Github Link of the project: https://github.com/sunnysavita10/Finetuning-on-aws

Step 1: Create IAM Roles

We'll create two roles:

- SageMakerLLMRole
- LambdaInvokeLLMRole

You can write custom name

A. Create SageMaker role

a. Go to AWS Console → IAM → Roles → Create Role
b. Choose AWS Service → SageMaker
c. Attach policies:
AmazonS3FullAccess
AmazonSageMakerFullAccess
CloudWatchFullAccess

B. Create Lambda role
a. Go to AWS Console → IAM → Roles → Create Role
b. Choose AWS Service → Lambda
c. Attach policies:
AmazonSageMakerFullAccess
AmazonDynamoDBFullAccess
AmazonS3FullAccess
CloudWatchLogsFullAccess

STEP 2 — Create S3 Buckets for Dataset & Model Artifacts

1. First bucket for dataset storage
2. Second bucket for model artifacts (after training, SageMaker will save the model)

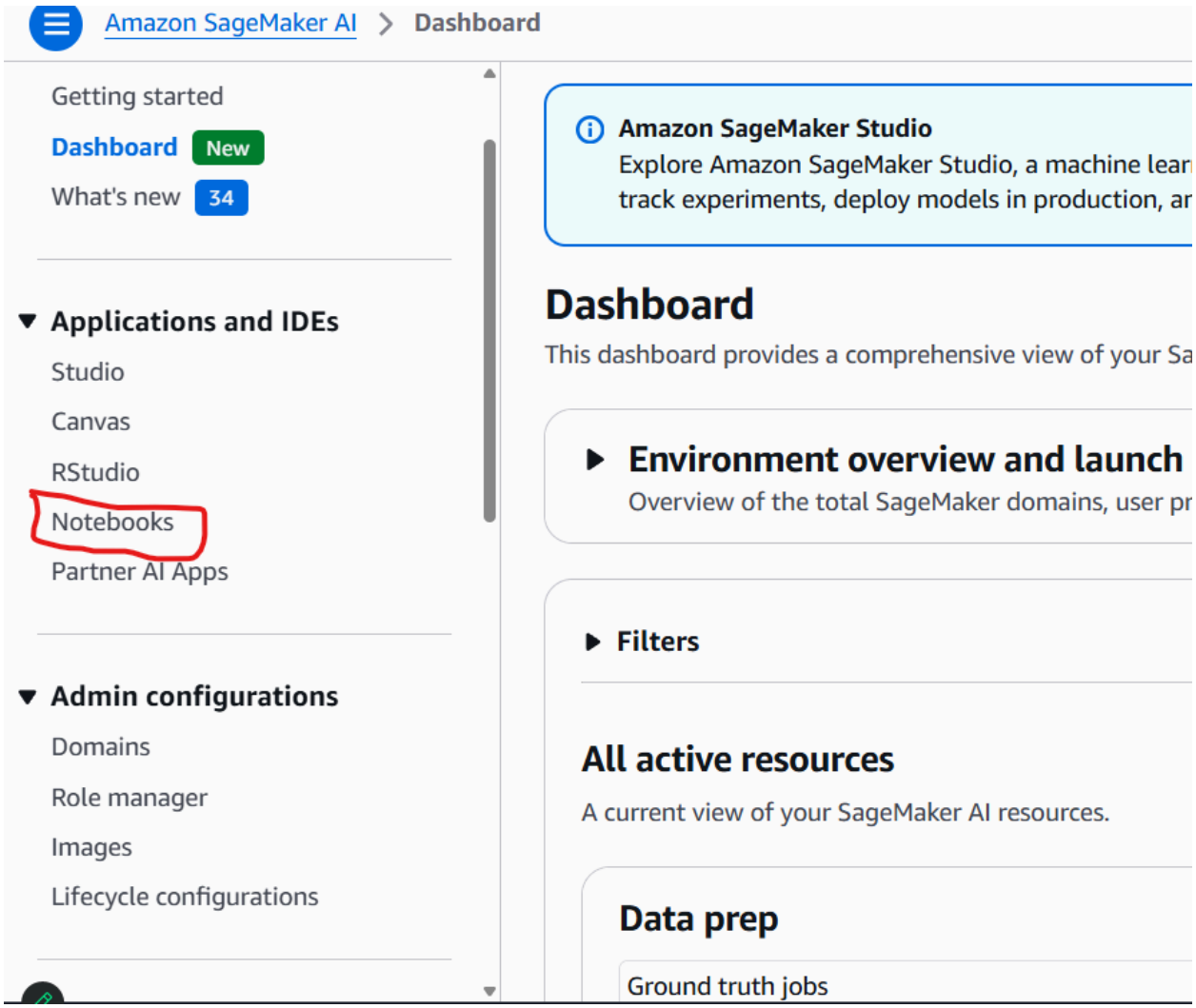
1. AWS Console → S3 → “Create bucket”
2. Bucket name: llm-finetune-dataset-<yourname>
3. Region = same as your SageMaker region (e.g., ap-south-1)
4. Create a bucket

