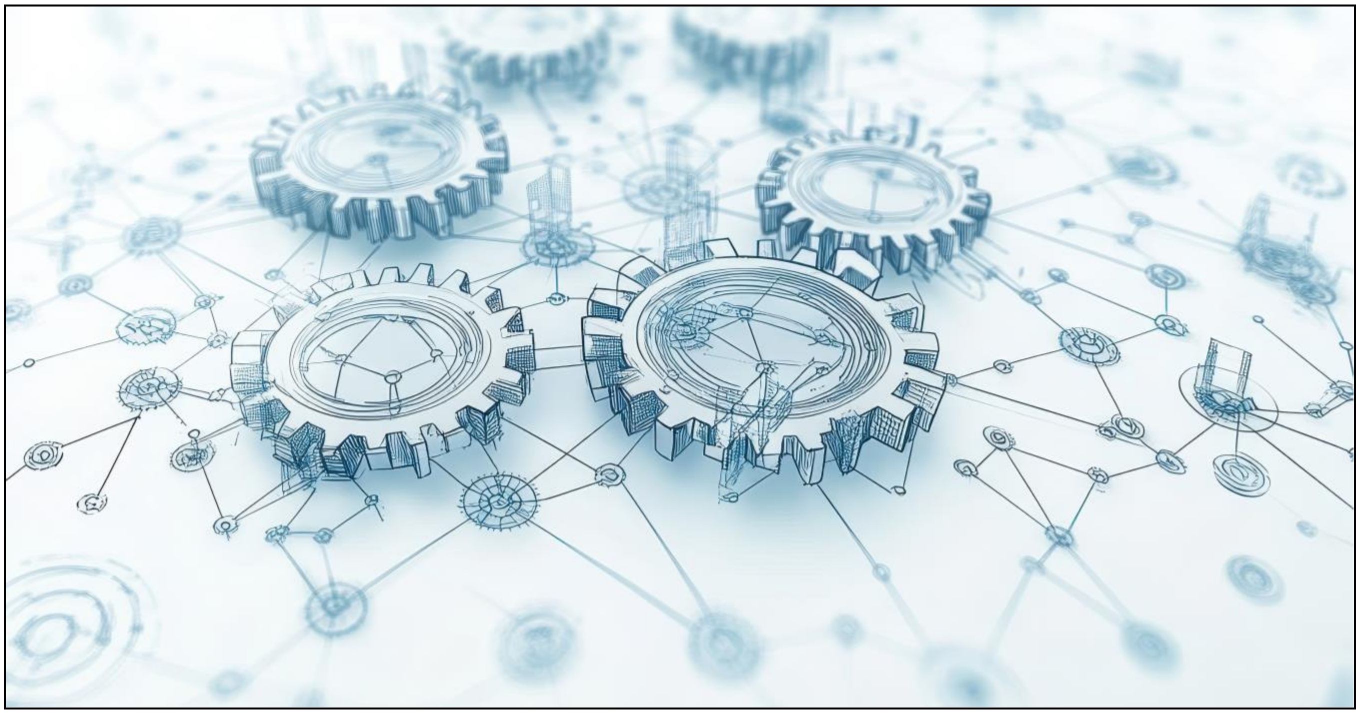
Level 4

Watsonx Orchestrate

Hands-on lab guide: Creating tools and agents



Wolf Bocanett
wolf.bocanett@ibm.com
Senior Customer Success Manager Architect, IBM Technology

Version 1.0

June 2025

Document edition: June 2025

Table of Contents

Introduction

Lab overview

This lab will show you how to create, configure, deploy and run agents that use tools, using the watsonx Orchestrate Agent Development Kit (ADK).

When complete, the agent created will be able to respond to queries leveraging tools that provide the agent with live information from online sources, in this example a weather API.

You will then be able to build on that knowledge in later labs, creating more complex agents using the watsonx Orchestrate Agent Development Kit (ADK).

What you will learn

Once you have completed this lab you will have developed skills enabling you to complete the following tasks:

1. Create tools based on web services using an OpenAPI definition.
2. Create, configure and deploy an agent that uses the tool.
3. Test the agent you have created.
4. Use the CLI to discover agents available in your environment.

How to get support

If you encounter an issue with this lab, you can request assistance through the following channels:

- Review the Troubleshooting appendix in this guide first.
- IBMers can use the #ba-techlcd-support Slack channel.
- If you are an IBM Business Partner and require assistance, please open a support case at IBM Technology Zone Help.

Prerequisites

Make sure the following prerequisites are satisfied before proceeding with this lab:

- You have performed the Preparation Lab guide including the environment validation step

- You have your ADK environment up and running (Orchestrate server started after validating the environment in the preparation guide OR after having performed another ADK related lab)

Note: If for some reason your server is not started correctly, reset your server following the Reset your environment appendix.

The tools created in this lab are using web-services based on the OpenAPI Specification (OAS), basic knowledge of the format is not required but it is useful to improve your own knowledge and to extend agents behavior using tools that leverage web-services.

Extensive OAS resources are available online, this link from the OpenAPI initiative provides further details (https://www.openapis.org/what-is-openapi).

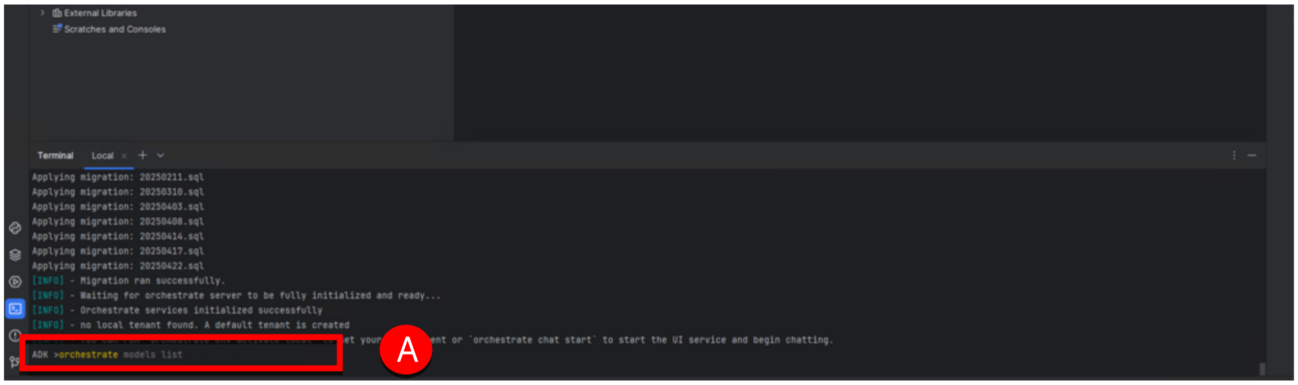
Listing available models using the CLI

In this section you will be using the CLI to view the list of LLM models available to your installation of watsonx Orchestrate ADK.

1. Open the ADK environment and start the watsonx Orchestrate server if not yet done. Refer to the Reset your environment appendix if your orchestrate server is not started.

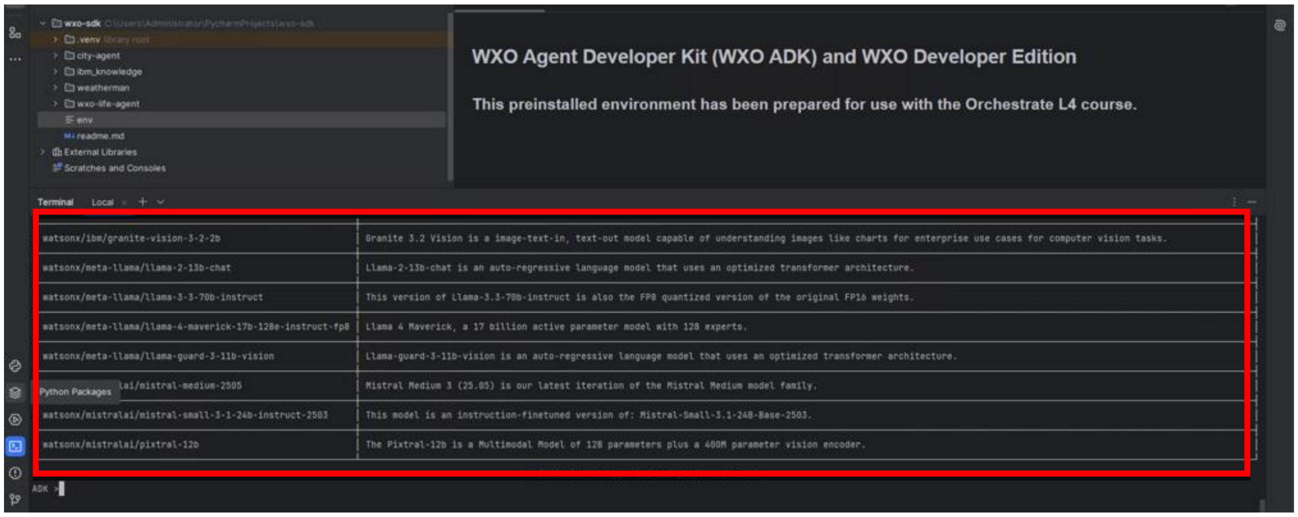
2. To list the LLM models available with watsonx Orchestrate installation SDK run the following command (A):

```
orchestrate models list
```



```
Applying migration: 20250211.sql
Applying migration: 20250310.sql
Applying migration: 20250403.sql
Applying migration: 20250408.sql
Applying migration: 20250414.sql
Applying migration: 20250417.sql
Applying migration: 20250422.sql
[INFO] - Migration ran successfully.
[INFO] - Waiting for orchestrate server to be fully initialized and ready...
[INFO] - Orchestrate services initialized successfully
[INFO] - no local tenant found. A default tenant is created
[illegible]et your [illegible]ent or `orchestrate chat start` to start the UI service and begin chatting.
ADK >orchestrate models list
```

