

Set Up Amazon SageMaker Notebook

You may install the [model.tar.gz](#) as it is too large to be put in github.

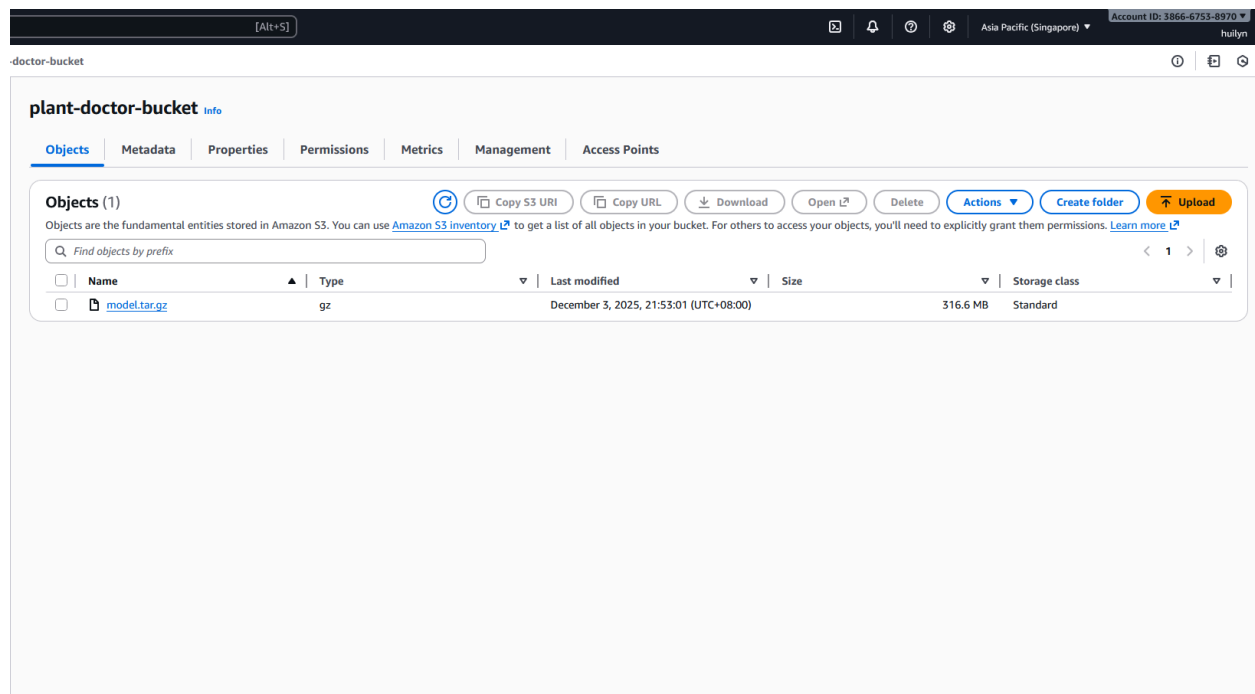
This guide will help you set up the SageMaker Inference, using the model's weights, allowing it to be called by the Plant-Doctor Backend Service.

This model is an image classification model that will predict the disease of a plant image (frog_eye_leaf_spot, multiple_diseases, powdery_mildew, rust, or scab). Will return the confidence interval as well (%).

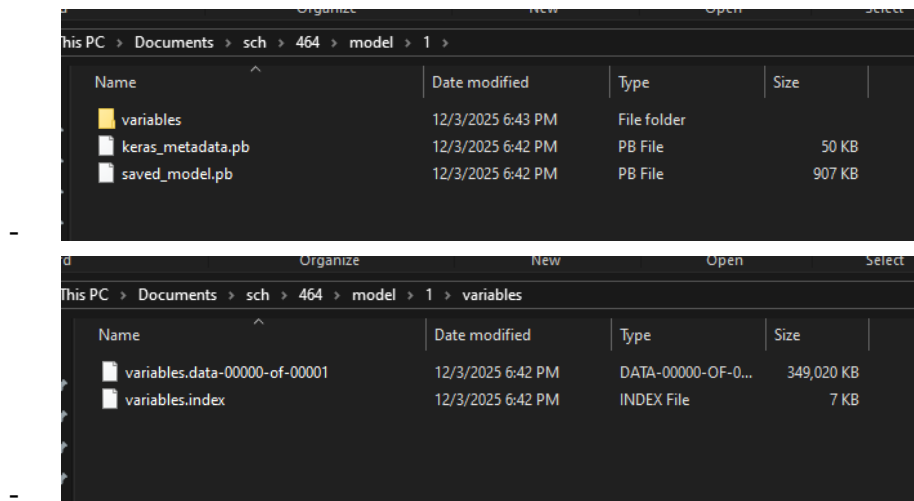
User (Mobile) → Plant Doctor Service POST /predict → SageMaker Inference (Returns the classification)