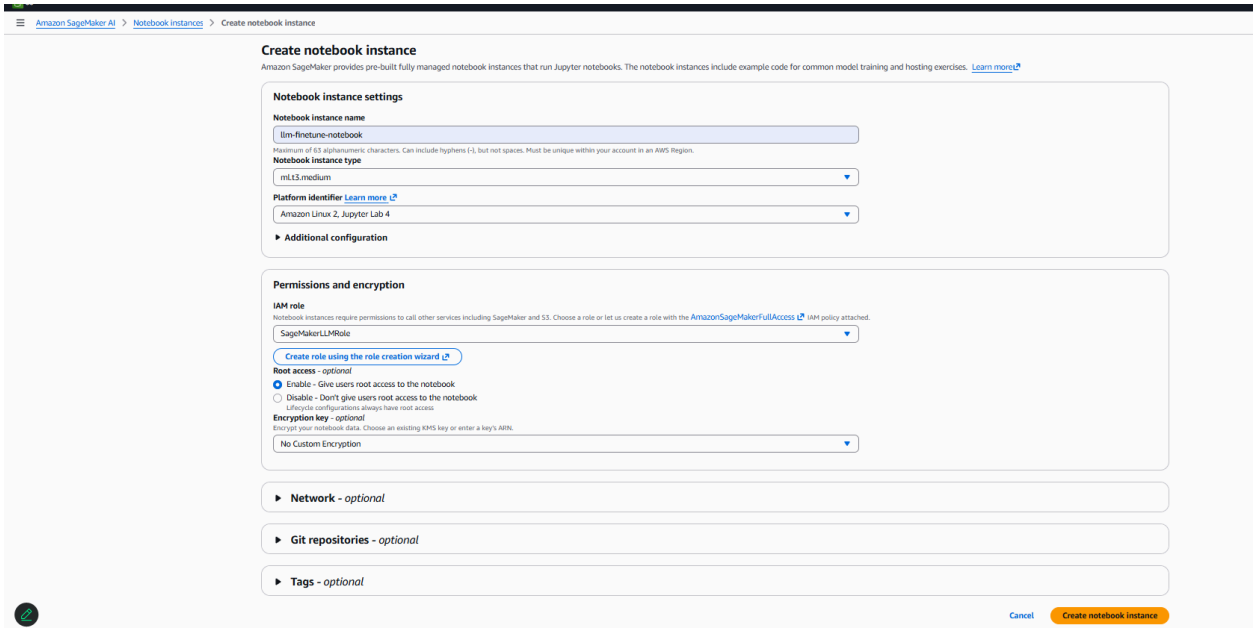
Why have we created this bucket for keeping the training files

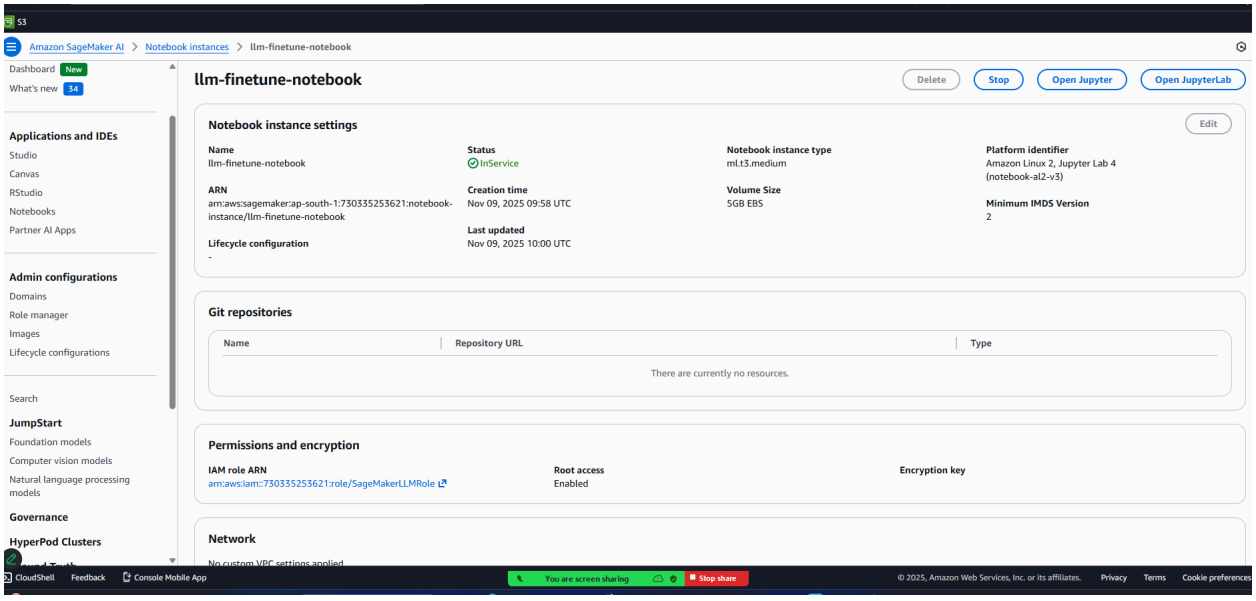
B. Create Second Bucket (for model artifacts)
1. AWS Console → S3 → “Create bucket”
2. Bucket name: llm-model-artifacts-<yourname>
3. Region = same as above
4. Create a bucket