3. The results should resemble the example below.



WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
| watsonx/ibm/granite-vision-3-2-2b                    | Granite 3.2 Vision is a image-text-in, text-out model capable of understanding images like charts for enterprise use cases for computer vision tasks. |
| watsonx/meta-llama/llama-2-13b-chat                  | Llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture. |
| watsonx/meta-llama/llama-3-3-70b-instruct            | This version of Llama-3.3-70b-instruct is also the FP8 quantized version of the original FP16 weights. |
| watsonx/meta-llama/llama-4-maverick-17b-128e-instruct-fp8 | Llama 4 Maverick, a 17 billion active parameter model with 128 experts. |
| watsonx/meta-llama/llama-guard-3-11b-vision          | Llama-guard-3-11b-vision is an auto-regressive language model that uses an optimized transformer architecture. |
| [illegible]lai/mistral-medium-2505                   | Mistral Medium 3 (25.05) is our latest iteration of the Mistral Medium model family. |
| watsonx/mistralai/mistral-small-3-1-24b-instruct-2503 | This model is an instruction-finetuned version of: Mistral-Small-3.1-24B-Base-2503. |
| watsonx/mistralai/pixtral-12b                        | The Pixtral-12b is a Multimodal Model of 12B parameters plus a 400M parameter vision encoder. |
ADK >
```

This output shows all the LLM models that are available in your installation and ready to be used with the agents we will create later.

Each available model identifier is listed on the left column and on the right, there is a description of the model and its characteristics.

Notice that a * mark in front of a model indicates supported and preferred models.

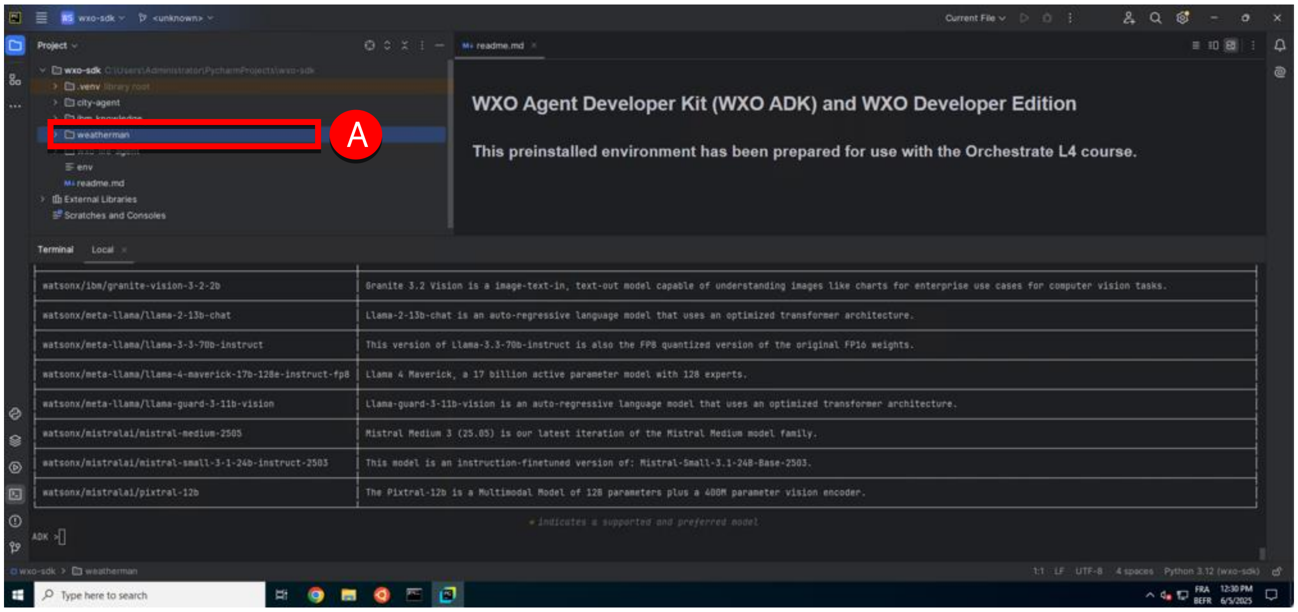
Creating and configuring a new agent

In this section you will create a tool that will enable agents to query for weather information (provided by open-meteo.com) to be used in responses to user queries. This tool will be built by importing the OpenAPI specification for the web service, eliminating the need to write any API-calling code manually. This is a requirement for creating the agent in a later section that will use this tool.

Creating a tool by importing a web-service

In this section we will review the OpenAPI fields required for importing a web-service as a tool in watsonx Orchestrate ADK and create a tool to retrieve weather details from.

1. Double-click the **weatherman** folder (A) to expand it.

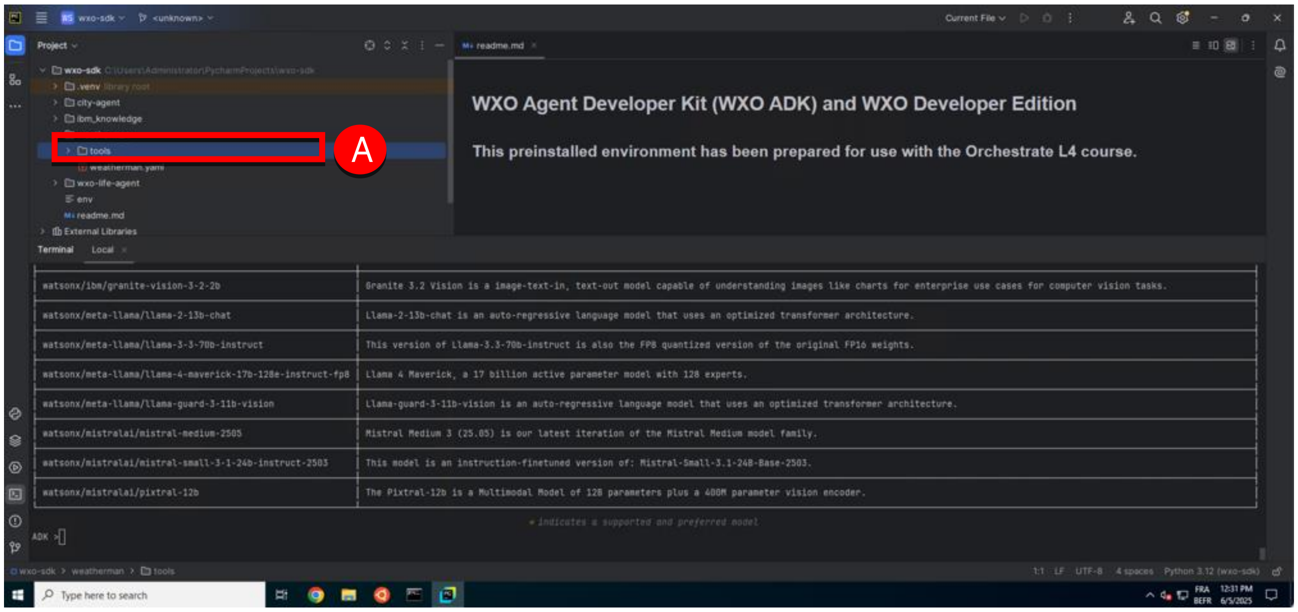


WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
| watsonx/ibm/granite-vision-3-2-2b                        | Granite 3.2 Vision is a image-text-in, text-out model capable of understanding images like charts for enterprise use cases for computer vision tasks. |
| watsonx/meta-llama/llama-2-13b-chat                      | Llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture. |
| watsonx/meta-llama/llama-3-3-70b-instruct                | This version of Llama-3.3-70b-instruct is also the FP8 quantized version of the original FP16 weights. |
| watsonx/meta-llama/llama-4-maverick-17b-128e-instruct-fp8 | Llama 4 Maverick, a 17 billion active parameter model with 128 experts. |
| watsonx/meta-llama/llama-guard-3-11b-vision              | Llama-guard-3-11b-vision is an auto-regressive language model that uses an optimized transformer architecture. |
| watsonx/mistralai/mistral-medium-2505                    | Mistral Medium 3 (25.05) is our latest iteration of the Mistral Medium model family. |
| watsonx/mistralai/mistral-small-3-1-24b-instruct-2503    | This model is an instruction-finetuned version of: Mistral-Small-3.1-24B-Base-2503. |
| watsonx/mistralai/pixtral-12b                            | The Pixtral-12b is a Multimodal Model of 12B parameters plus a 400M parameter vision encoder. |
                         ★ indicates a supported and preferred model
ADK >
```

2. Double-click the **tools** folder (A) to expand it.



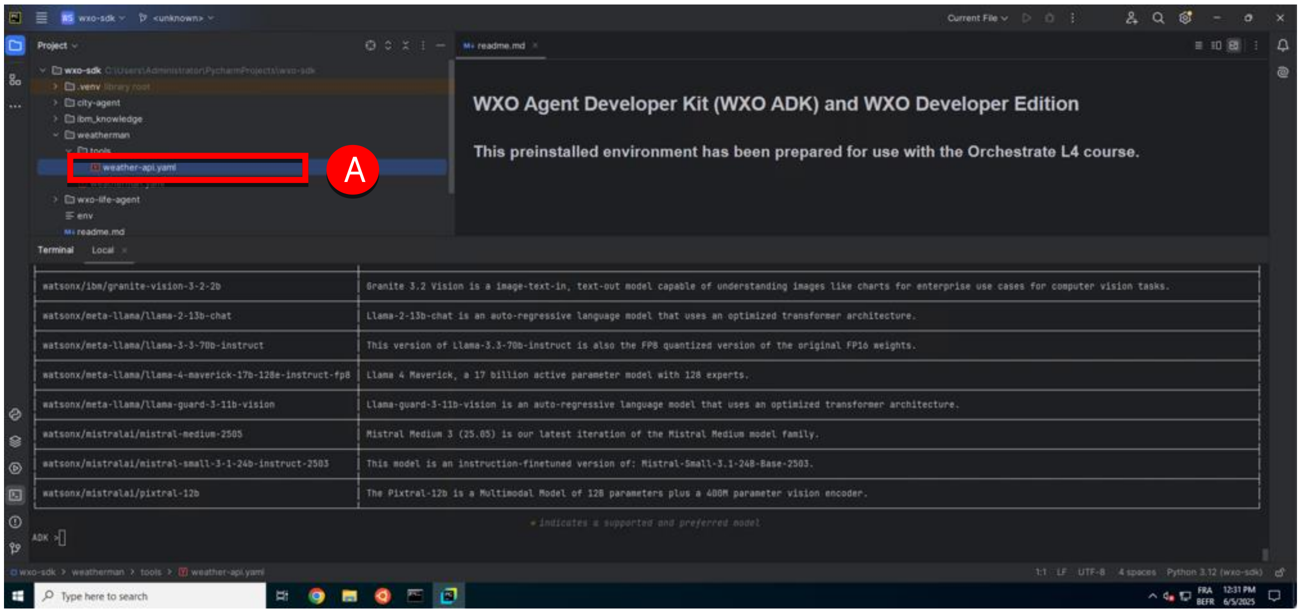
WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
│ watsonx/ibm/granite-vision-3-2-2b                       │ Granite 3.2 Vision is a image-text-in, text-out model capable of understanding images like charts for enterprise use cases for computer vision tasks. │
│ watsonx/meta-llama/llama-2-13b-chat                     │ Llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture. │
│ watsonx/meta-llama/llama-3-3-70b-instruct               │ This version of Llama-3.3-70b-instruct is also the FP8 quantized version of the original FP16 weights. │
│ watsonx/meta-llama/llama-4-maverick-17b-128e-instruct-fp8 │ Llama 4 Maverick, a 17 billion active parameter model with 128 experts. │
│ watsonx/meta-llama/llama-guard-3-11b-vision             │ Llama-guard-3-11b-vision is an auto-regressive language model that uses an optimized transformer architecture. │
│ watsonx/mistralai/mistral-medium-2505                   │ Mistral Medium 3 (25.05) is our latest iteration of the Mistral Medium model family. │
│ watsonx/mistralai/mistral-small-3-1-24b-instruct-2503   │ This model is an instruction-finetuned version of: Mistral-Small-3.1-24B-Base-2503. │
│ watsonx/mistralai/pixtral-12b                           │ The Pixtral-12b is a Multimodal Model of 12B parameters plus a 400M parameter vision encoder. │

