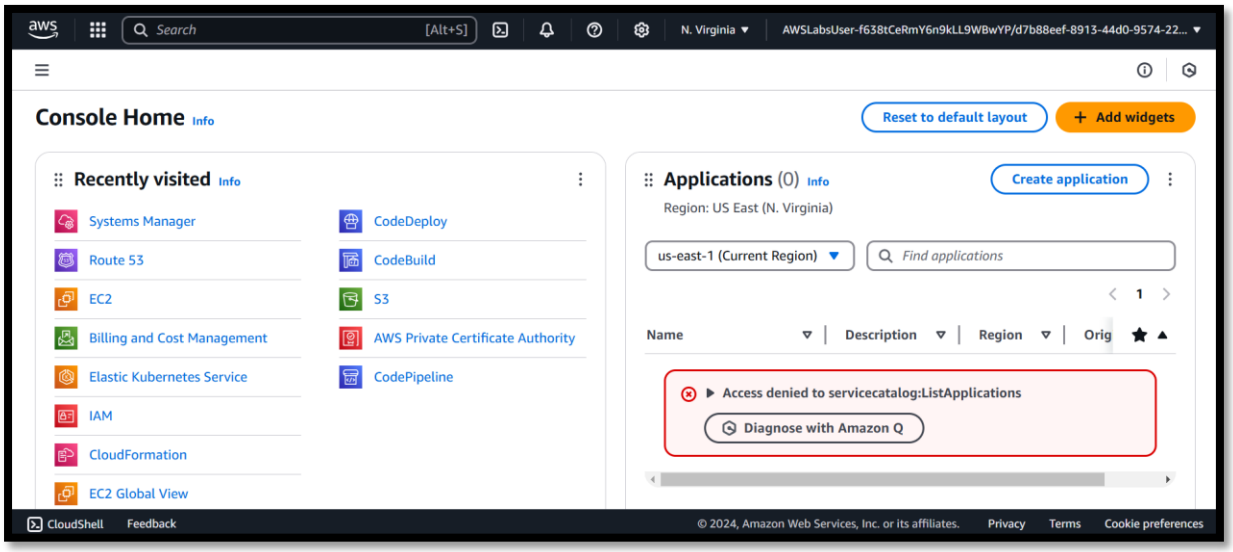
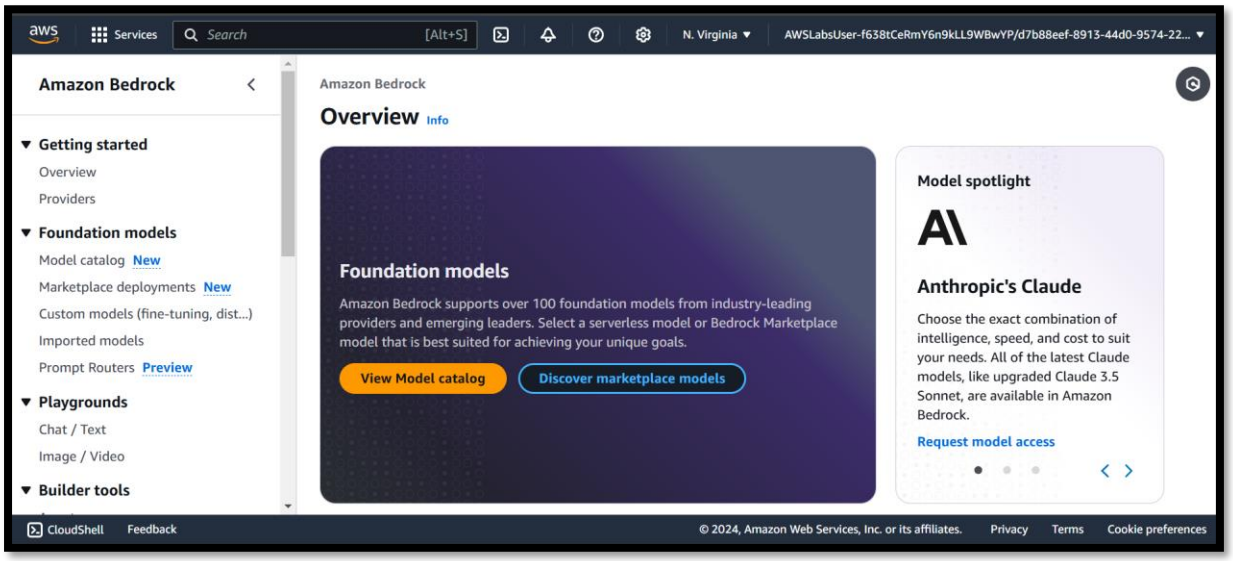
Objective: To integrate Amazon Bedrock models with LangChain agents by using the flexible Converse API to incorporate external capabilities into conversational applications.

