DATA SCIENCE ASSIGNMENT: FINAL

Subject: Building a Generative AI System

Name : Ng Hoi Yee

Objective:

Develop a generative AI system that includes models for converting text to images and text to videos. The project will also cover creating a web and mobile application, setting up a CI/CD pipeline, deploying to cloud infrastructure, and adding observability tools.

1-Data Collection:

Text-to-Image:

- Obtain and preprocess the MS COCO Dataset.

Text-to-Video:

- Acquire and preprocess the TDIUC Dataset.

MS COCO (Microsoft Common Objects in Context) dataset

Dataset can be downloaded from:

https://cocodataset.org/#download

There are train2017, val2017, test2017, Unlabeled image

TDIUC (Task Driven Image Understanding Challenge) Dataset

Dataset can be downloaded from:

https://kushalkafle.com/projects/tdiuc.html?utm_source=chatgpt.com#download

There are Q&A JSON files for train/val, Script(s) to fetch images via API or existing sources

2- Model Development:

Text-to-Image Model:

- Use Stable Diffusion or similar model.

- Implement code to generate images from text.

Text-to-Video Model:

- Use CogVideo or similar model (note that text-to-video is an emerging field).

- Implement code to generate videos from text.

Text-to-Image part - it's easier, faster, and gets results quickly.

Goal for Text-to-Image Model:

Use Stable Diffusion (or similar) to generate images from text.

Set Up Stable Diffusion

Install via Hugging Face:

pip install huggingface_hub==0.23.1

Confirm Version in Python

import huggingface_hub

print(huggingface_hub.__version__)

```
In [1]: import huggingface_hub
        print(huggingface_hub.__version__)

        0.23.1
```

Use StableDiffusionPipeline to generate images from COCO prompts

pip install diffusers transformers accelerate scipy safetensors

---

from diffusers import StableDiffusionPipeline

import torch

Load model

pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5", torch_dtype=torch.float16)

pipe = pipe.to("cuda") # If using GPU

Generate image

prompt = "a futuristic city at sunset"

image = pipe(prompt).images[0]

image.save("output.png")

Note: this part took some waiting time

Prepare Your Input Text Prompts

Load from COCO Captions JSON

```
import json
with open('C:/Users/Hoi Yee/Downloads/Generative_AI_System/annotations_trainval2017/
annotations/captions_train2017.json',"r") as f:
    data = json.load(f)

# Extract 100 prompts from the captions

prompts = [ann["caption"] for ann in data["annotations"][:100]]
```

I've now loaded **100 real-world captions** from MS COCO

Generate Images from Captions

used those 100 prompts to generate and save images

To test with just 5 prompts first:

```
prompts = [ann[“caption”] for ann in data[“annotations”][:5]]
```

Done:

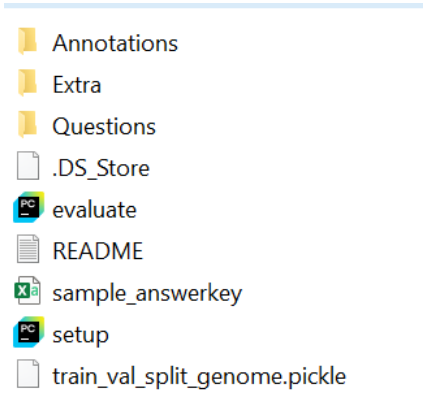
Loaded 5 captions from the MS COCO dataset
Used StableDiffusionPipeline (CPU) to generate images from those captions
Saved the generated images locally to the generated_images_cpu folder

---

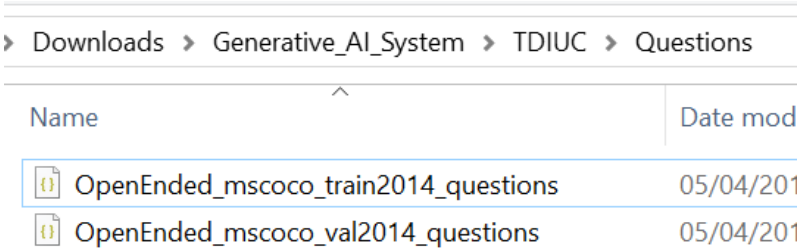
5 generated images:

I proceeded with text-to-video generation next.

TDIUC dataset includes:



Extract Prompts (from Questions):



Extract 5 Prompts:

```
# Extract the first 5 questions
video_prompts = [item["question"] for item in data["questions"][:5]]
```

The 5 Prompts:

1. What color handle does the middle man's racquet have?
2. Are there any couches in the photo?
3. What color is the bed sheets?
4. What color is in the bike?
5. Are there any snowboards in the photo?

For better video generation**,** I rephrased them into descriptive prompts (not questions), so the AI has a clearer visual scene to imagine.

The 5 Prompts rewritten (to be video-fiendly):

1. A man holding a tennis racquet with a colorful handle.
2. A living room with two modern couches and a coffee table.
3. A bedroom with white and blue bed sheets on the bed.
4. A red bicycle parked on a sunny city street.
5. A group of people snowboarding down a snowy mountain slope.

I used the text-to-video model Stable Video Diffusion.

To run a simple Stable Video Diffusion example on CPU

This created the frames of the video one by one. After generation finishes, the video will be saved as mp4 format.

3-Web Application Development:
 Backend (Flask):

- Create endpoints to handle text-to-image and text-to-video generation.

