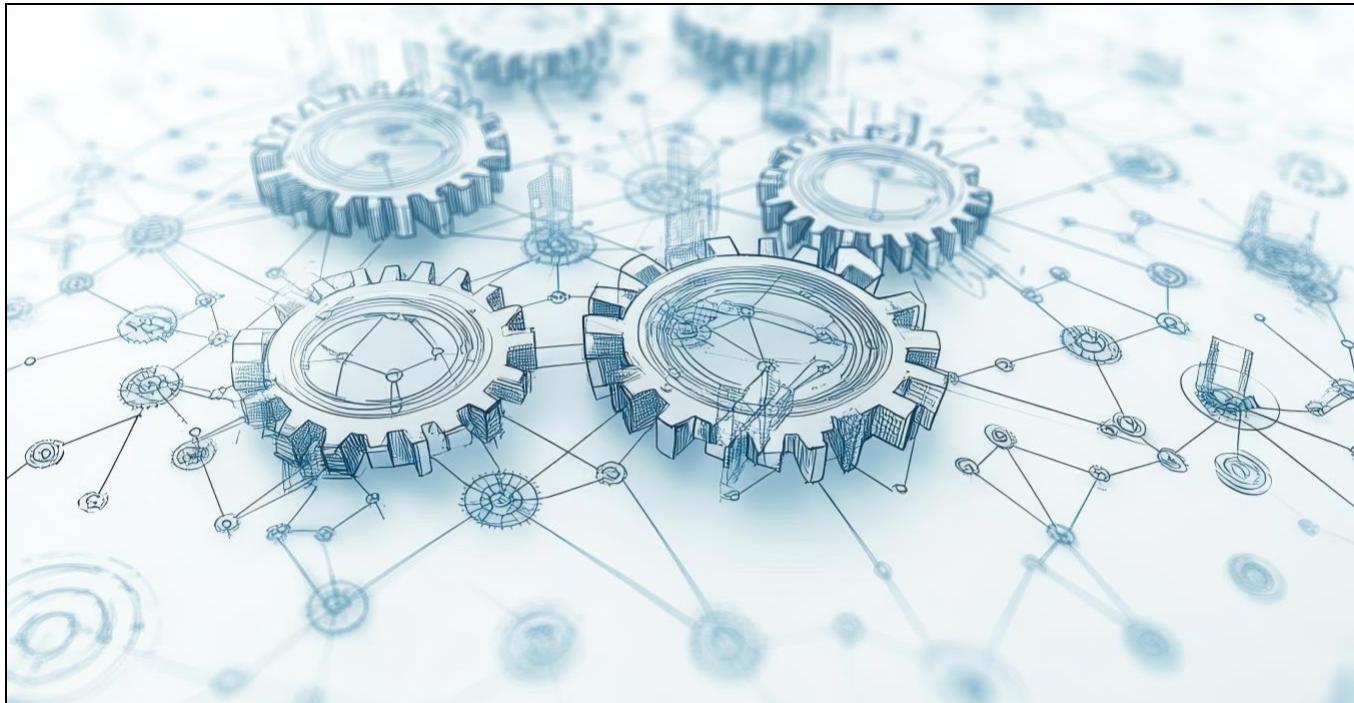


# Level 4

Watsonx Orchestrate

Hands-on lab guide: Creating tools and agents



Wolf Bocanett  
[wolf.bocanett@ibm.com](mailto:wolf.bocanett@ibm.com)  
Senior Customer Success Manager Architect, IBM Technology

Version 1.0

June 2025

## LEGAL NOTICES

This information was developed for products and services offered in the USA. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service. IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing IBM Corporation  
North Castle Drive, MD-NC119 Armonk, NY 10504-1785  
United States of America

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions; therefore, this statement may not apply to you. This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice. Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All names and references for organizations and other business institutions used in this deliverable's scenarios are fictional. Any match with real organizations or institutions is coincidental. All names and associated information for people in this deliverable's scenarios are fictional. Any match with a real person is coincidental.

## TRADEMARKS

IBM, the IBM logo, and [ibm.com](#) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at <https://www.ibm.com/legal/copyright-trademark>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. INFOBLOX is a trademark of Infoblox Inc. or its affiliated companies, registered in the United States and other countries.

Microsoft®, Azure, Windows, Windows NT, Windows Server, Microsoft® Excel®, Microsoft® Word®, Microsoft® Outlook®, Microsoft® Excel® logo, Microsoft® Word® logo, Microsoft® Outlook® logo, Microsoft® Windows App, Microsoft® Windows App logo and the Windows logo are trademarks of Microsoft® Corporation in the United States, other countries, or both.

Salesforce and Salesforce logo are registered trademarks of SALESFORCE, INC. (CORPORATION; CALIFORNIA, USA).

Slack is a registered trademark of Salesforce, Inc (CORPORATION; DELAWARE, USA).

Red Hat®, JBoss®, OpenShift®, Fedora®, Hibernate®, Ansible®, CloudForms®, RHCA®, RHCE®, RHCSA®, Ceph®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Google, Google logo, Google Tasks, Google Cloud Platform, GMAIL, GMAIL logo are registered trademarks of GOOGLE LLC and this material is not endorsed by or affiliated with Google in any way.

Elasticsearch, Elastic search logo, Elastic, Elastic logo are registered trademarks of Elasticsearch BV (LIMITED LIABILITY COMPANY; NETHERLANDS) in United States, other countries or both.

Milvus, Milvus logo are registered trademarks of LF Projects, LLC (LIMITED LIABILITY COMPANY; DELAWARE, USA) in United States, other countries or both.

Meta, Meta logos, Llama, Llama logos are registered trademarks of META PLATFORMS, INC. (CORPORATION; California, USA) in United States, other countries or both.

Mistral, Mistral logo, Mistral AI, Mistral AI logo are registered trademarks of Mistral AI (SOCIÉTÉ PAR ACTIONS SIMPLIFIÉE (SAS); FRANCE) in United States, other countries or both.

PyCharm, PyCharm logo are registered trademarks of JetBrains s.r.o. (CORPORATION; CZECH REPUBLIC) in United States, other countries or both.

Bruno is a trademark held by Anoop M D. Bruno logo is sourced from OpenMoji. License: CC BY-SA 4.0

Apple, Apple logo, Apple store, App store, App store logo are registered trademarks of Apple Inc. (CORPORATION; CALIFORNIA, USA) in United States, other countries or both.

VMWARE, VMWARE logos are registered trademarks of VMware LLC (LIMITED LIABILITY COMPANY; Delaware, USA) in United States, other countries or both.

RANCHER, RANCHER logo, Rancher Desktop, Rancher Desktop logo are registered trademarks of SUSE LLC (LIMITED LIABILITY COMPANY; MASSACHUSETTS, USA) in United States, other countries or both.

Langfuse, Langfuse logo are registered trademarks of Langfuse GmbH (GESELLSCHAFT MIT BESCHRÄNKTER HAFTUNG (GMBH); Germany, Germany) in United States, other countries or both.

OpenMeteo is a registered trademark owned by Patrick Zippenfenig (Individual, Germany) in United States, other countries or both.

© Copyright International Business Machines Corporation 2025. This document may not be reproduced in whole or in part without the prior written permission of IBM.

# Table of Contents

Introduction .....	1
Lab overview.....	1
What you will learn .....	1
How to get support.....	1
Prerequisites .....	1
Listing available models using the CLI.....	3
Creating and configuring a new agent .....	5
Creating a tool by importing a web-service .....	5
Agent creation.....	9
Listing available agents using the CLI.....	11
Testing the weatherman agent.....	12
Agent and tool testing.....	12
Summary.....	15
Next steps .....	16
Appendix: Troubleshooting.....	17
System unresponsive .....	17
LLM Error .....	17
Weatherman error .....	17
Appendix: Reset your environment .....	18

# 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.

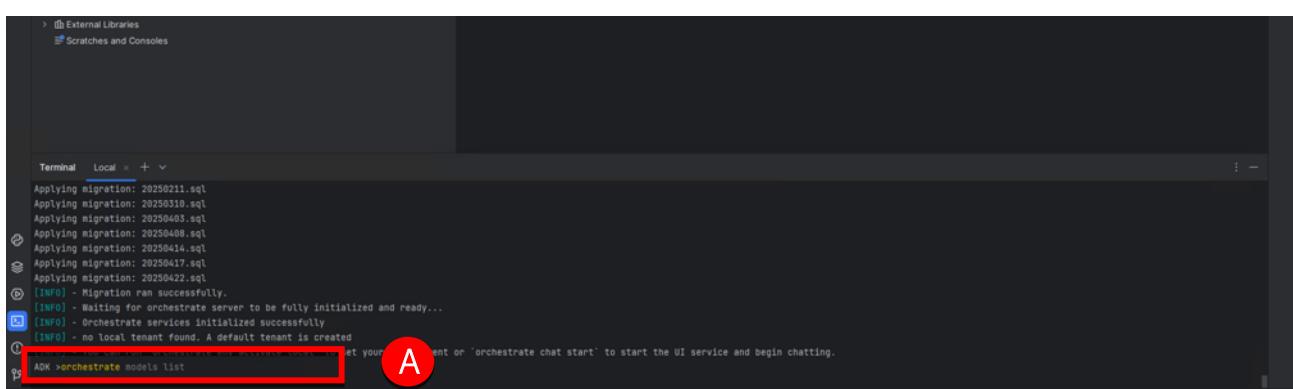
Extensive OAS resources are available online, this link from the OpenAPI initiative provides further details (<https://www.openapis.org/what-is-openapi>).

