LAB

Evaluate the performance of an AI agent using MLflow

IBM **SkillsBuild**

Contents

Introduction

In this lab, you'll assume the role of an AI specialist to evaluate a pilot version of an AI agent using MLflow.

Software requirements

To complete this lab, you'll need access to a Replicate account, which allows you to use artificial intelligence (AI) models to perform tasks. You'll also need a Replicate application programming interface (API) token, which acts as a secure key to connect your Colab environment so that the lab can run smoothly.

Basic familiarity with Python will enable you to better follow how the agent works and its tool usage.

Objective

After completing this lab, you should be able to:

- Evaluate the performance of an AI agent using MLflow

Lab steps

This lab requires you to complete the following procedures:

- Create a GitHub account
- Create a Replicate account
- Sign up for Google Colab
- Initialize the AI agent and tools
- Evaluate the trajectory and final response of the AI agent

Estimated duration to complete

30 minutes

Scenario

Background information

TechMart, a fast-growing e-commerce platform, is transforming its shopping experience with an AI agent built on IBM Granite. The agent replaces static catalogs, offering real-time recommendations, answering questions, and personalizing the customer's journey.

You are part of TechMart's newly formed AI evaluation team, serving as the AI specialist for this project. Your role is to assess the agent's performance by evaluating product recommendations, response accuracy, and the overall user experience.

Challenge
TechMart users often leave the site when the AI agent fails to provide relevant product recommendations or assist effectively with catalog navigation. Additionally, missing or inconsistent details about products, pricing, or stock levels can erode the user's trust.

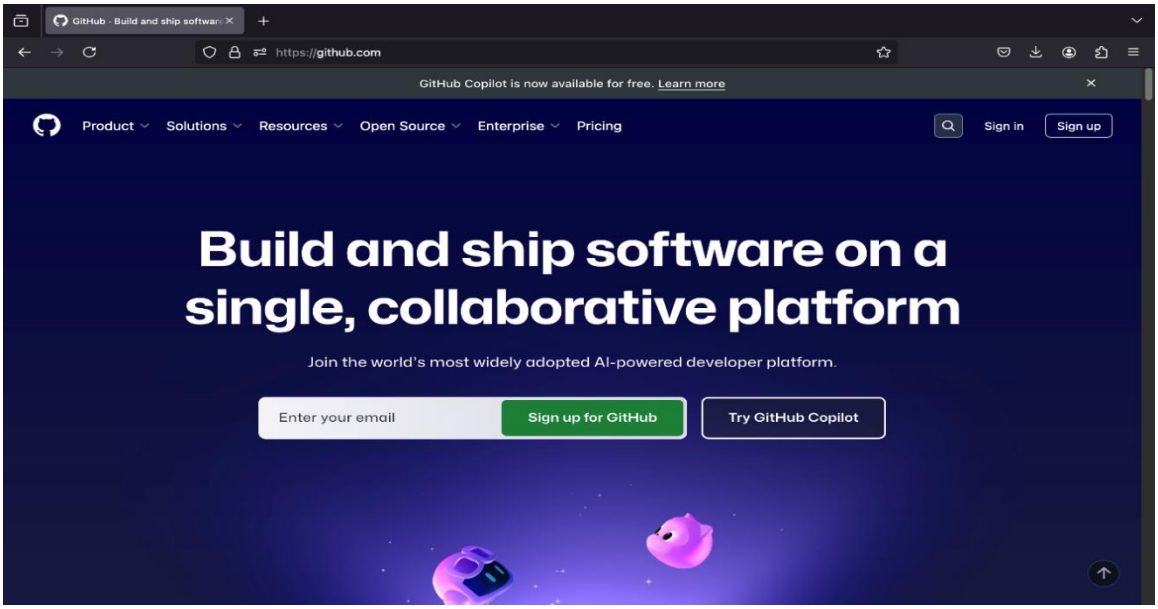
Solutions
To support this, your team uses MLflow to track, evaluate, and visualize the agent's performance. In this lab, you'll determine whether the AI agent is ready for full-scale deployment.

Create a GitHub account

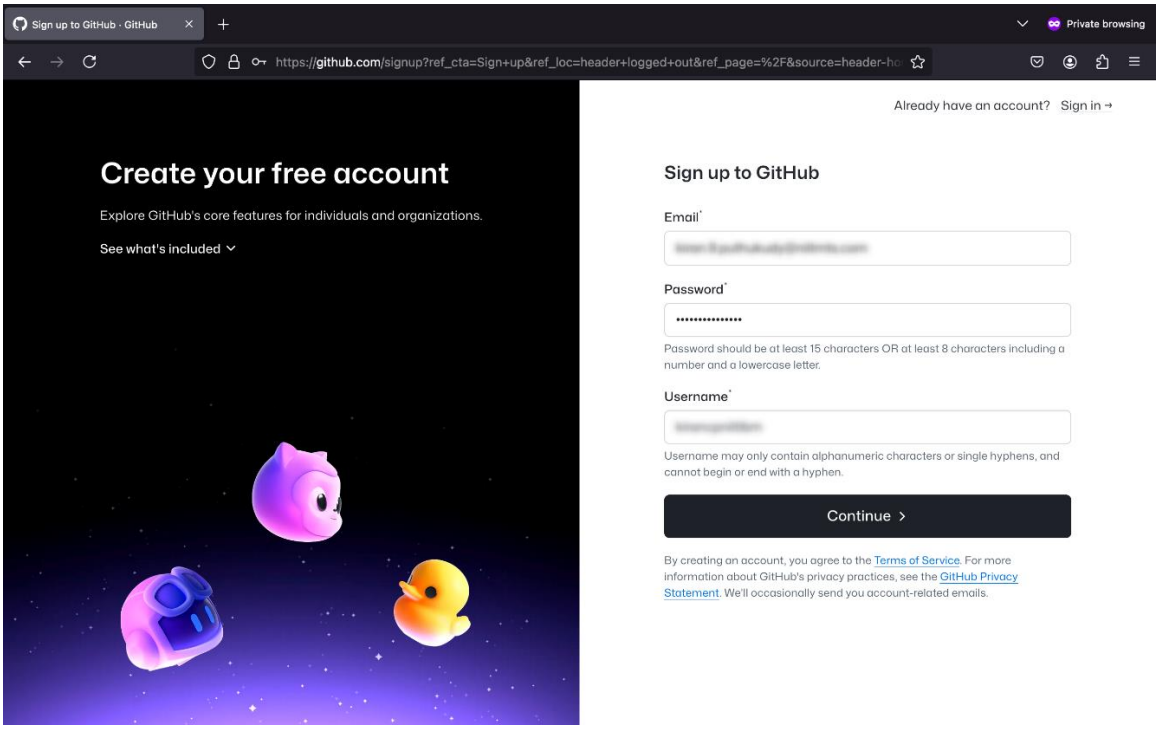
Overview
In this procedure, you'll set up a GitHub account. GitHub is a platform that helps developers store, manage, and share code, while also supporting collaboration through tools such as version control, bug tracking, and task management. Setting up a GitHub account provides access to the Replicate platform, which is required to complete the lab efficiently.

Instructions
1. To create a GitHub account, go to the GitHub website.

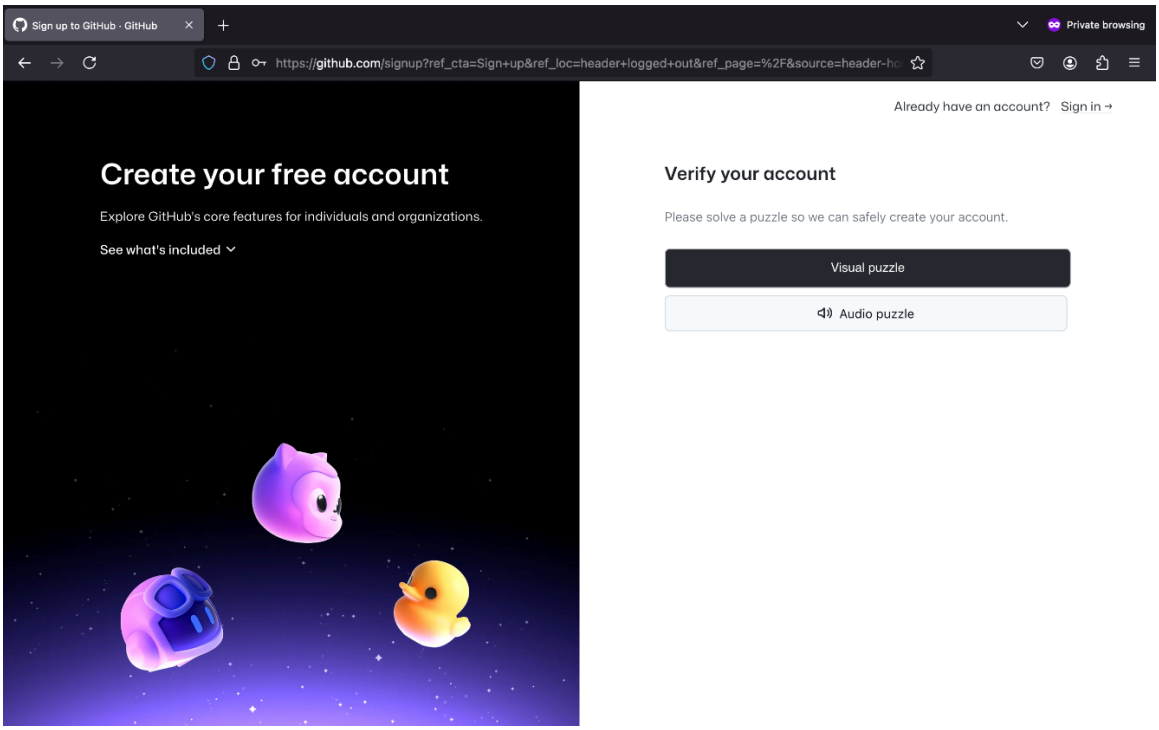
2. Select the **Sign up for GitHub** button, as shown in the following screenshot.



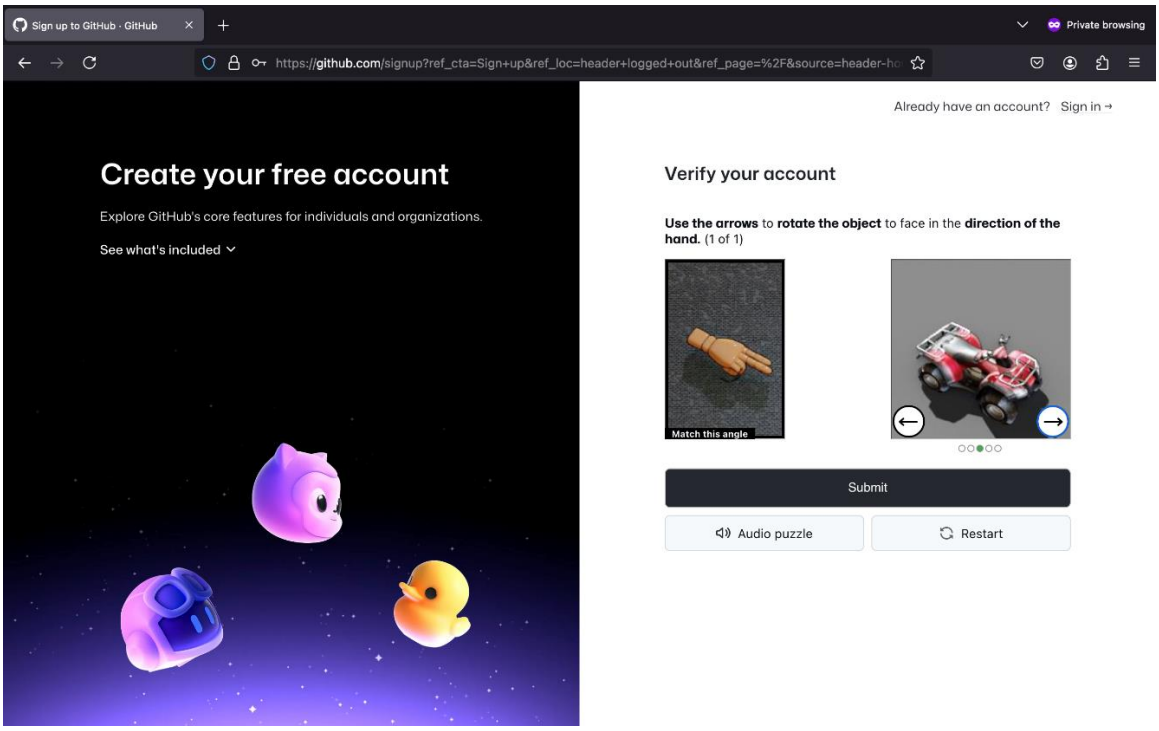
3. Enter your details in the **Email**, **Password**, and **Username** fields. Then, select **Continue**.
The following screenshot shows the "Sign up to GitHub" page.