 Frontend (HTML/CSS/JavaScript):

- Develop an interface for users to input text and view generated media.

 Example index.html:

- Create forms for text input and display generated images or videos.

Save or load Model in Jupyter Notebook

My model takes long to load (typical for Stable Diffusion), so I saved the loaded model to disk after first load, so my Flask backend can load the model quickly on startup instead of re-initializing each time.

Flask backend can load the model quickly on startup instead of re-initializing each time.

```
: from diffusers import StableDiffusionPipeline
  import torch

  # Load model (example for Stable Diffusion)
  model_id = "runwayml/stable-diffusion-v1-5"
  pipe = StableDiffusionPipeline.from_pretrained(model_id, torch_dtype=torch.float16)
  pipe = pipe.to("cpu")  # or "cpu" if no GPU

  # Save locally (optional)
  pipe.save_pretrained("./models/stable-diffusion-v1-5")
```

```
Loading pipeline components...: 100% 7/7 [01:01<00:00, 6.48s/it]
```

Web Application Development (Backend + Frontend)

After saving the model locally, I opened my VS code and create app.py for Flask

I also created requirements.txt with:
flask
torch
diffusers
transformers

Inside VS terminal:
pip install -r requirements.txt

To create a simple homepage or frontend page:

Create a folder named templates with index.html file

app.py will import render_template and add a route for /:

Run Flask app:

python app.py
It shows running at http://127.0.0.1:5000 (default).

Open browser http://127.0.0.1:5000

Text-to-Image

Enter prompt Generate

Create a video by generating several images from the prompt, then combining into a video:

Model used: Stable Video Diffusion (SVD-XT)

SVD-XT: An image-to-video latent diffusion model

It takes a single image and generates a short video (25 frames, ~2–4 seconds) using an image encoder. SVD-XT is open, supported, and straightforward. It can be downloaded using Hugging Face's diffusers library.

1. Prep your environment

Create a fresh .venv inside my folder:

python -m venv .venv

2. Activate it

.venv\Scripts\activate

3. Install all needed packages for Flask + text-to-video

pip install --upgrade pip

Install dependencies

pip install flask diffusers[torch] transformers accelerate safetensors imageio imageio-ffmpeg

cerspense/zeroscope_v2_576w model

Download the cerspense/zeroscope_v2_576w model from Hugging Face. It is a text-to-video diffusion model hosted on Hugging Face, designed to generate videos.

app.py is integrated that loads both your text-to-image (Stable Diffusion) and text-to-video

(Stable Video Diffusion / Zeroscope) pipelines in the same Flask app.

run app.py

```
(.venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python app.py
Loading text-to-image model...
Loading pipeline components...: 100%|██████████| 7/7 [01:08<00:00,  9.82s/it]
Loading text-to-video model...
Loading pipeline components...:  20%|██        | 1/5 [00:06<00:26,  6.52s/it]A
n error occurred while trying to fetch C:\Users\Hoi Yee\.cache\huggingface\hub\models--cerspense--zeros
cope_v2_576w\snapshots\6963642a64dbefa93663d1ecebb4ceda2d9ecb28\unet: Error no file named diffusion_pyt
orch_model.safetensors found in directory C:\Users\Hoi Yee\.cache\huggingface\hub\models--cerspense--ze
roscope_v2_576w\snapshots\6963642a64dbefa93663d1ecebb4ceda2d9ecb28\unet.
Defaulting to unsafe serialization. Pass `allow_pickle=False` to raise an error instead.
Loading pipeline components...:  40%|████      | 2/5 [00:56<01:36, 32.26s/it]A
n error occurred while trying to fetch C:\Users\Hoi Yee\.cache\huggingface\hub\models--cerspense--zeros
cope_v2_576w\snapshots\6963642a64dbefa93663d1ecebb4ceda2d9ecb28\vae: Error no file named diffusion_pyto
rch_model.safetensors found in directory C:\Users\Hoi Yee\.cache\huggingface\hub\models--cerspense--zer
oscope_v2_576w\snapshots\6963642a64dbefa93663d1ecebb4ceda2d9ecb28\vae.
Defaulting to unsafe serialization. Pass `allow_pickle=False` to raise an error instead.
Loading pipeline components...: 100%|██████████| 5/5 [00:59<00:00, 11.95s/it]
The TextToVideoSDPipeline has been deprecated and will not receive bug fixes or feature updates after D
iffusers version 0.33.1.
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI
server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
```

Open http://127.0.0.1:5000 in browser

API running: POST to /generate_image or /generate_video with JSON {"prompt": "your text"}

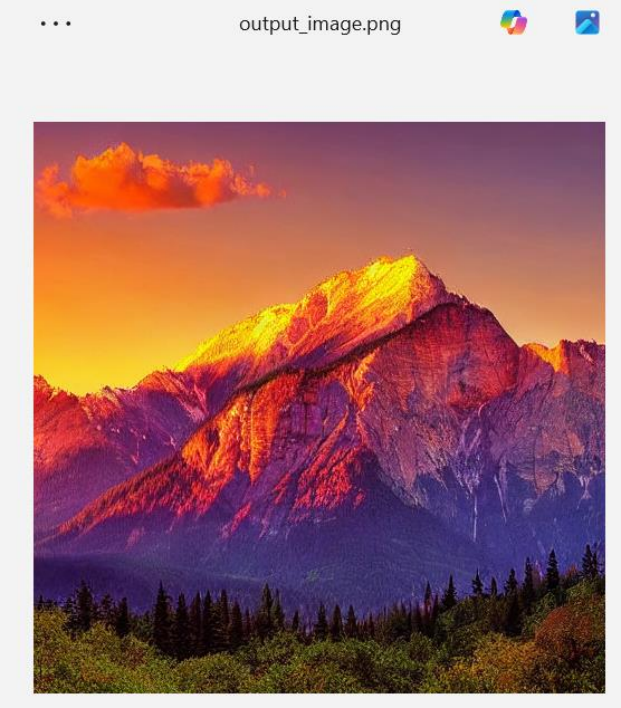
Open a new terminal window (don’t stop the Flask server):