                              * indicates a supported and preferred model
ADK >
```

3. Double-click the file **weather-api.yaml** (A) to open it.



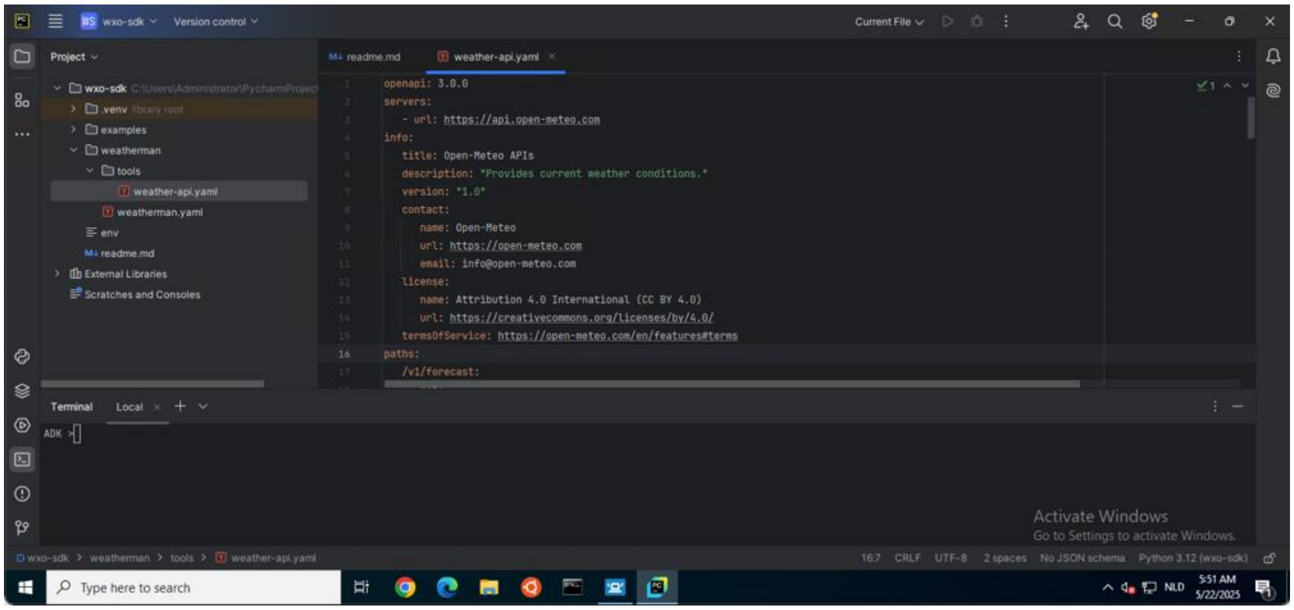
WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
│ watsonx/ibm/granite-vision-3-2-2b                       │ Granite 3.2 Vision is a image-text-in, text-out model capable of understanding images like charts for enterprise use cases for computer vision tasks. │
│ watsonx/meta-llama/llama-2-13b-chat                     │ Llama-2-13b-chat is an auto-regressive language model that uses an optimized transformer architecture. │
│ watsonx/meta-llama/llama-3-3-70b-instruct               │ This version of Llama-3.3-70b-instruct is also the FP8 quantized version of the original FP16 weights. │
│ watsonx/meta-llama/llama-4-maverick-17b-128e-instruct-fp8 │ Llama 4 Maverick, a 17 billion active parameter model with 128 experts. │
│ watsonx/meta-llama/llama-guard-3-11b-vision             │ Llama-guard-3-11b-vision is an auto-regressive language model that uses an optimized transformer architecture. │
│ watsonx/mistralai/mistral-medium-2505                   │ Mistral Medium 3 (25.05) is our latest iteration of the Mistral Medium model family. │
│ watsonx/mistralai/mistral-small-3-1-24b-instruct-2503   │ This model is an instruction-finetuned version of: Mistral-Small-3.1-24B-Base-2503. │
│ watsonx/mistralai/pixtral-12b                           │ The Pixtral-12b is a Multimodal Model of 12B parameters plus a 400M parameter vision encoder. │

                              * indicates a supported and preferred model
ADK >
```

4. Review the contents of the OpenAPI document.



```
1  openapi: 3.0.0
2  servers:
3    - url: https://api.open-meteo.com
4  info:
5    title: Open-Meteo APIs
6    description: "Provides current weather conditions."
7    version: "1.0"
8    contact:
9      name: Open-Meteo
10     url: https://open-meteo.com
11     email: info@open-meteo.com
12   license:
13     name: Attribution 4.0 International (CC BY 4.0)
14     url: https://creativecommons.org/licenses/by/4.0/
15   termsOfService: https://open-meteo.com/en/features#terms
16 paths:
17   /v1/forecast:
```

```
ADK >
```

An OpenAPI document is a machine-readable description of an API, typically written in JSON or YAML format, that outlines the API's endpoints, parameters, responses, and other relevant details. It serves as a formal specification that helps developers understand how to interact with the API and can be used to generate documentation and client libraries.

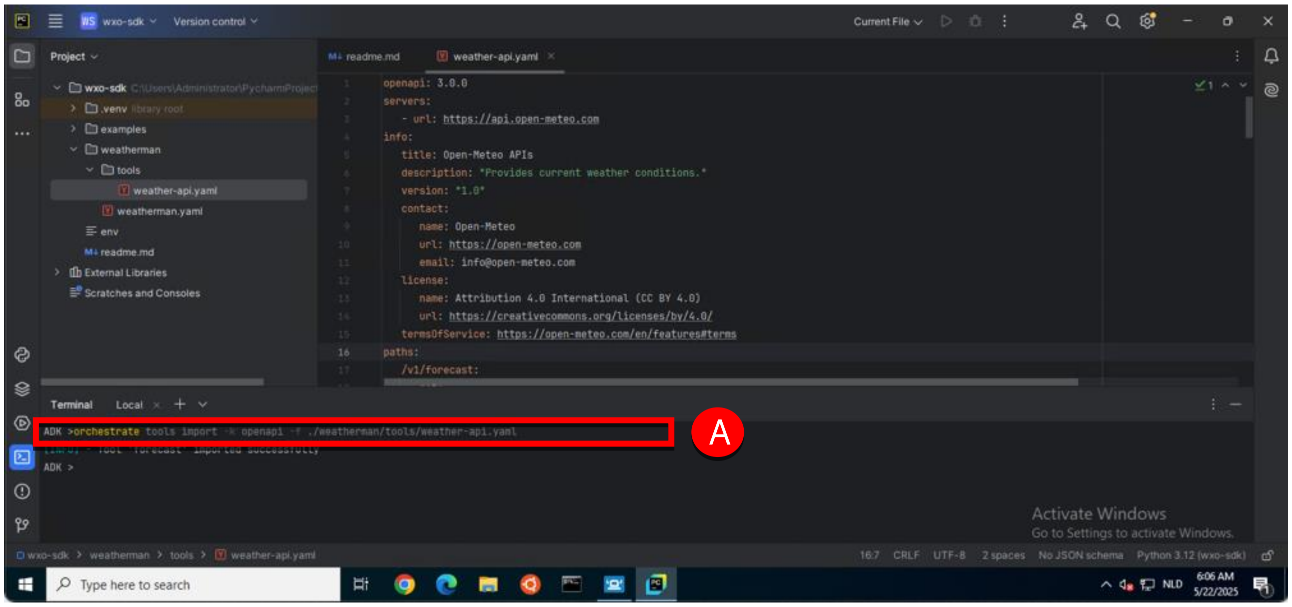
On line 2, the servers element specifies the base URLs for the API, indicating where the API can be accessed. It allows developers to define multiple server URLs, which can be useful for different environments like development, testing, and production.

On line 16 the paths element defines the available endpoints (or resources) of an API and the operations (HTTP methods) that can be performed on those endpoints. Our example only has a GET operation, but each path can be specified with a unique identifier and can support multiple operations like GET, POST, PUT, and DELETE.

On line 19, the operationId provides a logical name for this particular definition, which will be used in the next step, when we're creating a tool from it.

