

Hands-on tutorials

Extract information from unstructured documents with Amazon Bedrock and Amazon Textract



Extract information from unstructured documents with Amazon Bedrock and Amazon Textract: Hands-on tutorials

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Extract information from unstructured documents with Amazon Bedrock and Amazon Textract i

Overview	1
What you will accomplish	1
Prerequisites	2
Implementation	2
Congratulations	23

Extract information from unstructured documents with Amazon Bedrock and Amazon Textract

AWS experience	Beginner
Time to complete	20 minutes
Cost to complete	Less than \$0.15 if completed within two hours and the notebook is deleted at the end of the tutorial.
Get help	Troubleshooting Amazon Bedrock models Debugging training issues
Last update	November 14, 2024

Overview

In this tutorial, you will learn how to utilize Amazon Bedrock and Amazon Textract to extract and process information from unstructured documents.

Amazon Bedrock is a fully managed service that offers a choice of high-performing foundation models (FMs) from leading AI companies like AI21 Labs, Anthropic, Cohere, Meta, Mistral AI, Stability AI, and Amazon through a single API, along with a broad set of capabilities you need to build generative AI applications with security, privacy, and responsible AI.

Amazon Textract is a machine learning (ML) service that automatically extracts text, handwriting, layout elements, and data from scanned documents.

What you will accomplish

In this tutorial, you will:

- Enable access to a foundation model on your AWS account
- Create a new Jupyter notebook to write test code and run tests

- Generate code
- Clean up your resources

Prerequisites

Before starting this tutorial, you will need:

- An AWS account: if you don't already have one follow the [Setup Your Environment](#) tutorial.

Implementation

Step 1: Enable Anthropic FM

In this step, you will enable the use of Anthropic models on your AWS account.

Note

Already requested and obtained access to Anthropic models on Amazon Bedrock? Skip to [Step 2: Create a Jupyter Notebook](#).

1. Open Amazon Bedrock

Sign in to the AWS Management console, and **open** the Amazon Bedrock console at <https://console.aws.amazon.com/bedrock/home>.

In the left navigation pane, under **Bedrock configurations**, choose **Model Access**.

Amazon Bedrock <

- ▼ **Getting started**
 - Overview
 - Examples
 - Providers
- ▼ **Foundation models**
 - Base models
 - Custom models
 - Imported models
- ▼ **Playgrounds**
 - Chat/text
 - Image
- ▼ **Builder tools**
 - Prompt management
 - Knowledge bases
 - Agents
 - Prompt flows [Preview](#)
- ▼ **Safeguards**
 - Guardrails
 - Watermark detection
- ▼ **Inference**
 - Provisioned Throughput
 - Batch inference [New](#)
 - Cross-region inference [New](#)
- ▼ **Assessment**
 - Model Evaluation

[User guide](#)

[Bedrock Service Terms](#)

- ▼ **Bedrock configurations**
 - Model access
 - Bedrock Studio [Preview](#)
 - Settings

Amazon Bedrock

Overview [Info](#)

[Explore & Learn](#) | [Build & Test](#)

Foundation models

Amazon Bedrock supports foundation models from industry-leading providers. Choose the model that is best suited to achieving your unique goals.

Jamba 1.5
By AI21 Labs

Titan
By Amazon

Claude
By Anthropic

Command
By Cohere

Llama
By Meta

Mistral
By Mistral AI

Stable Diffusion
By Stability AI

Playgrounds

Chat/text

Generate text for a vast range of language processing tasks with various pre-trained models. You can use a single prompt or iterate on the result by submitting subsequent prompts that take into account the context of previous prompts and generated responses in a chat format.

[Open chat/text playground](#)

Image

Easily generate compelling images by providing text prompts to pre-trained models. In the playground, enter a text prompt to get started.

[Open image playground](#)

2. Enable a model

On the **What is Model access?** page, choose **Enable specific models**.

Amazon Bedrock > Model access

What is Model access?

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

Visit [Amazon Bedrock Quotas](#) for a quick guide to the default quotas and limits that apply to Amazon Bedrock.

3. Choose the Anthropic models

On the **Edit model access** page, select the **Anthropic models**, and choose **Next**.

- Step 1 Edit model access
- Step 2 Review and submit

Edit model access

Base models (8/40)

Not seeing a model you're interested in? Check out all supported models by region [here](#).

Group by provider ▾

<input type="checkbox"/>	Models	Access status	Modality	EULA
<input type="checkbox"/>	▶ AI21 Labs (5)	0/5 access granted		
<input type="checkbox"/>	▶ Amazon (8)	0/8 access granted		
<input checked="" type="checkbox"/>	▼ Anthropic (8)	0/8 access granted		
<input checked="" type="checkbox"/>	Claude 3.5 Haiku Cross-region inference	Available to request	Text	EULA
<input checked="" type="checkbox"/>	Claude 3.5 Sonnet v2 Cross-region inference	Available to request	Text & Vision	EULA
<input checked="" type="checkbox"/>	Claude 3.5 Sonnet	Available to request	Text & Vision	EULA
<input checked="" type="checkbox"/>	Claude 3 Opus Cross-region inference	Available to request	Text & Vision	EULA
<input checked="" type="checkbox"/>	Claude 3 Sonnet	Available to request	Text & Vision	EULA
<input checked="" type="checkbox"/>	Claude 3 Haiku	Available to request	Text & Vision	EULA
<input checked="" type="checkbox"/>	Claude	Available to request	Text	EULA
<input checked="" type="checkbox"/>	Claude Instant	Available to request	Text	EULA
<input type="checkbox"/>	▶ Cohere (6)	0/6 access granted		
<input type="checkbox"/>	▶ Meta (8)	0/8 access granted		
<input type="checkbox"/>	▶ Mistral AI (4)	0/4 access granted		
<input type="checkbox"/>	▶ Stability AI (1)	0/1 access granted		

4. Review and submit the change

On the **Review and submit** page, review your selections, and choose **Submit**.

Review and submit

Step 1: Edit model access

[Edit](#)

Model access modifications (8)

Models	Modifications
Claude 3.5 Haiku	Request access
Claude 3.5 Sonnet v2	Request access
Claude 3.5 Sonnet	Request access
Claude 3 Opus	Request access
Claude 3 Sonnet	Request access
Claude 3 Haiku	Request access
Claude	Request access
Claude Instant	Request access

i 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](#)

Step 2: Create a Jupyter Notebook

In this step, you will create a Jupyter notebook to write your Proof-of-Concept code and test it out with real documents.

1. Open Amazon SageMaker AI

Open the Amazon Sagemaker console at <https://console.aws.amazon.com/sagemaker/home>.

In the left navigation pane, under **Applications and IDEs**, choose **Notebooks**.