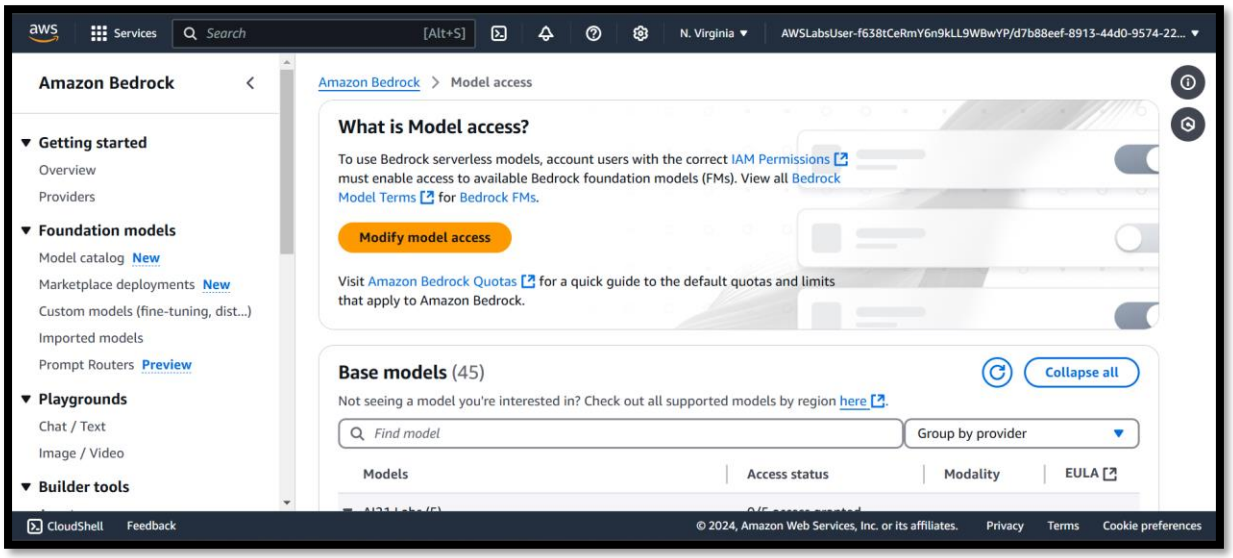
Task 0: Set up the environment

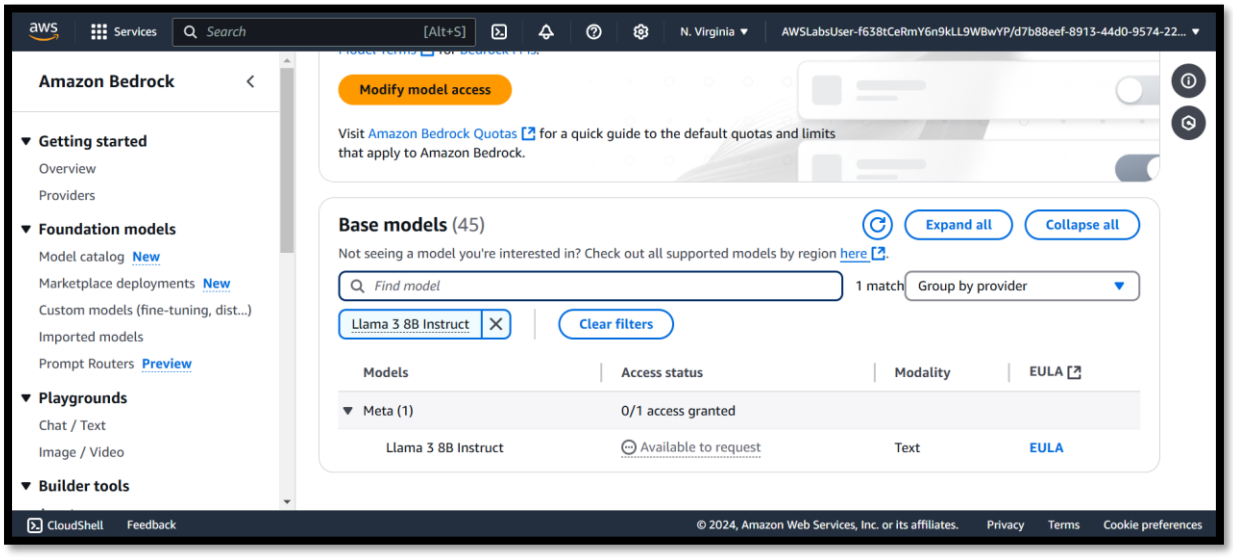
In this task, I registered the base models in the Amazon Bedrock console and launched an Amazon SageMaker Studio application to access my lab resources.



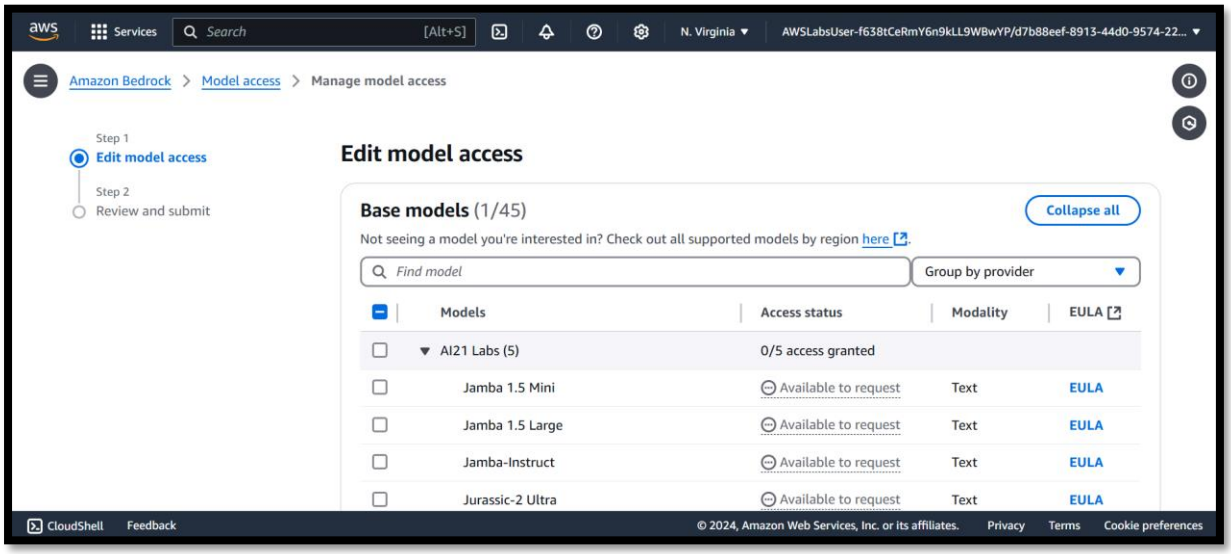


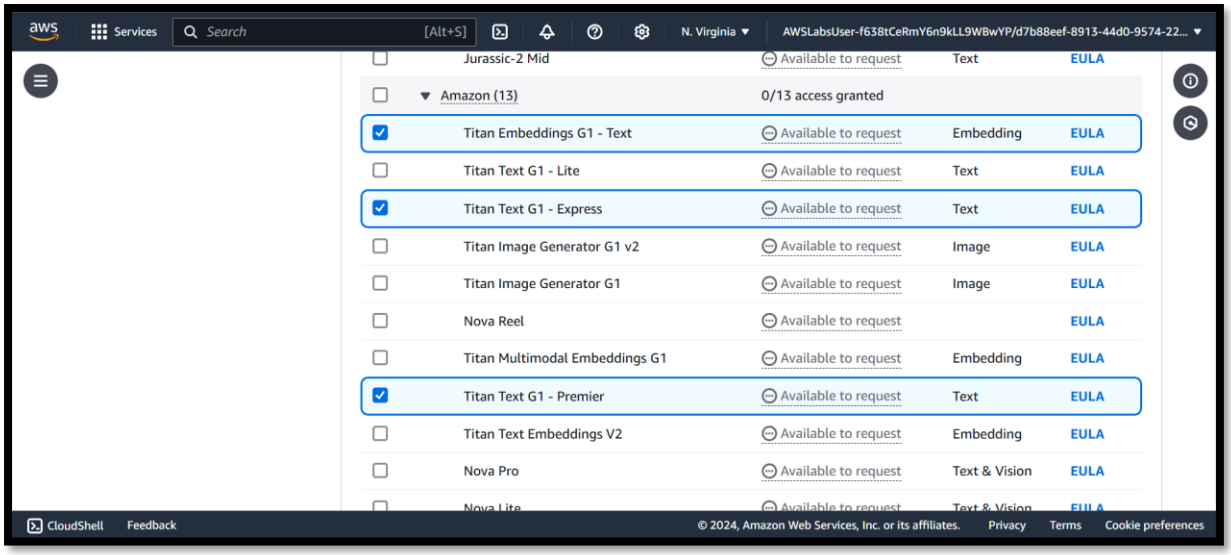
I reviewed the Access Status for each of the models. If the Access Status for one or more of the models was set to Available to request, I expanded this menu and followed the steps to enable access for them.

