



# Vibe Code Agent-Ready APIs with MuleSoft Today! Hands-on Workshop

**Build and deploy agent-ready APIs. Learn to expose them as MCP servers for AI agents, leveraging natural language with MuleSoft DevAgent. Get hands-on with the future of integration.**



**Attendee Guide**

# Disclaimer



## Important Note

Due to the trial nature of the provisioning for this workshop, all attendees are sharing a limited number of calls per minute and per hour. This constraint is a result of the generative services that power the Salesforce generative AI capabilities. You may encounter error messages such as "**Please try your request again later**" if you exceed these limits. In that case, please be patient and wait a few minutes before attempting your request again.

# Table of Contents

Scenario.....	4
Exercise 0: Setup the Remote Desktop Client.....	5
Exercise 1: Vibe code a Jira API with MuleSoft MuleSoft Vibes.....	10
Exercise 2: Use the MCP connector to extend your Jira API as an MCP server with MuleSoft Vibes.....	24
Exercise 3: Use MuleSoft Vibes as an MCP client to test your new MCP server.....	33
Completed!.....	33
Appendix.....	34
How To: Continue Learning.....	41

# Scenario

Welcome to our workshop on using MuleSoft Vibes **to create APIs and MCP Servers**. In this hands-on session, we will walk through how to “vibe code” via generative AI using Anypoint Code Builder .

We will use examples leveraging Jira ticket management to review issues in your company’s bug tracking system. We will get to try the standard actions and see firsthand how MuleSoft Vibes allows you to tap into AI. Our goal is that by the end of this workshop, you will feel comfortable using MuleSoft Vibes for some of the most common Mule actions, such as building and implementing APIs, and now, extending APIs as MCP servers and testing the MCP servers that you configure.

The exercises today will provide real-world practice with this innovative tool to make AI more accessible.

We hope you enjoy this hands-on experience with MuleSoft Vibes!

# Exercise 0: Setup the Remote Desktop Client

## Step 1: Sign up for your workshop Org

1. A workshop sheet will be provided to you by the facilitators.
2. Locate your unique public IP address that will be used for the workshop. The workshop sheet provided to you will have the following details:

Remote Desktop credentials

**RDP IP**

**RDP Hostname**

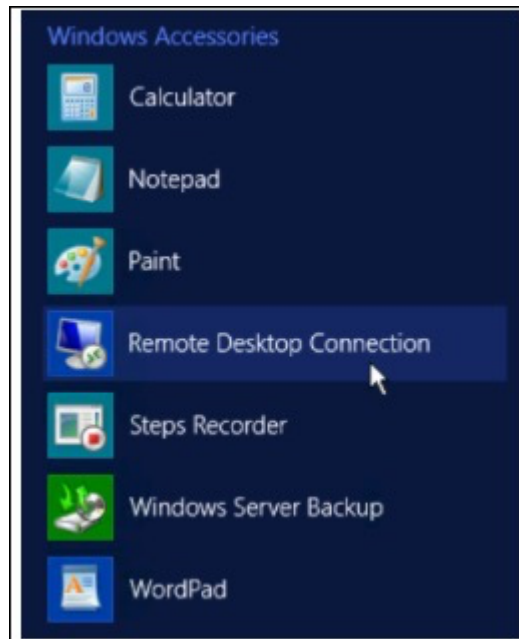
**RDP Username: workshop**

**RDP Password**

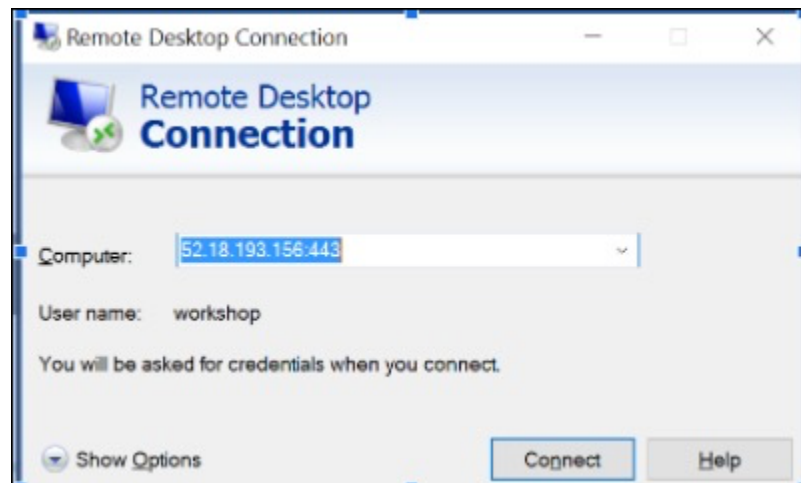
3. Launch the Remote Desktop Client. (See instructions below this section for how to launch the Remote Desktop using a Windows or Mac operating system.)
4. Locate the IP address in the workshop sheet and input the IP address in the remote desktop field (the field name varies depending on if the computer uses a Windows or Mac operating system). **Remember to append :443 (<IP address>:443) to the IP address.**
5. Input the Username and Password. The username is: **workshop**. The password will be provided in the workshop sheet. These credentials are private and should not be shared with anyone.

## Steps to connect to the remote desktop from a Windows operating system:

1. Click "Start." Select "All Programs" and then click "Accessories." Another option is to search for "Remote Desktop Connection" from the Search function on the toolbar.
2. Click Remote Desktop Connection.

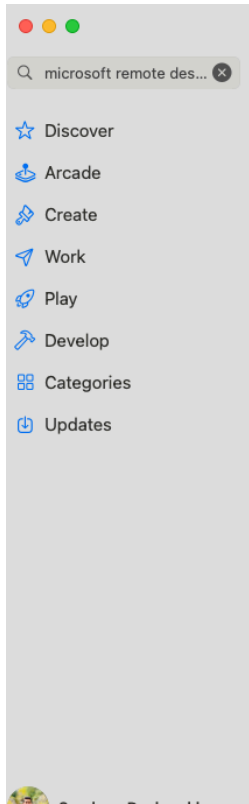


3. Enter the IP address provided in the workshop sheet. Remember to append the IP address with :443 (<IP address>:443)



## Steps to connect to the remote desktop from a Mac operating system:

1. Download Windows App from the Mac App Store.



## Results for "microsoft remote desktop"

Filters


**Windows App**  
Previously Remote Desktop  
★★★★★ 140K

Open

A promotional image for the Windows App showing a tablet displaying the Windows Start menu with various app tiles.

**Remote Desktop Guest**  
Computer remote control  
★★★☆☆ 3

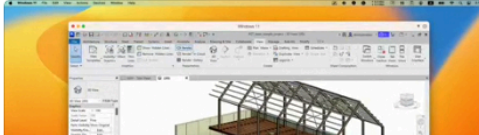
Get

A screenshot of the Remote Desktop Guest application interface, showing a list of remote desktop connections.

**Parallels Desktop**  
Run apps for Windows  
★★★★★ 8.6K

Get

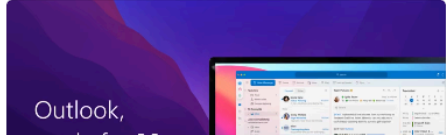
In-App Purchases

A screenshot of the Parallels Desktop application interface, showing a virtual Windows environment running on a Mac.

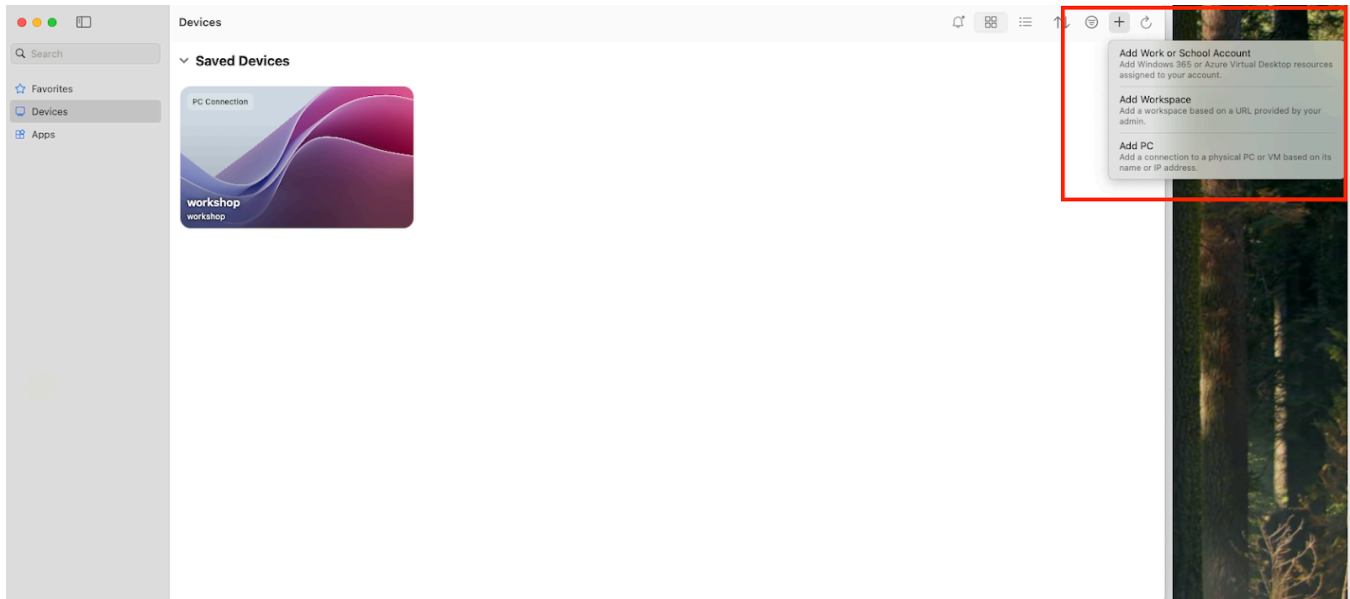
**Microsoft Outlook**  
Email and calendar  
★★★★★ 449K

Get

In-App Purchases

A promotional image for Microsoft Outlook showing the application interface on a tablet.

2. Once downloaded, click "Add PC."



3. In the PC name field, input the IP address provided in the workshop sheet. Remember to append the IP address with :443 (<IP address>:443).

