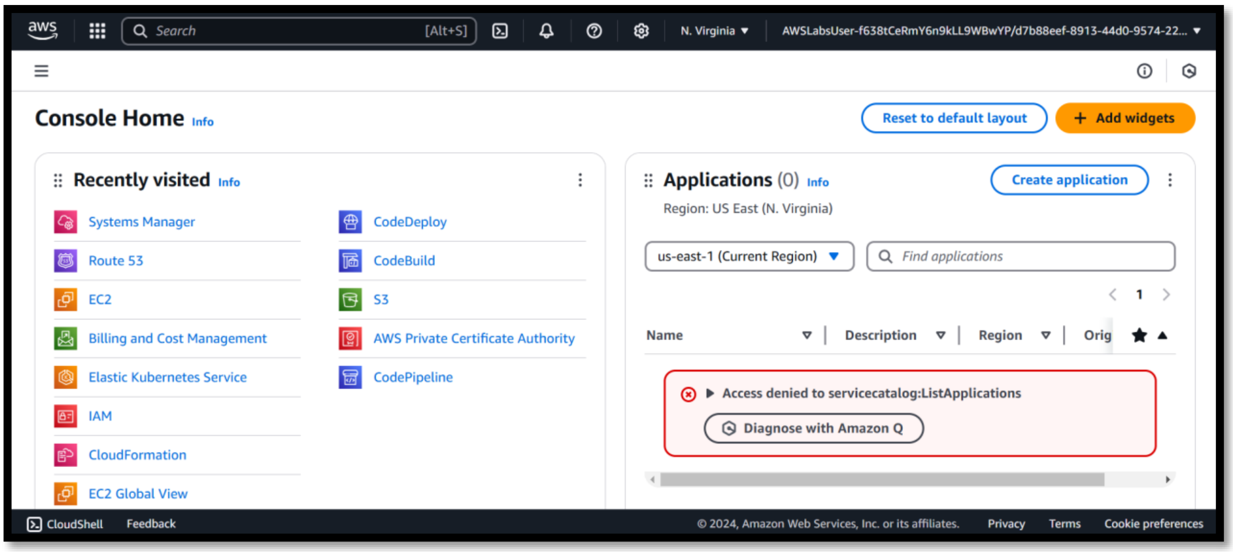
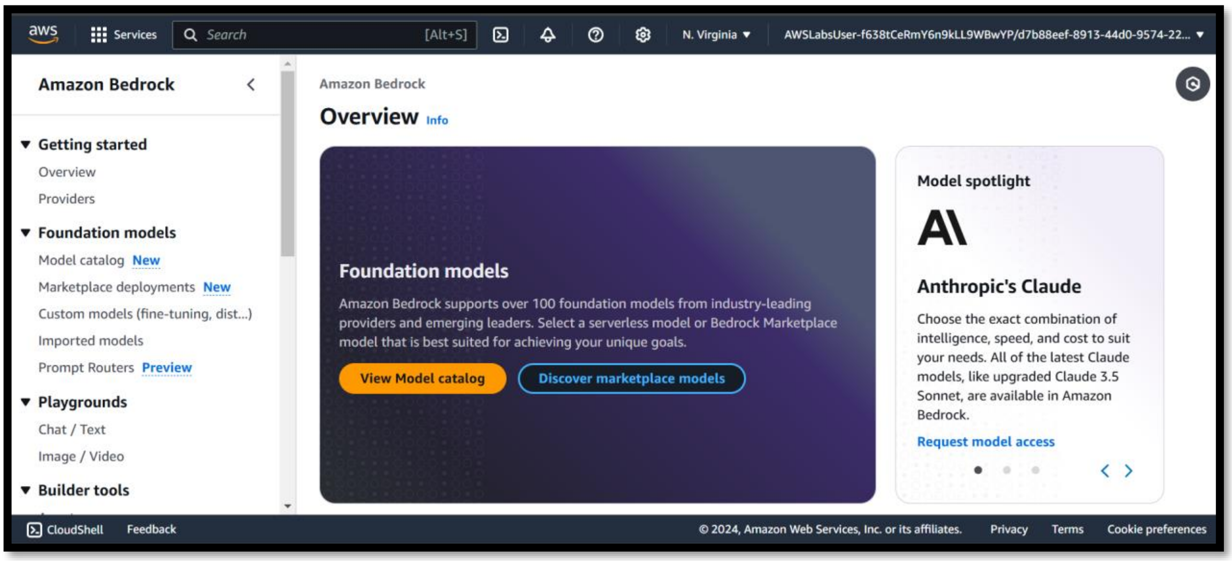
Objective: To run two notebook files: Task1a.ipynb, which invokes an Amazon Bedrock model for text generation using a zero-shot prompt, and Task1b.ipynb, which uses the LangChain framework to communicate with the Amazon Bedrock API and creates a custom LangChain prompt template to add context to the text generation request.

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.



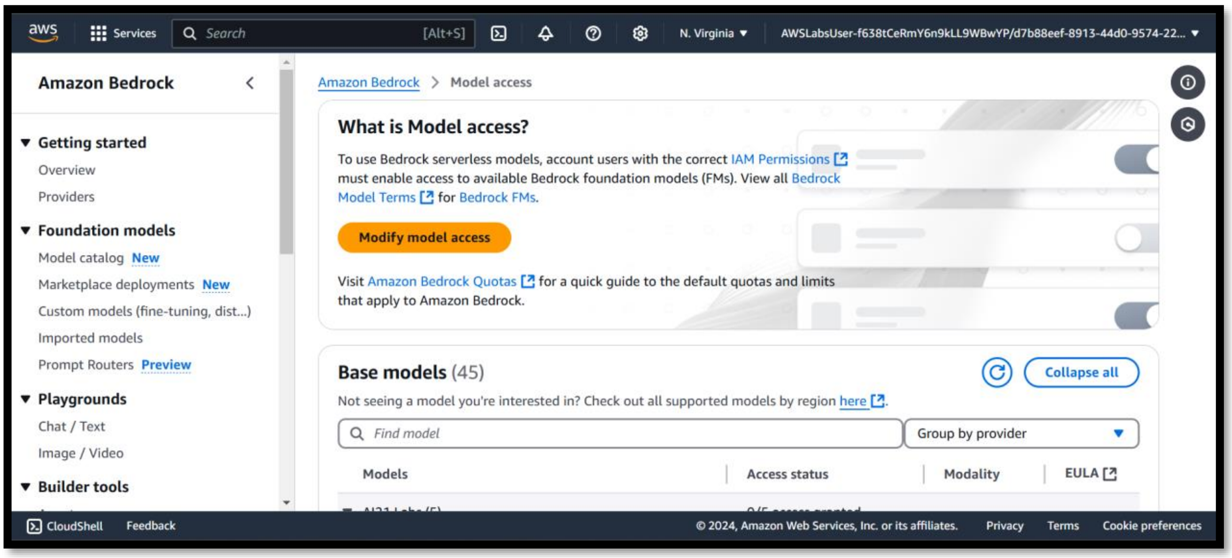
Access denied to servicecatalog:ListApplications



Amazon Bedrock supports over 100 foundation models from industry-leading providers and emerging leaders. Select a serverless model or Bedrock Marketplace model that is best suited for achieving your unique goals.

Choose the exact combination of intelligence, speed, and cost to suit your needs. All of the latest Claude models, like upgraded Claude 3.5 Sonnet, are available in Amazon Bedrock.

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.



aws Services Search [Alt+S] N. Virginia AWSLabsUser-f638tCeRmY6n9kLL9WBwYP/d7b88eef-8913-44d0-9574-22...

Amazon Bedrock

- Getting started
  - Overview
  - Providers
- Foundation models
  - Model catalog New
  - Marketplace deployments New
  - Custom models (fine-tuning, dist...)
  - Imported models
  - Prompt Routers Preview
- Playgrounds
  - Chat / Text
  - Image / Video
- Builder tools

Amazon Bedrock > Model access

What is Model access?

To use Bedrock serverless models, account users with the correct IAM Permissions must enable access to available Bedrock foundation models (FMs). View all Bedrock Model Terms for Bedrock FMs.

Modify model access

Visit Amazon Bedrock Quotas for a quick guide to the default quotas and limits that apply to Amazon Bedrock.

Base models (45)

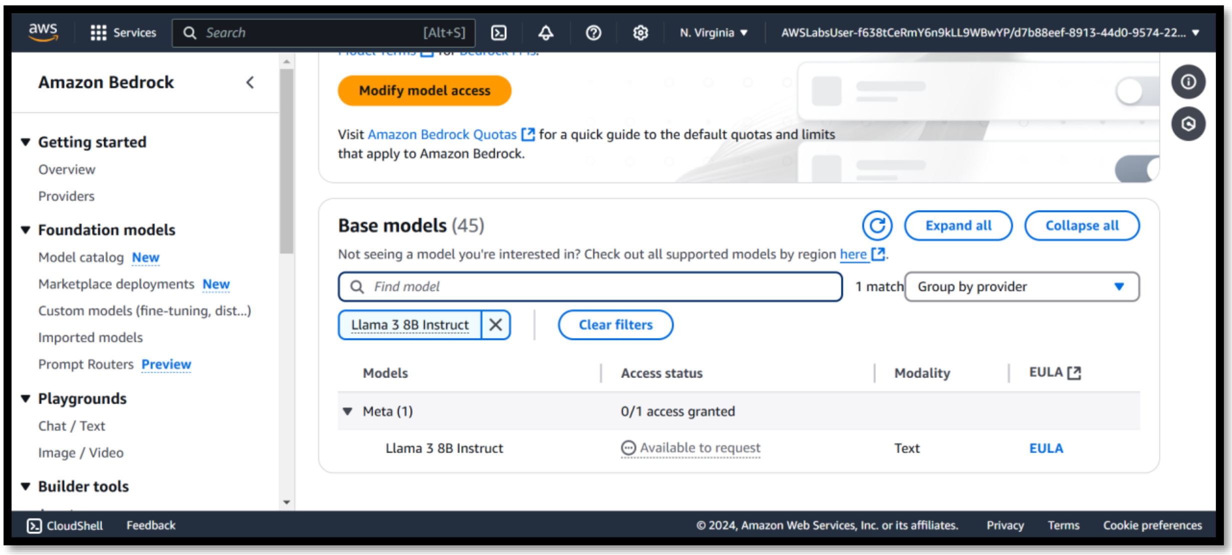
Collapse all

Not seeing a model you're interested in? Check out all supported models by region here.