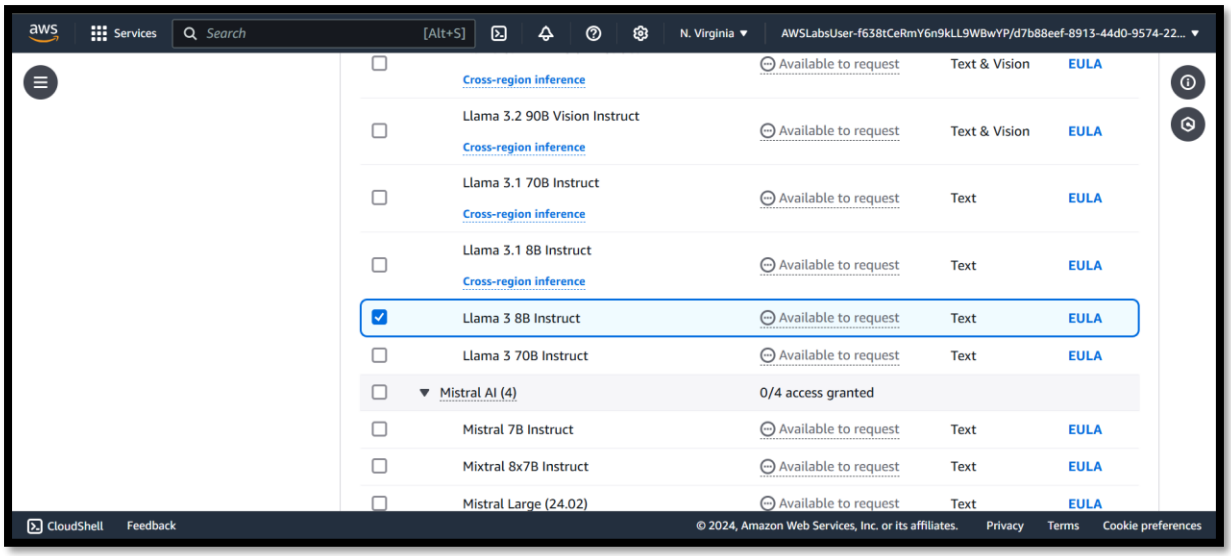


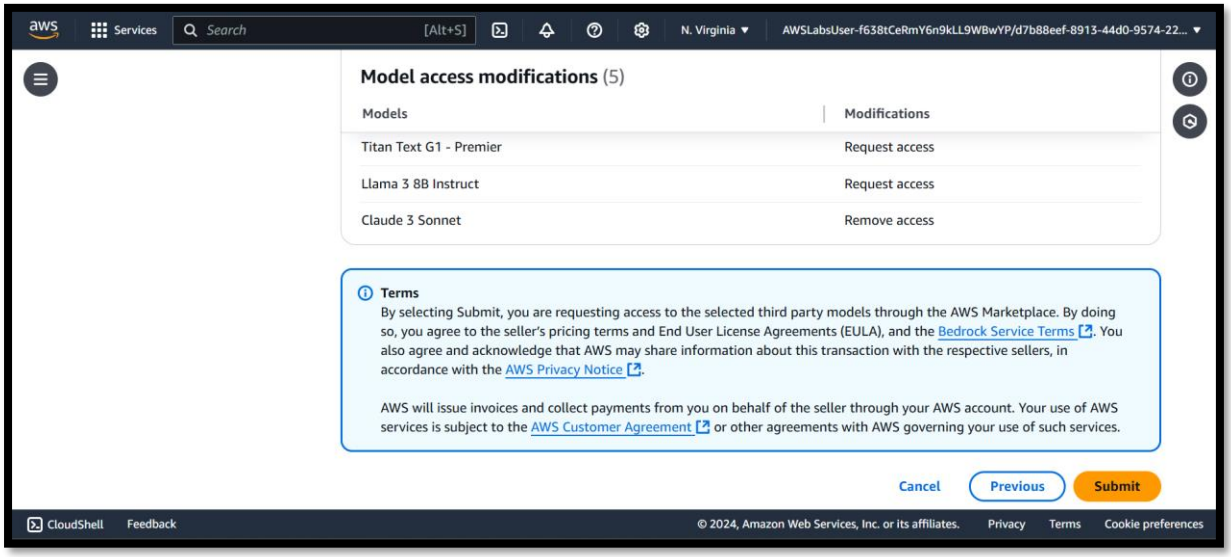


I chose Modify model access at the top of the screen.

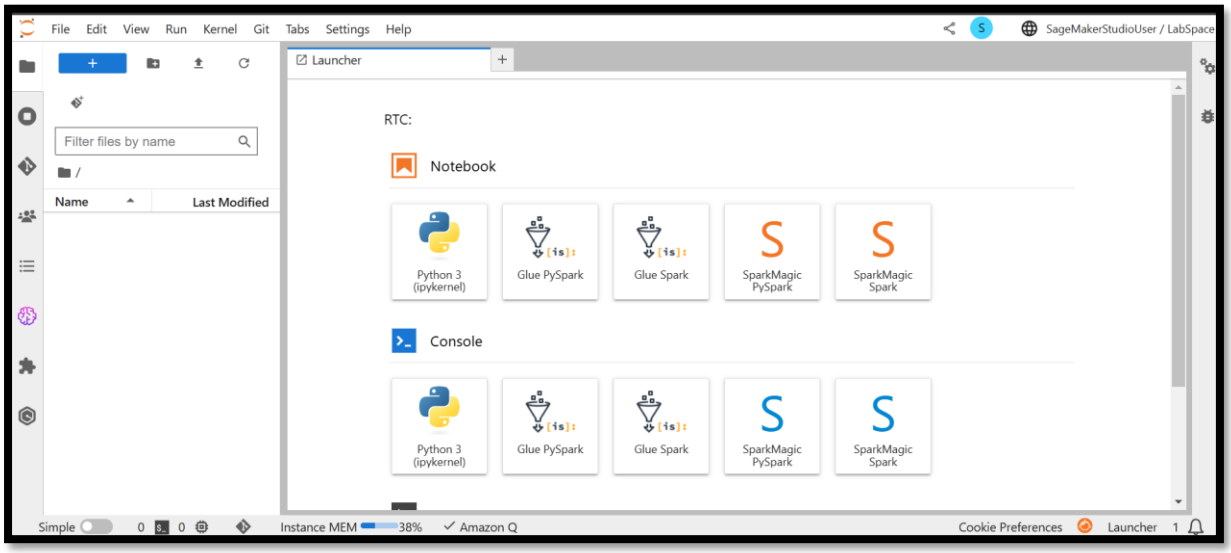





aws
Services
Search
[Alt+S]
N. Virginia
AWSLabsUser-f638tCeRmY6n9kLL9WBwYP/d7b88eef-8913-44d0-9574-22...
Cross-region inference
Available to request
Text & Vision
EULA
Llama 3.2 90B Vision Instruct
Cross-region inference
Available to request
Text & Vision
EULA
Llama 3.1 70B Instruct
Cross-region inference
Available to request
Text
EULA
Llama 3.1 8B Instruct
Cross-region inference
Available to request
Text
EULA
Llama 3 8B Instruct
Available to request
Text
EULA
Llama 3 70B Instruct
Available to request
Text
EULA
Mistral AI (4)
0/4 access granted
Mistral 7B Instruct
Available to request
Text
EULA
Mixtral 8x7B Instruct
Available to request
Text
EULA
Mistral Large (24.02)
Available to request
Text
EULA
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

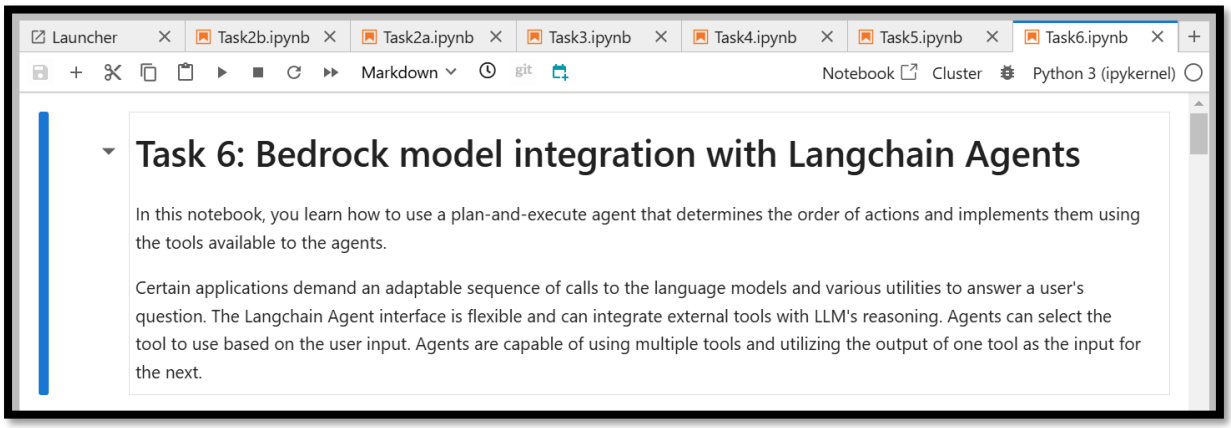

aws
Services
Search
[Alt+S]
N. Virginia
AWSLabsUser-f638tCeRmY6n9kLL9WBwYP/d7b88eef-8913-44d0-9574-22...
Model access modifications (5)
Models
Modifications
Titan Text G1 - Premier
Request access
Llama 3 8B Instruct
Request access
Claude 3 Sonnet
Remove access
Terms
By selecting Submit, you are requesting access to the selected third party models through the AWS Marketplace. By doing so, you agree to the seller's pricing terms and End User License Agreements (EULA), and the Bedrock Service Terms. You also agree and acknowledge that AWS may share information about this transaction with the respective sellers, in accordance with the AWS Privacy Notice.
AWS will issue invoices and collect payments from you on behalf of the seller through your AWS account. Your use of AWS services is subject to the AWS Customer Agreement or other agreements with AWS governing your use of such services.
Cancel
Previous
Submit
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