5. Enter the following command (A) into the terminal to create a tool using the API specification:

```
orchestrate tools import -k openapi -f ./weatherman/tools/weather-api.yaml
```



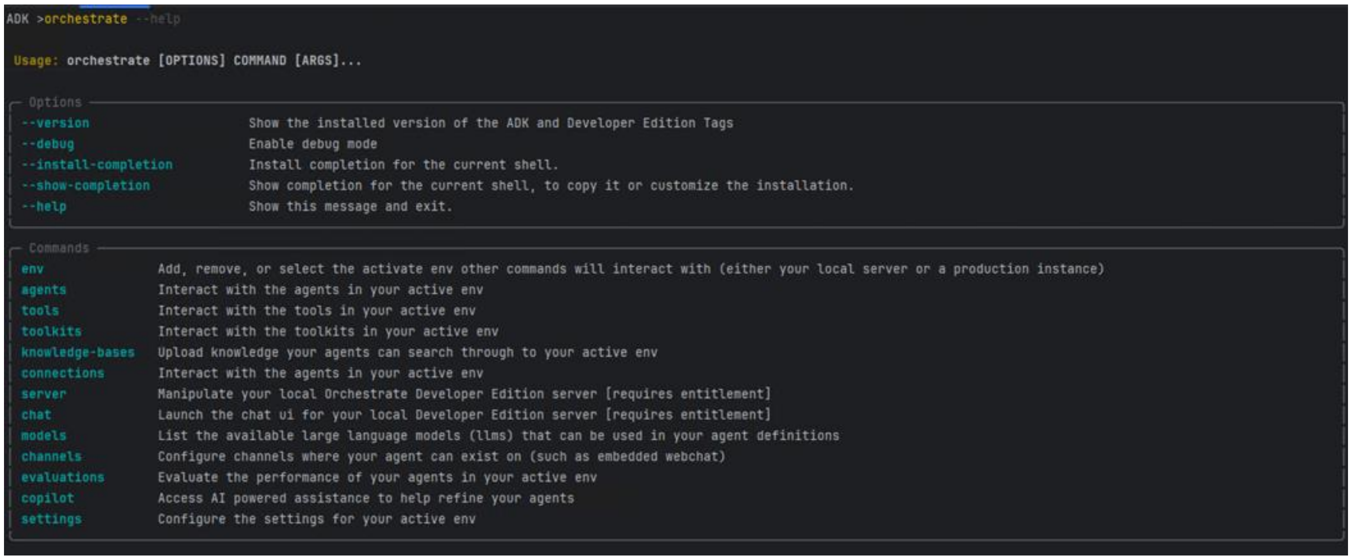
```
openapi: 3.0.0
servers:
  - url: https://api.open-meteo.com
info:
  title: Open-Meteo APIs
  description: "Provides current weather conditions."
  version: "1.0"
  contact:
    name: Open-Meteo
    url: https://open-meteo.com
    email: info@open-meteo.com
  license:
    name: Attribution 4.0 International (CC BY 4.0)
    url: https://creativecommons.org/licenses/by/4.0/
  termsOfService: https://open-meteo.com/en/features#terms
paths:
  /v1/forecast:
```

```
ADK >orchestrate tools import -k openapi -f ./weatherman/tools/weather-api.yaml
[INFO] - Tool 'forecast' imported successfully
ADK >
```

You should see a message that the tool has been successfully imported into your watsonx Orchestrate ADK environment.

Instructions for using the ADK command line interface are provided for each step in this lab. You can enter the command below to explore the CLI commands.

```
orchestrate --help
```



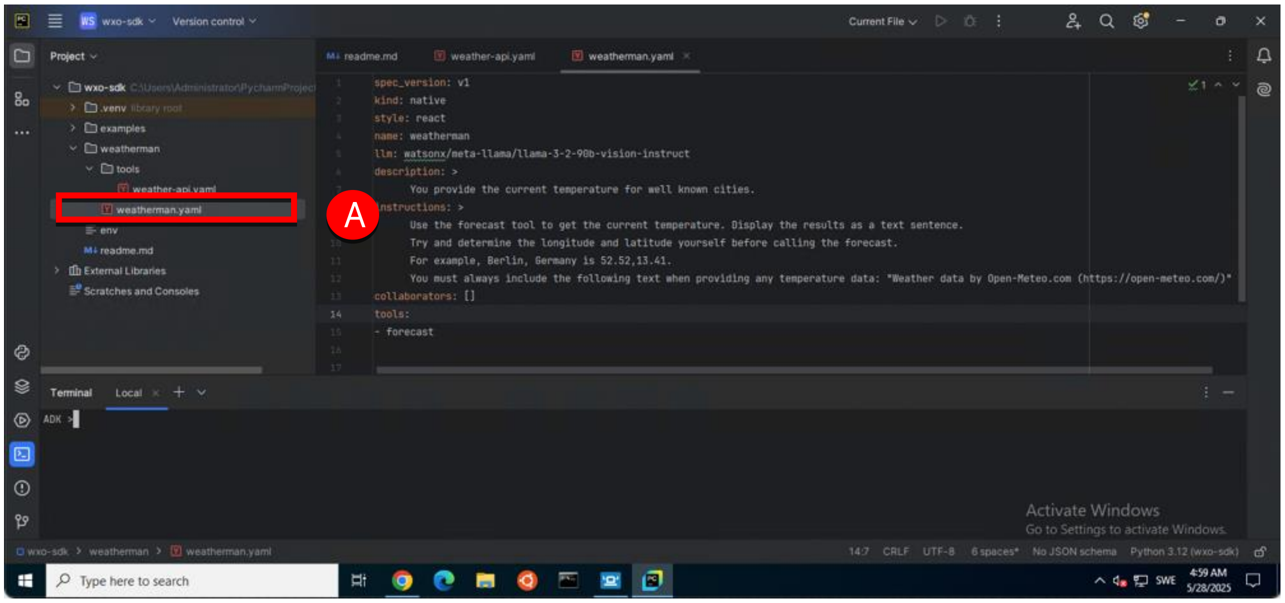
```
ADK >orchestrate --help

 Usage: orchestrate [OPTIONS] COMMAND [ARGS]...

╭─ Options ──────────────────────────────────────────────────────────────────────────────────────────────╮
│ --version                     Show the installed version of the ADK and Developer Edition Tags         │
│ --debug                       Enable debug mode                                                        │
│ --install-completion          Install completion for the current shell.                                │
│ --show-completion             Show completion for the current shell, to copy it or customize the installation. │
│ --help                        Show this message and exit.                                              │
╰────────────────────────────────────────────────────────────────────────────────────────────────────────╯
╭─ Commands ─────────────────────────────────────────────────────────────────────────────────────────────╮
│ env              Add, remove, or select the activate env other commands will interact with (either your local server or a production instance) │
│ agents           Interact with the agents in your active env                                           │
│ tools            Interact with the tools in your active env                                            │
│ toolkits         Interact with the toolkits in your active env                                         │
│ knowledge-bases  Upload knowledge your agents can search through to your active env                    │
│ connections      Interact with the agents in your active env                                           │
│ server           Manipulate your local Orchestrate Developer Edition server [requires entitlement]     │
│ chat             Launch the chat ui for your local Developer Edition server [requires entitlement]     │
│ models           List the available large language models (llms) that can be used in your agent definitions │
│ channels         Configure channels where your agent can exist on (such as embedded webchat)           │
│ evaluations      Evaluate the performance of your agents in your active env                            │
│ copilot          Access AI powered assistance to help refine your agents                               │
│ settings         Configure the settings for your active env                                            │
╰────────────────────────────────────────────────────────────────────────────────────────────────────────╯
```

Agent creation

1. Double-click **weatherman.yaml** file (A) and review the definition of the agent.



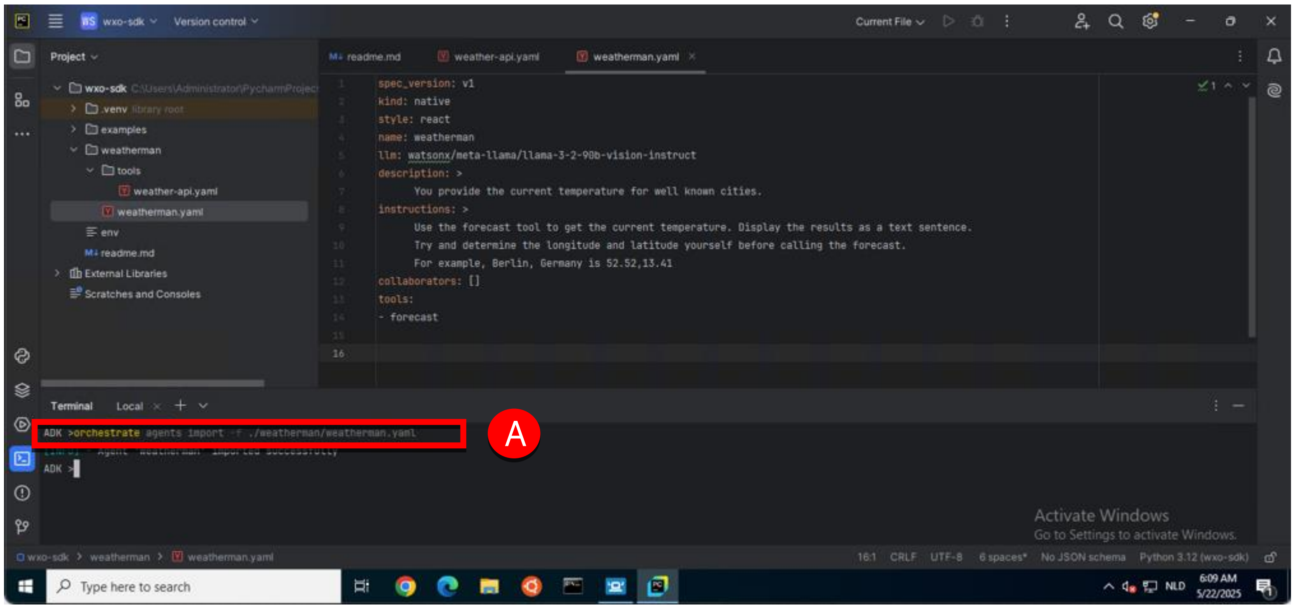
```
1   spec_version: v1
2   kind: native
3   style: react
4   name: weatherman
5   llm: watsonx/meta-llama/llama-3-2-90b-vision-instruct
6   description: >
7       You provide the current temperature for well known cities.
8   instructions: >
9       Use the forecast tool to get the current temperature. Display the results as a text sentence.
10      Try and determine the longitude and latitude yourself before calling the forecast.
11      For example, Berlin, Germany is 52.52,13.41.
12      You must always include the following text when providing any temperature data: "Weather data by Open-Meteo.com (https://open-meteo.com/)"
13  collaborators: []
14  tools:
15  - forecast
```

```
ADK >
```

This YAML file defines the agent using a the ADK specification for agents. It defines the agent description, the instructions which defines the agent's behavior, and the LLM model it will use for planning and reasoning. On line 14, under the tools element there is a reference to the id of the tool we imported in the former step – forecast, which the agent will be able to use.