Find model | Group by provider

Models	Access status	Modality	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...

Amazon Bedrock

- Getting started
  - Overview
  - Providers
- Foundation models
  - Model catalog New
  - Marketplace deployments New
  - Custom models (fine-tuning, dist...)
  - Imported models
  - Prompt Routers Preview
- Playgrounds
  - Chat / Text
  - Image / Video
- Builder tools

Modify model access

Visit Amazon Bedrock Quotas for a quick guide to the default quotas and limits that apply to Amazon Bedrock.

Base models (45)

Expand all Collapse all

Not seeing a model you're interested in? Check out all supported models by region here.

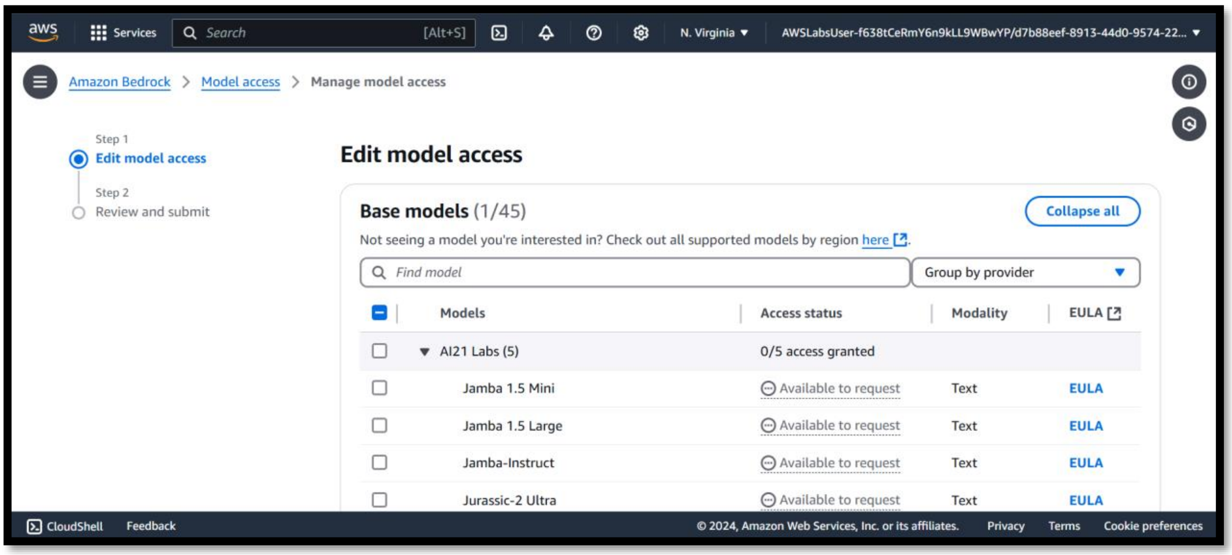
Find model 1 match Group by provider

Llama 3 8B Instruct ✕ Clear filters

Models	Access status	Modality	EULA
▼ Meta (1)	0/1 access granted		
Llama 3 8B Instruct	Available to request	Text	EULA

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

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



Amazon Bedrock > Model access > Manage model access

Step 1
Edit model access

Step 2
Review and submit

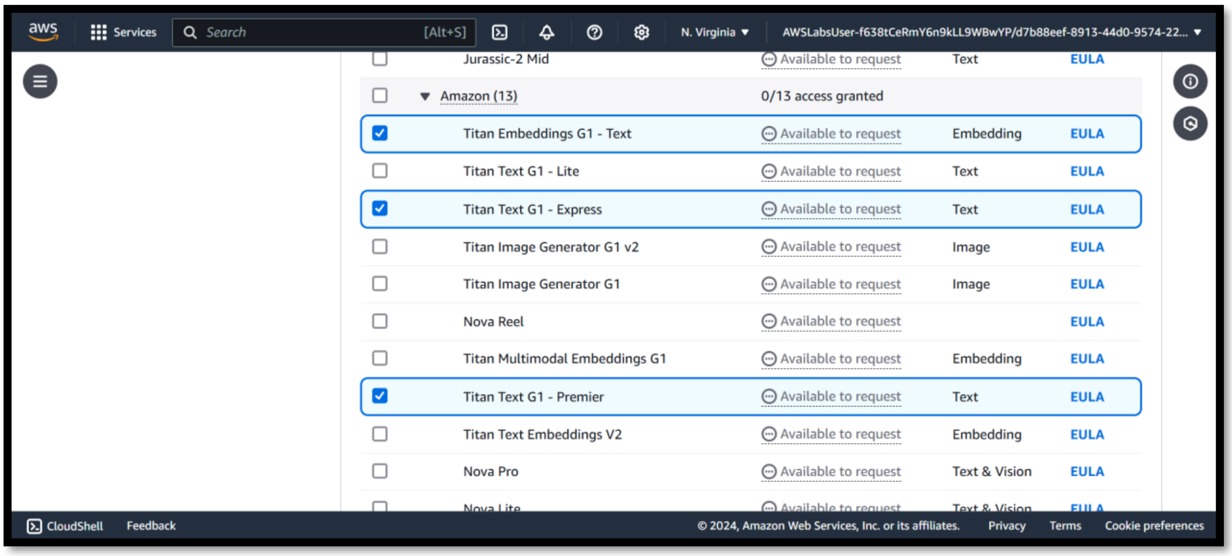
Edit model access

Base models (1/45)

Not seeing a model you're interested in? Check out all supported models by region here.

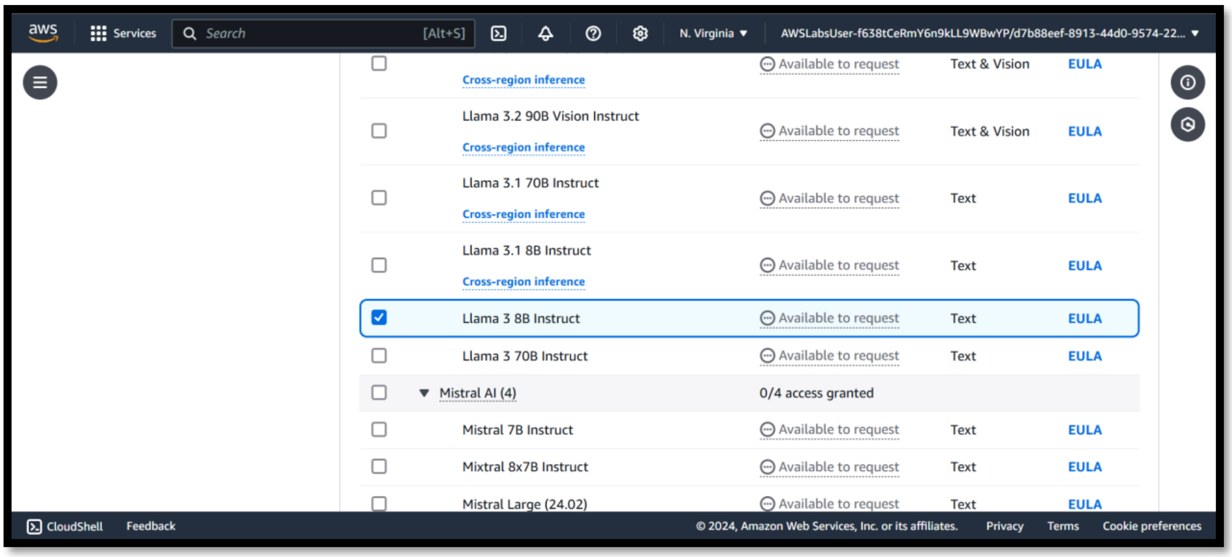
	Models	Access status	Modality	EULA
☐	▼ AI21 Labs (5)	0/5 access granted		
☐	Jamba 1.5 Mini	Available to request	Text	EULA
☐	Jamba 1.5 Large	Available to request	Text	EULA
☐	Jamba-Instruct	Available to request	Text	EULA
☐	Jurassic-2 Ultra	Available to request	Text	EULA

© 2024, Amazon Web Services, Inc. or its affiliates.