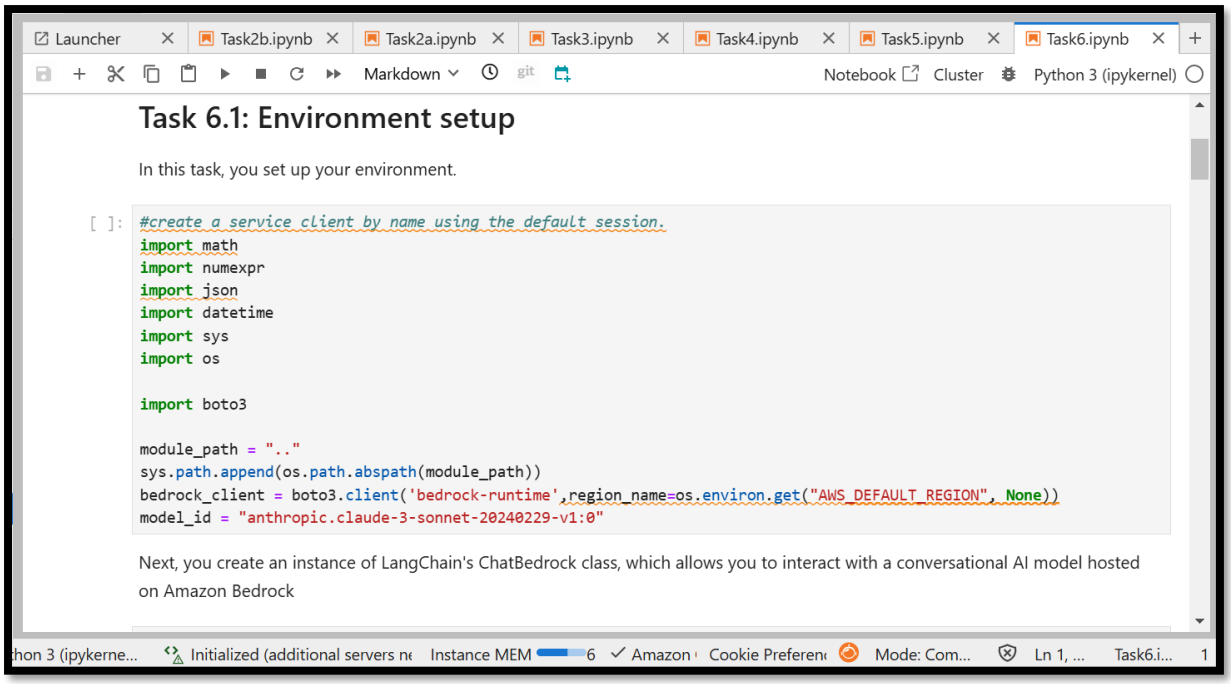
Launched an Amazon SageMaker Studio application

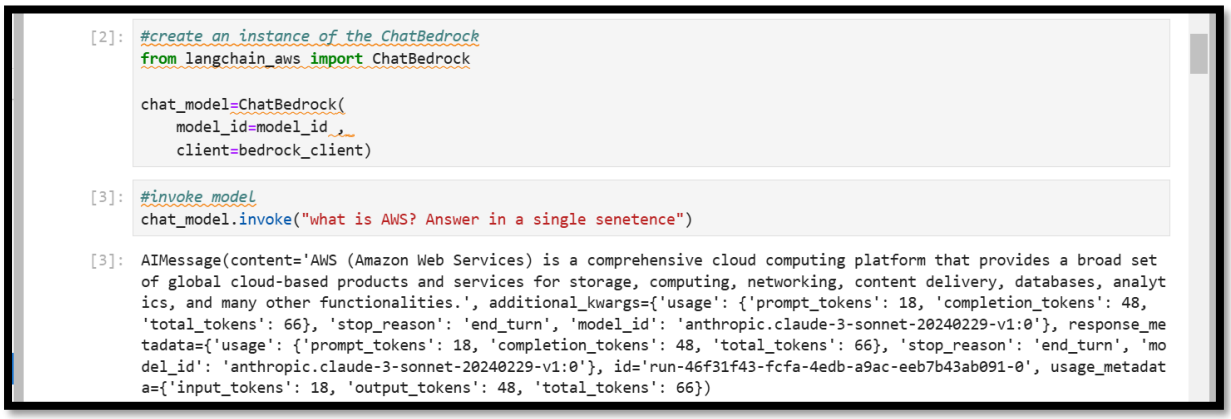


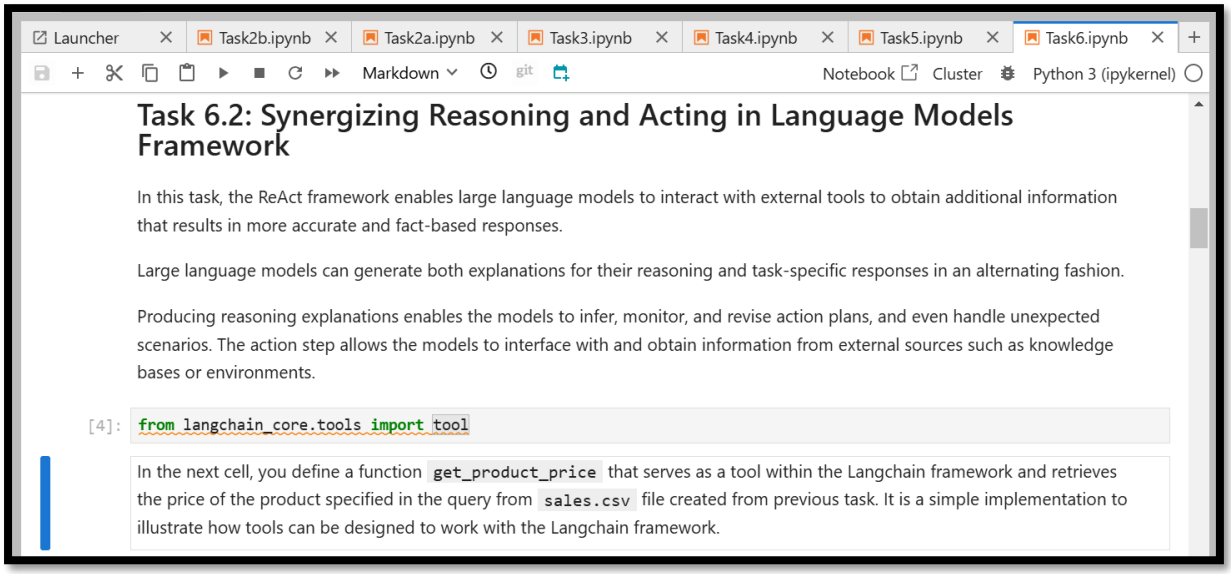
Task 1: Integrate Amazon Bedrock Models with LangChain Agents

