Open in app ↗

Search Medium

Get unlimited access to the best of Medium for less than $1/week. Become a member

Deploy Llama2–7B on AWS (Follow Along)

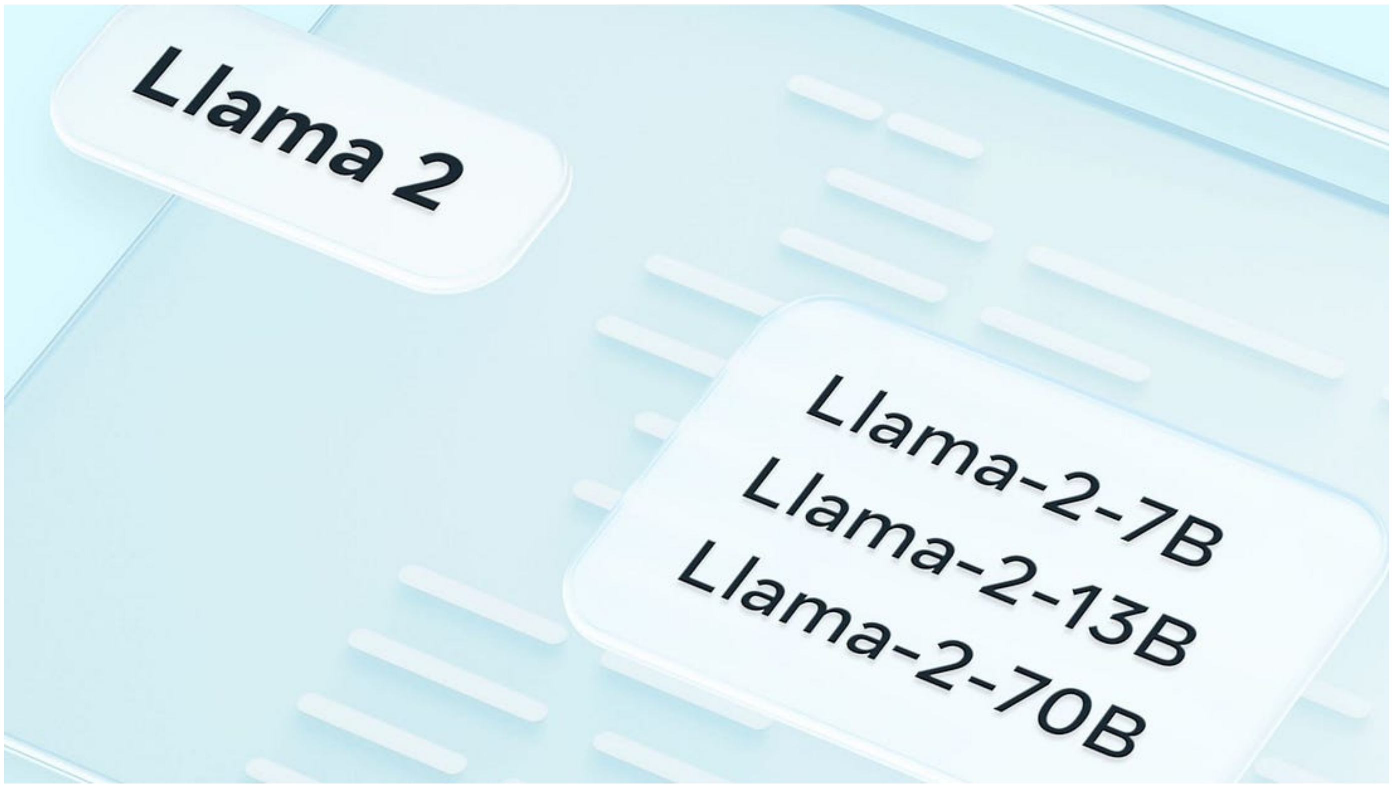
Mudassir Aqeel Ahmed · Follow
7 min read · Aug 25

Listen Share More

This blog follows the easiest flow to set and maintain any Llama2 model on the cloud, This one features the **7B** one, but you can follow the same steps for 13B or 70B. It is divided into two sections

Section — 1: Deploy model on AWS Sagemaker

Section — 2: Run as an API in your application



Llama 2 is a collection of pre-trained and fine-tuned generative text models developed by Meta. These models range in scale from 7 billion to 70 billion parameters and are designed for various text-generation tasks. The models in the Llama 2 family, particularly the Llama-2-Chat variations, are optimized for dialogue use cases, outperforming open-source chat models in most benchmarks and being on par with some popular closed-source models like ChatGPT and PaLM in terms of helpfulness and safety.

Key Details:

- **Training Data:** Pretraining data includes a mix of publicly available online data while fine-tuning data includes instruction datasets and new human-annotated examples.
- **Training Period:** Trained between January 2023 and July 2023.
- **Data Freshness:** Pretraining data is up to September 2022, and some fine-tuning data is more recent, up to July 2023.

Evaluation Results:

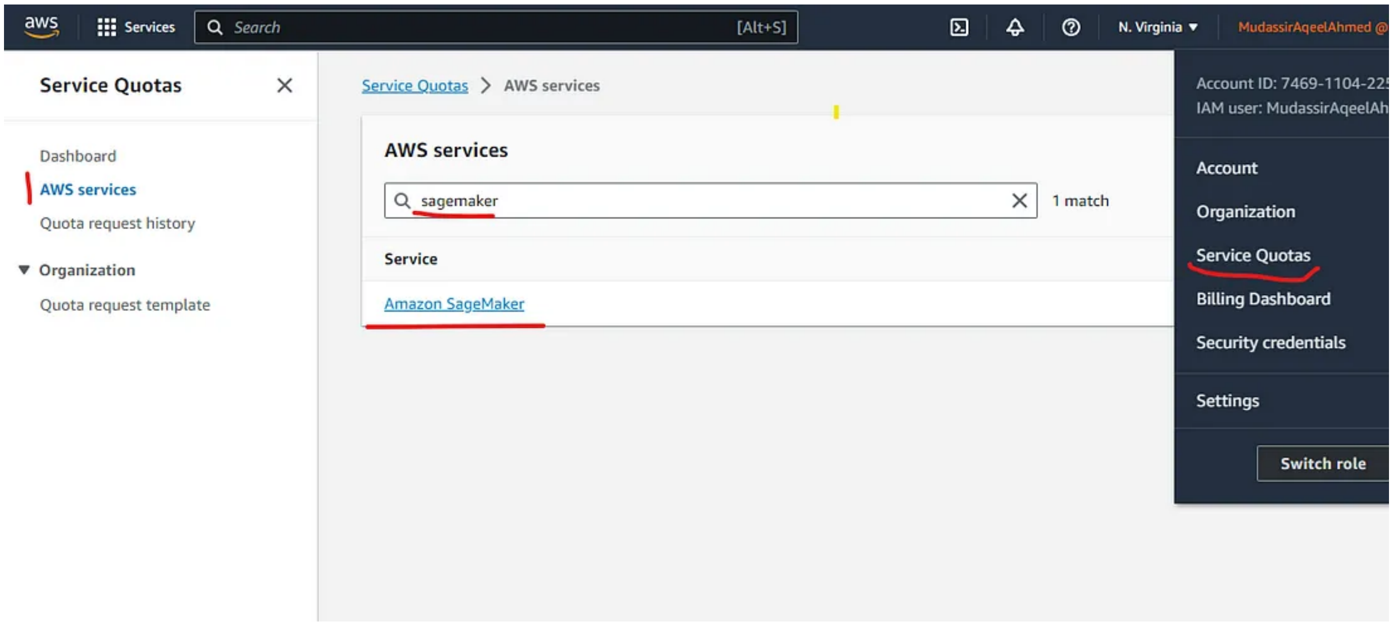
Llama 2 models show improved performance compared to Llama 1 models on various evaluation benchmarks, including commonsense reasoning, world knowledge, reading comprehension, math, and other linguistic tasks.

1. Deploying on AWS Sagemaker

You need to have an AWS Account with administrator privileges to be able to run and deploy the Llama-2–7B model, first login, and head to the Amazon Sagemaker console (Try to be on the us-east-1, N. Virginia region).

Request Quota:

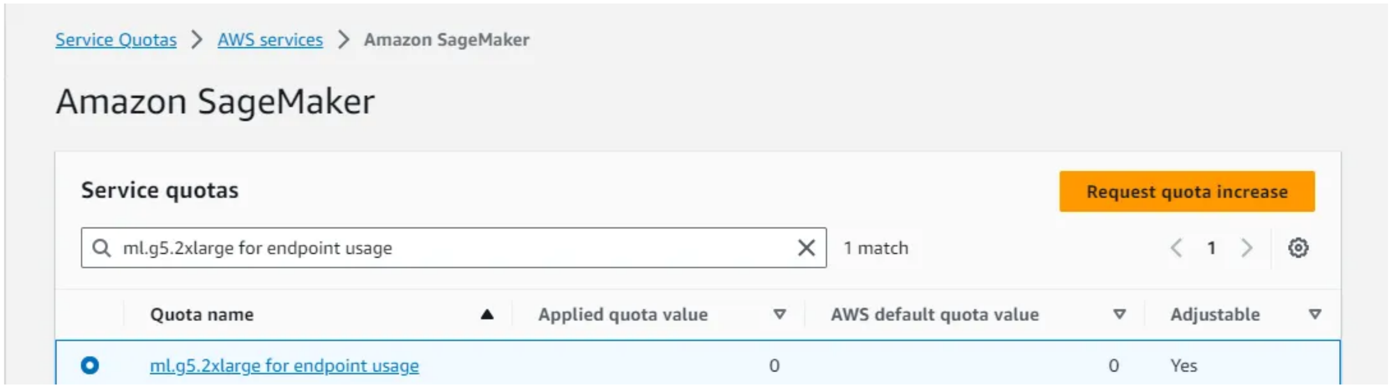
The resources in Amazon Sagemaker are not always granted so one should make a quick check



Search for these service quotas in Sagemaker,

- Total domains
- Maximum number of Studio user profiles allowed per account
- ml.g5.2xlarge for endpoint usage
- Maximum number of running Studio apps allowed per account

If the applied quota value is 0 for any of these services you need to request for quota increase, You can track the requests in the quota request history, it can take up to 2 days at times.



Service Quotas > AWS services > Amazon SageMaker

Amazon SageMaker

Service quotas

Request quota increase

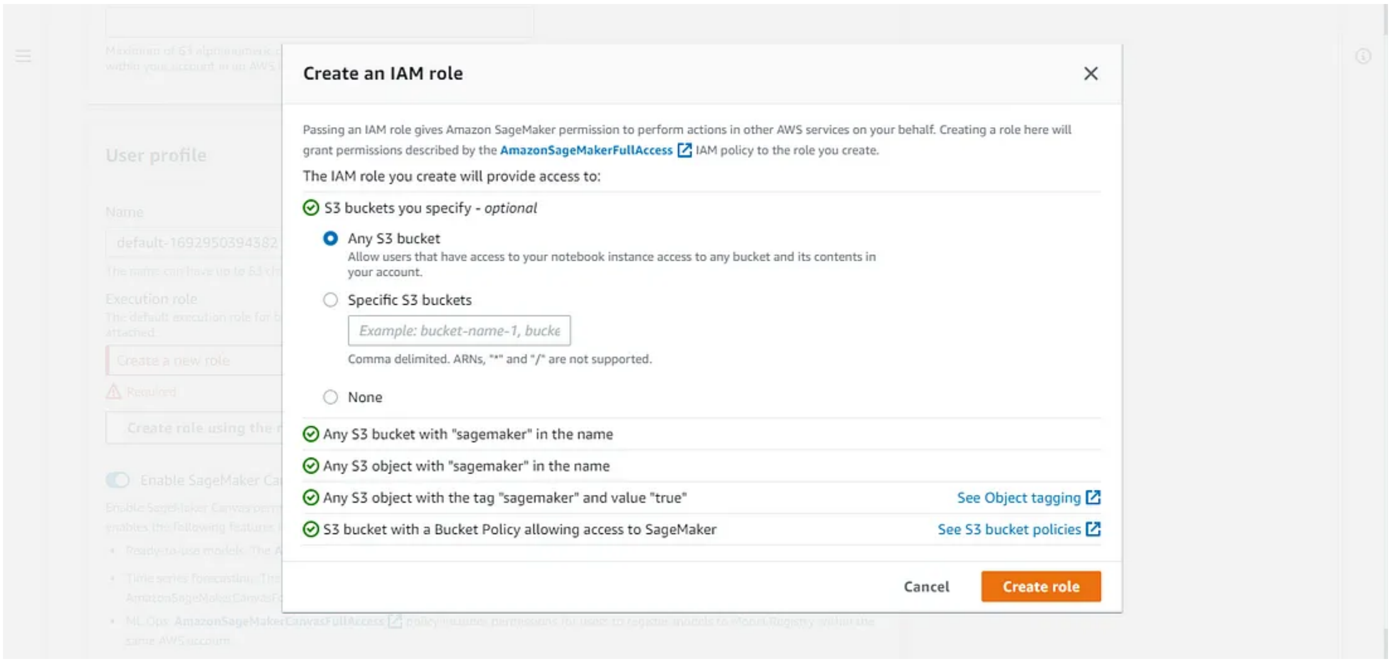
ml.g5.2xlarge for endpoint usage — 1 match

	Quota name	Applied quota value	AWS default quota value	Adjustable
●	ml.g5.2xlarge for endpoint usage	0	0	Yes

Create Domain:

The very first task is to create a domain if you don't have any (you will not, if this is your first time at Sagemaker)

- Select Quick Setup
- Choose a domain name
- You can keep the user profile name as default or change it if you want
- You will need to create a role if you don't have any.