Step 1: Create an S3 Bucket (if not already existing)

This step helps you make an S3 bucket that will house the model file and weights. Sagemaker will read from this S3 bucket.



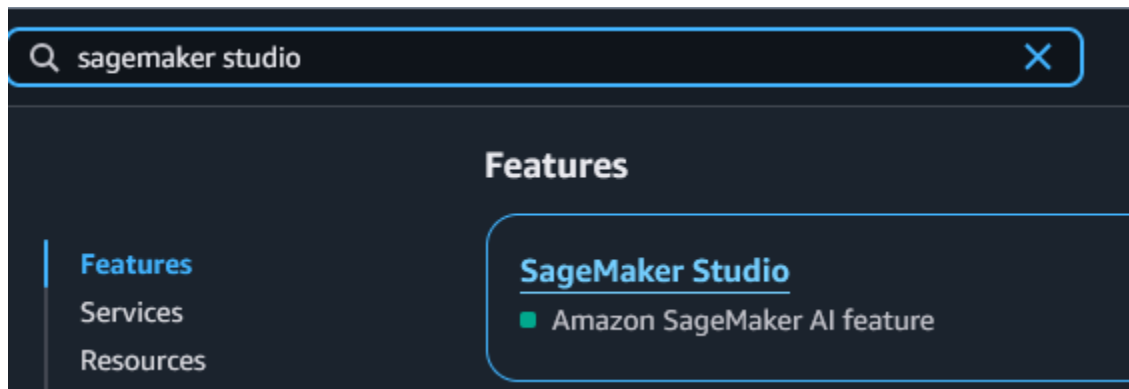
- We use plant-doctor-bucket.
- Upload [model.tar.gz](#) (from the plant-doctor-service repo) inside
 - This file should have this layout



Step 2: Launch SageMaker Notebook Instance

This step will tell you how to deploy the model so that the Plant-Doctor-Service backend can call it

Search Sagemaker Studio



Go to Notebooks from left sidebar > Create notebook instance

Amazon SageMaker AI

Dashboard
What's new 18

▼ Environment configuration

Domains
Images
Lifecycle configurations
Role manager

▼ Applications and IDEs

SageMaker Studio
Canvas
RStudio
Notebooks
Partner AI Apps

▼ Model training & customization

Training & tuning jobs
JumpStart model hub
HyperPod clusters

► Training plans

► Deployments & inference

► Model governance

► AWS Marketplace resources

Notebooks and Git repos

▼ Try the new JupyterLab in SageMaker Studio



Try the new JupyterLab in SageMaker Studio

- Launch notebooks in seconds and start coding instantly
- Use the similar underlying compute and storage as your notebook instances to enable more features at the same cost
- Seamlessly perform comprehensive ML and analytics workflows, all in one notebook
- Leverage GenAI-powered coding assistance from Amazon Q Developer and JupyterAI to accelerate development
- Collaborate with your peers in real-time on the same notebook for seamless ideation

Get Started

► How to access JupyterLab in Studio?

Notebook instances

Git repositories

Notebook instances [Info](#)



Actions ▾

Create notebook instance

Search notebook instances

< 1 > ⚙

	Name	Instance	Creation time	Status	Actions
<input type="radio"/>	plant-doctor-notebook	ml.t3.medium	12/3/2025, 6:50:05 PM	🟢 InService	Open Jupyter Open JupyterLab

Can use these notebook instance settings.