2. Enter the following command (A) into the terminal to create the agent:

```
orchestrate agents import -f ./weatherman/weatherman.yaml
```



```
spec_version: v1
kind: native
style: react
name: weatherman
llm: watsonx/meta-llama/llama-3-2-90b-vision-instruct
description: >
    You provide the current temperature for well known cities.
instructions: >
    Use the forecast tool to get the current temperature. Display the results as a text sentence.
    Try and determine the longitude and latitude yourself before calling the forecast.
    For example, Berlin, Germany is 52.52,13.41
collaborators: []
tools:
- forecast
```

```
ADK >orchestrate agents import -f ./weatherman/weatherman.yaml
[INFO] - Agent 'weatherman' imported successfully
ADK >
```

You should see a message that the agent has been successfully imported into your watsonx Orchestrate ADK environment.

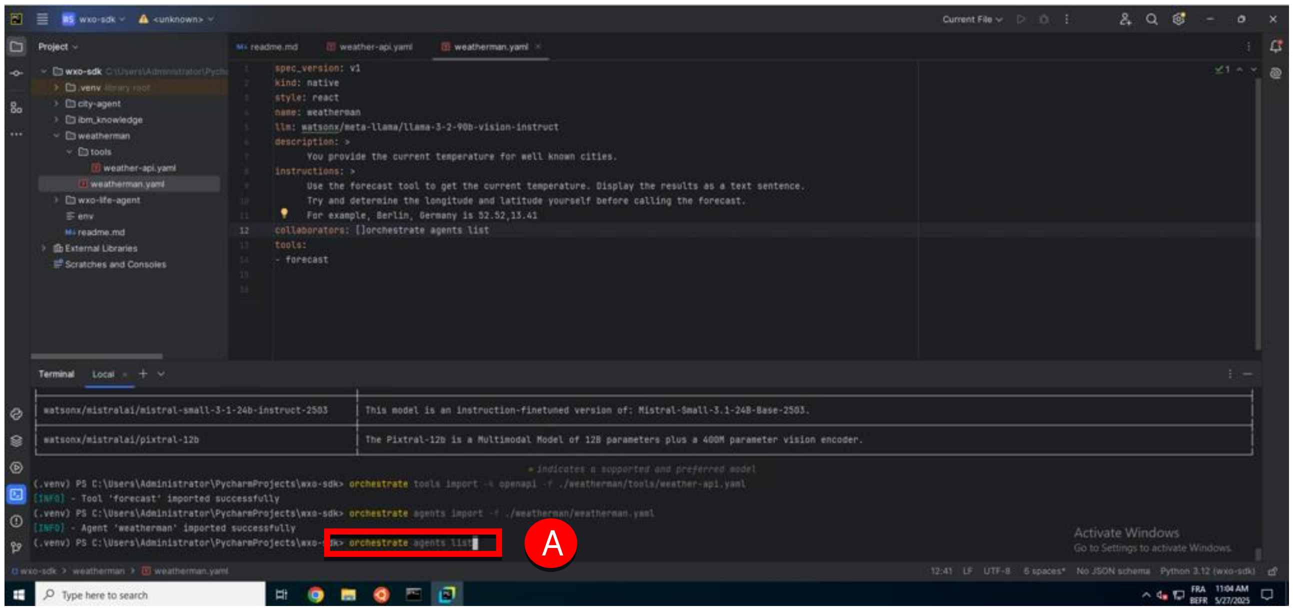
Congratulations, this concludes the creation of your agent.

Listing available agents using the CLI

In this section you will be using the CLI to view the list of agents available in your installation of watsonx Orchestrate ADK.

1. To list agents imported and available within watsonx Orchestrate run the following command (A):

```
orchestrate agents list
```



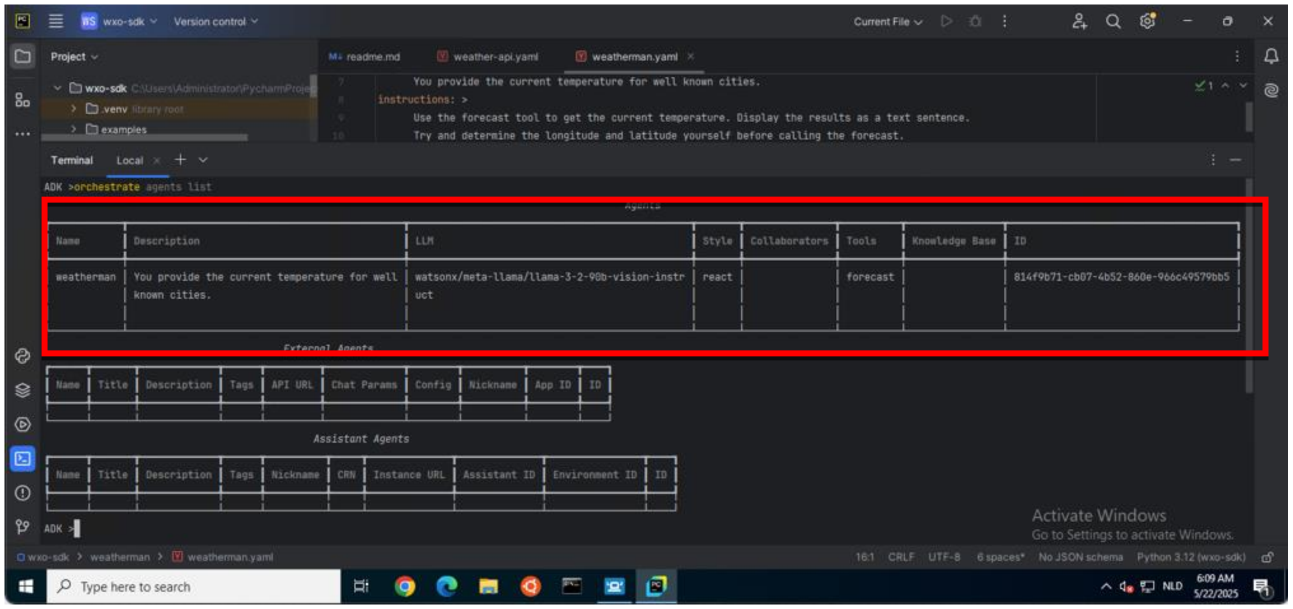
```
spec_version: v1
kind: native
style: react
name: weatherman
llm: watsonx/meta-llama/llama-3-2-90b-vision-instruct
description: >
    You provide the current temperature for well known cities.
instructions: >
    Use the forecast tool to get the current temperature. Display the results as a text sentence.
    Try and determine the longitude and latitude yourself before calling the forecast.
    For example, Berlin, Germany is 52.52,13.41
collaborators: []orchestrate agents list
tools:
- forecast
```

```
| watsonx/mistralai/mistral-small-3-1-24b-instruct-2503 | This model is an instruction-finetuned version of: Mistral-Small-3.1-24B-Base-2503.
| watsonx/mistralai/pixtral-12b                         | The Pixtral-12b is a Multimodal Model of 12B parameters plus a 400M parameter vision encoder.

                                                  * indicates a supported and preferred model
(.venv) PS C:\Users\Administrator\PycharmProjects\wxo-sdk> orchestrate tools import -k openapi -f ./weatherman/tools/weather-api.yaml
[INFO] - Tool 'forecast' imported successfully
(.venv) PS C:\Users\Administrator\PycharmProjects\wxo-sdk> orchestrate agents import -f ./weatherman/weatherman.yaml
[INFO] - Agent 'weatherman' imported successfully
(.venv) PS C:\Users\Administrator\PycharmProjects\wxo-sdk> orchestrate agents list
```

A

2. The results should resemble the example below, showing the **weatherman** agent:



```
ADK >orchestrate agents list
                                        Agents
| Name       | Description                          | LLM                                         | Style | Collaborators | Tools    | Knowledge Base | ID                                   |
| weatherman | You provide the current temperature for well | watsonx/meta-llama/llama-3-2-90b-vision-instr | react |        | forecast |                | 814f9b71-cb07-4b52-860e-966c49579bb5 |
|            | known cities.                        | uct                                         |       |               |          |                |                                      |

                        External Agents
| Name | Title | Description | Tags | API URL | Chat Params | Config | Nickname | App ID | ID |

                        Assistant Agents
| Name | Title | Description | Tags | Nickname | CRN | Instance URL | Assistant ID | Environment ID | ID |

ADK >
```

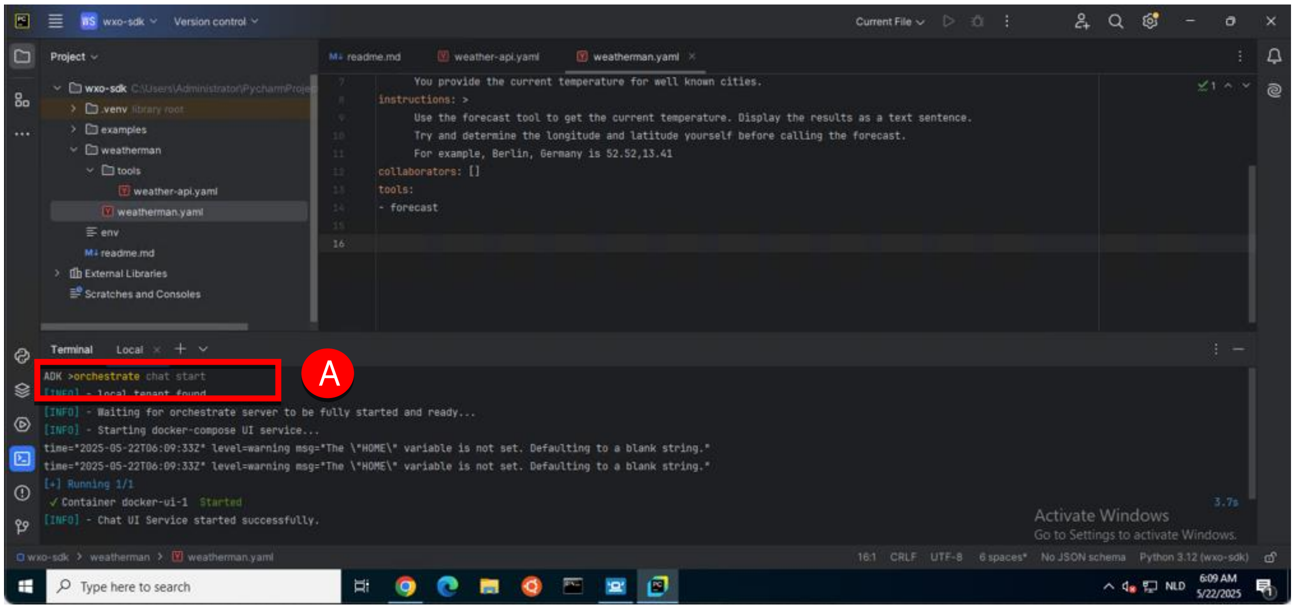
Testing the weatherman agent

In this section you will start the chat interface, test the agent and validate that it's using the tool created earlier to generate responses effectively, enabling the agent to include live weather data within its generated response.