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
Example: bucket-name-1, bucke
Comma delimited. ARNs, "*" and "/" are not supported.

○ None

✓ Any S3 bucket with "sagemaker" in the name

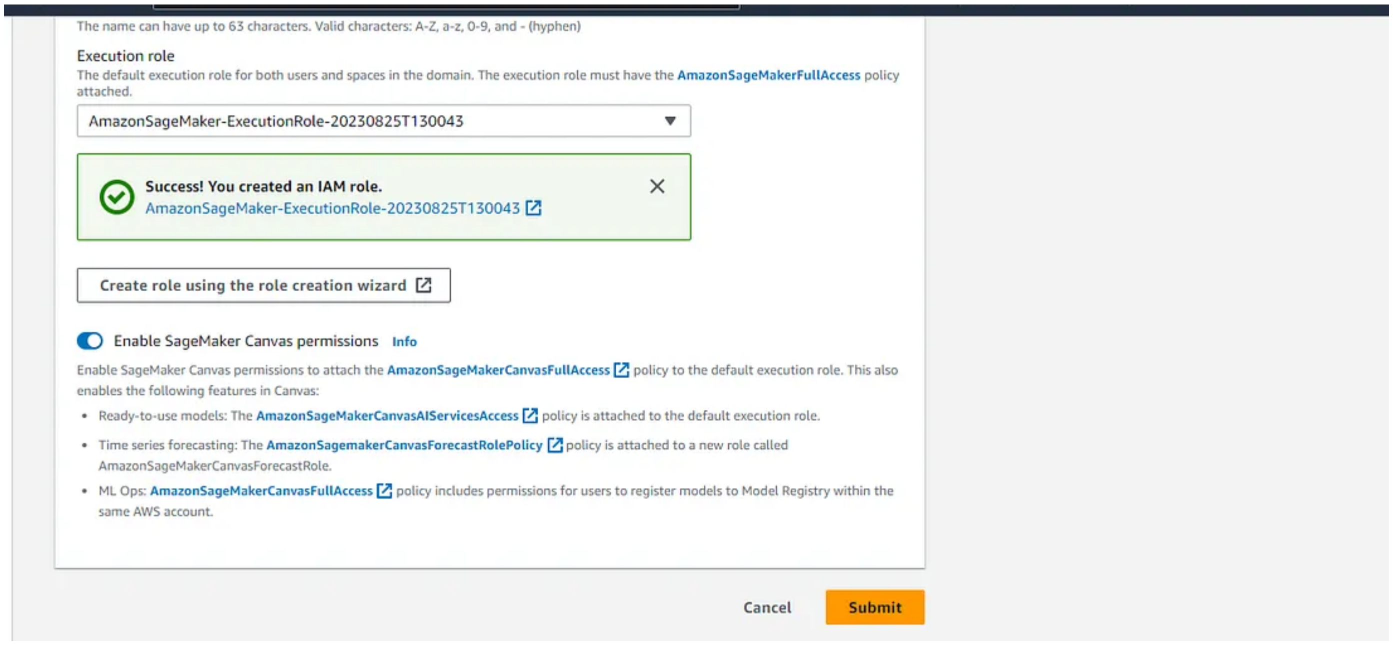
✓ Any S3 object with "sagemaker" in the name

✓ 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

Cancel | Create role

- choose "Any S3 bucket" and hit create.


The name can have up to 63 characters. Valid characters: A-Z, a-z, 0-9, and - (hyphen)

Execution role
The default execution role for both users and spaces in the domain. The execution role must have the AmazonSageMakerFullAccess policy attached.

AmazonSageMaker-ExecutionRole-20230825T130043

Success! You created an IAM role.
AmazonSageMaker-ExecutionRole-20230825T130043

Create role using the role creation wizard

Enable SageMaker Canvas permissions Info

Enable SageMaker Canvas permissions to attach the AmazonSageMakerCanvasFullAccess policy to the default execution role. This also enables the following features in Canvas:

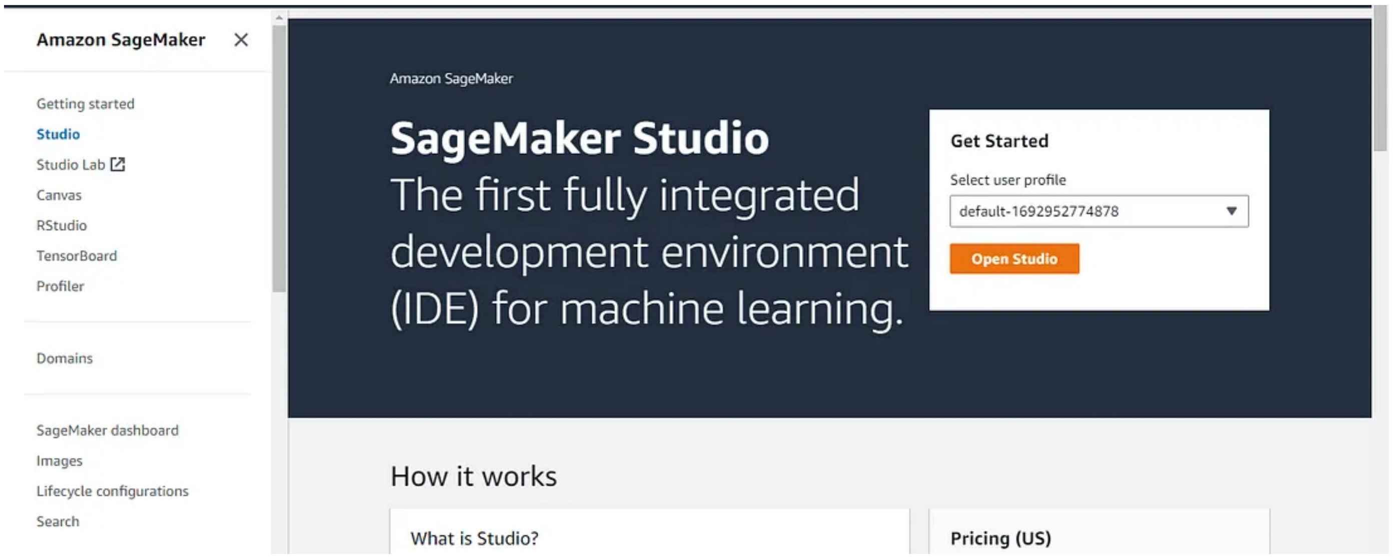
- Ready-to-use models: The AmazonSageMakerCanvasAIServicesAccess policy is attached to the default execution role.
- Time series forecasting: The AmazonSagemakerCanvasForecastRolePolicy policy is attached to a new role called AmazonSageMakerCanvasForecastRole.
- ML Ops: AmazonSageMakerCanvasFullAccess policy includes permissions for users to register models to Model Registry within the same AWS account.

This is how it should look, hit submit to create the domain.

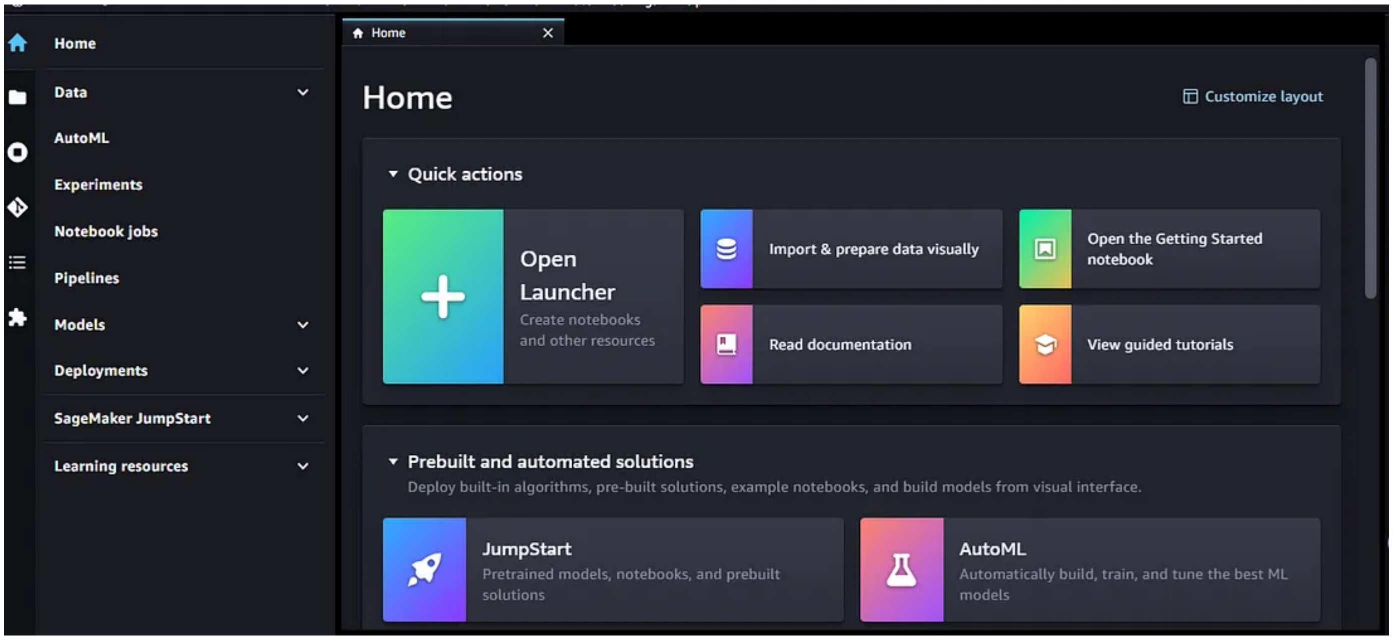
If there was an error during the creation of your domain, it probably stems from issues with user permissions or VPC configuration.

Launch Studio and Deploy Model

After you successfully create your domain and user profile, launch sagemaker studio

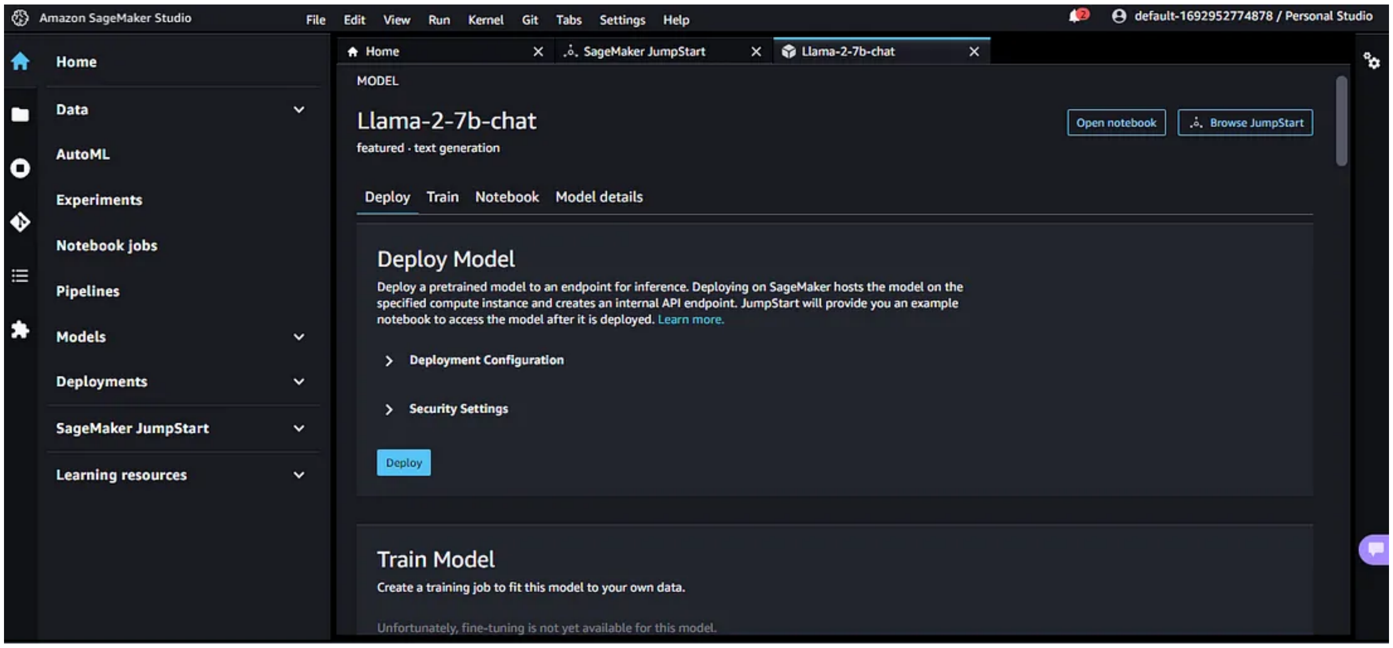


The user profile should be the one you just created in your domain


Home
Data
AutoML
Experiments
Notebook jobs
Pipelines
Models
Deployments
SageMaker JumpStart
Learning resources
Home
Customize layout
Quick actions
Open Launcher
Create notebooks and other resources
Import & prepare data visually
Open the Getting Started notebook
Read documentation
View guided tutorials
Prebuilt and automated solutions
Deploy built-in algorithms, pre-built solutions, example notebooks, and build models from visual interface.
JumpStart
Pretrained models, notebooks, and prebuilt solutions
AutoML
Automatically build, train, and tune the best ML models