## 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):

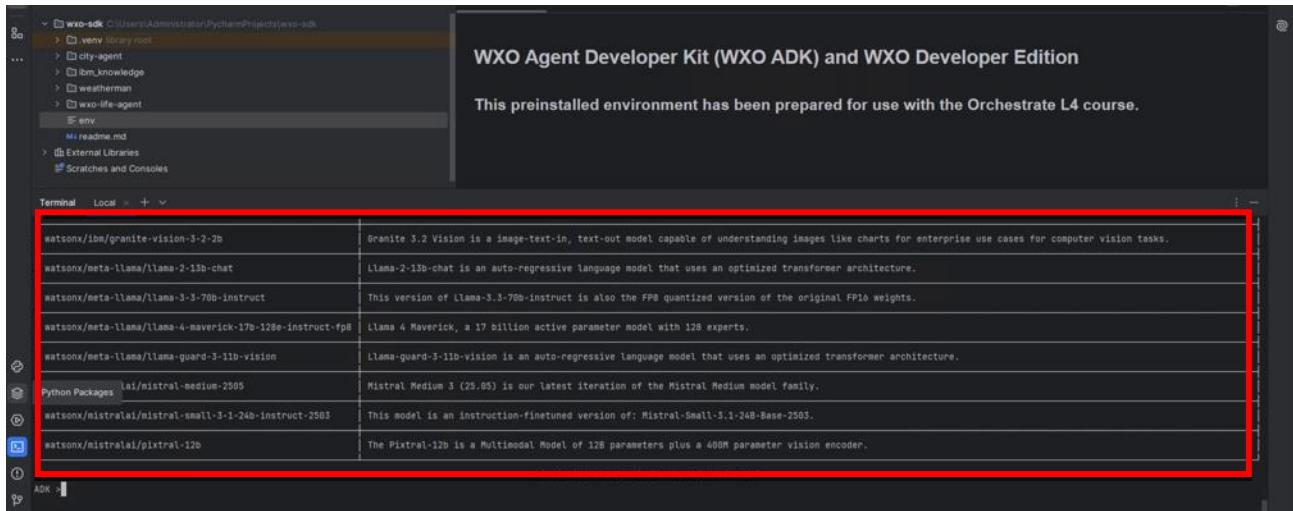
```
orchestrate models list
```



```
External Libraries
Scratches and Consoles

Terminal Local × + v
Applying migration: 20250211.sql
Applying migration: 20250310.sql
Applying migration: 20250405.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] - Orchestrator services initialized successfully
[INFO] - no local tenant found. A default tenant is created
ADK >orchestrate models list
```