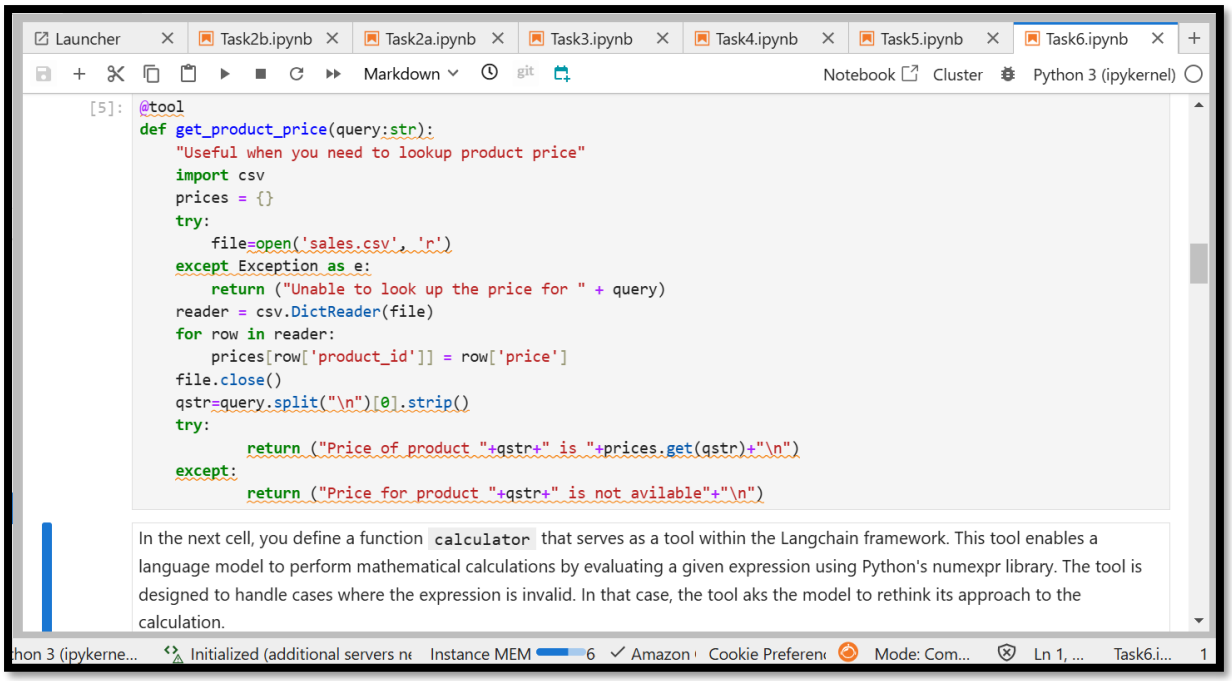
I learned how to use the flexible Converse API to integrate external capabilities into conversational applications.

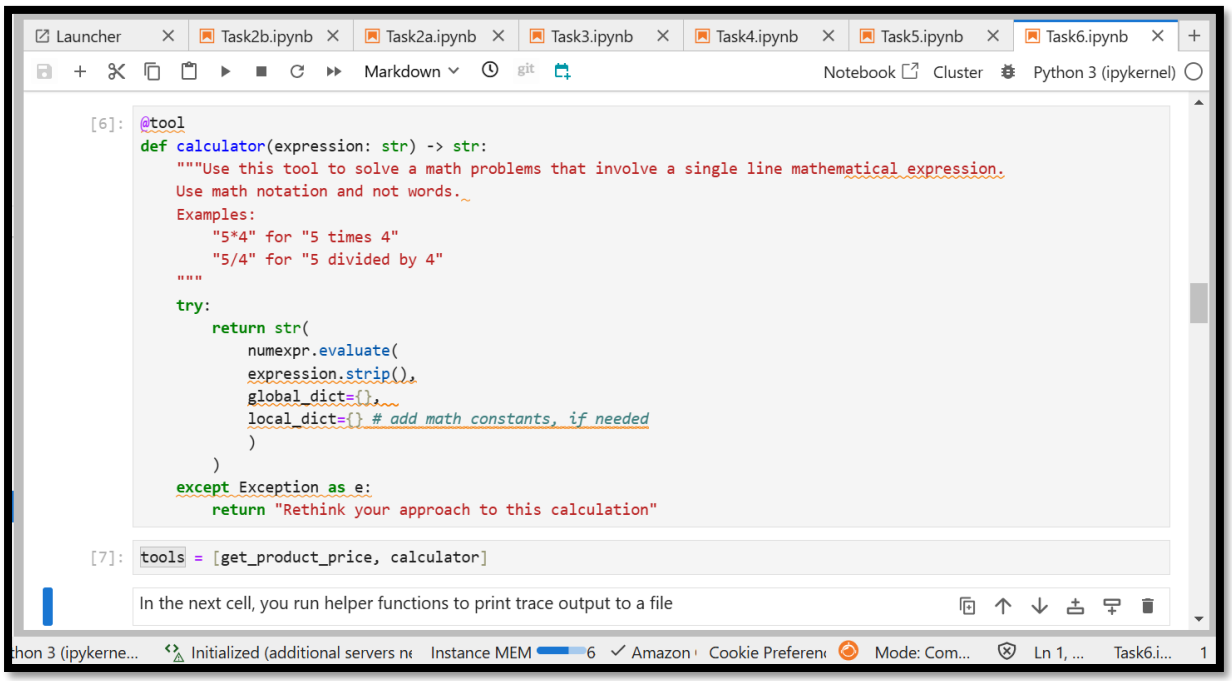



Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Markdown
Notebook
Cluster
Python 3 (ipykernel)
Task 6.1: Environment setup
In this task, you set up your environment.
[ ]:
#create a service client by name using the default session.
import math
import numexpr
import json
import datetime
import sys
import os
import boto3
module_path = ".."
sys.path.append(os.path.abspath(module_path))
bedrock_client = boto3.client('bedrock-runtime',region_name=os.environ.get("AWS_DEFAULT_REGION", None))
model_id = "anthropic.claude-3-sonnet-20240229-v1:0"
Next, you create an instance of LangChain's ChatBedrock class, which allows you to interact with a conversational AI model hosted on Amazon Bedrock
Initialized (additional servers ne
Instance MEM
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...