Agent and tool testing

1. Enter the following command into the terminal (A) to start the chat server.

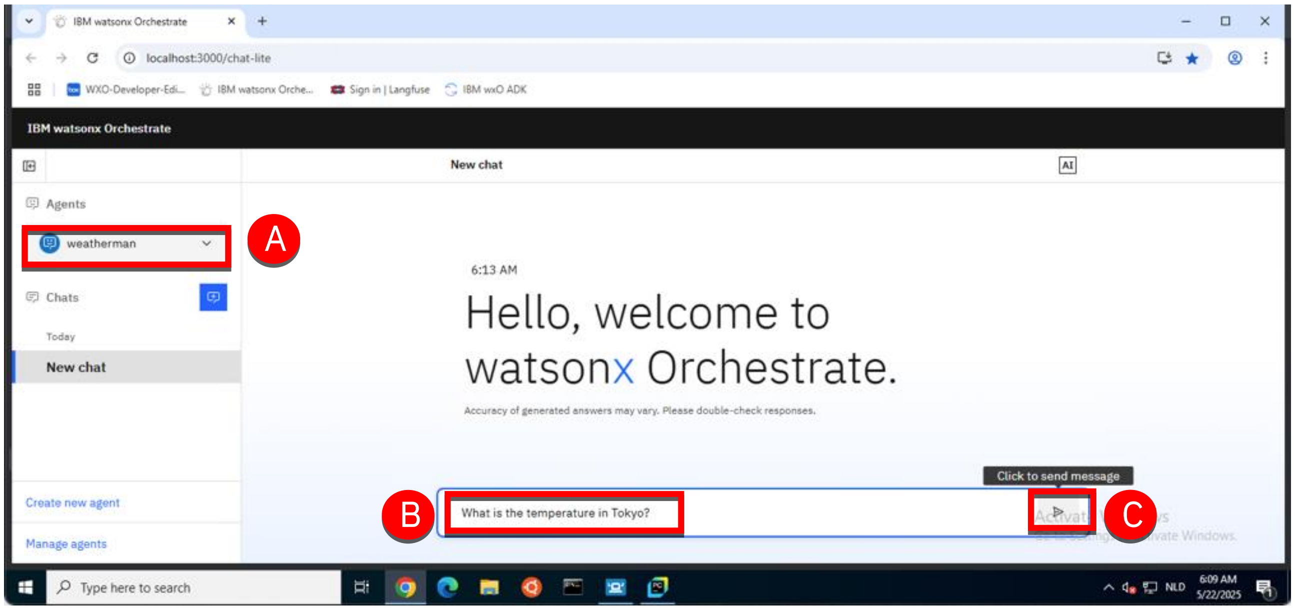
```
orchestrate chat start
```



```
      You provide the current temperature for well known cities.
instructions: >
      Use the forecast tool to get the current temperature. Display the results as a text sentence.
      Try and determine the longitude and latitude yourself before calling the forecast.
      For example, Berlin, Germany is 52.52,13.41
collaborators: []
tools:
- forecast
```

```
ADK >orchestrate chat start
[INFO] - local tenant found
[INFO] - Waiting for orchestrate server to be fully started and ready...
[INFO] - Starting docker-compose UI service...
time="2025-05-22T06:09:33Z" level=warning msg="The \"HOME\" variable is not set. Defaulting to a blank string."
time="2025-05-22T06:09:33Z" level=warning msg="The \"HOME\" variable is not set. Defaulting to a blank string."
[+] Running 1/1
 ✓ Container docker-ui-1  Started                                                                      3.7s
[INFO] - Chat UI Service started successfully.
```

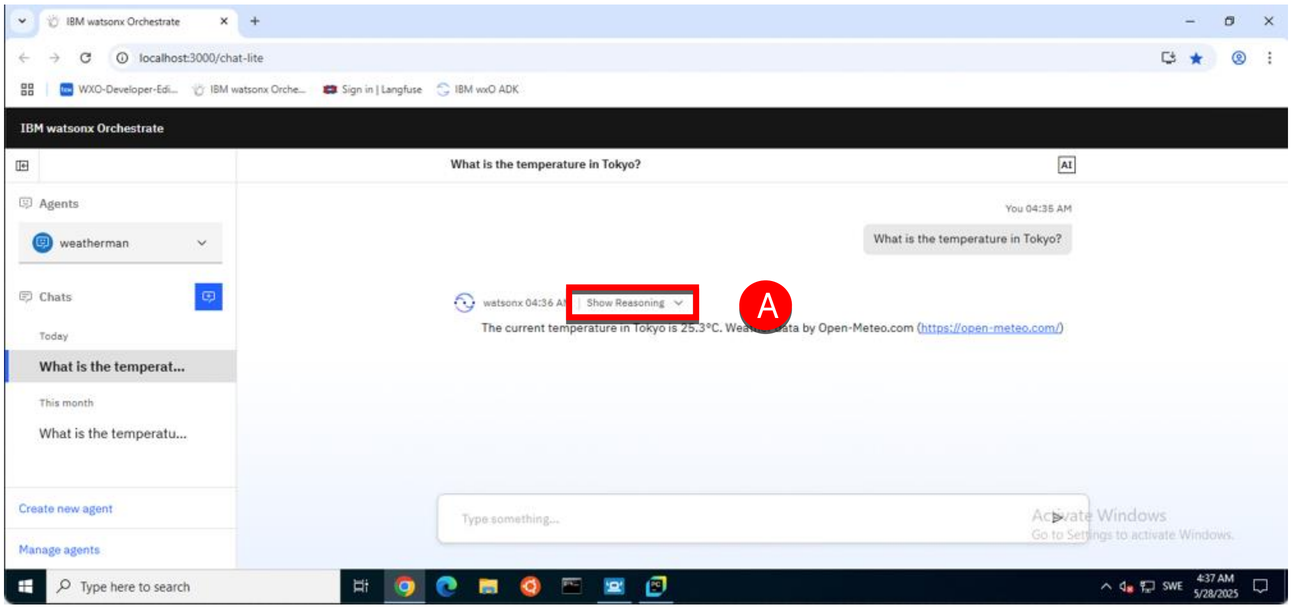
2. Wait for the server to load and when the browser opens, make sure to select the **weatherman** agent in the Agents dropdown (A), then in the chat, enter "`What is the temperature in Tokyo?`" into the chat input (B) and click the submit icon (C).



Wait for the agent to reason about how to perform the action and then wait for the response from the agent. Observe that the agent first began planning & reasoning and then responded to the question using live data from the web-service (service provided by open-meteo.com).

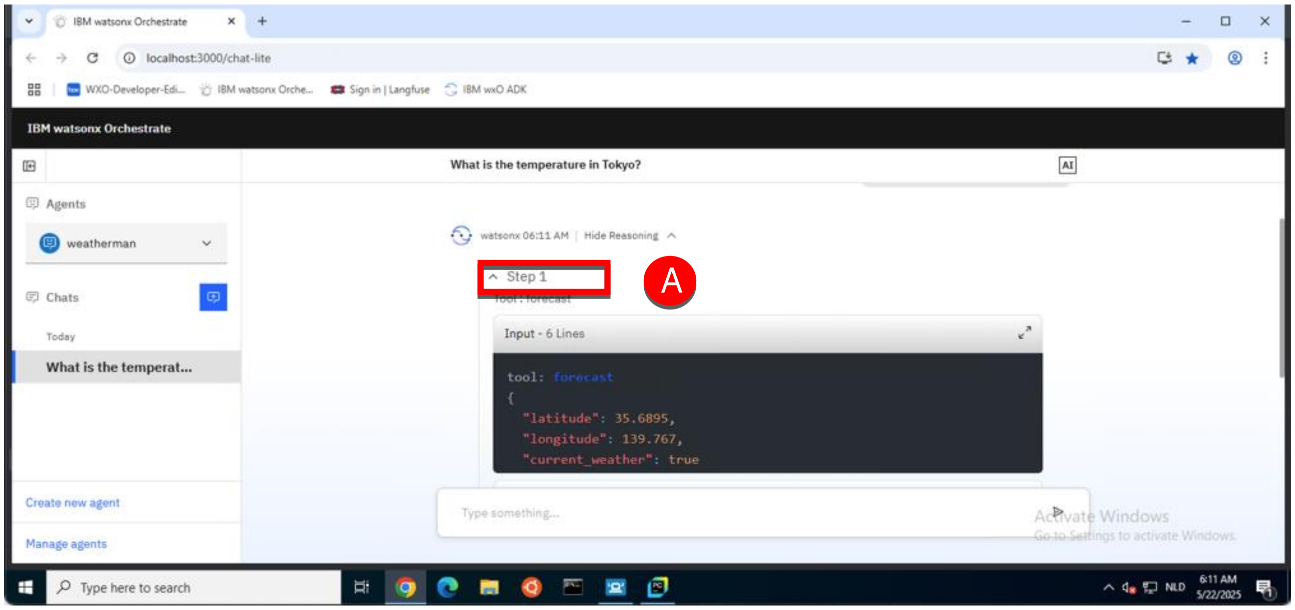
If you encounter an error, refer to the troubleshooting guide at the end of this document.