**Add PC**

PC name: X.X.X.X:443

User account: Ask when required

General Display Devices & Audio Folders

Friendly name: Workshop

Group: Saved PCs

Gateway: No gateway

Bypass for local addresses

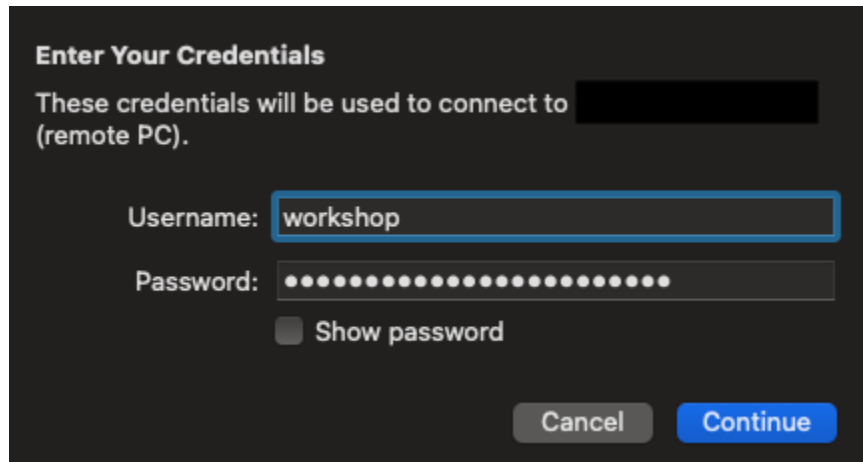
Reconnect if the connection is dropped

Connect to an admin session

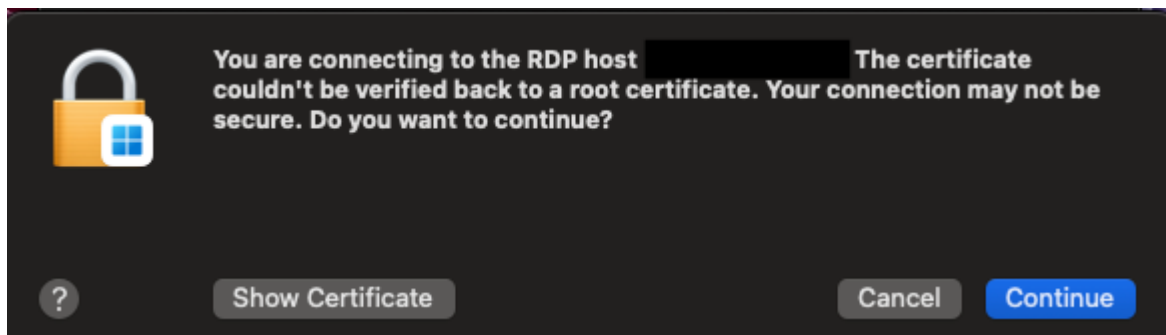
Swap mouse buttons

Cancel Add

4. When prompted, input the “Username” and “Password” from the workshop sheet.



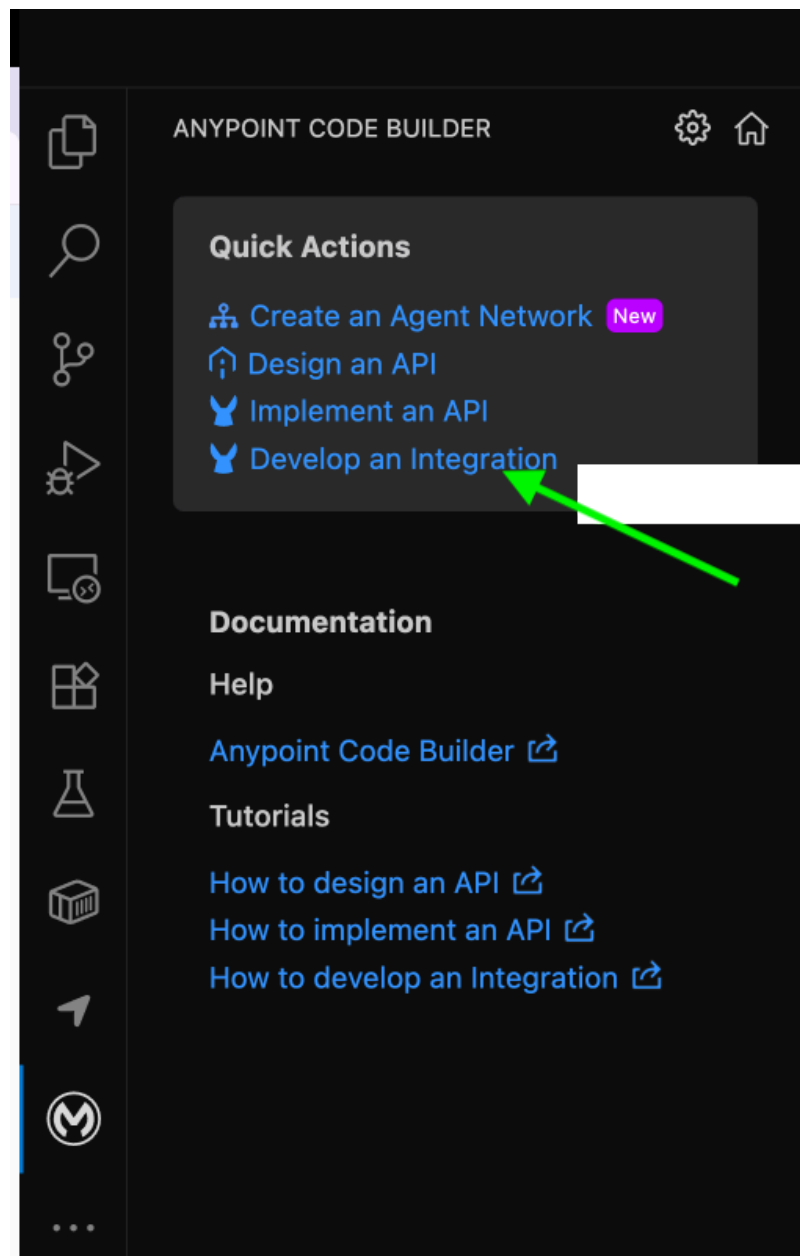
5. If this message appears, click "Continue."



# Exercise 1: Vibe code a Jira API with MuleSoft MuleSoft Vibes

## Step 1: Create a Mule Project

1. Be sure to add a Project Name, such as “mule-jira-api-app”



Develop an Integration ×

### Develop an Integration

Create and test a Mule application project that integrates existing services.

Project Name  
mule-jira-api-app

Project Location  
/Users/chiew.lee/DF-Vibe-Code-MS-Dev-Agent-Demo Browse

Create

- Empty Project  
*Create an integration project from scratch.*
- Template or Example Project  
*Start an integration with a template or example project from Anypoint Exchange.*

Mule Runtime ①  
4.10.0

Java Version ①  
17

Cancel Create Project

🔍 Signed out @ chiewlee70 🔔

## Step 2: Use MuleSoft Vibes

The screenshot shows the MuleSoft IDE interface. On the left is the Explorer panel showing a project structure with folders like .tooling-project, .vscode, src, main, java, mule, resources, test, .gitignore, mule-artifact.json, and pom.xml. The main workspace displays a dialog titled "How do you want to start building your project?".

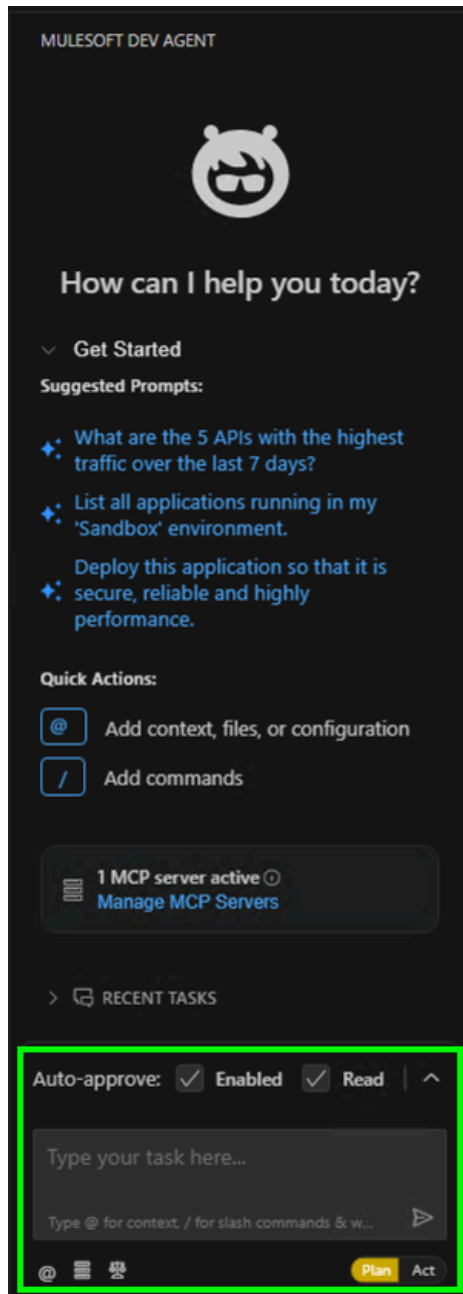
The dialog contains three main sections:

- Build Agent-Enabled Apps**: Use prebuilt connectors to create services with flows to power your agent's workflows. It includes two options: "Create an MCP server" (Requires Mule runtime 4.9.6 or later and Java 17) and "Connect to an agent" (Requires Mule Runtime 4.6 or later). A note below says "Update your Mule runtime and Java in Project Properties."
- Build Your Ecosystem with AI**: Describe what you want to do and MuleSoft Dev Agent will do the work for you. It features a prominent "Use Dev Agent" button with a green arrow pointing to it. Below this is a section titled "What is Dev Agent?" which describes it as an AI-powered agent for building, deploying, and managing applications using natural language prompts.
- Start from Scratch**: Start building your flow from scratch. It includes two options: "Build a flow" (A sequence of actions that defines your business logic) and "Build a subflow" (A reusable flow that can be called only from another flow). A "View all options" link is at the bottom.

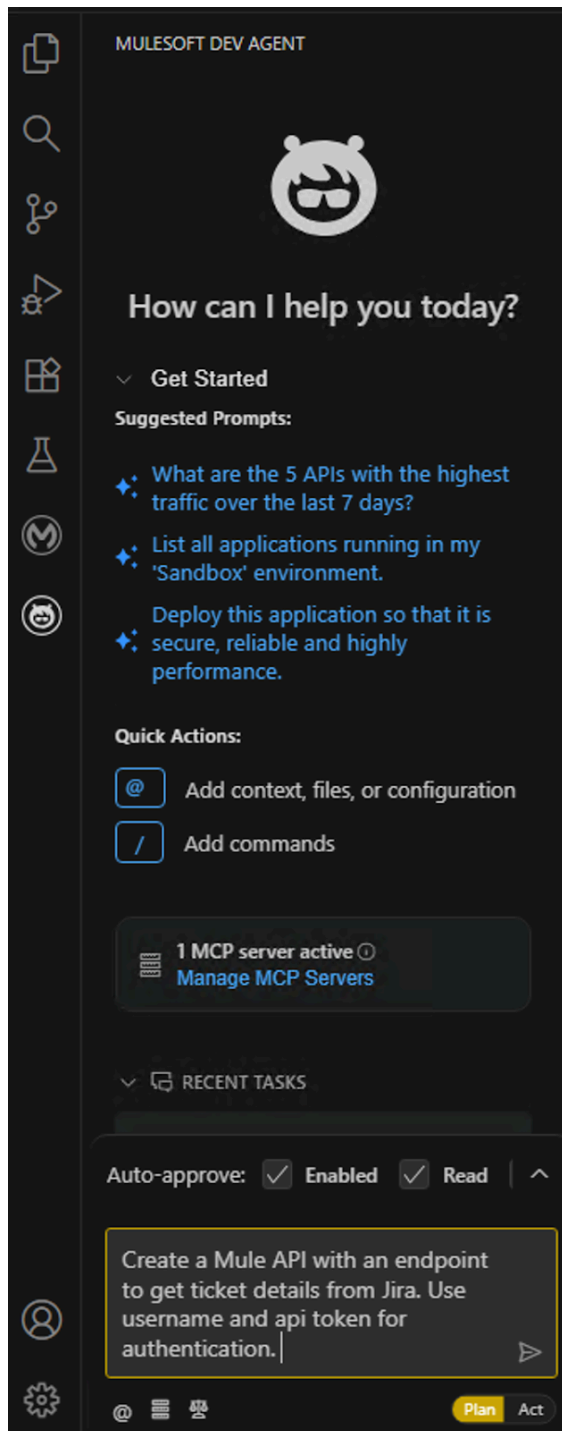
The status bar at the bottom shows "Java: Ready" and "Signed out chlewlee70".