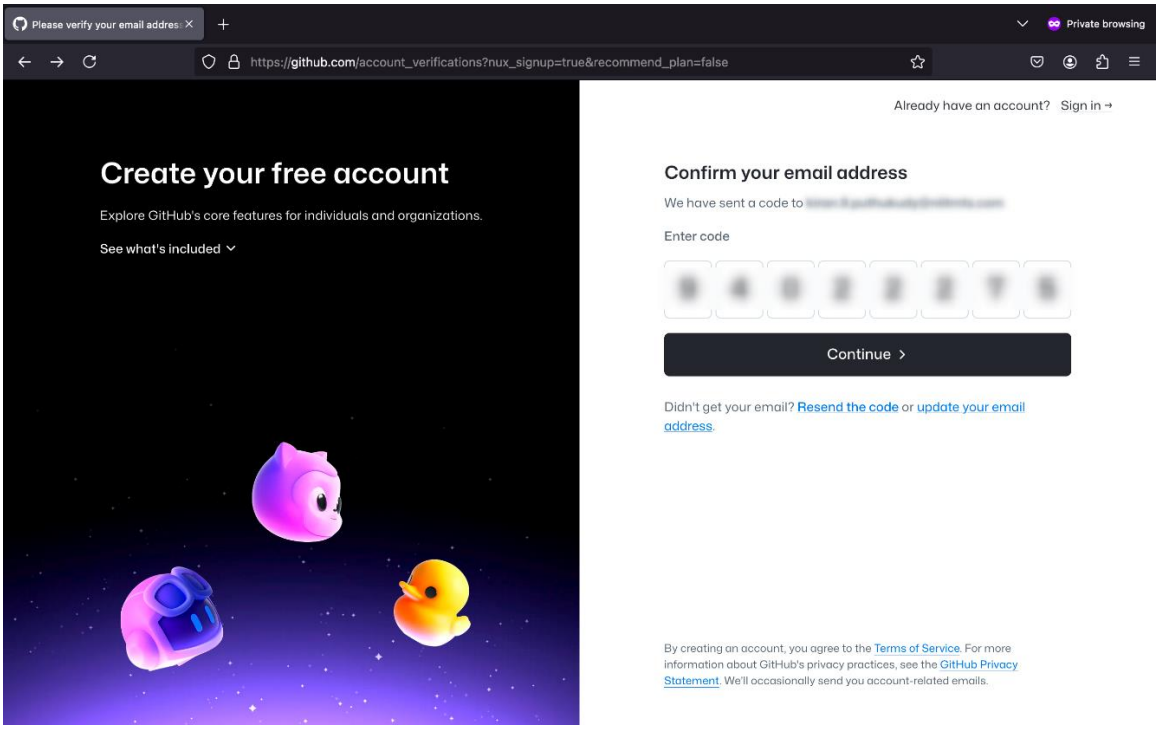
4. To verify your account, select the **Visual puzzle** button.



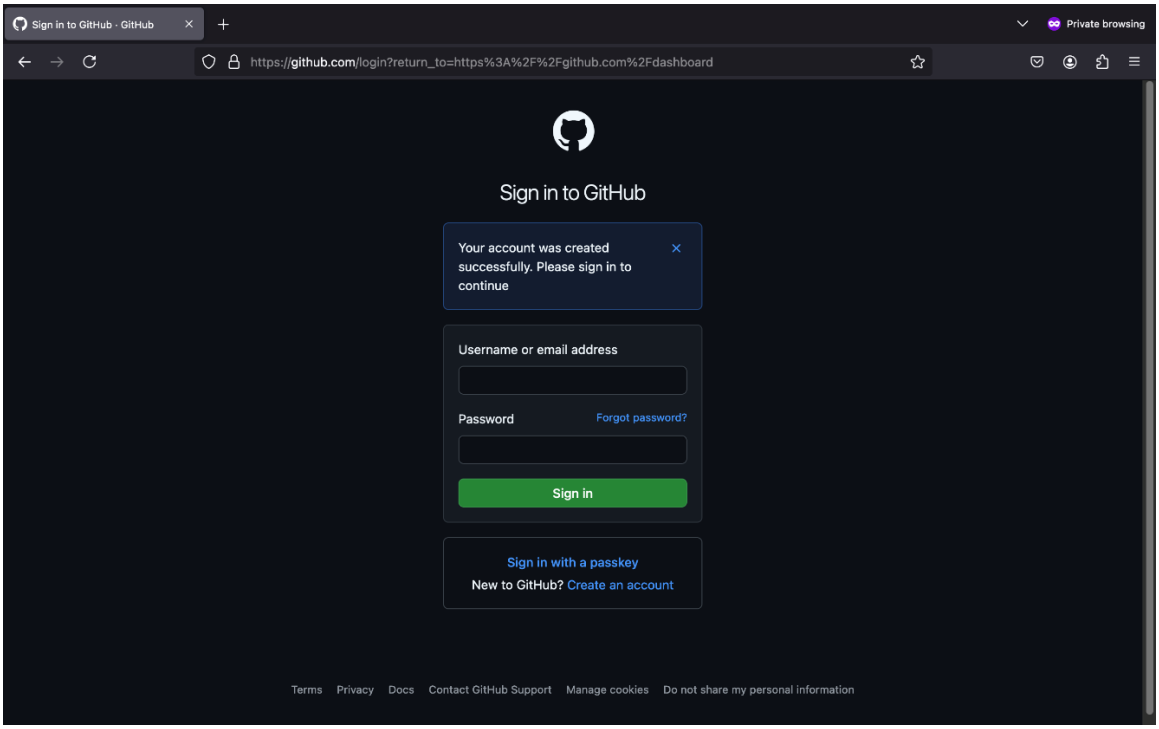
5. Solve the visual puzzle and select **Submit**.



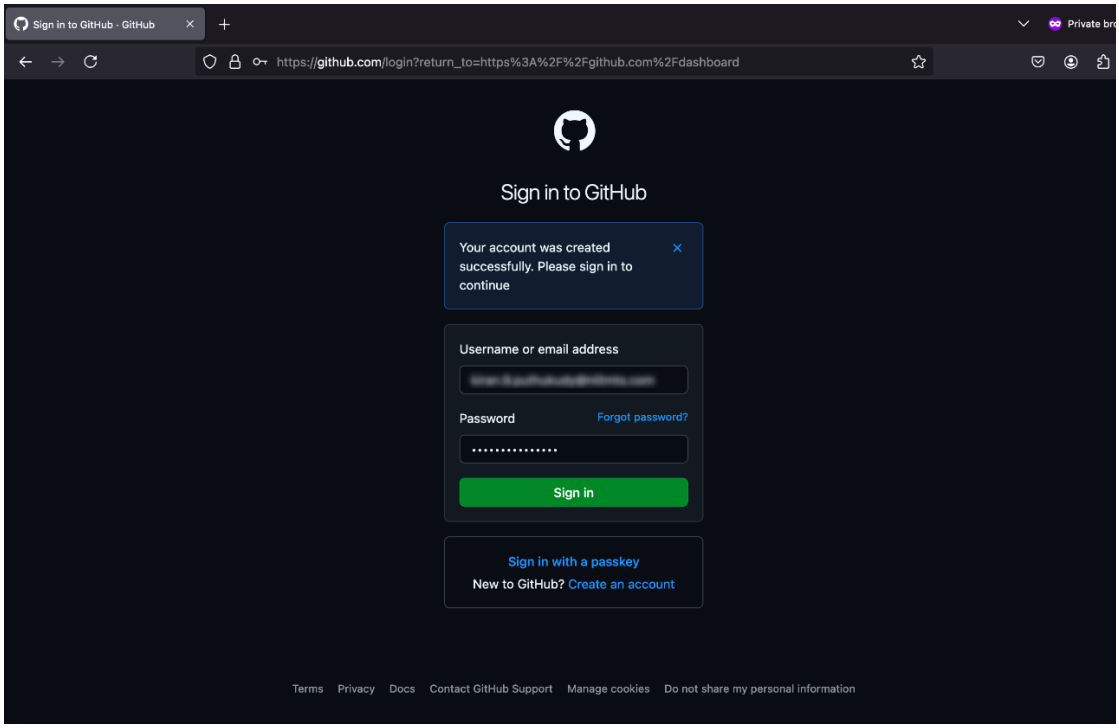
6. To confirm your email, enter the confirmation code sent to your registered email in the **Enter code** field and select **Continue**.



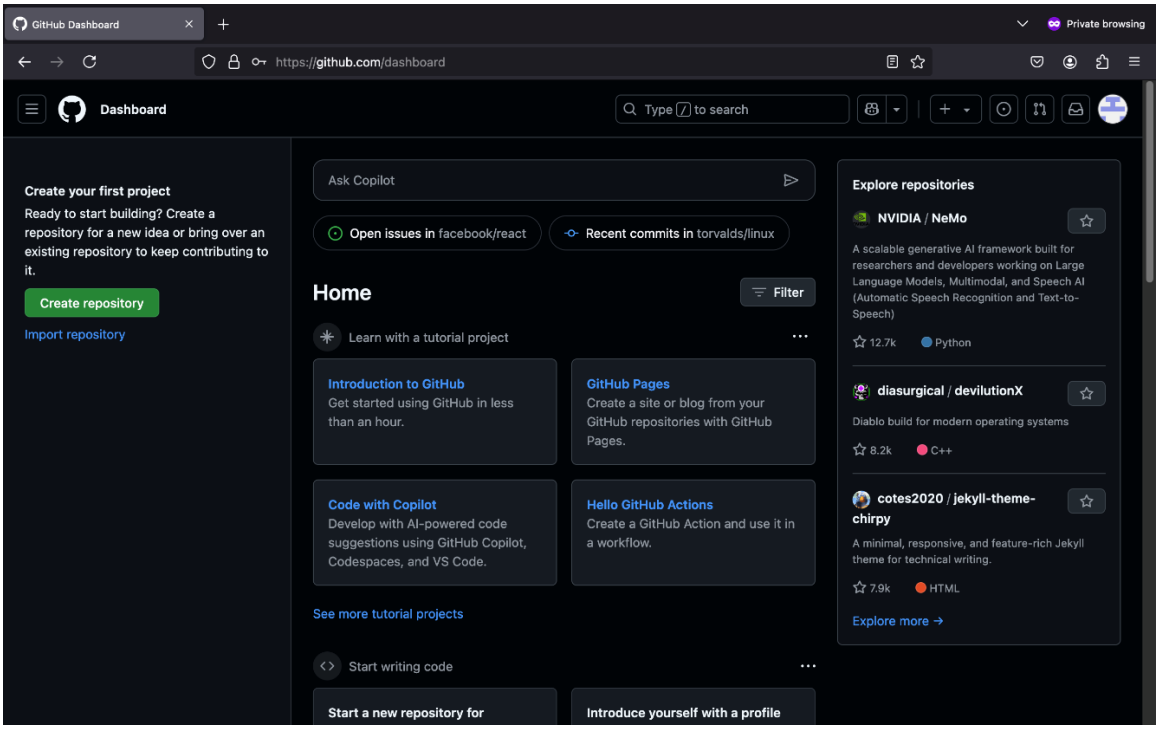
After your GitHub account has been successfully created, a confirmation message is displayed as shown in the following screenshot.



7. To sign in to your account, enter your credentials in the **Username or email address** and **Password** fields, and then select **Sign in**.
   The following screenshot shows the GitHub sign-in page which includes fields for entering the username and password.



After you sign in, the GitHub dashboard appears as shown in the following screenshot.



Create a Replicate account

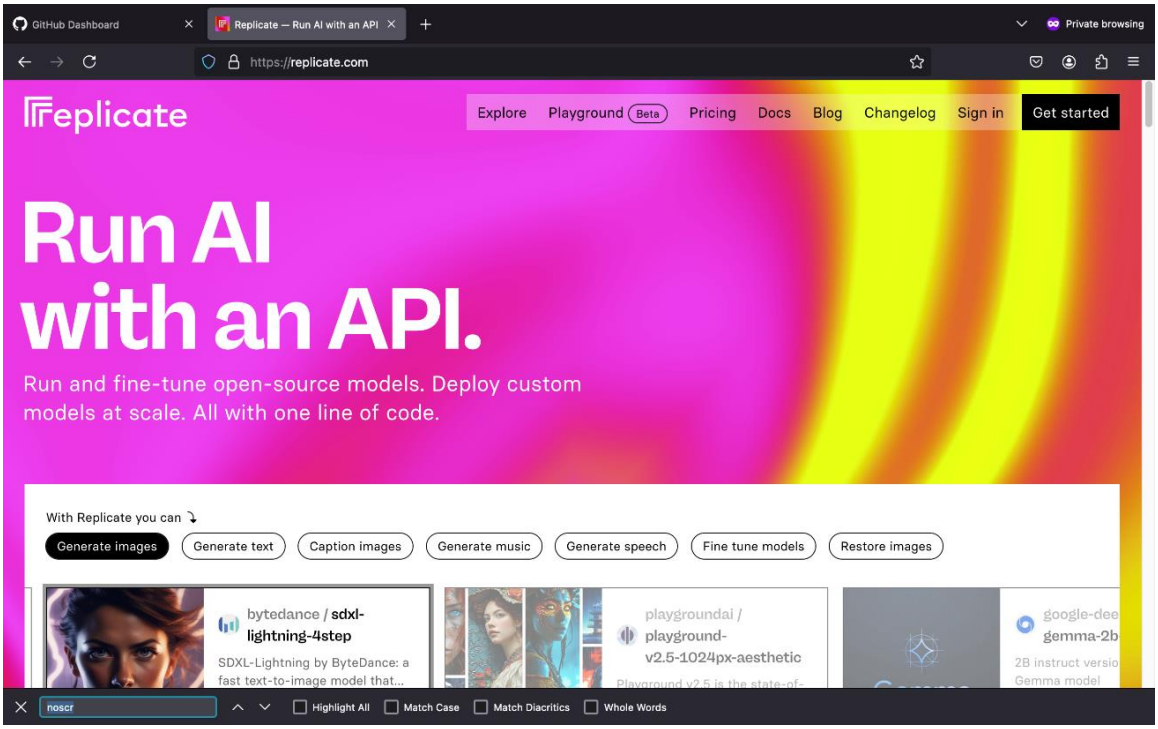
Overview

In this procedure, you'll use your GitHub account to register for a Replicate account. Replicate is a cloud-based platform that lets you use AI models such as IBM Granite without requiring advanced hardware. As part of this procedure, you'll create a Replicate token. A token is a secure access key that allows the lab environment to authenticate with Replicate and run models from your account in Google Colab.

