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

Lab steps
This lab requires you to complete the following steps:

- **Step 1**: Create a GitHub account
- **Step 2:** Create a Replicate account
- **Step 3:** Sign up for Google Colab
- **Step 4:** Initialize the AI agent and tools
- **Step 5:** 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.

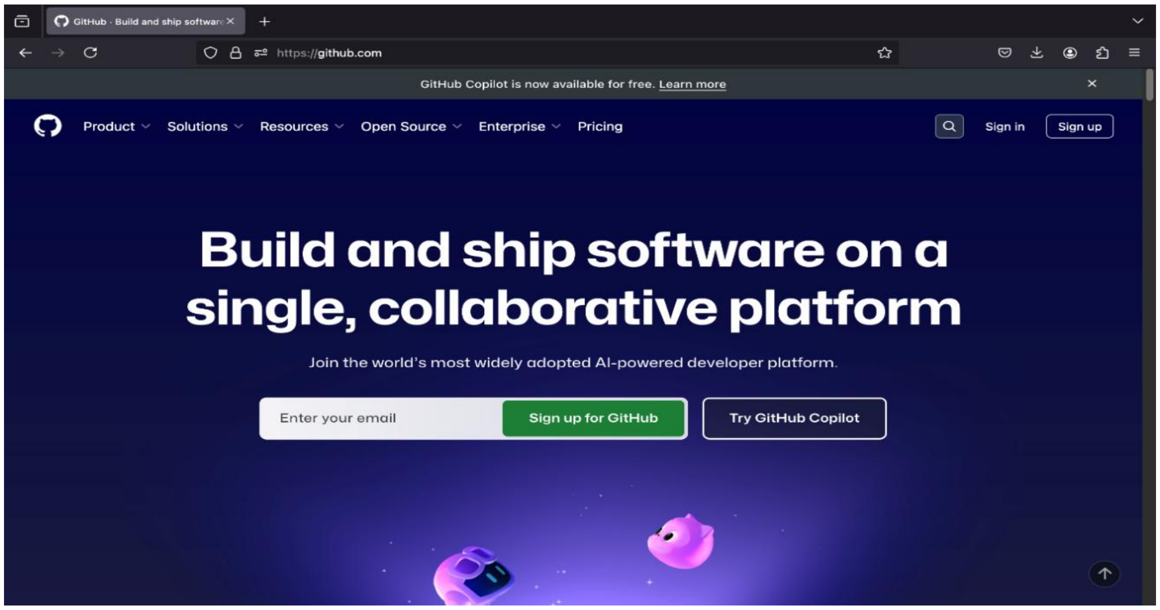
Challenge
TechMart users often leave the site when the AI agent fails to provide relevant product recommendations or assist effectively with catalog navigation. Additionally, missing or inconsistent details about products, pricing, or stock levels can erode the user's trust.

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.

Step 1: Create a GitHub account

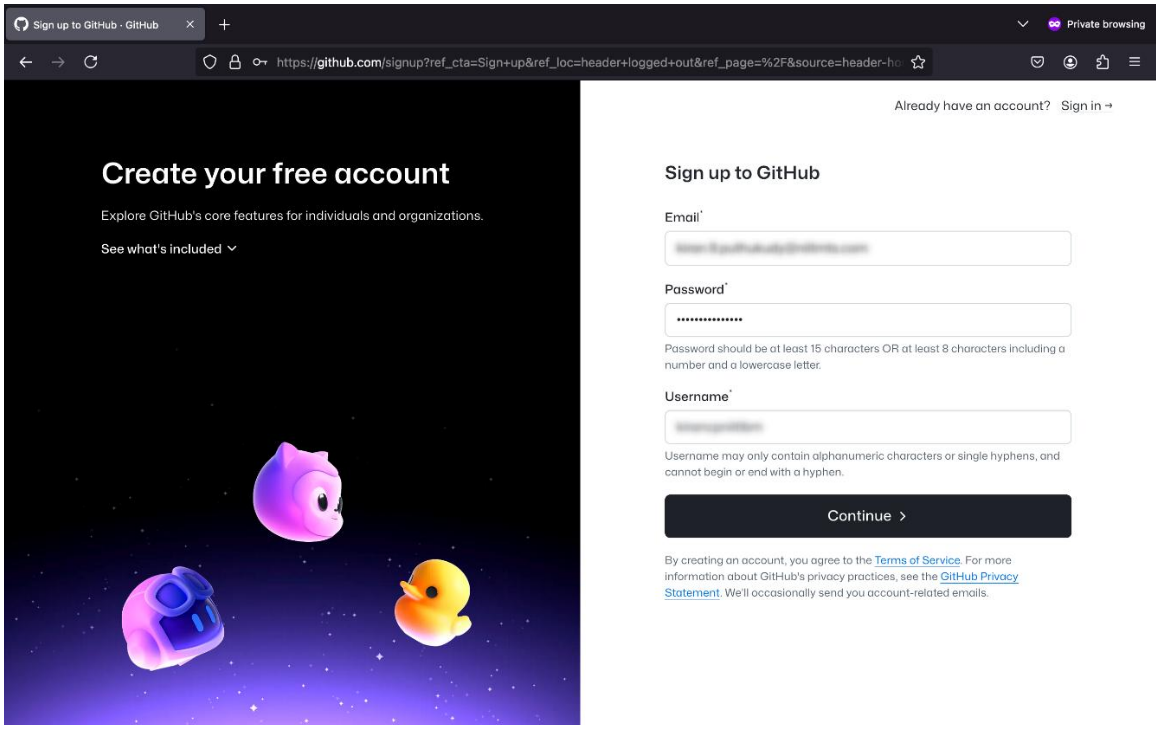
Overview
In this step, you'll set up a GitHub account. GitHub is a platform that helps developers store, manage, and share code, while also supporting collaboration through tools such as version control, bug tracking, and task management. Setting up a GitHub account ensures access to the Replicate platform,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

Instructions
1. To create a GitHub account, go to the GitHub website and select the **Sign up for GitHub** button.


GitHub · Build and ship software
https://github.com
GitHub Copilot is now available for free. Learn more
Product
Solutions
Resources
Open Source
Enterprise
Pricing
Sign in
Sign up
Build and ship software on a single, collaborative platform
Join the world's most widely adopted AI-powered developer platform.
Enter your email
Sign up for GitHub
Try GitHub Copilot

Alt text: An image showing the home page of the GitHub website with the **Sign up for GitHub** button

2. Enter your details in the **Email**, **Password**, and **Username** fields. Then, select **Continue**.


Sign up to GitHub · GitHub
Private browsing
https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-ho
Already have an account? Sign in →
Create your free account
Explore GitHub's core features for individuals and organizations.
See what's included
Continue >

Sign up to GitHub

Email*

Password*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

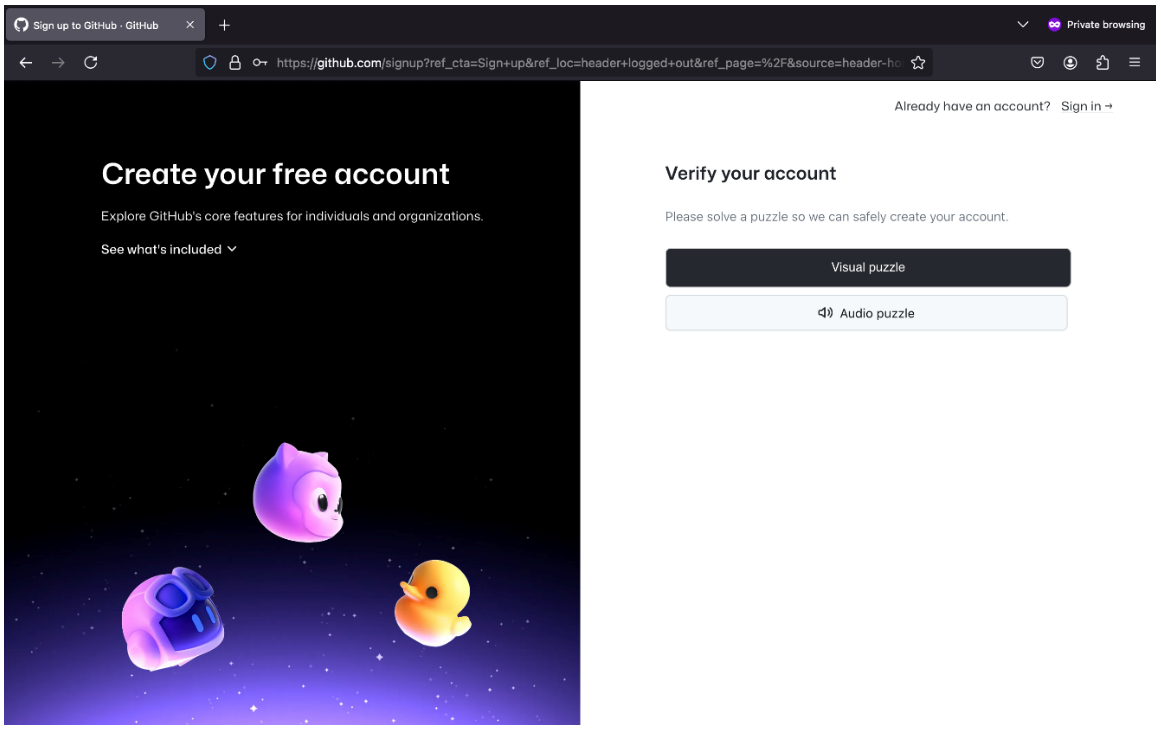
Username*

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

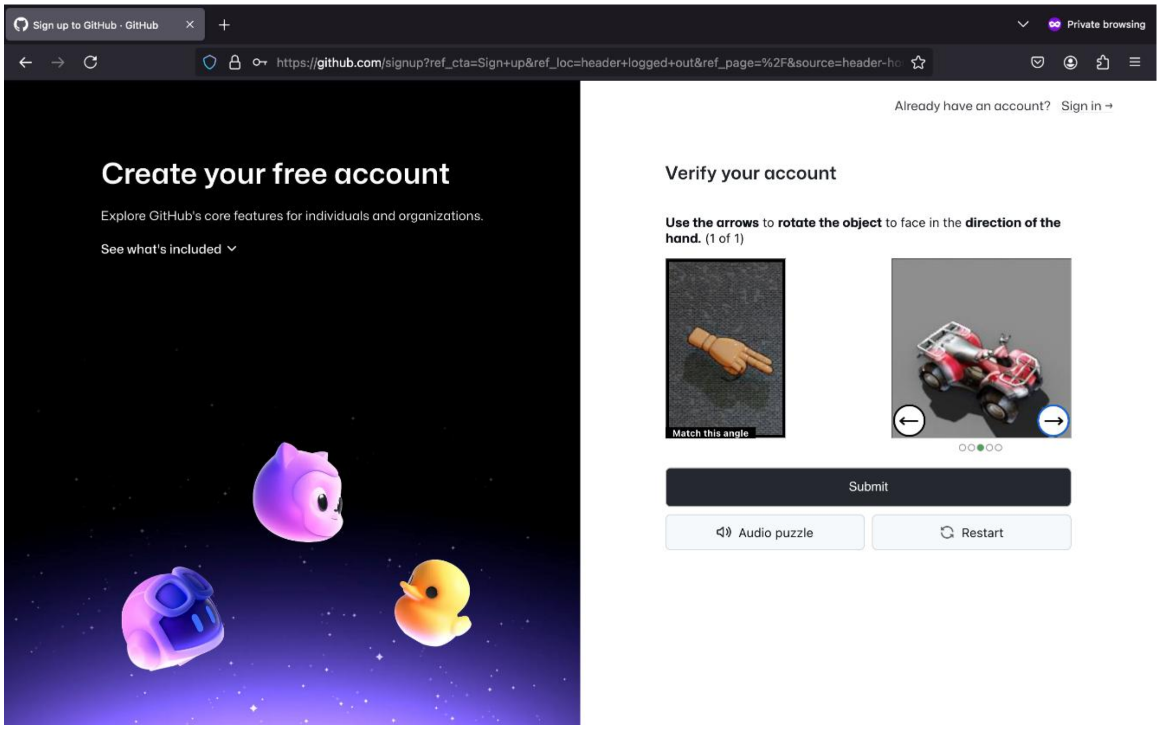
Alt text: An image showing the "GitHub sign-up" page with the **Email**, **Password**, and **Username** fields

3. To verify your account, select the **Visual puzzle** option.


Sign up to GitHub · GitHub
Private browsing
https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-ho
Already have an account? Sign in →
Create your free account
Explore GitHub's core features for individuals and organizations.
See what's included
Verify your account
Please solve a puzzle so we can safely create your account.
Visual puzzle
Audio puzzle

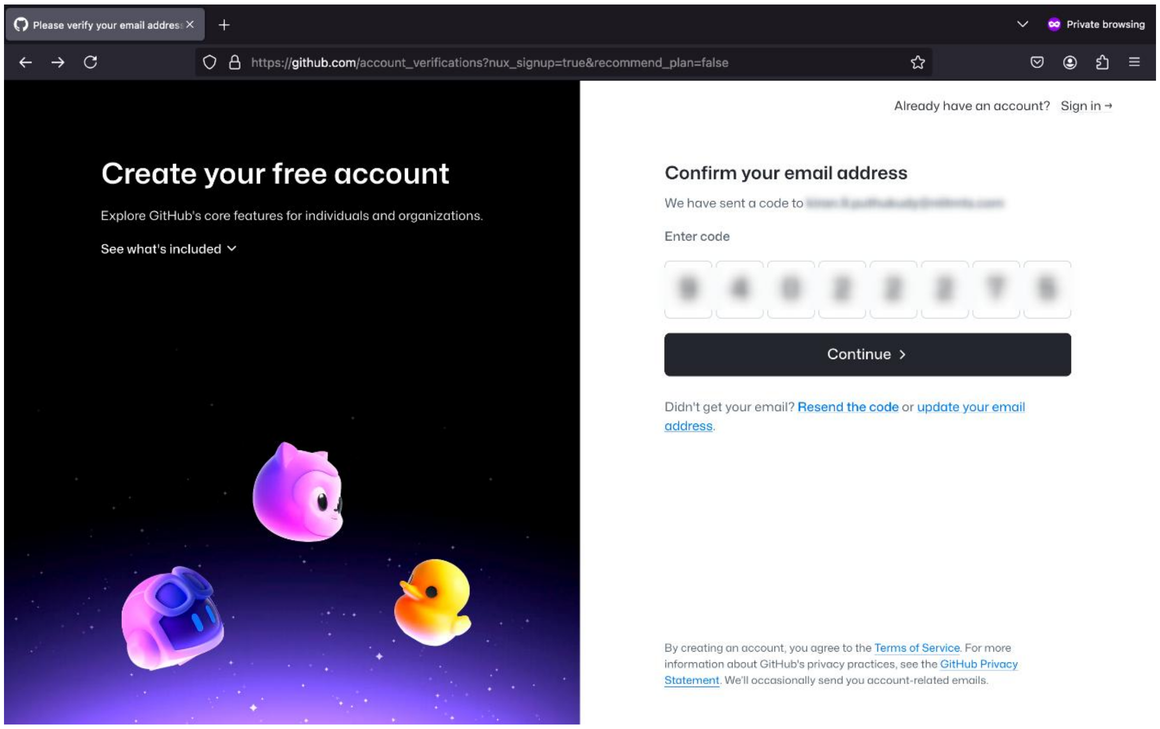
Alt text: An image showing the "GitHub sign-up" page and the visual and audio puzzle verification step

4. Next, solve the visual puzzle and select **Submit**.


Sign up to GitHub · GitHub
Private browsing
https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-ho
Already have an account? Sign in →
Create your free account
Explore GitHub's core features for individuals and organizations.
See what's included
Verify your account
Use the arrows to rotate the object to face in the direction of the hand. (1 of 1)
Match this angle
Submit
Audio puzzle
Restart

Alt text: An image showing the “GitHub sign-up” page with visual and audio puzzles and the **Submit** button

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


Please verify your email address
Private browsing
https://github.com/account_verifications?nux_signup=true&recommend_plan=false
Already have an account? Sign in →
Create your free account
See what's included
Enter code
Continue ›

Explore GitHub's core features for individuals and organizations.