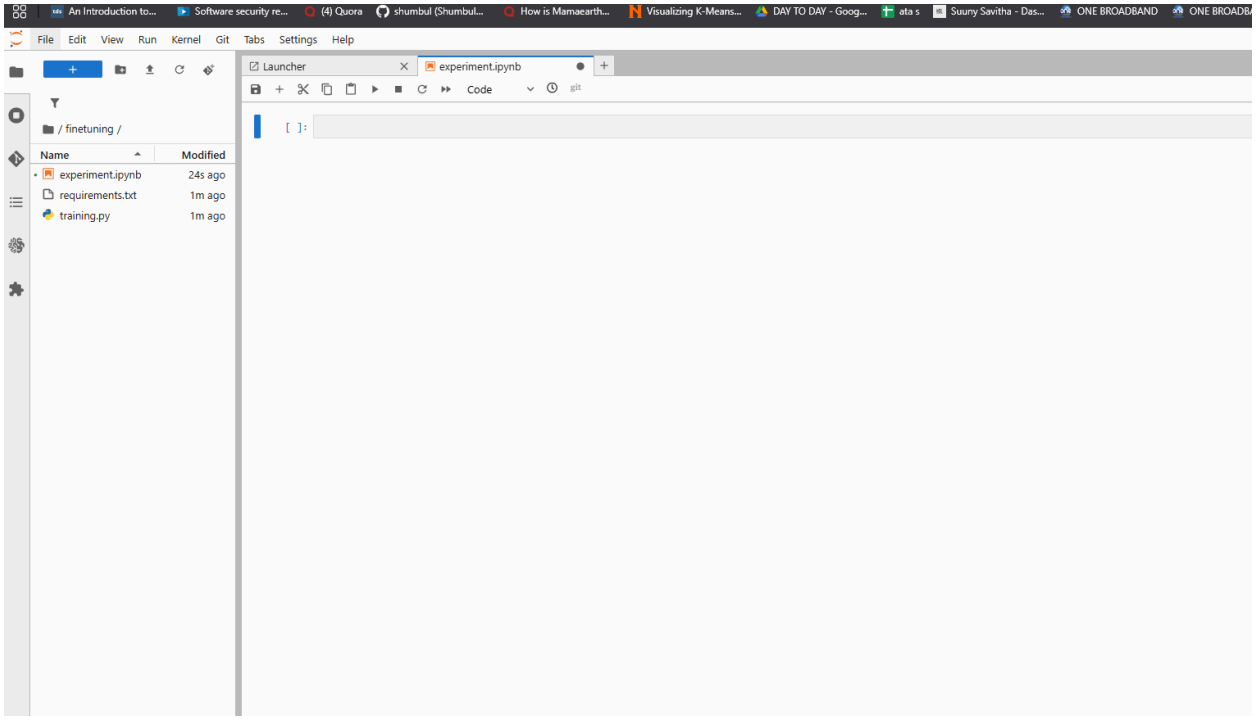
C. Verify and Note the paths
s3://llm-finetune-dataset-sunny/datasets/
s3://llm-model-artifacts-sunny/models/

STEP 3 — Create SageMaker Notebook Instance




Amazon SageMaker AI > Notebook instances > Create notebook instance
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
llm-finetune-notebook
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 4
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.
SageMakerLLMRole
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
Git repositories - optional
Tags - optional
Cancel
Create notebook instance