3. The results should resemble the example below.



```
External Libraries
Scratches and Consoles

Terminal Local × + v
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.
Python Packages 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.

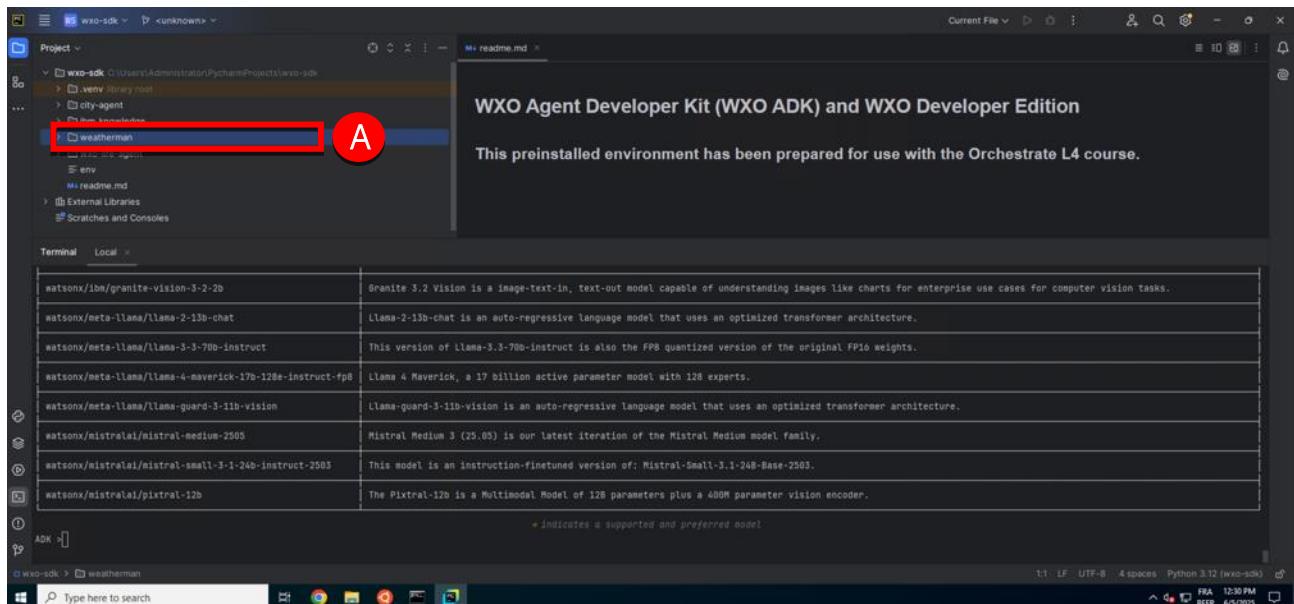
## Creating and configuring a new agent

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.

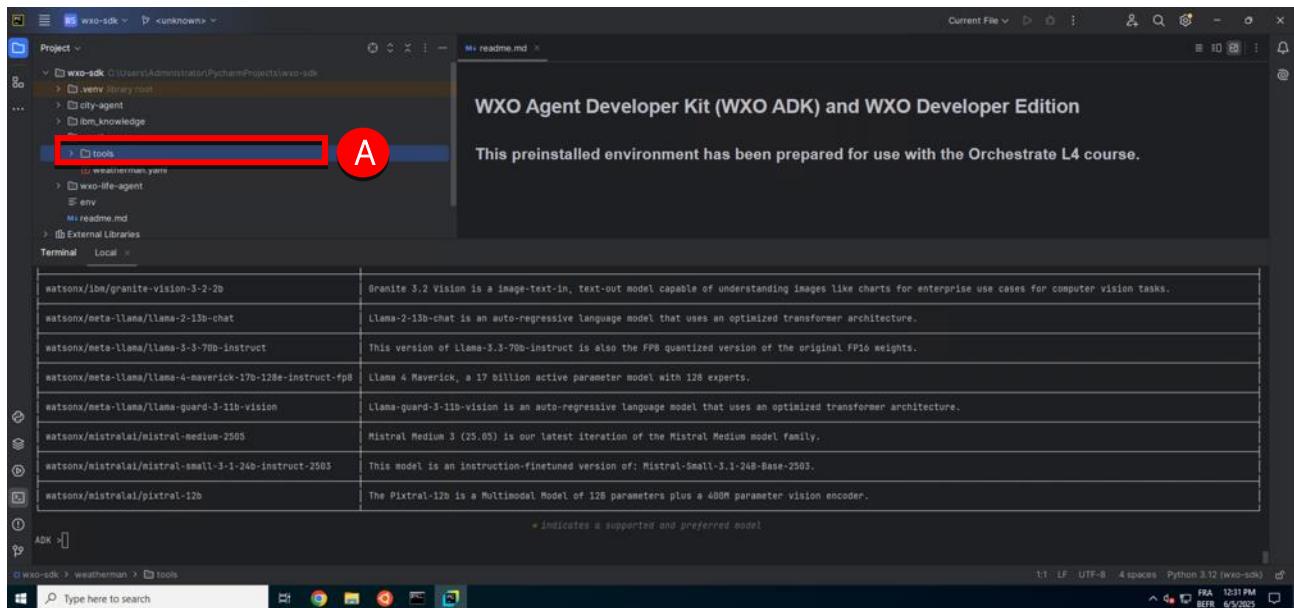
### Creating a tool by importing a web-service

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.

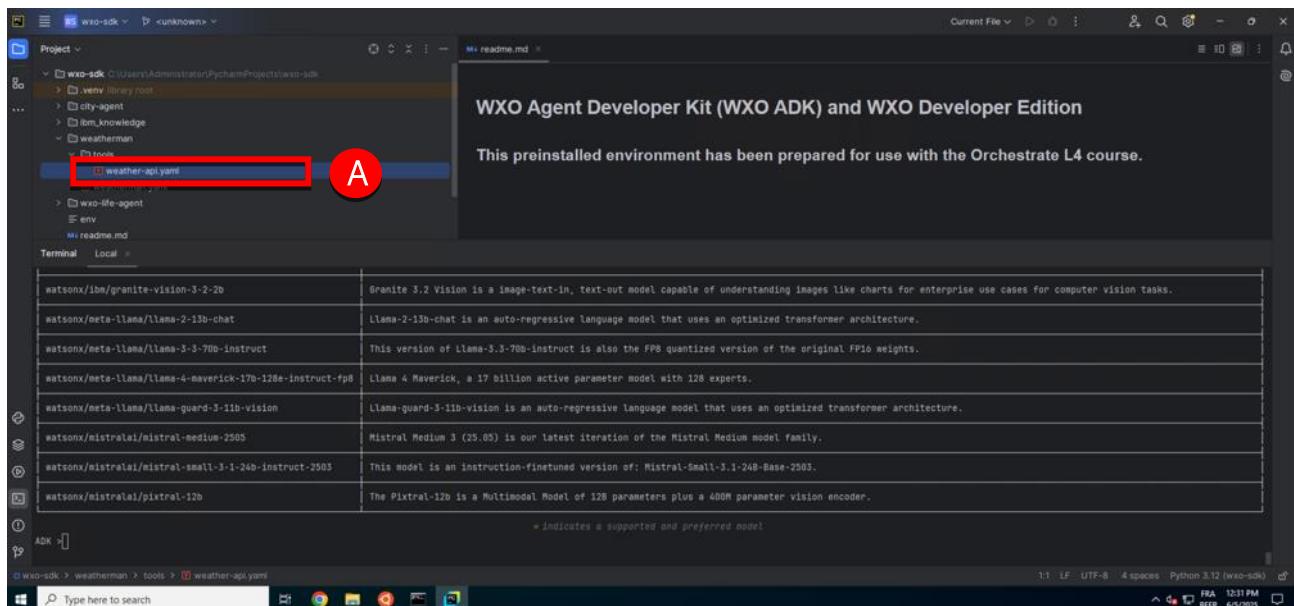
1. Double-click the **weatherman** folder (A) to expand it.



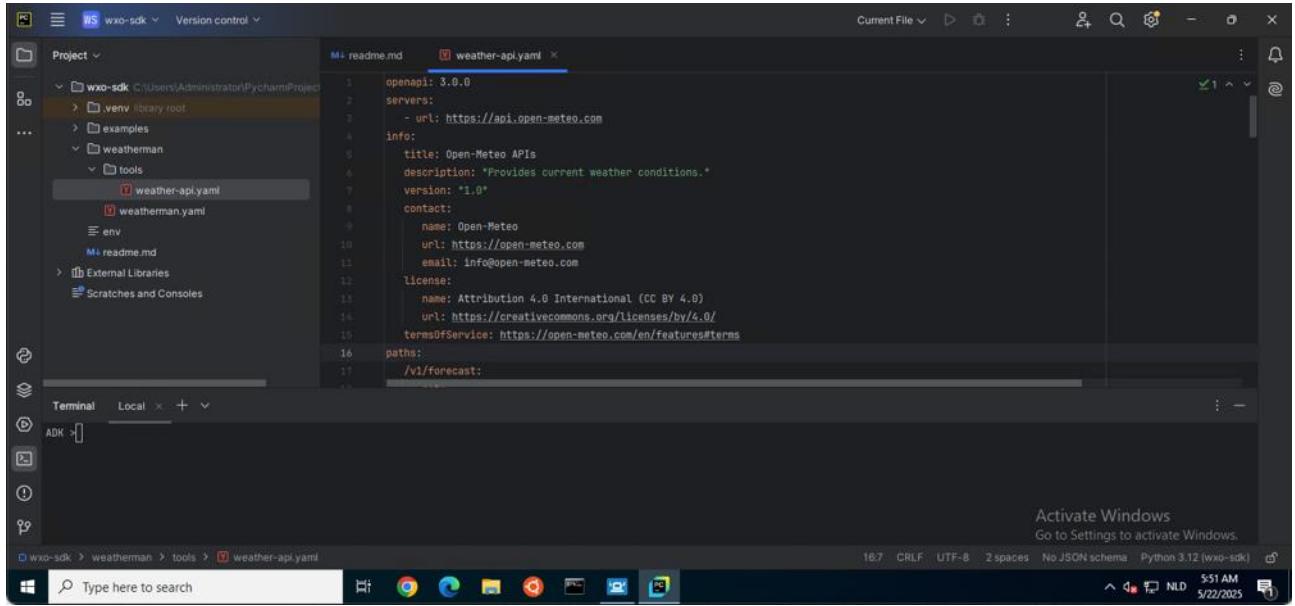
2. Double-click the **tools** folder (A) to expand it.



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



4. Review the contents of the OpenAPI document.



```
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:
```

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.

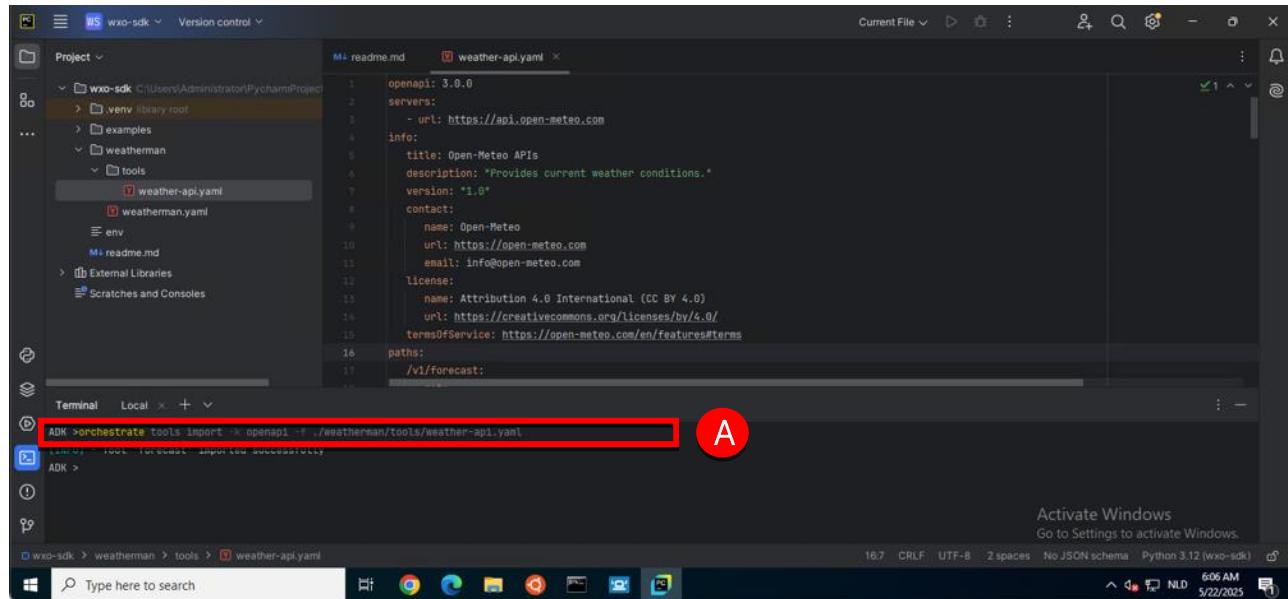
On line 2, the servers element specifies the base URLs for the API, indicating where the API can be accessed. It allows developers to define multiple server URLs, which can be useful for different environments like development, testing, and production.

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
```