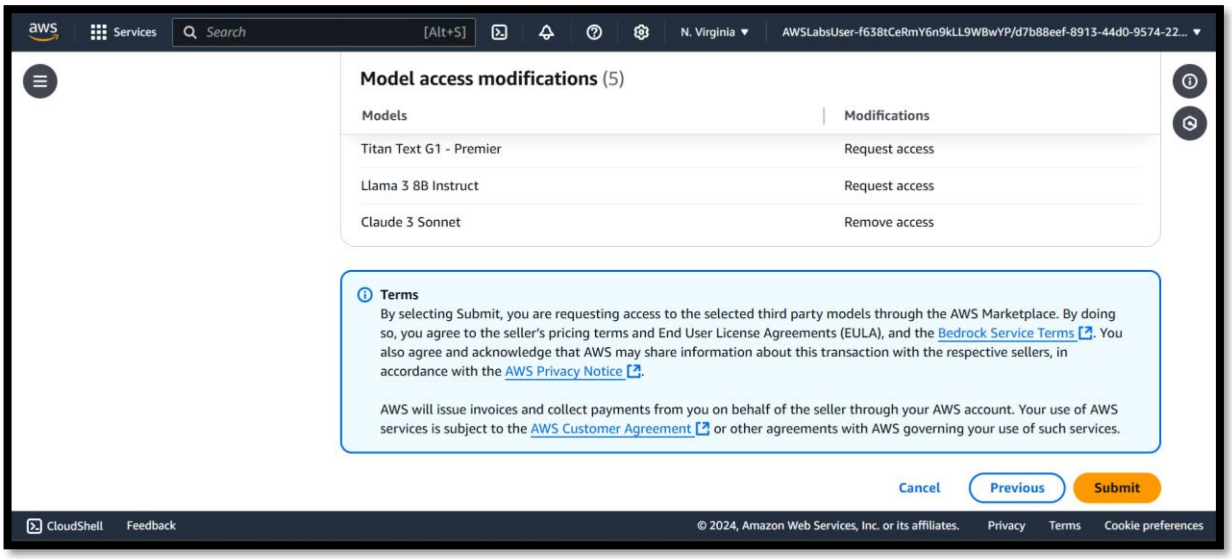
	Models	Access status	Modality	EULA
☐	Jurassic-2 Mid	Available to request	Text	EULA
☐	▼ Amazon (13)	0/13 access granted		
✓	Titan Embeddings G1 - Text	Available to request	Embedding	EULA
☐	Titan Text G1 - Lite	Available to request	Text	EULA
✓	Titan Text G1 - Express	Available to request	Text	EULA
☐	Titan Image Generator G1 v2	Available to request	Image	EULA
☐	Titan Image Generator G1	Available to request	Image	EULA
☐	Nova Reel	Available to request		EULA
☐	Titan Multimodal Embeddings G1	Available to request	Embedding	EULA
✓	Titan Text G1 - Premier	Available to request	Text	EULA
☐	Titan Text Embeddings V2	Available to request	Embedding	EULA
☐	Nova Pro	Available to request	Text & Vision	EULA
☐	Nova Lite	Available to request	Text & Vision	EULA

© 2024, Amazon Web Services, Inc. or its affiliates.



	Models	Access status	Modality	EULA
☐	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

aws
Services
Search [Alt+S]
N. Virginia
AWSLabsUser-f638tCeRmY6n9kLL9WBwYP/d7b88eef-8913-44d0-9574-22...
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Model access modifications (5)

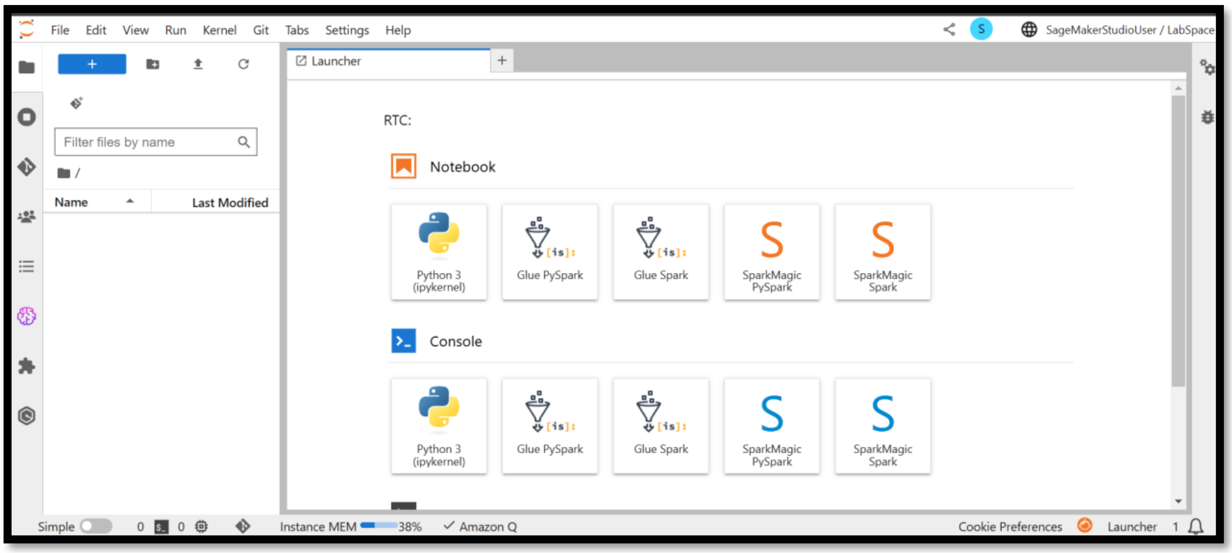
Models	Modifications
Titan Text G1 - Premier	Request access
Llama 3 8B Instruct	Request access
Claude 3 Sonnet	Remove access

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.

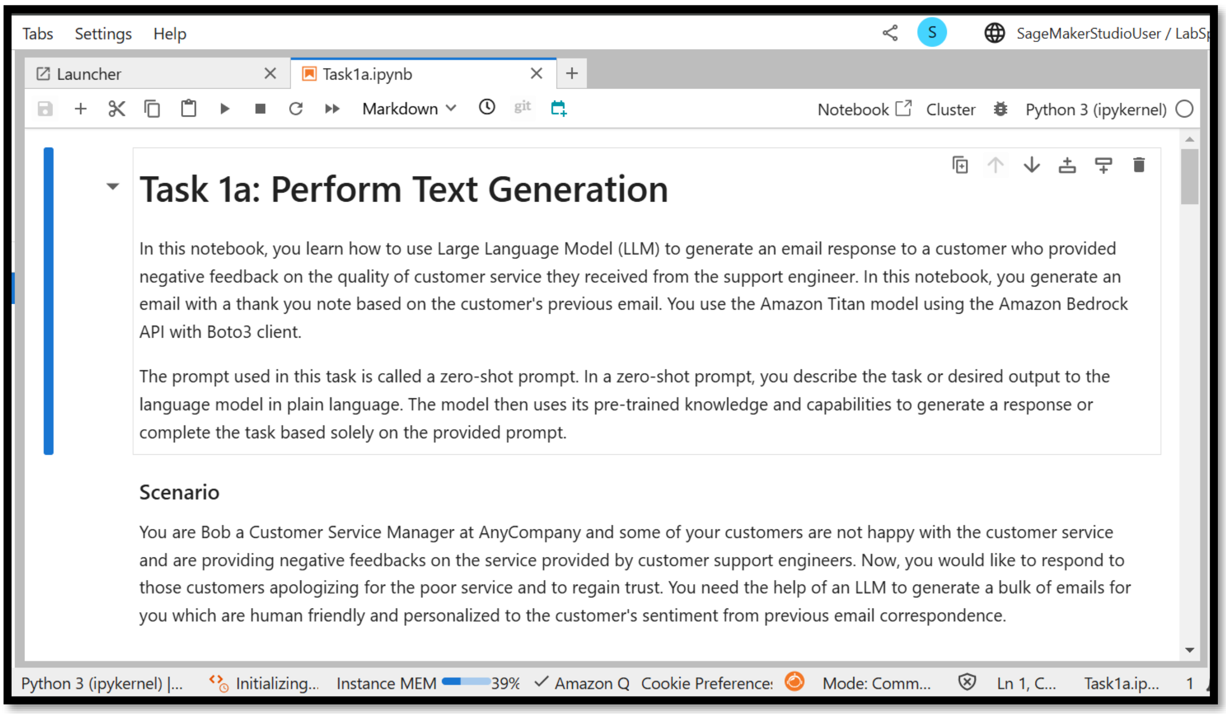
aws
Services
Search [Alt+S]
N. Virginia
AWSLabsUser-f638tCeRmY6n9kLL9WBwYP/d7b88eef-8913-44d0-9574-22...
Cancel
Previous
Submit
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Launch an Amazon SageMaker Studio application



Task 1: Perform Text Generation

Task1a.ipynb, which invoked an Amazon Bedrock model for text generation using a zero-shot prompt



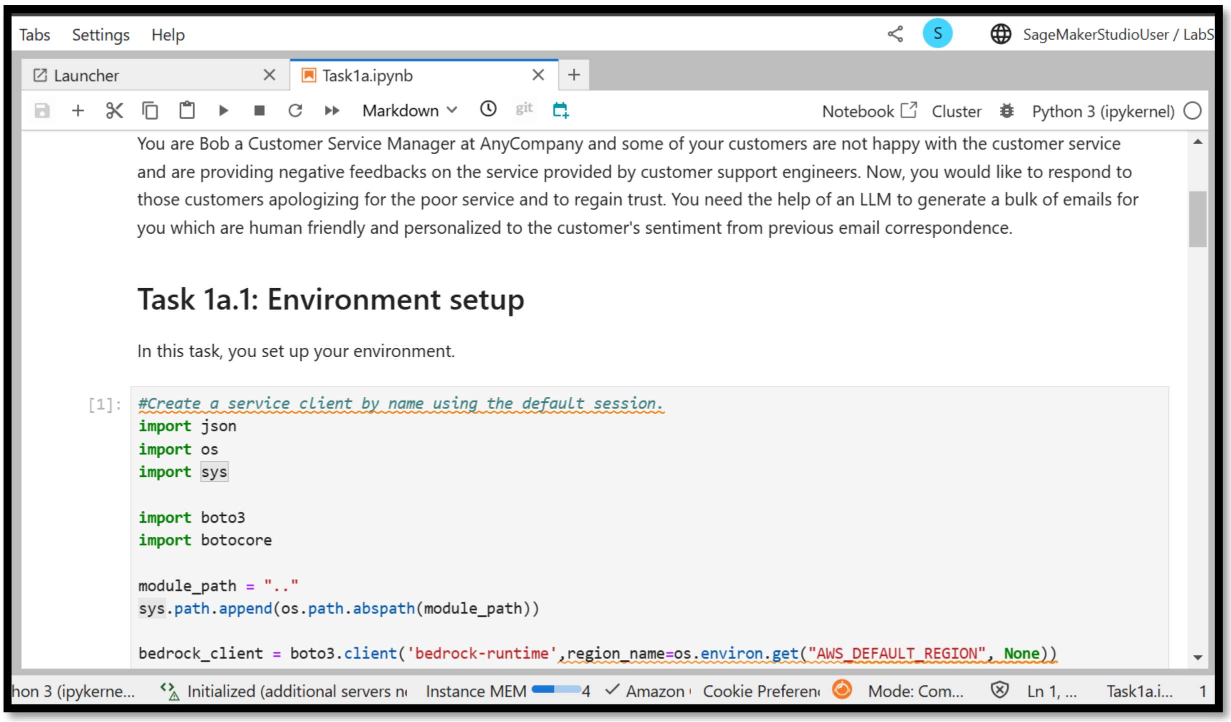
Task 1a: Perform Text Generation

In this notebook, you learn how to use Large Language Model (LLM) to generate an email response to a customer who provided negative feedback on the quality of customer service they received from the support engineer. In this notebook, you generate an email with a thank you note based on the customer's previous email. You use the Amazon Titan model using the Amazon Bedrock API with Boto3 client.