Amazon SageMaker AI > Notebook instances > llm-finetune-notebook
Dashboard New
What's new 34
Applications and IDEs
Studio
Canvas
RStudio
Notebooks
Partner AI Apps
Admin configurations
Domains
Role manager
Images
Lifecycle configurations
Search
JumpStart
Foundation models
Computer vision models
Natural language processing models
Governance
HyperPod Clusters
llm-finetune-notebook
Delete
Stop
Open Jupyter
Open JupyterLab
Notebook instance settings
Edit
Name
llm-finetune-notebook
Status
InService
Notebook instance type
ml.t3.medium
Platform identifier
Amazon Linux 2, Jupyter Lab 4 (notebook-al2-v3)
ARN
arn:aws:sagemaker:ap-south-1:730335253621:notebook-instance/llm-finetune-notebook
Creation time
Nov 09, 2025 09:58 UTC
Volume Size
5GB EBS
Minimum IMDS Version
2
Lifecycle configuration
-
Last updated
Nov 09, 2025 10:00 UTC
Git repositories
Name
Repository URL
Type
There are currently no resources.
Permissions and encryption
IAM role ARN
arn:aws:iam::730335253621:role/SageMakerLLMRole
Root access
Enabled
Encryption key
Network
No custom VPC settings applied
CloudShell
Feedback
Console Mobile App
You are screen sharing
Stop share
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

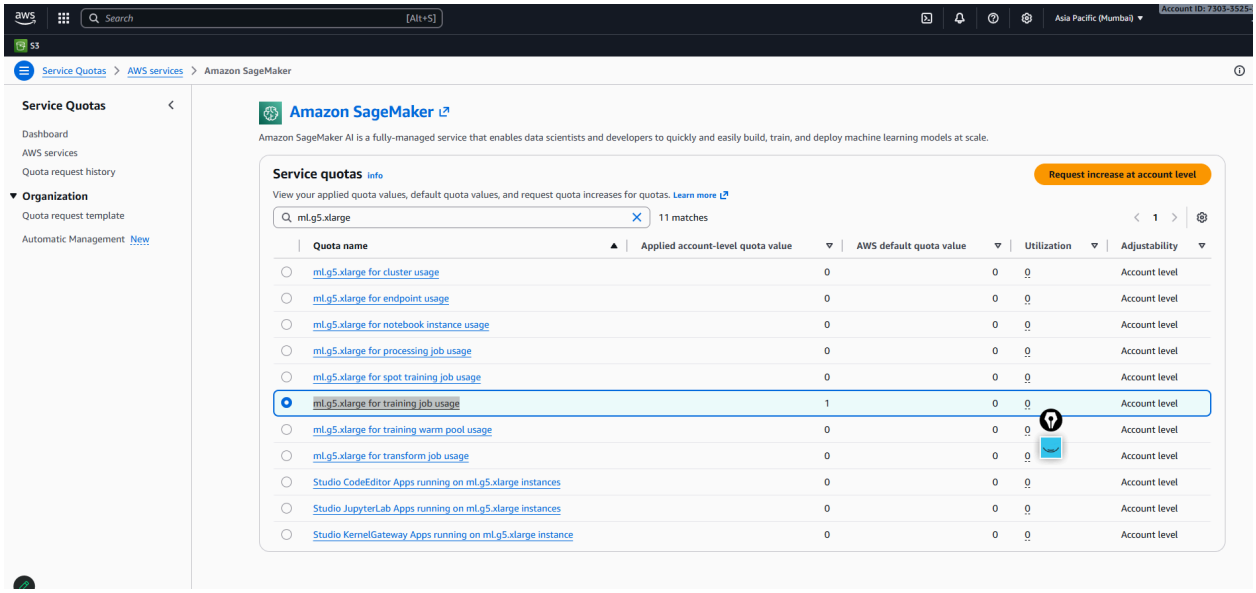


Keep the necessary requirements in the requirements.txt
Then install with the below command:
Pip install -r requirements.txt