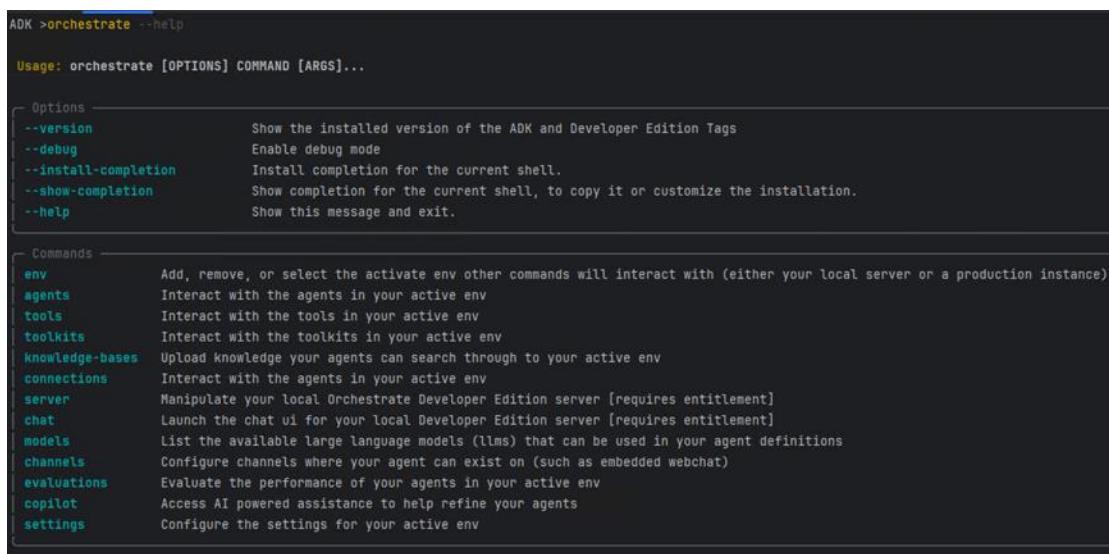


A screenshot of the PyCharm IDE interface. The left sidebar shows a project structure with a file named 'weather-api.yaml' selected. The main editor window displays the contents of this file, which is an OpenAPI specification. Below the editor is a terminal window. A red box highlights the command 'ADK >orchestrate tools import -k openapi -f ./weatherman/tools/weather-api.yaml'. A large red letter 'A' is overlaid on the terminal window. The status bar at the bottom right shows the date and time as '5/22/2025 6:06 AM'.

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
```



A screenshot of a terminal window showing the output of the 'orchestrate --help' command. The output provides usage information and lists various options and commands available in the ADK CLI. The 'Options' section includes '--version', '--debug', '--install-completion', '--show-completion', and '--help'. The 'Commands' section lists 'env', 'agents', 'tools', 'toolkits', 'knowledge-bases', 'connections', 'server', 'chat', 'models', 'channels', 'evaluations', 'copilot', and 'settings'.

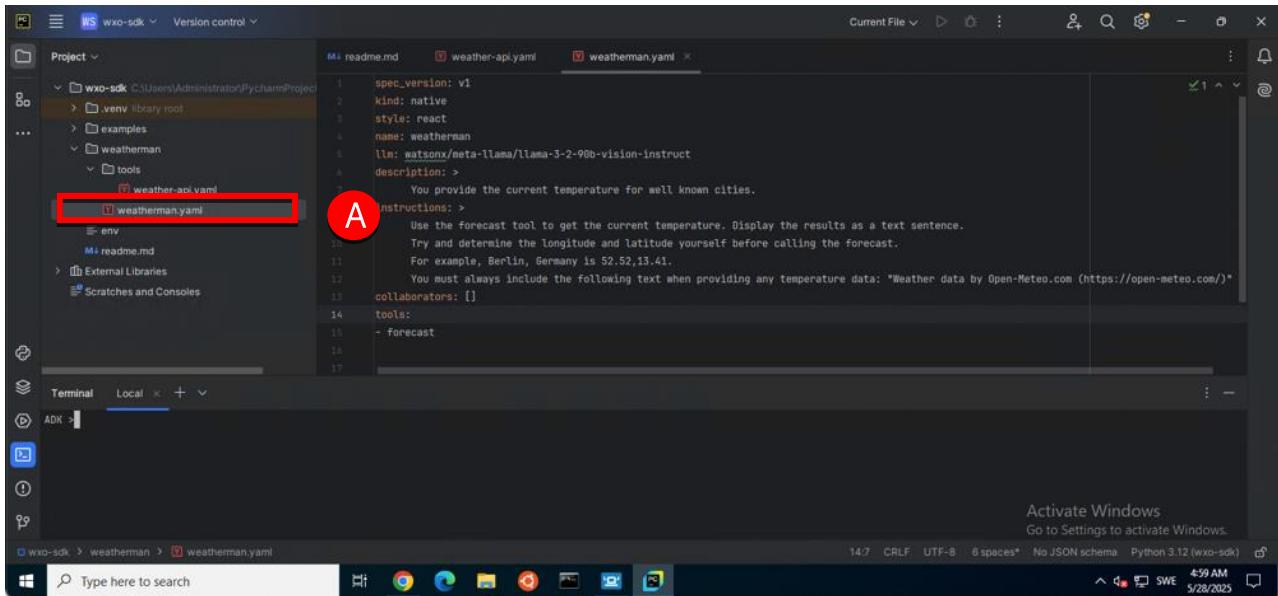
```
ADK >orchestrate --help
Usage: orchestrate [OPTIONS] COMMAND [ARGS]...

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.



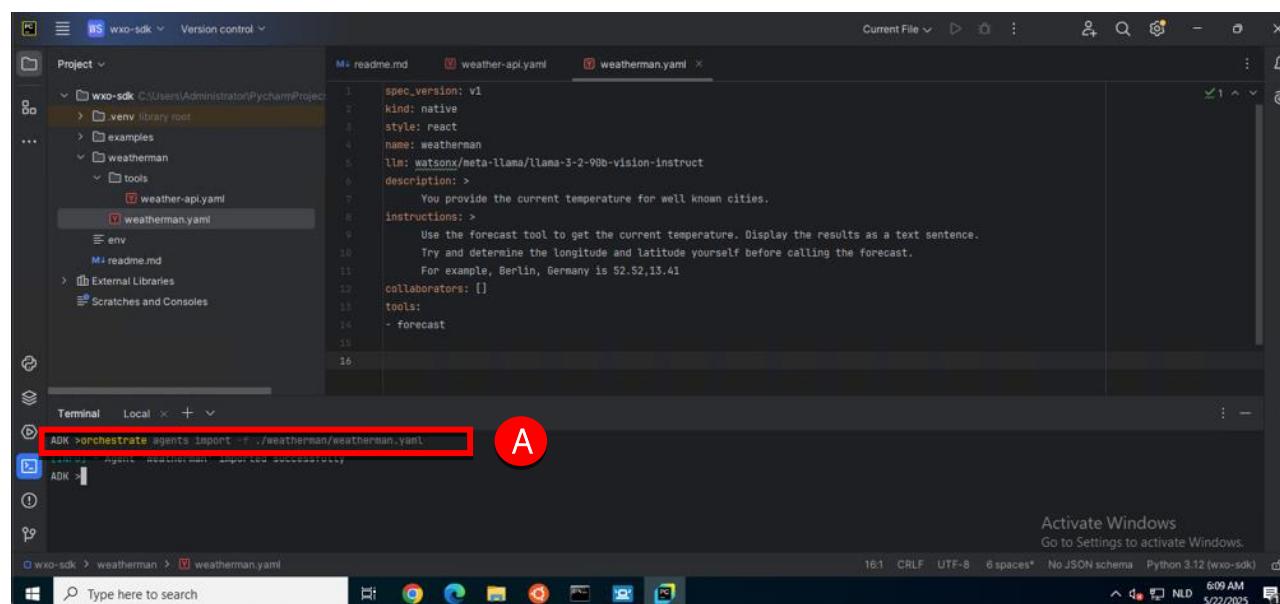
A screenshot of the PyCharm IDE interface. The left sidebar shows a project structure with a file named "weatherman.yaml" highlighted with a red box and labeled with a red letter "A". The main editor window displays the contents of the "weatherman.yaml" file. The terminal at the bottom shows the command "ADK >". The status bar at the bottom right indicates the date and time as "5/28/2025 4:59 AM".

```
spec_version: v1
kind: native
style: react
name: weatherman
llm: watsonx/meta-llama/llama-3-2-900-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.
    You must always include the following text when providing any temperature data: "Weather data by Open-Meteo.com (https://open-meteo.com)"
collaborators: []
tools:
- forecast
```