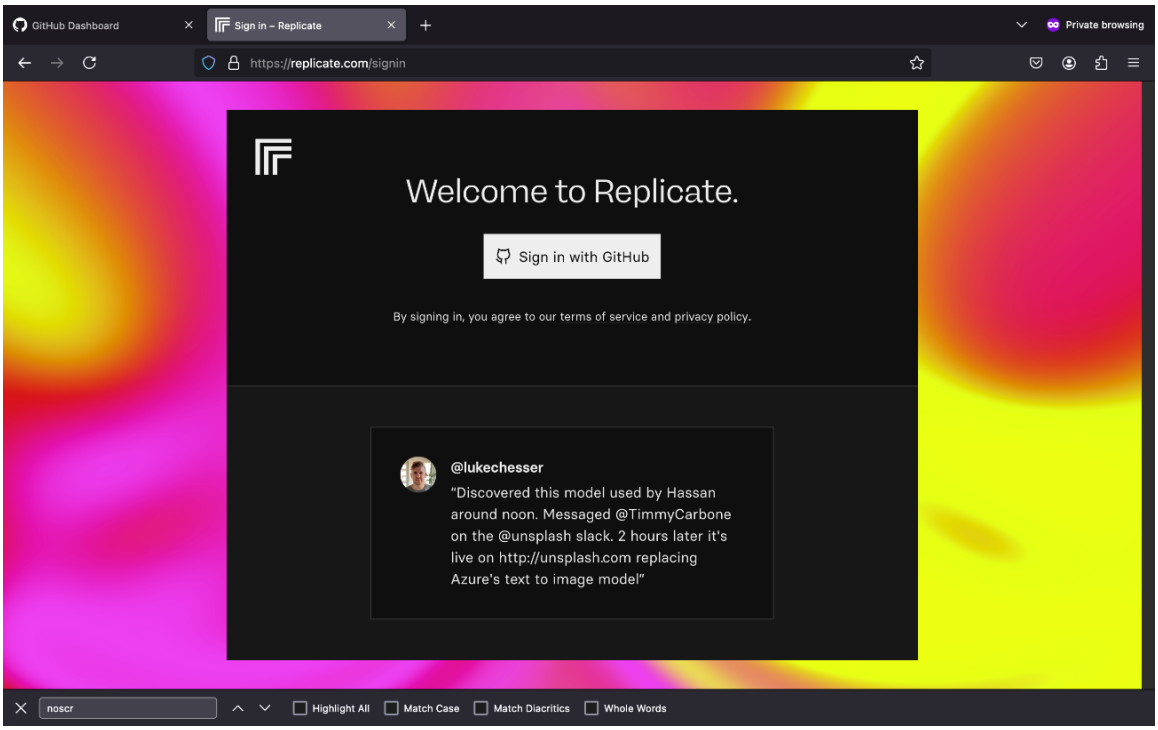
Instructions

1. Go to the Replicate website.

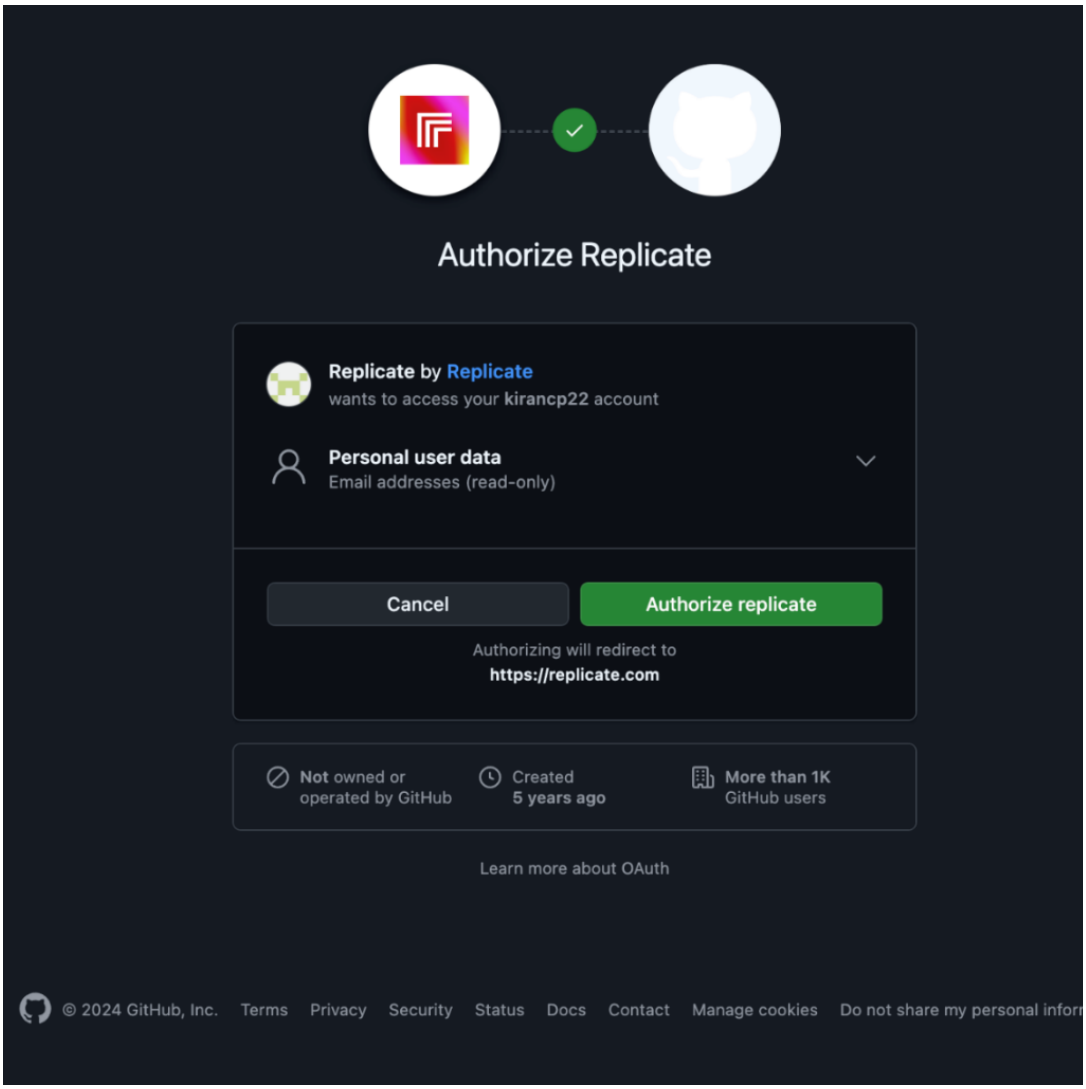
2. Select **Get started**, as shown in the following screenshot.



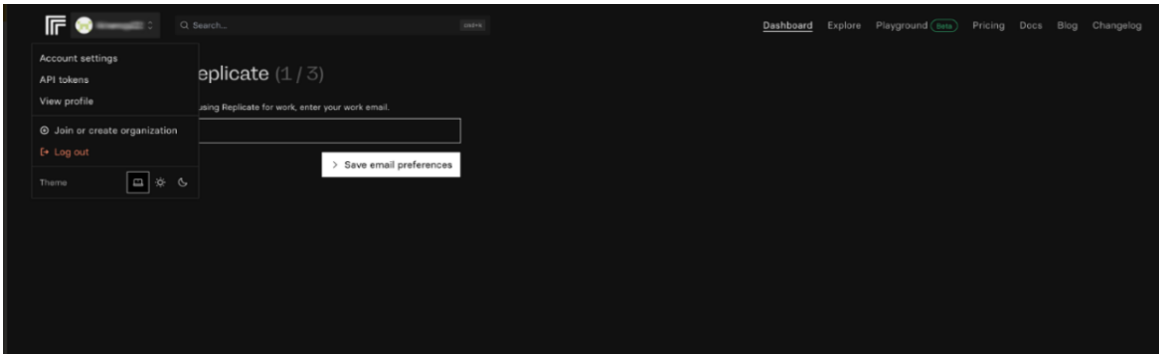
3. Select the **Sign in with GitHub** button, as shown in the following screenshot.



4. Select **Authorize replicate** to continue, as shown in the following screenshot.

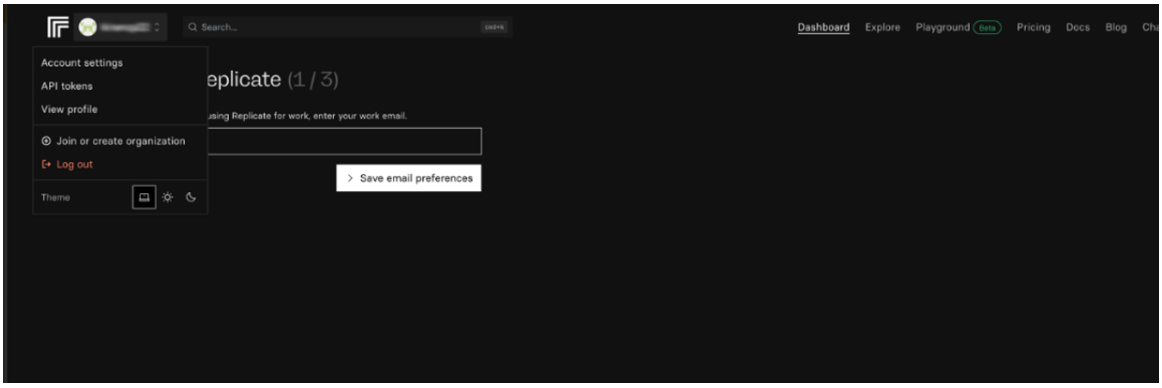


5. After your Replicate account is created, the Replicate dashboard appears as shown in the following screenshot. To create a Replicate token, select the **Account settings** option from the navigation menu.



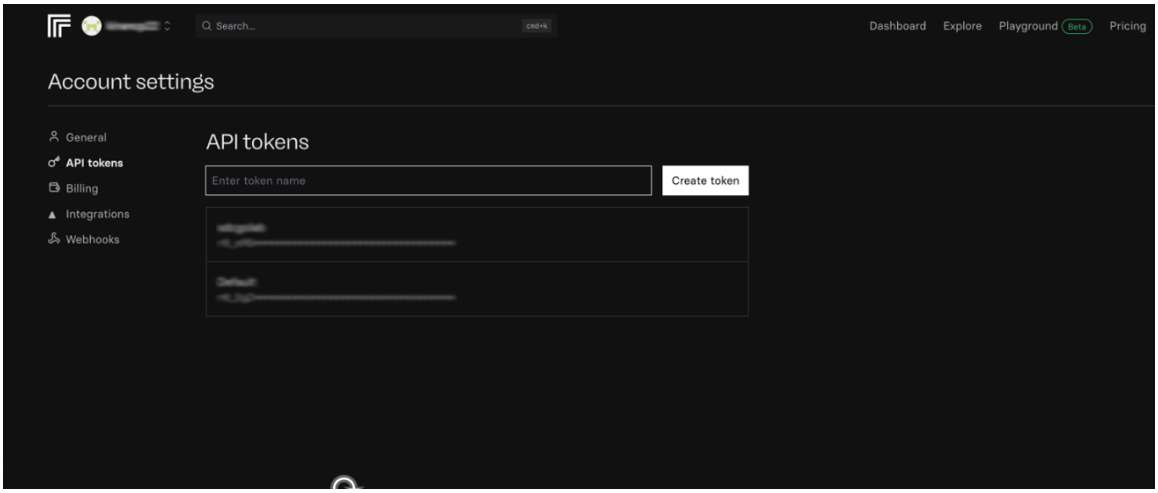
6. Select **API tokens** from the **Account settings** panel.

The following screenshot shows the Replicate dashboard.



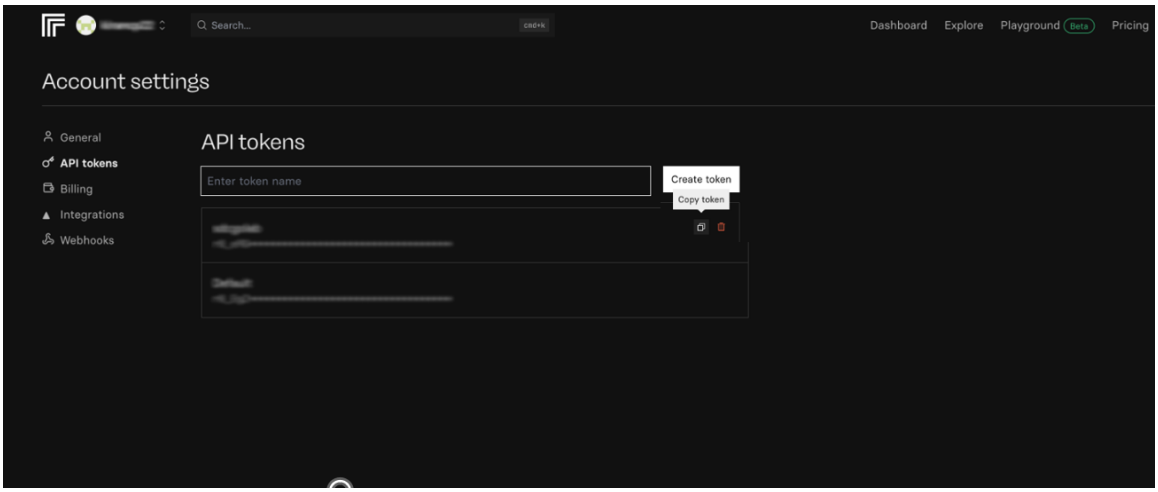
7. Enter a name for the token in the **API tokens** field and then select **Create token**, as shown in the following screenshot.

After your API token is created, it should be displayed on the “Account settings” page as shown in the following screenshot.