Note: please check with your requirements.txt path

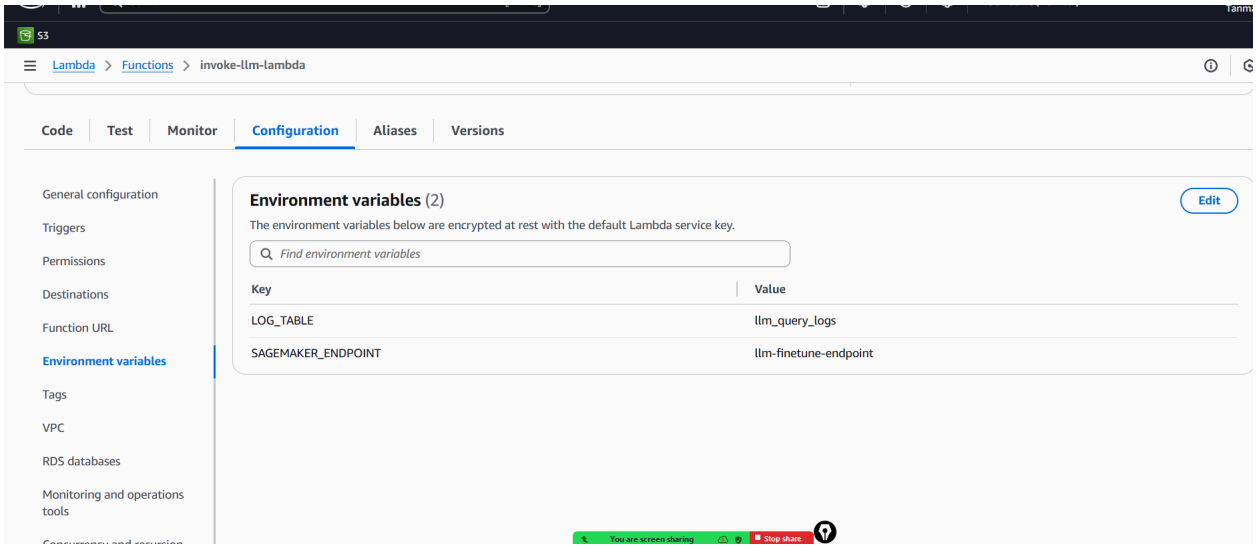
1. ls
2. pwd
3. cd /home
4. ls
5. cd ec2-user/
6. ls
7. cd SageMaker/
8. ls
9. cd Finetuning-on-aws/
10. ls
11. git status
12. Clear
13. Activate the env:
    source /home/ec2-user/anaconda3/bin/activate python3
14. pip install --upgrade pip setuptools wheel
15. pip install --upgrade cmake
16. sudo yum install -y gcc gcc-c++ make
17. pip install cmake==3.27.0 pyarrow==16.1.0 --no-cache-dir
18. pip install -r ./finetuning/requirements.txt
19. pip cache purge

Open the page below to increase the quota



Create AWS Lambda Function
Go to **Lambda → Create function**
Name: `invoke-llm-lambda`
Runtime: Python 3.10
Select Existing Role: LambdaInvokeLLMRole

Add the env variable in the lambda



Paste handler:

```
import json, os, time, boto3


runtime = boto3.client("sagemaker-runtime")
dynamo = boto3.resource("dynamodb").Table(os.environ["LOG_TABLE"])
ENDPOINT = os.environ["SAGEMAKER_ENDPOINT"]

def lambda_handler(event, context):
    body = json.loads(event.get("body", "{}"))
    text = body.get("inputs", "")

    resp = runtime.invoke_endpoint(
        EndpointName=ENDPOINT,
```

```
        ContentType="application/json",
        Body=json.dumps({"inputs": text})
    )
    result = json.loads(resp["Body"].read().decode())

    log_item = {
        "id": f"{int(time.time()*1000)}#{context.aws_request_id}",
        "prompt": text,
        "response": result,
        "timestamp": int(time.time())
    }
    dynamo.put_item(Item=log_item)

    return {
        "statusCode": 200,
        "body": json.dumps({"result": result})
    }
```

Create DynamoDB Table (for logs)

Go to DynamoDB → Create table

- Table name: llm_queries
- Partition key: id (String)
- Sort key: (optional)

Deploy-test with the sample

```
{
  "body": "{\"inputs\": \"Summarize the text about heart disease\"}"
}
```

Create API Gateway (REST)

1. Go to API Gateway → Create API → REST API → Build
    - Name: LLMInferenceAPI
2. Create resource /predict
3. Create method POST
    - Integration type: Lambda Function
    - Choose invoke-llm-lambda
4. Deploy API → stage: prod

Now create a UI for the inferencing and RAG appp Create a virtual env and install the requirementS_inference.txt

1. uv python list
2. uv venv env --python cpython-3.11.13-windows-x86_64-none
3. env/Scripts/activate
4. source env/Scripts/activate→ for bash terminal
5. uv pip install -r requirements_inference.txt

Then after creating a app will run it via these below commands

6. streamlit run inference_app.py