This YAML file defines the agent using 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
```



A screenshot of the PyCharm IDE interface. The left sidebar shows a project structure with a file named "weatherman.yaml" highlighted with a red box and labeled with a red letter "A". The terminal at the bottom shows the command "ADK >orchestrate agents import -f ./weatherman/weatherman.yaml". The status bar at the bottom right indicates the date and time as "5/28/2025 4:59 AM".

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

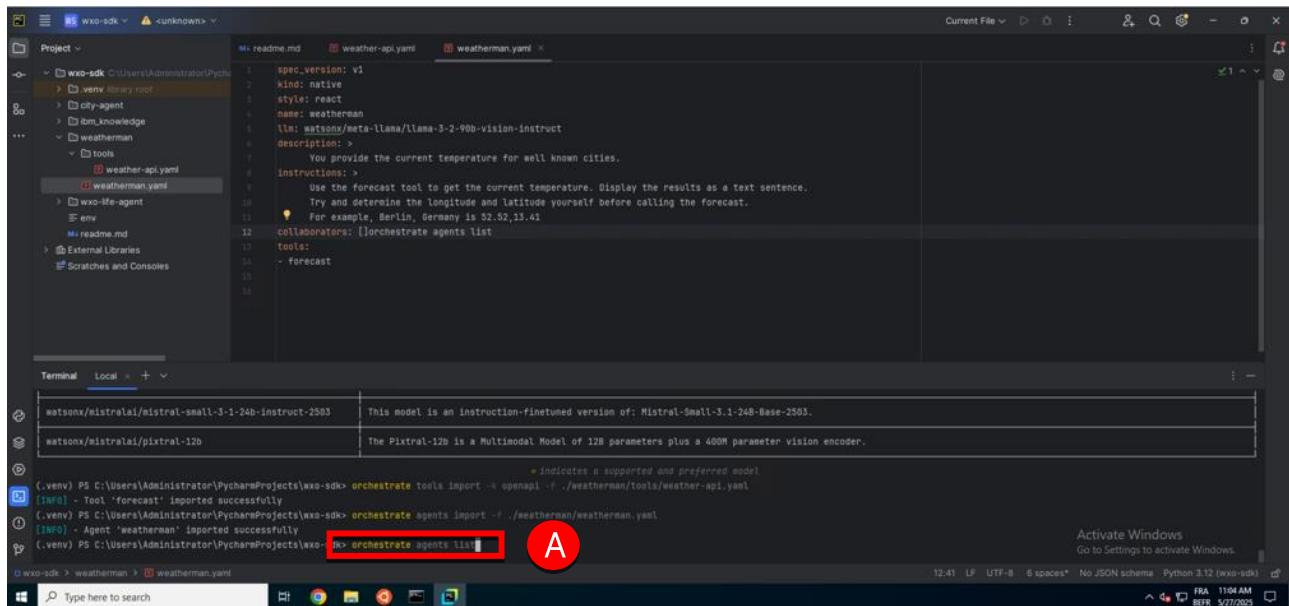
Congratulations, this concludes the creation of your agent.

## Listing available agents using the CLI

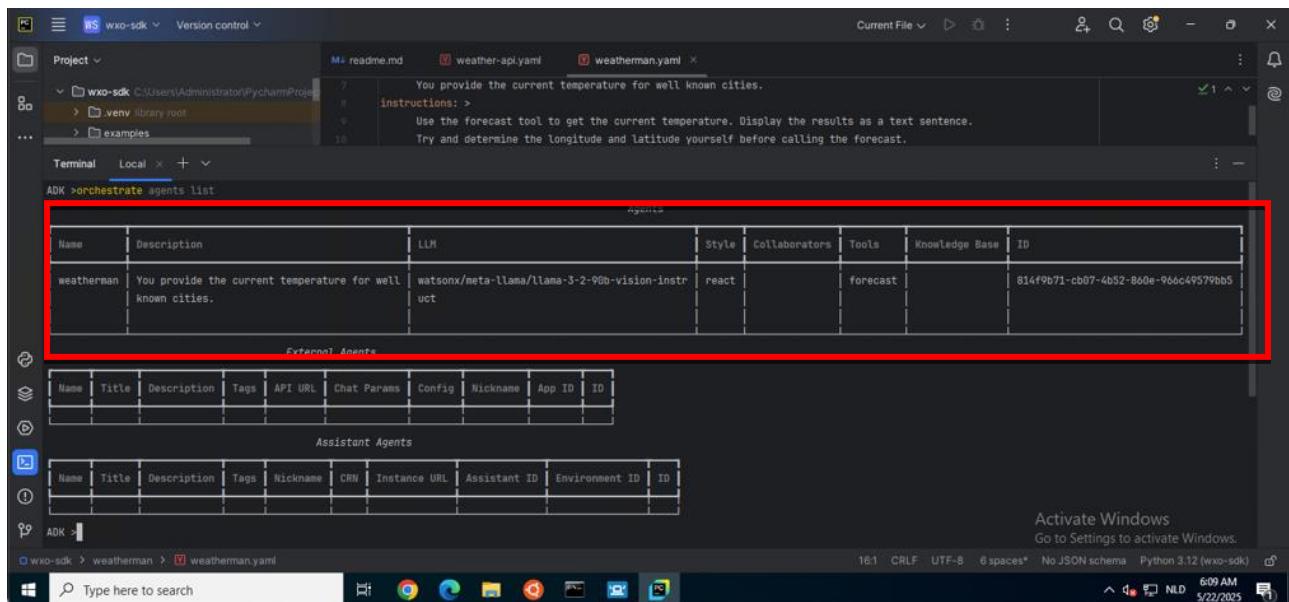
In this section you will be using the CLI to view the list of agents available in your installation of Watsonx Orchestrate ADK.

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

orchestrate agents list



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



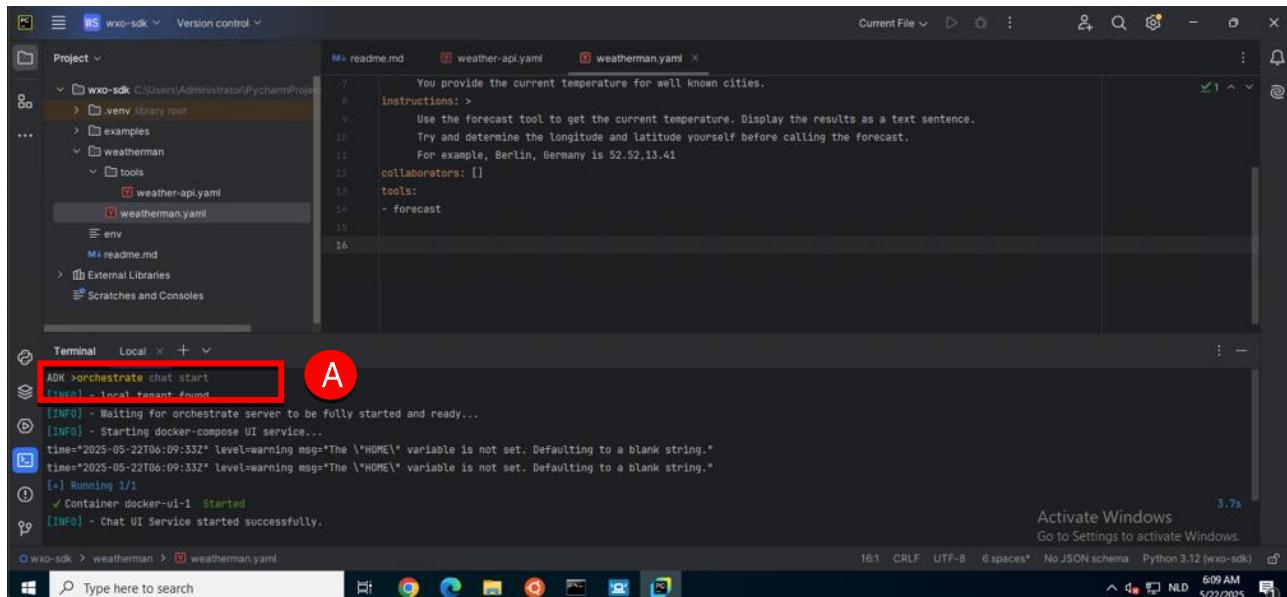
# Testing the weatherman agent

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
```