The screenshot shows the 'Create notebook instance' page in the Amazon SageMaker console. The page is divided into two main sections: 'Notebook instance settings' and 'Permissions and encryption'. In the 'Notebook instance settings' section, the 'Notebook instance name' is 'plant-doctor-notebook2', the 'Notebook instance type' is 'ml.t3.medium', and the 'Platform identifier' is 'Amazon Linux 2, Jupyter Lab 4'. The 'Additional configuration' section is collapsed. In the 'Permissions and encryption' section, the 'IAM role' is 'AmazonSageMaker-ExecutionRole-20251203T184975'. Below this, there is a link to 'Create role using the role creation wizard'. The 'Root access - optional' section has 'Enable - Give users root access to the notebook' selected. The 'Encryption key - optional' section has 'No Custom Encryption' selected.

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

plant-doctor-notebook2

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.

AmazonSageMaker-ExecutionRole-20251203T184975

[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

For IAM role, select “Create a new role” and click Any S3 bucket and create role

The screenshot shows the 'Permissions and encryption' section of the Amazon SageMaker console. It displays the 'IAM role' section with the role name 'AmazonSageMaker-ExecutionRole-20251203T184975'. Below this, there are three options: 'Create a new role', 'Enter a custom role', and 'Use existing role'. The 'Create a new role' option is selected, and a button labeled 'Create a new role' is visible. Below the 'Use existing role' option, the role name 'AmazonSageMaker-ExecutionRole-20251203T184975' is listed, and a note states 'Lifecycle configurations always have root access'.