Sagemaker Studio Homepage

Go to Jumpstart and search for Llama2–7b-chat


Amazon SageMaker Studio
File Edit View Run Kernel Git Tabs Settings Help
default-1692952774878 / Personal Studio
Home
SageMaker JumpStart
Llama-2-7b-chat
MODEL
Llama-2-7b-chat
featured · text generation
Open notebook
Browse JumpStart
Deploy Train Notebook Model details

Deploy Model

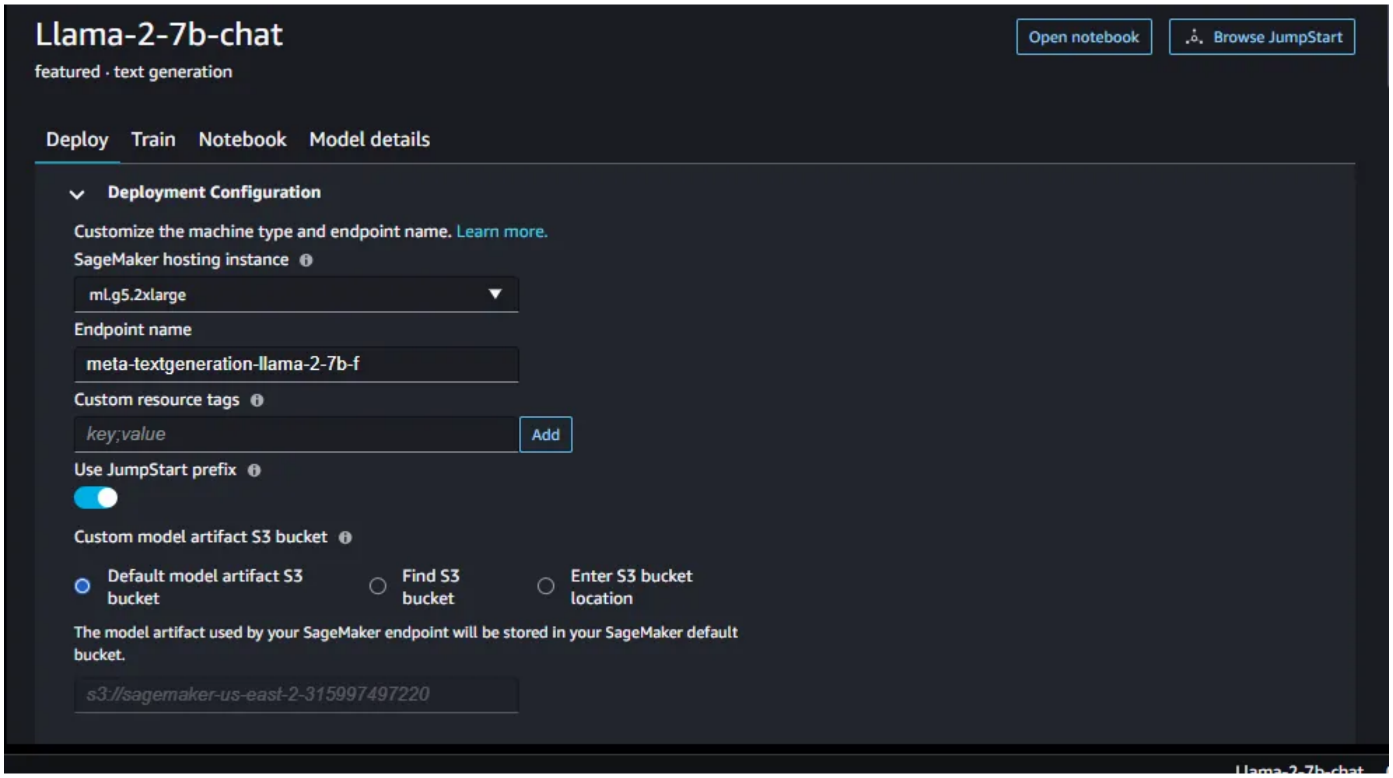
Deploy a pretrained model to an endpoint for inference. Deploying on SageMaker hosts the model on the specified compute instance and creates an internal API endpoint. JumpStart will provide you an example notebook to access the model after it is deployed. Learn more.

Train Model

Create a training job to fit this model to your own data.

Unfortunately, fine-tuning is not yet available for this model.

Llama2–7b-chat model page.


Llama-2-7b-chat
featured · text generation

Deployment Configuration

Customize the machine type and endpoint name. Learn more.

SageMaker hosting instance: ml.g5.2xlarge

Endpoint name: meta-textgeneration-llama-2-7b-f

Custom resource tags: key;value

Use JumpStart prefix: on

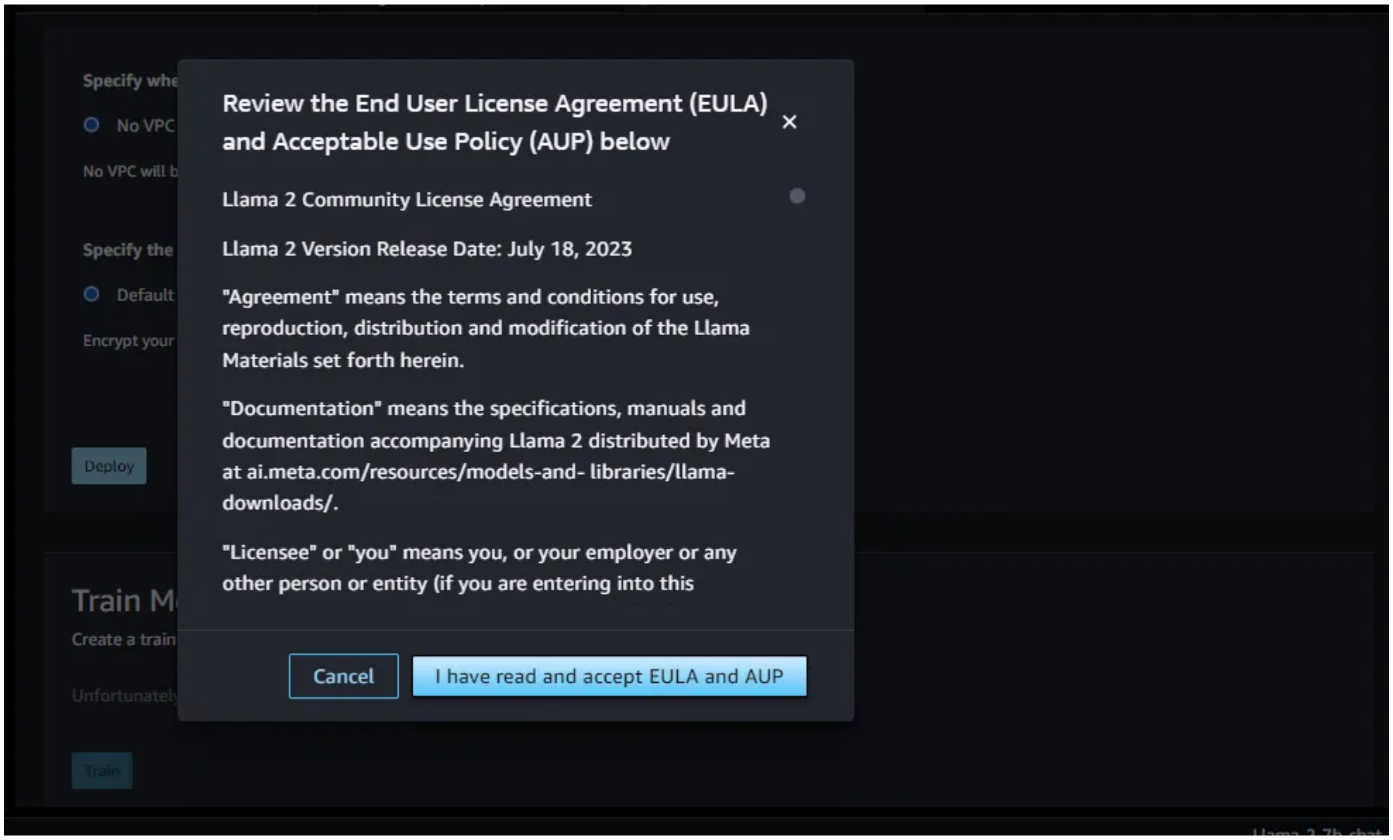
Custom model artifact S3 bucket: [✓ Default model artifact S3 bucket] Find S3 bucket Enter S3 bucket location

The model artifact used by your SageMaker endpoint will be stored in your SageMaker default bucket.

s3://sagemaker-us-east-2-315997497220

You can leave all configs to default, ml.g5.2xlarge is the least model required to run llama2–7b, it costs $1.515/hr, $36.36/day if you leave it running :-)

Click deploy to deploy the model as an endpoint, You will need to accept the license agreement, the deployment will take a few minutes.



Review the End User License Agreement (EULA) and Acceptable Use Policy (AUP) below

Llama 2 Community License Agreement

Llama 2 Version Release Date: July 18, 2023

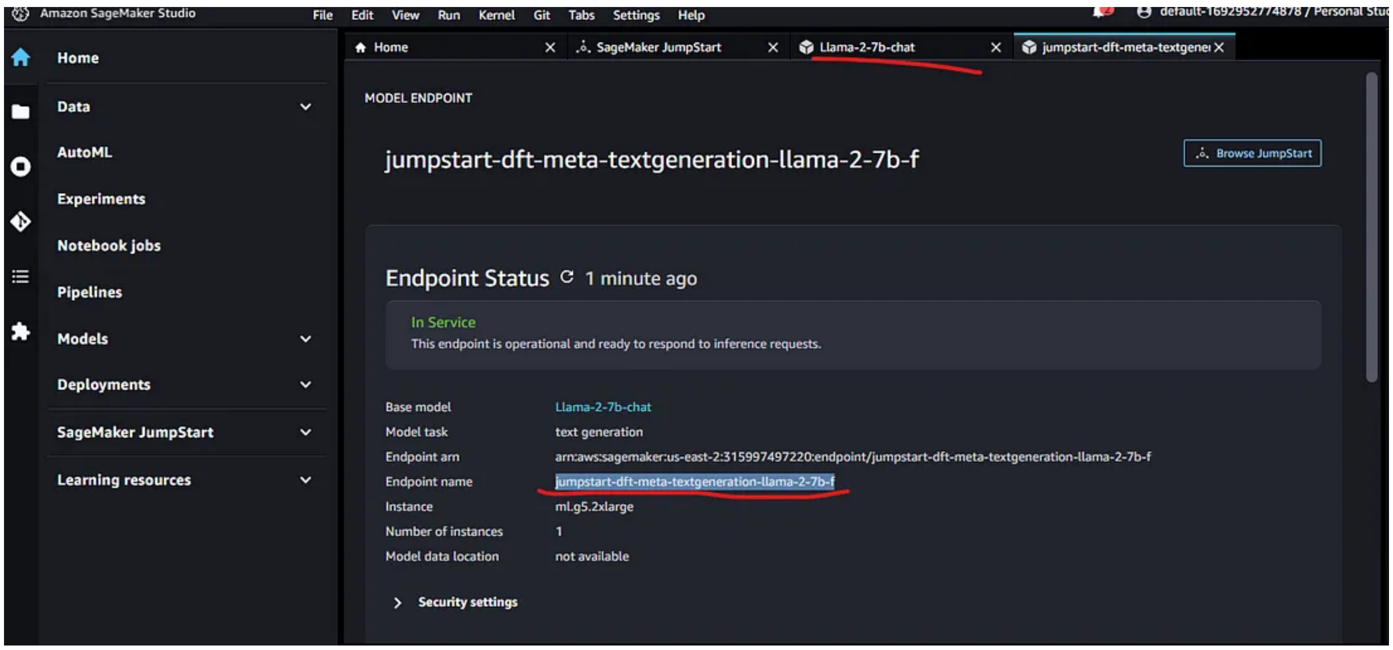
"Agreement" means the terms and conditions for use, reproduction, distribution and modification of the Llama Materials set forth herein.