The prompt used in this task is called a zero-shot prompt. In a zero-shot prompt, you describe the task or desired output to the language model in plain language. The model then uses its pre-trained knowledge and capabilities to generate a response or complete the task based solely on the provided prompt.

Scenario

You are Bob a Customer Service Manager at AnyCompany and some of your customers are not happy with the customer service and are providing negative feedbacks on the service provided by customer support engineers. Now, you would like to respond to those customers apologizing for the poor service and to regain trust. You need the help of an LLM to generate a bulk of emails for you which are human friendly and personalized to the customer's sentiment from previous email correspondence.

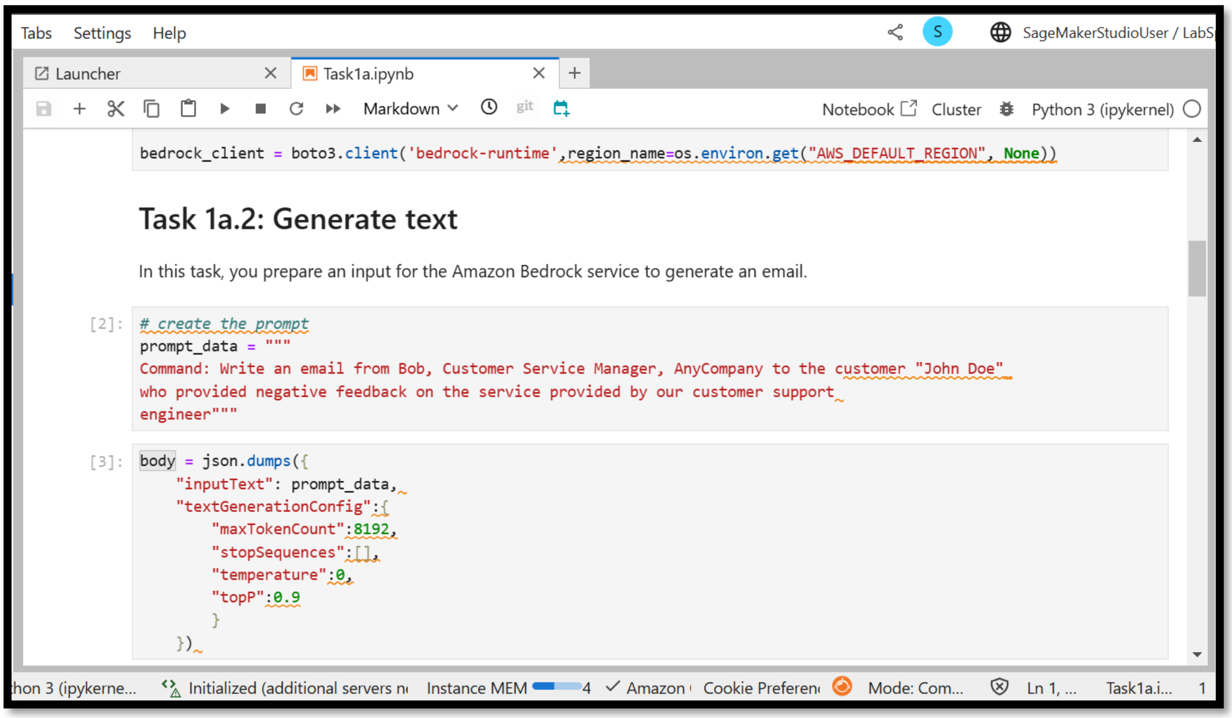


You are Bob a Customer Service Manager at AnyCompany and some of your customers are not happy with the customer service and are providing negative feedbacks on the service provided by customer support engineers. Now, you would like to respond to those customers apologizing for the poor service and to regain trust. You need the help of an LLM to generate a bulk of emails for you which are human friendly and personalized to the customer's sentiment from previous email correspondence.

Task 1a.1: Environment setup

In this task, you set up your environment.

```
[1]: #Create a service client by name using the default session.
import json
import os
import sys

import boto3
import botocore

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))
```



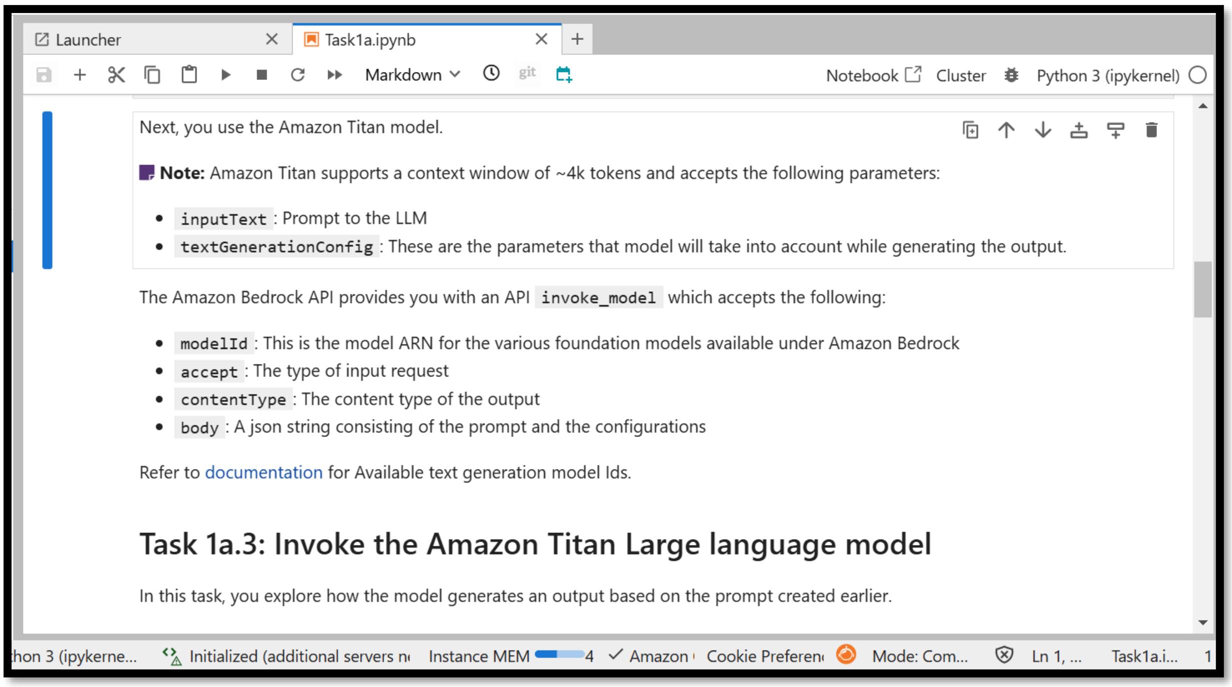
```
bedrock_client = boto3.client('bedrock-runtime',region_name=os.environ.get("AWS_DEFAULT_REGION", None))
```

Task 1a.2: Generate text

In this task, you prepare an input for the Amazon Bedrock service to generate an email.

```
[2]: # create the prompt
prompt_data = """
Command: Write an email from Bob, Customer Service Manager, AnyCompany to the customer "John Doe"
who provided negative feedback on the service provided by our customer support
engineer"""
```

```
[3]: body = json.dumps({
    "inputText": prompt_data,
    "textGenerationConfig":{
        "maxTokenCount":8192,
        "stopSequences":[],
        "temperature":0,
        "topP":0.9
        }
    })
```



Next, you use the Amazon Titan model.

Note: Amazon Titan supports a context window of ~4k tokens and accepts the following parameters:

- `inputText` : Prompt to the LLM
- `textGenerationConfig` : These are the parameters that model will take into account while generating the output.

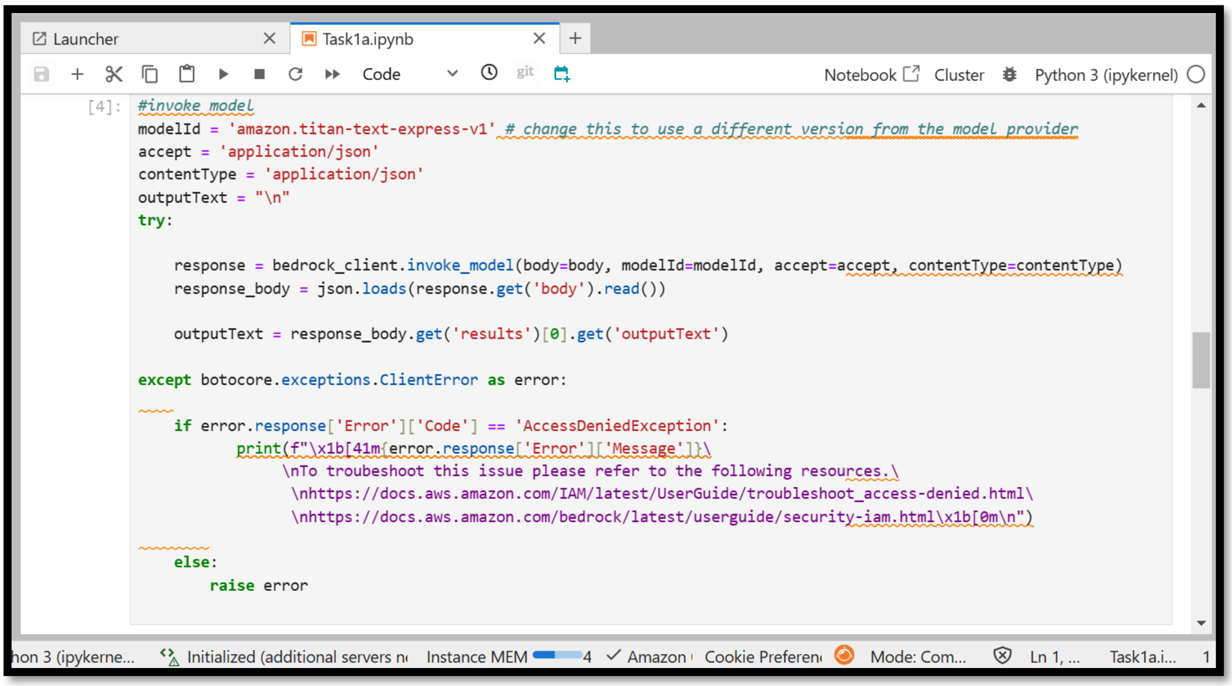
The Amazon Bedrock API provides you with an API `invoke_model` which accepts the following:

- `modelId` : This is the model ARN for the various foundation models available under Amazon Bedrock
- `accept` : The type of input request
- `contentType` : The content type of the output
- `body` : A json string consisting of the prompt and the configurations

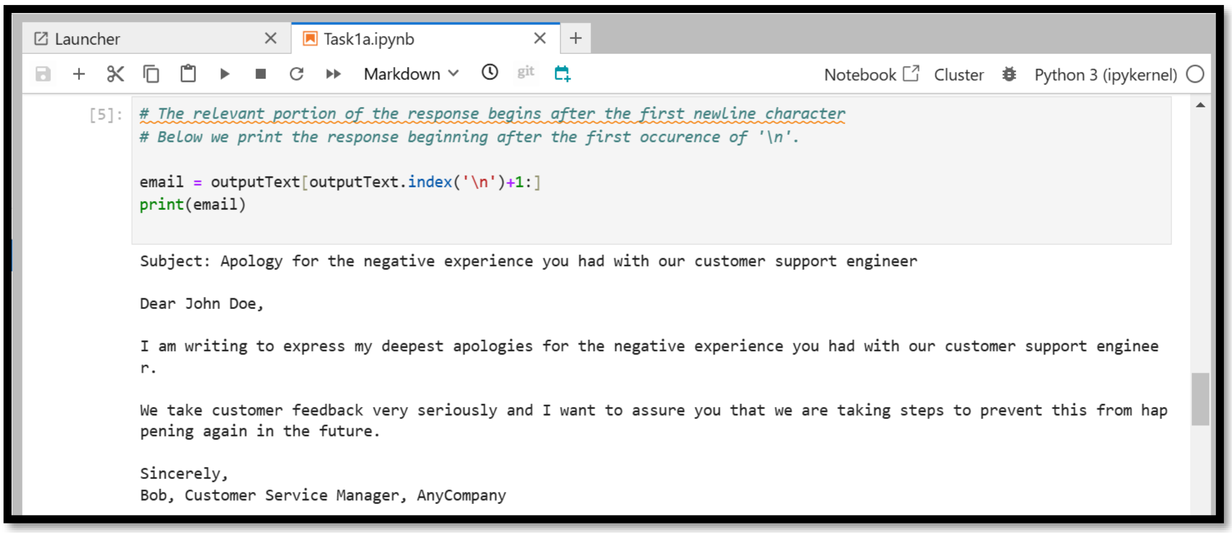
Refer to documentation for Available text generation model Ids.

Task 1a.3: Invoke the Amazon Titan Large language model

In this task, you explore how the model generates an output based on the prompt created earlier.

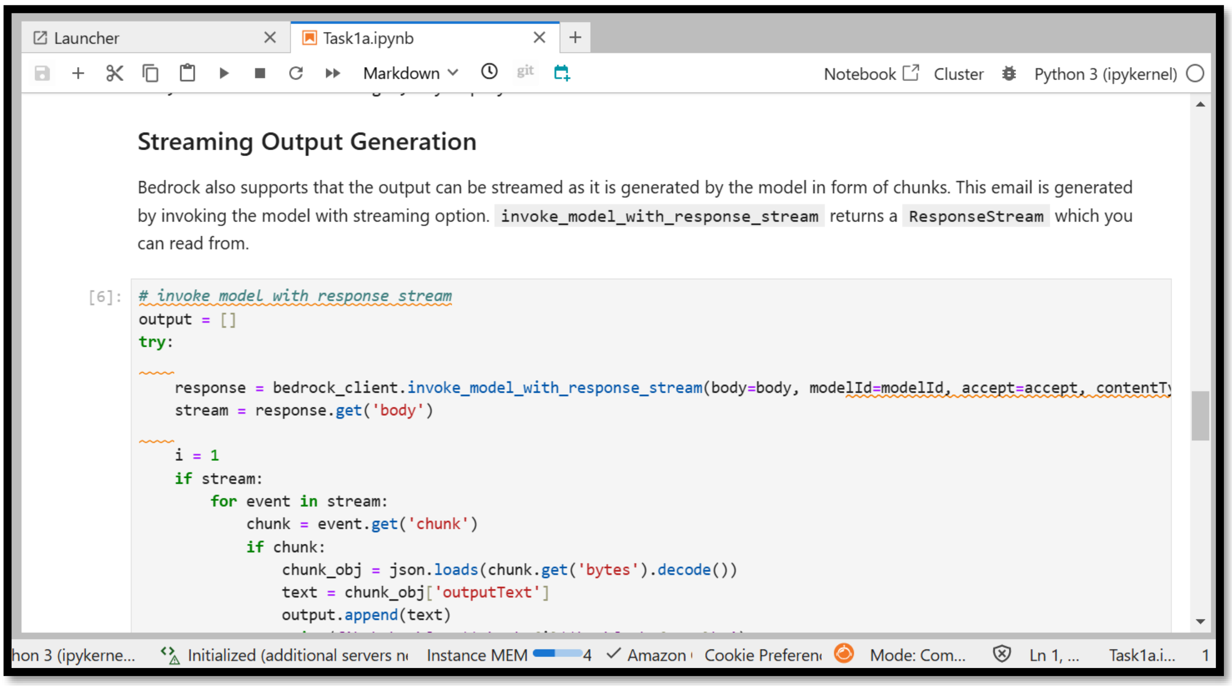


```
[4]: #invoke model
modelId = 'amazon.titan-text-express-v1' # change this to use a different version from the model provider
accept = 'application/json'
contentType = 'application/json'
outputText = "\n"
try:

    response = bedrock_client.invoke_model(body=body, modelId=modelId, accept=accept, contentType=contentType)
    response_body = json.loads(response.get('body').read())

    outputText = response_body.get('results')[0].get('outputText')

except botocore.exceptions.ClientError as error:

    if error.response['Error']['Code'] == 'AccessDeniedException':
           print(f"\x1b[41m{error.response['Error']['Message']}\
                \nTo troubeshoot this issue please refer to the following resources.\
                 \nhttps://docs.aws.amazon.com/IAM/latest/UserGuide/troubleshoot_access-denied.html\
                 \nhttps://docs.aws.amazon.com/bedrock/latest/userguide/security-iam.html\x1b[0m\n")

    else:
        raise error
```



[5]:
```
# The relevant portion of the response begins after the first newline character
# Below we print the response beginning after the first occurence of '\n'.

email = outputText[outputText.index('\n')+1:]
print(email)
```

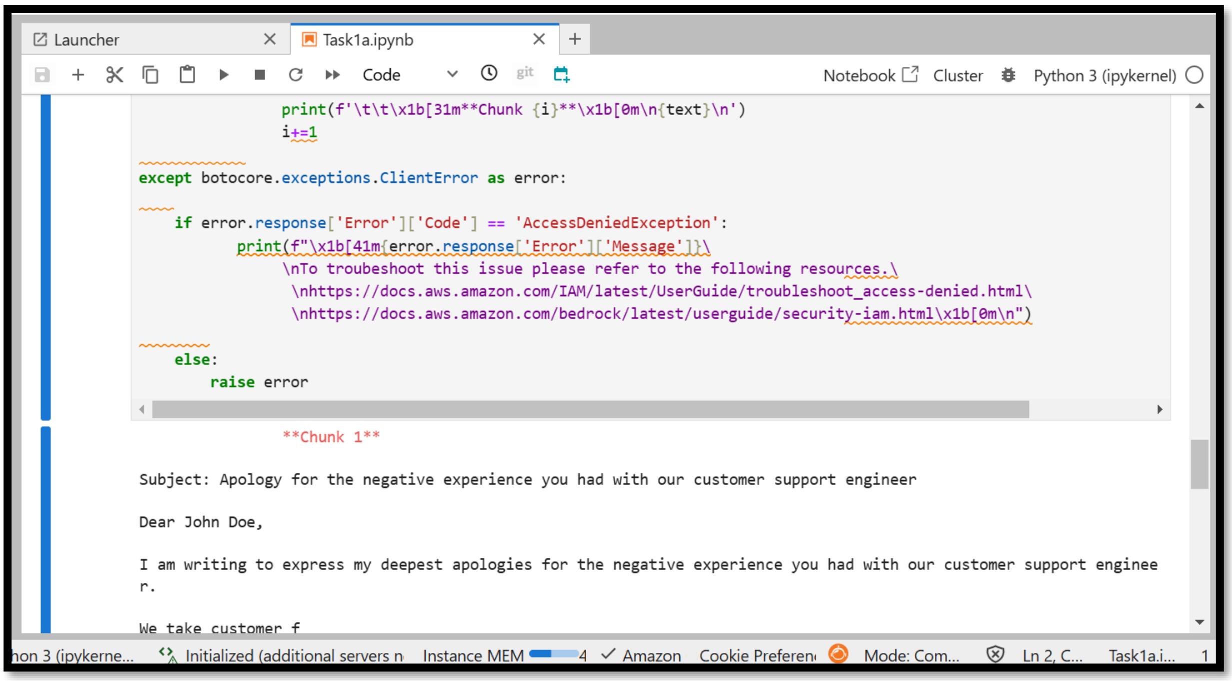
```
Subject: Apology for the negative experience you had with our customer support engineer

Dear John Doe,

I am writing to express my deepest apologies for the negative experience you had with our customer support enginee
r.

We take customer feedback very seriously and I want to assure you that we are taking steps to prevent this from hap
pening again in the future.

Sincerely,
Bob, Customer Service Manager, AnyCompany
```



Streaming Output Generation

Bedrock also supports that the output can be streamed as it is generated by the model in form of chunks. This email is generated by invoking the model with streaming option. `invoke_model_with_response_stream` returns a `ResponseStream` which you can read from.