Invoke-RestMethod -Uri http://127.0.0.1:5000/generate_image -Method POST -Headers @{ "Content-Type" = "application/json" } -Body '{"prompt": "a beautiful sunset over mountains"}'

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> & "C:\Users\Hoi Yee\Downloads\Generative_AI_S
ystem\.venv\Scripts\Activate.ps1"
(.venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> Invoke-RestMethod -Uri http://127.0.0
.1:5000/generate_image -Method POST -Headers @{ "Content-Type" = "application/json" } -Body '{"pr
ompt": "a beautiful sunset over mountains"}'

file             message
----             -------
output_image.png Image generated


(.venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

An image was generated and saved as output_image.png — that file is saved inside my project folder:

C:\Users\Hoi Yee\Downloads\Generative_AI_System\output_image.png



Instead of saving it locally, I modified the Flask app to **serve** the image via the browser. Serving the image helps to see the image instantly from the browser or download it automatically.

How it works:

Save the generated file to a static/ folder in Flask.

Return the file's URL in JSON

The frontend uses that URL in <img> or <video> tags.

1. Create a static/ folder inside my project folder Generative_AI_System/

Generative_AI_System/

│

├── app.py

├── static/

│ ├── images/

│ └── videos/

 static/images/ → for generated images

 static/videos/ → for generated videos

generate_video.py loads the Stable Video Diffusion.

1. Generates a base image from the text prompt.
2. Passes that image into SVD to create a sequence of frames.
3. Saves those frames as .mp4 in static/videos/.
4. Returns the video URL for the frontend/mobile app.

Frontend - index.html handles both text-to-image and text-to-video generation. It is placed in a templates/ folder and served through Flask.

Generative_AI_System/

│

├── app.py

├── static/

│ ├── images/

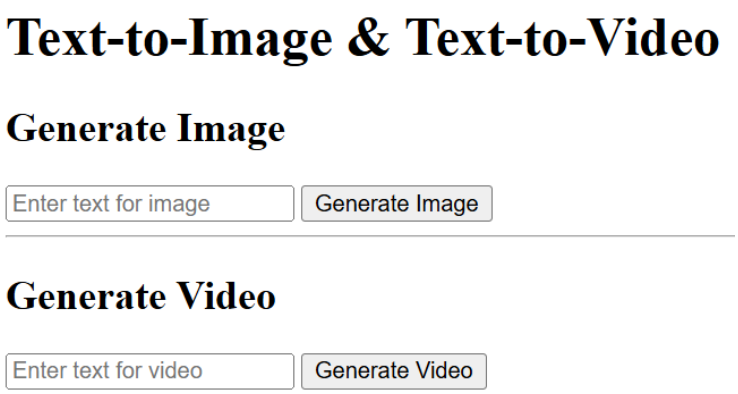
│ └── videos/

Backend + frontend setup:

- python app.py (backend with /generate_image and /generate_video, home route) → access http://127.0.0.1:5000/.
- templates/index.html (frontend form)
- static/images/ and static/videos/ (storage)

Test the web app locally - http://127.0.0.1:5000/

python app.py



I wanted to make video generation faster on CPU for smoother demos

Ways to make video generation faster on CPU:

Reduce number of frames(16–24 frames), resolution (256x256), FPS (Frames Per Second, fps=8**)**