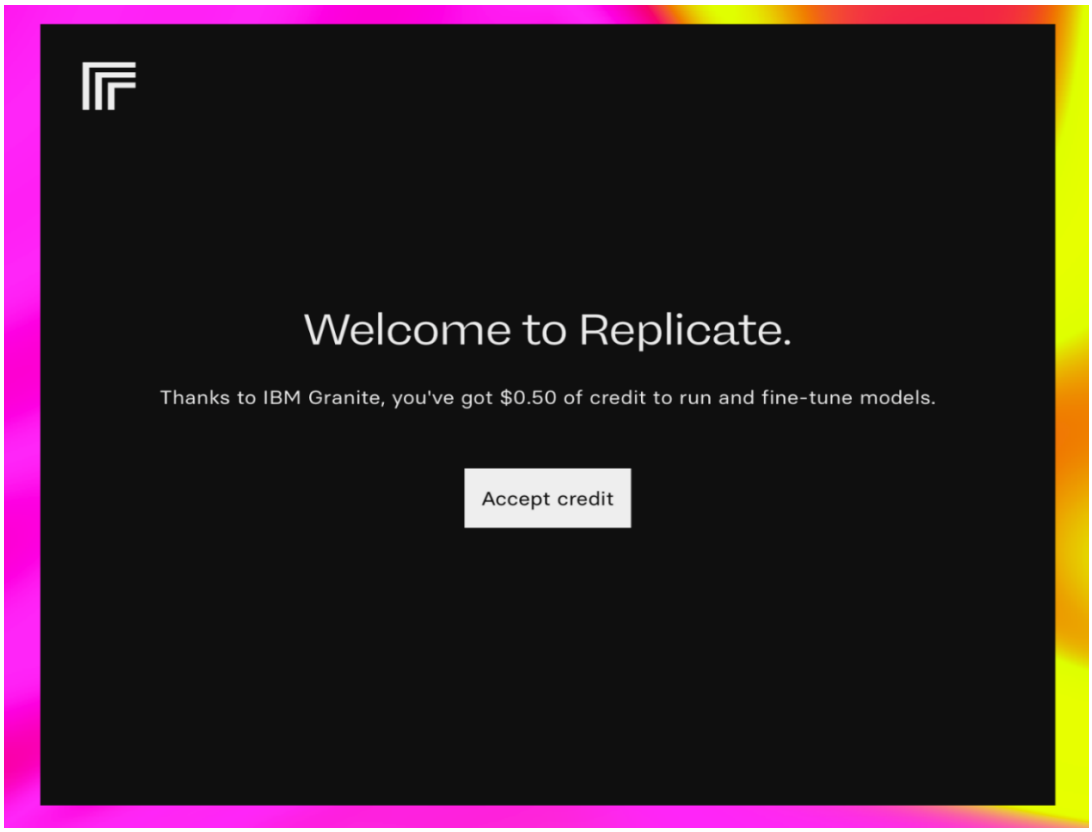
8. To copy the Replicate API, select the **Copy token** icon, as shown in the following screenshot.

   Note: Save the Replicate token because you'll need it to authenticate with the Replicate API when running code in the Google Colab environment later in this lab.

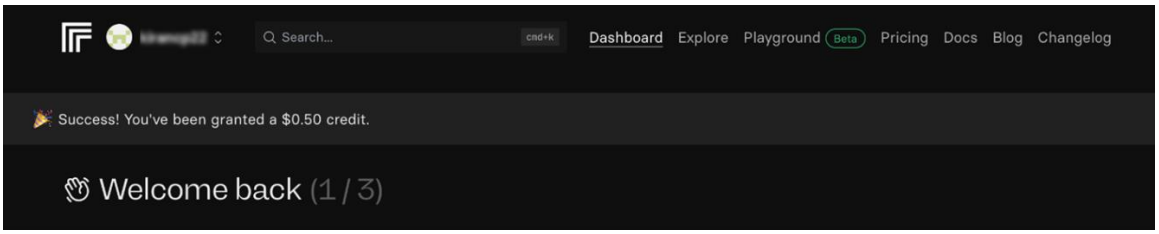


9. To run the lab without interruptions, you'll require some Replicate credits. Go to the Replicate invite link to claim your free $0.50 credit.

10. To claim the amount, select **Accept credit**.



A confirmation message is displayed indicating that a $0.50 credit has been added to your Replicate account as shown in the following screenshot.



Sign up for Google Colab

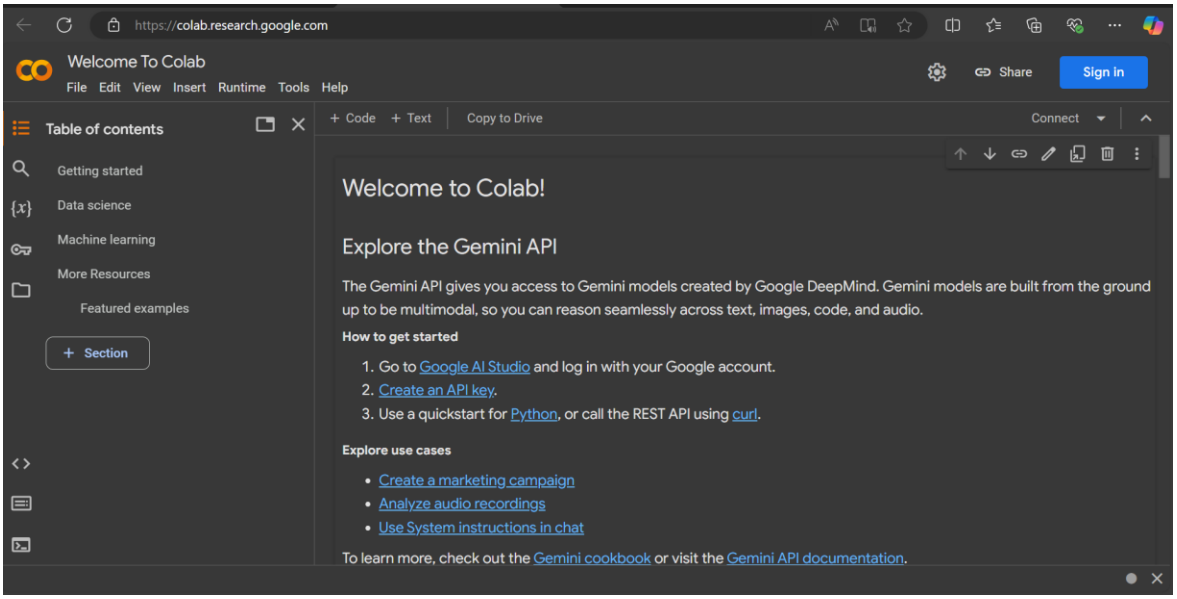
Overview

In this procedure, you'll set up a Google Colab account. Google Colab is a free cloud platform that lets you run code in notebooks, which are commonly used for machine learning, data science, and AI tasks. A Google Colab account allows you to install and use the tools needed to complete this lab.

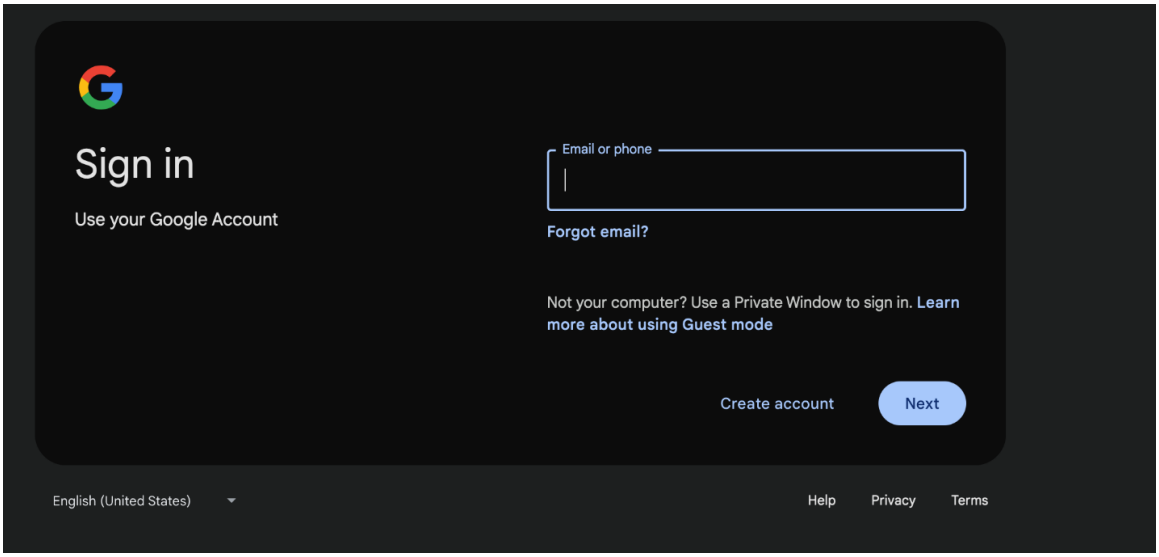
Instructions

1. To sign up, go to the Google Colab website.

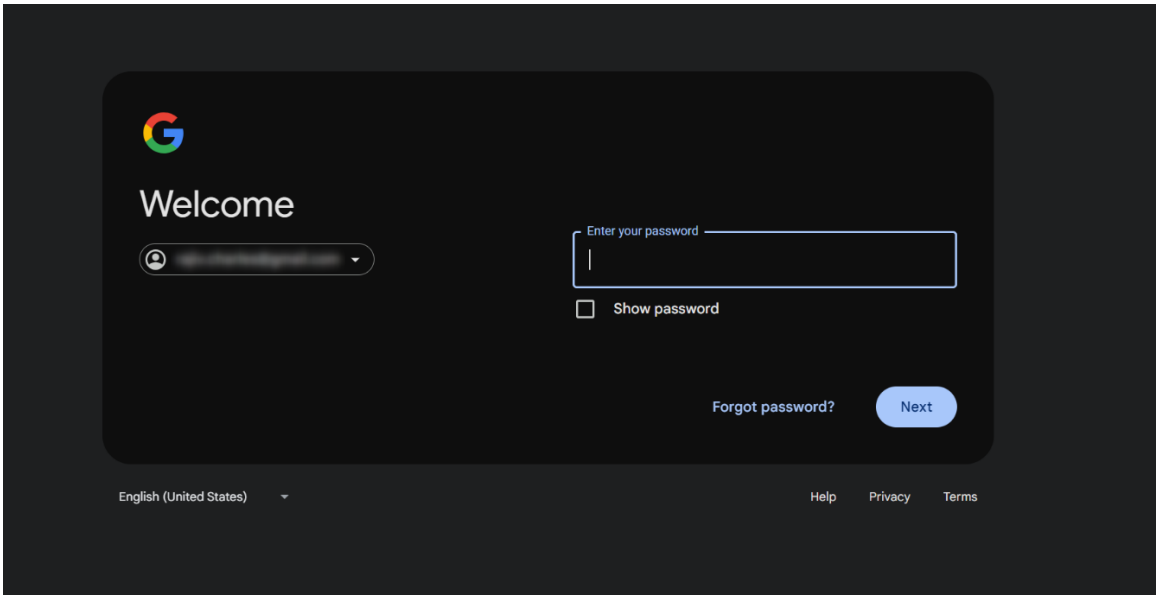
2. Select **Sign in**, as shown in the following screenshot.



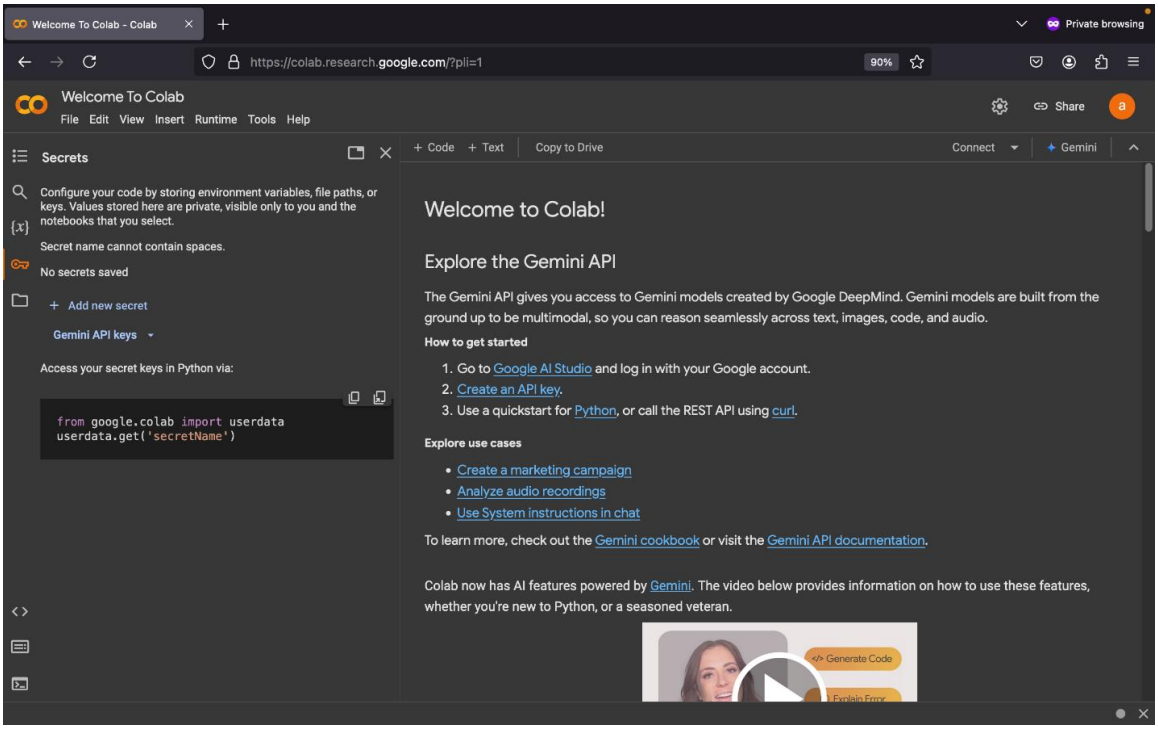
3. Enter your email or phone number in the **Email or phone** field, and then select **Next**. The following screenshot shows the Google sign-in page.