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-20251203T184975

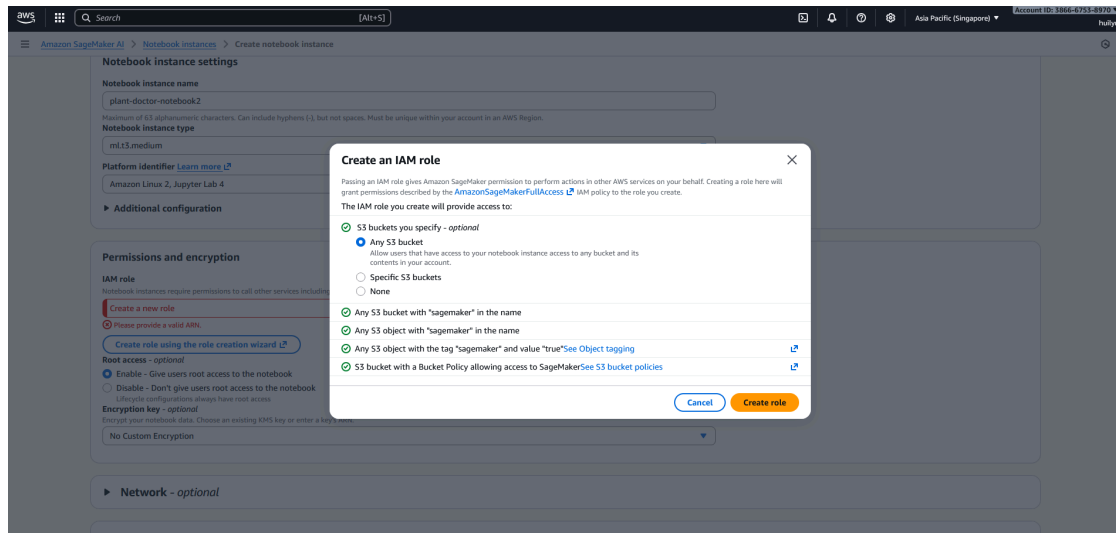
Create a new role

Enter a custom role

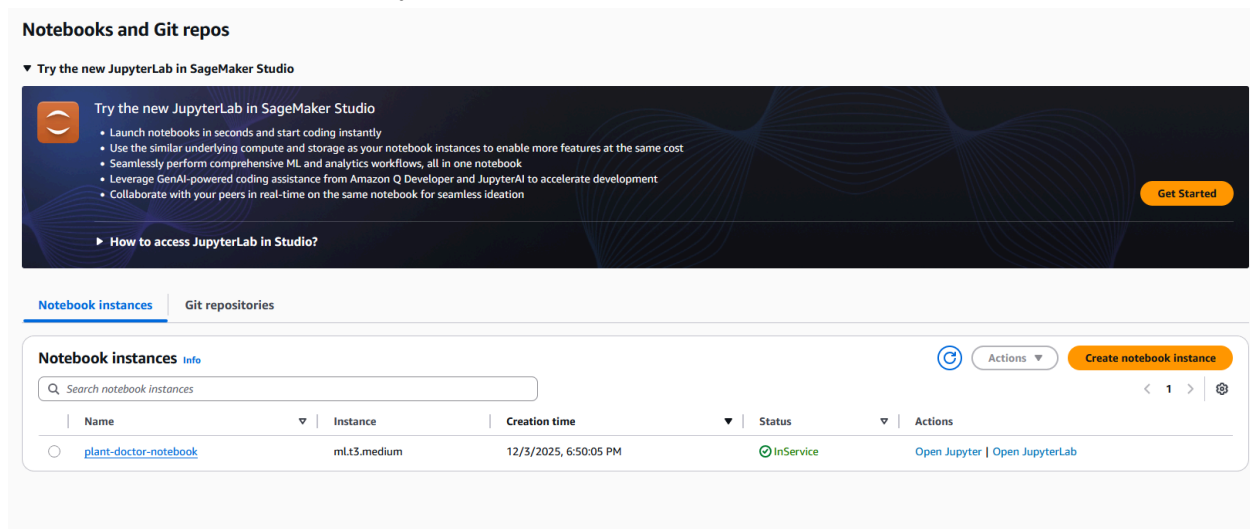
Use existing role

AmazonSageMaker-ExecutionRole-20251203T184975

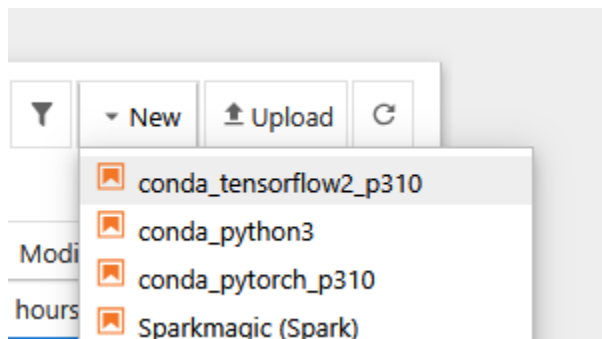
Lifecycle configurations always have root access



Once created, click “Open Jupyter”



Create a new tensorflow2 ipynb in jupyter and name it something like plant-doctor



Paste into the first cell and run:

```
%%writefile inference.py
import tensorflow as tf
import json
import numpy as np

def model_fn(model_dir):
    return tf.saved_model.load(model_dir)

def input_fn(request_body, content_type):
    data = json.loads(request_body)
    return tf.convert_to_tensor(data["instances"], dtype=tf.float32)

def predict_fn(input_data, model):
    infer = model.signatures["serving_default"]
    output = infer(input_data)
    return {k: v.numpy().tolist() for k, v in output.items()}

def output_fn(prediction, accept):
    return json.dumps(prediction)
```

This makes a [inference.py](#) in your jupyter which helps run the plant-doctor model

Paste into the second file

NOTE: replace plant-doctor-bucket with your s3 bucket name. Ours is plant-doctor-bucket, but change the name based on what u called ur s3

```
import sagemaker
from sagemaker.tensorflow import TensorFlowModel
from sagemaker import get_execution_role

# ✓ Step 1: Use Singapore region
region = "ap-southeast-1"
sagemaker_session = sagemaker.Session()

# ✓ Step 2: Correct S3 bucket and model path
model_s3_path = "s3://plant-doctor-bucket/model.tar.gz"

# ✓ Step 3: Use your execution role (must exist in ap-southeast-1)
role = get_execution_role()

# ✓ Step 4: Create and deploy the TensorFlow model
model = TensorFlowModel(
    model_data=model_s3_path,
    role=role,
    framework_version="2.11", # Match your TensorFlow version
```

```
        sagemaker_session=sagemaker_session,
    )

    predictor = model.deploy(
        initial_instance_count=1,
        instance_type="ml.t2.medium", # Cheapest instance for testing
    )

    print("✅ Deployment complete!")
    print("Endpoint name:", predictor.endpoint_name)
```