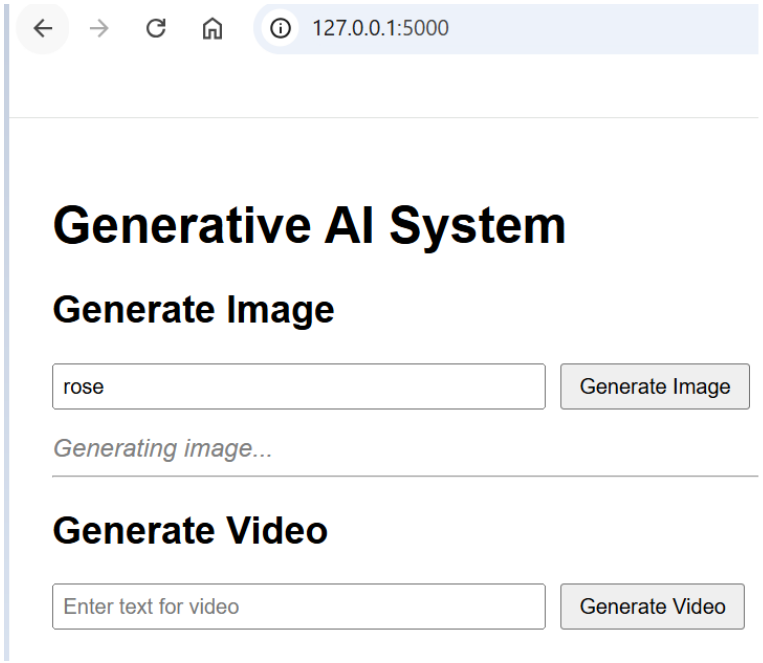
text_to_video_model.height = 256

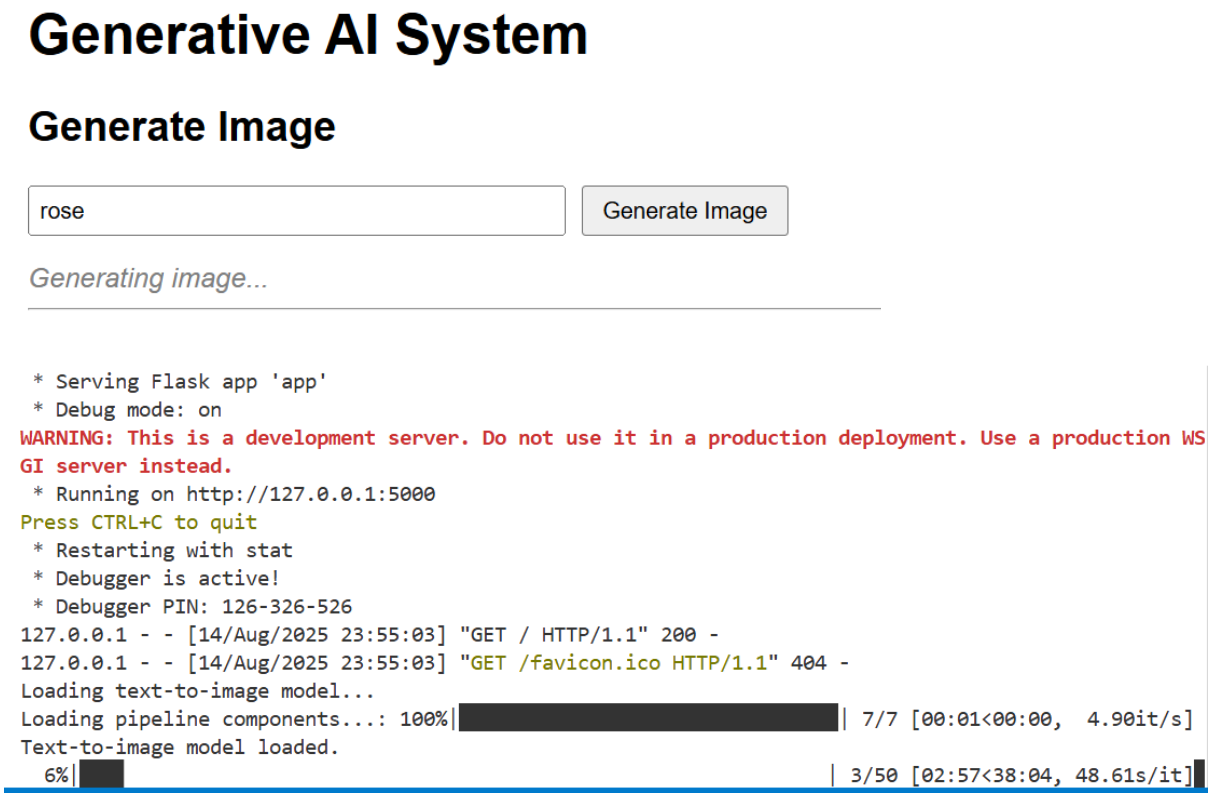
text_to_video_model.width = 256

video_frames = text_to_video_model(base_image, num_inference_steps=16).frames

imageio.mimsave(filepath, video_frames, fps=8)

This gives fast generation on CPU, still decent quality, and fewer errors/timeouts during demos.


127.0.0.1:5000
Generative AI System
Generate Image
rose
Generate Image
Generating image...
Generate Video
Enter text for video
Generate Video


Generative AI System
Generate Image
rose
Generate Image
Generating image...
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 126-326-526
127.0.0.1 - - [14/Aug/2025 23:55:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Aug/2025 23:55:03] "GET /favicon.ico HTTP/1.1" 404 -
Loading text-to-image model...
Loading pipeline components...: 100%| 7/7 [00:01<00:00, 4.90it/s]
Text-to-image model loaded.
6%| 3/50 [02:57<38:04, 48.61s/it]

```
GI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 126-326-526
127.0.0.1 - - [14/Aug/2025 23:55:03] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Aug/2025 23:55:03] "GET /favicon.ico HTTP/1.1" 404 -
Loading text-to-image model...
Loading pipeline components...: 100%|████████████████████████████| 7/7 [00:01<00:00,  4.90it/s]
Text-to-image model loaded.
100%|████████████████████████████████████████████████████████████| 50/50 [12:16<00:00, 14.72s/it]
127.0.0.1 - - [15/Aug/2025 00:08:14] "POST /generate_image HTTP/1.1" 200 -
127.0.0.1 - - [15/Aug/2025 00:08:15] "GET /static/images/20250814_235526.png HTTP/1.1" 200 -
```

Generate Image

rose | Generate Image

Generative AI System

Generate Image

spaceship | Generate Image

Generating image...

```
GI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 113-565-548
127.0.0.1 - - [17/Aug/2025 20:21:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 20:21:02] "GET /favicon.ico HTTP/1.1" 404 -
Loading text-to-image model...
Loading pipeline components...: 100%|████████████████████████████| 7/7 [00:02<00:00,  2.55it/s]
Text-to-image model loaded.
100%|████████████████████████████████████████████████████████████| 50/50 [18:35<00:00, 22.31s/it]
127.0.0.1 - - [17/Aug/2025 20:40:55] "POST /generate_image HTTP/1.1" 200 -
127.0.0.1 - - [17/Aug/2025 20:40:55] "GET /static/images/20250817_202131.png HTTP/1.1" 200 -
```

Generate Image

spaceship | Generate Image

Generate Image - took 20 min

Generated Images are saved

Downloads › Generative_AI_System › static › images


20250814_232330


20250814_235526


20250817_202131

Generate Video

jogging | Generate Video

Generating video...