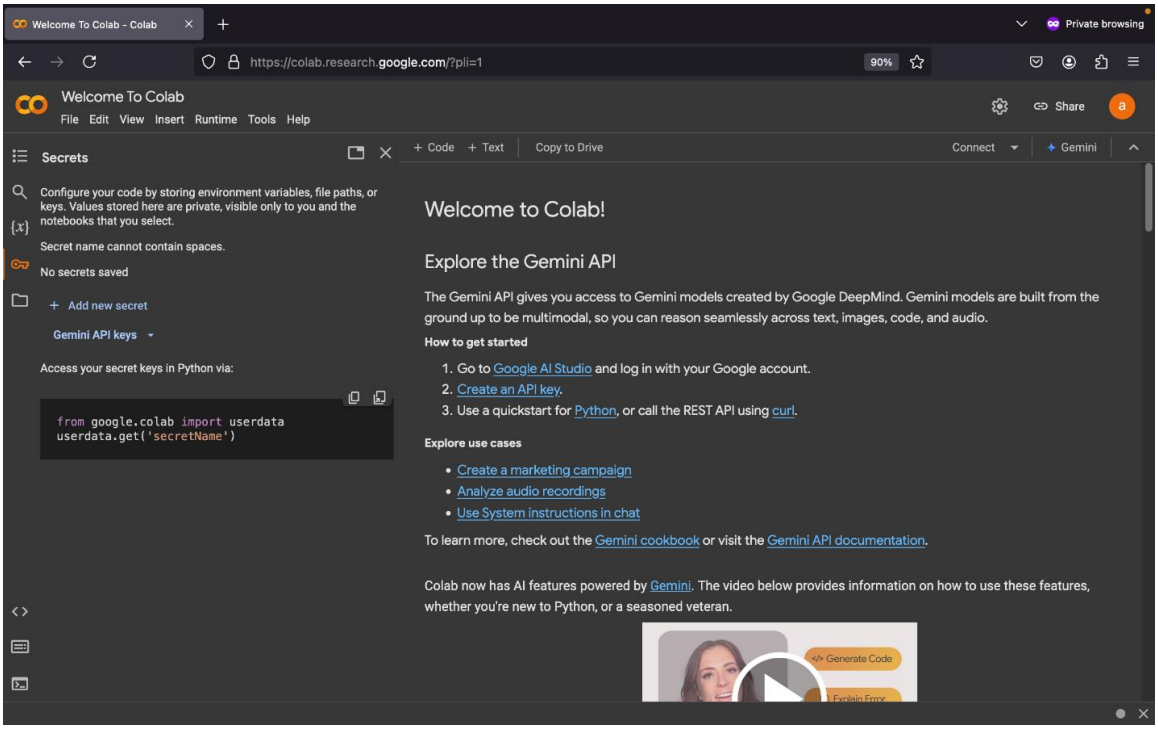
4. Enter your password in the **Enter your password** field, and then select **Next**.



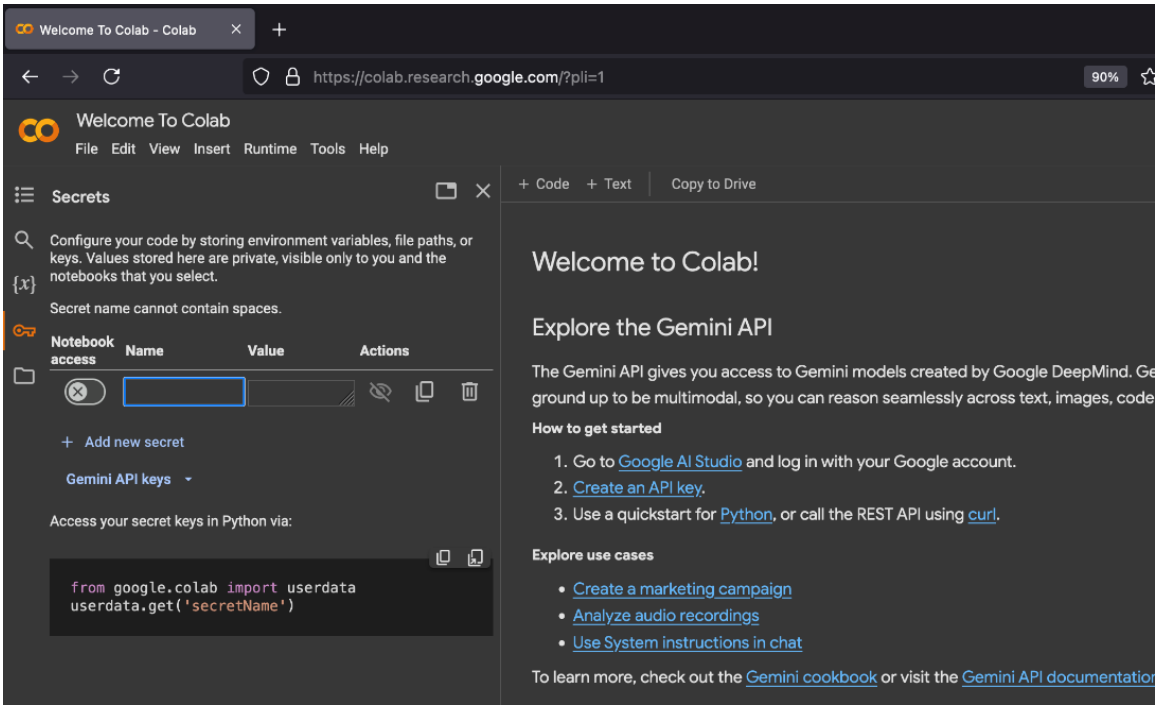
5. To store your Replicate API token, select the key icon from the **Secrets** tab on the Welcome to Colab page, as shown in the following screenshot.



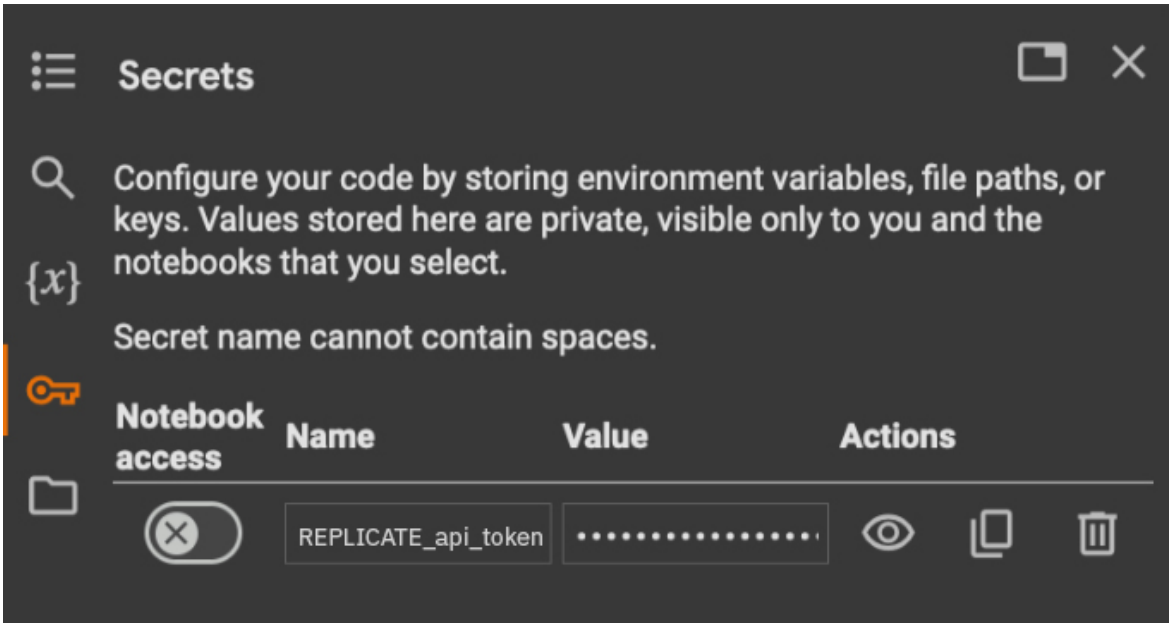
6. Select **Add new secret**.



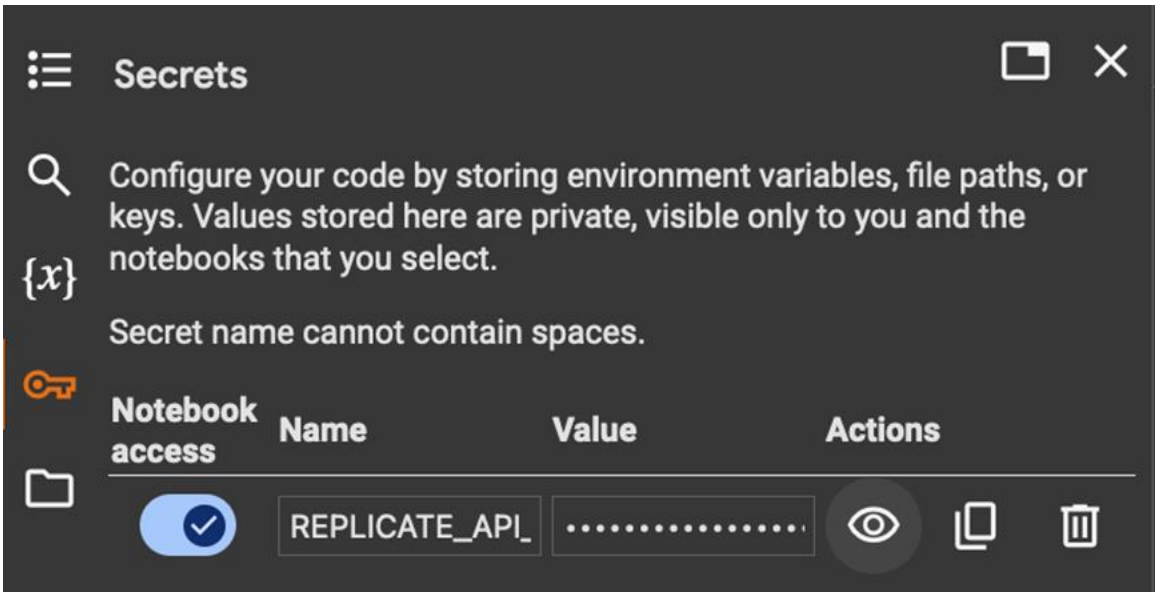
7. Type `REPLICATE_api_token` in the **Name** field.



8. Paste your Replicate API token you copied into the **Value** field, as shown in the following screenshot.



9. Set the **Notebook access** switch on, as shown in the following screenshot. Then, select the close icon to exit the configuration.



Initialize the AI agent and tools

Overview

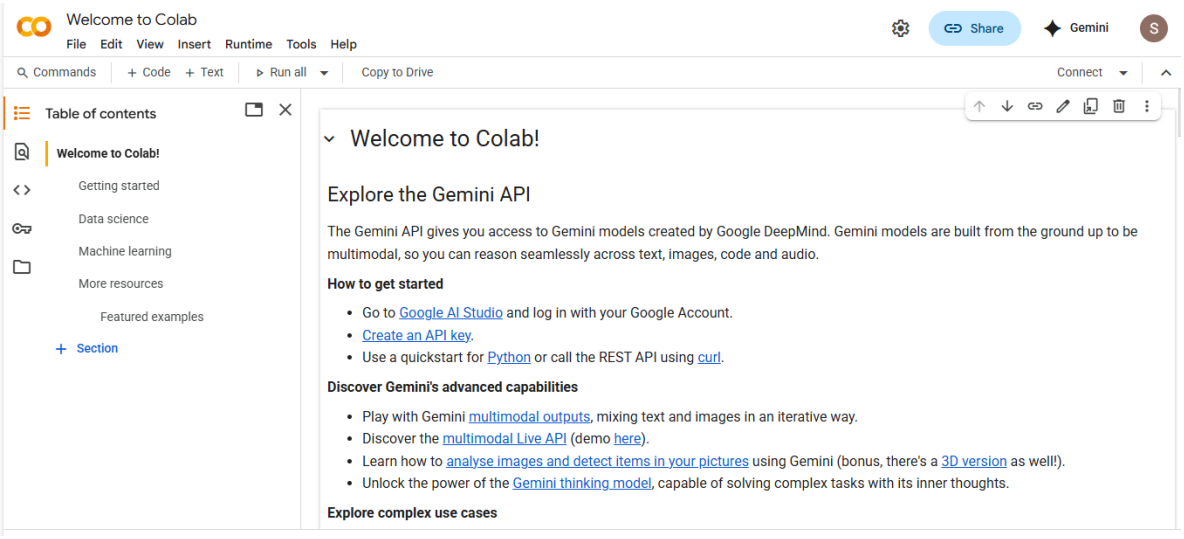
In this procedure you'll initialize TechMart's AI agent and the tools it uses by loading a Jupyter notebook.

In the context of an AI agent, **tools** are functions that help an AI agent perform specific tasks such as searching for information, checking product stock, or looking up data. The user provides prompts that define which tools the agent can access by including them in the agent's setup. When responding, the agent sends a query to the selected tool, receives the result, and uses that information to generate its reply.

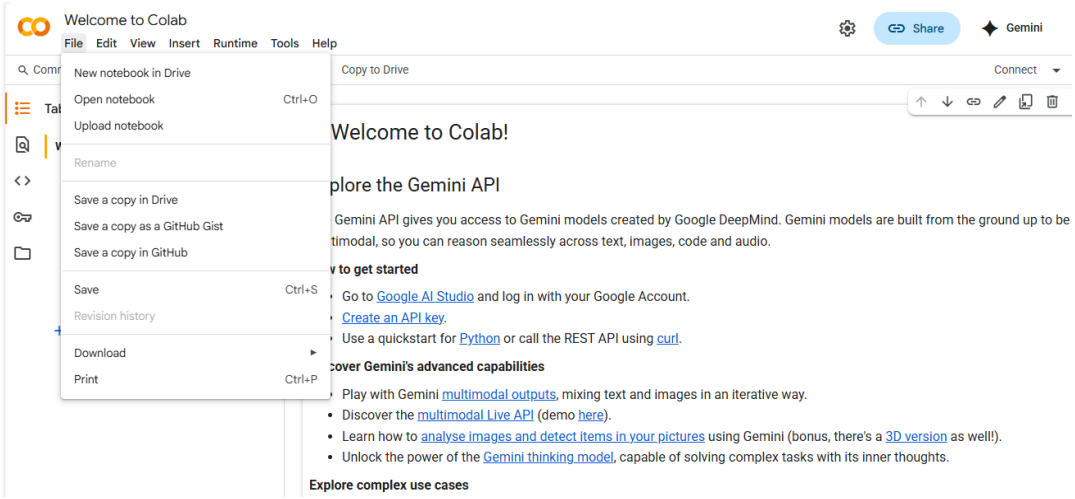
A **Jupyter notebook** is a shareable document that combines computer code, plain language descriptions, data, and visualizations.