Amazon SageMaker X

Getting started

▼ Applications and IDEs

- Studio
- Canvas
- RStudio
- TensorBoard
- Profiler
- Notebooks**

MACHINE LEARNING

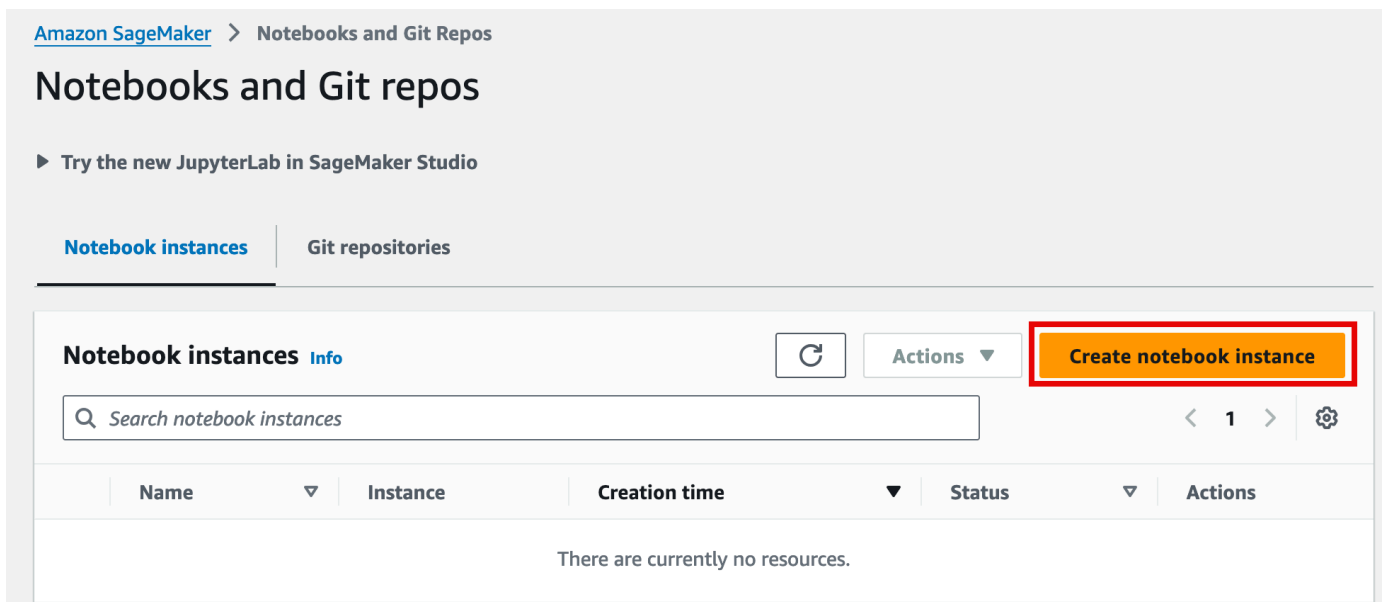
Amazon SageMaker

Build, train, and deploy machine learning models at scale

The quickest and easiest way to get ML models from idea to production.

2. Create a notebook instance

On the **Notebooks and Git repos** page, choose **Create notebook instance**.



Amazon SageMaker > Notebooks and Git Repos

Notebooks and Git repos

► Try the new JupyterLab in SageMaker Studio

Notebook instances | Git repositories

Notebook instances Info Refresh Actions **Create notebook instance**

Search notebook instances < 1 > Settings

Name	Instance	Creation time	Status	Actions
There are currently no resources.				

3. Configure notebook instance settings

On the **Create notebook instance** page:

In the **Notebook instance settings** section:

- For **Notebook instance name**, enter a **name** for your Jupyter instance.
- For **Notebook instance type**, verify **ml.t3.medium** is selected.

- **Keep all other default settings.**

Create notebook instance

Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. [Learn more](#)

Notebook instance settings

Notebook instance name

document-processing

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type

ml.t3.medium

Platform identifier [Learn more](#)

Amazon Linux 2, Jupyter Lab 3

► **Additional configuration**

Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the [AmazonSageMakerFullAccess](#) IAM policy attached.

AmazonSageMaker-ExecutionRole-20241113T113547



Success! You created an IAM role.

[AmazonSageMaker-ExecutionRole-20241113T113547](#)



[Create role using the role creation wizard](#)

Root access - *optional*

- Enable - Give users root access to the notebook
- Disable - Don't give users root access to the notebook
Lifecycle configurations always have root access

Encryption key - *optional*

Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.

No Custom Encryption

► **Network - optional**

4. Configure permissions and encryption

In the **Permissions and encryption** section:

- For **IAM role**, choose **Create a new role**.
- On the **Create an IAM role** pop up window, for **S3 buckets you specify – optional**, choose **None**, and then choose **Create role**.

Then, choose **Create notebook instance**.

ook instance type

Create an IAM role

Passing an IAM role gives Amazon SageMaker permission to perform actions in other AWS services on your behalf. Creating a role here will grant permissions described by the [AmazonSageMakerFullAccess](#) IAM policy to the role you create.

The IAM role you create will provide access to:

- S3 buckets you specify - *optional*
 - Any S3 bucket
Allow users that have access to your notebook instance access to any bucket and its contents in your account.
 - Specific S3 buckets

Comma delimited. ARNs, "*" and "/" are not supported.
- None

- Any S3 bucket with "sagemaker" in the name
- Any S3 object with "sagemaker" in the name
- Any S3 object with the tag "sagemaker" and value "true" [See Object tagging](#)
- S3 bucket with a Bucket Policy allowing access to SageMaker [See S3 bucket policies](#)

Step 3: Generate code to process your documents

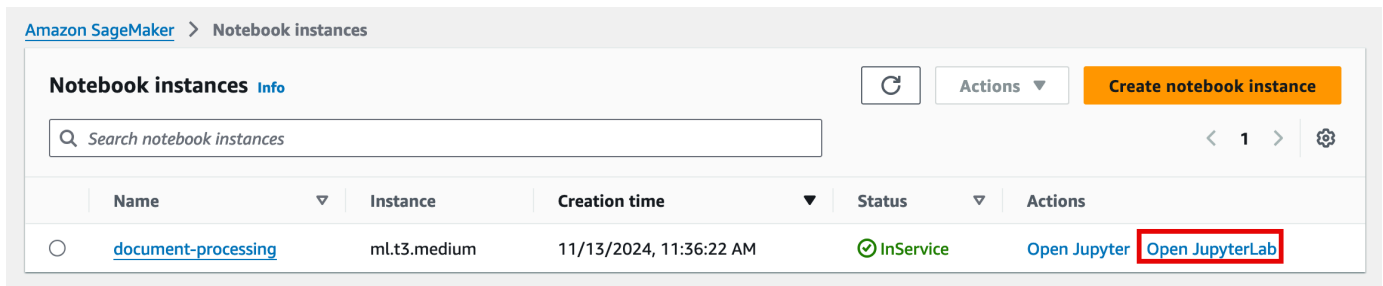
In this step, you will use Bedrock playground to generate code for your Jupyter notebook.

1. Open JupyterLab

On the **Notebook instance** page, choose **Open JupyterLab** for the instance you created in the previous step.

Note

The notebook will open in a separate browser tab.