[6]:
```
# invoke model with response stream
output = []
try:

    response = bedrock_client.invoke_model_with_response_stream(body=body, modelId=modelId, accept=accept, contentTy
    stream = response.get('body')

    i = 1
    if stream:
        for event in stream:
            chunk = event.get('chunk')
            if chunk:
                chunk_obj = json.loads(chunk.get('bytes').decode())
                text = chunk_obj['outputText']
                output.append(text)
```

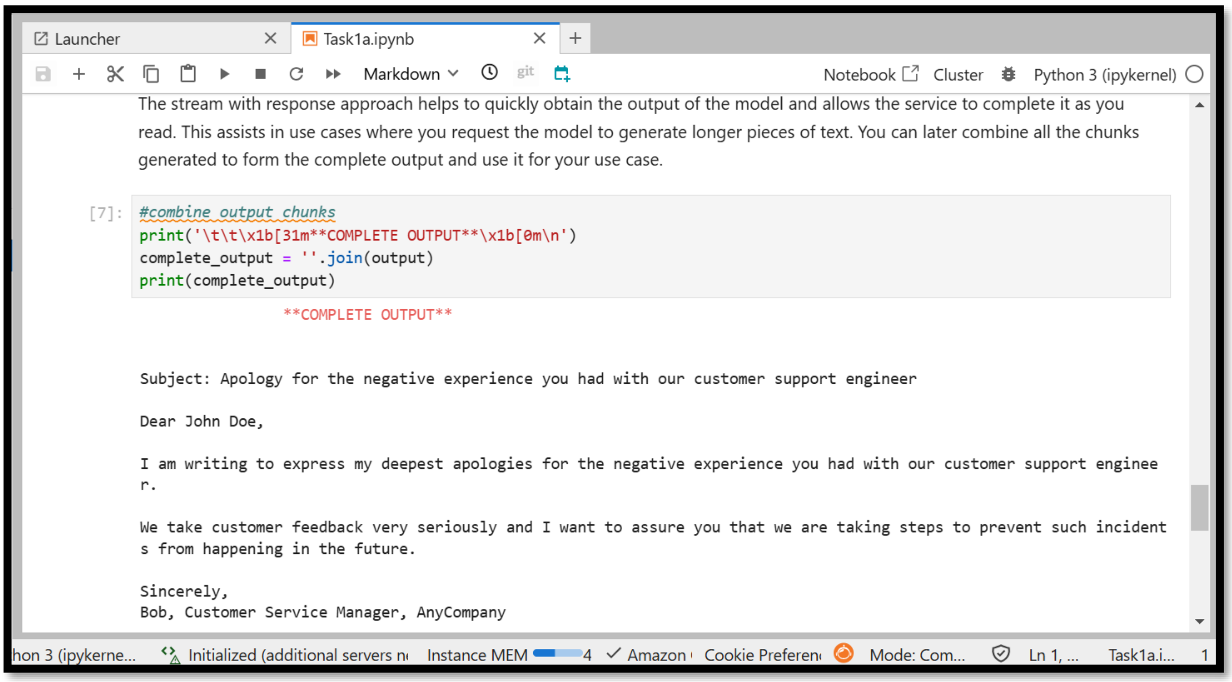


Launcher | Task1a.ipynb | Code | Notebook | Cluster | Python 3 (ipykernel)

```
            print(f'\t\t\x1b[31m**Chunk {i}**\x1b[0m\n{text}\n')
            i+=1

except botocore.exceptions.ClientError as error:

    if error.response['Error']['Code'] == 'AccessDeniedException':
           print(f"\x1b[41m{error.response['Error']['Message']}\
                \nTo troubeshoot this issue please refer to the following resources.\
                 \nhttps://docs.aws.amazon.com/IAM/latest/UserGuide/troubleshoot_access-denied.html\
                 \nhttps://docs.aws.amazon.com/bedrock/latest/userguide/security-iam.html\x1b[0m\n")

    else:
        raise error
```

```
                **Chunk 1**

Subject: Apology for the negative experience you had with our customer support engineer

Dear John Doe,

I am writing to express my deepest apologies for the negative experience you had with our customer support enginee
r.

We take customer f
```

hon 3 (ipykerne... Initialized (additional servers n Instance MEM 4 ✓ Amazon Cookie Preferen Mode: Com... Ln 2, C... Task1a.i... 1



Launcher | Task1a.ipynb | Markdown | Notebook | Cluster | Python 3 (ipykernel)

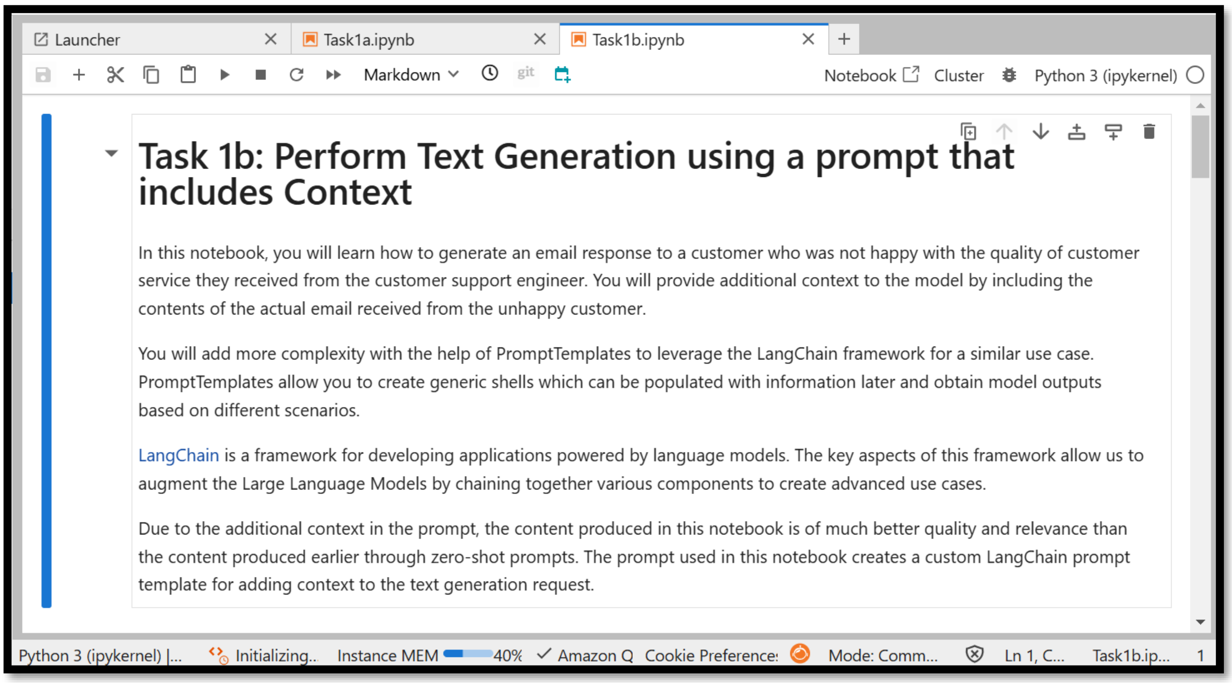
The stream with response approach helps to quickly obtain the output of the model and allows the service to complete it as you read. This assists in use cases where you request the model to generate longer pieces of text. You can later combine all the chunks generated to form the complete output and use it for your use case.

```
[7]: #combine output chunks
     print('\t\t\x1b[31m**COMPLETE OUTPUT**\x1b[0m\n')
     complete_output = ''.join(output)
     print(complete_output)
```

```
                **COMPLETE OUTPUT**


Subject: Apology for the negative experience you had with our customer support engineer

Dear John Doe,

I am writing to express my deepest apologies for the negative experience you had with our customer support enginee
r.

We take customer feedback very seriously and I want to assure you that we are taking steps to prevent such incident
s from happening in the future.

Sincerely,
Bob, Customer Service Manager, AnyCompany
```

hon 3 (ipykerne... Initialized (additional servers n Instance MEM 4 ✓ Amazon Cookie Preferen Mode: Com... Ln 1, ... Task1a.i... 1

Task1b.ipynb, which used the LangChain framework to communicate with the Amazon Bedrock API and created a custom LangChain prompt template to add context to the text generation request.