"Documentation" means the specifications, manuals and documentation accompanying Llama 2 distributed by Meta at ai.meta.com/resources/models-and- libraries/llama-downloads/.

"Licensee" or "you" means you, or your employer or any other person or entity (if you are entering into this

meta's EULA before using any llama2 models

At this point your model is deployed you can run inference (queries) with it, by opening the notebook from the llama-7b-chat model page, and test the model



MODEL ENDPOINT

jumpstart-dft-meta-textgeneration-llama-2-7b-f

Endpoint Status 1 minute ago

In Service
This endpoint is operational and ready to respond to inference requests.

Base model	Llama-2-7b-chat
Model task	text generation
Endpoint arn	arn:aws:sagemaker:us-east-2:315997497220:endpoint/jumpstart-dft-meta-textgeneration-llama-2-7b-f
Endpoint name	jumpstart-dft-meta-textgeneration-llama-2-7b-f
Instance	ml.g5.2xlarge
Number of instances	1
Model data location	not available

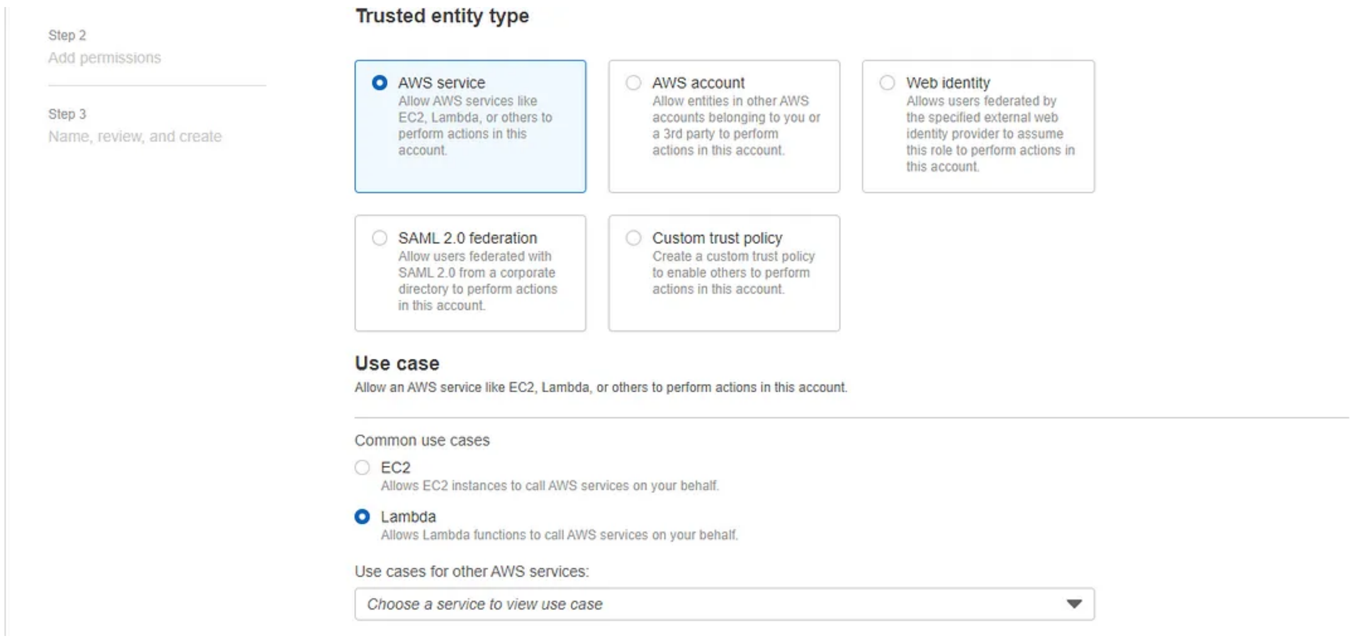
Security settings

2. Run as an API

Create IAM role for AWS Lambda

Go to IAM > Roles > create role

Select AWS Service and lambda service and click Next.



Step 2
Add permissions

Step 3
Name, review, and create

Trusted entity type

- [✓ AWS service] Allow AWS services like EC2, Lambda, or others to perform actions in this account.
- AWS account: Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.
- Web identity: Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.
- SAML 2.0 federation: Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.
- Custom trust policy: Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- EC2: Allows EC2 instances to call AWS services on your behalf.
- [✓ Lambda] Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case

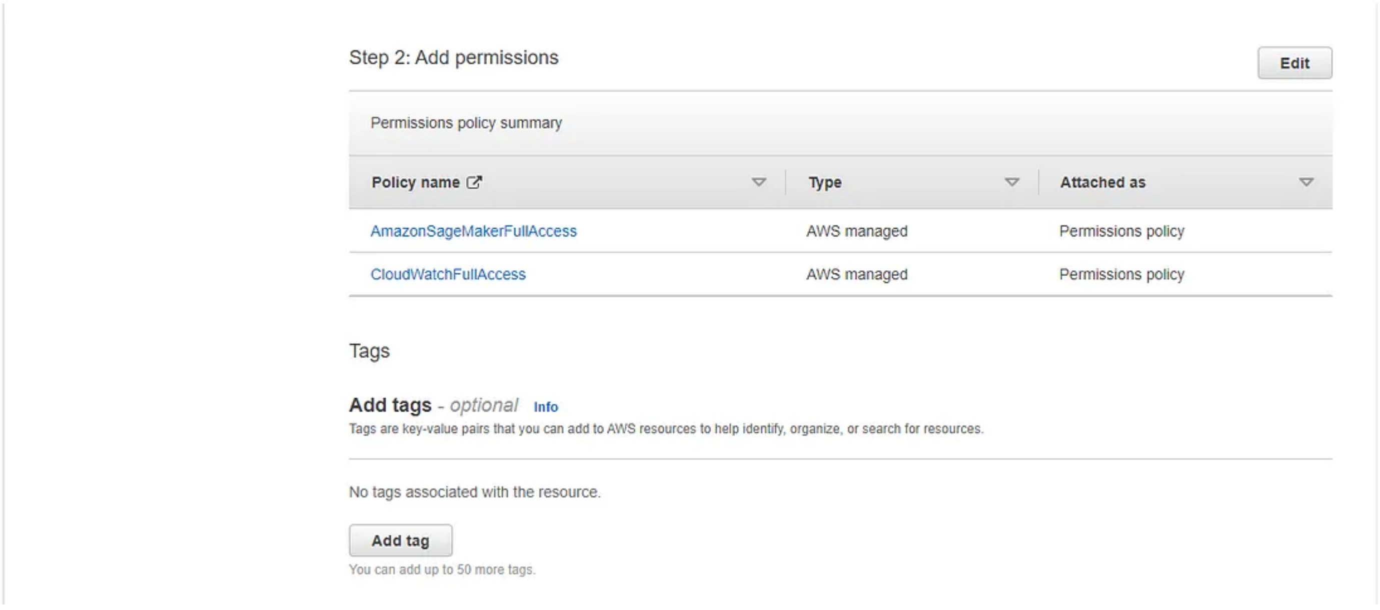
Search for these two policies, and click Next

1. CloudWatchFullAccess
2. AmazonSageMakerFullAccess

These are probably overkill for the task at hand but take away the complexity.

Add your role name and description (optional), and verify the policies you selected are added as permissions to the role.

Click Create role to create.


Step 2: Add permissions

Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonSageMakerFullAccess	AWS managed	Permissions policy
CloudWatchFullAccess	AWS managed	Permissions policy

Tags

Add tags - optional Info

Tags are key-value pairs that you can add to AWS resources to help identify, organize, or search for resources.

No tags associated with the resource.

Add tag

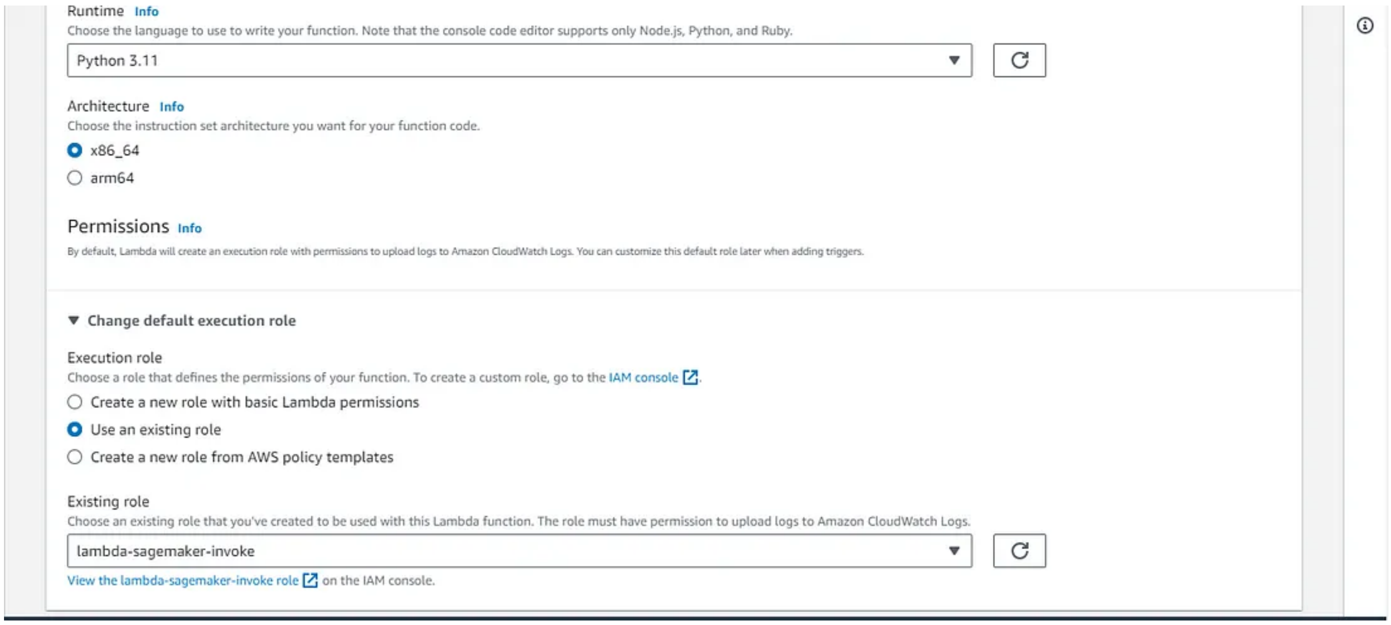
You can add up to 50 more tags.

Create Lambda function

Go to Lambda > Create function

1. Author from scratch
2. Give it a name
3. Select runtime as Python 3.11
4. change default execution role > choose an existing role, select the role you just created

Click on the create function (leave the advanced setting as default).


Runtime Info

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.11

Architecture Info

Choose the instruction set architecture you want for your function code.

[✓ x86_64] arm64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

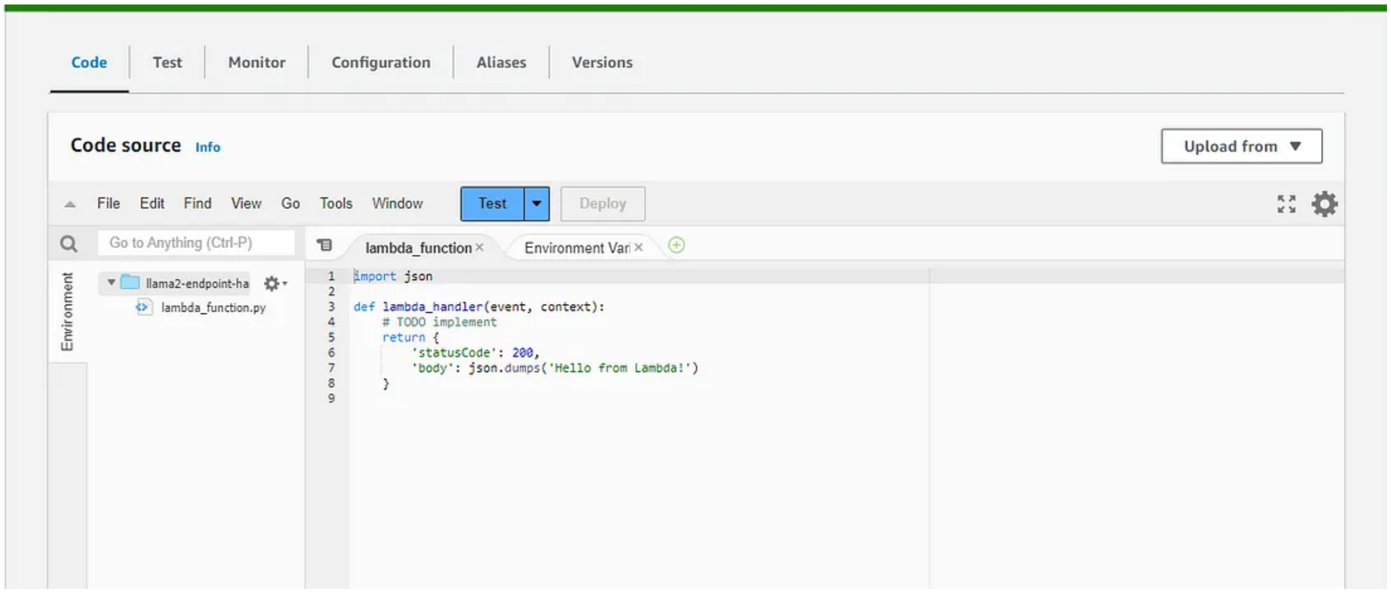
Create a new role with basic Lambda permissions [✓ Use an existing role] Create a new role from AWS policy templates

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

lambda-sagemaker-invoke

View the lambda-sagemaker-invoke role on the IAM console.