Amazon SageMaker > Notebook instances

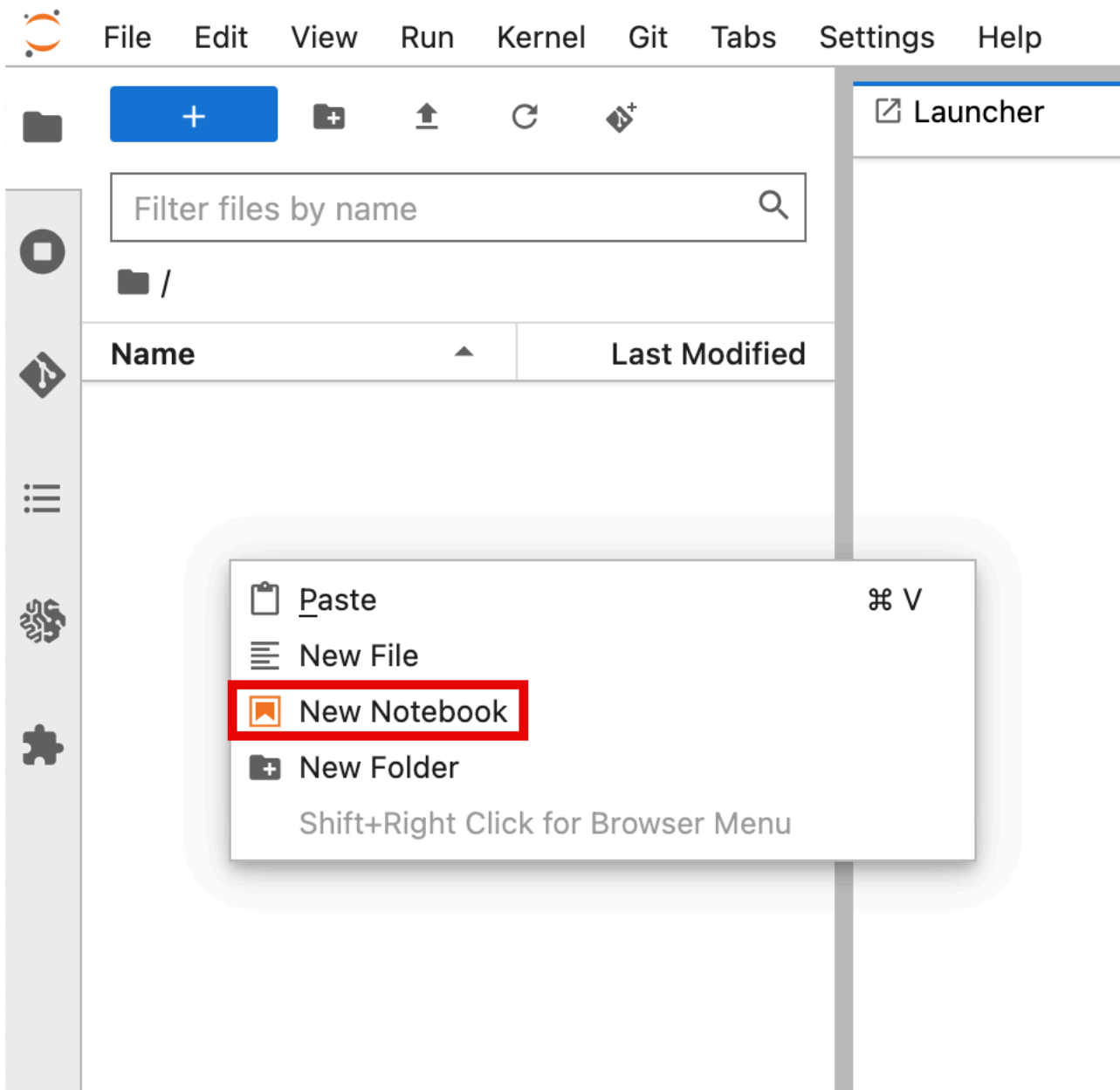
Notebook instances [Info](#) Refresh Actions Create notebook instance

Search notebook instances

Name	Instance	Creation time	Status	Actions
document-processing	ml.t3.medium	11/13/2024, 11:36:22 AM	InService	Open Jupyter Open JupyterLab

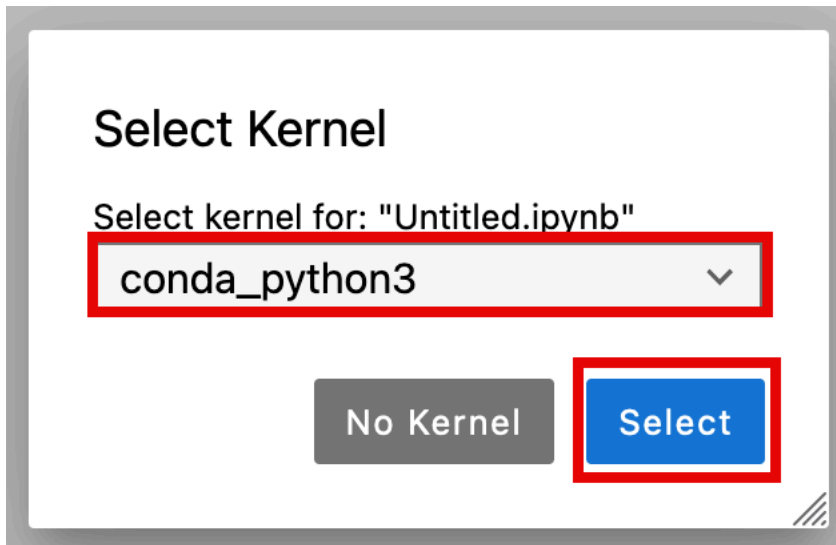
2. Create a new notebook

On the **JupyterLab** tab, right-click the file area, and then select **New Notebook**.