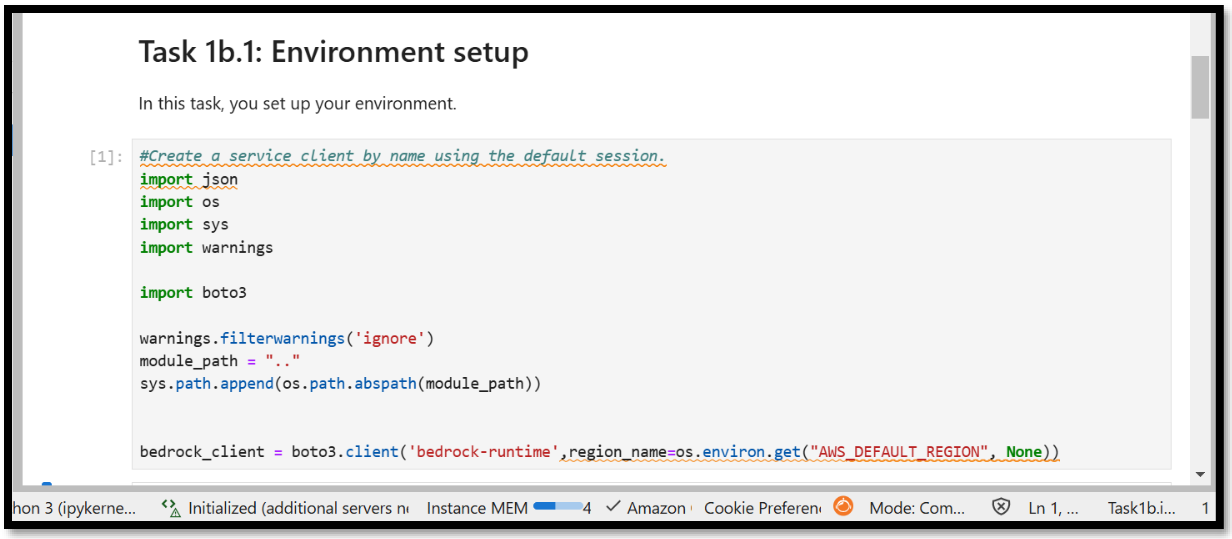
Task 1b: Perform Text Generation using a prompt that includes Context

In this notebook, you will learn how to generate an email response to a customer who was not happy with the quality of customer service they received from the customer support engineer. You will provide additional context to the model by including the contents of the actual email received from the unhappy customer.

You will add more complexity with the help of PromptTemplates to leverage the LangChain framework for a similar use case. PromptTemplates allow you to create generic shells which can be populated with information later and obtain model outputs based on different scenarios.

LangChain is a framework for developing applications powered by language models. The key aspects of this framework allow us to augment the Large Language Models by chaining together various components to create advanced use cases.

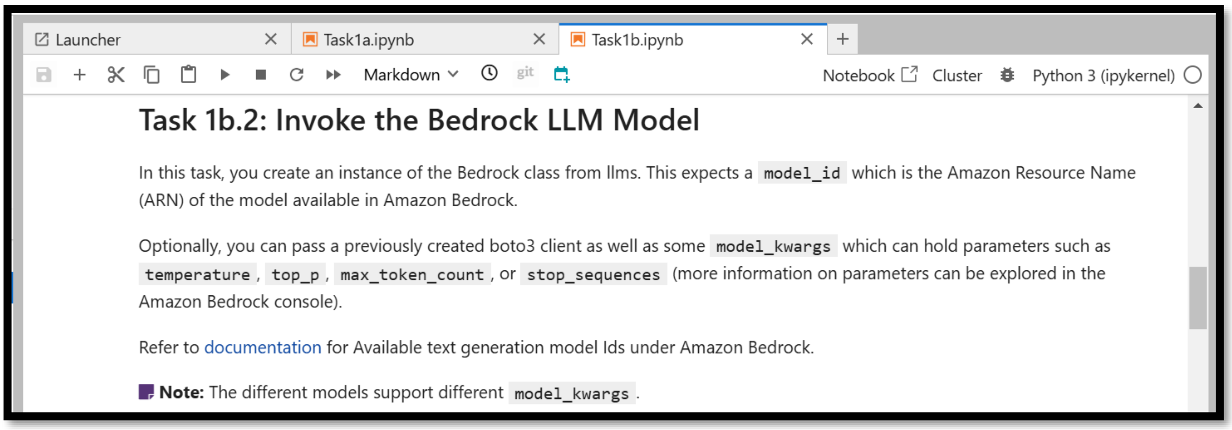
Due to the additional context in the prompt, the content produced in this notebook is of much better quality and relevance than the content produced earlier through zero-shot prompts. The prompt used in this notebook creates a custom LangChain prompt template for adding context to the text generation request.



Task 1b.1: Environment setup

In this task, you set up your environment.

```
[1]: #Create a service client by name using the default session.
import json
import os
import sys
import warnings

import boto3

warnings.filterwarnings('ignore')
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))
```



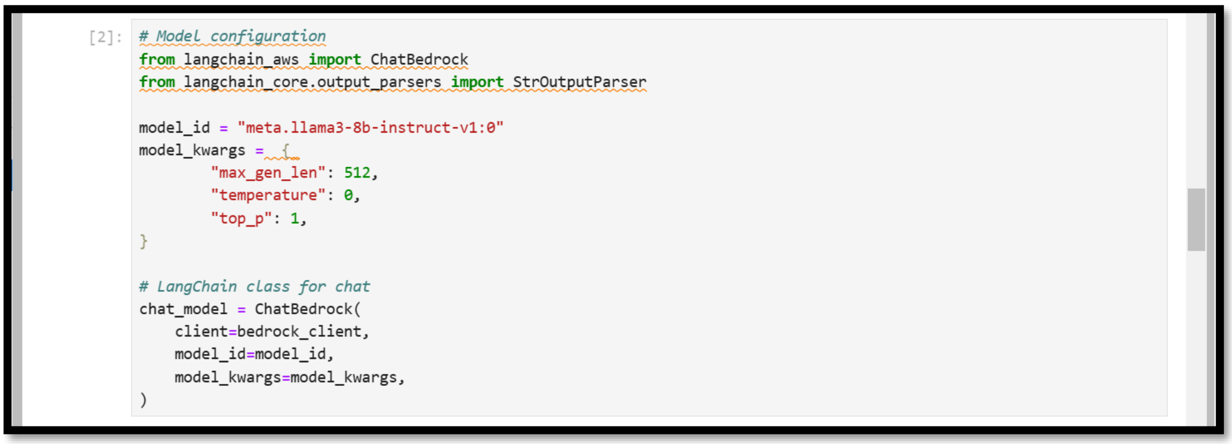
Task 1b.2: Invoke the Bedrock LLM Model

In this task, you create an instance of the Bedrock class from llms. This expects a `model_id` which is the Amazon Resource Name (ARN) of the model available in Amazon Bedrock.

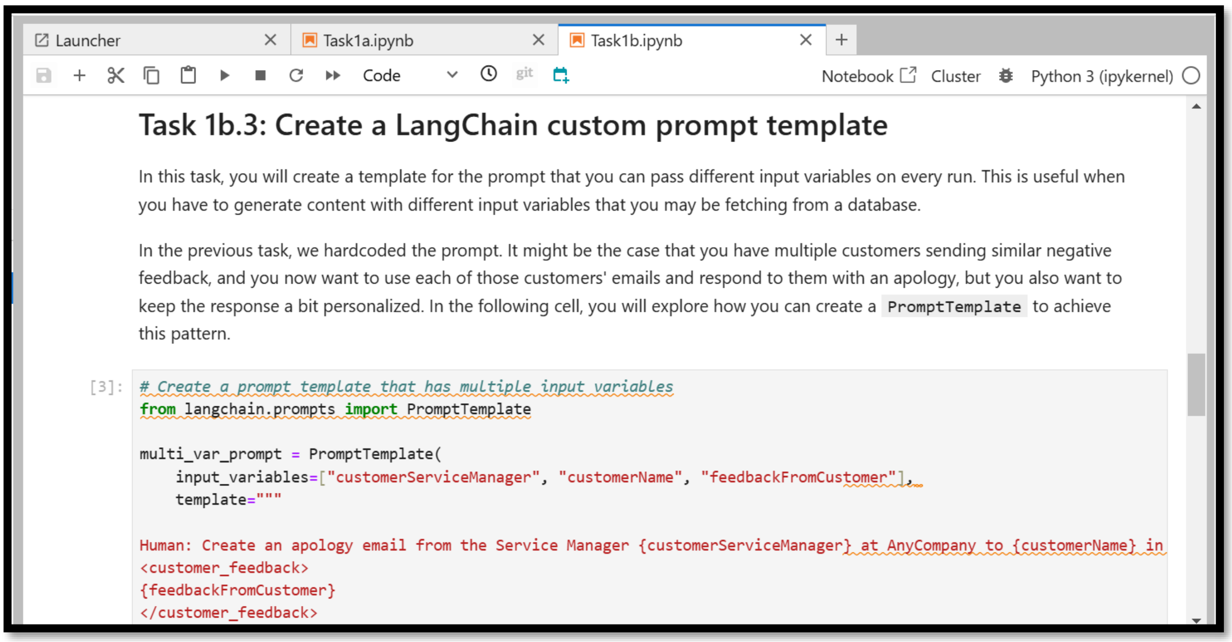
Optionally, you can pass a previously created boto3 client as well as some `model_kwargs` which can hold parameters such as `temperature`, `top_p`, `max_token_count`, or `stop_sequences` (more information on parameters can be explored in the Amazon Bedrock console).

Refer to documentation for Available text generation model Ids under Amazon Bedrock.