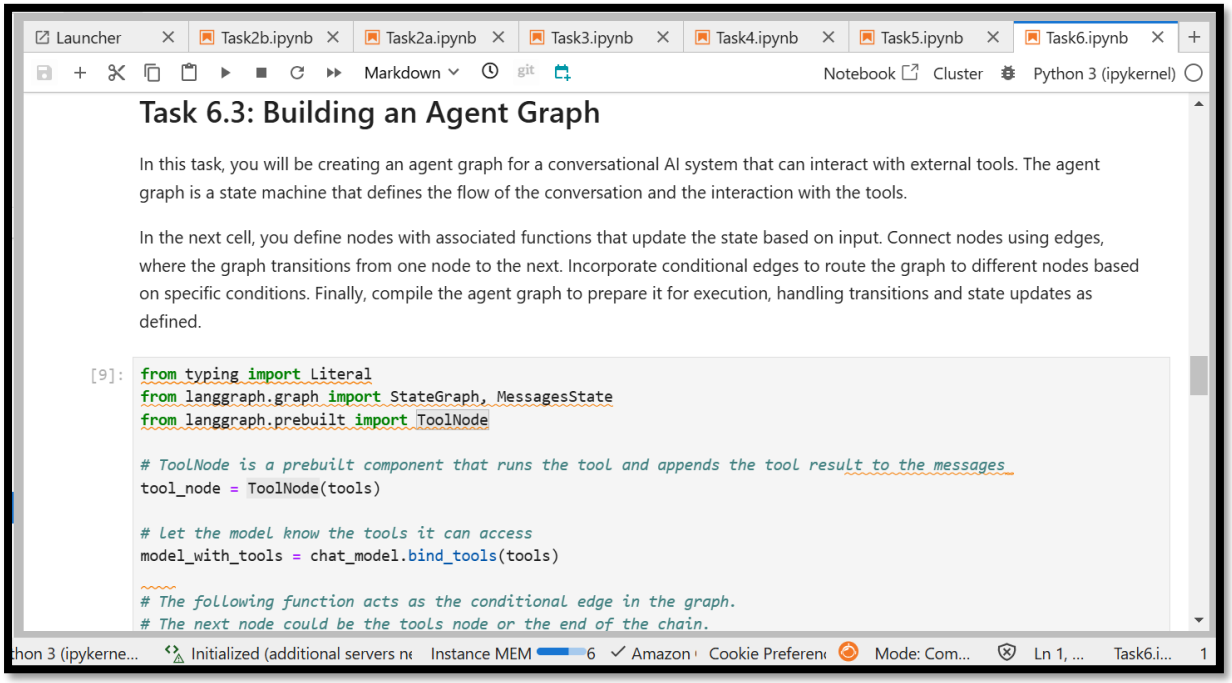

[2]:
#create an instance of the ChatBedrock
from langchain_aws import ChatBedrock
chat_model=ChatBedrock(
model_id=model_id ,
client=bedrock_client)
[3]:
#invoke model
chat_model.invoke("what is AWS? Answer in a single senetence")
[3]:
AIMessage(content='AWS (Amazon Web Services) is a comprehensive cloud computing platform that provides a broad set of global cloud-based products and services for storage, computing, networking, content delivery, databases, analytics, and many other functionalities.', additional_kwargs={'usage': {'prompt_tokens': 18, 'completion_tokens': 48, 'total_tokens': 66}, 'stop_reason': 'end_turn', 'model_id': 'anthropic.claude-3-sonnet-20240229-v1:0'}, response_metadata={'usage': {'prompt_tokens': 18, 'completion_tokens': 48, 'total_tokens': 66}, 'stop_reason': 'end_turn', 'model_id': 'anthropic.claude-3-sonnet-20240229-v1:0'}, id='run-46f31f43-fcfa-4edb-a9ac-eeb7b43ab091-0', usage_metadata={'input_tokens': 18, 'output_tokens': 48, 'total_tokens': 66})


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Markdown
Notebook
Cluster
Python 3 (ipykernel)
Task 6.2: Synergizing Reasoning and Acting in Language Models Framework
In this task, the ReAct framework enables large language models to interact with external tools to obtain additional information that results in more accurate and fact-based responses.
Large language models can generate both explanations for their reasoning and task-specific responses in an alternating fashion.
Producing reasoning explanations enables the models to infer, monitor, and revise action plans, and even handle unexpected scenarios. The action step allows the models to interface with and obtain information from external sources such as knowledge bases or environments.
[4]:
from langchain_core.tools import tool
In the next cell, you define a function get_product_price that serves as a tool within the Langchain framework and retrieves the price of the product specified in the query from sales.csv file created from previous task. It is a simple implementation to illustrate how tools can be designed to work with the Langchain framework.


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Markdown
Notebook
Cluster
Python 3 (ipykernel)
[5]:
@tool
def get_product_price(query:str):
"Useful when you need to lookup product price"
import csv
prices = {}
try:
file=open('sales.csv', 'r')
except Exception as e:
return ("Unable to look up the price for " + query)
reader = csv.DictReader(file)
for row in reader:
prices[row['product_id']] = row['price']
file.close()
qstr=query.split("\n")[0].strip()
try:
return ("Price of product "+qstr+" is "+prices.get(qstr)+"\n")
except:
return ("Price for product "+qstr+" is not avilable"+"\n")
In the next cell, you define a function calculator that serves as a tool within the Langchain framework. This tool enables a language model to perform mathematical calculations by evaluating a given expression using Python's numexpr library. The tool is designed to handle cases where the expression is invalid. In that case, the tool aks the model to rethink its approach to the calculation.
Initialized (additional servers ne
Instance MEM
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Markdown
Notebook
Cluster
Python 3 (ipykernel)
[6]:
@tool
def calculator(expression: str) -> str:
"""Use this tool to solve a math problems that involve a single line mathematical expression.
Use math notation and not words.
Examples:
"5*4" for "5 times 4"
"5/4" for "5 divided by 4"
"""
try:
return str(
numexpr.evaluate(
expression.strip(),
global_dict={},
local_dict={} # add math constants, if needed
)
)
except Exception as e:
return "Rethink your approach to this calculation"
[7]:
tools = [get_product_price, calculator]
In the next cell, you run helper functions to print trace output to a file
Initialized (additional servers ne
Instance MEM
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...

[8]:
```python
from langchain_core.messages import HumanMessage, SystemMessage, AIMessage, ToolMessage
def output_trace(element:str, trace, node=True):
    global trace_handle
    if trace_enabled:
        print(datetime.datetime.now(),file=trace_handle)
        print(("Node: " if node else "Edge: ")+ element, file=trace_handle)
        if element == "ask_model_to_reason (entry)":
            for single_trace in trace:
                print(single_trace, file=trace_handle)
        else:
            print(trace, file=trace_handle)
        print('----', file=trace_handle)

def consolidate_tool_messages(message):
    tool_messages=[]
    for msg in message:
        if isinstance(msg, ToolMessage):
            tool_messages.append(msg)
    return tool_messages
```




```python
def next_step(state: MessagesState) -> Literal["tools", "__end__"]:
    messages = state["messages"]
    last_message = messages[-1]
    if last_message.tool_calls:
        output_trace("next_step: Proceed to tools",last_message, node=False)
        return "tools"
    output_trace("next_step: Proceed to end",last_message, node=False)
    return "__end__"

#.The following node function invokes the model that has information about the available tools
def ask_model_to_reason(state: MessagesState):
    messages = state["messages"]
    output_trace("ask_model_to_reason (entry)", consolidate_tool_messages(messages))
    try:
        response = model_with_tools.invoke(messages)
    except Exception as e:
        output_trace("ask_model_to_reason", messages)
        output_trace("ask_model_to_reason", "Exception: "+str(e))
        return {"messages": [messages.append("Unable to invoke the model")]}
    output_trace("ask_model_to_reason (exit)", response)
    return {"messages": [response]}
```

```python
agent_graph = StateGraph(MessagesState)
# Describe the nodes.
# The first argument is the unique node name, and the second argument is the
# function or object that will be called when the node is reached
agent_graph.add_node("agent", ask_model_to_reason)
agent_graph.add_node("tools", tool_node)

# Connect the entry node to the agent for the graph to start running
agent_graph.add_edge("__start__", "agent")

# Once the graph transitions to the tools node, the graph will transition to the agent node
agent_graph.add_edge("tools", "agent")

# The transition out of the agent node is conditional.
# If the output from ask_model_to_reason function included a call to the tools, call the tool;
# otherwise end the chain
agent_graph.add_conditional_edges(
    "agent",
    next_step,
)

# Compile the graph definition so that it can run
react_agent = agent_graph.compile()
```