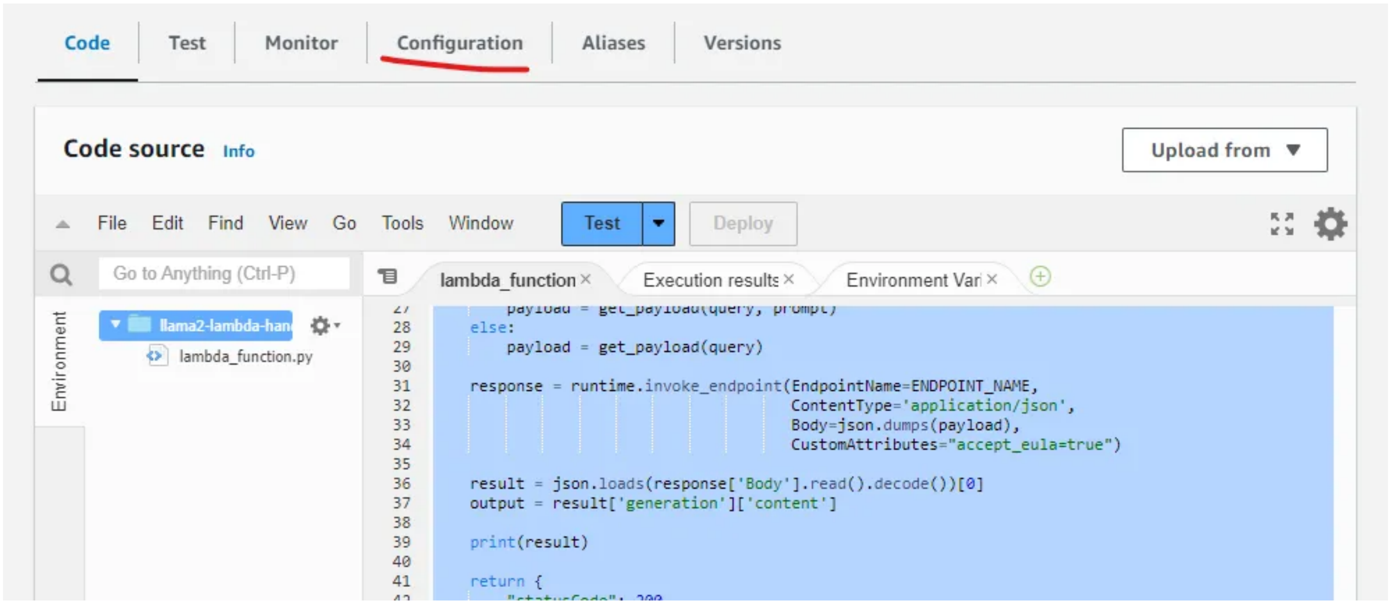


```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

lambda function created

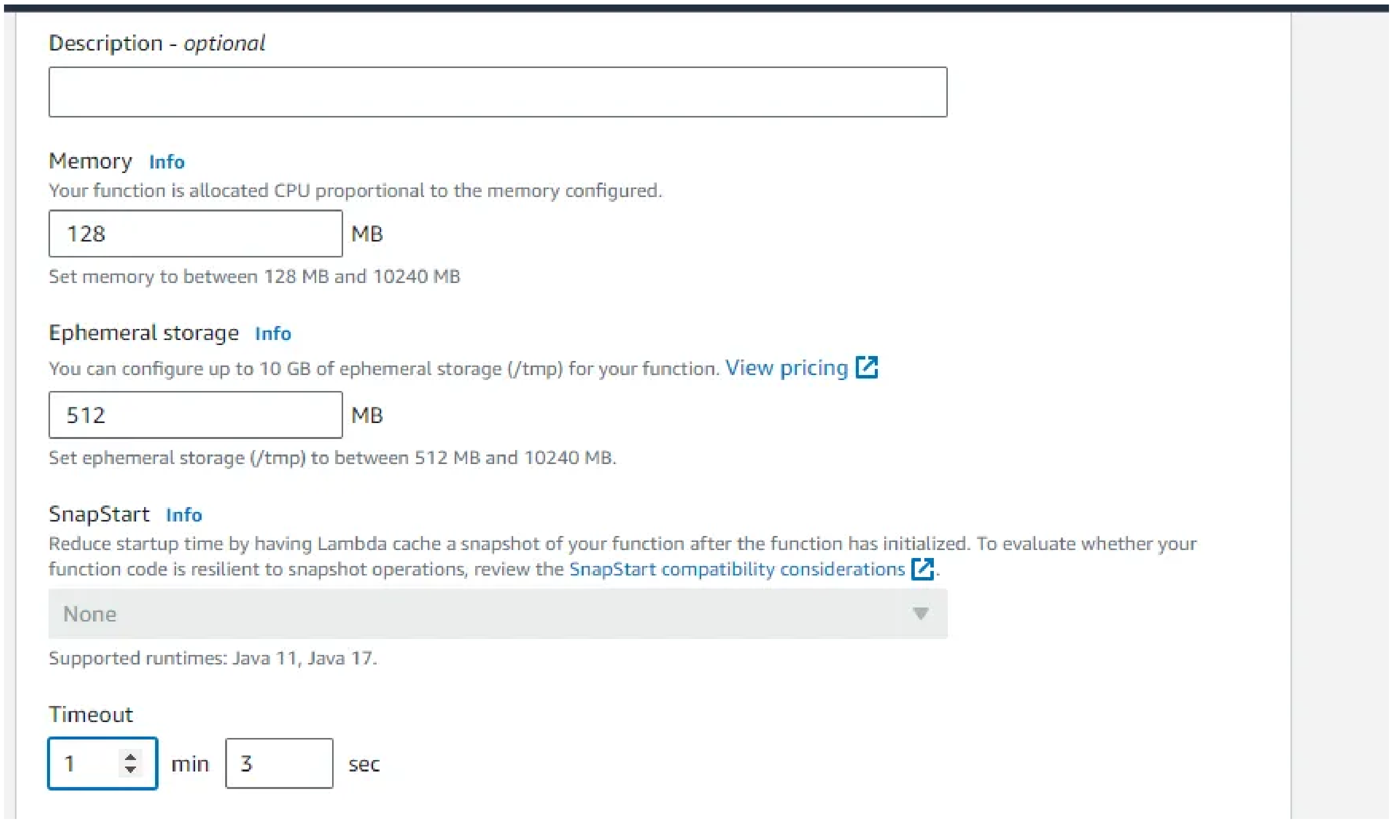
```
import os
import io
import boto3
import json

# grab environment variables
ENDPOINT_NAME = os.environ['ENDPOINT_NAME']
runtime= boto3.client('runtime.sagemaker')

def get_payload(query: str, prompt: str | None = None, max_new_tokens: int = 4000
    if prompt:
        inputs = [
            {"role": "system", "content": prompt},
            {"role": "user", "content": query}]
    else:
        inputs = [{"role": "user", "content": query}]
    payload = {
        "inputs": [inputs],
        "parameters": {"max_new_tokens": max_new_tokens, "top_p": top_p, "tempera
    }
    return payload

def lambda_handler(event, context):
    query = event["query"]
    if "prompt" in event:
        prompt = event["prompt"]
        payload = get_payload(query, prompt)
    else:
        payload = get_payload(query)
    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='application/json',
                                       Body=json.dumps(payload),
                                       CustomAttributes="accept_eula=true")

    result = json.loads(response['Body'].read().decode())[0]
    output = result['generation']['content']

    print(result)

    return {
        "statusCode": 200,
        "body": output
    }
```

Copy this code to your lambda function, and go to configurations



```
        payload = get_payload(query, prompt)
    else:
        payload = get_payload(query)

    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       ContentType='application/json',
                                       Body=json.dumps(payload),
                                       CustomAttributes="accept_eula=true")

    result = json.loads(response['Body'].read().decode())[0]
    output = result['generation']['content']

    print(result)