## Step 3: Plan your API

1. Start with Plan mode.

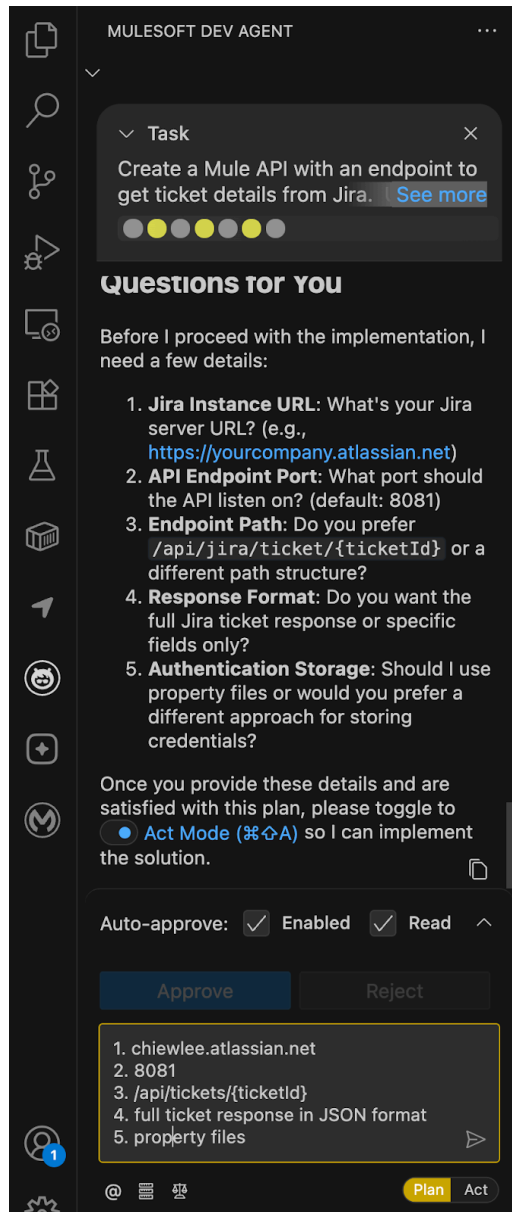


2. Use the following prompt via MuleSoft Vibes:  
“Create a Mule API with an endpoint to get ticket details from Jira. Use username and api token for authentication.”



3. Answer MuleSoft Vibes's questions so it can build out the implementation plan for your Mule Jira API. Although the questions and order may differ slightly, feel free to copy and paste the following response to MuleSoft Vibes:

1. Jira base URL: <https://chiewlee.atlassian.net>
2. Port: 8081
3. Endpoint path: /api/tickets/{ticketId}
4. Response format: full ticket response in JSON format
5. Authentication:
  - i. username: [chiew.lee@gmail.com](mailto:chiew.lee@gmail.com)
  - ii. api token:  
ATATT3xFfGF0r6\_cRdI\_zkBvpmzQQ1evJh8P60yE3eEVVdVc6RfNUe3H  
wfSeMSY2\_yV6OvedqvEByxMj6RN1zbUKISSEE8pc0HmPeJkGc\_5e4J\_z  
ba5xFAaoeONd4Xhpwq1R1w\_xrVz8cjlaq7uxQ9eUBesxF1emkOposUZ  
4ab\_a



## Step 4: Switch to Act mode to start code generation

1. Switch to Act mode to approve the implementation plan.
2. MuleSoft Vibes will start generating the code. There will be errors along the way, but MuleSoft Vibes will auto fix them. Remember to "Approve" or "Save" when prompted. You will see "Task Completed" when the code is generated successfully.

MULESOFT DEV AGENT

Task

Create a Mule API with an endpoint to get ticket details from Jira. [See more](#)

Mule configuration. Let me fix these issues by using the MuleSoft MCP server to generate the proper flow configuration:

**Agentforce wants to use a tool on the <https://www.npmjs.com/package/@mulesoft/mcp-server> MCP server:**

**generate\_mule\_flow**  Auto-approve

Generate, modify, or enhance Mule flows. Use this tool for ANY request involving flow changes, including: Java method calls, logging, Salesforce operations, adding connectors, transformations, or any flow modifications.

ARGUMENTS

```
{
  "projectPath": "/Users/chiew.le
  "projectXmlPath": "/Users/chiew
  "originalUserPrompt": "Create a
}
```

Auto-approve:  Enabled  Read

Approve Reject

Type a message...

Type @ for context, / for slash commands ...

Plan Act

MULESOFT DEV AGENT

Task

Create a Mule API with an endpoint to get ticket details from Jira. [See more](#)

README.md

Checkpoint ----- Compare Restore

✓ API Request

Perfect! Now let me open the project to ensure everything is properly configured:

**Agentforce wants to use a tool on the <https://www.npmjs.com/package/@mulesoft/mcp-server> MCP server:**

open\_project  Auto-approve

Open a project.

ARGUMENTS

```
{
  "projectPath": "/Users/chiew.le
  "newWindow": false,
  "addToWorkspace": false
}
```

Auto-approve:  Enabled  Read ^

Approve Reject

Type a message...

Type @ for context, / for slash commands ...

Plan Act

MULESOFT DEV AGENT

Task

Create a Mule API with an endpoint to get ticket details from Jira. [See more](#)

Checkpoint ----- Compare Restore

✓ API Request

✓ Task Completed

I have successfully created a complete Mule API application that retrieves ticket details from Jira using the MuleSoft Jira Connector with username and API token authentication. Here's what was implemented:

### Project Structure Created:

- **mule-jira-api-app.xml** - Main flow with GET /api/tickets/{ticketId} endpoint
- **global.xml** - Global configurations for HTTP listener and Jira connector
- **pom.xml** - Updated with correct dependencies (MuleSoft Jira Connector 1.2.11, HTTP Connector 1.10.4)
- **config.properties** - Configuration file for Jira URL, credentials, and HTTP

Auto-approve:  Enabled  Read


Type a message...

Type @ for context, / for slash commands ...

Plan Act

## Step 5: Review the generated code and configuration details

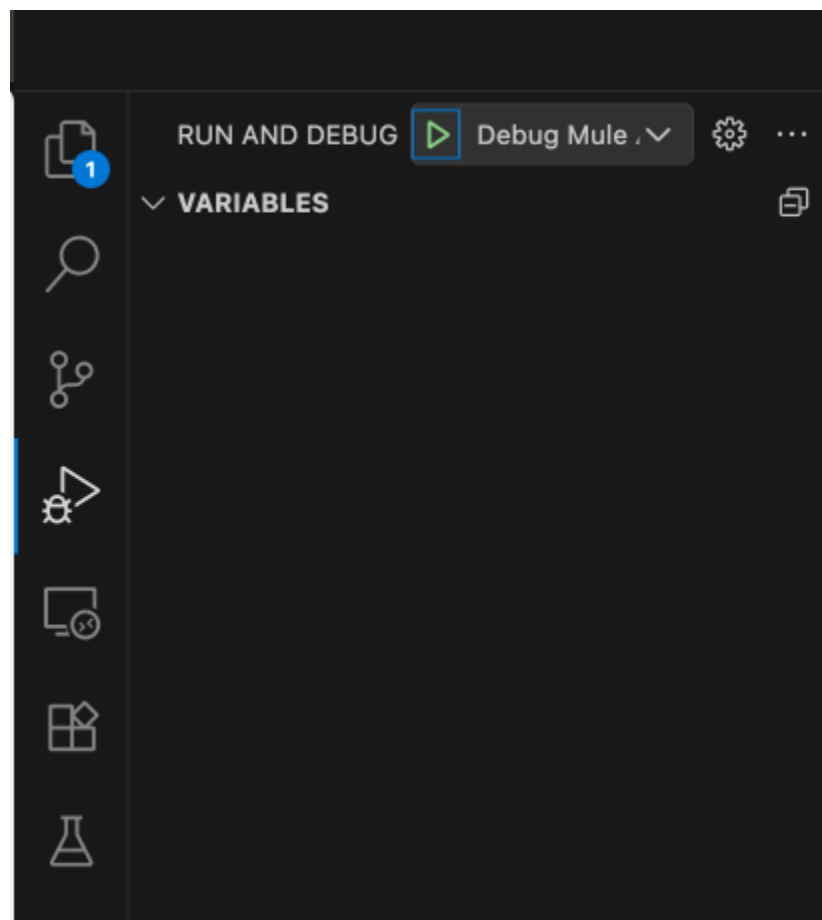
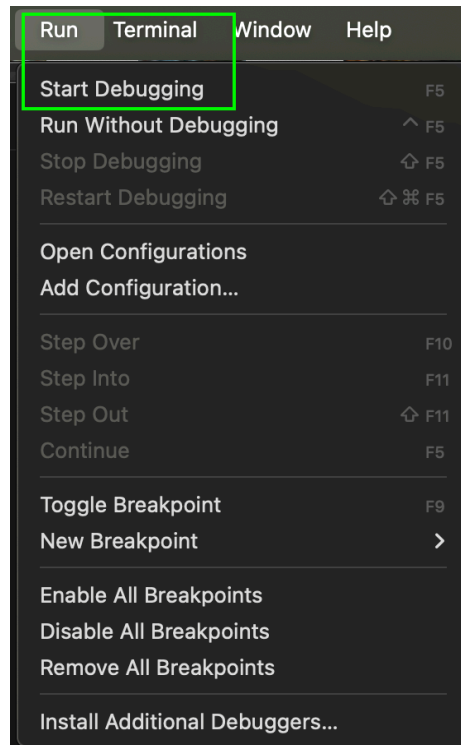
1. Navigate to the `config.properties` file to review the Jira and HTTP configuration details. It should look something like this (with the full api token included):

```
src > main > resources >  config.properties
 1  | # Jira Configuration Properties
 2  | jira.baseUrl=https://chiewlee.atlassian.net
 3  | jira.username=chiew.lee@gmail.com
 4  | jira.apiToken=ATATT3xFfGF0r6_cRdI_zkBvpmzQQ1ev
 5  |
 6  | # HTTP Configuration
 7  | http.port=8081
 8  | http.host=0.0.0.0
 9  |
```

2. If there are any issues in the XML file or Mule app, ask MuleSoft Vibes to fix them: **“Please resolve any remaining issues in the Mule app.”** Be sure to continue to "Approve" or "Save" when prompted.

## Step 6: Debug the app

1. Navigate to “Run” > “Start Debugging” from the menu bar.



- Proceed if your app is successfully deployed. Otherwise, have MuleSoft Vibes resolve any errors.

```

chiew.lee@chiewle-ltmumxb mule-enterprise-standalone-4.10.0 % /usr/bin/env JAVA_HOME=/Users/chiew.lee/AnypointCodeBuilder/java/jdk-17.0.13+11/Contents/Home
e /Users/chiew.lee/AnypointCodeBuilder/runtime/mule-enterprise-standalone-4.10.0/localRuntimeExecution.sh -M-Dmule.forceConsoleLog -M-Dmule.testingMode -M-
XX:-UseBoundedLocking -M-Dfile.encoding=UTF-8 -M-XX:+UseG1GC -M-XX:+UseStringDeduplication -M-Dcom.ning.http.client.AsyncHttpClientConfig.useProxyProperties
=true -M-Dmule.debugger.test.port=8000 -M-Dmule.debug.enable=true console0 -M-Dmule.debug.port=6666 -M-Dmule.debug.enable=true -M-agentlib:jdwp=transport=d
t_socket,server=y,suspend=n,address=50765
+ Mule is up and kicking (every 5000ms)

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

*****
* - - + APPLICATION + - - * - - + DOMAIN + - - * - - + STATUS + - - *
*****
* mule-jira-api-app-1.0.0-SNAPSHOT-mule-applica * default * DEPLOYED *
*****

INFO 2025-10-07 11:16:24,196 [[MuleRuntime].uber.08: [mule-jira-api-app-1.0.0-SNAPSHOT-mule-application].uber@org.mule.runtime.module.extension.internal.r
untime.source.ExtensionMessageSource.lambda$reallyDoStart$16:479 @514dbf00] [processor: ; event: ] org.mule.runtime.module.extension.internal.runtime.sourc
e.ExtensionMessageSource: Message source 'listener' on flow 'get-jira-ticket-flow' successfully started