Confirm your email address

We have sent a code to [redacted]

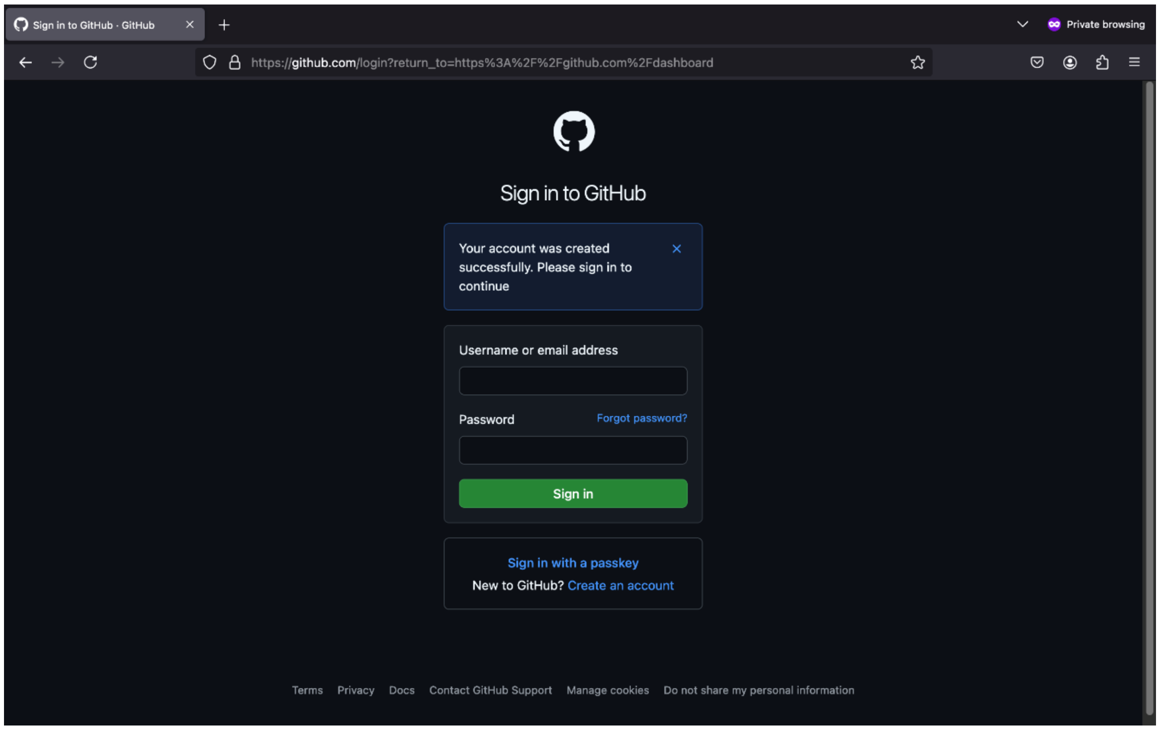
[redacted]

Didn't get your email? Resend the code or update your email address.

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Alt text: An image showing the “GitHub sign-up" page with the email confirmation step

6. You'll see a confirmation message after your GitHub account has been successfully created.



Sign in to GitHub · GitHub

https://github.com/login?return_to=https%3A%2F%2Fgithub.com%2Fdashboard

Sign in to GitHub

Your account was created successfully. Please sign in to continue

Username or email address

Password Forgot password?

Sign in

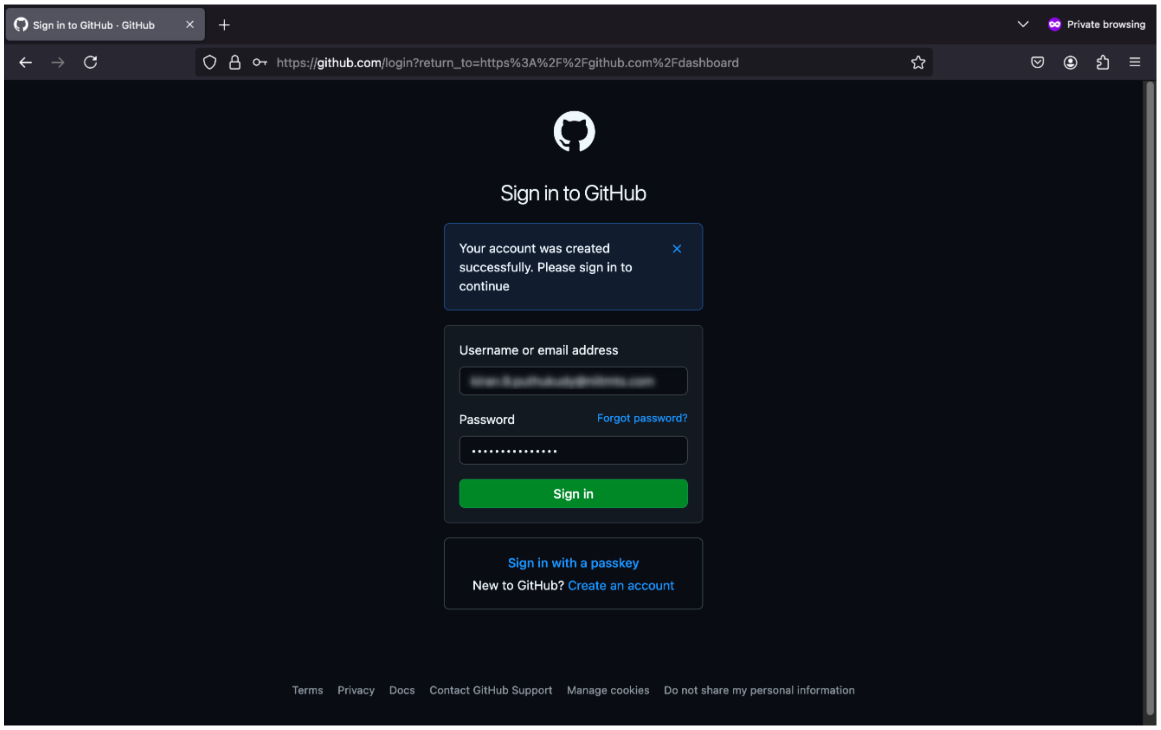
Sign in with a passkey

New to GitHub? Create an account

Private browsing
Terms Privacy Docs Contact GitHub Support Manage cookies Do not share my personal information

Alt text: An image showing the "GitHub sign-in" page with a message indicating that the account has been successfully created

7. To sign in to your account, enter your credentials in the **Username or email address** and **Password** fields, and then select **Sign in**.


Sign in to GitHub · GitHub
Private browsing
https://github.com/login?return_to=https%3A%2F%2Fgithub.com%2Fdashboard
Sign in to GitHub
Username or email address
Password
Forgot password?
Sign in
Sign in with a passkey
New to GitHub? Create an account
Terms
Privacy
Docs
Contact GitHub Support
Manage cookies
Do not share my personal information

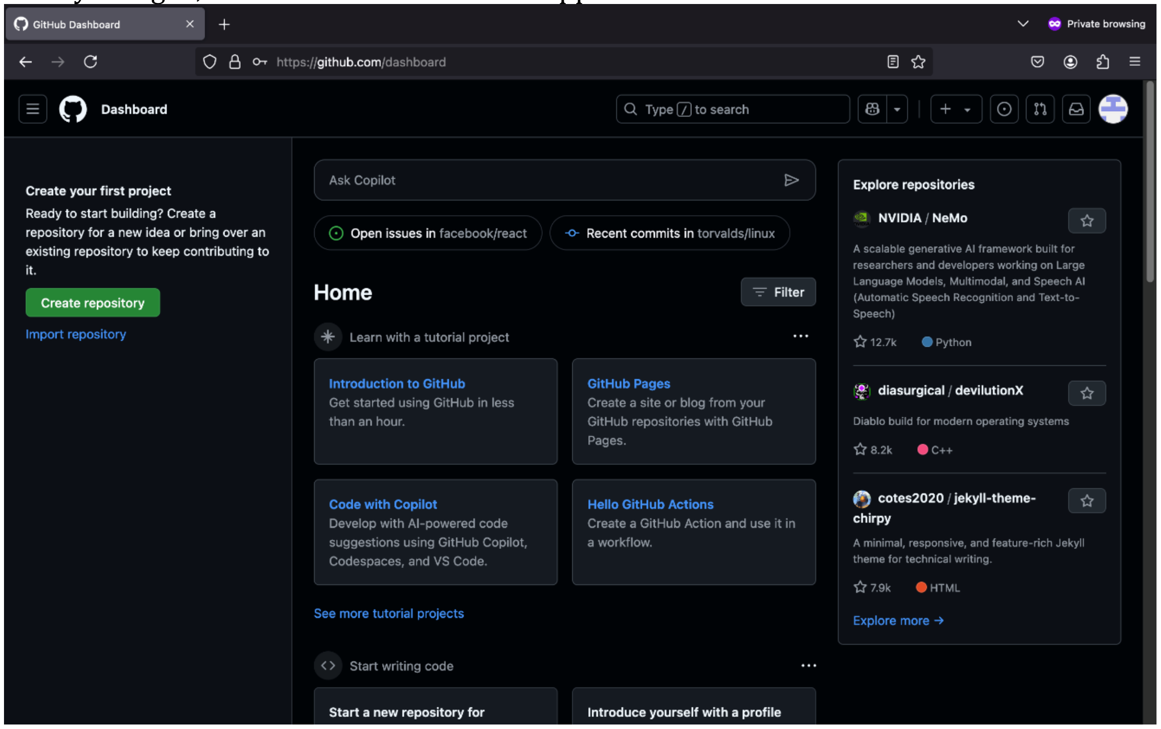
Your account was created successfully. Please sign in to continue

Username or email address: [redacted]

Password: ················

Alt text: An image showing the "GitHub sign-in" page with the **Username or email address** and **Password** fields, and the **Sign in** button

8. After you log in, the GitHub dashboard will appear.


GitHub Dashboard
Private browsing
https://github.com/dashboard
Dashboard
Type / to search
Create repository
Import repository
Ask Copilot
Open issues in facebook/react
Recent commits in torvalds/linux
Home
Filter
Learn with a tutorial project
See more tutorial projects
Start writing code
Start a new repository for
Introduce yourself with a profile
Explore repositories
Explore more →

Create your first project
Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Introduction to GitHub
Get started using GitHub in less than an hour.

GitHub Pages
Create a site or blog from your GitHub repositories with GitHub Pages.

Code with Copilot
Develop with AI-powered code suggestions using GitHub Copilot, Codespaces, and VS Code.

Hello GitHub Actions
Create a GitHub Action and use it in a workflow.

NVIDIA / NeMo
A scalable generative AI framework built for researchers and developers working on Large Language Models, Multimodal, and Speech AI (Automatic Speech Recognition and Text-to-Speech)
12.7k Python

diasurgical / devilutionX
Diablo build for modern operating systems
8.2k C++

cotes2020 / jekyll-theme-chirpy
A minimal, responsive, and feature-rich Jekyll theme for technical writing.
7.9k HTML

Alt text: An image showing the GitHub dashboard with options to create a project, explore repositories, access learning resources and interact with various elements

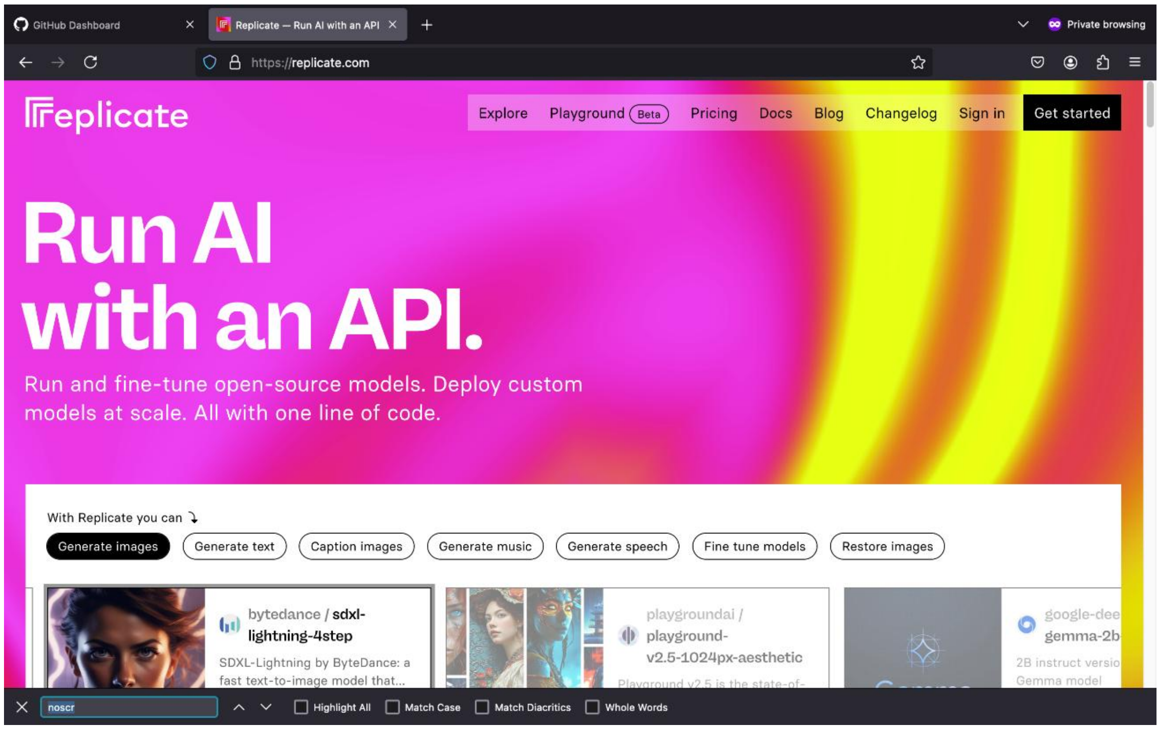
Step 2: Create a Replicate account

Overview

In this step, 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 step, 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.