    return {
        "statusCode": 200
```

on general configuration, click edit and change timeout from 3 sec to 1 min 3 sec (the max is 15 mins, but we don't need that much)


Description - *optional*

Memory Info
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

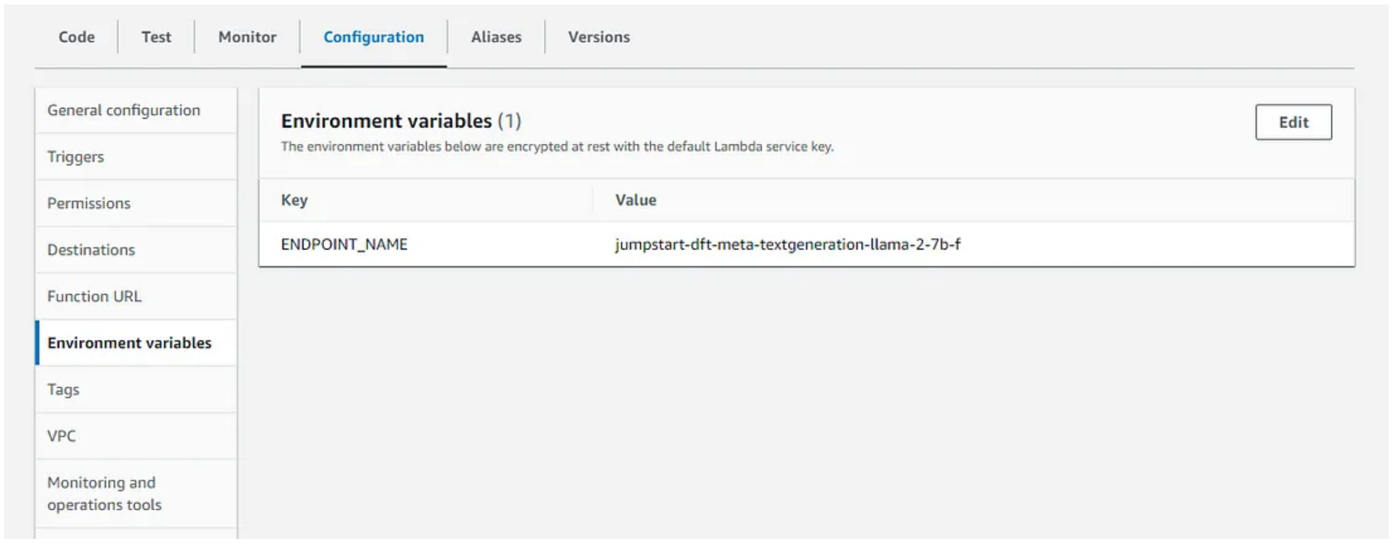
Ephemeral storage Info
You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

SnapStart Info
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17.

Timeout
1 min 3 sec

Edit environment variables and add your ENDPOINT_NAME (it was on the deployment page)

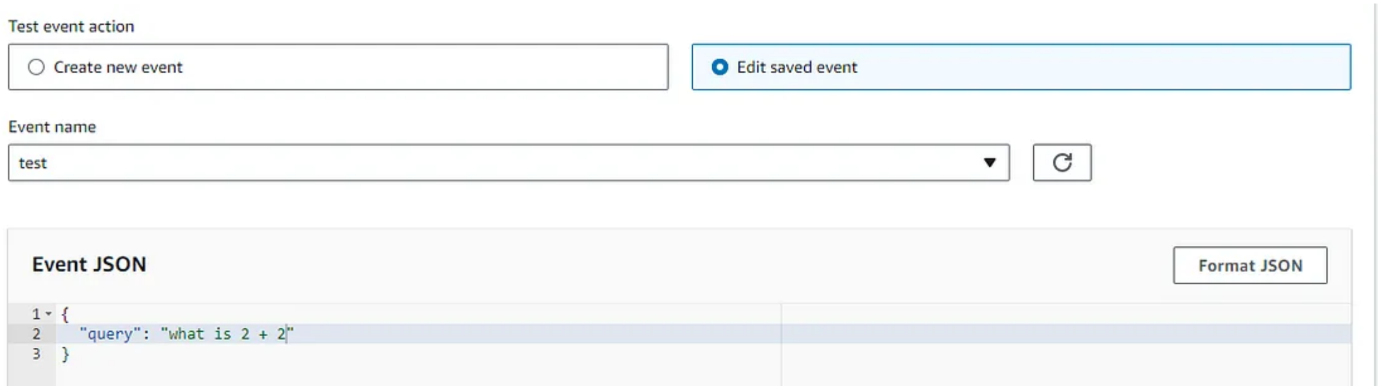
You can find it again on Sagemaker > inference > Endpoints (or from the studio deployment page, if you still have running)



Environment variables (1)
The environment variables below are encrypted at rest with the default Lambda service key.

Key	Value
ENDPOINT_NAME	jumpstart-dft-meta-textgeneration-llama-2-7b-f

After that, you can deploy your lambda and run a quick test,


Test event action

( ) Create new event [✓ Edit saved event]

Event name

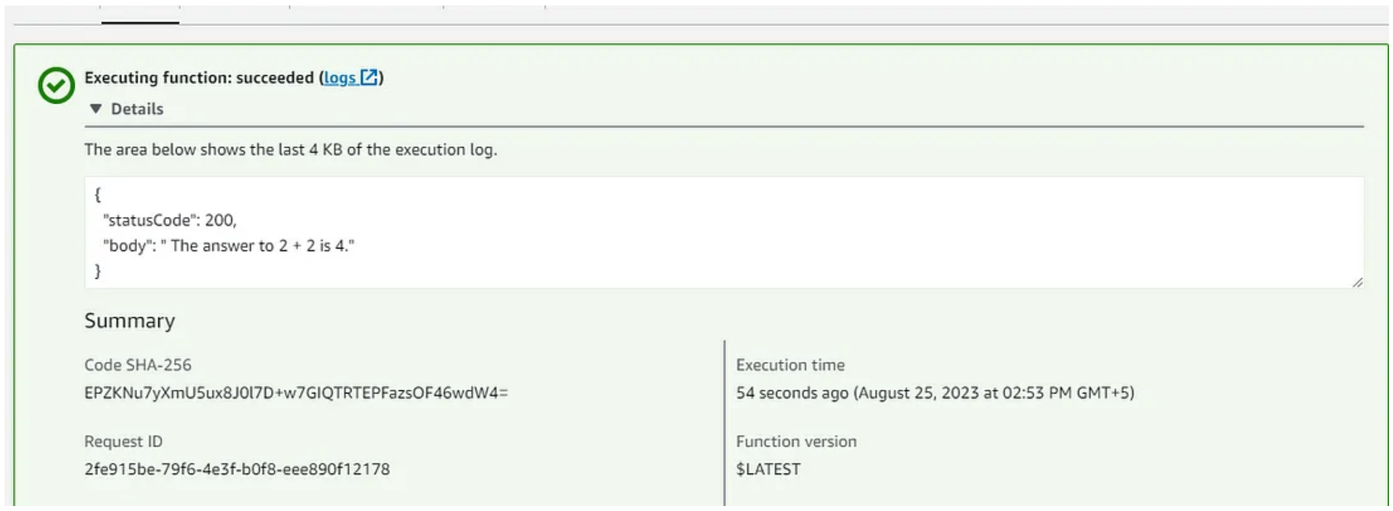
test

Event JSON

Format JSON