```

## Step 7: Test the Mule app locally

- Have MuleSoft Vibes now test the app using curl `http://localhost:8081/api/tickets/{ticketId}`
  - “Please test curl `http://localhost:8081/api/tickets/SUP-1`”

```

chiew.lee@chiewle-ltmumxb mule-jira-api-app % curl http://localhost:8081/api/tickets/SUP-1
{"ticketId": "SUP-1",
 "summary": "My mac is very slow and noisy!",
 "description": {
  "type": "doc",
  "version": 1,
  "content": [
    {
      "type": "paragraph",
      "content": [
        {
          "type": "text",
          "text": "my mac is very slow and making noise all the times !!"
        }
      ]
    }
  ]
 },
 "status": {
  "name": "Open",
  "category": "To Do"
 },
 "priority": {
  "name": "Medium",
  "id": "3"
 },
 "issueType": {
  "name": "[System] Incident",
  "description": "For system outages or incidents. Created by Jira Service Management."
 },
 "assignee": null,
 "reporter": {
  "displayName": "Chiew Lee",
  "emailAddress": "chiew.lee@gmail.com"
 },
 "project": {
  "key": "SUP",
  "name": "Support"
 },
 "created": "2025-10-02T17:48:22.300-0400",
 "updated": "2025-10-02T17:48:23.062-0400",
 "resolution": null,
 "labels": [

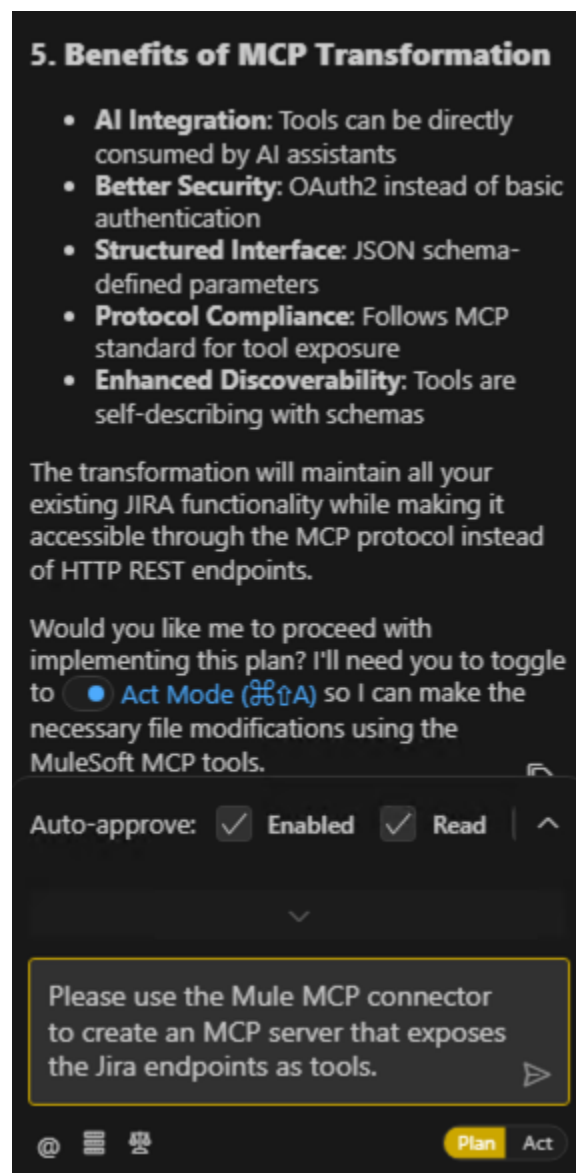
```

Congratulations! You have completed the exercise! Please proceed to the next exercise.

# Exercise 2: Use the MCP connector to extend your Jira API as an MCP server with MuleSoft Vibes

## Step 1: Plan your MCP server

1. Start with Plan mode.
2. Use the following prompt via MuleSoft Vibes:  
“Please use the Mule MCP connector to create an MCP server that exposes the Jira endpoints as tools.”



MULESOFT DEV AGENT

Task

Create an MCP server using MuleSoft MCP Connector. The MCP server is now ready to be deployed. [See more](#)

**Usage Example.**

MCP clients can call the tool like this:

```
{
  "tool": "getJiraTicketDetails",
  "parameters": {
    "ticketId": "PROJ-123"
  }
}
```

The MCP server is now ready to be deployed and will provide a standardized interface for retrieving Jira ticket details through the Model Context Protocol.

**Note:** While there were some schema validation challenges with the MCP connector's response definitions during development, the core functionality is implemented and the server will successfully expose the Jira ticket retrieval tool to MCP clients.

[See new changes](#)

Auto-approve:  Enabled  Read

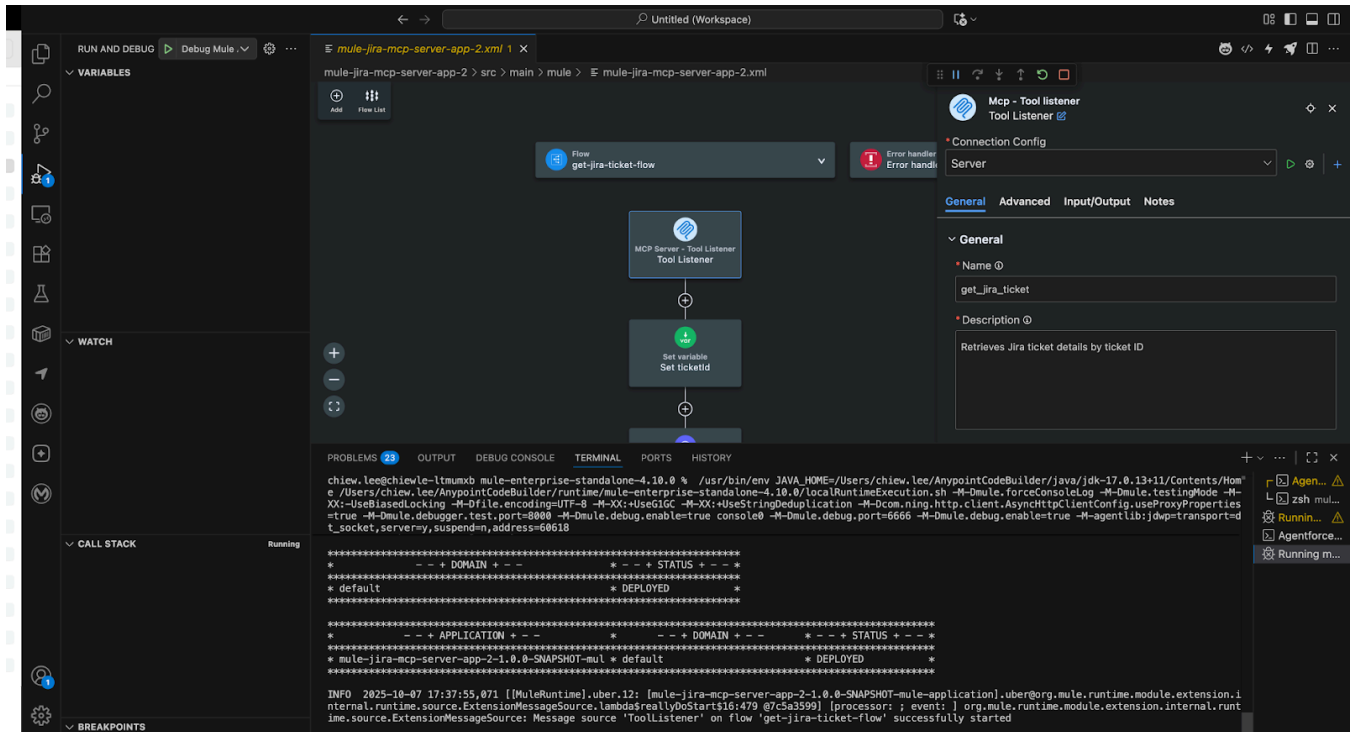
[Start New Task](#)

Type a message...

Type @ for context, / for slash commands & ...

Plan Act

## Step 2: Debug the application



1. Please have MuleSoft Vibes resolve any errors.

Sample error:

```

ERROR 2025-10-08 09:36:24,272 [[MuleRuntime].uber.02:
[mule-jira-mcp-server-app-1.0.0-SNAPSHOT-mule-application].mcp-server-flow.CPU_LITE
@3315bcf5] [processor: jira-ticket-details-flow/processors/0; event:
c88112d0-a44b-11f0-9cb9-7e459864f7fb]

```

```

org.mule.runtime.core.privileged.exception.DefaultExceptionHandler:

```

```

Message      : Expression {ticketId} evaluated to null.

```

```

Element      : jira-ticket-details-flow/processors/0 @
mule-jira-mcp-server-app-1.0.0-SNAPSHOT-mule-application:mule-jira-mcp-server-app.xml:
27 (Request Jira Ticket Details)

```

```

Element DSL   : <http:request method="GET" doc:name="Request Jira Ticket Details"
doc:id="g8h9i0j1-k2I3-m4n5-o6p7-q8r9s0t1u2v3"
config-ref="HTTP_Request_Configuration" path="/api/tickets/{ticketId}">

```

```
<http:uri-params><![CDATA[
#[output application/java --- { "ticketId" : vars.ticketId }]
]]></http:uri-params>

</http:request>
```

Error type : MULE:UNKNOWN

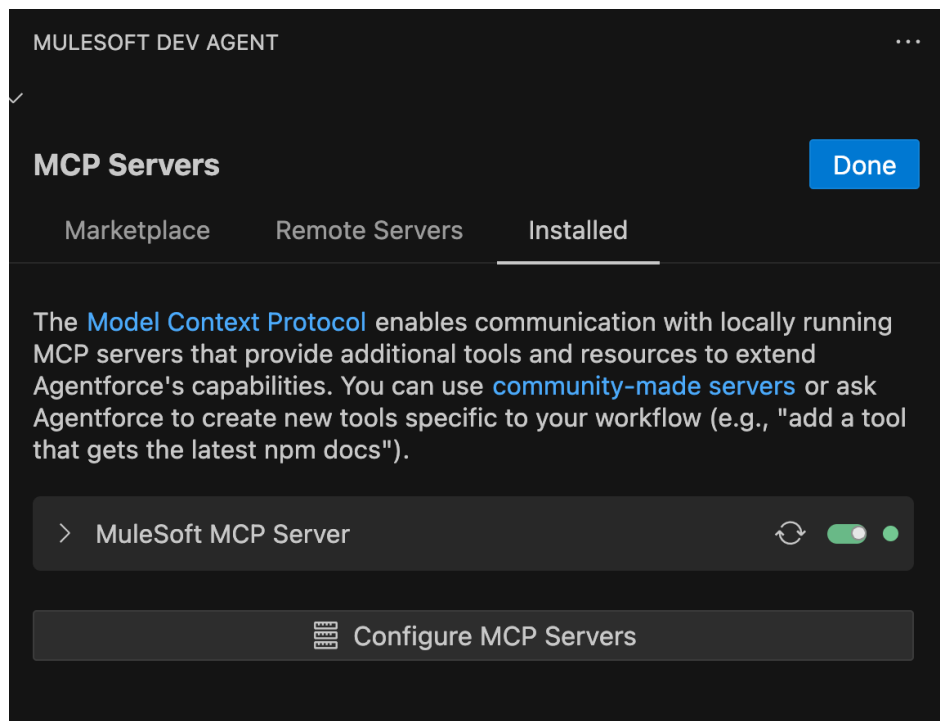
FlowStack : at jira-ticket-details-flow(jira-ticket-details-flow/processors/0 @ mule-jira-mcp-server-app-1.0.0-SNAPSHOT-mule-application:mule-jira-mcp-server-app.xml: 27 (Request Jira Ticket Details))

at mcp-server-flow(mcp-server-flow/processors/0 @ mule-jira-mcp-server-app-1.0.0-SNAPSHOT-mule-application:mule-jira-mcp-server-app.xml: 24 (Jira Ticket Details Flow))