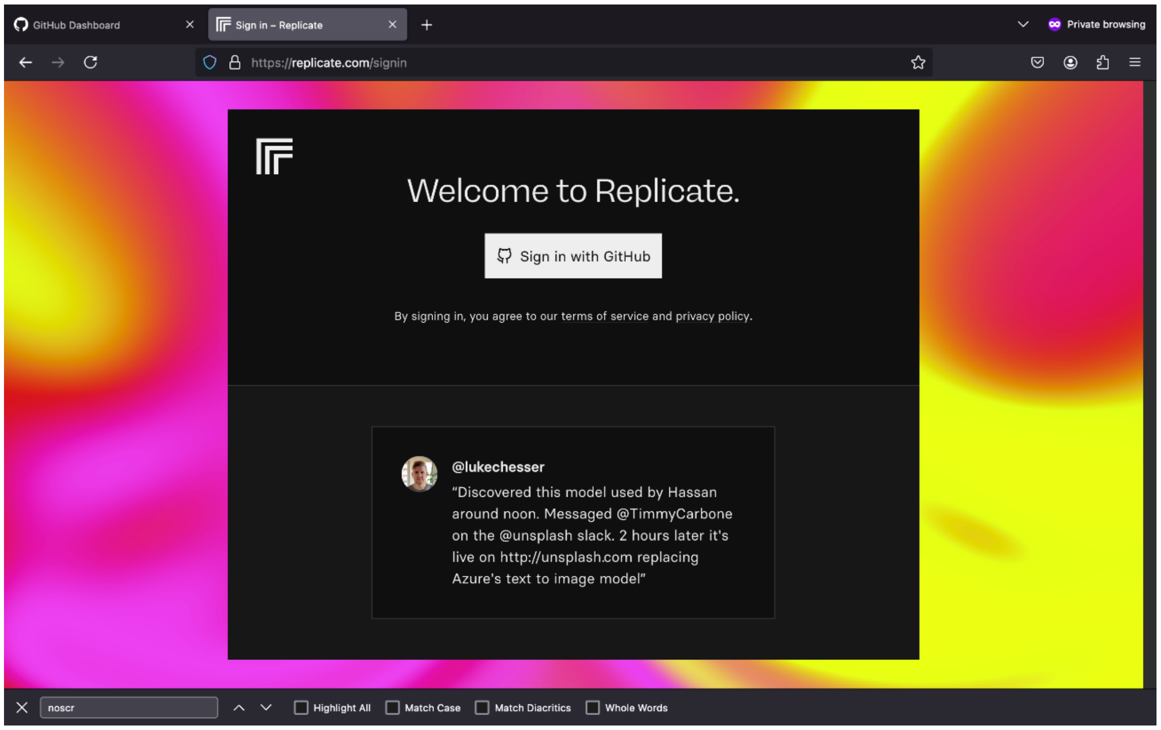
Instructions

9. Go to the Replicate website and select **Get started**.



Alt text: An image showing the Replicate website with the **Get started** button

10. Select the **Sign in with GitHub** option on the Welcome to Replicate page.

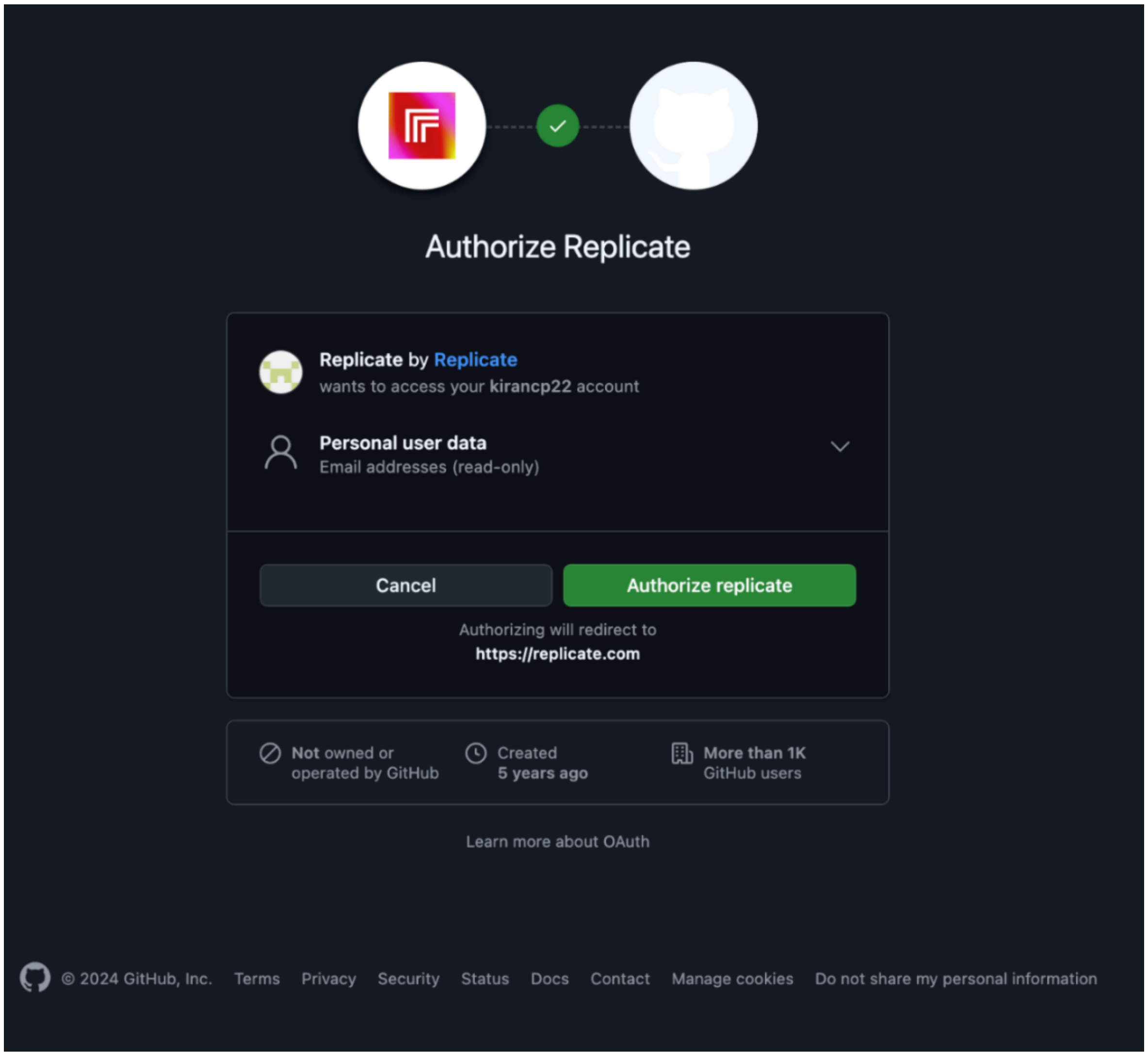


By signing in, you agree to our terms of service and privacy policy.

@lukechesser
"Discovered this model used by Hassan around noon. Messaged @TimmyCarbone on the @unsplash slack. 2 hours later it's live on http://unsplash.com replacing Azure's text to image model"

Alt text: An image showing the Welcome to Replicate page with the **Sign in with GitHub** option

11. Select **Authorize replicate** to continue.



Authorize Replicate

Replicate by Replicate
wants to access your kirancp22 account

Personal user data
Email addresses (read-only)

Authorizing will redirect to
https://replicate.com

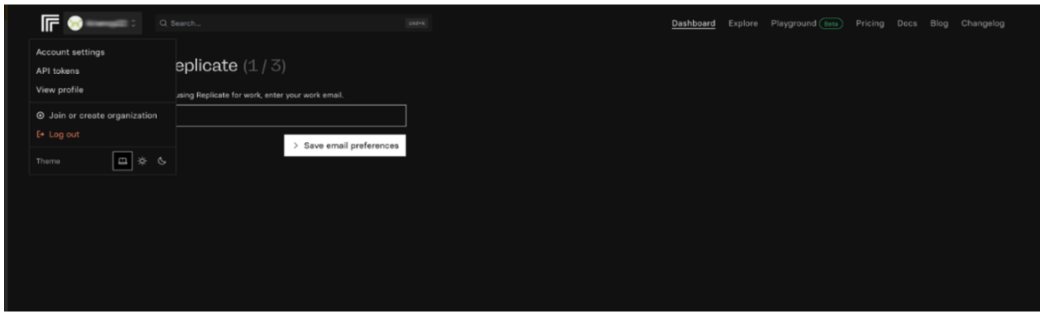
Not owned or operated by GitHub
Created 5 years ago
More than 1K GitHub users

Learn more about OAuth

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

Alt text: An image showing the authorization screen for Replicate, requesting access to the user's account

12. After your Replicate account is created, you'll be taken to the Replicate dashboard. To create a Replicate token, select the **Account settings** option from the navigation bar.

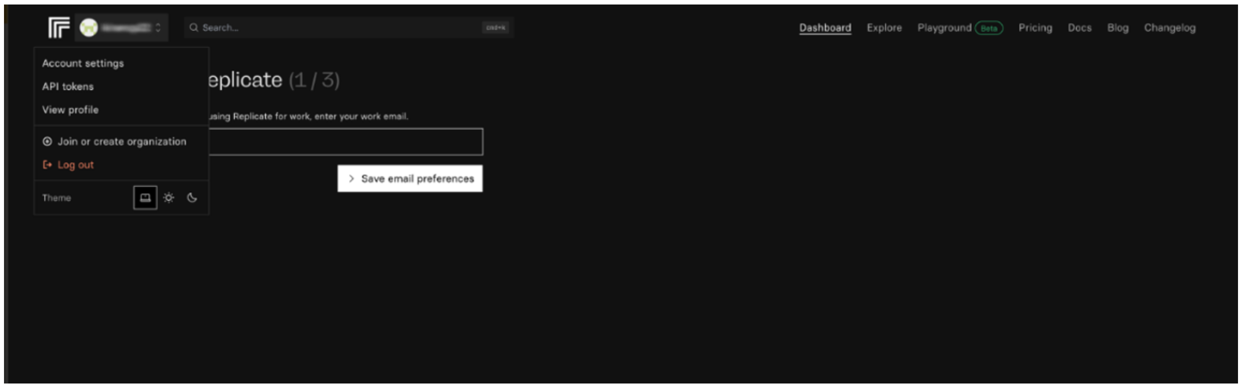


eplicate (1 / 3)

ısing Replicate for work, enter your work email.

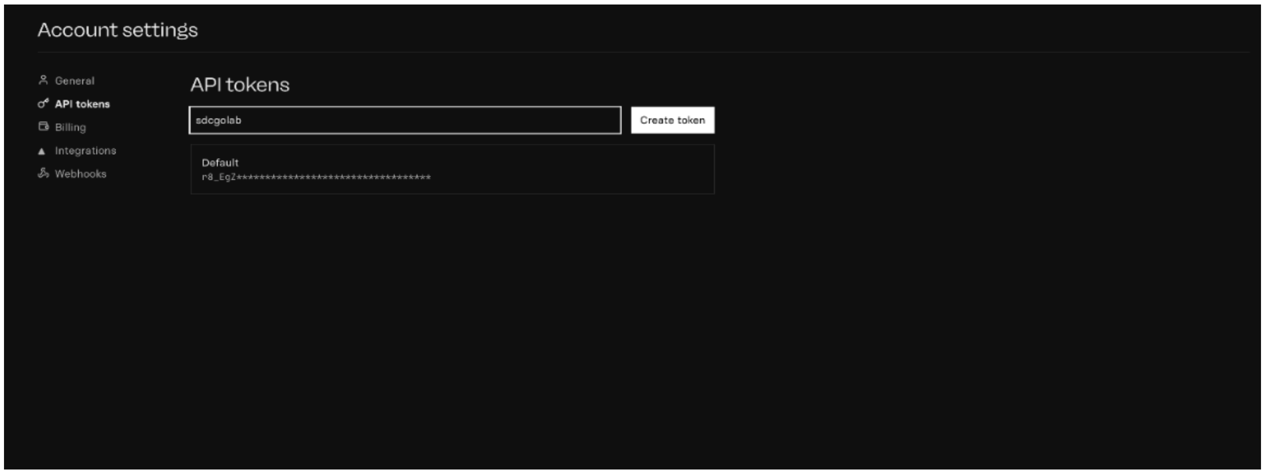
Alt text: An image showing the Replicate dashboard with a drop-down list containing options such as **API tokens**, **View profile,** and **Log out**, and an email prompt with the **Save email preferences** button

13. Select **API tokens** from the **Account settings** menu.



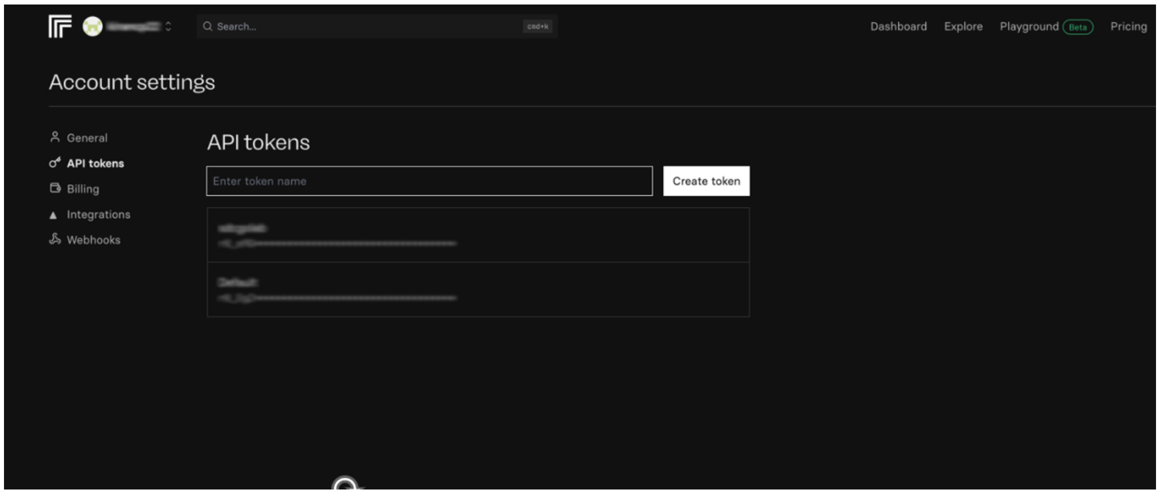
Alt text: An image showing the Replicate dashboard with a drop-down list containing options such as **API tokens**, **View profile**, and **Log out**, and an email prompt with the **Save email preferences** button

14. Enter a name for the token in the **API tokens** field and then select **Create token**.



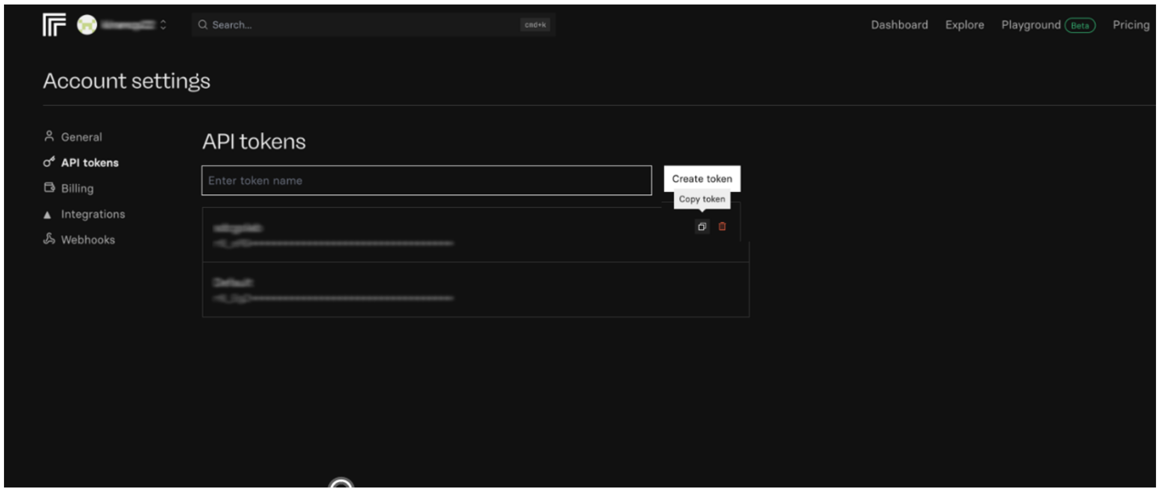
Alt text: An image showing the “Account settings” page displaying the **API tokens** field and a **Create token** button

15. After your API token is created, it should appear on the "Account settings" page.



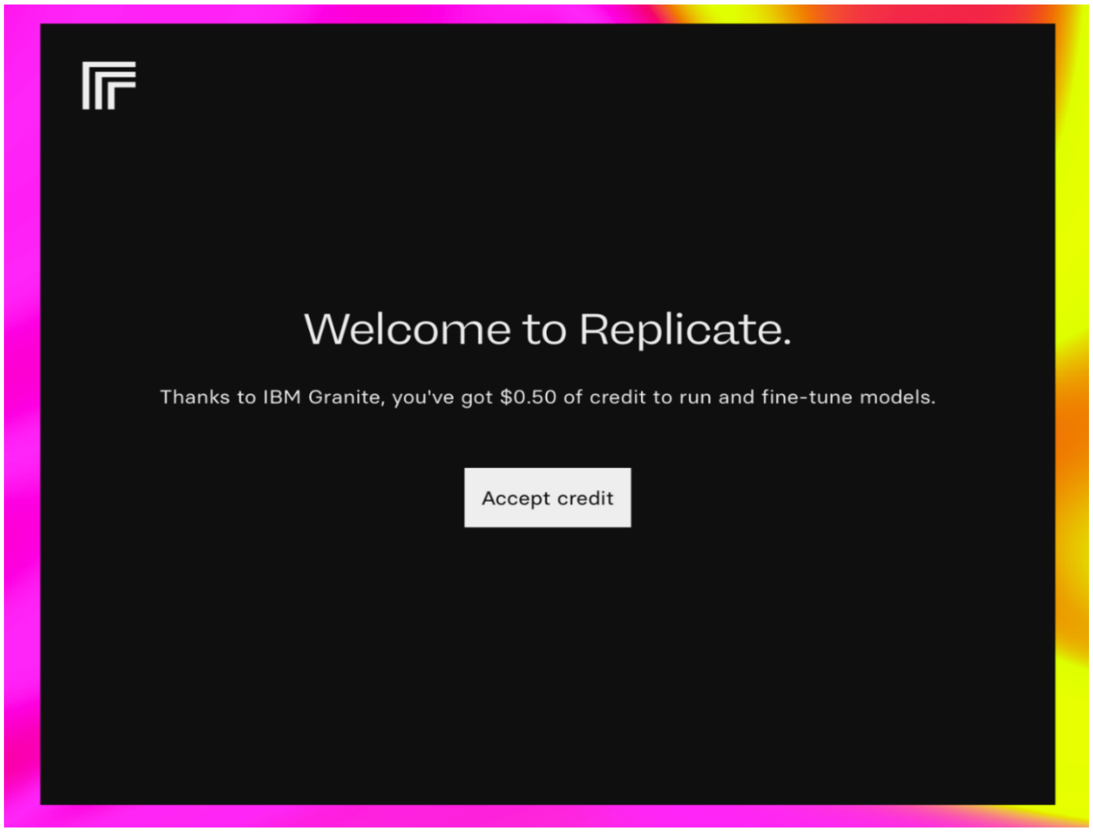
16. Select the **Copy token** icon to copy the Replicate API token.