3. Select kernel

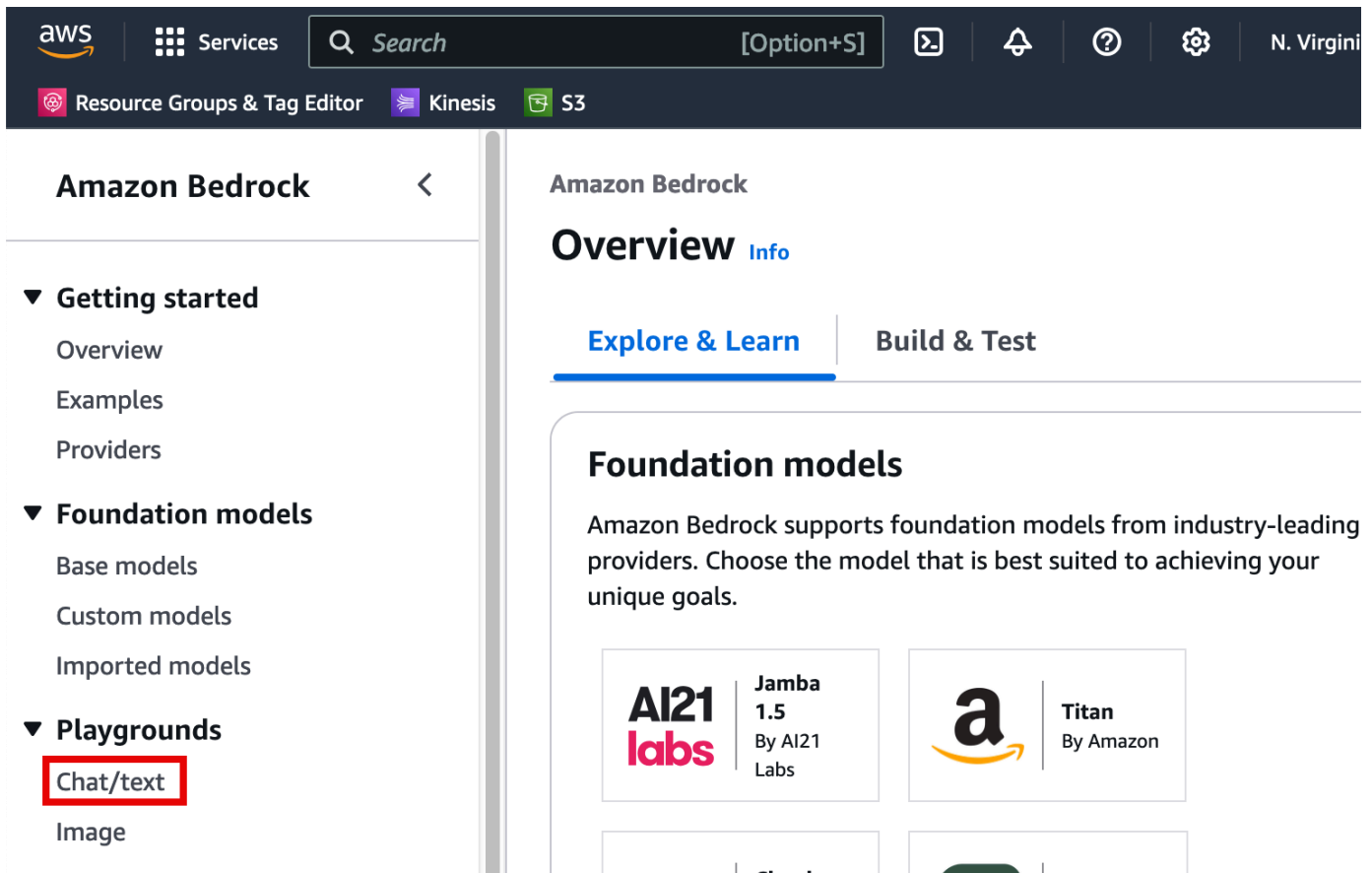
On the **Select Kernel** pop up window, choose **conda_python3**, and choose **Select**.



4. Open the chat playground

In a new tab, **open** the Amazon Bedrock console at <https://console.aws.amazon.com/bedrock/home>.

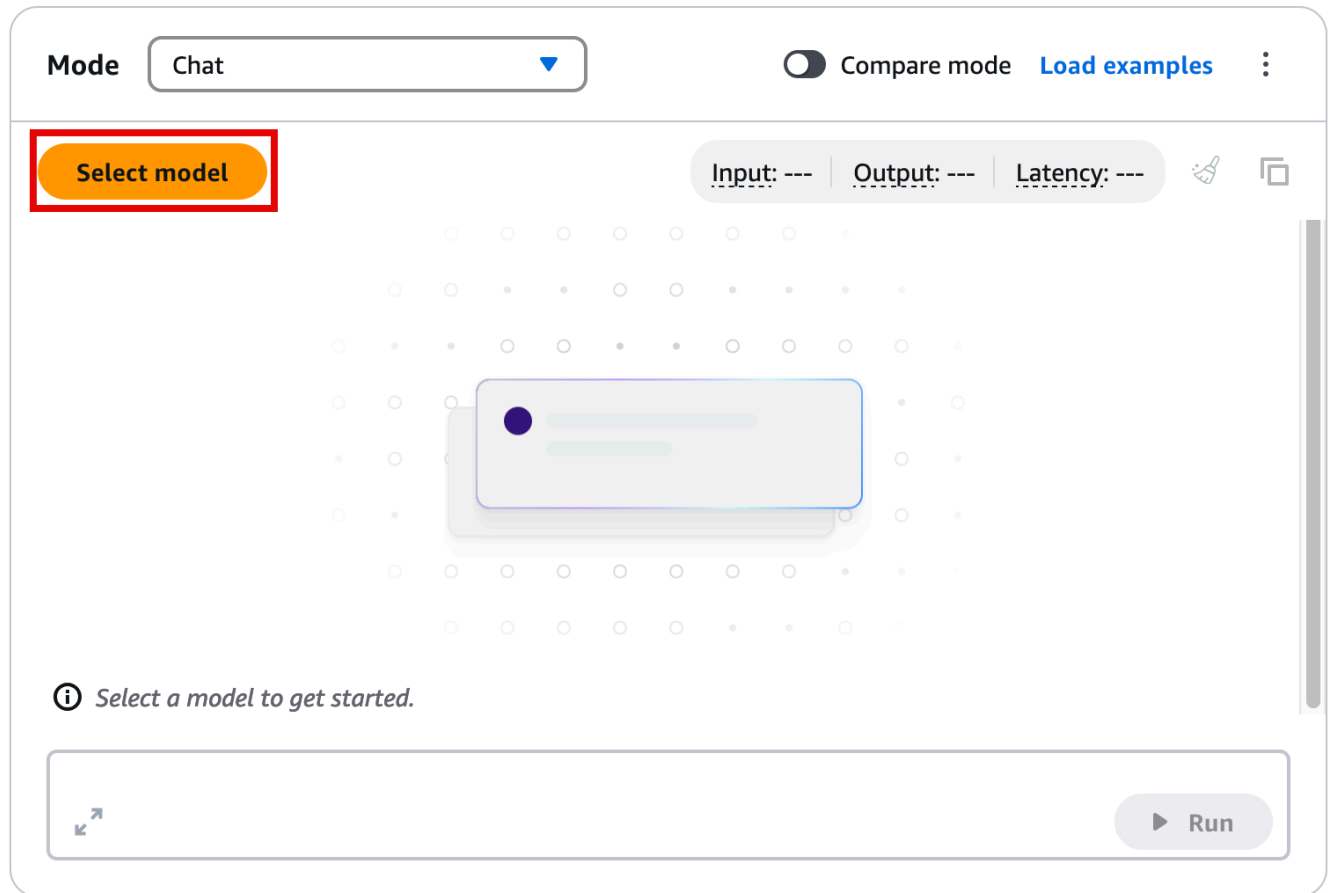
In the left navigation pane, under **Playgrounds**, choose **Chat/text**.



5. Select the model

On the **Mode** page, choose **Select model**.

[Amazon Bedrock](#) > Chat/text playground



6. Specify the model details

In the **Select model** dialog box:

- For **Categories**, choose **Anthropic**.
- For **Models with access**, choose the **Claude 3.5 Sonnet** model.
- Then, choose **Apply**.

Note

The Claude 3.5 Sonnet is the most intelligent model from Anthropic. You can see a more detailed model comparison [here](#).