Instructions

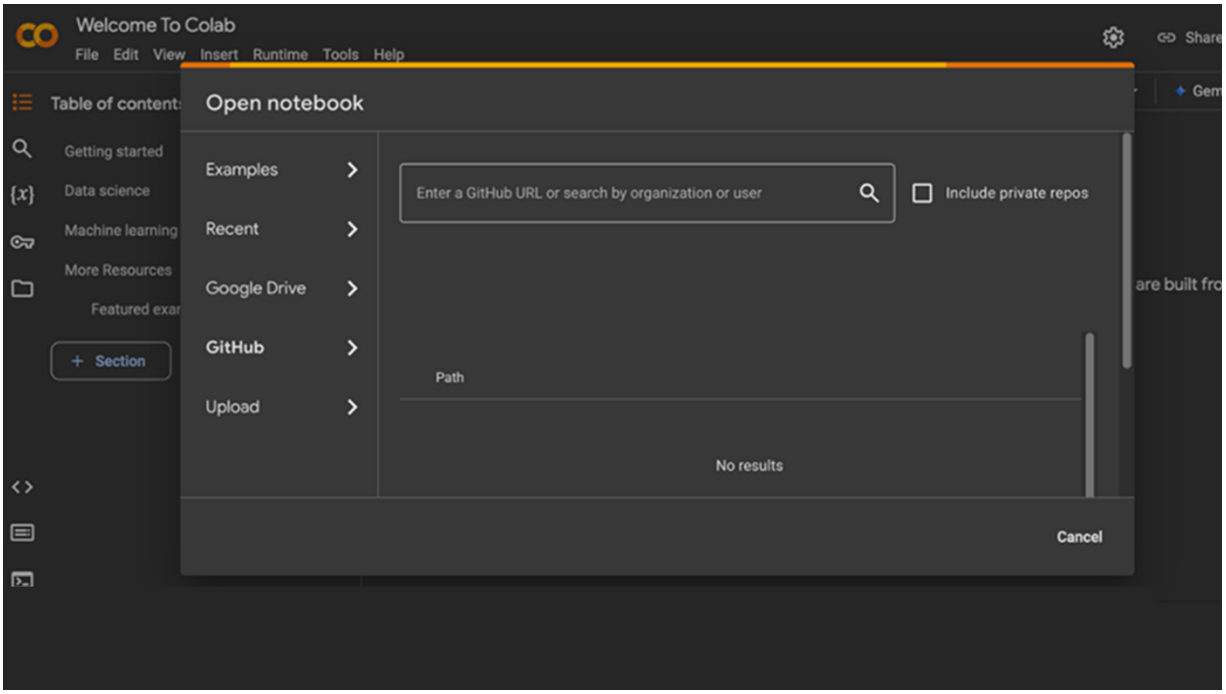
1. Select Google Colab to open Google Colab as shown in the following screenshot. To begin working on the notebook, select **File** from the menu.



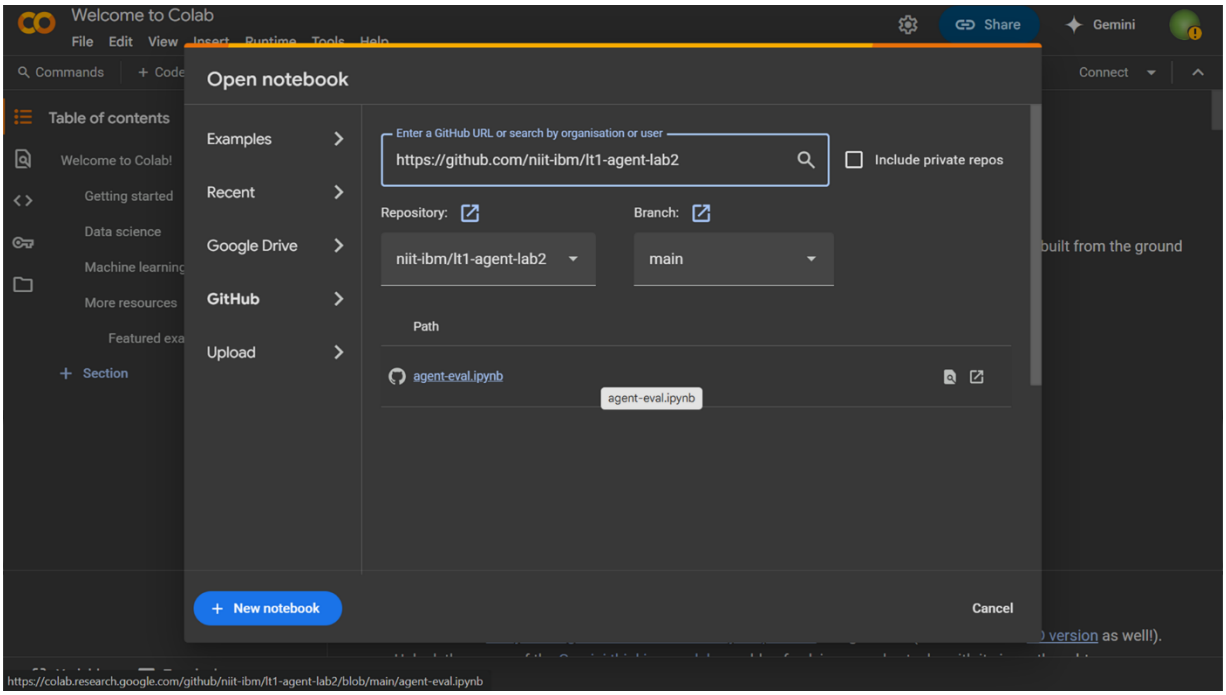
2. Select **Open notebook**.



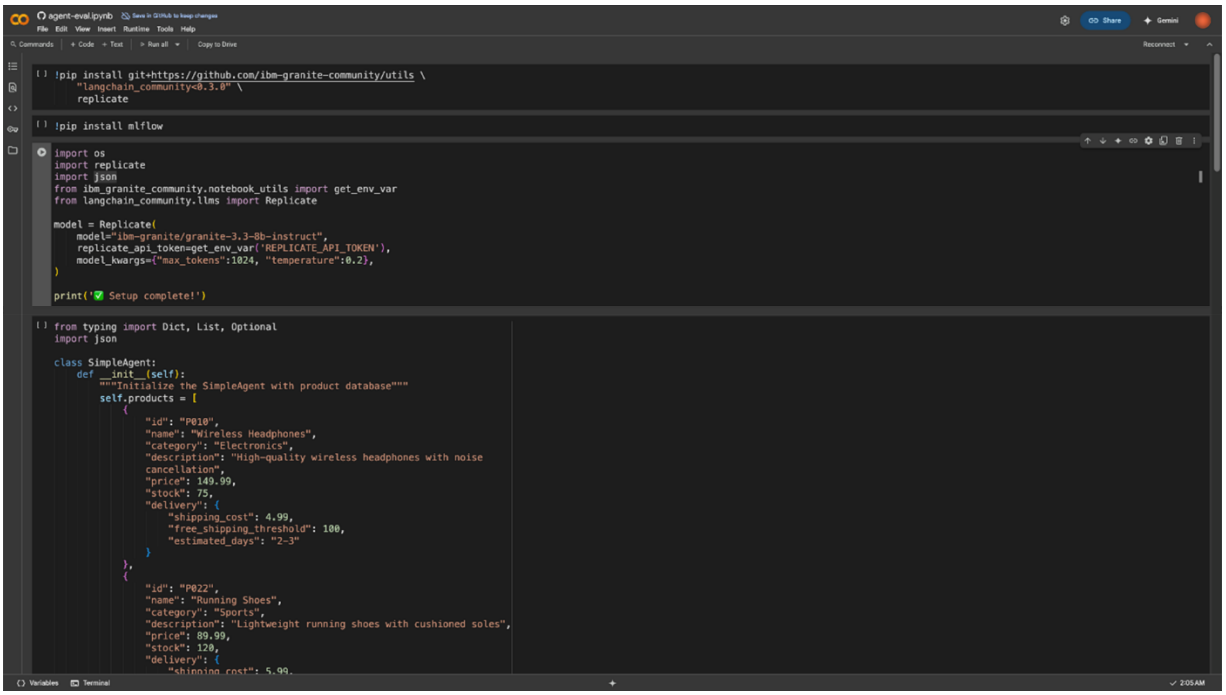
3. In the “Open notebook” dialog, select **GitHub**.
   To find and load the Jupyter notebook directly in Google Colab, type the uniform resource locator (URL) https://github.com/niit-ibm/lt1-agent-lab2 in the **Enter a GitHub URL or search by organization or user** field. Then, press Enter.



4. After you enter the URL, the **Repository** and **Branch** fields are automatically populated. To open the notebook, select **agent-eval.ipynb**.



The agent-eval.ipynb notebook opens in the Colab workspace, as shown in the following screenshot.

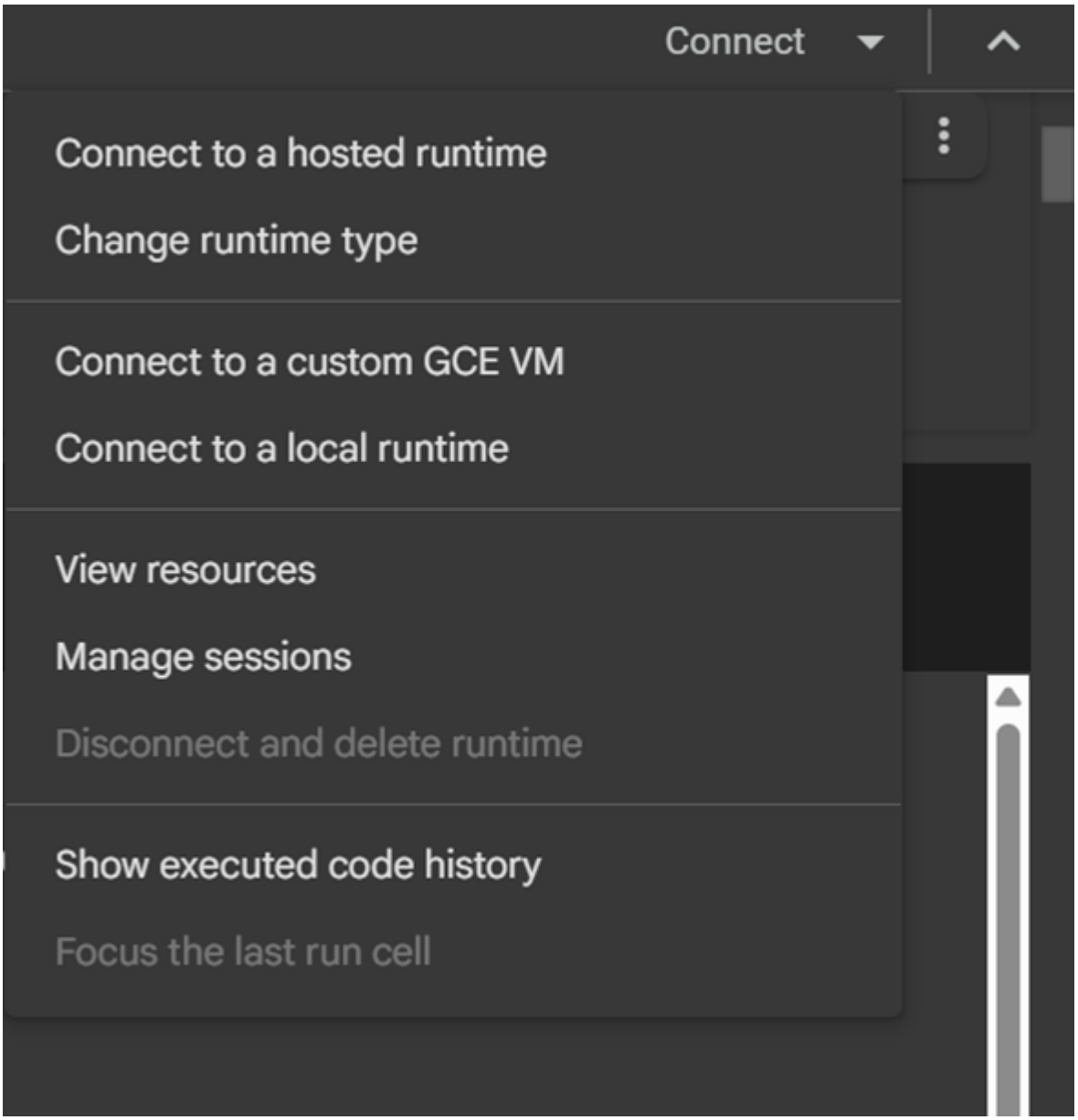


5. To run the code, select the **Connect** drop-down arrow.

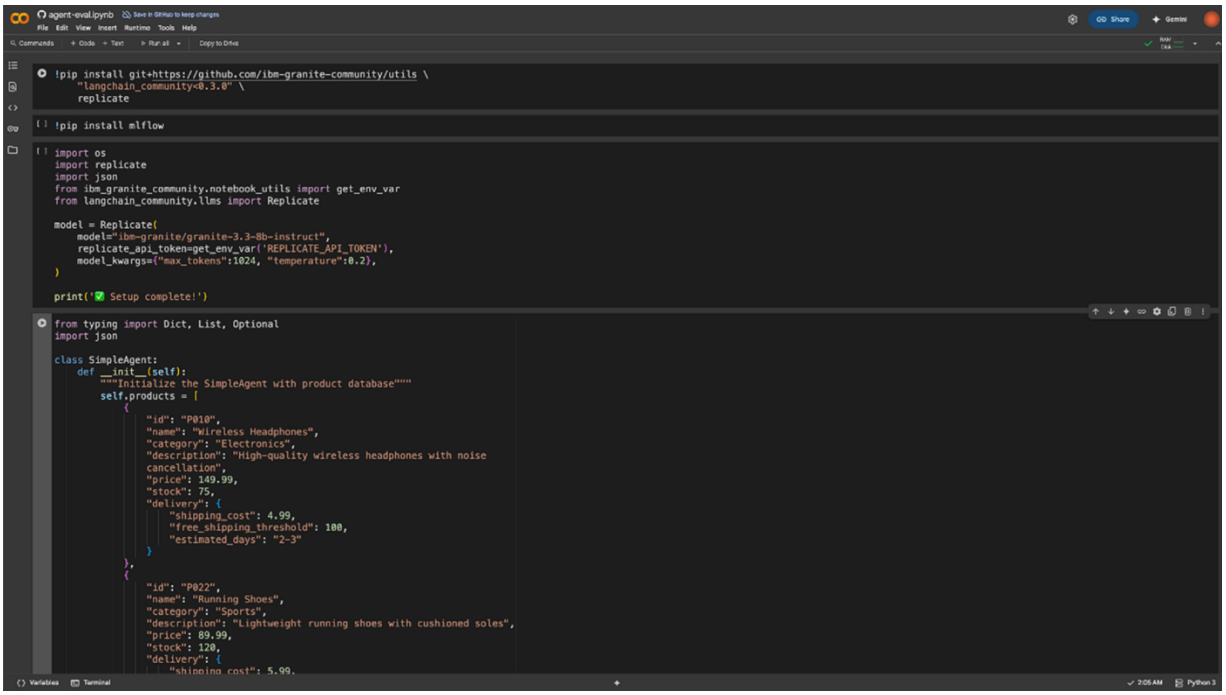
6. From the list, select **Connect to a hosted runtime.**

The following screenshot shows a Jupyter notebook with the Connect to a hosted runtime option in the Connect list.