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.



Alt text: An image showing the "Account settings" page with the **Create token** field and the **Copy token** option

17. 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. Then, select **Accept credit** to claim the amount.

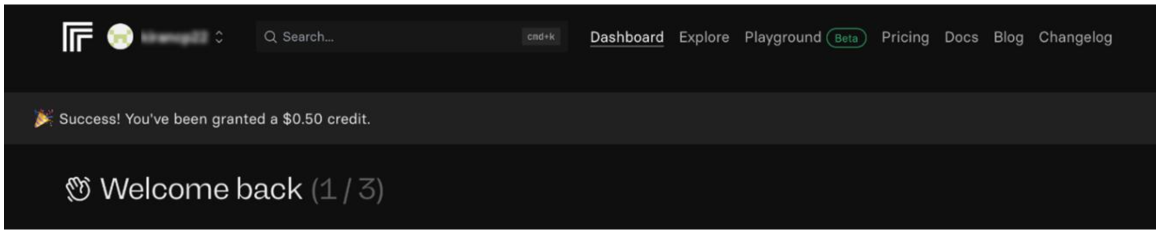


Welcome to Replicate.

Thanks to IBM Granite, you've got $0.50 of credit to run and fine-tune models.

Alt text: An image showing the Welcome to Replicate screen, with an option to accept a $0.50 credit

18. A confirmation message will appear indicating that a $0.50 credit has been added to your Replicate account.



🎉 Success! You've been granted a $0.50 credit.

Welcome back (1 / 3)

Alt text: An image showing the "Welcome back" message and the confirmation message about the $0.50 credit amount

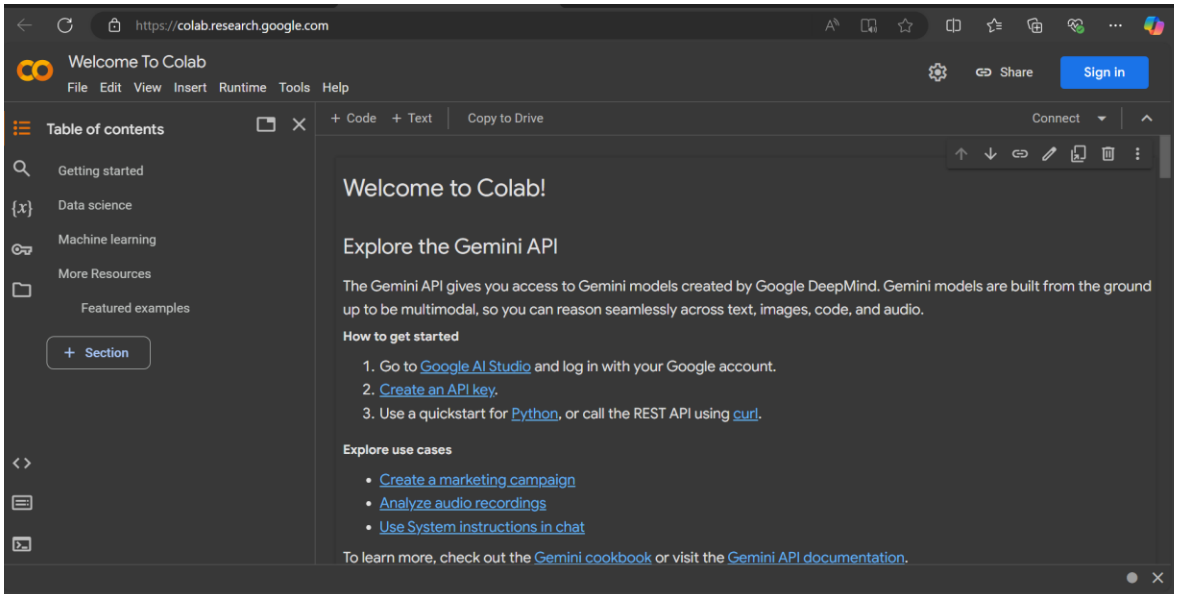
Step 3: Sign up for Google Colab

Overview

In this step, 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 will allow you to install and use the tools needed to complete this lab.

Instructions

19. To sign up, go to the [Google Colab]() website and select **Sign in**.



Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

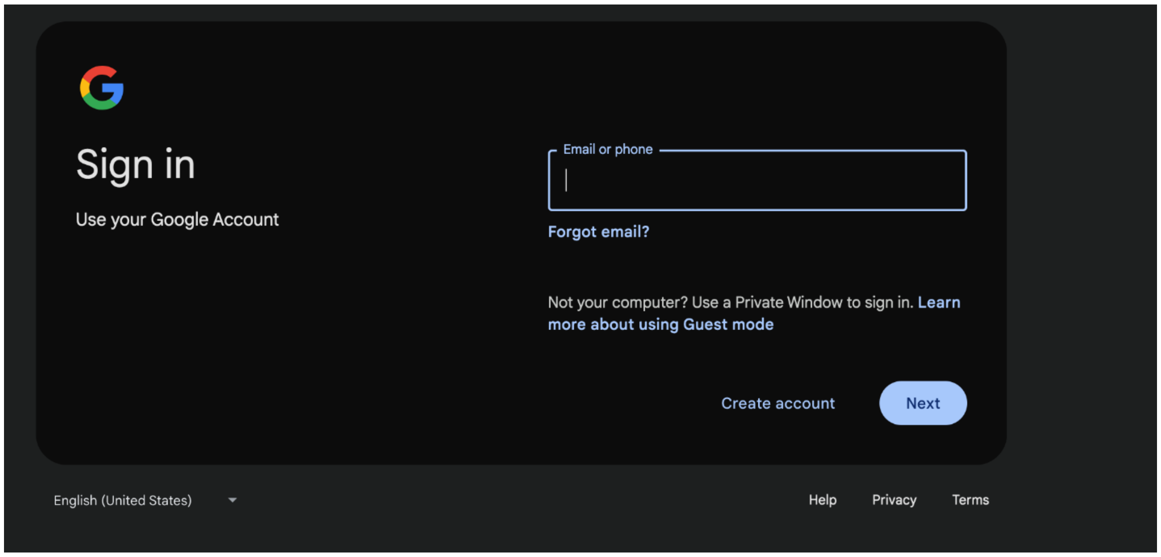
Explore use cases

- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

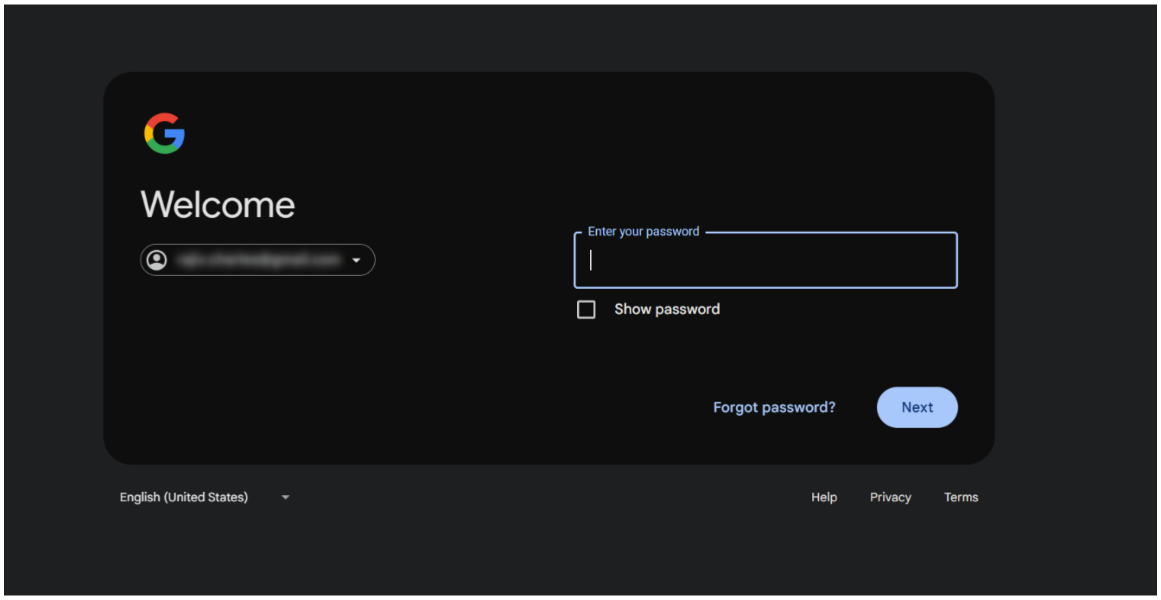
Alt text: An image showing the Welcome to Colab screen, the **Sign in** button, and instructions on how to get started

20. Enter your email or phone number in the **Email or phone** field, and then select **Next**.



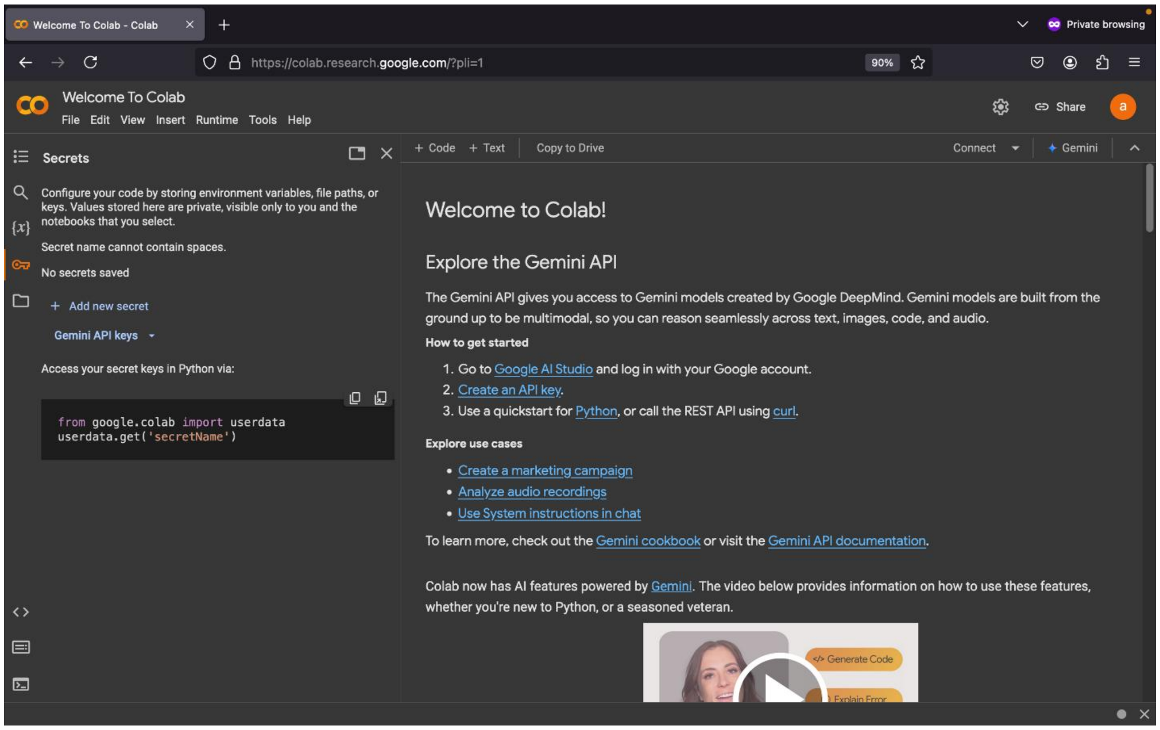
Alt text: An image showing the Google Sign-in page with the **Email or phone** field, the **Create account** option, and the **Next** button

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



Alt text: An image showing the Welcome screen with the **Enter your password** field, the **Forgot password** option, and the **Next** button

22. Next, to store your Replicate API token in the Google Colab **Secrets** tab, select the key icon from the sidebar menu on the Welcome to Colab page.



Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

No secrets saved

Access your secret keys in Python via:

```
from google.colab import userdata
userdata.get('secretName')
```

Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

Explore use cases

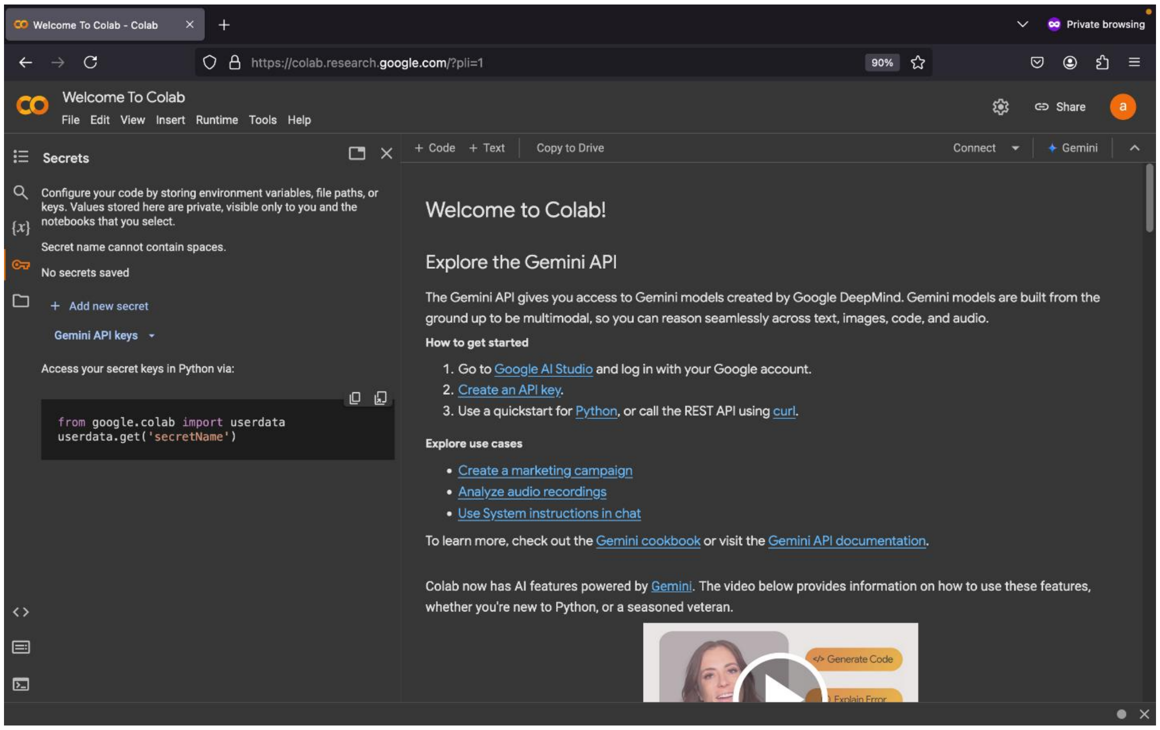
- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

Colab now has AI features powered by Gemini. The video below provides information on how to use these features, whether you're new to Python, or a seasoned veteran.

Alt text: An image showing the Welcome to Colab page with the **Secrets** tab

23. Select **Add new secret**.



Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

No secrets saved

Access your secret keys in Python via:

```
from google.colab import userdata
userdata.get('secretName')
```

Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

How to get started

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

Explore use cases

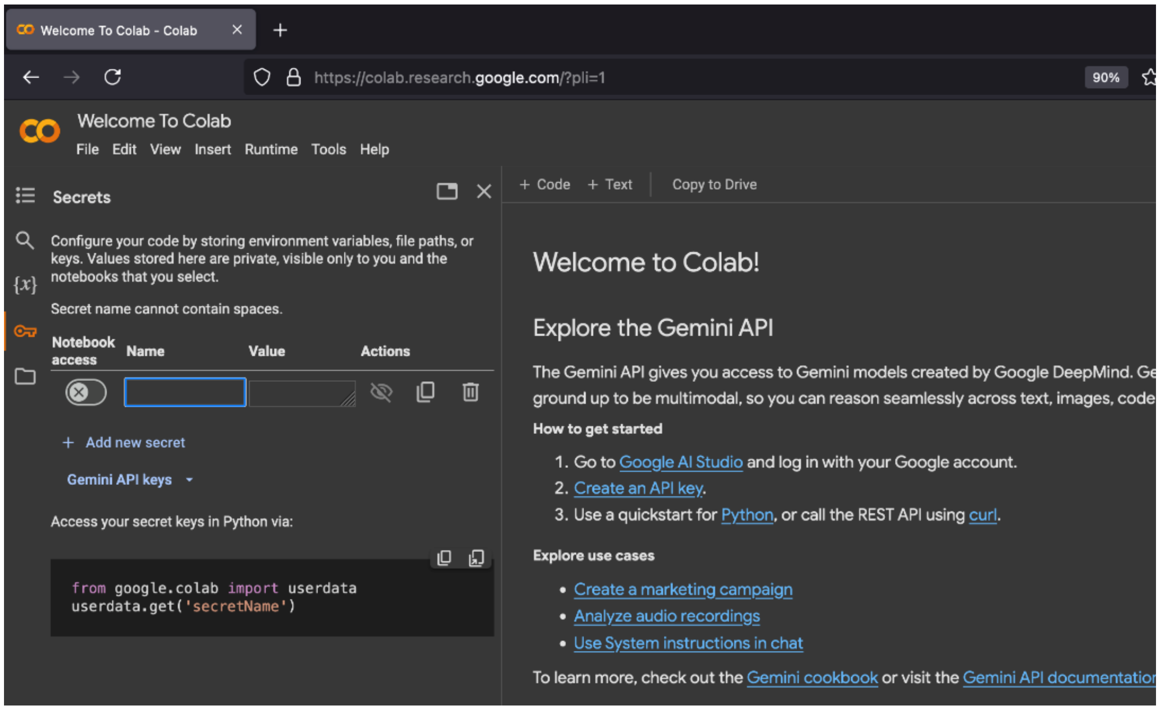
- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To learn more, check out the Gemini cookbook or visit the Gemini API documentation.