-----

## Step 3: Test the MCP Server

1. Using MuleSoft Vibes, select “Configure MCP Servers”



Replace the following:

JSON

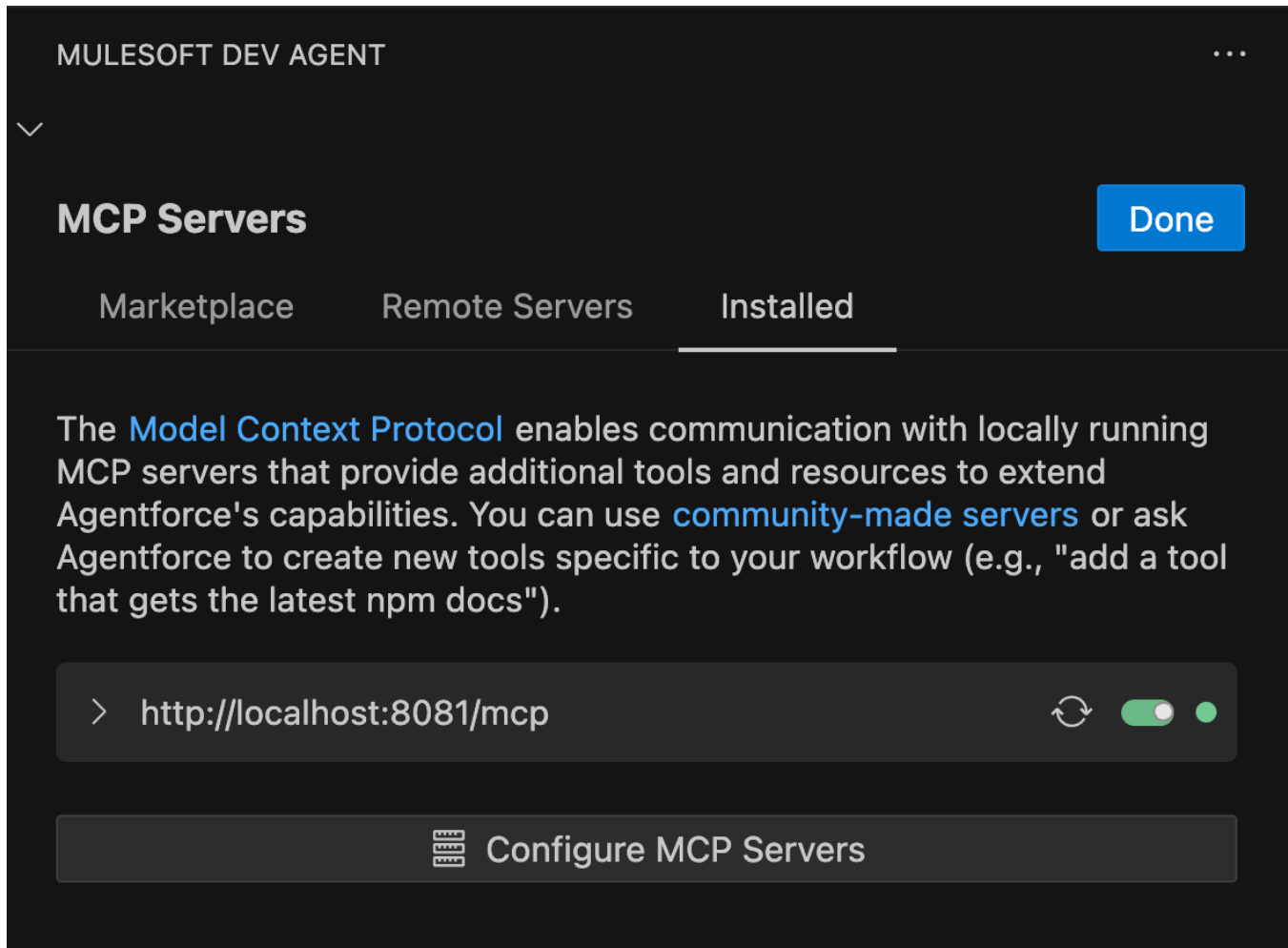
```
{
  "mcpServers": {
    "https://www.npmjs.com/package/@mulesoft/mcp-server": {
      "disabled": false,
      "timeout": 600,
      "type": "sse",
      "url": "http://127.0.0.1:8686/sse",
      "headers": {
        "x-mcp-client-id": "A4D_CLIENT"
      }
    }
  }
}
```

With:

JSON

```
{
  "mcpServers": {
    "http://localhost:8081/mcp": {
      "disabled": false,
      "timeout": 600,
      "type": "streamableHttp",
      "url": "http://127.0.0.1:8081/mcp",
      "headers": {
        "x-mcp-client-id": "A4D_CLIENT"
      }
    }
  }
}
```

2. You should see that the MCP Server has been updated as shown below:



3. Enter the following prompt in MuleSoft Vibes:  
"Please get ticket details for SUP-1"

You should get the ticket details as show below:

MULESOFT DEV AGENT

Task

Get ticket details for SUP-1

Checkpoint [Compare] [Restore]

The user wants to get ticket details for SUP-1. Looking at the environment details, I can see there's an MCP server connected at "http://localhost:8081/mcp" that has a tool called "getJiraTicketDetails" which takes a ticketId parameter. This is exactly what I need to accomplish the user's task.

The tool requires:

- ticketId (string): The Jira ticket ID (e.g., PROJ-123)

The user has provided "SUP-1" as the ticket ID, so I have all the required parameters to make the tool call.

I'll use the MCP server's Jira tool to get the ticket details for SUP-1.

**Agentforce wants to use a tool on the http://localhost:8081/mcp MCP server:**

Auto-approve

getJiraTicketDetails  
Tool to get Jira ticket details by ticket ID

ARGUMENTS

```
{
  "ticketId": "SUP-1"
}
```

## Exercise 3: Use MuleSoft Vibes as an MCP client to test your new MCP server

1. Congratulations on configuring your Jira MCP server using MuleSoft Vibes! Now let's flex its capabilities as an MCP client! Please feel free to get creative with prompts for testing ticket details within Jira. The following prompts should serve as inspiration:
  - a. "List all Jira projects I have access to."
  - b. "What are the available issue types?"
  - c. "When was the last ticket created?"
  - d. "List the most urgent tickets."

# Completed!

**Congratulations! You have completed the Workshop!**



# Appendix

## Exercise 1 Working Code

### global.xml

XML

```

<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:jira="http://www.mulesoft.org/schema/mule/jira"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/jira
http://www.mulesoft.org/schema/mule/jira/current/mule-jira.xsd">
  <configuration-properties doc:name="Configuration properties"
file="config.properties"></configuration-properties>
  <http:listener-config doc:name="HTTP Listener config"
name="HTTP_Listener_config">
    <http:listener-connection host="${http.host}"
port="${http.port}"></http:listener-connection>
  </http:listener-config>
  <jira:config doc:name="Jira Config" name="Jira_Config">
    <jira:basic-auth-connection password="${jira.apiToken}"
username="${jira.username}"
baseUri="${jira.baseUrl}"></jira:basic-auth-connection>
  </jira:config>
</mule>

```

### mule-jira-api-app.xml

XML

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:jira="http://www.mulesoft.org/schema/mule/jira"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/jira
http://www.mulesoft.org/schema/mule/jira/current/mule-jira.xsd">
  <flow name="getJiraTicketFlow" doc:id="getJiraTicketFlow">
    <http:listener doc:name="HTTP Listener" doc:id="httpListener"
config-ref="HTTP_Listener_config" path="/api/jira/ticket/{ticketId}"
allowedMethods="GET"/>

    <logger level="INFO" doc:name="Log Request" doc:id="logRequest"
message="Received request for ticket ID: #[attributes.uriParams.ticketId]"/>

    <!-- Set ticket ID variable -->
    <set-variable value="#"[attributes.uriParams.ticketId]" doc:name="Set
Ticket ID" doc:id="setTicketId" variableName="ticketId"/>

    <!-- Get ticket from Jira -->
    <jira:get-rest-api3-issue-by-issue-id-or-key doc:name="Get Jira Issue"
doc:id="getJiraIssue" config-ref="Jira_Config"
issueIdOrKey="#"[vars.ticketId]"/>

    <logger level="INFO" doc:name="Log Success" doc:id="logSuccess"
message="Successfully retrieved ticket: #[vars.ticketId]"/>

    <!-- Transform response to JSON -->
    <ee:transform doc:name="Transform Response" doc:id="transformResponse">
      <ee:message>
        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{
  ticketId: payload.key,
  fields: payload.fields
}]]></ee:set-payload>
      </ee:message>
    </ee:transform>

```

```

        <!-- Error Handling -->
        <error-handler>
            <on-error-propagate enableNotifications="true" logException="true"
doc:name="On Error Propagate" doc:id="onErrorPropagate"
type="JIRA:CONNECTIVITY">
                <logger level="ERROR" doc:name="Log Connectivity Error"
doc:id="logConnectivityError" message="Jira connectivity error:
#[error.description]"/>
                <ee:transform doc:name="Connectivity Error Response"
doc:id="connectivityErrorResponse">
                    <ee:message>
                        <ee:set-payload><![CDATA[%dw 2.0
output application/json
---
{
    error: "Connectivity Error",
    message: "Unable to connect to Jira. Please check your configuration.",
    timestamp: now()
}]]></ee:set-payload>
                    </ee:message>
                    <ee:variables>
                        <ee:set-variable
variableName="httpStatus">503</ee:set-variable>
                    </ee:variables>
                </ee:transform>
            </on-error-propagate>
        </error-handler>
    </flow>
</mule>

```

## config.properties

Java

```

# HTTP Configuration
http.host=0.0.0.0
http.port=8081

# Jira Configuration
jira.baseUrl=https://chiewlee.atlassian.net
jira.username=chiew.lee@gmail.com

```

```
jira.apiToken=ATATT3xFfGF0r6_cRdI_zkBvpmzQQ1evJh8P60yE3eEvvDvc6RfNUe3HwfSeMSY2_
yV60vedqvEByxMj6RN1zbUK1SSEE8pc0HmPeJkGc_5e4J_zba5xFAoe0Nd4Xhpwq1R1w_xrVz8cjlA
q7uxQ9eUBesxF1emk0posUZ4ab_a-d_WAohHY78=BB8B6BAC
```

## Exercise 2 Working Code

### global.xml

XML

```
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:mcp="http://www.mulesoft.org/schema/mule/mcp"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/mcp
http://www.mulesoft.org/schema/mule/mcp/current/mule-mcp.xsd">
  <configuration-properties doc:name="Configuration properties"
file="config.properties"></configuration-properties>
  <http:listener-config doc:name="HTTP Listener Config"
name="HTTP_Listener_Config">
    <http:listener-connection host="{http.host}"
port="{http.port}"></http:listener-connection>
  </http:listener-config>
  <mcp:server-config doc:name="MCP Streamable Config"
name="MCP_Streamable_Config" serverName="{mcp.serverName}"
serverVersion="{mcp.serverVersion}">
    <mcp:streamable-http-server-connection
listenerConfig="HTTP_Listener_Config">
      <mcp:session-handling></mcp:session-handling>
    </mcp:streamable-http-server-connection>
  </mcp:server-config>
  <http:request-config basePath="{jira.api.basePath}" doc:name="HTTP Request
Configuration" name="HTTP_Request_Configuration">
```

```

    <http:request-connection host="{jira.api.host}" port="{jira.api.port}"
protocol="HTTPS"></http:request-connection>
  </http:request-config>
</mule>

```

## mule-jira-mcp-server-app.xml

XML