3. Expand the **Show Reasoning** panel by clicking the expander icon (A).



The current temperature in Tokyo is 25.3°C. Weather data by Open-Meteo.com (https://open-meteo.com/)

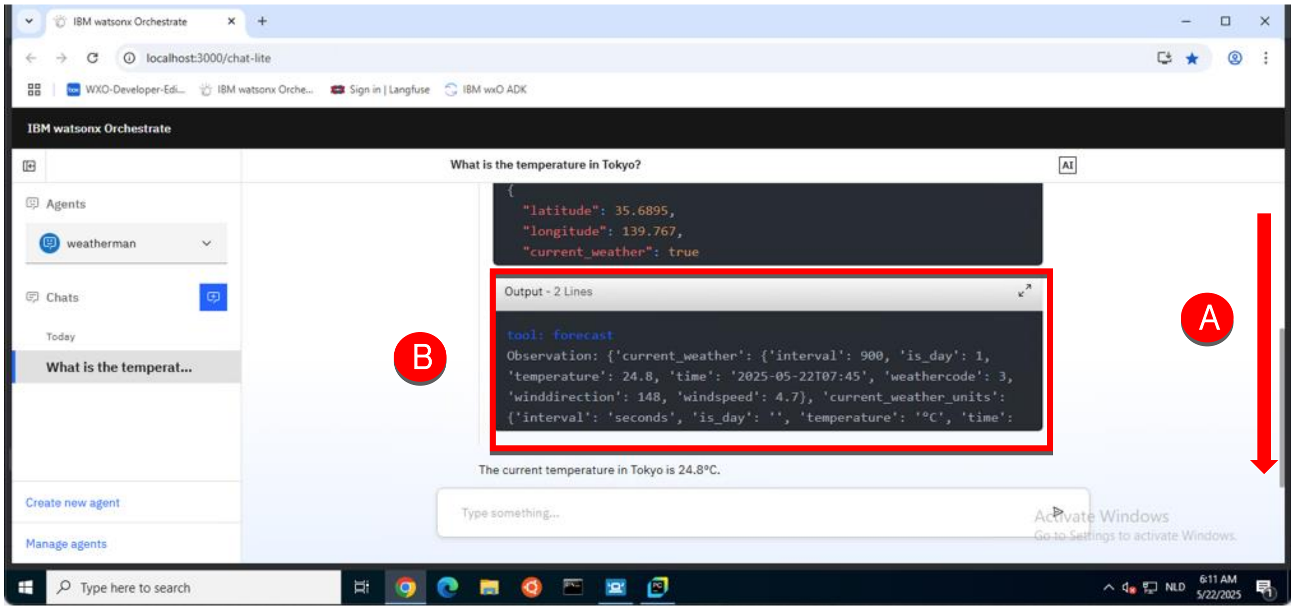
4. Click to expand **Step 1** (A).



```
tool: forecast
{
  "latitude": 35.6895,
  "longitude": 139.767,
  "current_weather": true
```

Observe how the agent understands the format of the information required by the web-service to return the current temperature for the location of Tokyo and formats it according to the OpenAPI specification that was provided earlier. The Llama vision model used by the agent is able to infer the longitude and latitude for Tokyo.

5. Scroll down (A) to the **Output** panel (B):



```
{
  "latitude": 35.6895,
  "longitude": 139.767,
  "current_weather": true
```

```
tool: forecast
Observation: {'current_weather': {'interval': 900, 'is_day': 1,
'temperature': 24.8, 'time': '2025-05-22T07:45', 'weathercode': 3,
'winddirection': 148, 'windspeed': 4.7}, 'current_weather_units':
{'interval': 'seconds', 'is_day': '', 'temperature': '°C', 'time':
```

The current temperature in Tokyo is 24.8°C.

Observe the response from the web-service call and how the agent was able to infer the right field to take temperature result from, generate the response text and format the response for the user.

Summary

In this lab you created a tool by importing an OpenAPI web-service definition and then created an agent that was able to provide live temperature data in response to user queries. The agent was able to adapt to the users request and interpret the information provided from the service and format it as a natural language response using the reasoning power of the LLM.

Next steps

Congratulations. You have completed this watsonx Orchestrate L4 lab. Continue to build on the knowledge acquired in this lab, in the following labs you will be building agents based on knowledge, and more complex agents that use Python tools that collaborate with each other.

Appendix: Troubleshooting

System unresponsive

If your ADK environment becomes unresponsive use the command below to stop all running containers.

```
docker stop $(docker ps -q)
```

Enter the following command into the terminal to reset your server, then press Enter.

```
orchestrate server reset
```

Enter the following command into the terminal to restart your server, then press Enter.

```
orchestrate server start -l -e env
```

Enter the following command into the terminal (A) to start the chat server.

```
orchestrate chat start
```

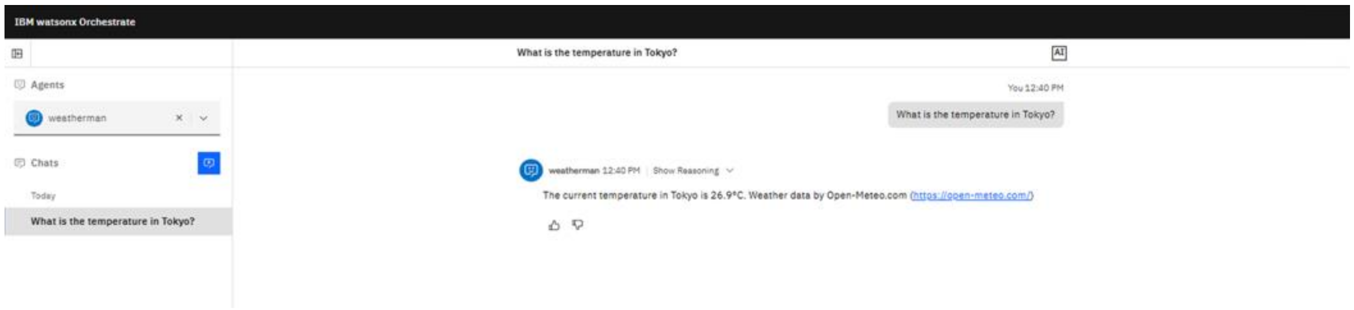
You will need to repeat any commands to create tools and agents.

LLM Error

If you receive an error when using your agent (that worked previously), your API key may have been deleted. You will need to recreate the API key from your watsonx Orchestrate instance, update your env file, and restart the server.

Weatherman error

If your weatherman agent does not respond as expected, click the browser refresh button then click the new chat icon (A) and repeat the question. The LLM used by the agent can sometimes respond unpredictably, especially under heavy load.



What is the temperature in Tokyo?

You 12:40 PM

What is the temperature in Tokyo?

weatherman 12:40 PM

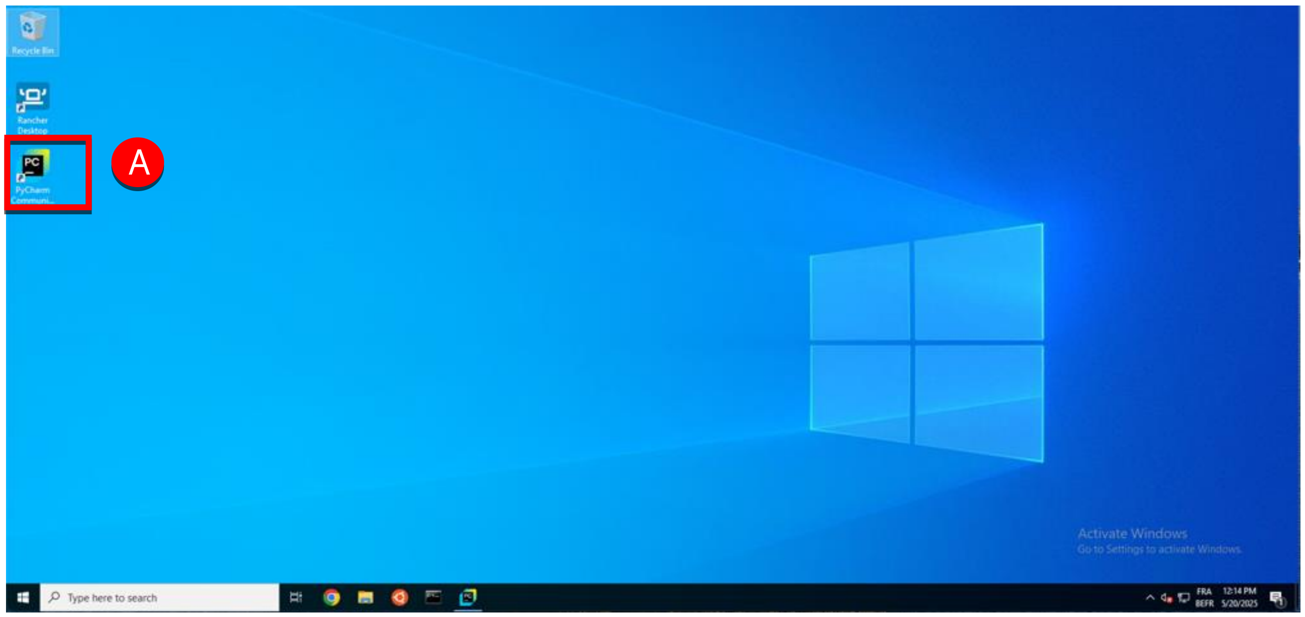
The current temperature in Tokyo is 26.9°C. Weather data by Open-Meteo.com (https://open-meteo.com/)

Appendix: Reset your environment

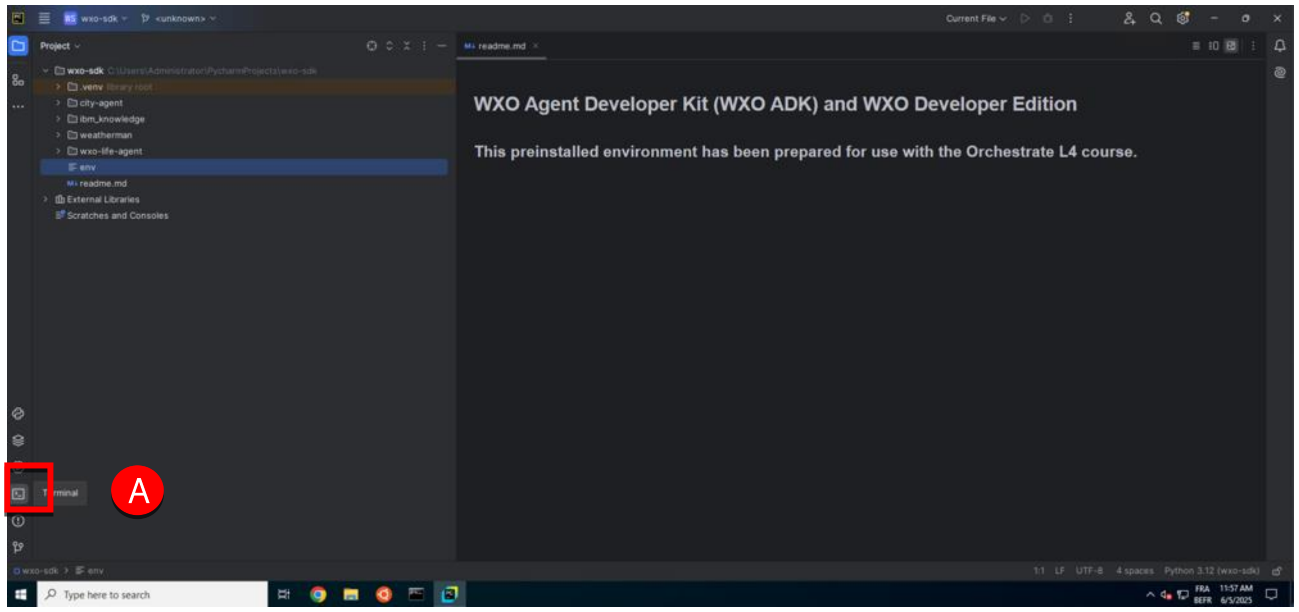
These steps will remove any existing tools and agents from your environment and restart the local watsonx Orchestrate server in readiness to start the lab.