Colab now has AI features powered by Gemini. The video below provides information on how to use these features, whether you're new to Python, or a seasoned veteran.

Alt text: An image showing the Welcome to Colab page with the **Secrets** tab and the **Add new secret** option

24. Type **REPLICATE_api_token** in the **Name** field.



Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

Access your secret keys in Python via:

```
from google.colab import userdata
userdata.get('secretName')
```

Welcome to Colab!

Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Ge ground up to be multimodal, so you can reason seamlessly across text, images, code

How to get started

1. Go to Google AI Studio and log in with your Google account.
2. Create an API key.
3. Use a quickstart for Python, or call the REST API using curl.

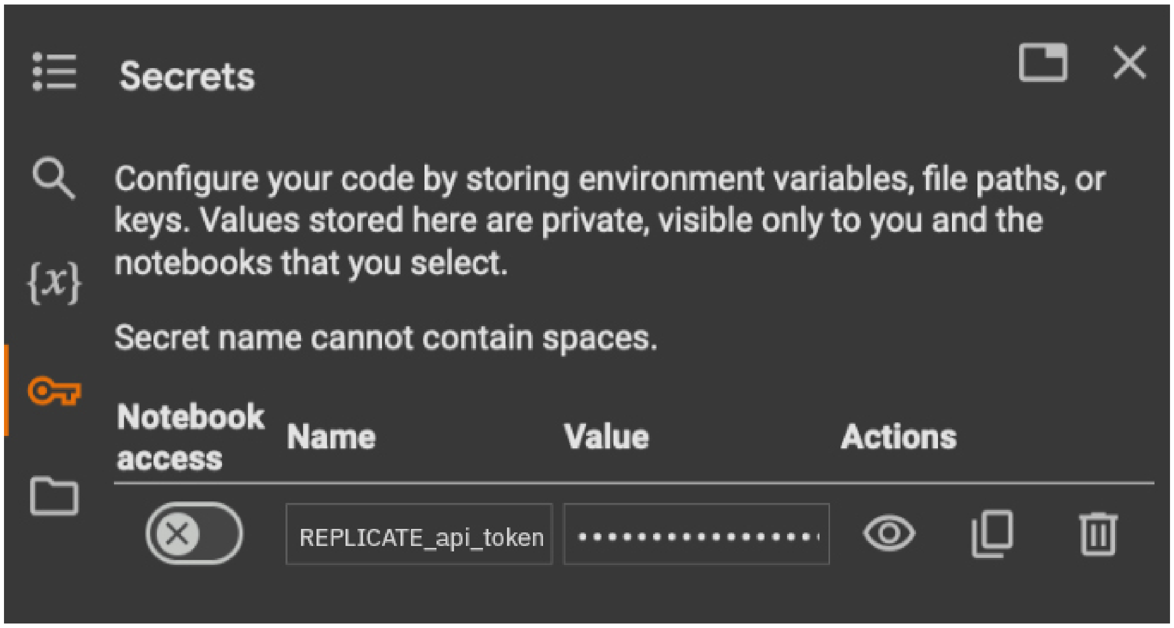
Explore use cases

- Create a marketing campaign
- Analyze audio recordings
- Use System instructions in chat

To learn more, check out the Gemini cookbook or visit the Gemini API documentatio

Alt text: An image showing the **Secrets** tab of Google Colab with the **Name** and **Value** fields

25. Paste your Replicate API token into the **Value** field.



Secrets

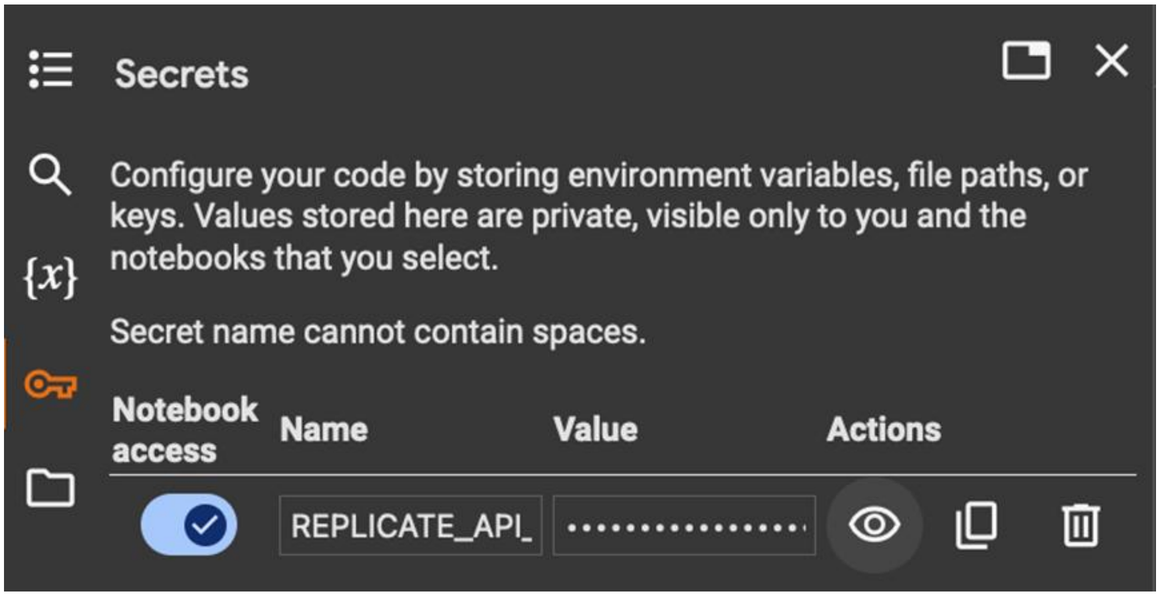
Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

Notebook access	Name	Value	Actions
	REPLICATE_api_token	•••••••••••••••••	

Alt text: An image showing the **Secrets** tab displaying the **Name** and **Value** fields

26. Select the toggle button to enable Notebook access. Next, select the close icon to exit the configuration.



Secrets

Configure your code by storing environment variables, file paths, or keys. Values stored here are private, visible only to you and the notebooks that you select.

Secret name cannot contain spaces.

Notebook access	Name	Value	Actions
✓	REPLICATE_API_	•••••••••••••••••	

Alt text: An image showing the **Secrets** section displaying the **Name** and **Value** fields and the enabled toggle button

Step 4: Initialize the AI agent and tools

Overview

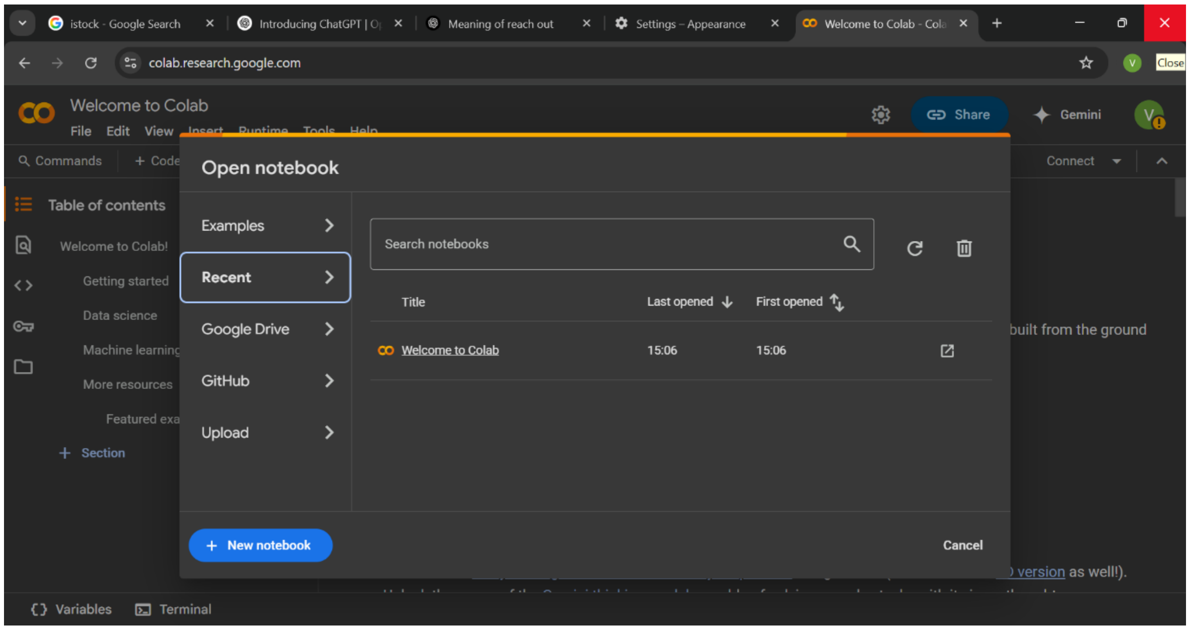
In this step, you'll initialize TechMart's AI agent and the tools it uses by loading a Jupyter notebook.

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.

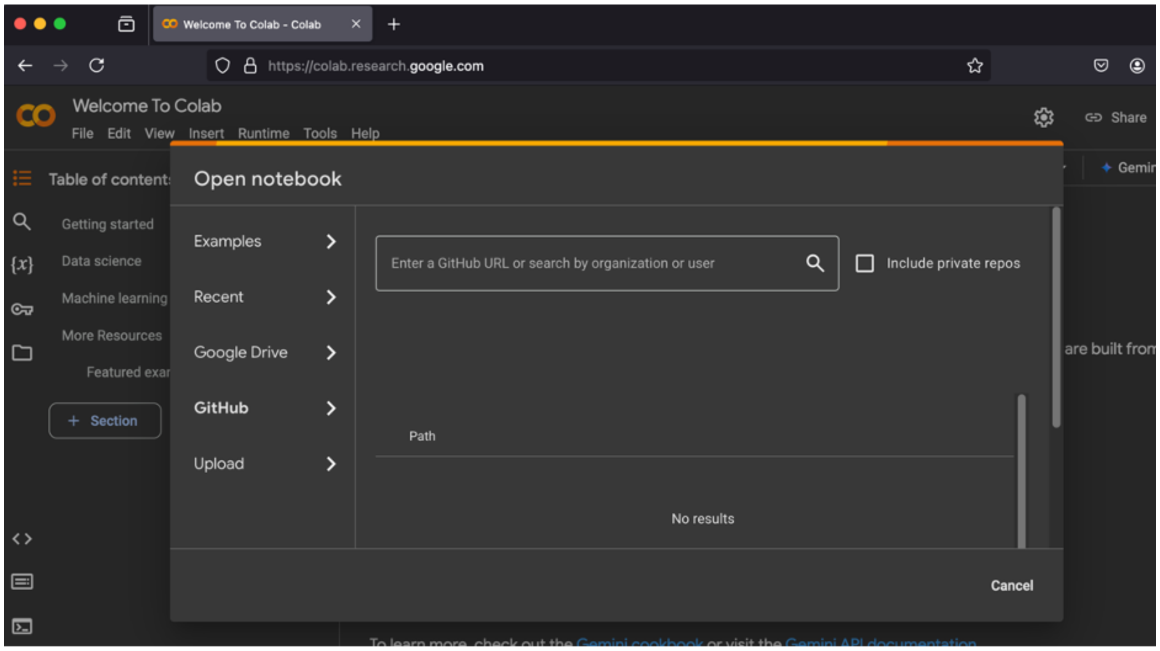
Instructions

27. Select [Google Colab]() to open the Google Colab welcome screen with the "Open notebook" dialog box where you can open your Jupyter notebook.



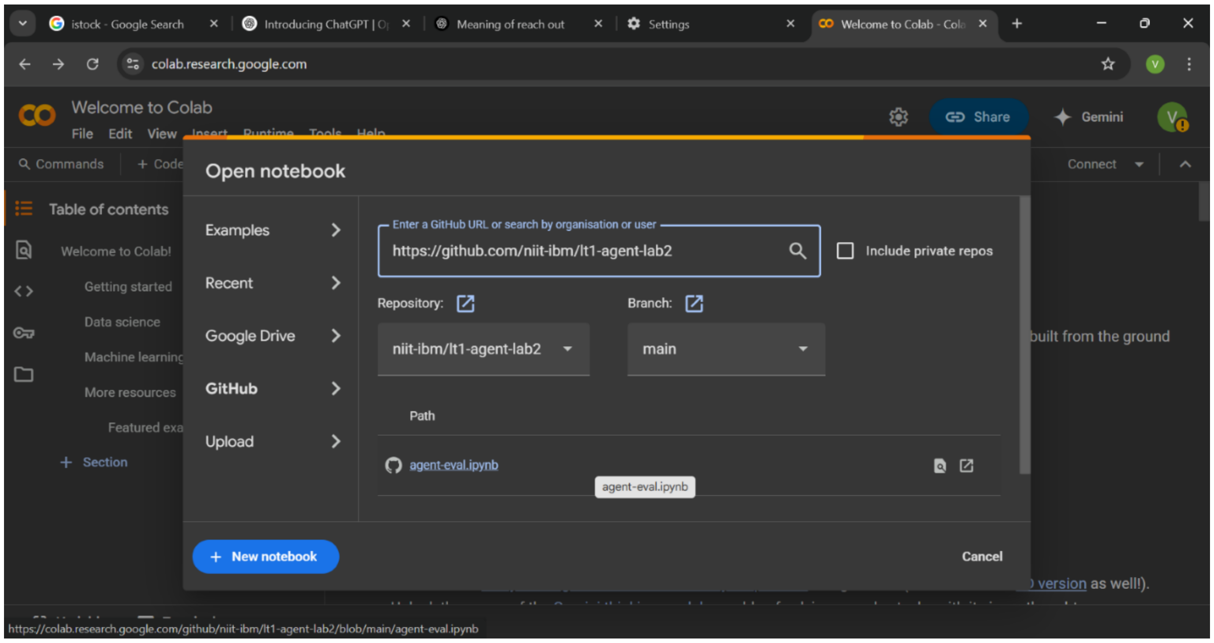
Alt text: An image showing the Google Colab welcome screen with the "Open notebook" dialog box

28. In the “Open notebook” dialog box, select the **GitHub** tab. 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.



Alt text: An image showing the “Open notebook” dialog box with GitHub as the selected tab

29. After you enter the URL, the **Repository** and **Branch** fields will be automatically populated. Now, to open the notebook, select **agent-eval.ipynb**.