Select model

1. Categories

Model providers

- AI21 Labs
- Amazon
- Anthropic**
- Cohere
- Meta
- Mistral AI

2. Models

Text model | Max 100k tokens

- Claude 3 Sonnet v1**
Text & vision model | Max 200k tokens
- Claude 3 Haiku v1**
Text & vision model | Max 200k tokens
- Claude 3 Opus v1**
Text & vision model | Max 200k tokens
- Claude 3.5 Sonnet v1**
Text & vision model | Max 200k tokens
- Claude 3.5 Sonnet v2 v2**
Text & vision model | Max 200k tokens
- Claude 3.5 Haiku v1**
Text model | Max 200k tokens

Not seeing a model you are interested in? Check out all supported models [here](#)

3. Inference

- On-demand**
- US Anthropic Claude 3.5 Sonnet

Cancel **Apply**

7. Generate code

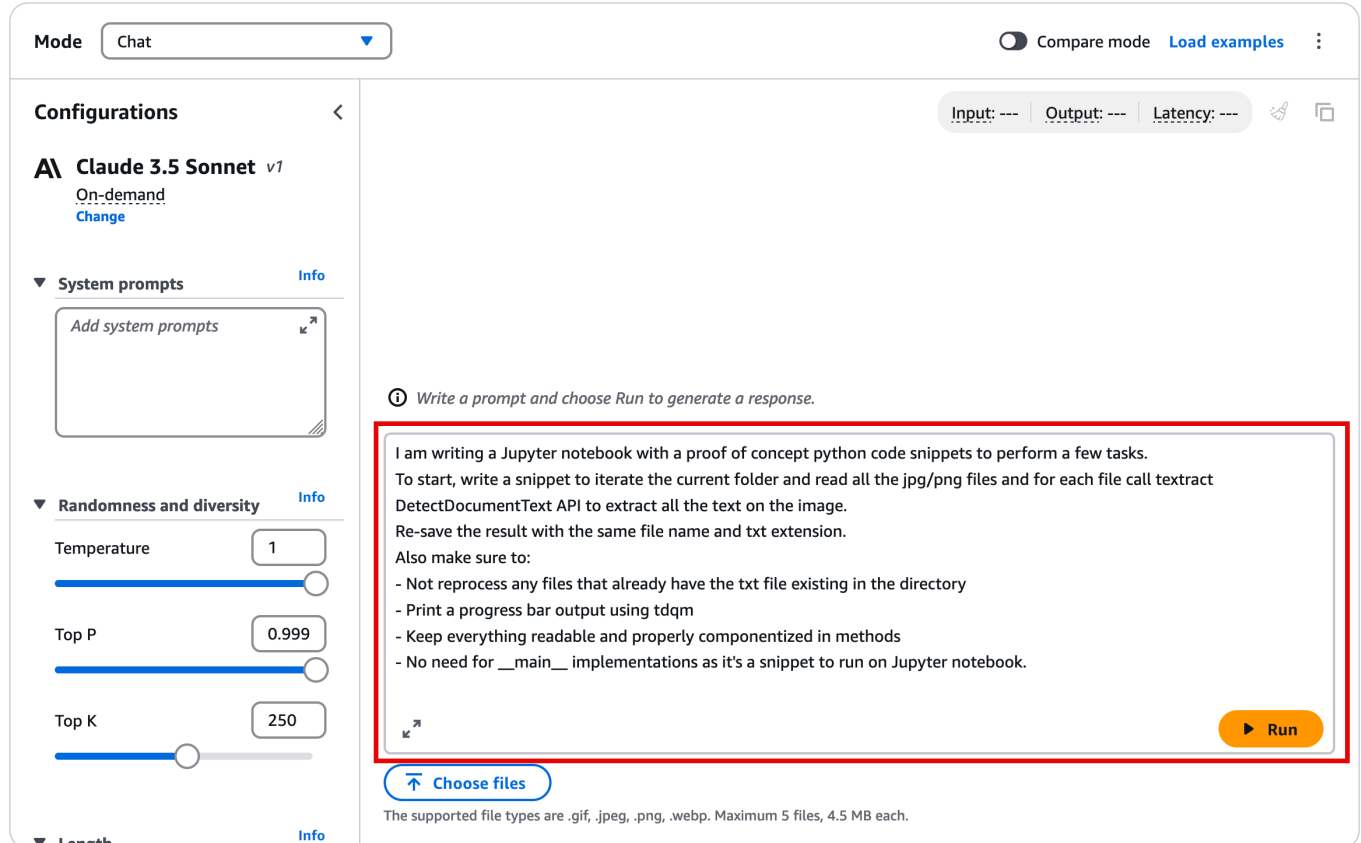
In the **Chat playground**, you can now ask the LLM to write sample code. The following is an example prompt that you can use to extract information from an unstructured document.

```
I am writing a Jupyter notebook with a proof of concept python code snippets to perform a few tasks.
To start, write a snippet to iterate the current folder and read all the jpg/png files and for each file call textract DetectDocumentText API to extract all the text on the image.
Re-save the result with the same file name and txt extension.
Also make sure to:
- Not reprocess any files that already have the txt file existing in the directory
```

- Print a progress bar output using `tdqm`
- Keep everything readable and properly componentized in methods
- No need for `__main__` implementations as it's a snippet to run on Jupyter notebook.

Once you enter your prompt, and choose **Run**, the prompt response will include code and also explanation of everything that the model generated. The code will typically be enclosed in quotation marks.

Amazon Bedrock > Chat/text playground



Mode: Chat Compare mode Load examples

Configurations

AI Claude 3.5 Sonnet v1
On-demand
[Change](#)

System prompts Info

Add system prompts

Randomness and diversity Info

Temperature: 1

Top P: 0.999

Top K: 250

Length Info

Write a prompt and choose Run to generate a response.

I am writing a Jupyter notebook with a proof of concept python code snippets to perform a few tasks. To start, write a snippet to iterate the current folder and read all the jpg/png files and for each file call `extractDocumentText` API to extract all the text on the image. Re-save the result with the same file name and txt extension. Also make sure to:

- Not reprocess any files that already have the txt file existing in the directory
- Print a progress bar output using `tdqm`
- Keep everything readable and properly componentized in methods
- No need for `__main__` implementations as it's a snippet to run on Jupyter notebook.

[Choose files](#)

The supported file types are .gif, .jpeg, .png, .webp. Maximum 5 files, 4.5 MB each.