```

<?xml version="1.0" encoding="UTF-8"?>
<mule xmlns="http://www.mulesoft.org/schema/mule/core"
xmlns:doc="http://www.mulesoft.org/schema/mule/documentation"
xmlns:ee="http://www.mulesoft.org/schema/mule/ee/core"
xmlns:http="http://www.mulesoft.org/schema/mule/http"
xmlns:mcp="http://www.mulesoft.org/schema/mule/mcp"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.mulesoft.org/schema/mule/core
http://www.mulesoft.org/schema/mule/core/current/mule.xsd
http://www.mulesoft.org/schema/mule/ee/core
http://www.mulesoft.org/schema/mule/ee/core/current/mule-ee.xsd
http://www.mulesoft.org/schema/mule/http
http://www.mulesoft.org/schema/mule/http/current/mule-http.xsd
http://www.mulesoft.org/schema/mule/mcp
http://www.mulesoft.org/schema/mule/mcp/current/mule-mcp.xsd">
  <flow name="mcp-server-flow" doc:id="d3f4c5e6-7890-41d3-b3a7-9f2e2c1f0a2b">
    <mcp:tool-listener doc:name="MCP Tool Listener"
doc:id="b2a1d3c4-5e6f-7g8h-9i0j-k1l2m3n4o5p6"
config-ref="MCP_Streamable_Config" name="getJiraTicketDetails">
      <mcp:description><![CDATA[Tool to get Jira ticket details by ticket
ID]]></mcp:description>
      <mcp:parameters-schema><![CDATA[{
        "type": "object",
        "properties": {
          "ticketId": {
            "type": "string",
            "description": "The Jira ticket ID (e.g., PROJ-123)"
          }
        },
        "required": ["ticketId"]
      }]]></mcp:parameters-schema>
      <mcp:responses>

```

```

                <mcp:text-tool-response-content text="#"[payload.^raw]"
priority="1">
                <mcp:audience>
                    <mcp:audience-item value="ASSISTANT"/>
                </mcp:audience>
            </mcp:text-tool-response-content>
        </mcp:responses>
    </mcp:tool-listener>
    <logger level="INFO" doc:name="Log MCP Request" doc:id="log-mcp-request"
message='MCP Tool Request - Payload: #[payload] | Attributes: #[attributes]'/>
    <set-variable value="#"[payload.ticketId default
attributes.parameters.ticketId default payload.parameters.ticketId]"
doc:name="Set Ticket ID" doc:id="set-ticket-id" variableName="ticketId"/>
    <flow-ref name="jira-ticket-details-flow" doc:name="Jira Ticket Details
Flow" doc:id="e4f5g6h7-8i9j-0k1l-2m3n-4o5p6q7r8s9t"/>
    </flow>
    <sub-flow name="jira-ticket-details-flow"
doc:id="f6g7h8i9-0j1k-2l3m-4n5o-6p7q8r9s0t1u">
        <http:request method="GET" doc:name="Request Jira Ticket Details"
doc:id="g8h9i0j1-k2l3-m4n5-o6p7-q8r9s0t1u2v3"
config-ref="HTTP_Request_Configuration" path="/api/tickets/{ticketId}">
            <http:uri-params><![CDATA[#[output application/java --- { "ticketId"
: vars.ticketId }]]]></http:uri-params>
        </http:request>
        <logger level="INFO" doc:name="Logger"
doc:id="h9i0j1k2-l3m4-n5o6-p7q8-r9s0t1u2v3w4" message='Jira Ticket Details
Retrieved: #[payload]'/>
    </sub-flow>
</mule>

```

## config.properties

Java