https://github.com/niit-ibm/lt1-agent-lab2

Repository: niit-ibm/lt1-agent-lab2

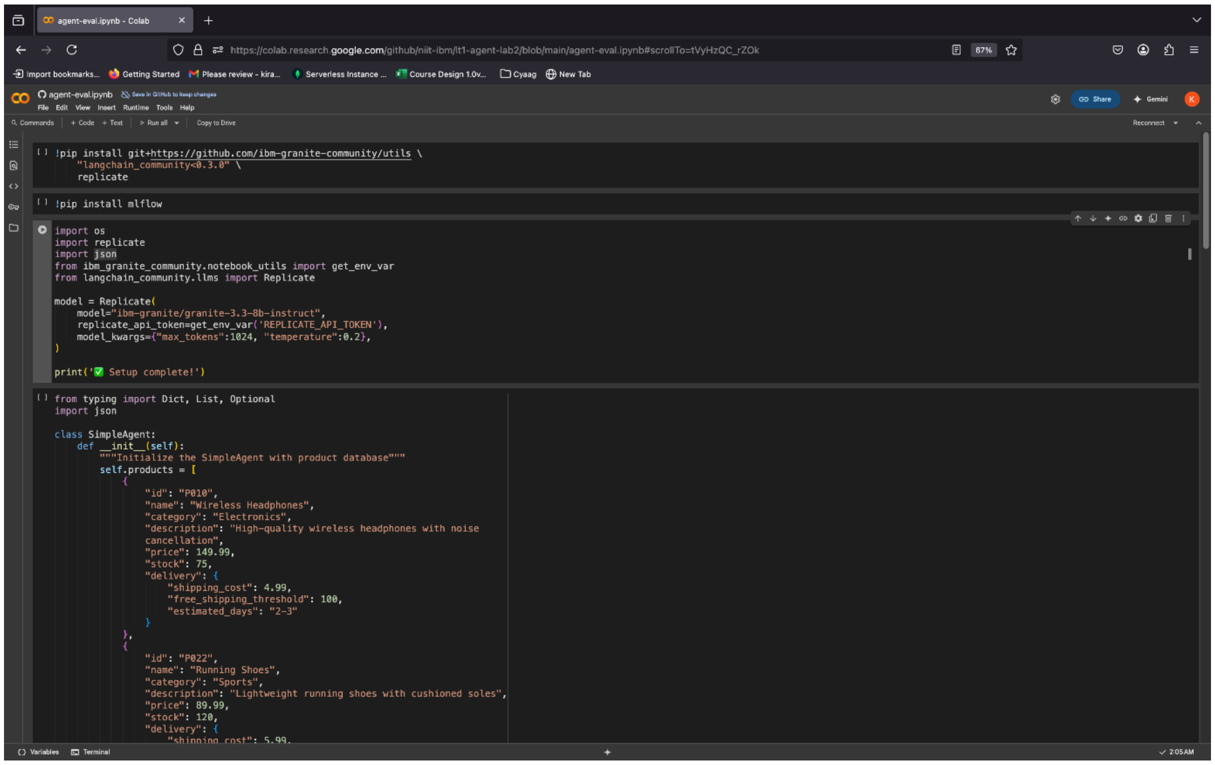
Branch: main

agent-eval.ipynb

https://colab.research.google.com/github/niit-ibm/lt1-agent-lab2/blob/main/agent-eval.ipynb

Alt text: An image showing the “Open notebook” dialog box with the GitHub URL in the search field

30. The “agent-eval.ipynb” notebook opens in the Colab workspace.



```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
!pip install mlflow
```

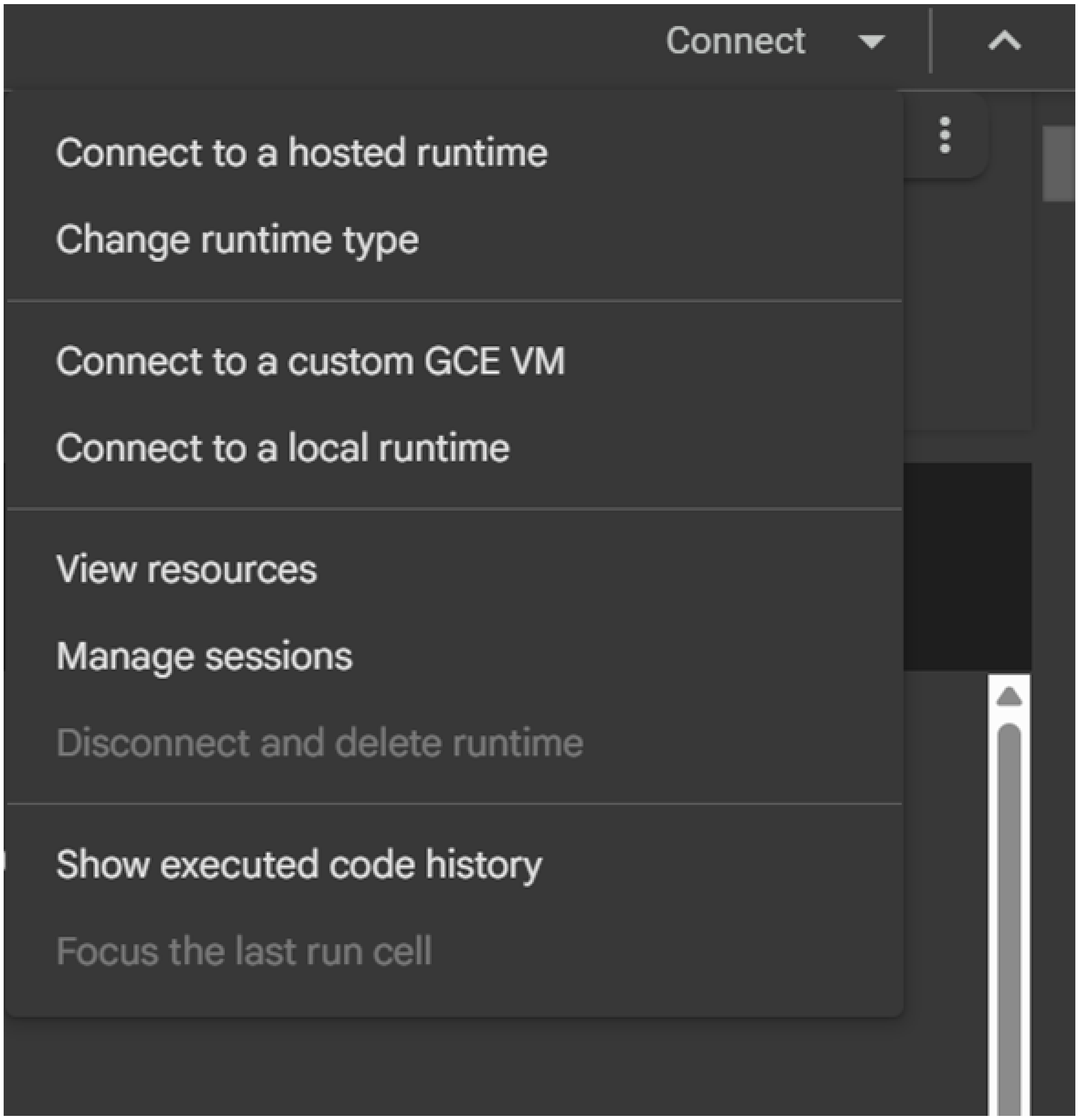
```
import os
import replicate
import json
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)

print('✅ Setup complete!')
```

```
from typing import Dict, List, Optional
import json

class SimpleAgent:
    def __init__(self):
        """Initialize the SimpleAgent with product database"""
        self.products = [
            {
                "id": "P010",
                "name": "Wireless Headphones",
                "category": "Electronics",
                "description": "High-quality wireless headphones with noise
                cancellation",
                "price": 149.99,
                "stock": 75,
                "delivery": {
                    "shipping_cost": 4.99,
                    "free_shipping_threshold": 100,
                    "estimated_days": "2-3"
                }
            },
            {
                "id": "P022",
                "name": "Running Shoes",
                "category": "Sports",
                "description": "Lightweight running shoes with cushioned soles",
                "price": 89.99,
                "stock": 120,
                "delivery": {
                    "shipping_cost": 5.99,
```

Alt text: An image showing the Google Colab workspace displaying the cells in the “agent-eval.ipynb” Jupyter notebook

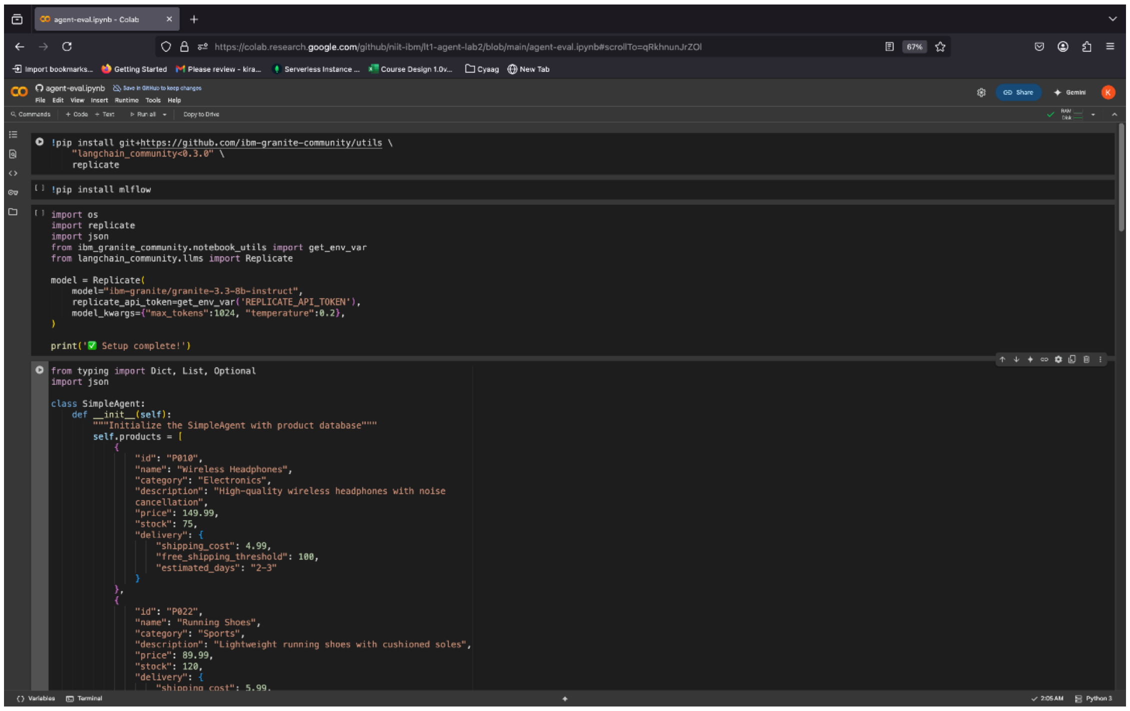
31. To run the code, select the **Connect to a hosted runtime** option from the **Connect** drop-down list of the notebook.

    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.

    
    Connect
Connect to a hosted runtime
Change runtime type
Connect to a custom GCE VM
Connect to a local runtime
View resources
Manage sessions
Disconnect and delete runtime
Show executed code history
Focus the last run cell

    Alt text: An image showing a Jupyter notebook with the **Connect to a hosted runtime** option in the **Connect** drop-down list

32. Check the **Colab** toolbar for a green checkmark. This indicates that the connection is successful, and your code is now ready to run.



```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
!pip install mlflow
```

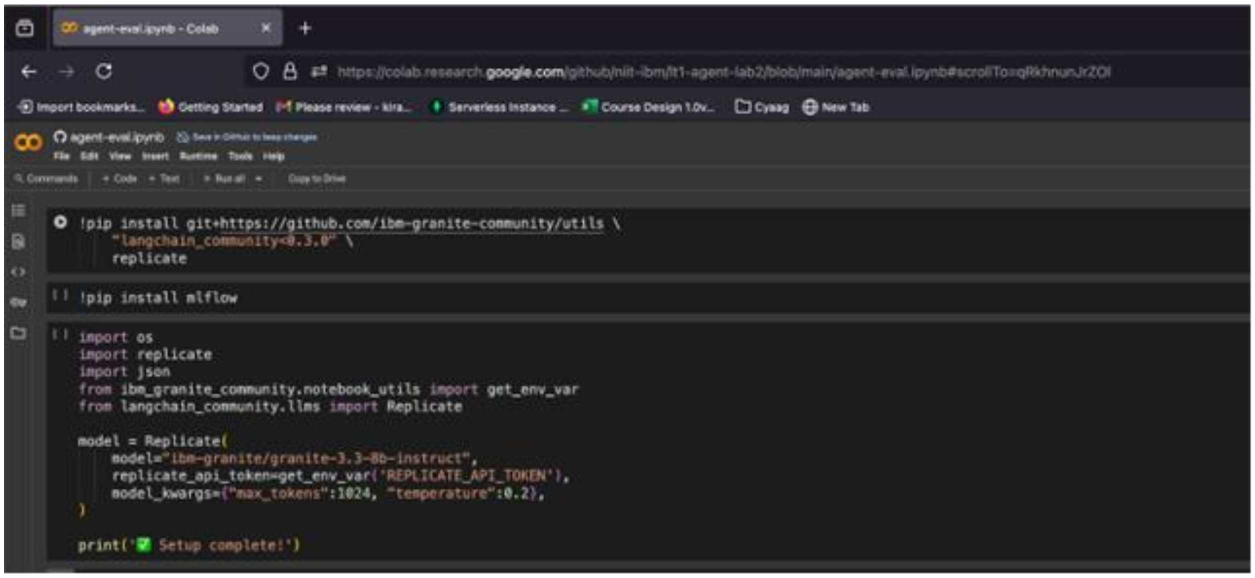
```
import os
import replicate
import json
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)

print('✅ Setup complete!')
```

```
from typing import Dict, List, Optional
import json

class SimpleAgent:
    def __init__(self):
        """Initialize the SimpleAgent with product database"""
        self.products = [
            {
                "id": "P010",
                "name": "Wireless Headphones",
                "category": "Electronics",
                "description": "High-quality wireless headphones with noise
                cancellation",
                "price": 149.99,
                "stock": 75,
                "delivery": {
                    "shipping_cost": 4.99,
                    "free_shipping_threshold": 100,
                    "estimated_days": "2-3"
                }
            },
            {
                "id": "P022",
                "name": "Running Shoes",
                "category": "Sports",
                "description": "Lightweight running shoes with cushioned soles",
                "price": 89.99,
                "stock": 120,
                "delivery": {
                    "shipping_cost": 5.99,
```

Alt text: An image showing the Google Colab workspace displaying a green checkmark indicating successful connection to the Google Colab environment

33. 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 can use to execute the code within that cell.


agent-eval.ipynb - Colab
https://colab.research.google.com/github/niit-ibm/lt1-agent-lab2/blob/main/agent-eval.ipynb#scrollTo=qRkhnunJrZOI
agent-eval.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all Copy to Drive

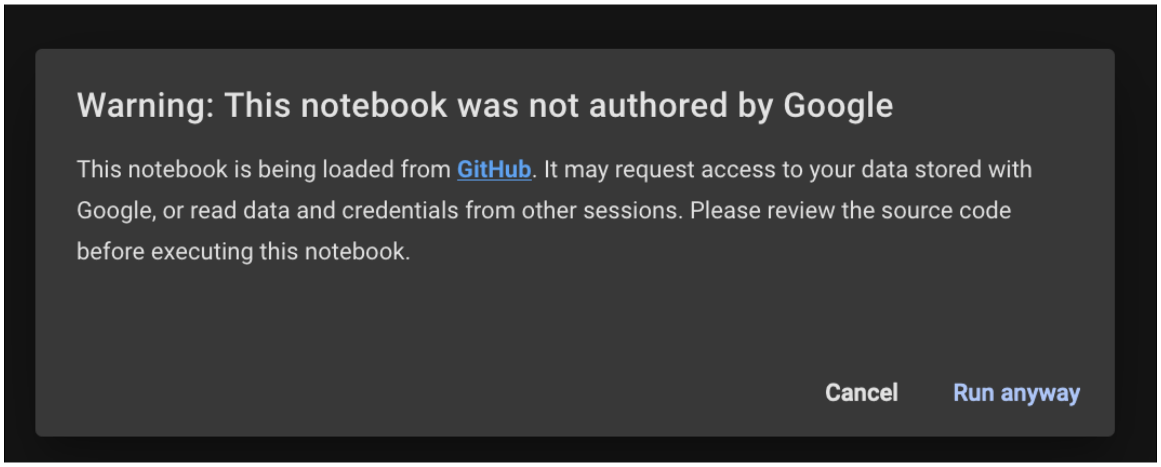
```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
!pip install mlflow
```

```
import os
import replicate
import json
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
    replicate_api_token=get_env_var('REPLICATE_API_TOKEN'),
    model_kwargs={"max_tokens":1024, "temperature":0.2},
)

print('✅ Setup complete!')
```

Alt text: An image showing a cell in the "agent-eval-ipynb" Jupyter notebook, with the run icon and a prompt to install the required libraries

34. Select the **Run anyway** option to continue loading the required libraries.

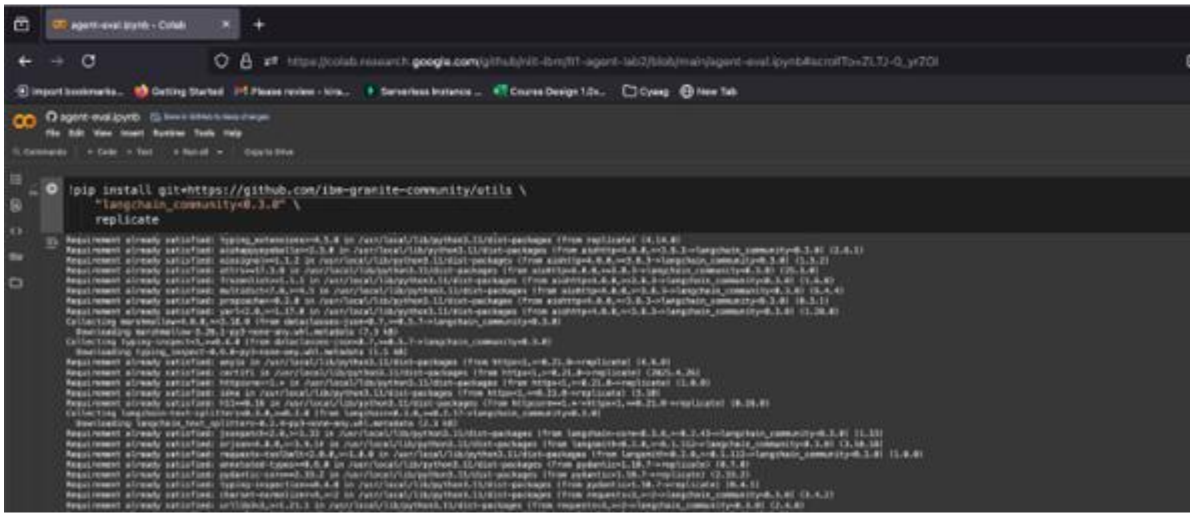


Warning: This notebook was not authored by Google

This notebook is being loaded from GitHub. It may request access to your data stored with Google, or read data and credentials from other sessions. Please review the source code before executing this notebook.