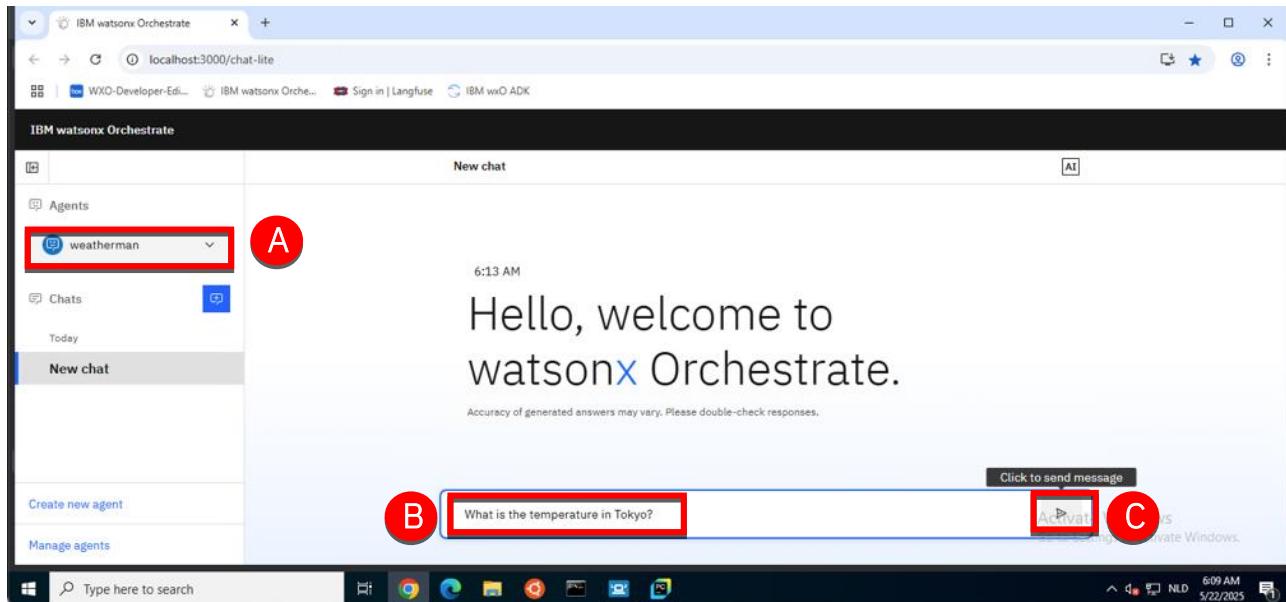
The screenshot shows the PyCharm IDE interface. The terminal window at the bottom has a red circle labeled 'A' over it, indicating the location of the command being run. The command 'ADK >orchestrate chat start' is highlighted with a red box. The output of the command shows the process starting, including Docker compose and a Chat UI service. The code editor window above shows a YAML configuration file with instructions for the weatherman agent.

```
ADK >orchestrate chat start
[Terminal] - 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
  [INFO] - Chat UI Service started successfully.

ADK > weatherman > weatherman.yaml
```

```
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
```

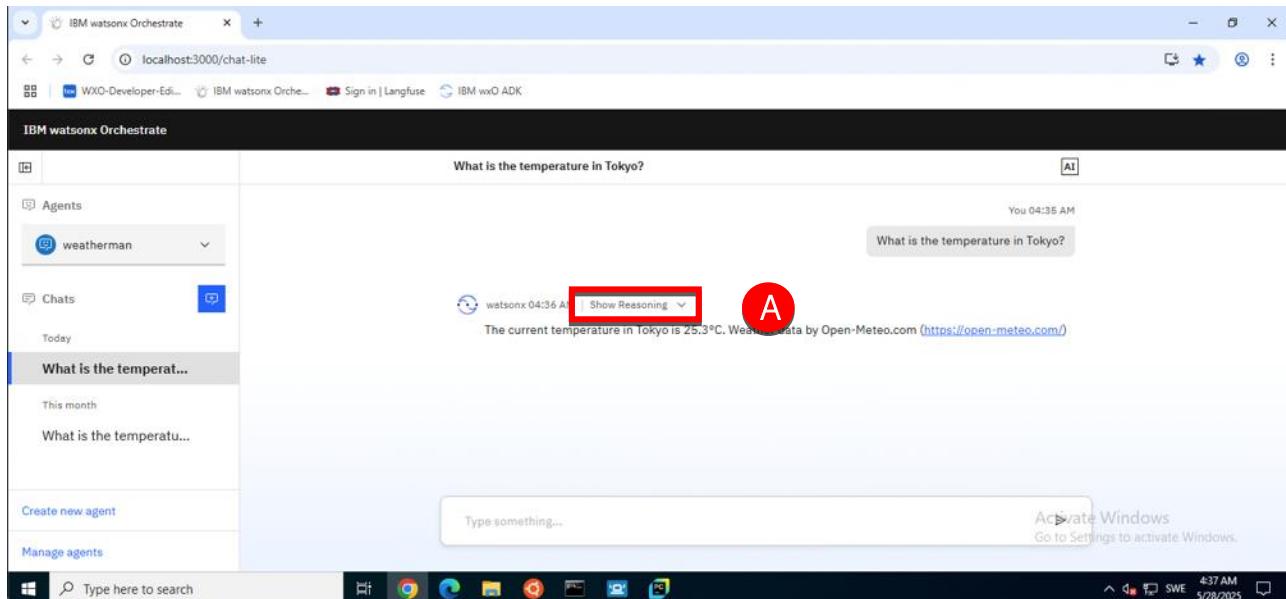
- Wait for the server to load and when the browser opens, make sure to select the **weatherman** agent in the Agents dropdown (A), then in the chat, enter “What is the temperature in Tokyo?” into the chat input (B) and click the submit icon (C).



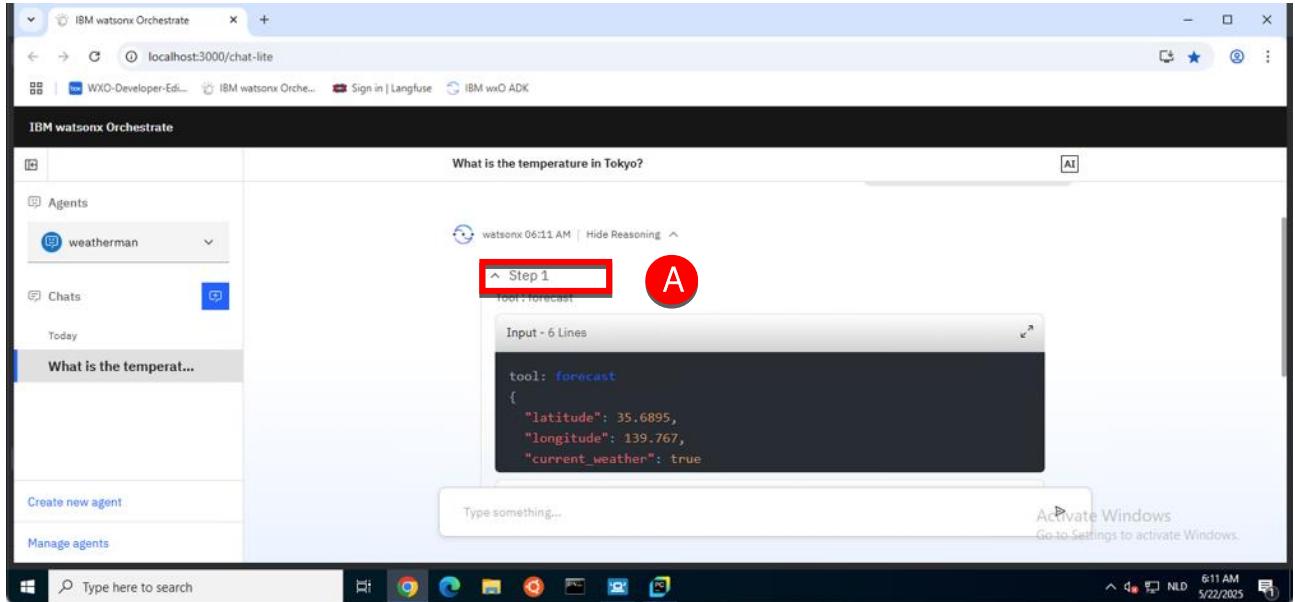
Wait for the agent to reason about how to perform the action and then wait for the response from the agent. Observe that the agent first began planning & reasoning and then responded to the question using live data from the web-service (service provided by open-meteo.com).