Note: The different models support different `model_kwargs`.



```
[2]: # Model configuration
from langchain_aws import ChatBedrock
from langchain_core.output_parsers import StrOutputParser

model_id = "meta.llama3-8b-instruct-v1:0"
model_kwargs =  {
        "max_gen_len": 512,
        "temperature": 0,
        "top_p": 1,
}

# LangChain class for chat
chat_model = ChatBedrock(
    client=bedrock_client,
    model_id=model_id,
    model_kwargs=model_kwargs,
)
```

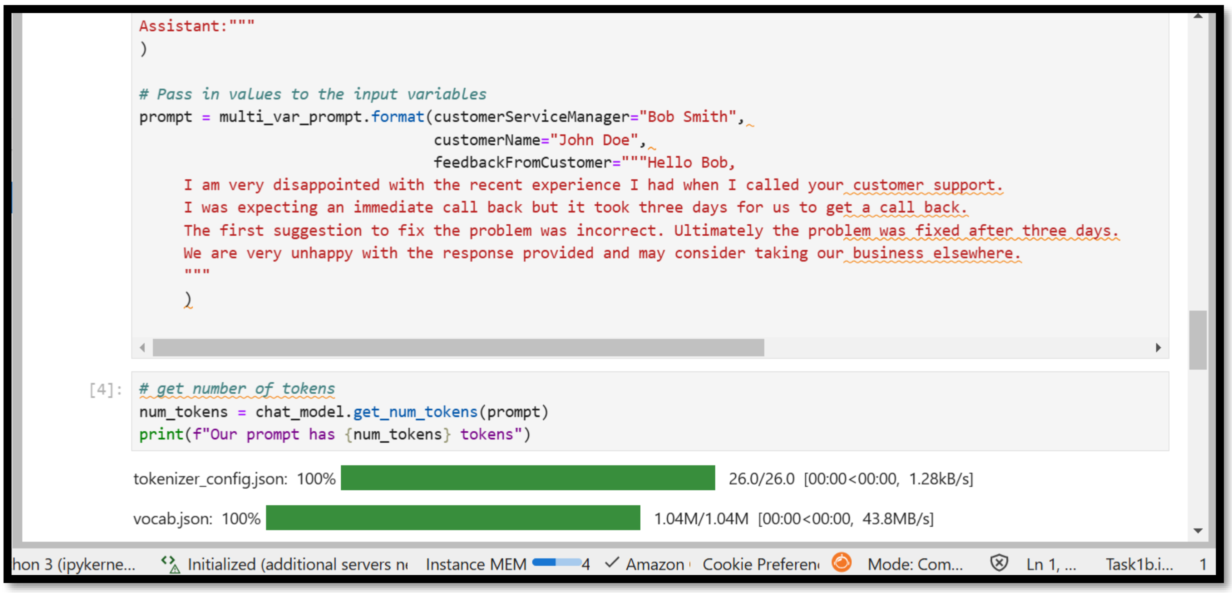


Task 1b.3: Create a LangChain custom prompt template

In this task, you will create a template for the prompt that you can pass different input variables on every run. This is useful when you have to generate content with different input variables that you may be fetching from a database.

In the previous task, we hardcoded the prompt. It might be the case that you have multiple customers sending similar negative feedback, and you now want to use each of those customers' emails and respond to them with an apology, but you also want to keep the response a bit personalized. In the following cell, you will explore how you can create a `PromptTemplate` to achieve this pattern.

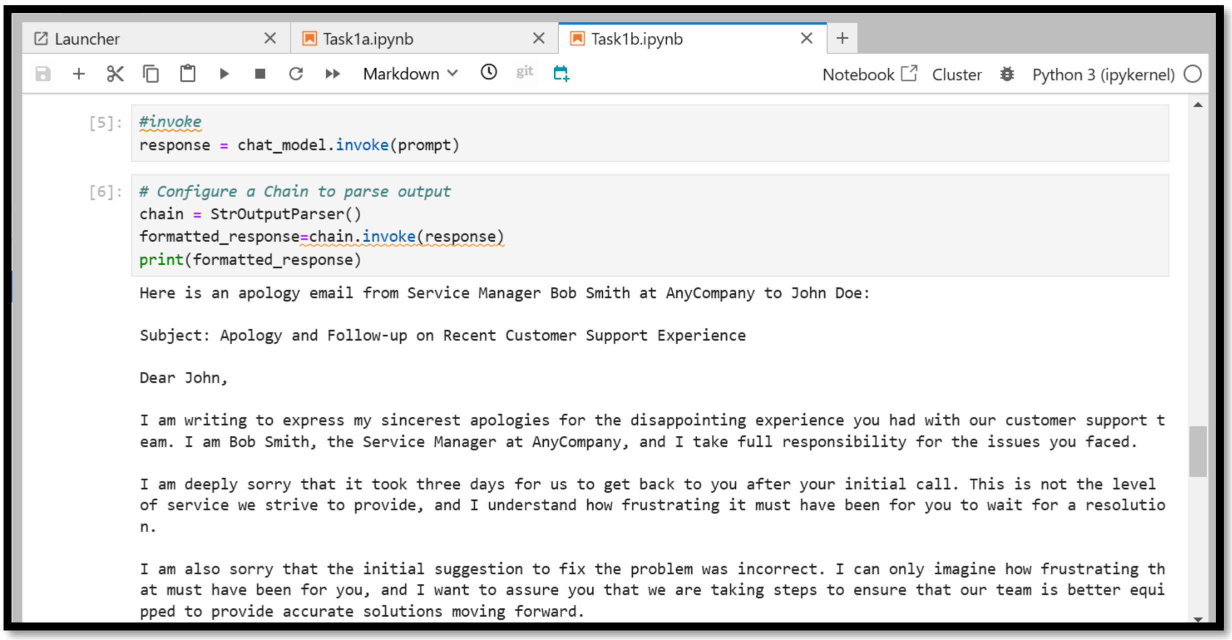
```
[3]: # Create a prompt template that has multiple input variables
from langchain.prompts import PromptTemplate

multi_var_prompt = PromptTemplate(
    input_variables=["customerServiceManager", "customerName", "feedbackFromCustomer"],
    template="""

Human: Create an apology email from the Service Manager {customerServiceManager} at AnyCompany to {customerName} in
<customer_feedback>
{feedbackFromCustomer}
</customer_feedback>
```



```
Assistant:"""
)

# Pass in values to the input variables
prompt = multi_var_prompt.format(customerServiceManager="Bob Smith",
                                 customerName="John Doe",
                                 feedbackFromCustomer="""Hello Bob,
    I am very disappointed with the recent experience I had when I called your customer support.
    I was expecting an immediate call back but it took three days for us to get a call back.
    The first suggestion to fix the problem was incorrect. Ultimately the problem was fixed after three days.
    We are very unhappy with the response provided and may consider taking our business elsewhere.
    """
    )
```

```
[4]: # get number of tokens
num_tokens = chat_model.get_num_tokens(prompt)
print(f"Our prompt has {num_tokens} tokens")
```

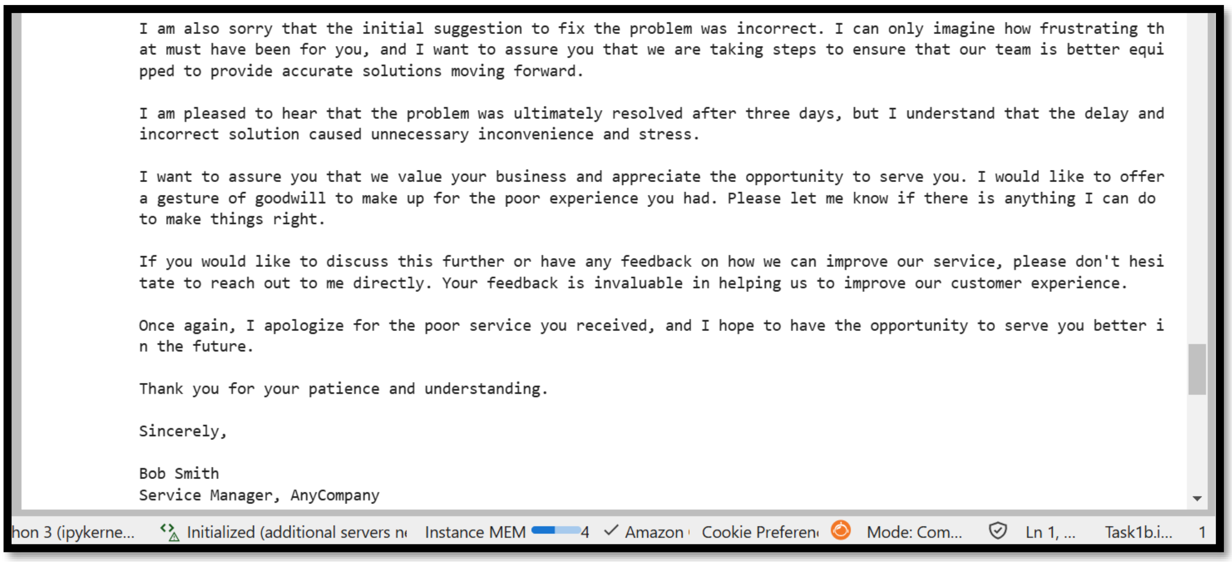
```
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 1.28kB/s]

vocab.json: 100%  1.04M/1.04M [00:00<00:00, 43.8MB/s]
```



```
[5]: #invoke
response = chat_model.invoke(prompt)
```

```
[6]: # Configure a Chain to parse output
chain = StrOutputParser()
formatted_response=chain.invoke(response)
print(formatted_response)
```

```
Here is an apology email from Service Manager Bob Smith at AnyCompany to John Doe:

Subject: Apology and Follow-up on Recent Customer Support Experience

Dear John,

I am writing to express my sincerest apologies for the disappointing experience you had with our customer support t
eam. I am Bob Smith, the Service Manager at AnyCompany, and I take full responsibility for the issues you faced.

I am deeply sorry that it took three days for us to get back to you after your initial call. This is not the level
of service we strive to provide, and I understand how frustrating it must have been for you to wait for a resolutio
n.

I am also sorry that the initial suggestion to fix the problem was incorrect. I can only imagine how frustrating th
at must have been for you, and I want to assure you that we are taking steps to ensure that our team is better equi
pped to provide accurate solutions moving forward.
```



```
I am also sorry that the initial suggestion to fix the problem was incorrect. I can only imagine how frustrating th
at must have been for you, and I want to assure you that we are taking steps to ensure that our team is better equi
pped to provide accurate solutions moving forward.

I am pleased to hear that the problem was ultimately resolved after three days, but I understand that the delay and
incorrect solution caused unnecessary inconvenience and stress.

I want to assure you that we value your business and appreciate the opportunity to serve you. I would like to offer
a gesture of goodwill to make up for the poor experience you had. Please let me know if there is anything I can do
to make things right.

If you would like to discuss this further or have any feedback on how we can improve our service, please don't hesi
tate to reach out to me directly. Your feedback is invaluable in helping us to improve our customer experience.

Once again, I apologize for the poor service you received, and I hope to have the opportunity to serve you better i
n the future.

Thank you for your patience and understanding.

Sincerely,

Bob Smith
Service Manager, AnyCompany
```