Alt text: An image showing the "Warning: This notebook was not authored by Google" message with the **Run anyway** option

35. A green checkmark will appear next to the run icon, indicating that the required libraries have been successfully installed.

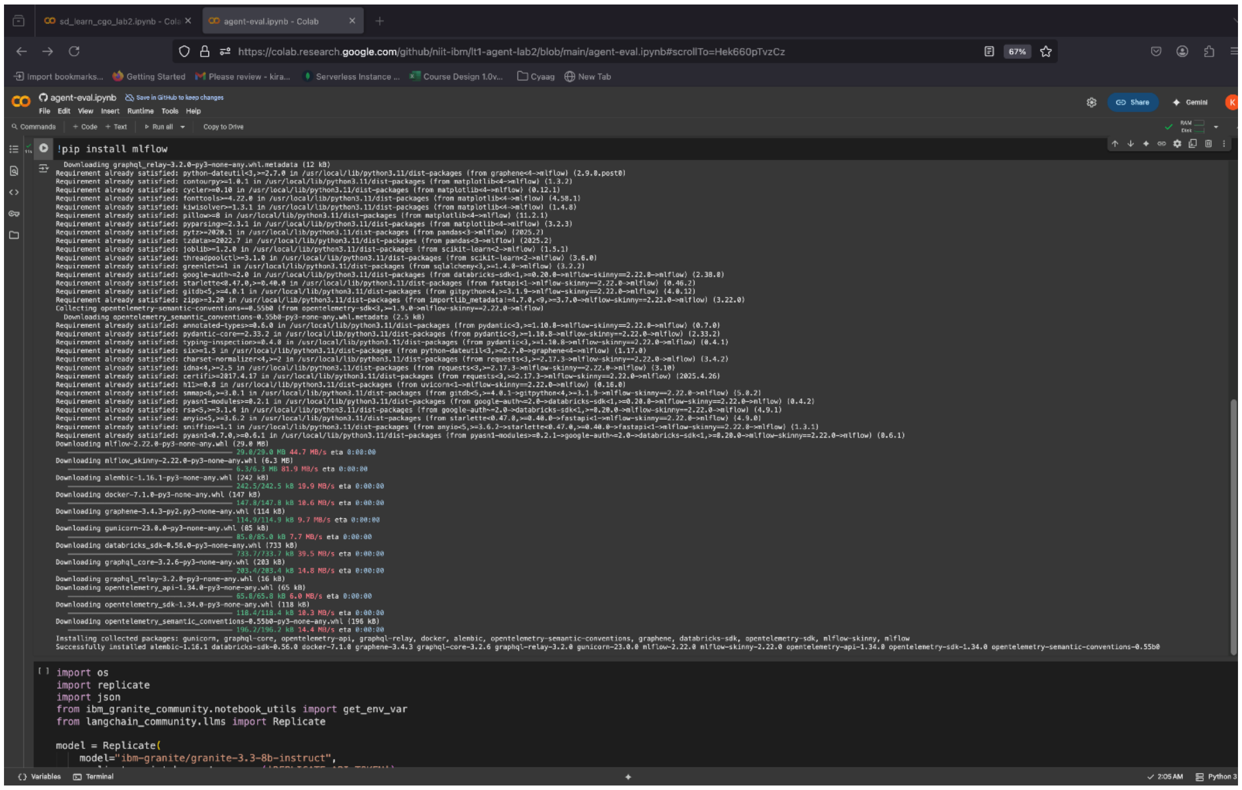


```
!pip install git+https://github.com/ibm-granite-community/utils \
    "langchain_community<0.3.0" \
    replicate
```

```
[illegible]
```

Alt text: An image showing the Google Colab workspace, with a checkmark in the first cell of the Jupyter notebook indicating successful installation of the required libraries

36. Next, install MLFlow by selecting the run icon in the second cell.



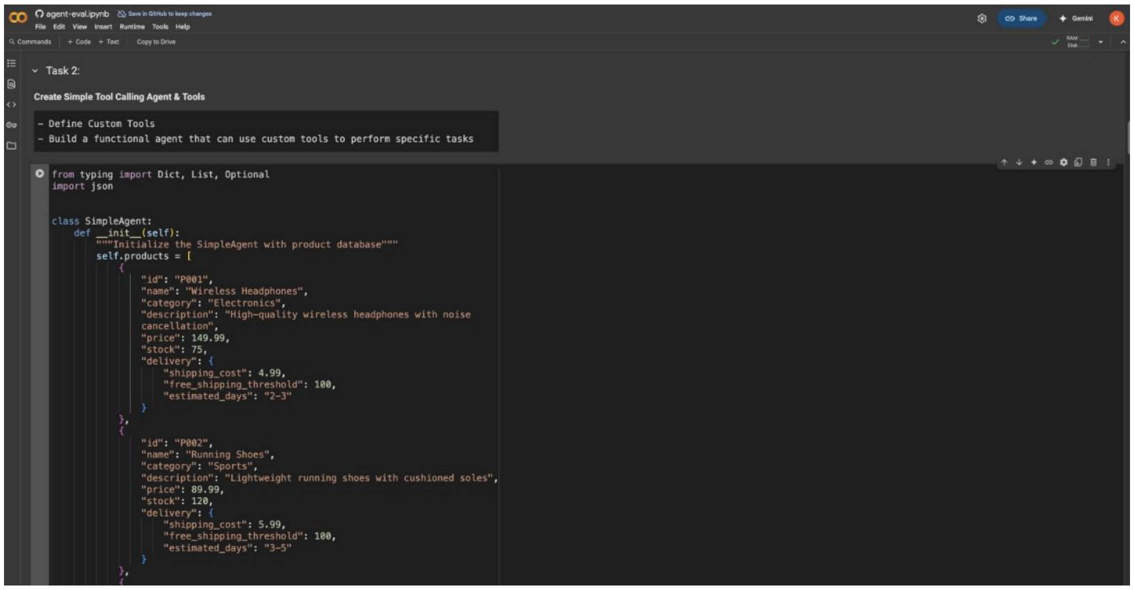
```
!pip install mlflow
```

```
  Downloading graphql_relay-3.2.0-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: python-dateutil<3,>=2.7.0 in /usr/local/lib/python3.11/dist-packages (from graphene<4->mlflow) (2.9.0.post0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (4.58.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (1.4.8)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib<4->mlflow) (3.2.3)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3->mlflow) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3->mlflow) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn<2->mlflow) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn<2->mlflow) (3.6.0)
Requirement already satisfied: greenlet>=1 in /usr/local/lib/python3.11/dist-packages (from sqlalchemy<3,>=1.4.0->mlflow) (3.2.2)
Requirement already satisfied: google-auth~=2.0 in /usr/local/lib/python3.11/dist-packages (from databricks-sdk<1,>=0.20.0->mlflow-skinny==2.22.0->mlflow) (2.38.0)
Requirement already satisfied: starlette<0.47.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from fastapi<1->mlflow-skinny==2.22.0->mlflow) (0.46.2)
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/dist-packages (from gitpython<4,>=3.1.9->mlflow-skinny==2.22.0->mlflow) (4.0.12)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.11/dist-packages (from importlib_metadata!=4.7.0,<9,>=3.7.0->mlflow-skinny==2.22.0->mlflow) (3.22.0)
Collecting opentelemetry-semantic-conventions==0.55b0 (from opentelemetry-sdk<3,>=1.9.0->mlflow-skinny==2.22.0->mlflow)
  Downloading opentelemetry_semantic_conventions-0.55b0-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.10.8->mlflow-skinny==2.22.0->mlflow) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.10.8->mlflow-skinny==2.22.0->mlflow) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<3,>=1.10.8->mlflow-skinny==2.22.0->mlflow) (0.4.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil<3,>=2.7.0->graphene<4->mlflow) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.17.3->mlflow-skinny==2.22.0->mlflow) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.17.3->mlflow-skinny==2.22.0->mlflow) (3.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.17.3->mlflow-skinny==2.22.0->mlflow) (2025.4.26)
Requirement already satisfied: h11>=0.8 in /usr/local/lib/python3.11/dist-packages (from uvicorn<1->mlflow-skinny==2.22.0->mlflow) (0.16.0)
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/dist-packages (from gitdb<5,>=4.0.1->gitpython<4,>=3.1.9->mlflow-skinny==2.22.0->mlflow) (5.0.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from google-auth~=2.0->databricks-sdk<1,>=0.20.0->mlflow-skinny==2.22.0->mlflow) (0.4.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-packages (from google-auth~=2.0->databricks-sdk<1,>=0.20.0->mlflow-skinny==2.22.0->mlflow) (4.9.1)
Requirement already satisfied: anyio<5,>=3.6.2 in /usr/local/lib/python3.11/dist-packages (from starlette<0.47.0,>=0.40.0->fastapi<1->mlflow-skinny==2.22.0->mlflow) (4.9.0)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5,>=3.6.2->starlette<0.47.0,>=0.40.0->fastapi<1->mlflow-skinny==2.22.0->mlflow) (1.3.1)
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.11/dist-packages (from pyasn1-modules>=0.2.1->google-auth~=2.0->databricks-sdk<1,>=0.20.0->mlflow-skinny==2.22.0->mlflow) (0.6.1)
Downloading mlflow-2.22.0-py3-none-any.whl (29.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 29.0/29.0 MB 44.7 MB/s eta 0:00:00
Downloading mlflow_skinny-2.22.0-py3-none-any.whl (6.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.3/6.3 MB 81.9 MB/s eta 0:00:00
Downloading alembic-1.16.1-py3-none-any.whl (242 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 242.5/242.5 kB 19.9 MB/s eta 0:00:00
Downloading docker-7.1.0-py3-none-any.whl (147 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 147.8/147.8 kB 10.6 MB/s eta 0:00:00
Downloading graphene-3.4.3-py2.py3-none-any.whl (114 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 114.9/114.9 kB 9.7 MB/s eta 0:00:00
Downloading gunicorn-23.0.0-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.0/85.0 kB 7.7 MB/s eta 0:00:00
Downloading databricks_sdk-0.56.0-py3-none-any.whl (733 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 733.7/733.7 kB 39.5 MB/s eta 0:00:00
Downloading graphql_core-3.2.6-py3-none-any.whl (203 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 203.4/203.4 kB 14.8 MB/s eta 0:00:00
Downloading graphql_relay-3.2.0-py3-none-any.whl (16 kB)
Downloading opentelemetry_api-1.34.0-py3-none-any.whl (65 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 6.0 MB/s eta 0:00:00
Downloading opentelemetry_sdk-1.34.0-py3-none-any.whl (118 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.4/118.4 kB 10.3 MB/s eta 0:00:00
Downloading opentelemetry_semantic_conventions-0.55b0-py3-none-any.whl (196 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.2/196.2 kB 14.4 MB/s eta 0:00:00
Installing collected packages: gunicorn, graphql-core, opentelemetry-api, graphql-relay, docker, alembic, opentelemetry-semantic-conventions, graphene, databricks-sdk, opentelemetry-sdk, mlflow-skinny, mlflow
Successfully installed alembic-1.16.1 databricks-sdk-0.56.0 docker-7.1.0 graphene-3.4.3 graphql-core-3.2.6 graphql-relay-3.2.0 gunicorn-23.0.0 mlflow-2.22.0 mlflow-skinny-2.22.0 opentelemetry-api-1.34.0 opentelemetry-sdk-1.34.0 opentelemetry-semantic-conventions-0.55b0
```

```
import os
import replicate
import json
from ibm_granite_community.notebook_utils import get_env_var
from langchain_community.llms import Replicate

model = Replicate(
    model="ibm-granite/granite-3.3-8b-instruct",
```

Alt text: An image showing Python code with the run icon, used to install MLFlow

37. Next, to create a database for TechMart's product inventory, select the run icon in the first cell of Task 2.



agent-eval.ipynb
Save in GitHub to keep changes
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Copy to Drive
Share
Gemini

Task 2:

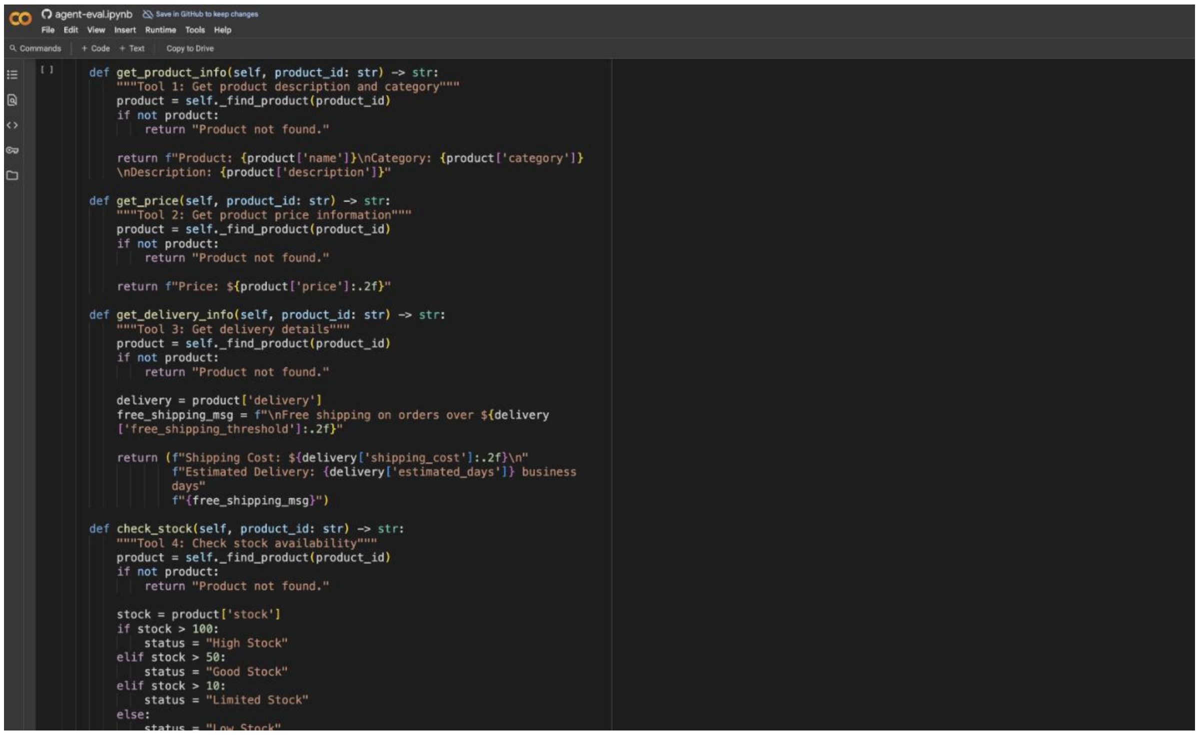
Create Simple Tool Calling Agent & Tools

```
- Define Custom Tools
- Build a functional agent that can use custom tools to perform specific tasks
```

```
from typing import Dict, List, Optional
import json


class SimpleAgent:
    def __init__(self):
        """Initialize the SimpleAgent with product database"""
        self.products = [
            {
                "id": "P001",
                "name": "Wireless Headphones",
                "category": "Electronics",
                "description": "High-quality wireless headphones with noise
                cancellation",
                "price": 149.99,
                "stock": 75,
                "delivery": {
                    "shipping_cost": 4.99,
                    "free_shipping_threshold": 100,
                    "estimated_days": "2-3"
                }
            },
            {
                "id": "P002",
                "name": "Running Shoes",
                "category": "Sports",
                "description": "Lightweight running shoes with cushioned soles",
                "price": 89.99,
                "stock": 120,
                "delivery": {
                    "shipping_cost": 5.99,
                    "free_shipping_threshold": 100,
                    "estimated_days": "3-5"
                }
            },
```