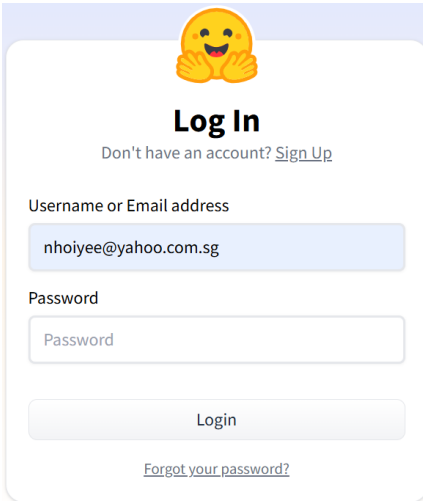
The output will be in static/videos/output.mp4.

```
lar HTTP download. For better performance, install the package with: `pip install huggingface_hub[h
_xet]` or `pip install hf_xet`

diffusion_pytorch_model.fp16.safetensors:  38%|[redacted]          | 73.4M/196M [01:09<01:43, 1.18MB/s]
model.fp16.safetensors:  12%|[redacted]                          | 157M/1.26G [01:13<09:29, 1.94MB/s]
diffusion_pytorch_model.fp16.safetensors: 100%|[redacted]| 196M/196M [02:43<00:00, 1.20MB/s]
model.fp16.safetensors: 100%|[redacted]| 1.26G/1.26G [08:23<00:00, 2.51MB/s]
diffusion_pytorch_model.fp16.safetensors: 100%|[redacted]| 3.05G/3.05G [14:07<00:00, 3.60MB/s]
Fetching 9 files: 100%|[redacted]| 9/9 [14:07<00:00, 94.22s/it]
Loading pipeline components...: 100%|[redacted]| 5/5 [00:59<00:00, 11.83s/it]
Generating video frames...16.safetensors: 100%|[redacted]| 3.05G/3.05G [14:07<00:00, 6.63MB/s]
  0%|                                                                | 0/25 [00:00<?, ?it/s]
(.venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

It was not successful. So I login

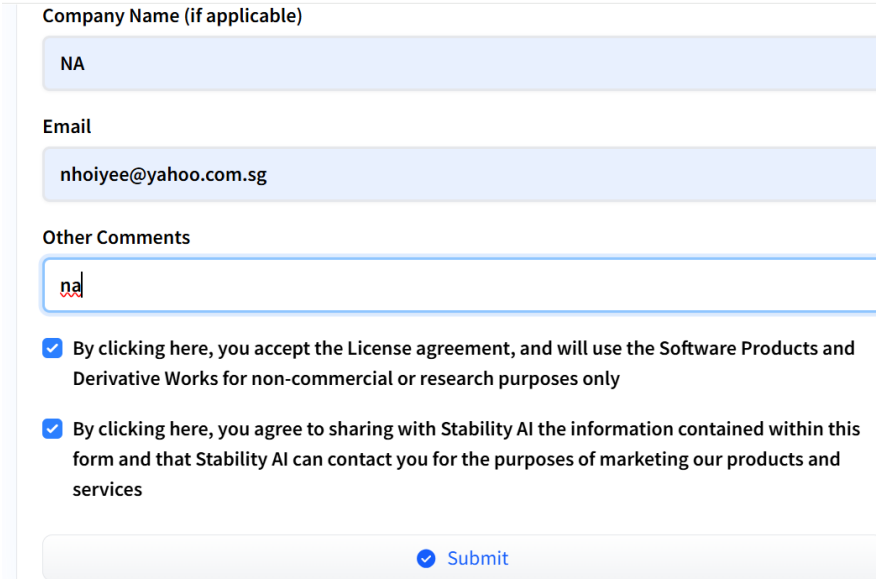
huggingface-cli login

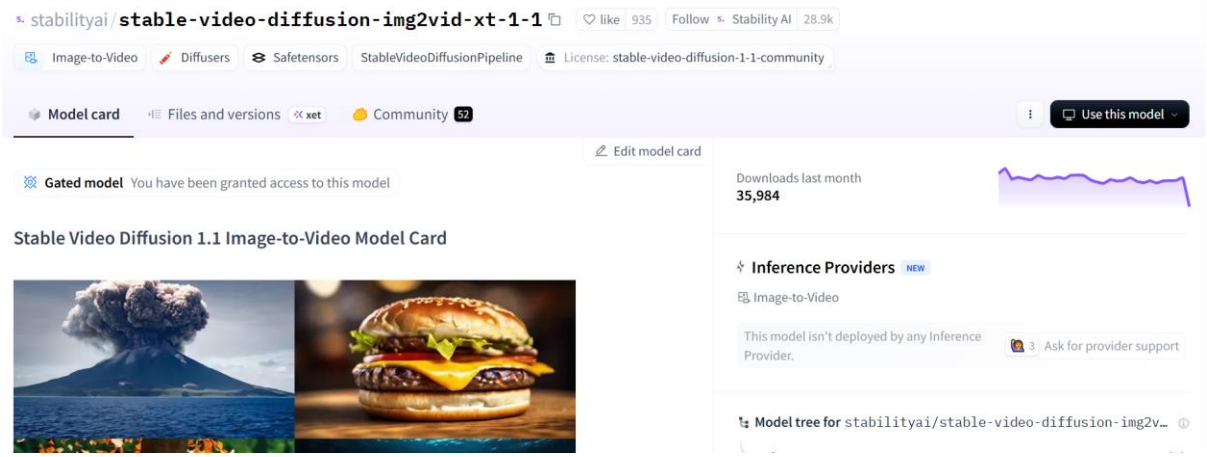
```
    _|
    _|_|_|_|  _|    _|  _|  _|_|  _|  _|_|    _|    _|  _|  _|  _|  _|_|      _|_|_|    _|_|_|_|  _|
        _|_|_|
    _|    _|  _|    _|  _|    _|  _|    _|    _|    _|    _|_|  _|    _|      _|        _|    _|  _|
        _|
    _|    _|    _|_|      _|_|_|    _|_|_|  _|_|_|  _|      _|    _|_|_|      _|        _|    _|
_|_|_|  _|_|_|_|