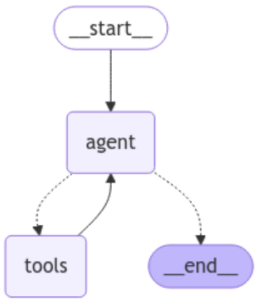
hon 3 (ipykerne... Initialized (additional servers n Instance MEM 6 Amazon Cookie Preferen Mode: Com... Ln 35, C... Task6.i... 1

Launcher | Task2b.ipynb | Task2a.ipynb | Task3.ipynb | Task4.ipynb | Task5.ipynb | Task6.ipynb

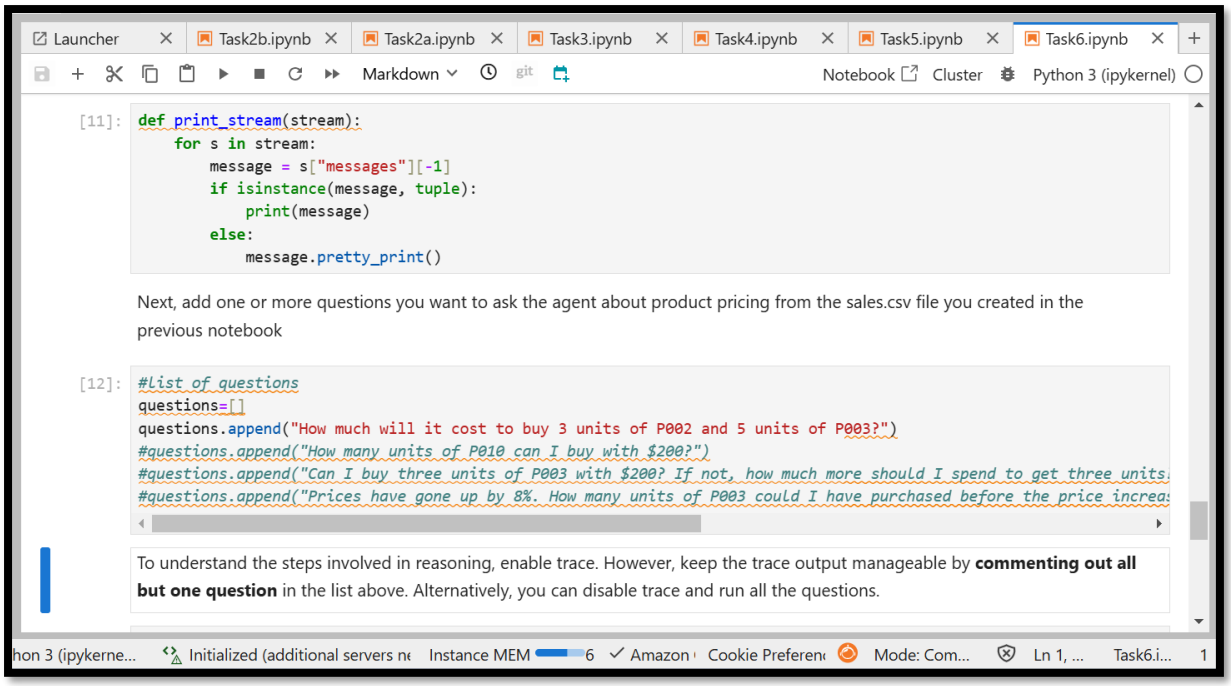
Markdown | Notebook | Cluster | Python 3 (ipykernel)

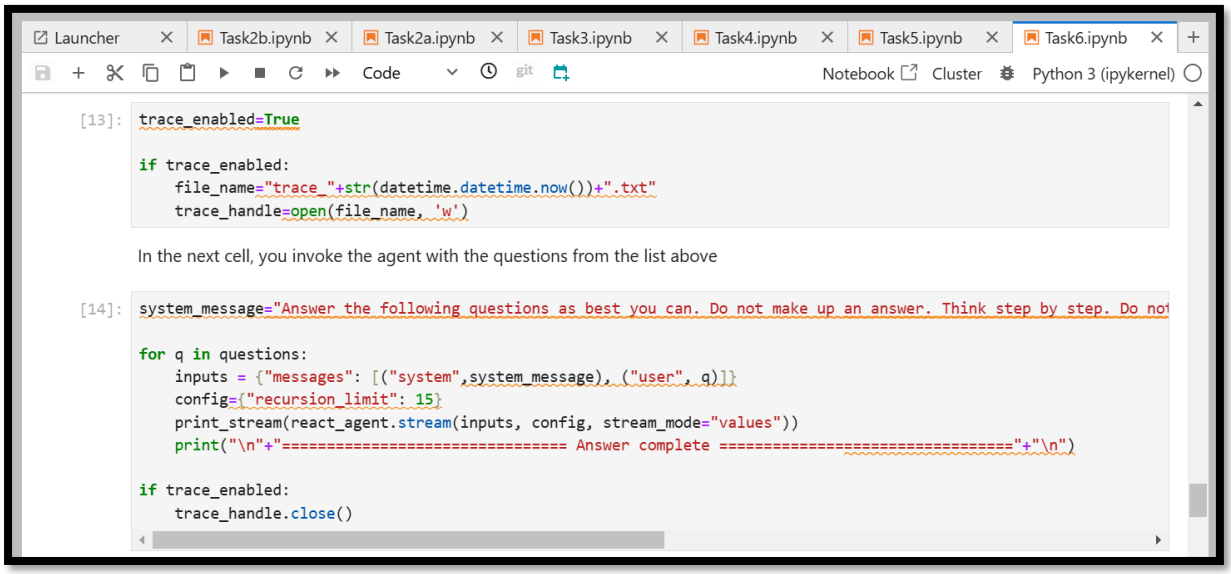
Next, you visualize the compiled graph. Observe the transition out of the agent node is conditional as indicated by the dotted line.

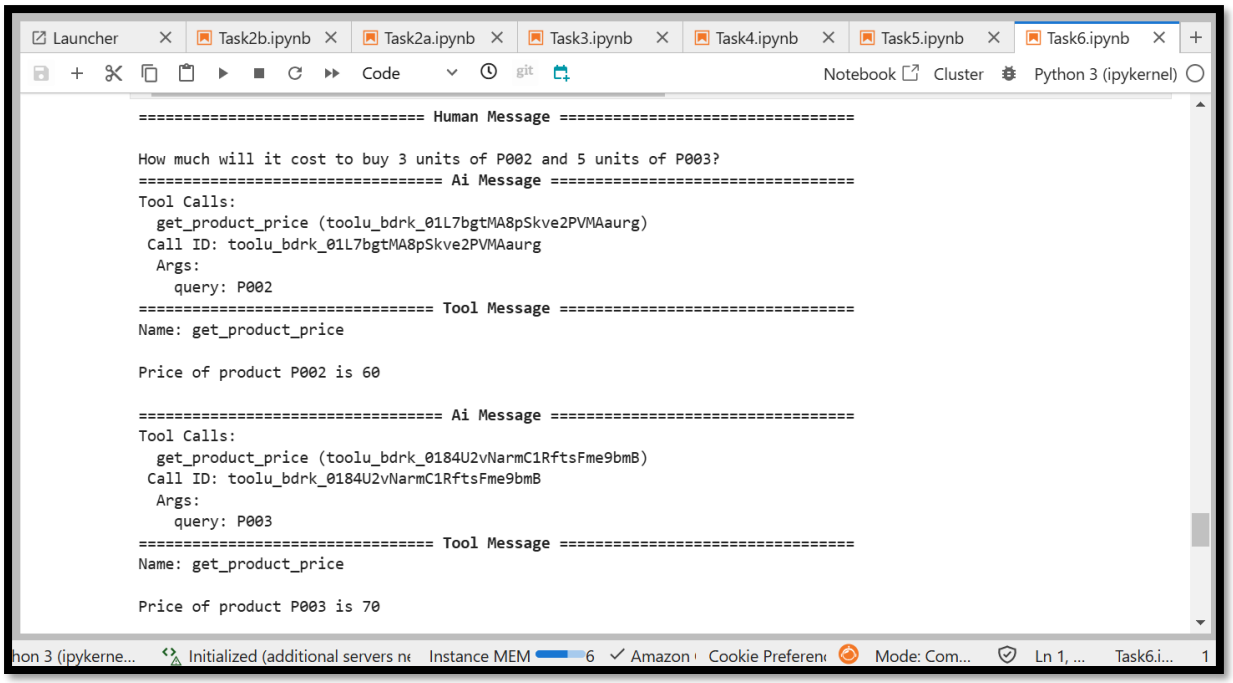
[10]:
```python
from IPython.display import Image, display

try:
    display(Image(react_agent.get_graph().draw_mermaid_png()))
except Exception:
    # This requires some extra dependencies and is optional
    pass
```