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

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



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



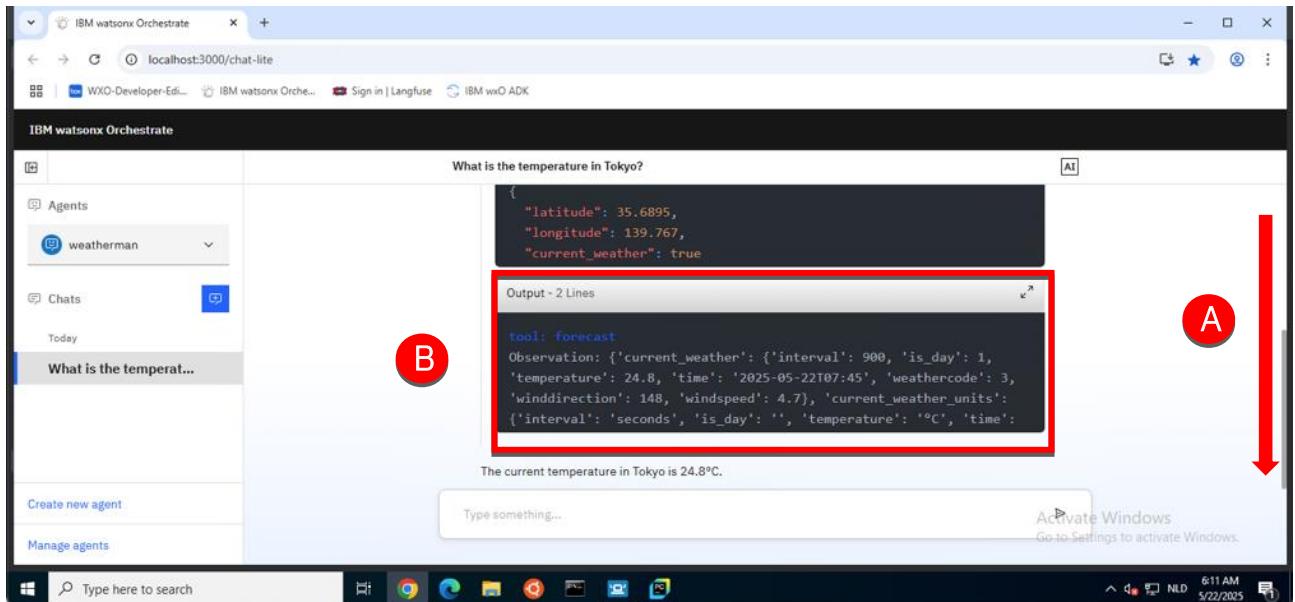
The screenshot shows the IBM Watsonx Orchestrate interface. A user query "What is the temperature in Tokyo?" is at the top. Below it, under the "Chats" section, there's a list item "What is the temperat...". On the right, the "Input - 6 Lines" panel shows a JSON payload:

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

A red box highlights the "Step 1" button above the input panel, and a red circle labeled "A" is positioned to its right.

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.

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



The screenshot shows the same interface as before, but now the "Output - 2 Lines" panel is expanded, indicated by a red box and a red arrow pointing down from the previous "Input" panel. The output shows the expanded JSON response:

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

Below this, the response text "The current temperature in Tokyo is 24.8°C." is displayed. A red circle labeled "B" is on the left side of the output panel, and another red circle labeled "A" is on the right side of the expanded panel.

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
```

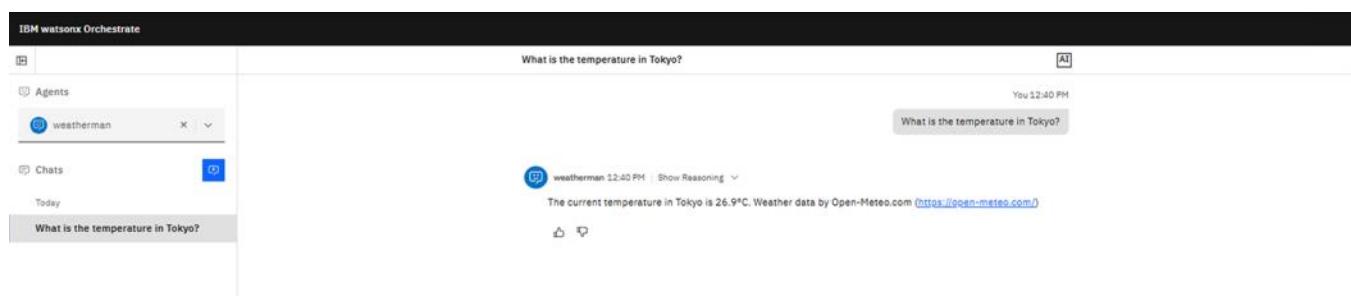
You will need to repeat any commands to create tools and agents.

### 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

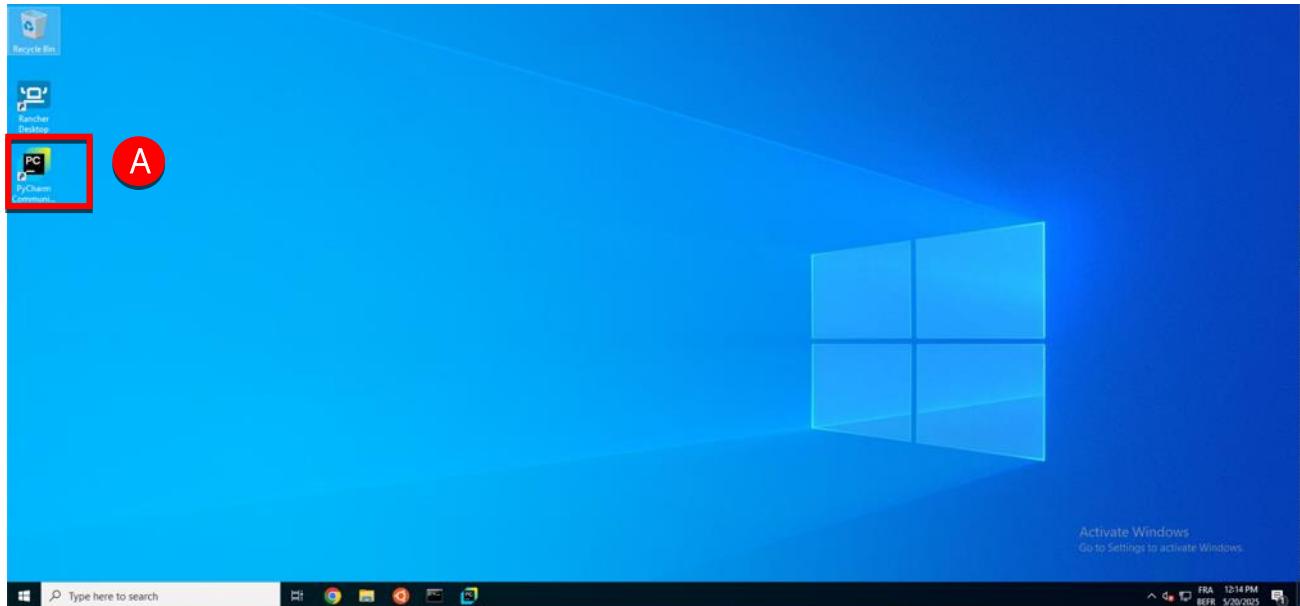
If your weatherman agent does not respond as expected, click the browser refresh button then click the new chat icon (A) and repeat the question. The LLM used by the agent can sometimes respond unpredictably, especially under heavy load.



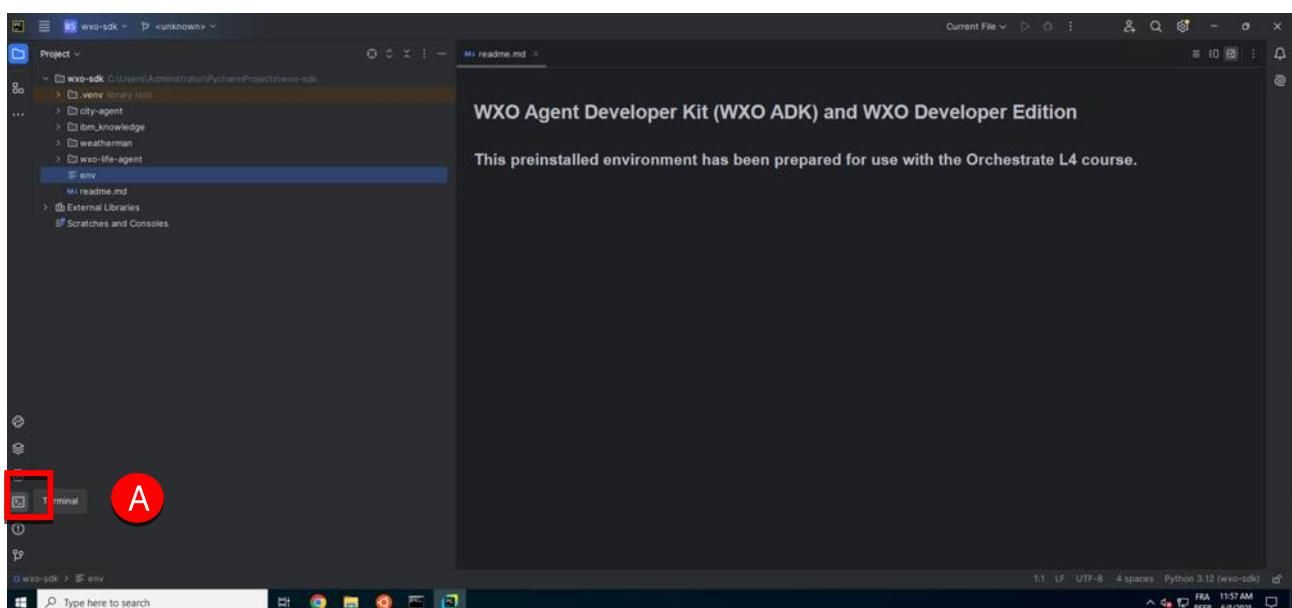
## Appendix: Reset your environment

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.)

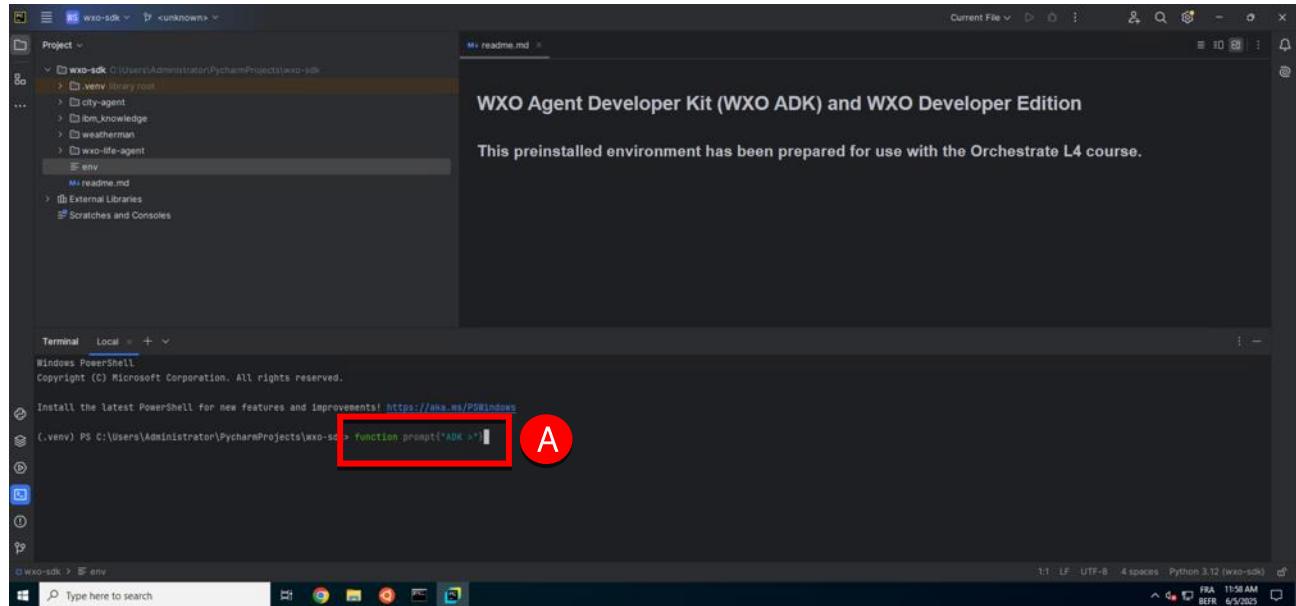


3. Click the terminal icon (A), to open the terminal (if required).



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

