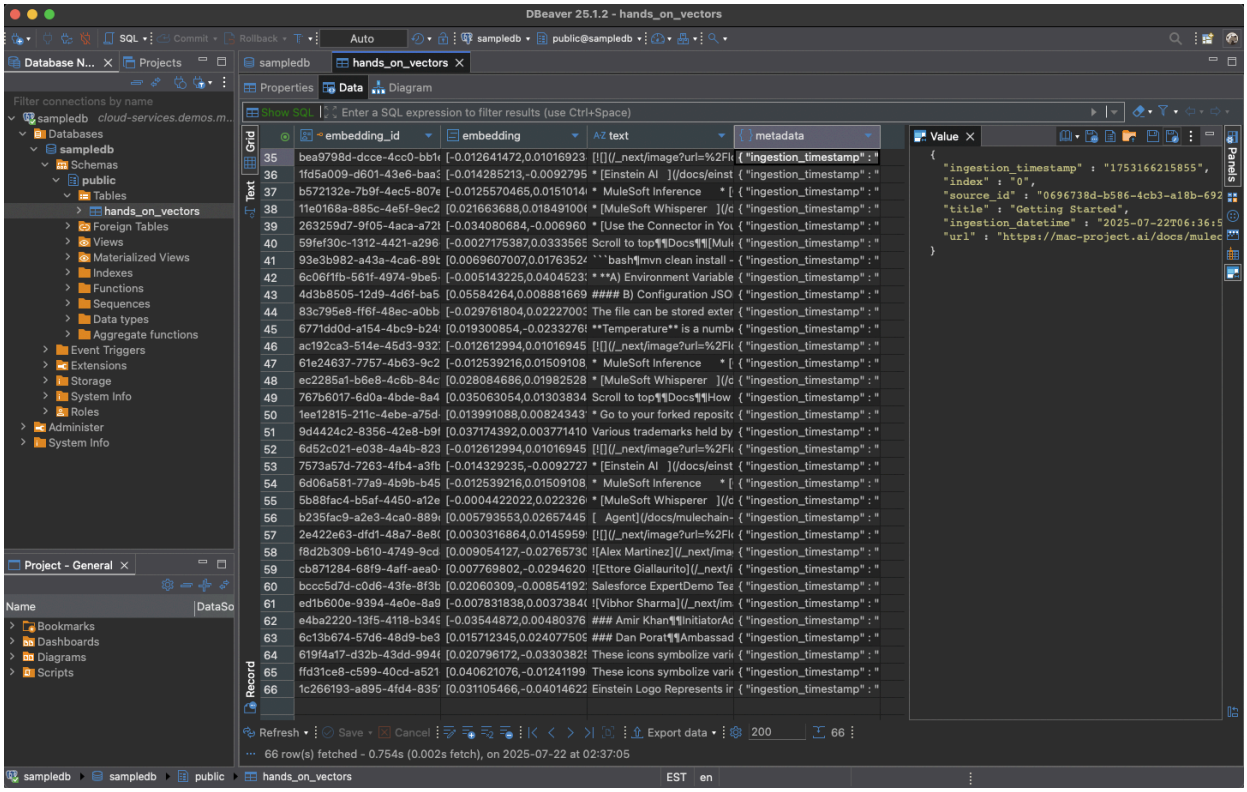


Test hands-on-mule-webcrawler-app

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



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


DBeaver 25.1.2 - hands_on_vectors
Database N...
Projects
Filter connections by name
sampledb cloud-services.demos.m...
Databases
sampledb
Schemas
public
Tables
hands_on_vectors
Foreign Tables
Views
Materialized Views
Indexes
Functions
Sequences
Data types
Aggregate functions
Event Triggers
Extensions
Storage
System Info
Roles
Administer
System Info
Project - General
Name
DataSo
Bookmarks
Dashboards
Diagrams
Scripts
sampledb
hands_on_vectors
Properties
Data
Diagram
Show SQL
Enter a SQL expression to filter results (use Ctrl+Space)
embedding_id
embedding
text
metadata
Value
Refresh
Save
Cancel
Export data
66 row(s) fetched - 0.754s (0.002s fetch), on 2025-07-22 at 02:37:05
EST
en