Alt text: An image showing Python code in the Google Colab workspace that defines a data set for product inventory, including product id, name, category, description, price, stock, and delivery information

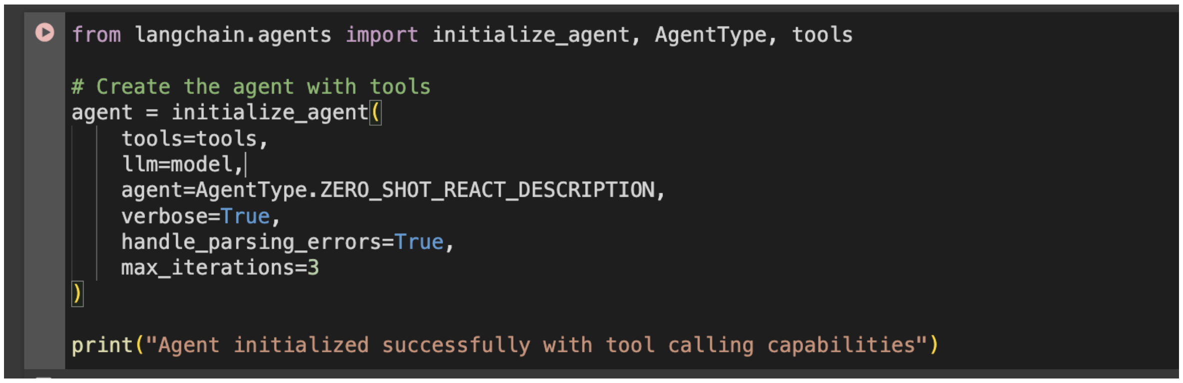
38. To enable the agent to process user queries and generate meaningful responses, define the required tools such as **get_product_info**, **get_price, get_delivery_info**, and **check_stock**.


agent-eval.ipynb
Save in GitHub to keep changes
File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Copy to Drive

```python
    def get_product_info(self, product_id: str) -> str:
        """Tool 1: Get product description and category"""
        product = self._find_product(product_id)
        if not product:
            return "Product not found."

        return f"Product: {product['name']}\nCategory: {product['category']}
        \nDescription: {product['description']}"

    def get_price(self, product_id: str) -> str:
        """Tool 2: Get product price information"""
        product = self._find_product(product_id)
        if not product:
            return "Product not found."

        return f"Price: ${product['price']:.2f}"

    def get_delivery_info(self, product_id: str) -> str:
        """Tool 3: Get delivery details"""
        product = self._find_product(product_id)
        if not product:
            return "Product not found."

        delivery = product['delivery']
        free_shipping_msg = f"\nFree shipping on orders over ${delivery
        ['free_shipping_threshold']:.2f}"

        return (f"Shipping Cost: ${delivery['shipping_cost']:.2f}\n"
                f"Estimated Delivery: {delivery['estimated_days']} business
                days"
                f"{free_shipping_msg}")

    def check_stock(self, product_id: str) -> str:
        """Tool 4: Check stock availability"""
        product = self._find_product(product_id)
        if not product:
            return "Product not found."

        stock = product['stock']
        if stock > 100:
            status = "High Stock"
        elif stock > 50:
            status = "Good Stock"
        elif stock > 10:
            status = "Limited Stock"
        else:
            status = "Low Stock"
```

Alt text: An image showing Python code displaying the definition of several functions, including **get_product_info**, **get_price**, **get_delivery_info**, and **check_stock**

39. To initialize the agent, select the run icon. 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.



```python
from langchain.agents import initialize_agent, AgentType, tools

# Create the agent with tools
agent = initialize_agent(
    tools=tools,
    llm=model,
    agent=AgentType.ZERO_SHOT_REACT_DESCRIPTION,
    verbose=True,
    handle_parsing_errors=True,
    max_iterations=3
)

print("Agent initialized successfully with tool calling capabilities")
```

Alt text: An image showing Python code displaying the initialization of an AI agent using the **langchain.agents** library

Step 5: Evaluate the trajectory and final response of the AI agent

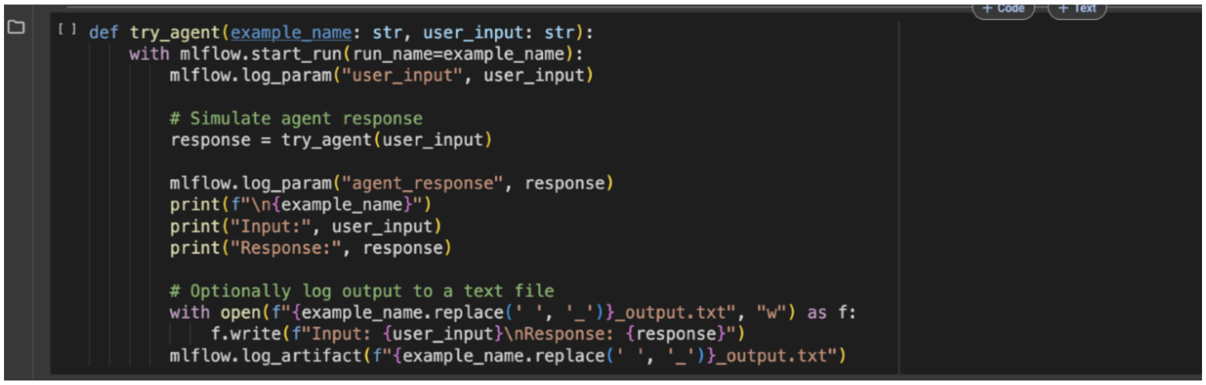
Overview

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

Instructions

40. To test how the agent responds, select the run icon to create an evaluation set. This set includes different types of user requests, expected factual answers, and operational rules that guide how the agent should behave.

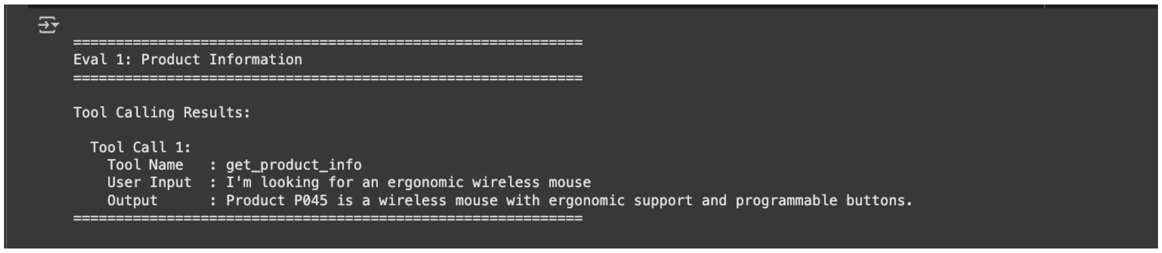
    Note: This evaluation set will be used by MLFlow to test the agent's performance. Each entry specifies the user's request, the expected output, and any specific guidelines the agent's response should follow.



```python
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")
```

Alt text: An image showing the code for analyzing the trajectory of the "agent-eval.ipynb" AI agent

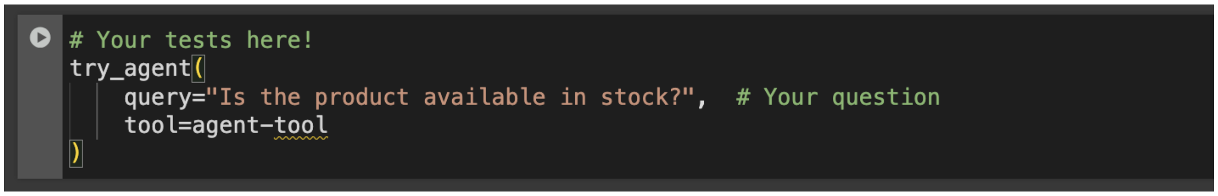
41. Analyze the tool-calling results to determine whether the agent correctly interprets the user's query and uses the appropriate tools to generate a response.



```
============================================================
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.
============================================================
```

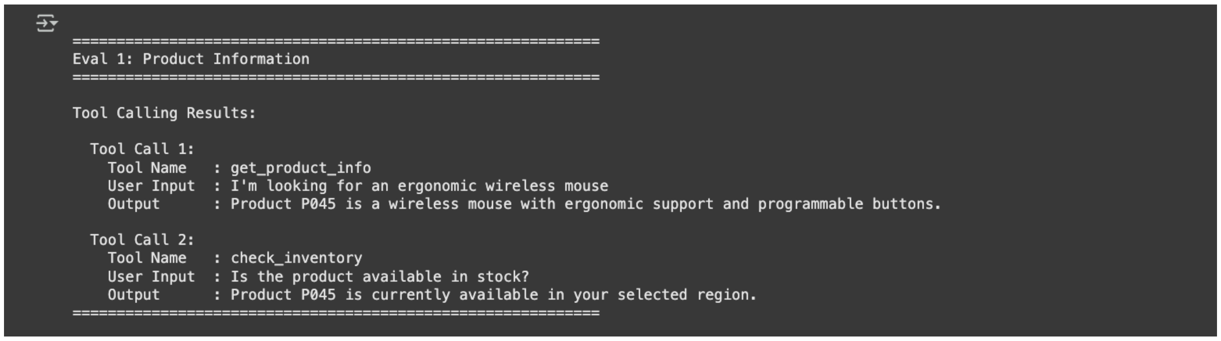
Alt text: An image showing the tool-calling results from the evaluation of product information

42. To evaluate whether the agent is able to handle follow-up questions by using multiple tools while retaining the user's original query and context, select the run icon. The code snippet shows the user asking, "Is the product available in stock?" prompting the agent to call multiple backend tools, such as **get_product_info** and **check_inventory**, and return a response based on this query.



```
# Your tests here!
try_agent(
    query="Is the product available in stock?",  # Your question
    tool=agent-tool
)
```

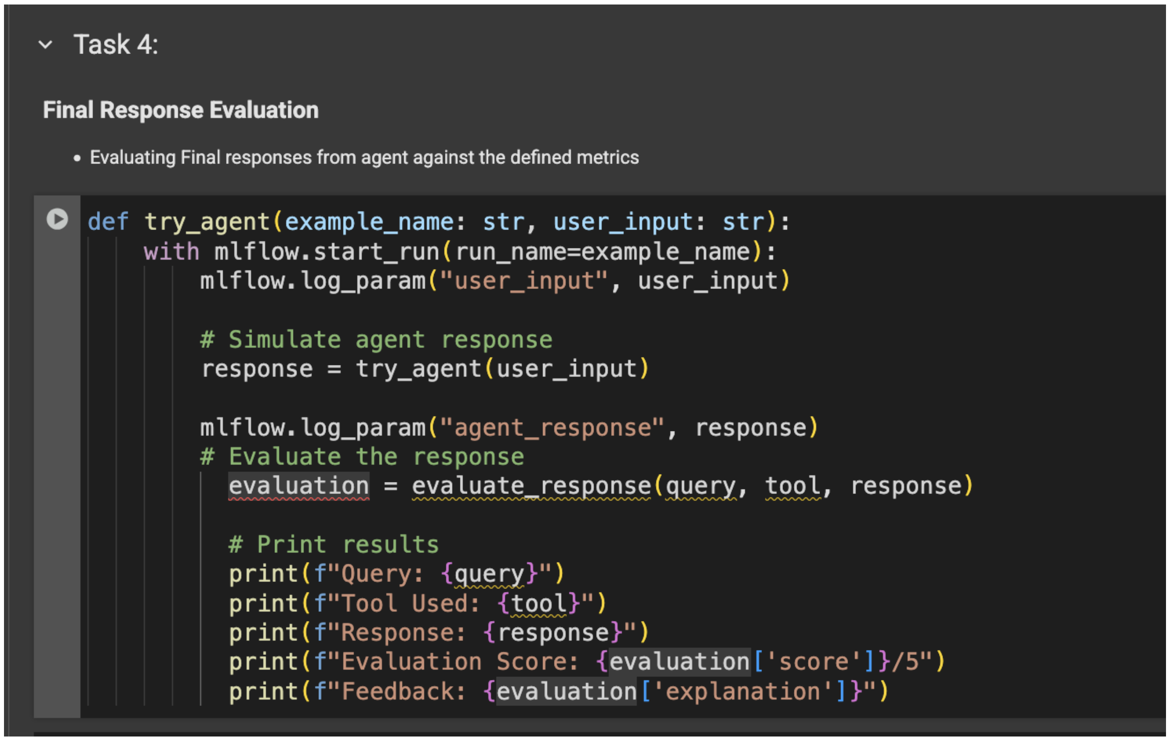
Alt text: An image showing a code snippet with the user's query, "Is the product available in stock?"



```
============================================================
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.
============================================================
```

Alt text: An image showing the results from the evaluation of product information based on two tool calls

43. For the final agent evaluation, select the run icon.



Task 4:

Final Response Evaluation

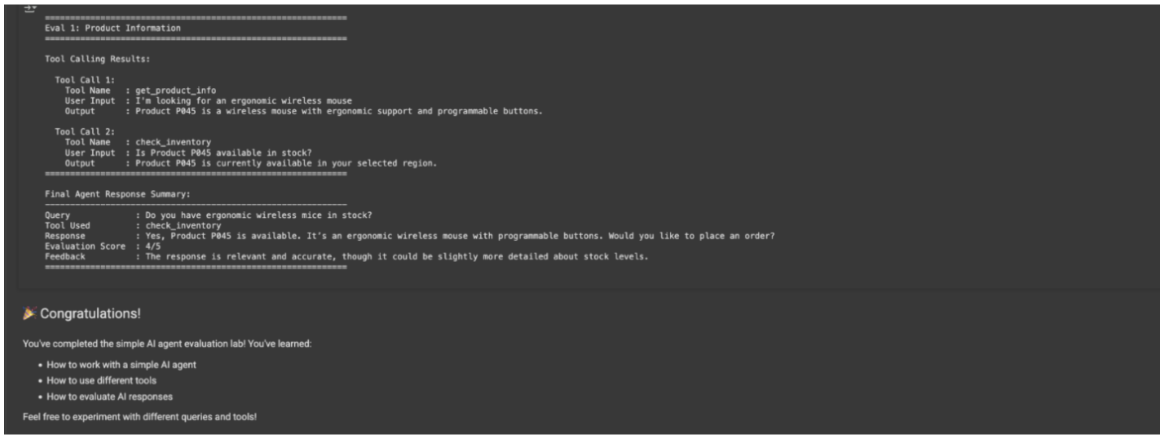
- Evaluating Final responses from agent against the defined metrics

```python
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']}")
```

Alt text: An image showing Python code that evaluates the final responses from the agent against the defined metrics

44. After the execution is finished, you'll see the response summary. Review the output and evaluate the results for accuracy and relevance.

    Note: This step shows that even when the agent successfully uses tools to gather information (as in Tool Calls 1 and 2), its final response still requires evaluation and refinement. The evaluation score and feedback are important for improving the agent's performance over time, even when its answers are already relevant and accurate.



```
============================================================
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 Product P045 available in stock?
    Output      : Product P045 is currently available in your selected region.
============================================================

Final Agent Response Summary:
------------------------------------------------------------
Query             : Do you have ergonomic wireless mice in stock?
Tool Used         : check_inventory
Response          : Yes, Product P045 is available. It's an ergonomic wireless mouse with programmable buttons. Would you like to place an order?
Evaluation Score  : 4/5
Feedback          : The response is relevant and accurate, though it could be slightly more detailed about stock levels.
============================================================
```

🎉 Congratulations!

You've completed the simple AI agent evaluation lab! You've learned:

- How to work with a simple AI agent
- How to use different tools
- How to evaluate AI responses

Feel free to experiment with different queries and tools!

Alt text: An image showing the agent's final response output from different tool calls, along with the response summary

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 product information delivery.

IBM SkillsBuild