```
{
  "query": "what is 2 + 2"
}
```

If everything went well this should be your output response


Executing function: succeeded (logs)

Details

The area below shows the last 4 KB of the execution log.

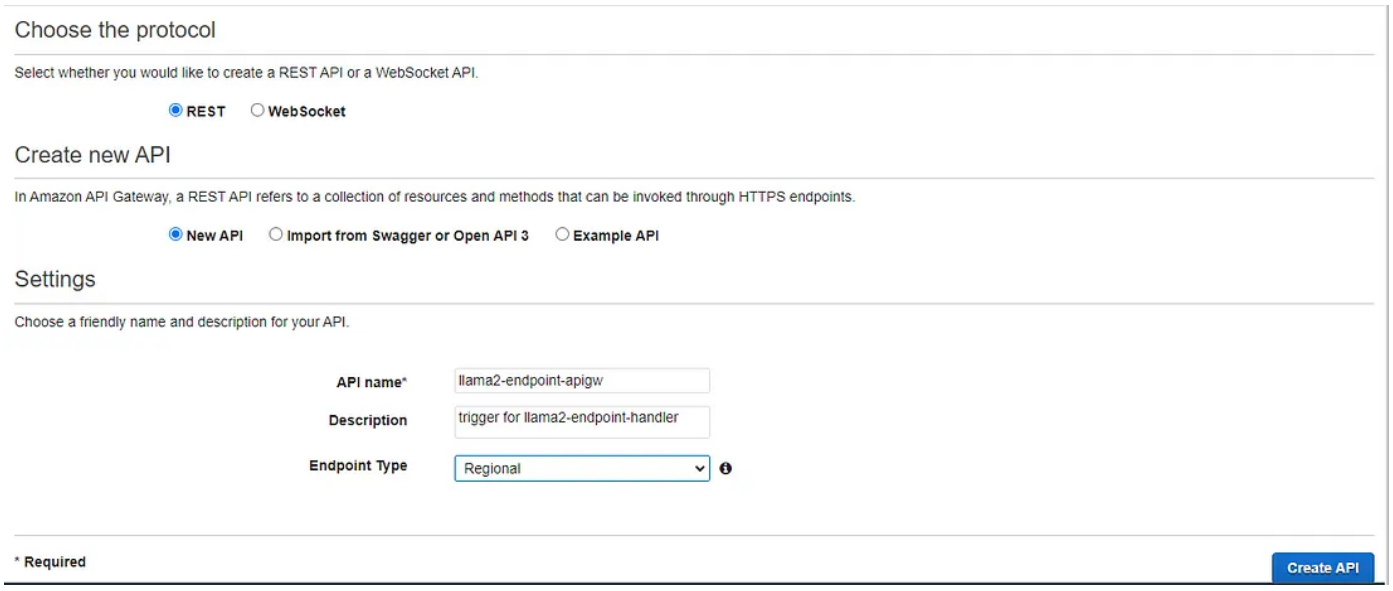
```
{
  "statusCode": 200,
  "body": " The answer to 2 + 2 is 4."
}
```

Summary

Code SHA-256	EPZKNu7yXmU5ux8J0l7D+w7GIQTRTEPFazsOF46wdW4=
Execution time	54 seconds ago (August 25, 2023 at 02:53 PM GMT+5)
Request ID	2fe915be-79f6-4e3f-b0f8-eee890f12178
Function version	$LATEST

Rest API with API Gateway

Go to API gateway, From APIs > Rest API > Build > New API > Create API


Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

[✓ REST] WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Import from Swagger or Open API 3 Example API

Settings

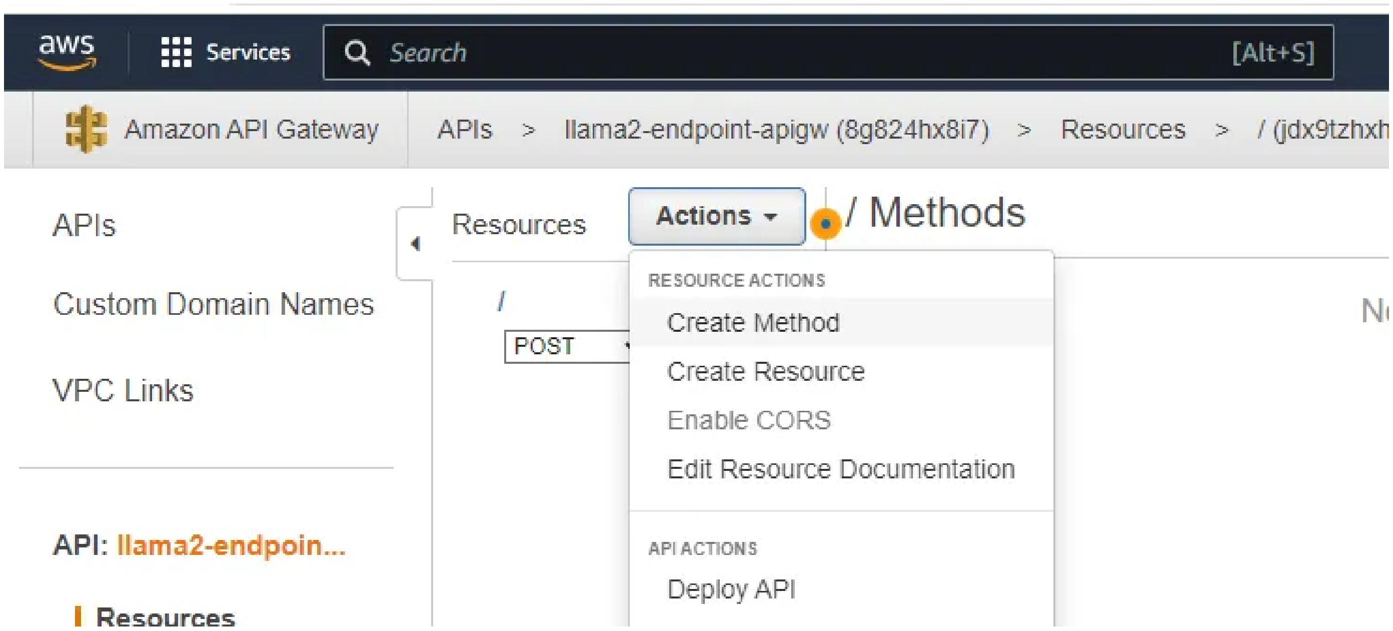
Choose a friendly name and description for your API.

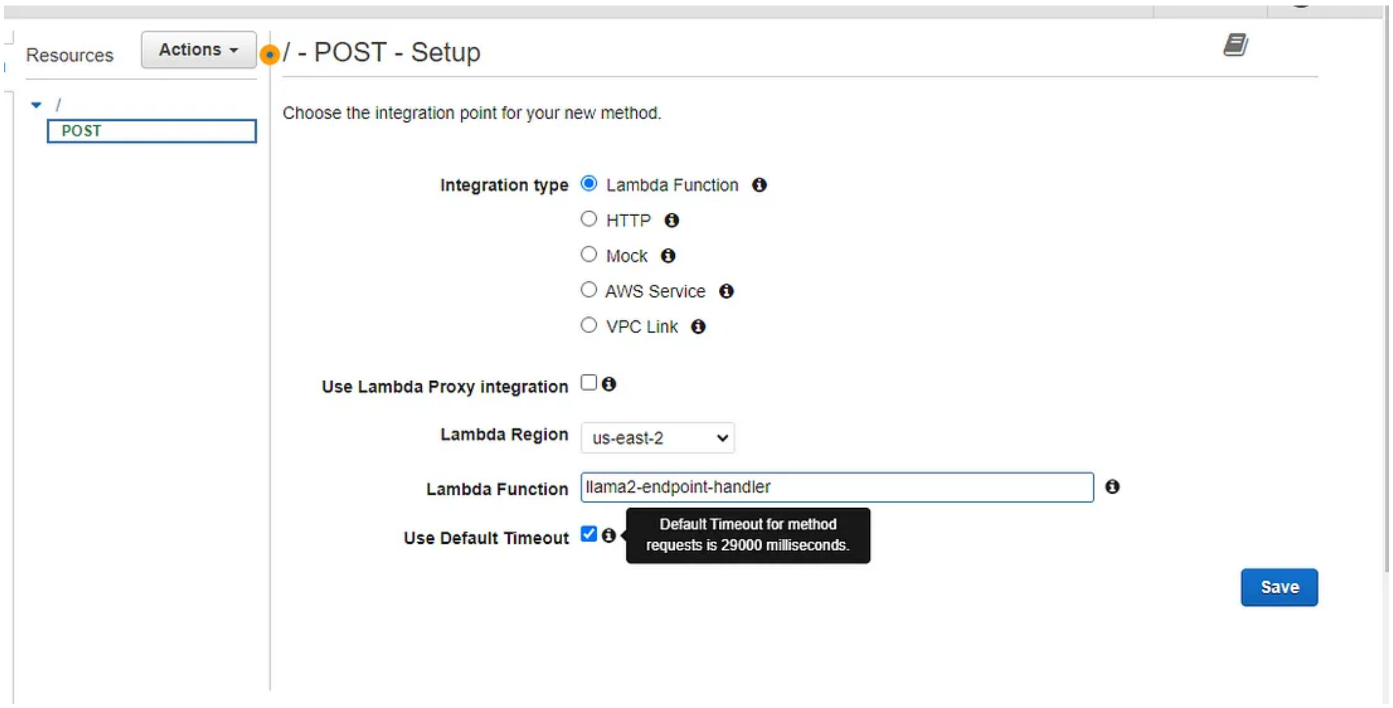
API name*	llama2-endpoint-apigw
Description	trigger for llama2-endpoint-handler
Endpoint Type	Regional

* Required

API Gateway Rest API creation console

Go to Actions > Create Method > Post





/ - POST - Setup

Choose the integration point for your new method.

Integration type: [✓ Lambda Function] HTTP Mock AWS Service VPC Link

Use Lambda Proxy integration: ☐

Lambda Region: us-east-2

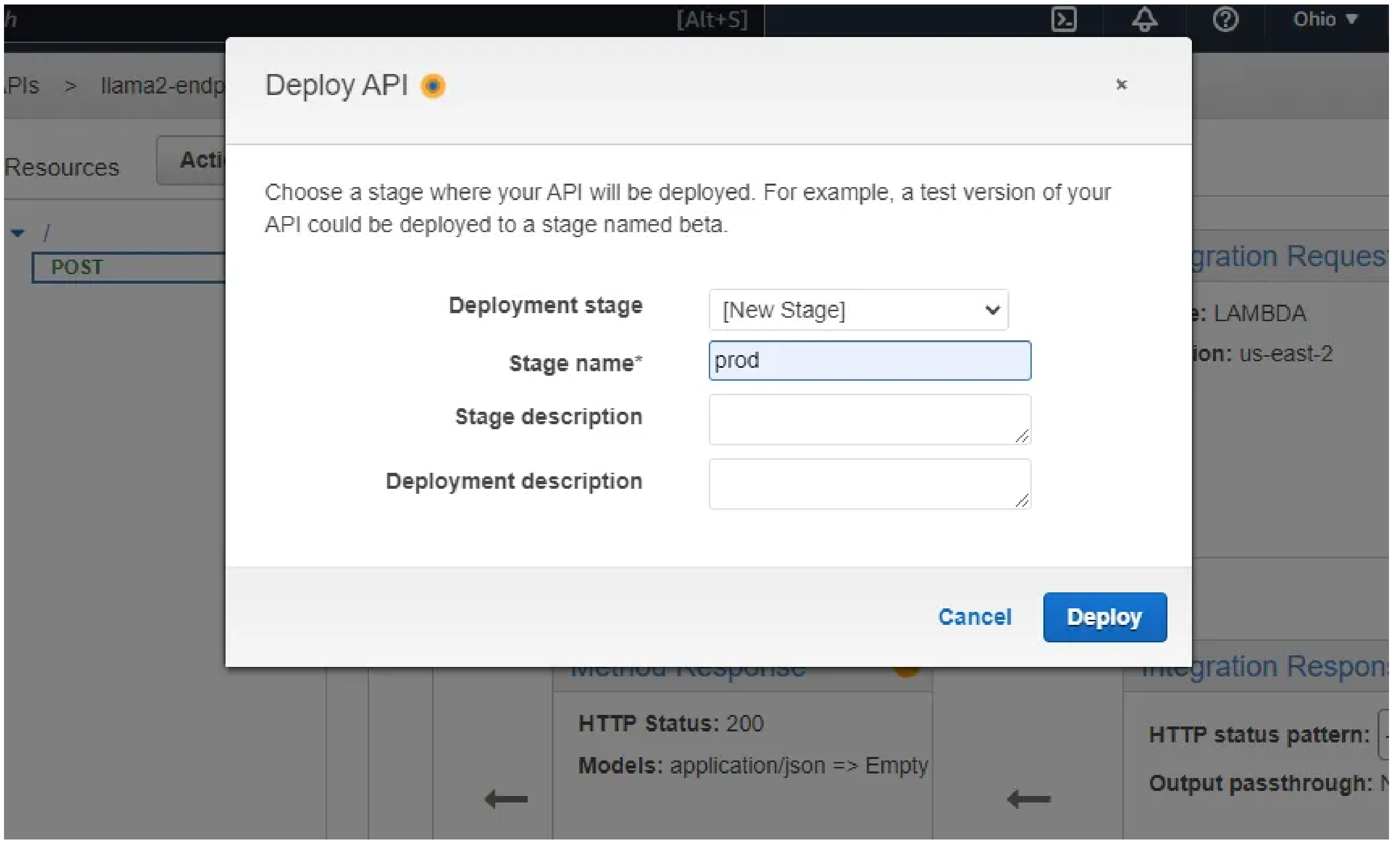
Lambda Function: llama2-endpoint-handler

Use Default Timeout: ✓

Default Timeout for method requests is 29000 milliseconds.

Click Save

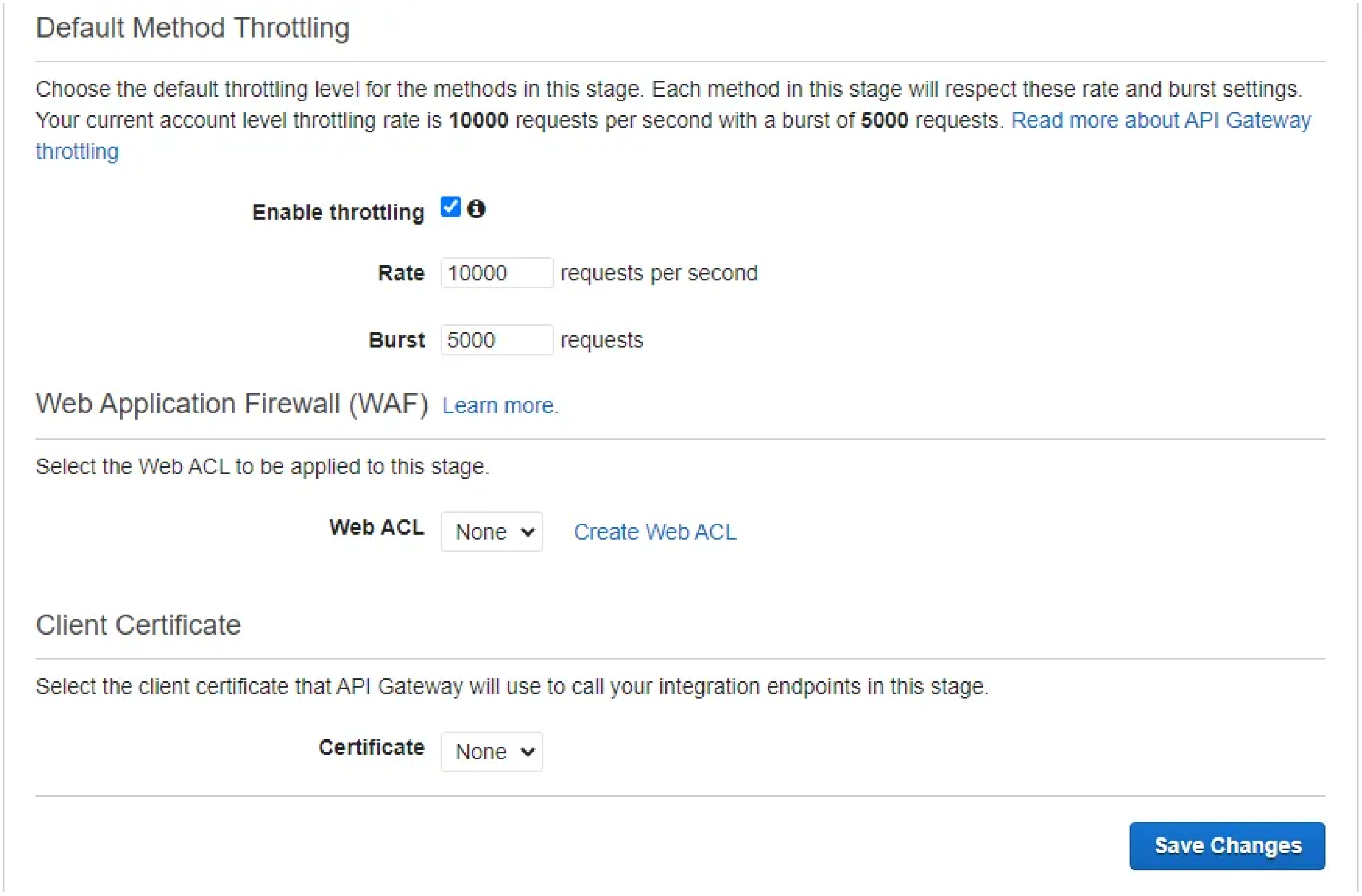
Finally go to Actions > API Actions > Deploy API



Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

Deployment stage	[New Stage]
Stage name*	prod
Stage description	
Deployment description	

deploying with a stage



Default Method Throttling

Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is **10000** requests per second with a burst of **5000** requests. Read more about API Gateway throttling

Enable throttling	✓
Rate	10000 requests per second
Burst	5000 requests

Web Application Firewall (WAF) Learn more.

Select the Web ACL to be applied to this stage.

Web ACL	None

Client Certificate

Select the client certificate that API Gateway will use to call your integration endpoints in this stage.

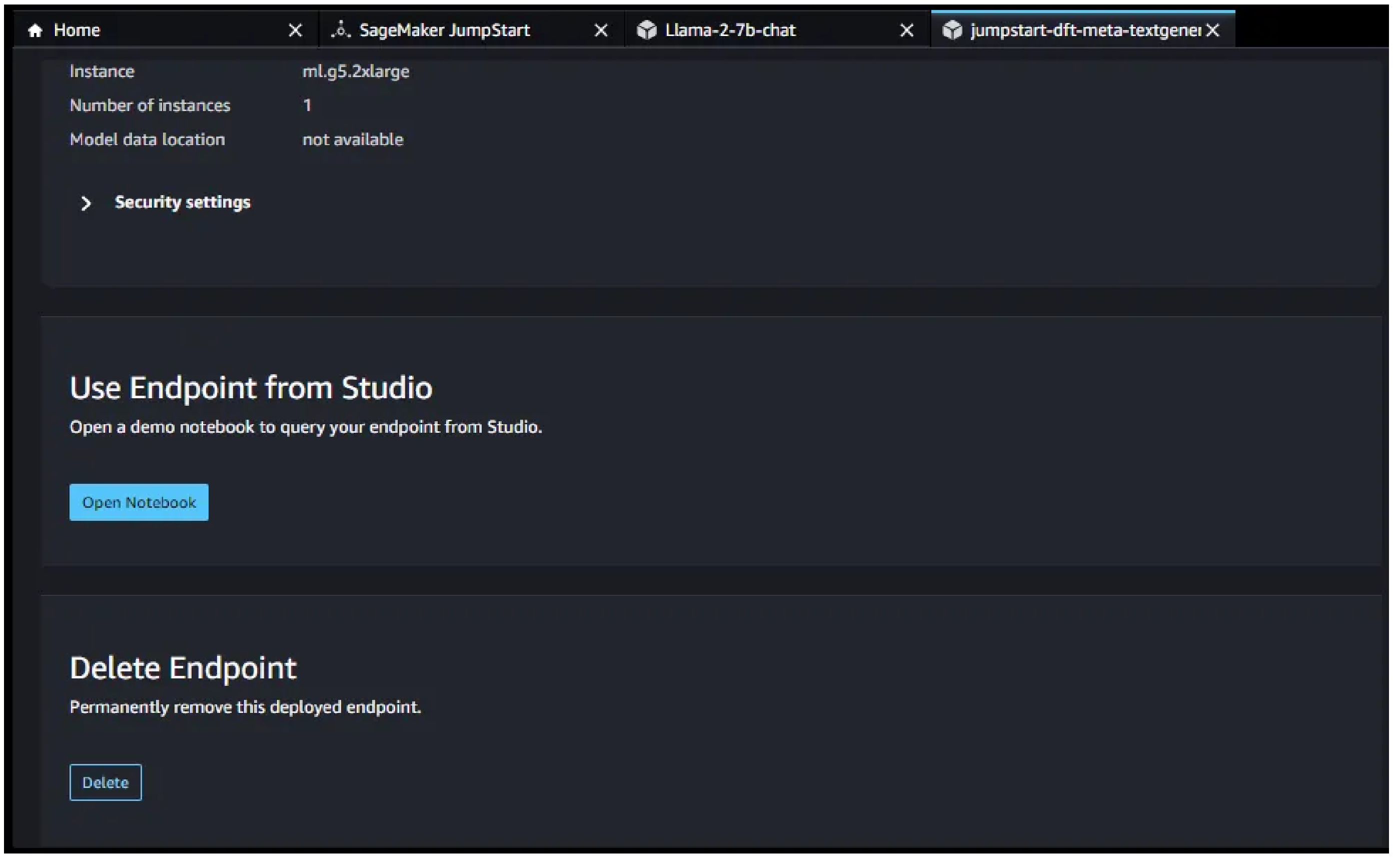
Certificate	None

stage for API deployment

Save changes, scroll up to copy the invoke URL (you can find it on you lambda function from the triggers section), and there you have it.