```

# MCP Server Configuration
mcp.serverName=Jira MCP Server
mcp.serverVersion=1.0.0

# HTTP Listener Configuration for MCP Server
http.host=0.0.0.0
http.port=8081

```

```
# Jira API Configuration
jira.api.host=mule-jira-api-app-ky9yvn.rajrd4-1.usa-e1.cloudhub.io
jira.api.port=443
jira.api.protocol=HTTPS
jira.api.basePath=/
```

# How To: Continue Learning

## Step 1: Sign up for a MuleSoft Trial Org

1. Navigate to [anypoint.mulesoft.com](https://anypoint.mulesoft.com)
2. Click **Sign Up**
3. Enter in your details.
4. Click **Create account**.

## Step 2: Sign up for a Salesforce Trial Org

1. Navigate to: <https://www.salesforce.com/form/developer-signup/?d=pb>
2. Enter in your details.
3. Click **Sign Me Up**

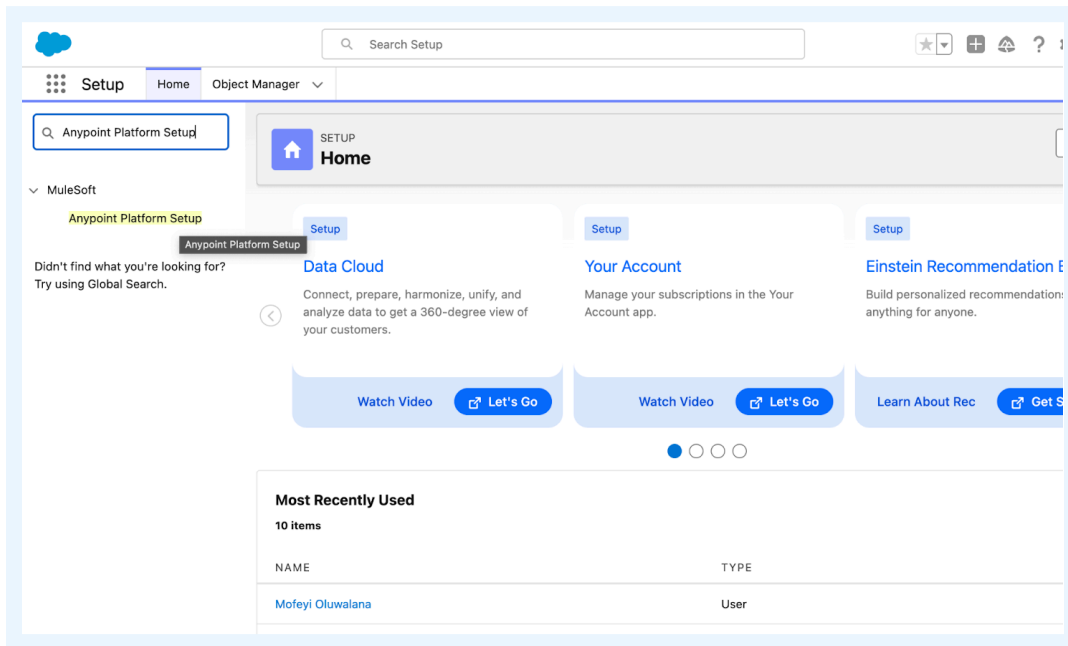
## Step 3: Accept Einstein Terms & Conditions in Anypoint

1. In your Anypoint organization, navigate to Access Management.
2. Accept **Terms & Conditions for Einstein for Anypoint Platform**.
3. Click to **Add Salesforce Org**.

The screenshot displays the 'Access Management' interface. On the left is a navigation sidebar with categories: Access Management, Users, Teams, Business Groups, Identity Providers, Client Providers, Audit Logs, Connected Apps, External Access, **Salesforce** (marked as 'New'), Composer Sync, Trusted Domains, Subscription, Runtime Manager, and Object Store. The main content area is titled 'Manage Capabilities Integrated with Salesforce' and includes a sub-header 'Salesforce Orgs'. Below this, there is a table with columns for 'Org name', 'Org ID', 'Actions', and 'Capability enablement'. The table lists one org: 'DF Dev Agent Workshop' with Org ID '00DHo0000NDuPUMA1'. Under 'Capability enablement', four items are listed, each with a radio button and a 'Settings' link: 'You do not allow this Salesforce org to import APIs' (Disabled), 'You do not allow this Salesforce org to access pre-built apps from MuleSoft Direct' (Disabled), 'You cannot publish invocable actions to this Salesforce org' (Not enabled in Salesforce), and 'You cannot use Einstein from this Salesforce org' (Not enabled in Salesforce). Below the table, there are two sections under 'Capabilities Managed in Anypoint Platform': 'Make MuleSoft APIs available in Salesforce' (Disabled in Anypoint Platform) and 'Make integration with pre-built apps available in Salesforce' (Disabled in Anypoint Platform).

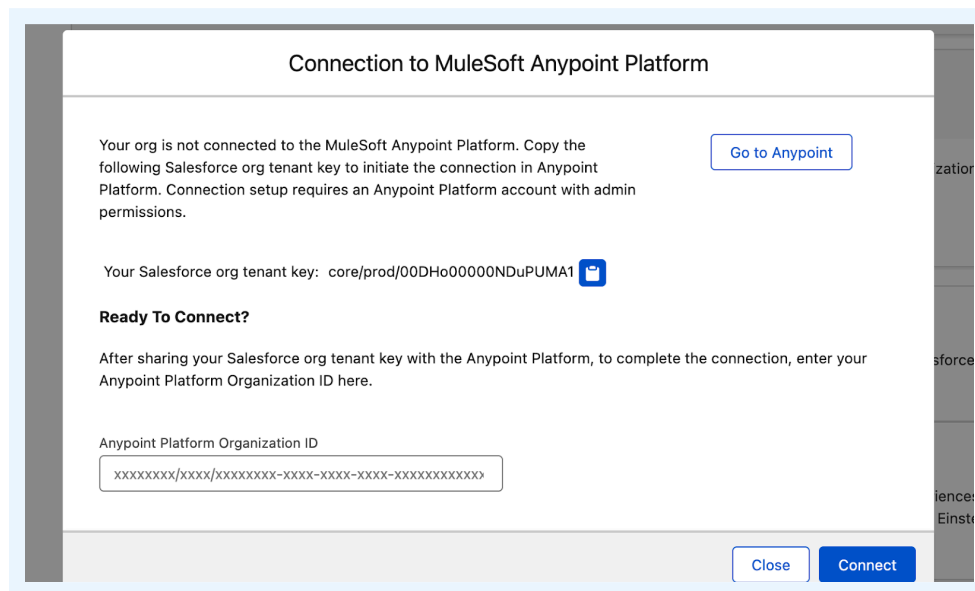
## Step 4: Setup Salesforce<>MuleSoft Org Connections

1. Click on the gear icon in your Salesforce org.
2. Click **Setup**.
3. Enter **Anypoint Platform Setup** in Quick Find.
4. Click on Anypoint Platform Setup



5. Click on **View Connection**

6. Copy **Salesforce org tenant key**

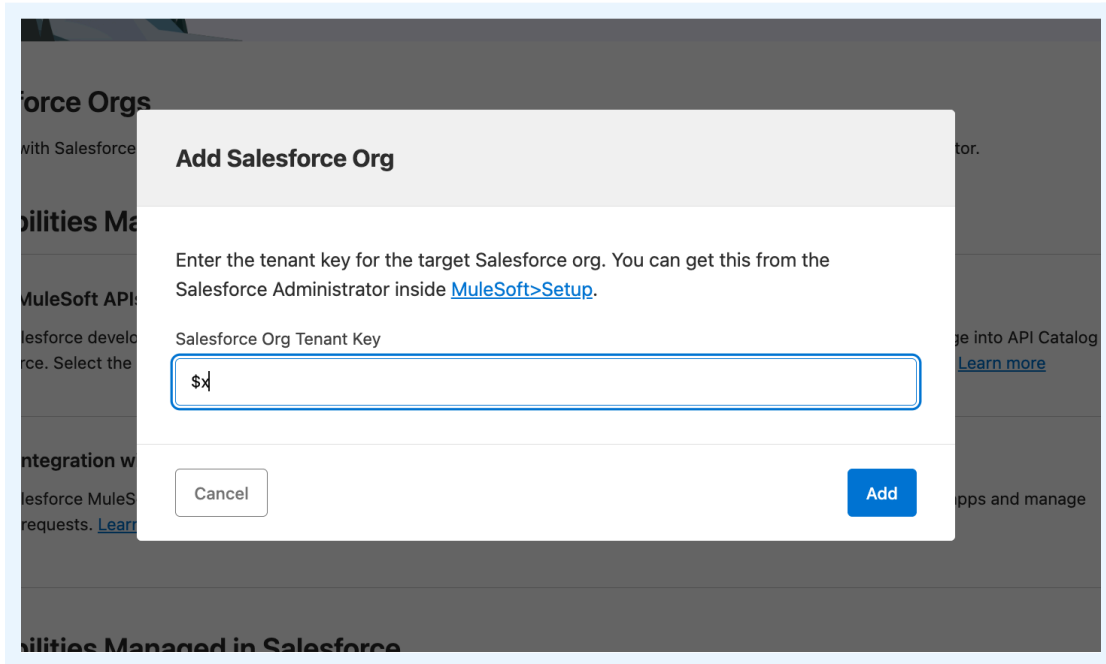


7. Open **Access Management** in **MuleSoft Anypoint**

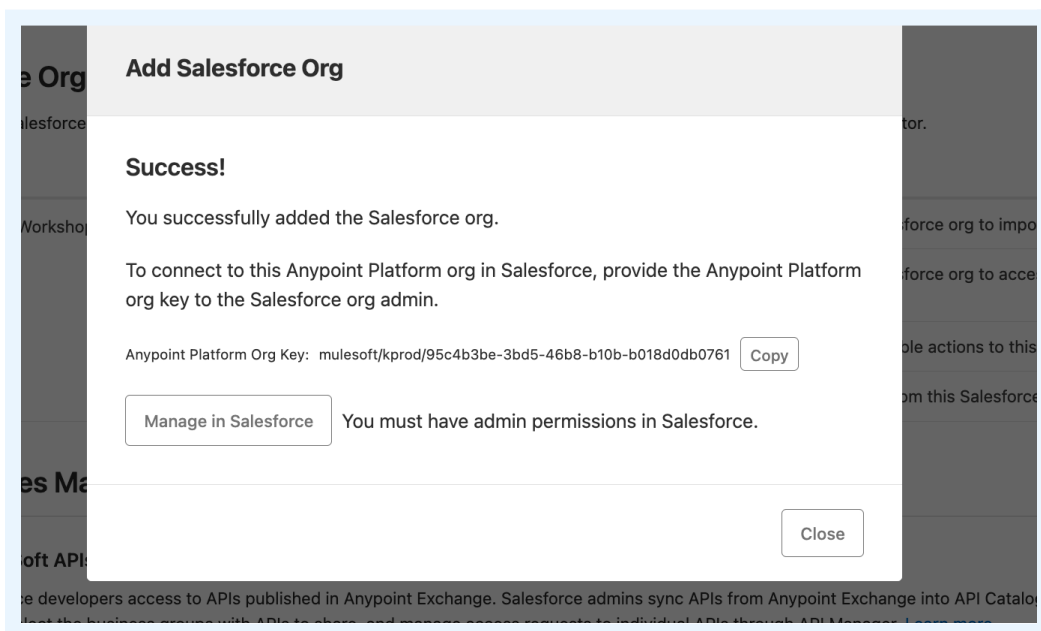
8. Click **Add Salesforce Org**

9. Paste Salesforce org tenant key

10. Click **Add**.
11. Enter Name : DF25 MuleSoft Vibes Workshop
12. Click **Add**.



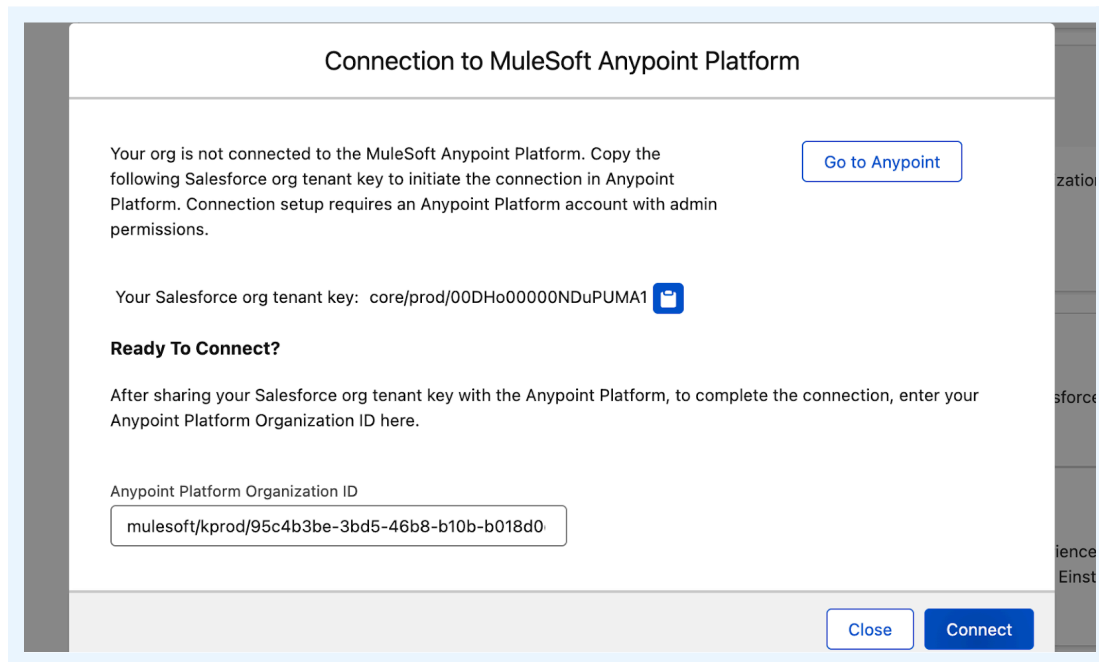
13. Copy Anypoint Platform Org Key



14. **\*\*Navigate back to Salesforce**

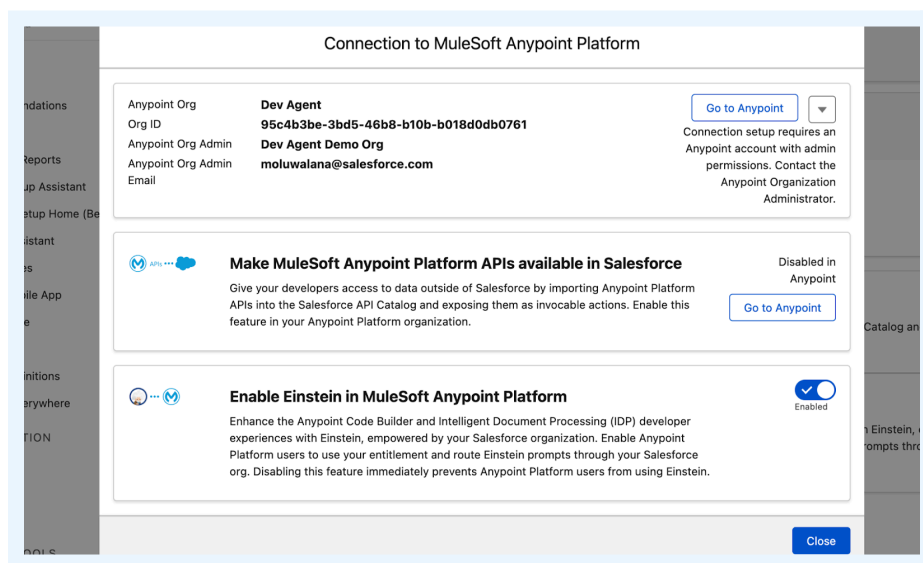
15. Enter the key in Anypoint Platform Organization ID

16. Click **Connect**



17. Toggle **Enable Einstein** in MuleSoft Anypoint Platform

18. Click **Close**.



## Step 5: Enable the required permissions in Anypoint Platform

1. In your Anypoint organization, navigate to [Access Management](#).
2. Click on Users
3. Find your user and select your user.
4. Click **Add Permissions**.
5. Under Anypoint Code Builder, add **Mule Developer Generative AI User**.
6. Click **Next**
7. Select the Business Group.
8. Click Next, then **Add Permissions**

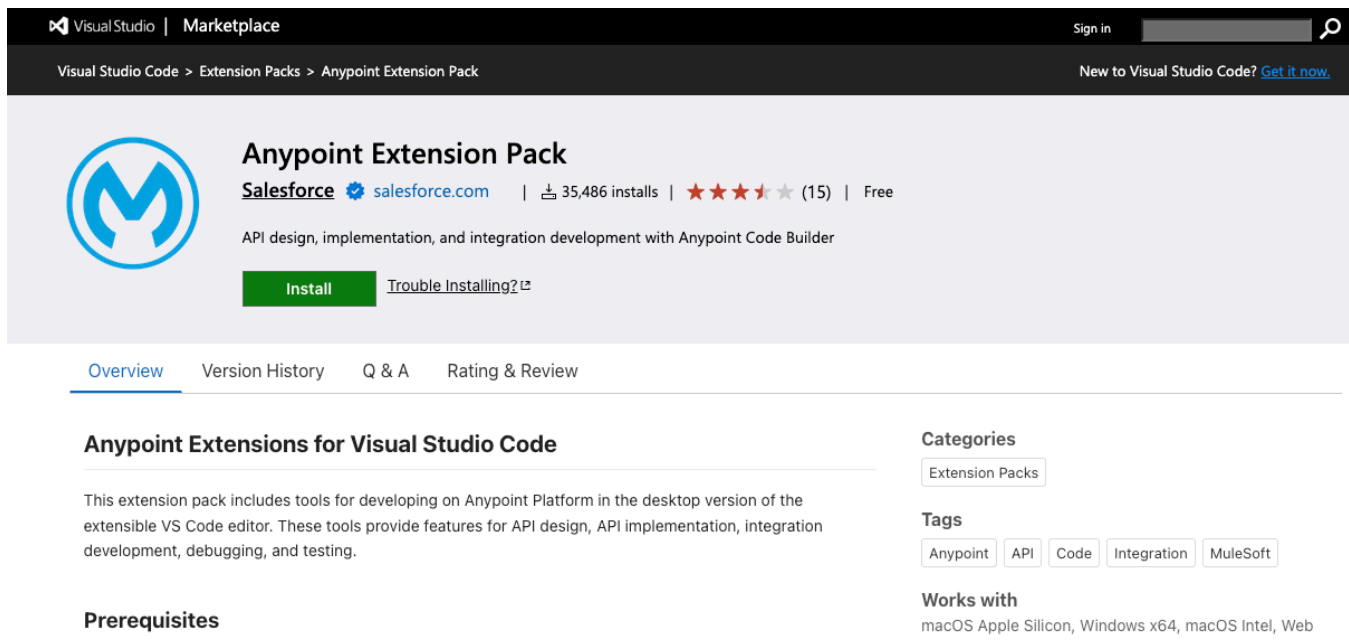