The final file should look like this

Jupyter plant-doctor Last Checkpoint: 15 hours ago

File Edit View Run Kernel Git Settings Help

conda_tensorflow2_p310

```
[3]: %%writefile inference.py
import tensorflow as tf
import json
import numpy as np

def model_fn(model_dir):
    return tf.saved_model.load(model_dir)

def input_fn(request_body, content_type):
    data = json.loads(request_body)
    return tf.convert_to_tensor(data["instances"], dtype=tf.float32)

def predict_fn(input_data, model):
    infer = model.signatures["serving_default"]
    output = infer(input_data)
    return {k: v.numpy().tolist() for k, v in output.items()}

def output_fn(prediction, accept):
    return json.dumps(prediction)

Writing inference.py

[7]: import sagemaker
from sagemaker.tensorflow import TensorFlowModel
from sagemaker import get_execution_role

# [X] Step 1: Use Singapore region
region = "ap-southeast-1"
sagemaker_session = sagemaker.Session()

# [X] Step 2: Correct S3 bucket and model path (from your screenshot)
model_s3_path = "s3://plant-doctor-bucket/model.tar.gz"

# [X] Step 3: Use your execution role (must exist in ap-southeast-1)
role = get_execution_role()

# [X] Step 4: Create and deploy the TensorFlow model
model = TensorFlowModel(
    model_data=model_s3_path,
    role=role,
    framework_version="2.11", # Match your TensorFlow version
    sagemaker_session=sagemaker_session,
)

predictor = model.deploy(
    initial_instance_count=1,
    instance_type="ml.t2.medium", # Cheapest instance for testing
)

print("[X] Deployment complete!")
print("Endpoint name:", predictor.endpoint_name)

----! [X] Deployment complete!
Endpoint name: tensorflow-inference-2025-12-03-13-53-42-096

[ ]:
```

ADDITIONAL NOTES!

For plant-doctor-service task, do check if you have the following env values

AWS_REGION=ap-southeast-1

SAGEMAKER_ENDPOINT={Endpoint generated from plant-doctor}

[i.e, tensorflow-inference-2025-12-03-13-53-42-096]

Amazon Elastic Container Service

Task definitions

plant-doctor-task

Revision 2

Containers

Express Mode

Clusters

Namespaces

Task definitions

Account settings

Amazon ECR

Repositories

AWS Batch

Documentation

Discover products

Subscriptions

Tell us what you think

plant-doctor-task:2

Last updated
December 4, 2025, 14:58 (UTC+8:00)

Deploy

Actions

Create new revision

Overview

ARN
arn:aws:ecs:ap-southeast-1:386667538970:task-definition/plant-doctor-task:2

Status
ACTIVE

Time created
December 3, 2025, 22:53 (UTC+8:00)

App environment
Fargate

Task role
plant-doctor-role

Task execution role
ecsTaskExecutionRole

Operating system/Architecture
Linux/X86_64

Network mode
awsipc

Fault injection
Turned off

Containers

JSON

Task placement

Volumes (0)

Requires attributes

Tags

Task size

Task CPU
1,024 units (1 vCPU)

Task CPU maximum allocation for containers

CPU (unit)

plant-doctor-service

Shared task CPU

Task memory
3,072 MiB (3 GB)

Task memory maximum allocation for container memory reservation

Memory (MiB)

plant-doctor-service

Shared task memory

Container: plant-doctor-service

Details

JSON

Image
386667538970.dkr.ecr.ap-southeast-1.amazonaws.com/plant-doctor-service@sha256:a5fd08d0c5de93360ea8b0b5565764143dff4e2549ac3350b7b5a69376f5e847

Private registry
Turned off

Secrets Manager ARN or name

Container: plant-doctor-service

Details | JSON

Image

386667538970.dkr.ecr.ap-southeast-1.amazonaws.com/plant-doctor-service@sha256:a5fd08d0c5de93360ea8b0b5565764143dff4e2549ac3350b7b5a69376f5e847

CPU

0

Private registry

Turned off

Memory hard/soft limit

-/-

Environment and secrets

Network settings

Security and permissions

Lifecycle and dependencies

M

Environment variables (2)

Key	Type	Value
AWS_REGION	value	ap-southeast-1
SAGEMAKER_ENDPOINT	value	tensorflow-inference-2025-12-03-13-53-42-096

Environment files (S3 ARN)

-