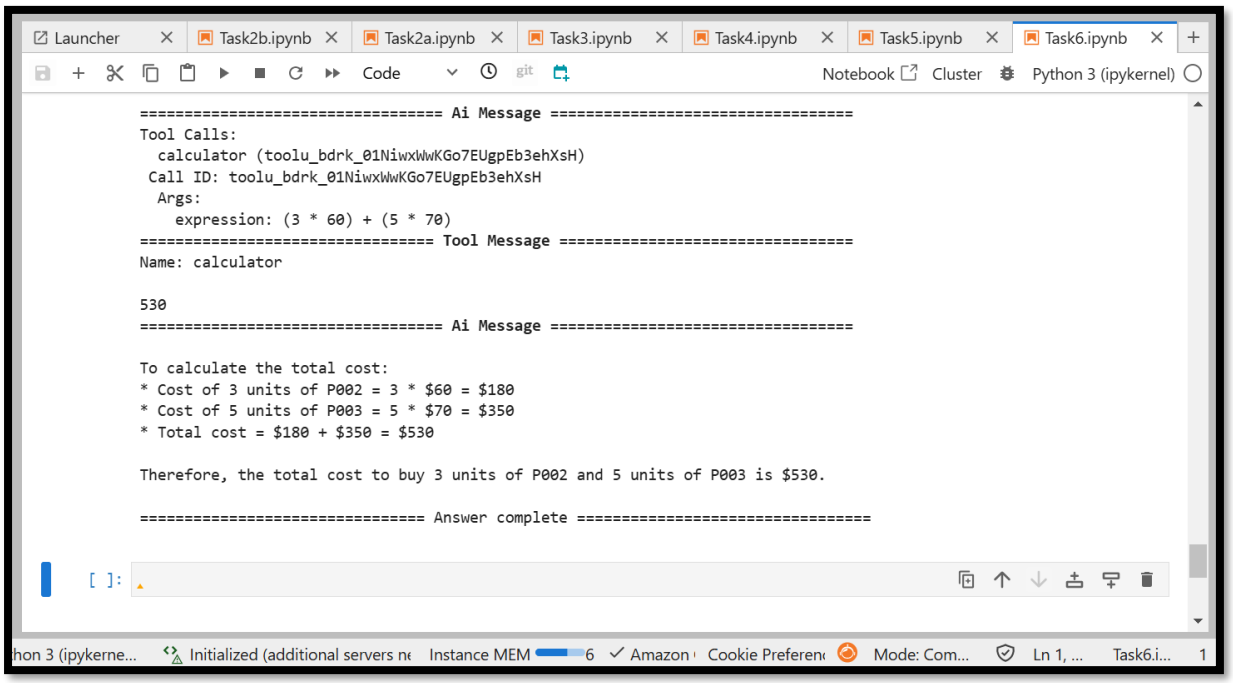


hon 3 (ipykerne... Initialized (additional servers n Instance MEM 6 Amazon Cookie Preferen Mode: Com... Ln 1, ... Task6.i... 1


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Markdown
Notebook
Cluster
Python 3 (ipykernel)
[11]:
def print_stream(stream):
for s in stream:
message = s["messages"][-1]
if isinstance(message, tuple):
print(message)
else:
message.pretty_print()
Next, add one or more questions you want to ask the agent about product pricing from the sales.csv file you created in the previous notebook
[12]:
#List of questions
questions=[]
questions.append("How much will it cost to buy 3 units of P002 and 5 units of P003?")
#questions.append("How many units of P010 can I buy with $200?")
#questions.append("Can I buy three units of P003 with $200? If not, how much more should I spend to get three units
#questions.append("Prices have gone up by 8%. How many units of P003 could I have purchased before the price increas
To understand the steps involved in reasoning, enable trace. However, keep the trace output manageable by commenting out all but one question in the list above. Alternatively, you can disable trace and run all the questions.
hon 3 (ipykerne...
Initialized (additional servers ne
Instance MEM
6
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...
1


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Code
Notebook
Cluster
Python 3 (ipykernel)
[13]:
trace_enabled=True
if trace_enabled:
file_name="trace_"+str(datetime.datetime.now())+".txt"
trace_handle=open(file_name, 'w')
In the next cell, you invoke the agent with the questions from the list above
[14]:
system_message="Answer the following questions as best you can. Do not make up an answer. Think step by step. Do not
for q in questions:
inputs = {"messages": [("system",system_message), ("user", q)]}
config={"recursion_limit": 15}
print_stream(react_agent.stream(inputs, config, stream_mode="values"))
print("\n"+"================================ Answer complete ================================="+"\n")
if trace_enabled:
trace_handle.close()


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Code
Notebook
Cluster
Python 3 (ipykernel)
========== Human Message ==========
How much will it cost to buy 3 units of P002 and 5 units of P003?
========== Ai Message ==========
Tool Calls:
get_product_price (toolu_bdrk_01L7bgtMA8pSkve2PVMAaurg)
Call ID: toolu_bdrk_01L7bgtMA8pSkve2PVMAaurg
Args:
query: P002
========== Tool Message ==========
Name: get_product_price
Price of product P002 is 60
========== Ai Message ==========
Tool Calls:
get_product_price (toolu_bdrk_0184U2vNarmC1RftsFme9bmB)
Call ID: toolu_bdrk_0184U2vNarmC1RftsFme9bmB
Args:
query: P003
========== Tool Message ==========
Name: get_product_price
Price of product P003 is 70
Initialized (additional servers ne
Instance MEM
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...


Launcher
Task2b.ipynb
Task2a.ipynb
Task3.ipynb
Task4.ipynb
Task5.ipynb
Task6.ipynb
Code
Notebook
Cluster
Python 3 (ipykernel)
========== Ai Message ==========
Tool Calls:
calculator (toolu_bdrk_01NiwxWwKGo7EUgpEb3ehXsH)
Call ID: toolu_bdrk_01NiwxWwKGo7EUgpEb3ehXsH
Args:
expression: (3 * 60) + (5 * 70)
========== Tool Message ==========
Name: calculator
530
========== Ai Message ==========
To calculate the total cost:
* Cost of 3 units of P002 = 3 * $60 = $180
* Cost of 5 units of P003 = 5 * $70 = $350
* Total cost = $180 + $350 = $530
Therefore, the total cost to buy 3 units of P002 and 5 units of P003 is $530.
========== Answer complete ==========
[ ]:
Initialized (additional servers ne
Instance MEM
Amazon
Cookie Preferen
Mode: Com...
Ln 1, ...
Task6.i...