    To log in, `huggingface_hub` requires a token generated from https://huggingface.co/settings/tok
ens .
Token can be pasted using 'Right-Click'.
Enter your token (input will not be visible): [redacted]
```

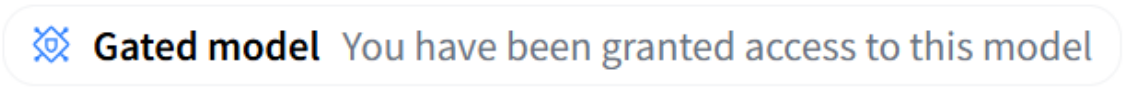


Model used:

https://huggingface.co/stabilityai/stable-video-diffusion-img2vid-xt-1-1







I needed both accepted license and proper authentication:

setx HF_HOME "C:\Users\Hoi Yee\.cache\huggingface"

setx HF_TOKEN "<YOUR_HUGGINGFACE_TOKEN>"

Modified script to use the token explicitly:

```
pipe = StableVideoDiffusionPipeline.from_pretrained(
    "stabilityai/stable-video-diffusion-img2vid-xt-1-1",
    torch_dtype=torch.float32,
```

```
    low_cpu_mem_usage=True,
    use_auth_token=os.environ.get("HF_TOKEN")  # <-- explicitly use token
)
```

Create a new virtual env:

& "C:\Users\Hoi Yee\AppData\Local\Programs\Python\Python311\python.exe" -m venv ".\venv313"

Activate the new venv:

PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> & ".\venv313\Scripts\Activate.ps1"

Install moviepy and other dependencies inside this venv:

python -m pip install --upgrade pip

python -m pip install moviepy

Install PyTorch

python -m pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu

Reinstall diffusers

python -m pip install --upgrade diffusers

Install Transformers in your venv313:

pip install torch torchvision diffusers transformers

python generate_video_cpu.py (better quality , longer videos). Different from python generate_video.py (lower frames, easier on CPU).

Text-to-Video Model is done.

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python generate_video_cpu.py
>>
Loading pipeline components...: 100%| | 5/5 [00:01<00:00,  3.11it/s]
The TextToVideoSDPipeline has been deprecated and will not receive bug fixes or feature updates aft
er Diffusers version 0.33.1.
100%| | 15/15 [34:37<00:00, 138.52s/it]
MoviePy - Building video output_video.mp4.
MoviePy - Writing video output_video.mp4

MoviePy - Done !
MoviePy - video ready output_video.mp4
Video saved to output_video.mp4
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Next step

Text-to-Image Model (Stable Diffusion): implementing the Text-to-Image **pipeline** to generate an image from text. That unlocks the ability to chain into video later.

Generate a high-quality image:

python generate_image.py --prompt "A futuristic cityscape at sunset" --output city.png

```
config.json: 100%| | 743/743 [00:00<00:00, 579kB/s]
config.json: 100%| | 547/547 [00:00<?, ?B/s]
vocab.json: 1.06MB [00:00, 5.97MB/s]                                | 0.00/806 [00:00<?, ?B/s]
vocab.json: 0.00B [00:00, ?B/s]
model.safetensors:   6%|                                          | 31.5M/492M [00:11<03:21, 2.29MB/s]
model.safetensors:   3%|                                         | 31.5M/1.22G [00:14<10:42, 1.84MB/s]
diffusion_pytorch_model.safetensors:   7%|                       | 231M/3.44G [00:13<03:09, 17.0MB/s]
diffusion_pytorch_model.safetensors: 100%| | 335M/335M [01:39<00:00, 3.35MB/s]
model.safetensors: 100%| | 492M/492M [01:51<00:00, 4.43MB/s]
model.safetensors: 100%| | 1.22G/1.22G [03:13<00:00, 6.28MB/s]
diffusion_pytorch_model.safetensors: 100%| | 3.44G/3.44G [03:58<00:00, 14.4MB/s]
Fetching 15 files: 100%| | 15/15 [04:02<00:00, 16.17s/it]
Loading pipeline components...: 100%| | 7/7 [00:02<00:00,  3.31it/s]
📥 Generating image at 512x512 with 25 steps...                  | 3.44G/3.44G [03:58<00:00, 24.4MB/s]
100%| | 25/25 [05:33<00:00, 13.34s/it]
✅ Image saved to city_1756122902.png
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Next, **chain generate_image.py + generate_video.py** to get:

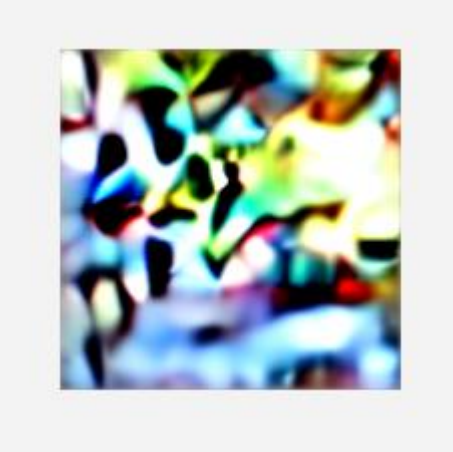
text → image → video pipeline

generate_media.py

python generate_media.py --prompt "A dragon flying over a mountain" --fast

```
model.safetensors: 100%|| 1.36G/1.36G [02:23<00:00, 9.49MB/s]
model.safetensors: 100%|| 1.22G/1.22G [03:15<00:00, 6.21MB/s]
diffusion_pytorch_model.safetensors: 100%|| 3.46G/3.46G [04:28<00:00, 12.9MB/s]
Fetching 15 files: 100%|| 15/15 [04:31<00:00, 18.13s/it]
Loading pipeline components...: 100%|| 7/7 [00:01<00:00,  4.94it/s]
✅ Image pipeline ready.safetensors:  51%|| 1.78G/3.46G [03:15<03:11, 8.76MB/s]
🚀 Running instant CPU test pipeline...0%|| 3.46G/3.46G [04:28<00:00, 25.0MB/s]
🖼️Generating tiny test image (128x128)...
100%|| 5/5 [00:26<00:00,  5.23s/it]
✅ Image saved: test_image_1756129018.png
🎬 Generating tiny test video (2 frames, CPU-safe)...
✅ Video saved: test_video_1756129018.mp4

🎉 Done! Outputs:
🖼️test_image_1756129018.png
🎬 test_video_1756129018.mp4
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

test_image_1756129018



This confirms:

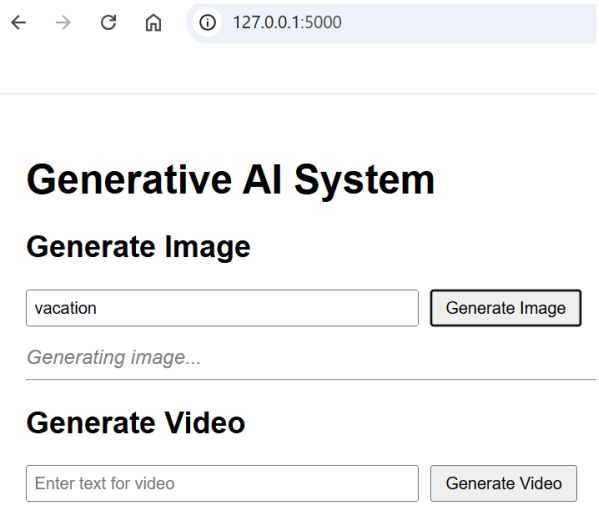
- The **pipeline runs without crashing**.
- Image generation works.
- Video generation function executes, writes an MP4.

Next, I created a minimal Flask backend that integrates generate_media.py to test:

prompt → image → video from a web interface.

Install flask:

python -m pip install flask



Generate Image

vacation | Generate Image