```
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
```

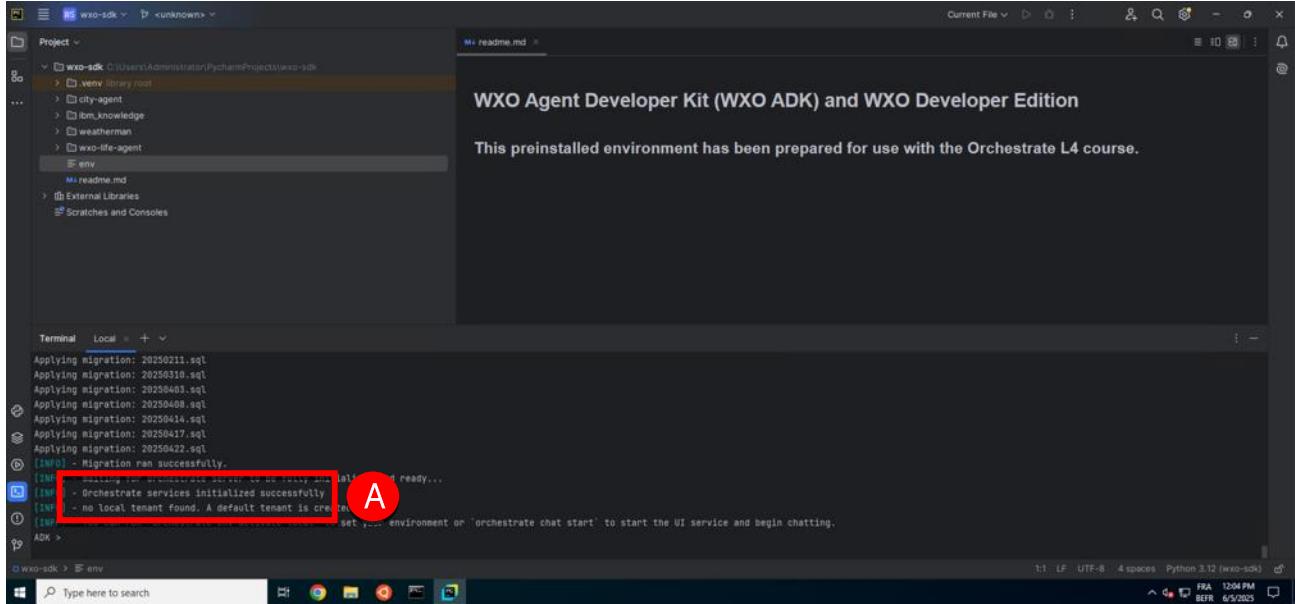
6. Enter the following command into the terminal to reset your server, then press Enter.

```
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.



The terminal output shows the following logs:

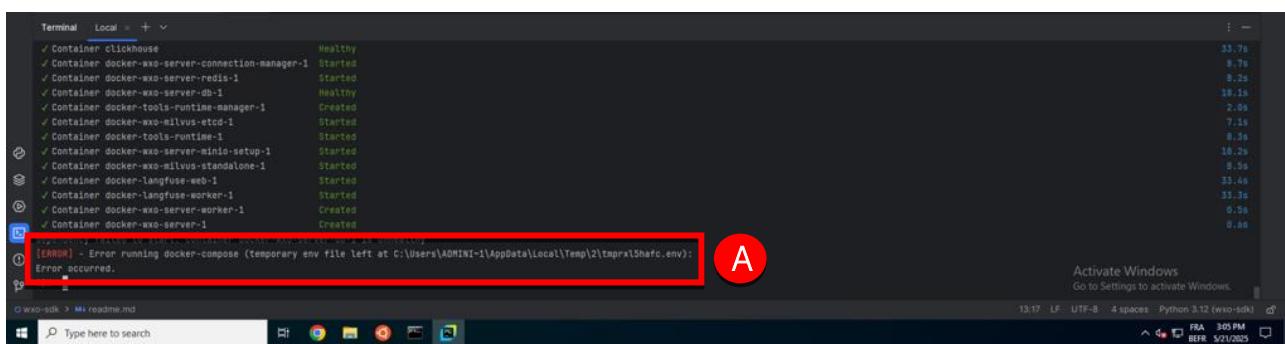
```

Terminal Local + v
Applying migration: 20230211.sql
Applying migration: 20230310.sql
Applying migration: 20230403.sql
Applying migration: 20230408.sql
Applying migration: 20230414.sql
Applying migration: 20230417.sql
Applying migration: 20230422.sql
[INFO] - Migration ran successfully.
[INFO] - Waiting for Orchestrate services to start up... all ready...
[INFO] - Orchestrate services initialized successfully
[INFO] - no local tenant found. A default tenant is created
[INFO] - Set 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:



The terminal output shows the following error:

```

Container clickhouse
✓ Container docker-wxo-server-connection-manager-1 Healthy
✓ Container docker-wxo-server-redis-1 Started
✓ Container docker-wxo-server-db-1 Healthy
✓ Container docker-tools-runtime-manager-1 Created
✓ Container docker-wxo-milvus-etcd-1 Started
✓ Container docker-tools-runtime-1 Started
✓ Container docker-wxo-server-milvus-setup-1 Started
✓ Container docker-wxo-milvus-standalone-1 Started
✓ Container docker-tangfuse-web-1 Started
✓ Container docker-tangfuse-worker-1 Started
✓ Container docker-wxo-server-worker-1 Created
✓ Container docker-wxo-server-1 Created

[ERROR] - Error running docker-compose (temporary env file left at C:\Users\ADMINI-1\AppData\Local\Temp\1\tmprx15hafc.env):
Error occurred.

ADK >

```