Note: When you use Colab, your code doesn't run on your local machine. Instead, it runs in a cloud-based environment provided by Google, known as a **hosted runtime**. This means you don't need to install Python or any libraries on your device; they're already available in the cloud environment.

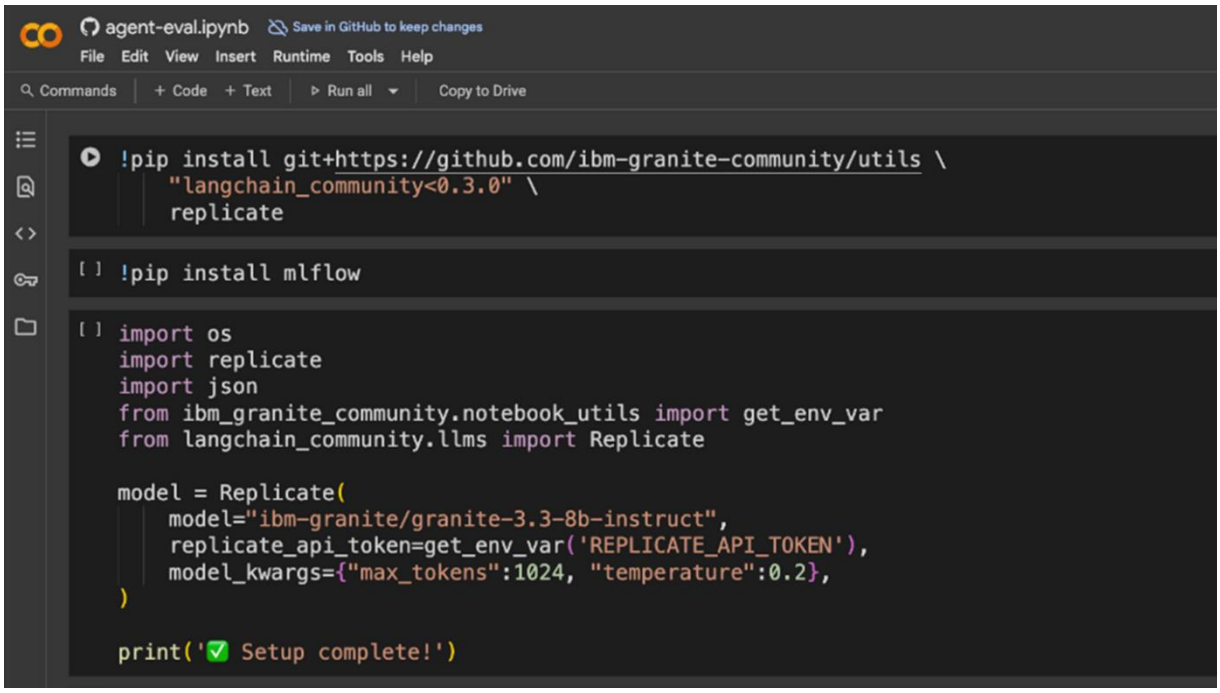


Check the **Colab** toolbar for a green checkmark, as shown in the following screenshot. This indicates that the connection is successful, and your code is ready to run.

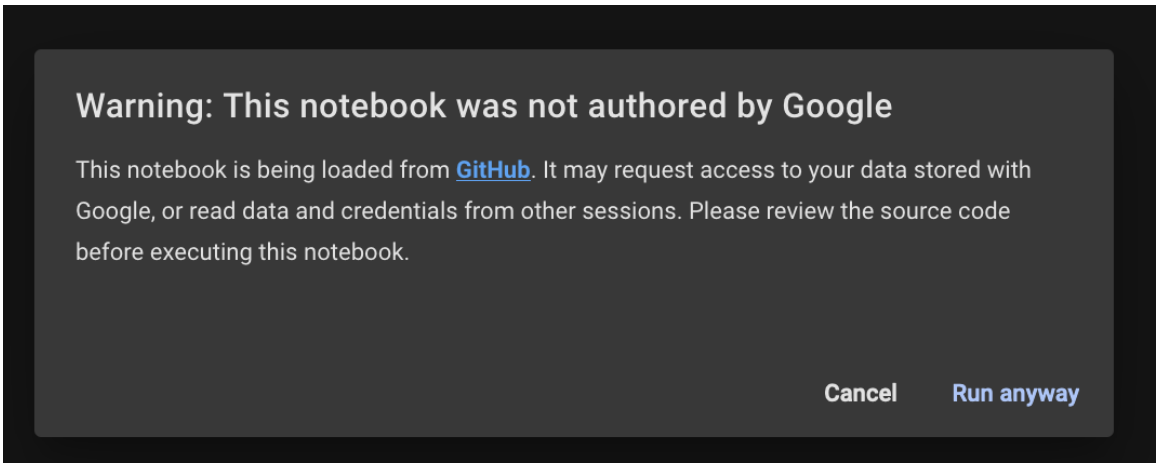


7. To enable your AI agent to work with the IBM Granite model and generate helpful responses, certain libraries must be installed. In the first cell, select the **Run** icon to begin the installation of the libraries.

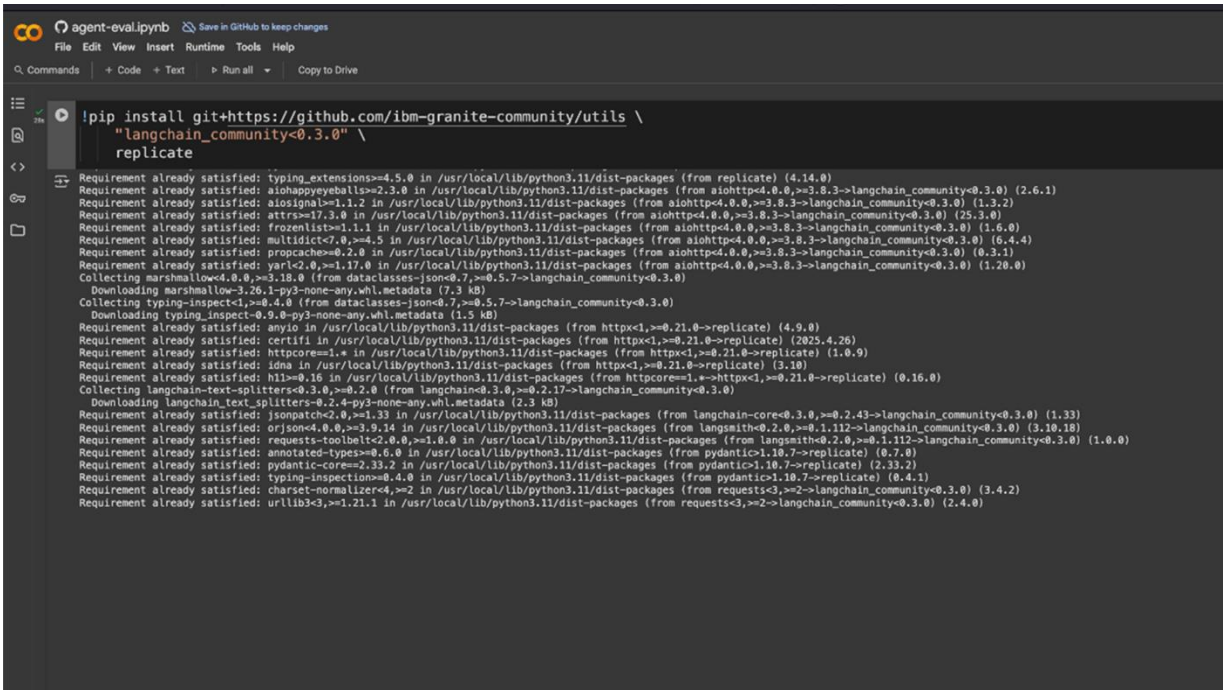
Note: In a Jupyter notebook, each row is called a **cell**. These cells are not numbered but are organized to run sequentially, beginning from the first cell. Each code cell includes a **Run** icon (▶), which you use to execute the code within that cell.



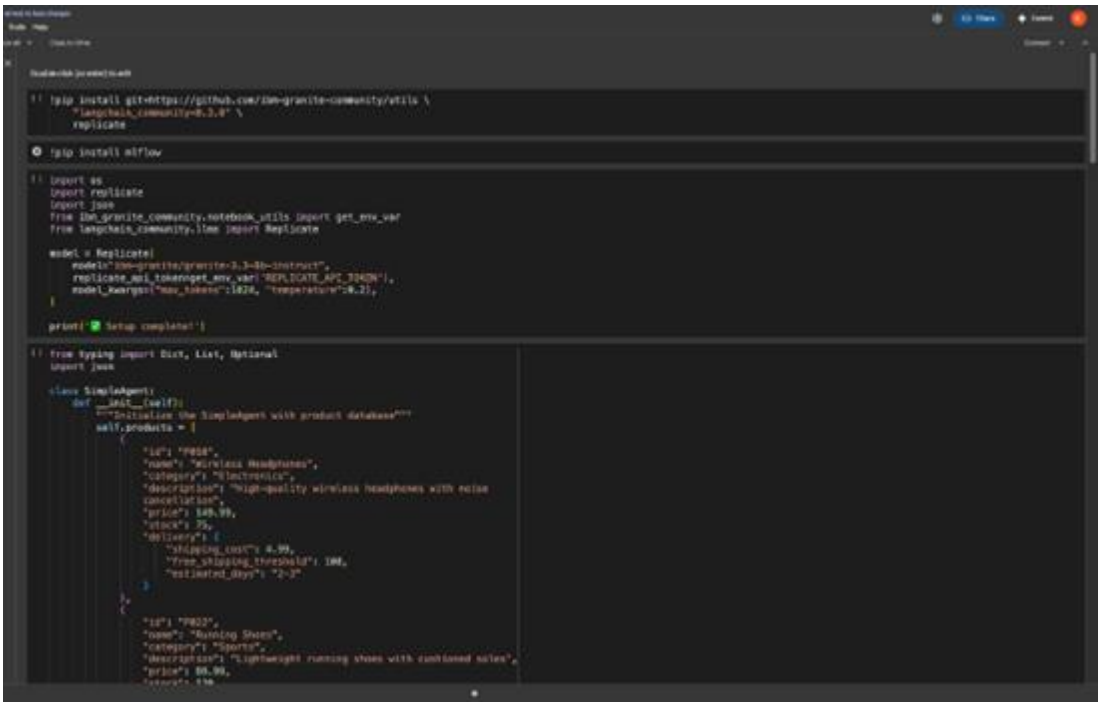
8. Select **Run anyway** to continue loading the required libraries.