Run

8. Check the output

The generated code with the example prompt should look similar to the following example. You can also use the **copy** function to **paste** the code directly into the Jupyter notebook.

```
import os
import boto3
from tqdm import tqdm
from PIL import Image

def get_image_files(directory):
```

```
"""Get all jpg and png files in the given directory."""
return [f for f in os.listdir(directory) if f.lower().endswith(('.jpg',
'.png'))]

def should_process_file(file_path):
    """Check if the file should be processed (i.e., no corresponding txt file
exists)."""
    txt_path = os.path.splitext(file_path)[0] + '.txt'
    return not os.path.exists(txt_path)

def extract_text_from_image(image_path):
    """Extract text from the image using Amazon Textract."""
    client = boto3.client('textract')

    with open(image_path, 'rb') as image:
        response = client.detect_document_text(Document={'Bytes': image.read()})

    extracted_text = []
    for item in response['Blocks']:
        if item['BlockType'] == 'LINE':
            extracted_text.append(item['Text'])

    return '\n'.join(extracted_text)

def save_text_to_file(text, file_path):
    """Save the extracted text to a file."""
    txt_path = os.path.splitext(file_path)[0] + '.txt'
    with open(txt_path, 'w', encoding='utf-8') as f:
        f.write(text)

def process_images_in_directory(directory):
    """Process all images in the given directory."""
    image_files = get_image_files(directory)

    for image_file in tqdm(image_files, desc="Processing images"):
        image_path = os.path.join(directory, image_file)

        if should_process_file(image_path):
            extracted_text = extract_text_from_image(image_path)
            save_text_to_file(extracted_text, image_path)

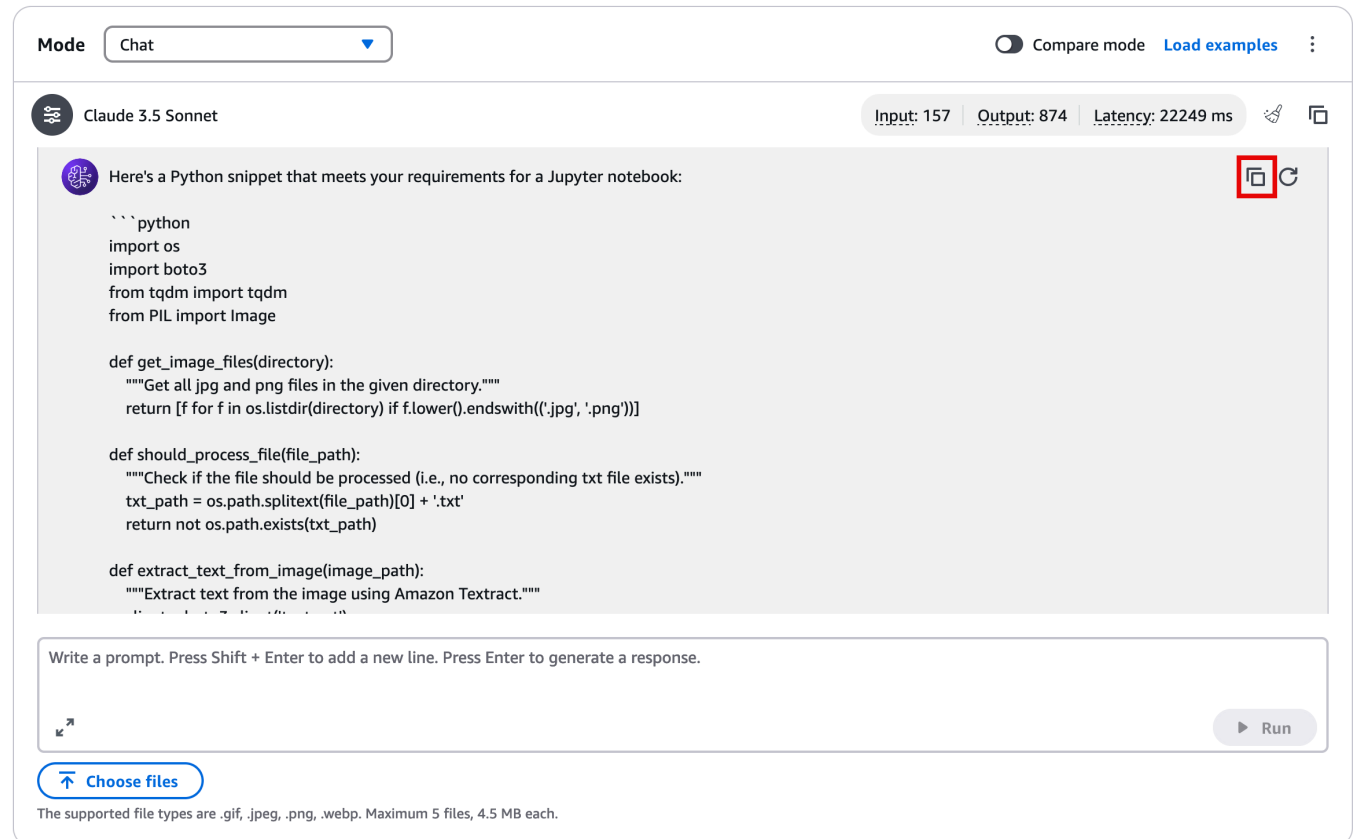
# Usage in Jupyter notebook
directory = '.' # Current directory
```

```
process_images_in_directory(directory)
```

Note

The previous example code is built to process all files on the current directory and needs an image in order to fully process the code.

Amazon Bedrock > Chat/text playground



The screenshot shows the Amazon Bedrock Chat/text playground interface. At the top, the mode is set to "Chat". The model being used is "Claude 3.5 Sonnet". The input and output statistics are displayed as "Input: 157", "Output: 874", and "Latency: 22249 ms". The main content area displays a Python code snippet for processing images in a directory. The code includes imports for os, boto3, tqdm, and PIL, and defines three functions: get_image_files, should_process_file, and extract_text_from_image. The code is highlighted in a light gray box. Below the code, there is a text input field with a placeholder "Write a prompt. Press Shift + Enter to add a new line. Press Enter to generate a response." and a "Run" button. At the bottom, there is a "Choose files" button and a note about supported file types: ".gif, .jpeg, .png, .webp. Maximum 5 files, 4.5 MB each."

```
```\npython\nimport os\nimport boto3\nfrom tqdm import tqdm\nfrom PIL import Image\n\ndef get_image_files(directory):\n    """Get all jpg and png files in the given directory."""\n    return [f for f in os.listdir(directory) if f.lower().endswith(('.jpg', '.png'))]\n\ndef should_process_file(file_path):\n    """Check if the file should be processed (i.e., no corresponding txt file exists)."""\n    txt_path = os.path.splitext(file_path)[0] + '.txt'\n    return not os.path.exists(txt_path)\n\ndef extract_text_from_image(image_path):\n    """Extract text from the image using Amazon Textract."""\n    # Implementation details for image text extraction\n    # ...
```

## 9. Prepare your image file

You can use your own image or download and save this [image](#). Then, **find** the image you want to use on your local machine, and **drag** the file to the Jupyter Notebook file explorer in order to copy and paste it.

The screenshot shows a JupyterLab environment with a file browser on the left and a code editor on the right. The code editor contains the following Python code:

```
[3]: import os
import boto3
from tqdm import tqdm
from PIL import Image

def get_image_files(directory):
 """Get all jpg and png files in the given directory."""
 return [f for f in os.listdir(directory) if f.lower().endswith(('.jpg', '.png'))]

def should_process_file(file_path):
 """Check if the file should be processed (i.e., no corresponding txt file exists)."""
 txt_path = os.path.splitext(file_path)[0] + '.txt'
 return not os.path.exists(txt_path)

def extract_text_from_image(image_path):
 """Extract text from the image using Amazon Textract"""
 client = boto3.client('textract')

 with open(image_path, 'rb') as image:
 response = client.detect_document_text(ImageBytes=image.read())

 extracted_text = []
 for item in response['Blocks']:
 if item['BlockType'] == 'LINE':
 extracted_text.append(item['Text'])

 return '\n'.join(extracted_text)

def save_text_to_file(text, file_path):
 """Save the extracted text to a file."""
```

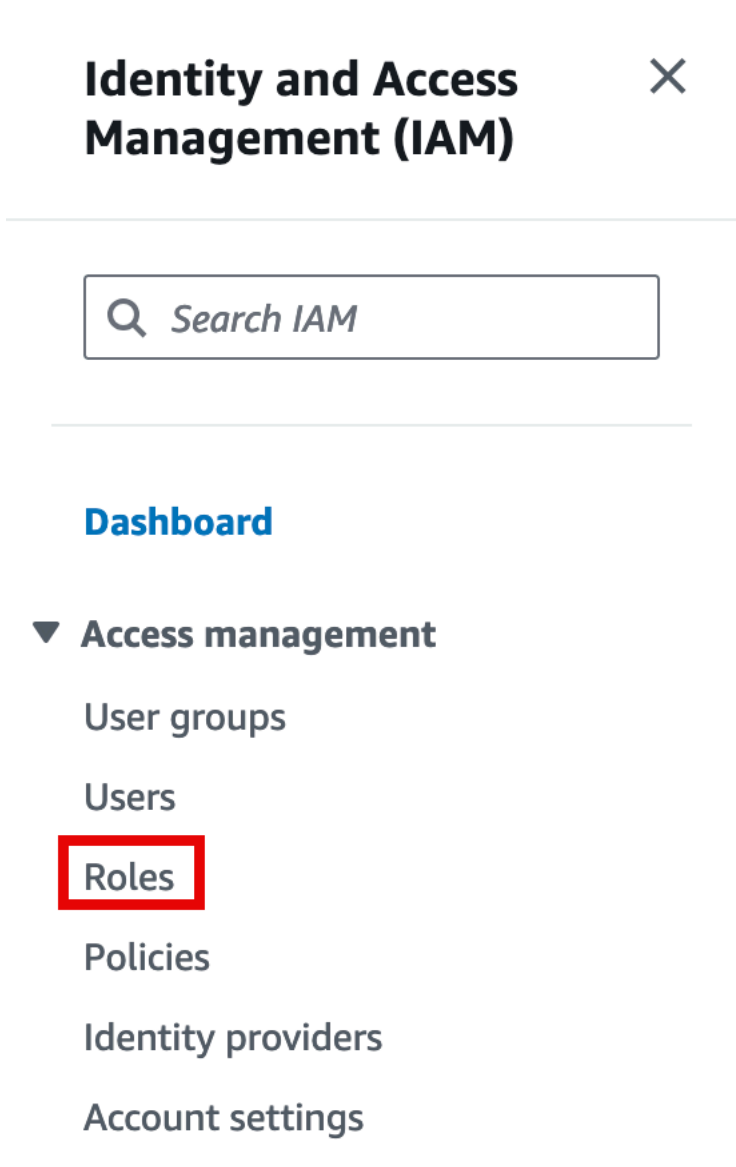
An overlay window titled "Extract information..." is shown, displaying the extracted text from an image: "health\_insurance\_card\_r...cted.png".

## 10. Configure permissions

Before you can run the code in your JupyterLab, the IAM role that was previously created for your Jupyter notebook, needs the appropriate permissions to run the AWS services that your code is going to use. If you chose to use the previous example, Amazon Textract is the AWS service that would need the appropriate permissions.

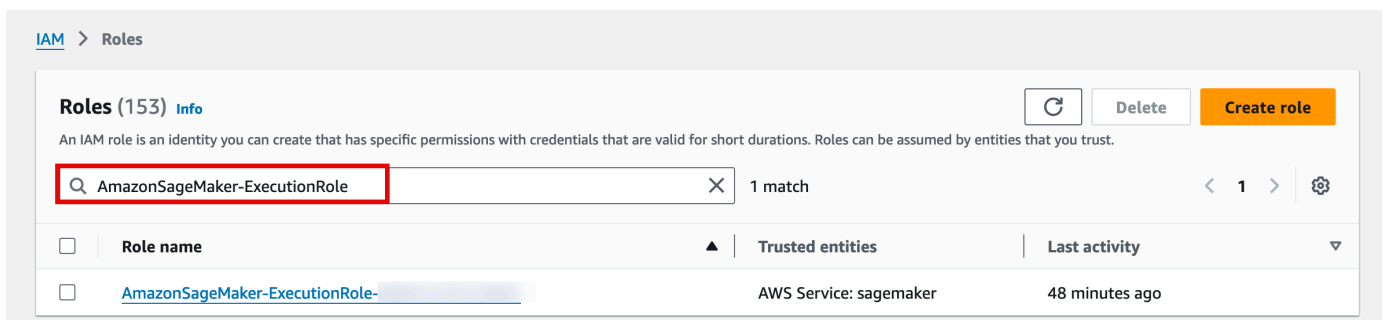
**Open** the AWS IAM console at <https://console.aws.amazon.com/iam/home>.

In the left navigation pane, choose **Roles**.



### 11. Search for the IAM role

In the **search box**, find the previously created **AmazonSageMaker AI-ExecutionRole-  
<timestamp>** role, and **open** the role.



## 12. Add permissions

On the **AmazonSageMaker AI-ExecutionRole-<timestamp>** page, choose the **Add permissions** drop down, and select **Attach policies**.

The screenshot shows the Amazon SageMaker console page for an execution role. The page title is "AmazonSageMaker-ExecutionRole-**[redacted]**". Below the title, there is a description: "SageMaker execution role created from the SageMaker AWS Management Console." and a "Delete" button. The "Summary" section includes details like "Creation date: November 13, 2024, 11:35 (UTC-05:00)", "Last activity: 49 minutes ago", "ARN: arn:aws:iam::[redacted]:role/service-role/AmazonSageMaker-ExecutionRole-[redacted]", and "Maximum session duration: 1 hour". Below the summary, there are tabs for "Permissions", "Trust relationships", "Tags", "Last Accessed", and "Revoke sessions". The "Permissions" tab is active, showing "Permissions policies (2)". A dropdown menu is open, with "Add permissions" selected. The dropdown options are "Add permissions", "Attach policies", and "Create inline policy". Below the dropdown, there is a search bar and a table of permissions policies.

Policy name	Type	Attached entities
AmazonSageMaker-Execution...	Customer managed	1
AmazonSageMakerFullAcc...	AWS managed	3

## 13. Attach the policy

On the **Attach policy to AmazonSageMaker AI-ExecutionRole-<timestamp>** page, in the **Other permissions policies** section search bar, enter **AmazonAmazon TextractFullAccess**. Then, select the **policy**, and choose **Add permissions**.


### Attach policy to AmazonSageMaker-ExecutionRole-

► **Current permissions policies (2)**

**Other permissions policies (1/996)** ↻

Filter by Type

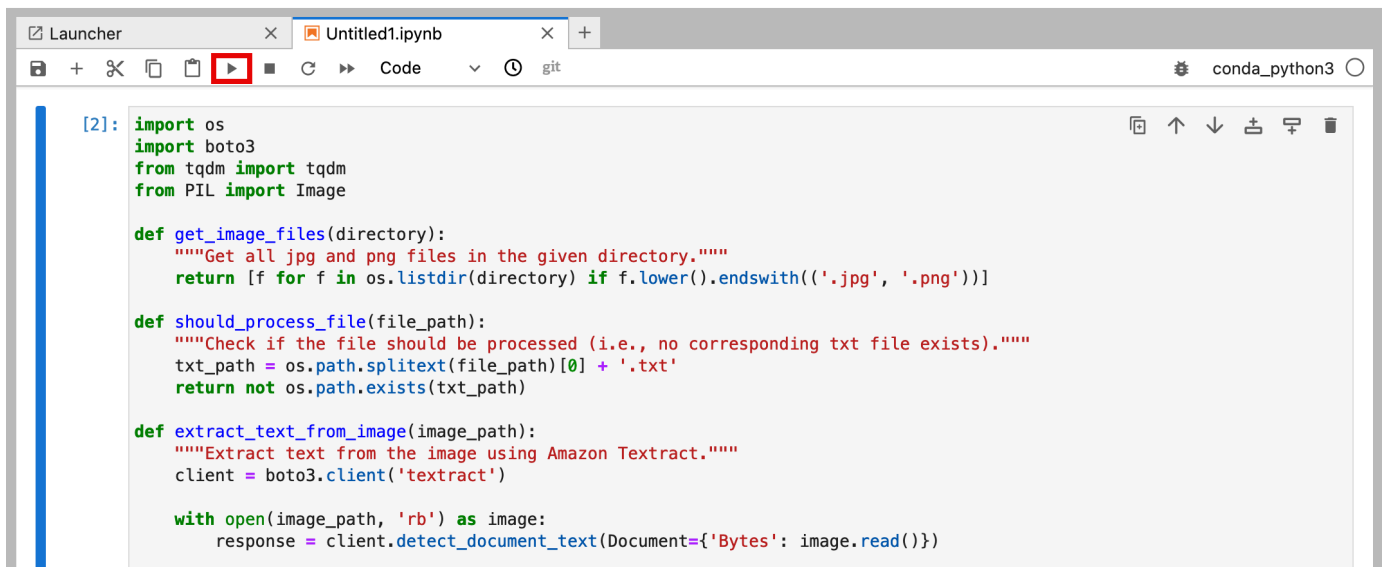
✕ All types ▼ 1 match < 1 > ⚙️

<input checked="" type="checkbox"/>	Policy name	Type
<input checked="" type="checkbox"/>	 <a href="#">AmazonTextractFullAccess</a>	AWS managed

Cancel Add permissions

## 14. Run the notebook

Navigate back to your **JupyterLab** tab, and select **Run**.