1. Login your VM image containing your ADK (Please refer to the preparation guide for the detailed instructions).

2. Double-click the **PyCharm** icon (A), to open the editor. (PyCharm may be open already if you have used this environment to complete previous labs.)



3. Click the terminal icon (A), to open the terminal (if required).

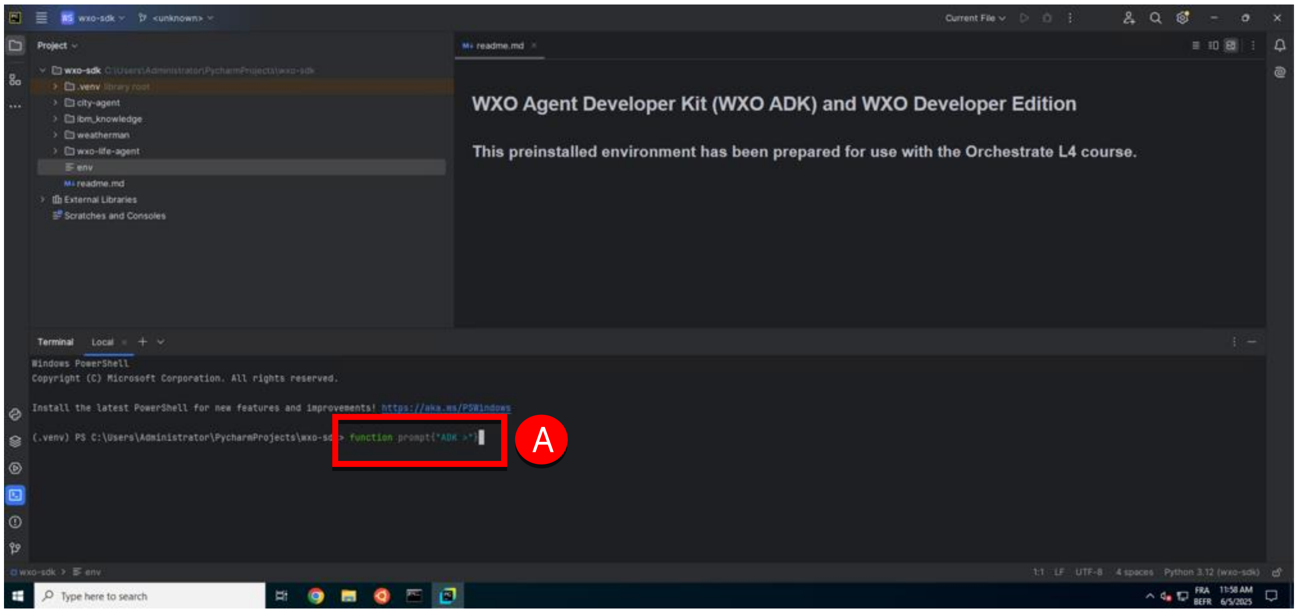


WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

4. Enter the following command into the terminal (A), then press Enter:

```
function prompt{"ADK >"}
```



WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

(.venv) PS C:\Users\Administrator\PycharmProjects\wxo-sdk> function prompt{"ADK >"}
```

This changes the prompt and will improve the readability of commands used in this lab.

5. Enter the following command into the terminal to stop your chat service, then press Enter.

```
orchestrate chat stop
```

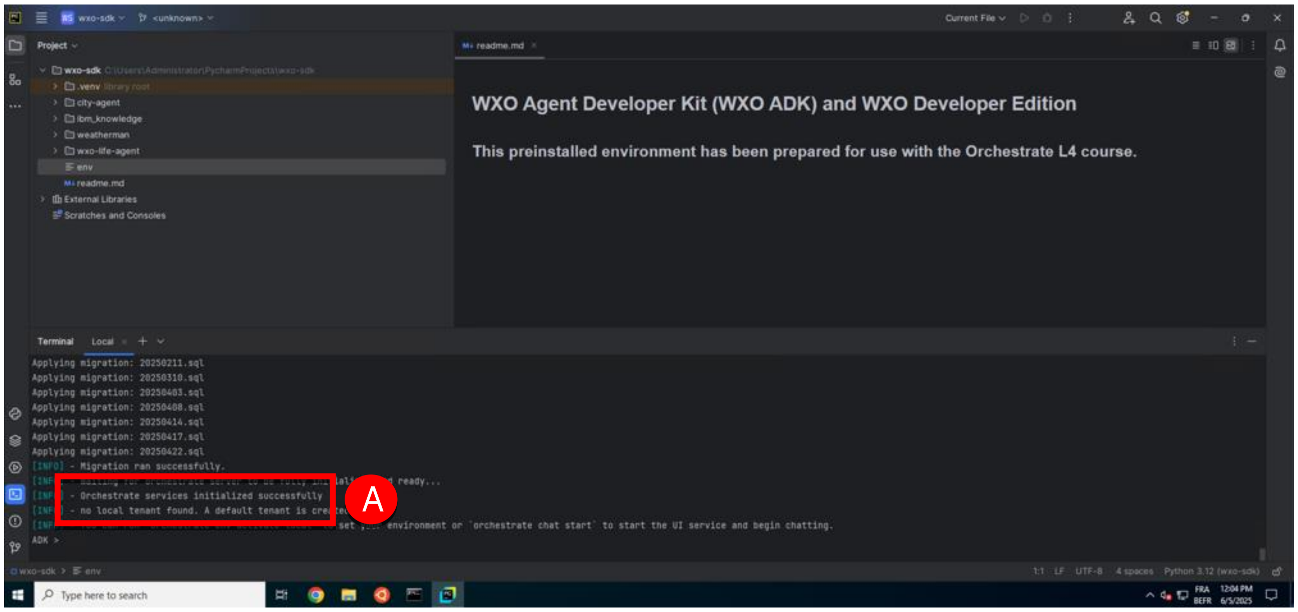
6. Enter the following command into the terminal to reset your server, then press Enter.

```
orchestrate server reset
```

7. Enter the following command into the terminal to restart your server, then press Enter.

```
orchestrate server start -l -e env
```

8. When restart is complete (A) the terminal output will resemble the example below, restart can take 2-3 minutes to complete.

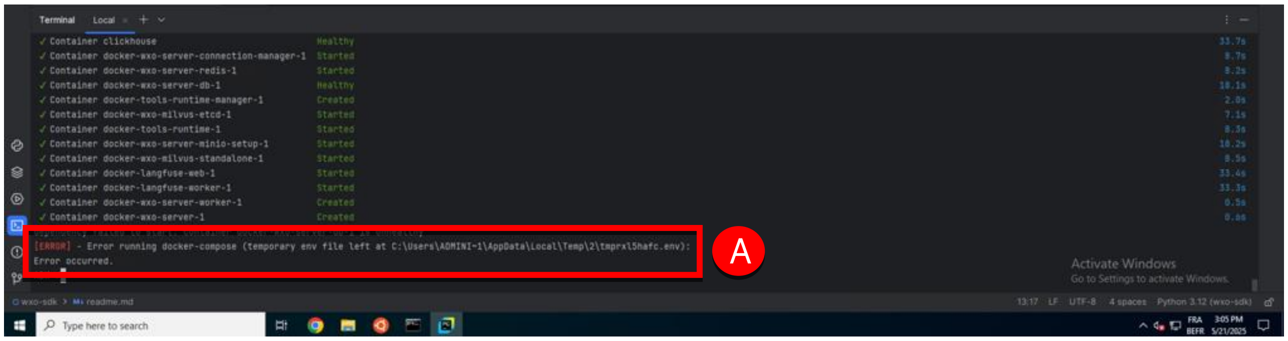


WXO Agent Developer Kit (WXO ADK) and WXO Developer Edition

This preinstalled environment has been prepared for use with the Orchestrate L4 course.

```
Applying migration: 20250211.sql
Applying migration: 20250310.sql
Applying migration: 20250403.sql
Applying migration: 20250408.sql
Applying migration: 20250414.sql
Applying migration: 20250417.sql
Applying migration: 20250422.sql
[INFO] - Migration ran successfully.
[INFO] - Waiting for orchestrate server to be fully initialized and ready...
[INFO] - Orchestrate services initialized successfully
[INFO] - no local tenant found. A default tenant is created
[INFO] - You can run `orchestrate env activate local` to set your environment or `orchestrate chat start` to start the UI service and begin chatting.
ADK >
```

Note: If you encounter an error (A), please repeat the start command as some images require a little extra time to start:



```
✓ Container clickhouse                                   Healthy      33.7s
✓ Container docker-wxo-server-connection-manager-1     Started       8.7s
✓ Container docker-wxo-server-redis-1                  Started       8.2s
✓ Container docker-wxo-server-db-1                     Healthy      18.1s
✓ Container docker-tools-runtime-manager-1             Created       2.0s
✓ Container docker-wxo-milvus-etcd-1                   Started       7.1s
✓ Container docker-tools-runtime-1                     Started       8.3s
✓ Container docker-wxo-server-minio-setup-1            Started      18.2s
✓ Container docker-wxo-milvus-standalone-1             Started       8.5s
✓ Container docker-langfuse-web-1                      Started      33.4s
✓ Container docker-langfuse-worker-1                   Started      33.3s
✓ Container docker-wxo-server-worker-1                 Created       0.5s
✓ Container docker-wxo-server-1                        Created       0.8s
dependency failed to start: container docker-wxo-server-db-1 is unhealthy
[ERROR] - Error running docker-compose (temporary env file left at C:\Users\ADMINI~1\AppData\Local\Temp\2\tmprxl5hafc.env):
Error occurred.
```