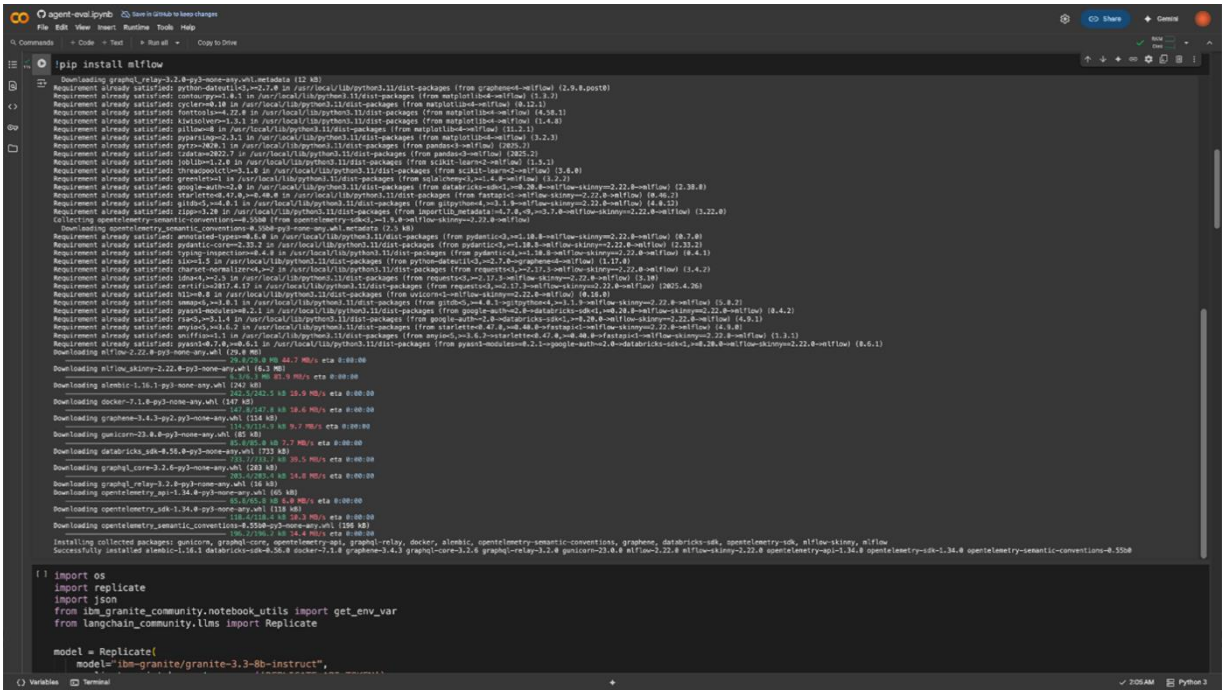
A green checkmark is displayed next to the **Run** icon, indicating that the required libraries have been successfully installed.



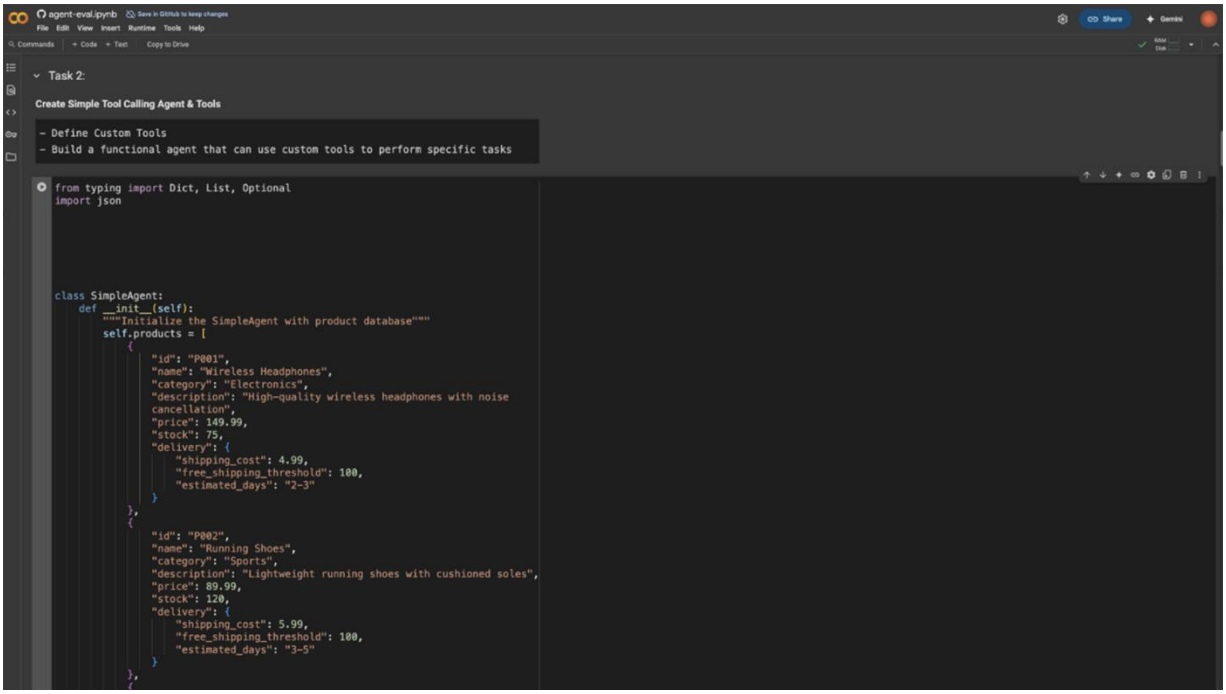
9. To install MLflow, select the **Run** icon in the second cell, as shown in the following screenshot.



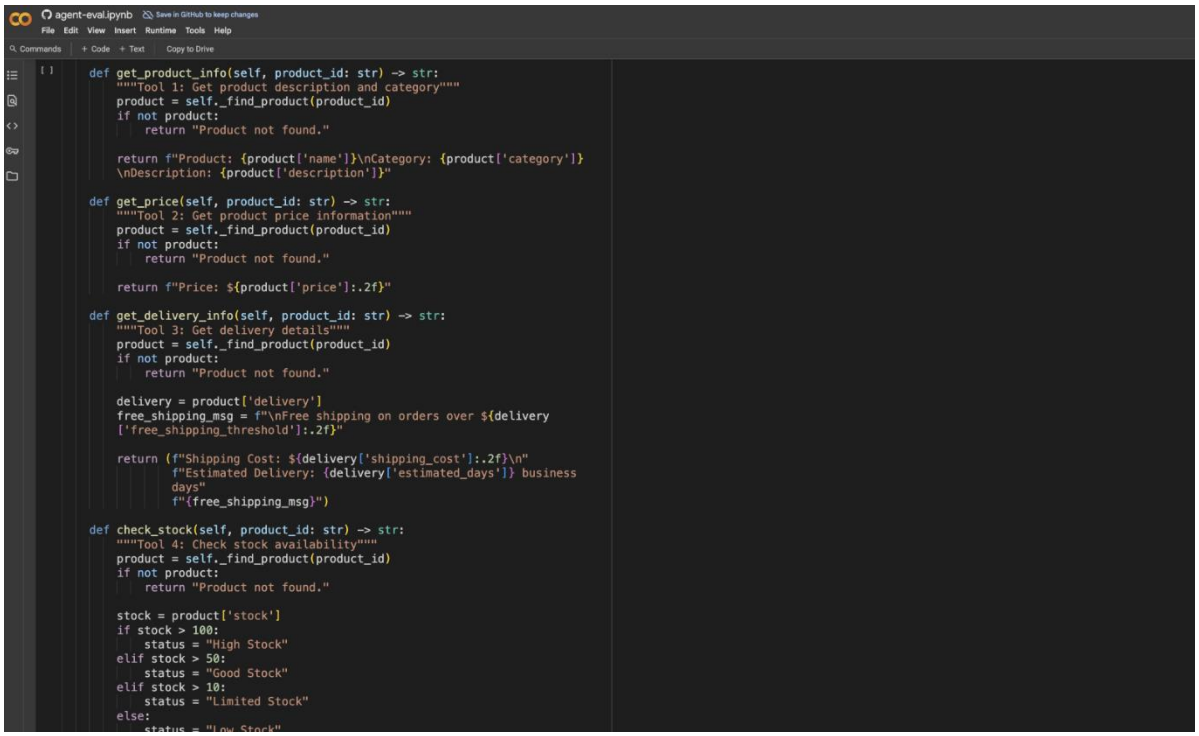
A green checkmark is displayed next to the **Run** icon, confirming that MLflow is successfully installed.



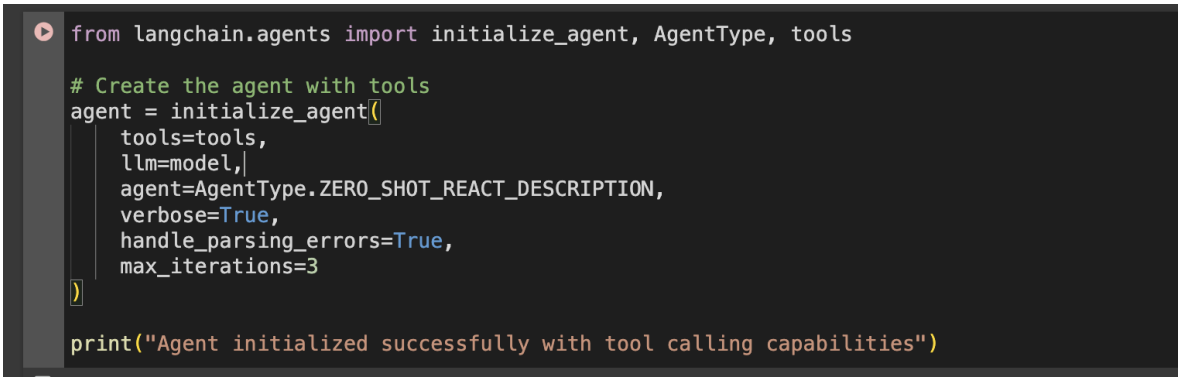
10. To create a database for TechMart's product inventory, select the **Run** icon in the first cell of Task 2, as shown in the following screenshot.



11. To enable the agent to process user queries and generate meaningful responses, define the required tools such as **get_product_info**, **get_price, get_delivery_info**, and **check_stock,** as shown in the following screenshot.



12. To initialize the agent, select the **Run** icon, as shown in the following screenshot. The agent analyzes the user's input to identify whether it's asking for a product description or cost. Based on this, it uses the appropriate tools. For other types of input, the agent generates a general response.



Evaluate the trajectory and final response of the AI agent

Overview

In this procedure, you'll evaluate the agent's trajectory and assess the accuracy, clarity, and relevance of its final responses using MLflow.

Instructions

1. To test how the agent responds, select the **Run** icon to invoke the **try_agent** function.

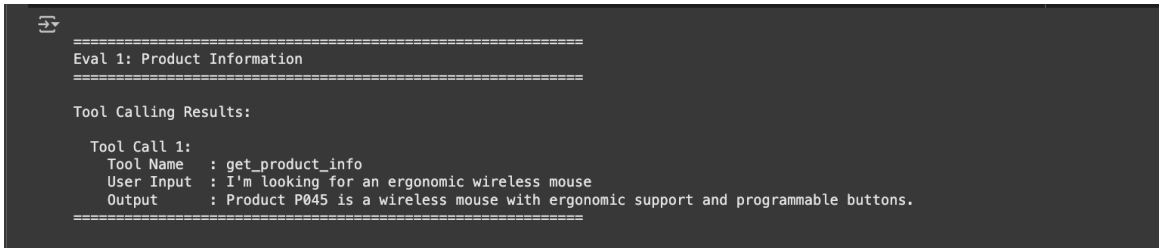
   Note: The try_agent function captures and logs the agent's response, helping determine if the agent correctly analyzes the user's query, and invokes the appropriate tool. It validates whether the agent's behavior aligns with the expected context-driven tool usage and response generation.

```
def try_agent(example_name: str, user_input: str):
    with mlflow.start_run(run_name=example_name):
        mlflow.log_param("user_input", user_input)

        # Simulate agent response
        response = try_agent(user_input)

        mlflow.log_param("agent_response", response)
        print(f"\n{example_name}")
        print("Input:", user_input)
        print("Response:", response)

        # Optionally log output to a text file
        with open(f"{example_name.replace(' ', '_')}_output.txt", "w") as f:
            f.write(f"Input: {user_input}\nResponse: {response}")
        mlflow.log_artifact(f"{example_name.replace(' ', '_')}_output.txt")
```

2. To test how the agent interprets a user query and uses the appropriate tool, select the **Run** icon to execute the code as shown in the following screenshot.

```
# Agent Evaluation tests starts here!
try_agent(
    query="I'm looking for an ergonomic wireless mouse",  # Your question
    tool=agent_tool
)
```

3. Review the tool-calling results to determine whether the agent correctly interprets the user's query, "I'm looking for an ergonomic wireless mouse," and selects the right tools to generate a response.



```
============================================================
Eval 1: Product Information
============================================================

Tool Calling Results:

  Tool Call 1:
    Tool Name   : get_product_info
    User Input  : I'm looking for an ergonomic wireless mouse
    Output      : Product P045 is a wireless mouse with ergonomic support and programmable buttons.
============================================================
```

4. To evaluate whether the agent can handle follow-up questions by using multiple tools while retaining the user's original query and context, select the **Run** icon to execute the code as shown in the following screenshot.

```
try_agent(
    query="Is the product available in stock?",  # Your question
    tool=agent_tool
)
```

5. Review the output showing how the agent handles the query, "Is the product available in stock?" It uses tools, such as get_product_info and check_inventory, to generate a response based on the query.

```
============================================================
Eval 1: Product Information
============================================================

Tool Calling Results:

  Tool Call 1:
    Tool Name   : get_product_info
    User Input  : I'm looking for an ergonomic wireless mouse
    Output      : Product P045 is a wireless mouse with ergonomic support and programmable buttons.

  Tool Call 2:
    Tool Name   : check_inventory
    User Input  : Is the product available in stock?
    Output      : Product P045 is currently available in your selected region.
============================================================
```

6. For the final agent evaluation, select the **Run** icon.

Task 4:

Final Response Evaluation

- Evaluating Final responses from agent against the defined metrics

```
def try_agent(example_name: str, user_input: str):
    with mlflow.start_run(run_name=example_name):
        mlflow.log_param("user_input", user_input)

        # Simulate agent response
        response = try_agent(user_input)

        mlflow.log_param("agent_response", response)
        # Evaluate the response
          evaluation = evaluate_response(query, tool, response)

          # Print results
          print(f"Query: {query}")
          print(f"Tool Used: {tool}")
          print(f"Response: {response}")
          print(f"Evaluation Score: {evaluation['score']}/5")
          print(f"Feedback: {evaluation['explanation']}")
```

After the code is executed, the response summary is displayed as shown in the following screenshot. Note that the agent can use tools such as get_product_info and check_inventory successfully and retrieve relevant information in response to user's query.

Note: Even when an agent successfully retrieves relevant information using tools, its final response requires human evaluation for confirmation.

Conclusion

Congratulations! You've effectively evaluated TechMart's AI agent. Through structured testing and analysis with MLflow, you assessed the agent's tool-calling efficiency and the quality of its final responses, ensuring accurate, relevant, and clear delivery of product information.