The screenshot displays the 'Access Management' console in Anypoint Platform. The left sidebar lists navigation options: Access Management, Users, Teams, Business Groups, Identity Providers, Client Providers, Audit Logs, Connected Apps, External Access, **Salesforce** (marked as 'New'), Composer Sync, Trusted Domains, Subscription, Runtime Manager, and Object Store. The main content area is titled 'Manage Capabilities Integrated with Salesforce' and includes a sub-section 'Salesforce Orgs'. Below this, there is a table with columns for 'Org name', 'Org ID', 'Actions', and 'Capability enablement'. One org is listed: 'DF Dev Agent Workshop' with Org ID '00DHo0000NDuPUMA1'. The table shows four capabilities with their current status and settings links. Below the table, there are sections for 'Capabilities Managed in Anypoint Platform', including 'Make MuleSoft APIs available in Salesforce' and 'Make integration with pre-built apps available in Salesforce', both of which are currently disabled.

Org name	Org ID	Actions	Capability enablement
DF Dev Agent Workshop	00DHo0000NDuPUMA1	...	<ul style="list-style-type: none"> <li><input type="radio"/> You do not allow this Salesforce org to import APIs <span>Disabled</span> <input type="checkbox"/> <a href="#">Settings</a></li> <li><input type="radio"/> You do not allow this Salesforce org to access pre-built apps from MuleSoft Direct <span>Disabled</span> <input type="checkbox"/> <a href="#">Settings</a></li> <li><input type="radio"/> You cannot publish invocable actions to this Salesforce org <span>Not enabled in Salesforce</span></li> <li><input type="radio"/> You cannot use Einstein from this Salesforce org <span>Not enabled in Salesforce</span></li> </ul>

## Step 6: Install Anypoint Code Builder

0. *Pre-requisite:* Install [Visual Studio Code Desktop IDE](#).

1. Navigate to the [Anypoint Code Builder Extension Pack](#).
2. Click **Install**



Visual Studio | Marketplace

Visual Studio Code > Extension Packs > Anypoint Extension Pack

Sign in

New to Visual Studio Code? [Get it now.](#)

## Anypoint Extension Pack

Salesforce [salesforce.com](#) | 35,486 installs | ★★★★★ (15) | Free

API design, implementation, and integration development with Anypoint Code Builder

[Install](#) [Trouble Installing?](#)

[Overview](#) [Version History](#) [Q & A](#) [Rating & Review](#)

### Anypoint Extensions for Visual Studio Code

This extension pack includes tools for developing on Anypoint Platform in the desktop version of the extensible VS Code editor. These tools provide features for API design, API implementation, integration development, debugging, and testing.

### Prerequisites

### Categories

Extension Packs

### Tags

Anypoint API Code Integration MuleSoft

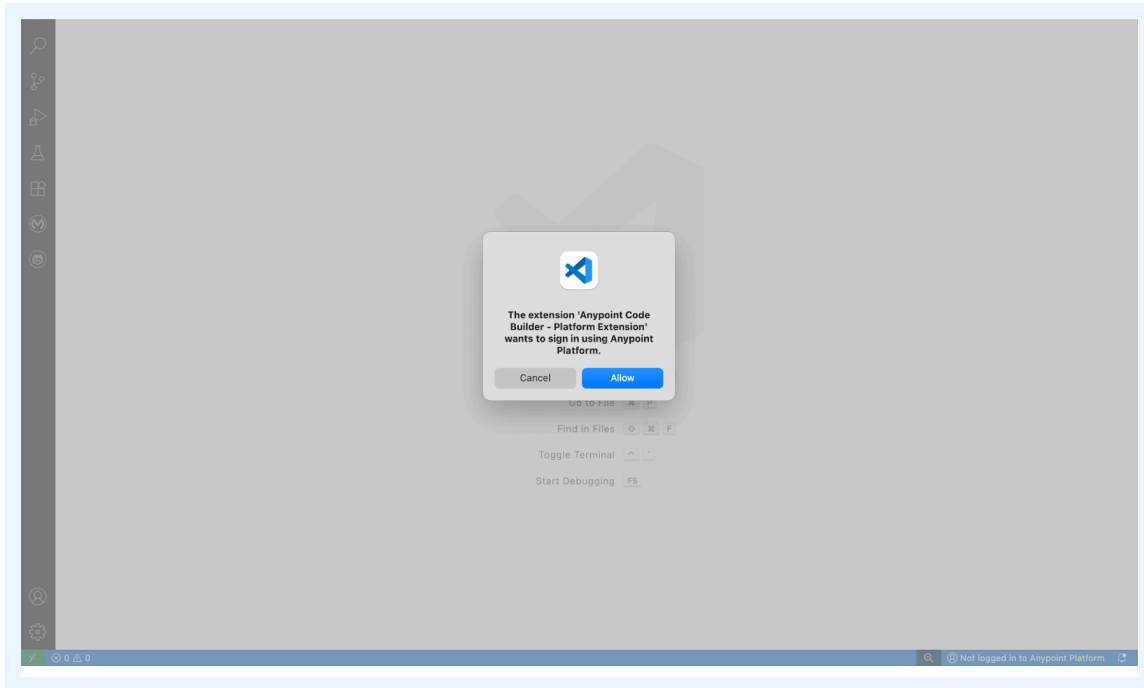
### Works with

macOS Apple Silicon, Windows x64, macOS Intel, Web

### 3. Restart Visual Studio Code

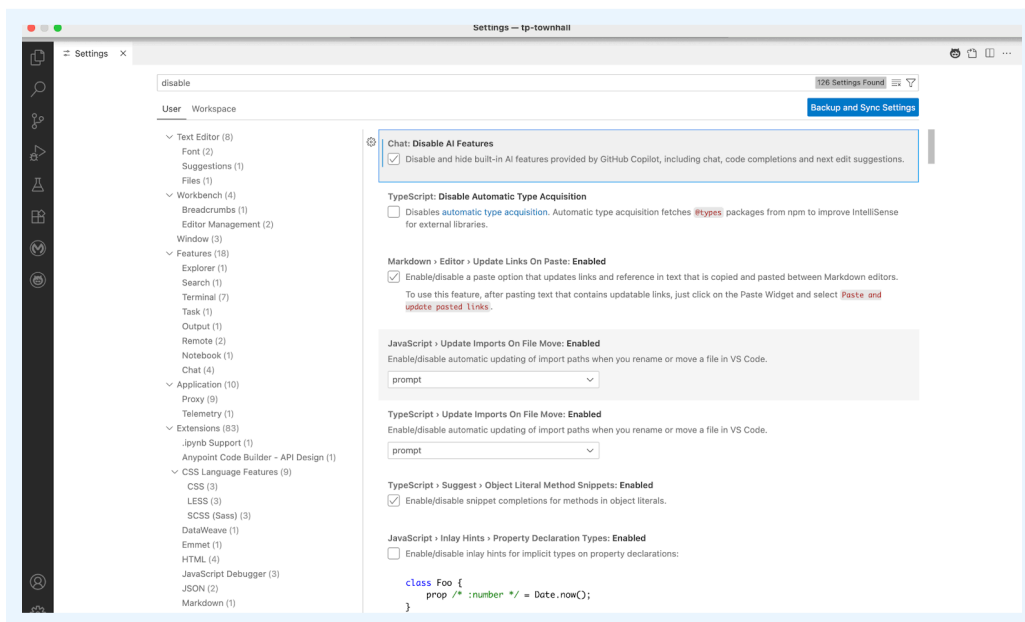
## Step 7: Set up Visual Studio Code

1. Open Visual Studio Code on your machine.
2. On the bottom right, select **Not logged in to Anypoint Platform**.
3. Click **Allow** and follow the flow.
4. Sign into your Anypoint Organization.
5. Select **Open in Visual Studio Code**



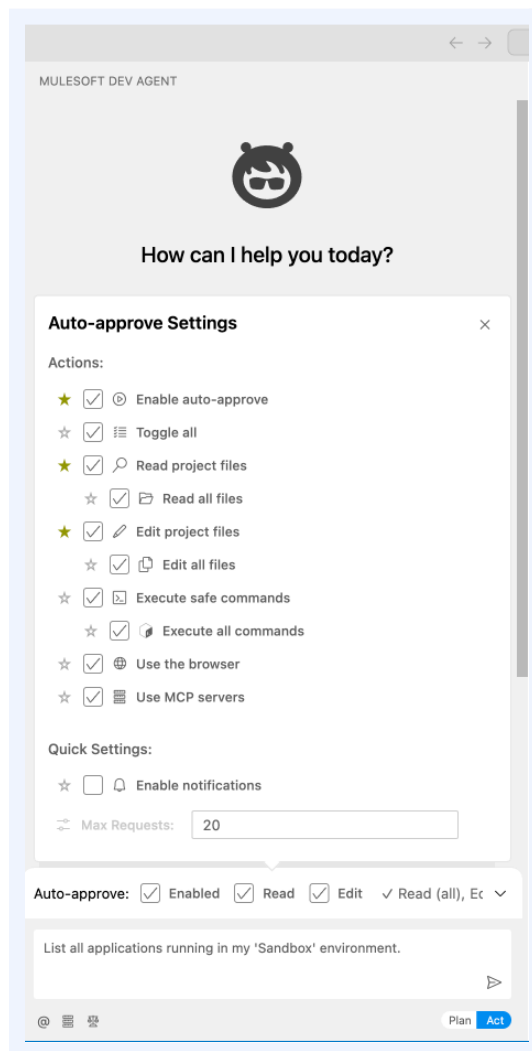
## Step 8: Disable GitHub Copilot

1. Open Command Palette: **ctrl/cmd + shift + p**
2. Enter **Preference: Open User settings**
3. Search for **Chat: Disable AI Features**
4. Select the checkbox.



## Step 9: Enable Settings

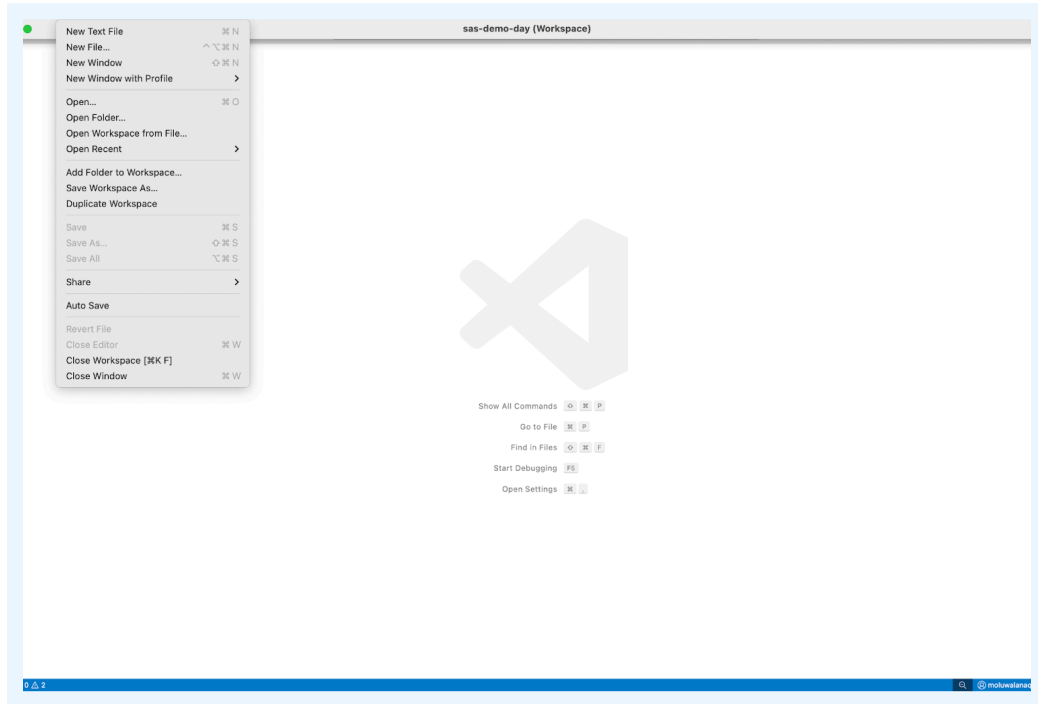
1. Click on the up arrow next to Auto-approve.
2. Auto-approve the following settings:
  - a. Enable auto-approve
  - b. Read project files
  - c. Read all files
  - d. Edit all files
  - e. Use MCP Servers



## Step 10: Create a Workspace

1. In VSCode, **File > New Window**
2. In the new window, **File > Add Folder to Workspace**

3. Select all the project folders you want to work with.
4. After the workspace loads, **File > Save Workspace As >** Choose a location to save your workspace



## Step 11: Try out MuleSoft Vibes with a Prompt!

1. Select the MuleSoft Vibes icon in the left navigation bar.
2. Click on one of 3 example prompts, i.e. *List all applications running in my 'Sandbox' environment.*
3. Click on Approve to allow the request!