```
import requests

def llama_chain(query):

  api_url = 'https://n0f3c5se9l.execute-api.us-east-1.amazonaws.com/prod/' # Repl

  prompt = "You are an expert mathematician given a user query do a step by step 
  json = {"query": query, "prompt": prompt}

  r = requests.post(api_url, json = json)

  answer = r.json()["body"].strip()

  return answer

llama_chain("what is 2 + 2")
```

You can run this function to call your API gateway (the prompt field is optional in this JSON). Delete the endpoint if you are no longer using it either from the Sagemaker studio deployment page or from Sagemaker > inference > endpoints/models/endpoint configuration



Instance ml.g5.2xlarge
Number of instances 1
Model data location not available

Use Endpoint from Studio
Open a demo notebook to query your endpoint from Studio.

Delete Endpoint
Permanently remove this deployed endpoint.

Sagemaker studio llama2 deployment page

Comment out, if you face any issues. I plan to create an app on top of this API for RAG (chat with your data) using langchain and pinecone/chroma.

Thanks

Generative Ai Tools LLm Llama 2 OpenAI Sagemaker

Follow

Written by Mudassir Aqeel Ahmed

7 Followers

A Solutions Architect by heart and SWE in practice, I'm exploring life, people, opportunities, and the extent of my capabilities

More from Mudassir Aqeel Ahmed

Mudassir Aqeel Ahmed

"Continuous Security Monitoring and Insights" and the importance of security in DevOps

Continuous Security Monitoring and Insights: We need to have continuous security monitoring so that we know at any point whether we're...

2 min read · Jul 28

2

Mudassir Aqeel Ahmed

Identifying Anti-Patterns and Ensuring Scrum Team Health in Agile Development

In the world of Agile development, Scrum teams (a scrum team is all the people you need to build the application not just the...

2 min read · Aug 2

1

Mudassir Aqeel Ahmed

Unveiling the Power of Software Documentation: A Guided Journey

Introduction

3 min read · Aug 3

1

See all from Mudassir Aqeel Ahmed

Recommended from Medium

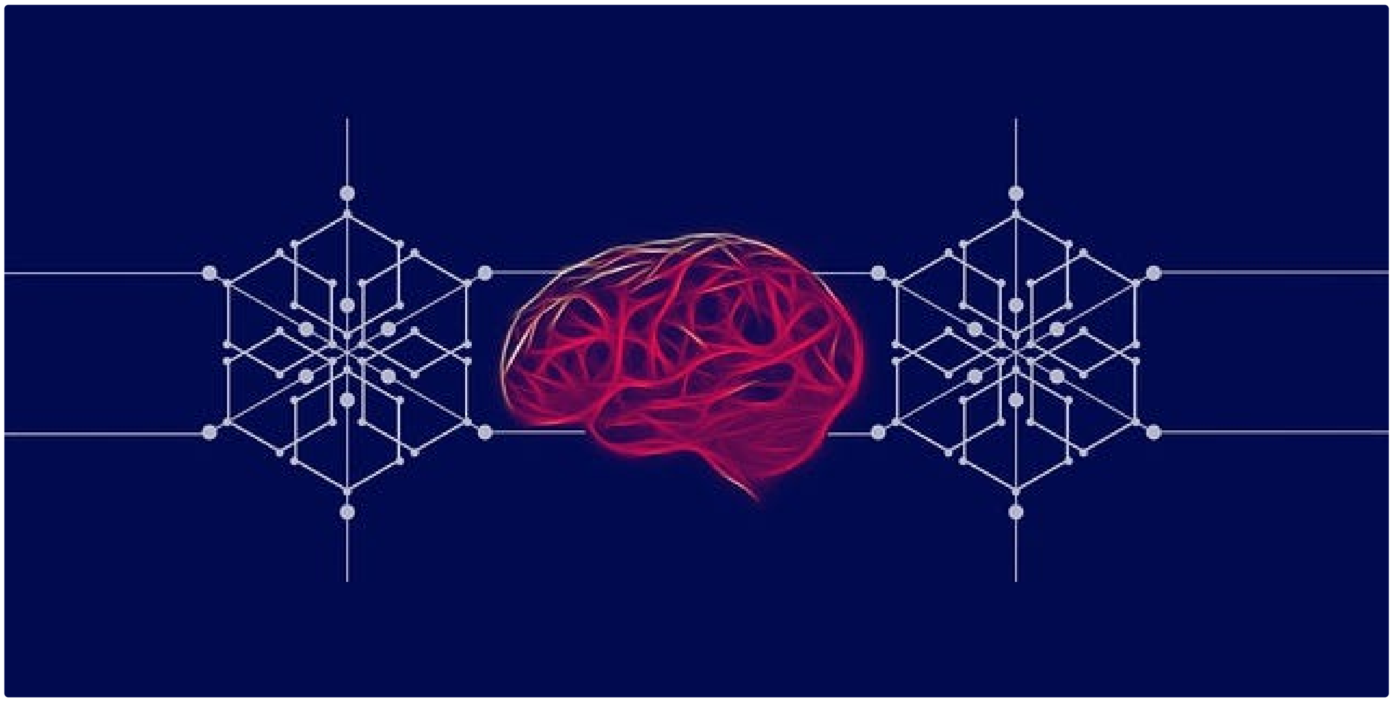
Woyera

How to host BLOOM open source LLM on AWS

A comprehensive guide on how to host one of the most popular large language models on AWS Sagemaker

5 min read · Jun 11

15 1



ai geek (wishesh)

Best Practices for Deploying Large Language Models (LLMs) in Production

Large Language Models (LLMs) have revolutionized the field of natural language processing and understanding, enabling a wide range of AI…

10 min read · Jun 26

35 1

Lists

Now in AI: Handpicked by Better Programming

266 stories · 153 saves

Natural Language Processing

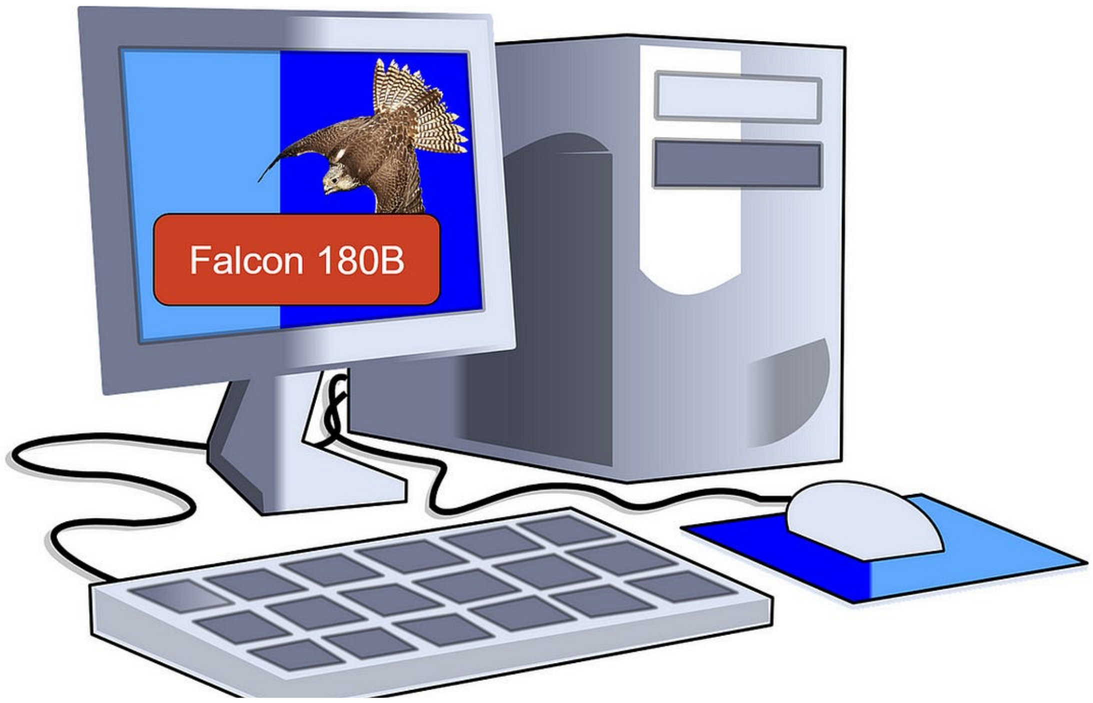
620 stories · 235 saves

AI Regulation

6 stories · 125 saves

Coding & Development

11 stories · 182 saves



Benjamin Marie in Towards Data Science

Falcon 180B: Can It Run on Your Computer?

Yes, if you have enough CPU RAM

✦ · 7 min read · Sep 12

1K 5

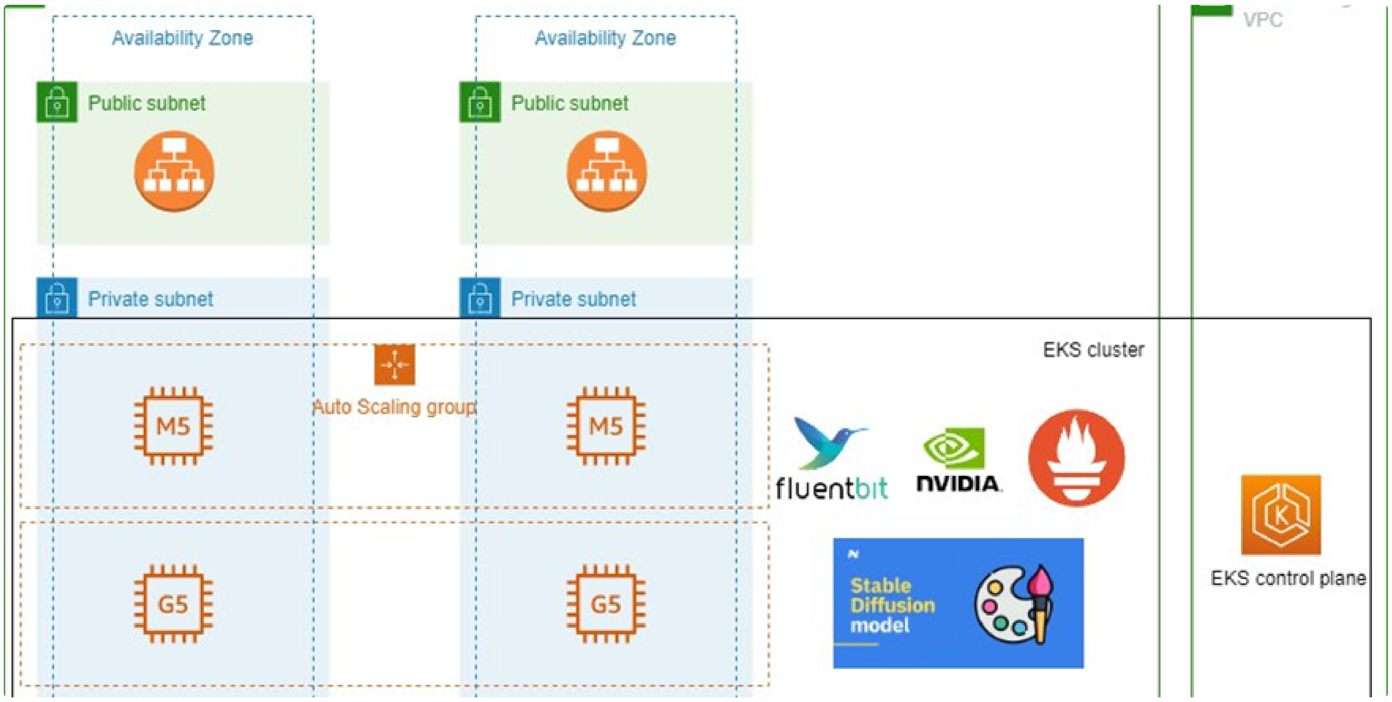
Zhi Kai Chen

Document Insights: Document QA with Langchain and ChatGPT deployed on AWS EC2

Enabling users to ask questions about their documents using the power of Generative AI

6 min read · Jul 31

3



Victor Gu in DevOps.dev

Deploy Generative AI models to Amazon EKS cost efficiently with GPU nodes auto scaling and sharing

Introduction

8 min read · May 1

2

Alissa in Python in Plain English

Run Your Own LLM-Powered Chatbot in 5 Easy Steps

🤖 = Streamlit + Langchain 🦜 + LLaMA 🦙

3 min read · Jul 21

152

See more recommendations