```
[2]: import os
import boto3
from tqdm import tqdm
from PIL import Image

def get_image_files(directory):
 """Get all jpg and png files in the given directory."""
 return [f for f in os.listdir(directory) if f.lower().endswith(('.jpg', '.png'))]

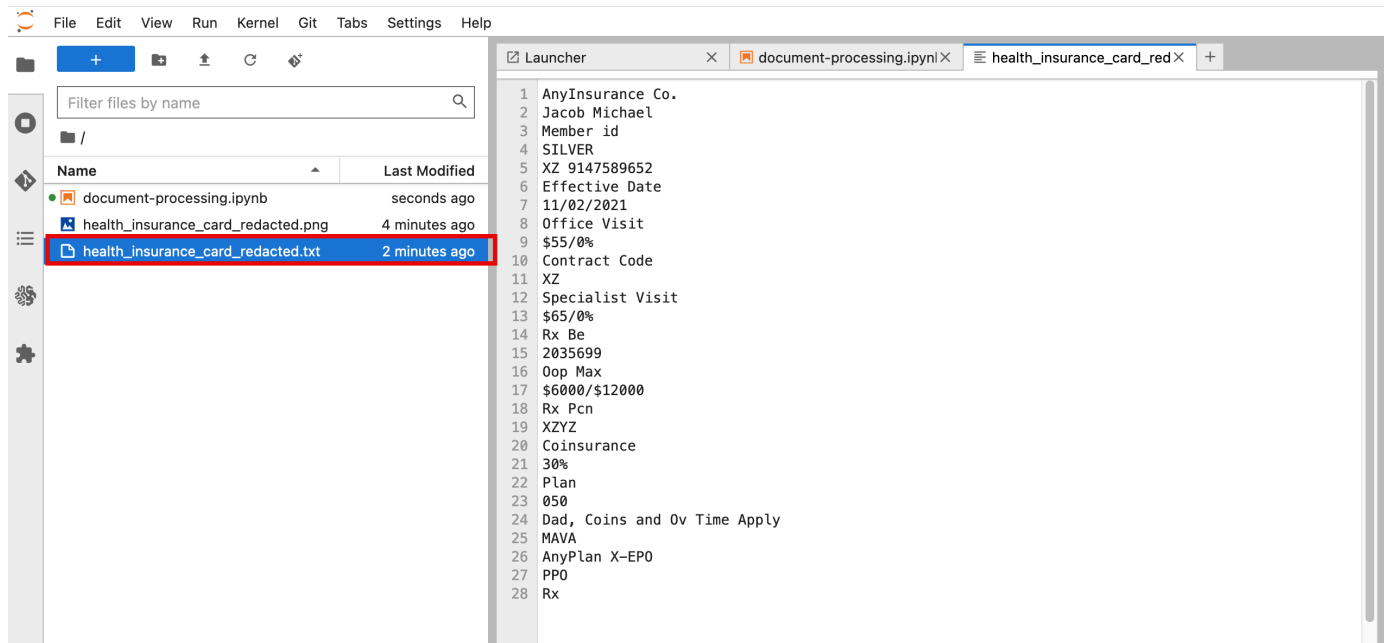
def should_process_file(file_path):
 """Check if the file should be processed (i.e., no corresponding txt file exists)."""
 txt_path = os.path.splitext(file_path)[0] + '.txt'
 return not os.path.exists(txt_path)

def extract_text_from_image(image_path):
 """Extract text from the image using Amazon Textract."""
 client = boto3.client('textract')

 with open(image_path, 'rb') as image:
 response = client.detect_document_text(Document={'Bytes': image.read()})
```

## 15. View the text file

After your code runs you should now be able to see a **.txt file** with the extracted text in the left navigation pane of your JupyterLab.



## Clean up resources

In this step, you will go through the steps to delete all the resources you created throughout this tutorial. It is recommended that you stop the Jupyter notebook you created to prevent unexpected costs.

- Delete the notebook

In the SageMaker AI console, in the left navigation pane, choose **Notebooks**, and select the **Notebook**. Then, choose **Actions**, and select **Stop**.

### Note

The stop operation might take around 5 minutes. Once the notebook is stopped you can also delete it by choosing **Actions** and selecting **Delete**.

Amazon SageMaker > Notebooks and Git Repos



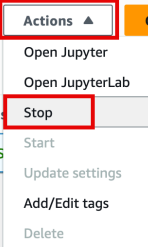
## Notebooks and Git repos

Try the new JupyterLab in SageMaker Studio

Notebook instances | Git repositories

Notebook instances Info

Search notebook instances

Name	Instance	Creation time	Status	Actions
 document-processing	ml.t3.medium	11/13/2024, 11:36:22 AM		

Start | Open JupyterLab

## Congratulations

You have created a sample Proof-of-Concept to extract information from documents.