```
config.json: 100%|████████████████████████████████████████████████| 743/743 [00:00<?, ?B/s]
config.json: 100%|████████████████████████████████████████████████| 547/547 [00:00<?, ?B/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to re
gular HTTP download. For better performance, install the package with: `pip install huggingface_hub
[hf_xet]` or `pip install hf_xet`
vocab.json: 1.06MB [00:00, 30.0MB/s]
model.safetensors: 100%|████████████████████████████████████████| 492M/492M [00:36<00:00, 13.4MB/s]
diffusion_pytorch_model.safetensors: 100%|██████████████████████| 335M/335M [00:55<00:00, 6.05MB/s]
model.safetensors: 100%|████████████████████████████████████████| 1.22G/1.22G [02:27<00:00, 8.24MB/s]
diffusion_pytorch_model.safetensors: 100%|██████████████████████| 3.44G/3.44G [03:37<00:00, 15.8MB/s]
Fetching 15 files: 100%|████████████████████████████████████████| 15/15 [03:40<00:00, 14.71s/it]
Loading pipeline components...: 100%|███████████████████████████| 7/7 [00:01<00:00,  4.26it/s]
Text-to-image model loaded.etensors: 100%|██████████████████████| 3.44G/3.44G [03:37<00:00, 29.9MB/s]
100%|███████████████████████████████████████████████████████████| 50/50 [12:57<00:00, 15.55s/it]
127.0.0.1 - - [25/Aug/2025 22:21:51] "POST /generate_image HTTP/1.1" 200 -
127.0.0.1 - - [25/Aug/2025 22:21:52] "GET /static/images/20250825_220810.png HTTP/1.1" 200 -
```

For a video, it is too large. Run generate_media_cpu_test.py test script to generate a small video:

generate_media_cpu_demo.py

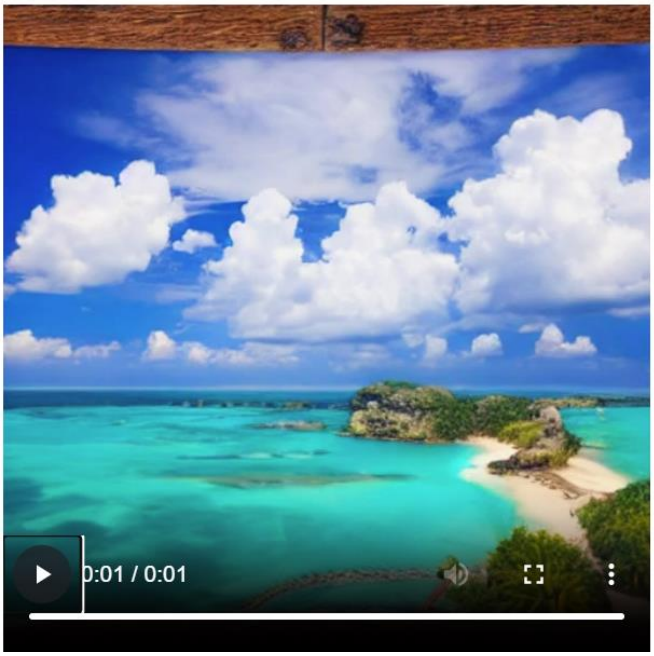
python generate_media_cpu_demo.py --prompt "A dragon flying over a mountain"

```
🚀 Running CPU test pipeline...
🖼️Generating tiny test image (128x128)...
Loading pipeline components...: 100%|                                        | 7/7 [00:02<00:00,  3.10it/s]
100%|                                                                        | 5/5 [01:07<00:00, 13.52s/it]
✅ Image saved: test_image_1756134285.png
🎬 Generating tiny test video (2 frames)...
Loading pipeline components...:   0%|                                              | 0/5 [00:00<?, ?it/s]
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python generate_media_cpu_demo.py --p
rompt "A dragon flying over a mountain"
>>
🚀 Running CPU demo pipeline...
📝 Prompt: A dragon flying over a mountain
🖼️Generating tiny test image (128x128)...
✅ Image saved: test_image_1756134952.png
🎬 Generating tiny test video (2 frames)...
✅ Tiny test video saved: test_video_1756134952.mp4
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative AI System>
```

workflow: prompt → image → video

Generate Video

rock | Generate Video



Step 4: Mobile Application Development

□ React Native:

- Build a mobile app to interact with the backend, allowing users to generate and view images and videos from text.

Mobile app part:

Node.js is installed:

node -v

npm -v

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> node -v
v22.18.0
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> npm -v
10.9.3
```

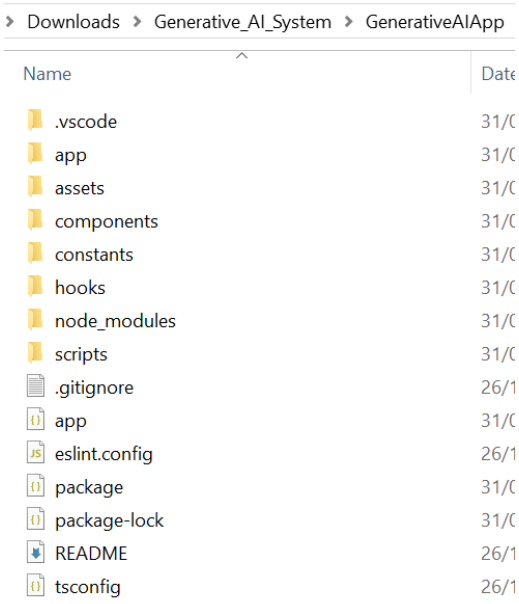
Start a new React Native project using Expo:

npx create-expo-app GenerativeAIApp

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> npx create-expo-app GenerativeAIApp
Need to install the following packages:
create-expo-app@3.5.3
Ok to proceed? (y) y

Creating an Expo project using the default template.

To choose from all available templates (https://github.com/expo/expo/tree/main/templates) pass in t
he --template arg:
  $ npx create-expo-app --template

To choose from all available examples (https://github.com/expo/examples) pass in the --example arg:
  $ npx create-expo-app --example

√ Downloaded and extracted project files.
> npm install
```

```
174 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

✅ Your project is ready!

To run your project, navigate to the directory and run one of the following npm commands.

- cd GenerativeAIApp
- npm run android
- npm run ios # you need to use macOS to build the iOS project - use the Expo app if you need to do
 iOS development without a Mac
- npm run web
npm notice
npm notice New major version of npm available! 10.9.3 -> 11.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
npm notice To update run: npm install -g npm@11.5.2
npm notice
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

This creates a folder GenerativeAIApp:



cd GenerativeAIApp

npx expo start

http://localhost:8081

Welcome! 👋

Step 1: Try it

Edit **app/(tabs)/index.tsx** to see changes. Press **F12** to open developer tools.

Step 2: Explore

Tap the Explore tab to learn more about what's included in this starter app.

Step 3: Get a fresh start

When you're ready, run **npm run reset-project** to get a fresh **app** directory. This will move the current **app** to **app-example**.

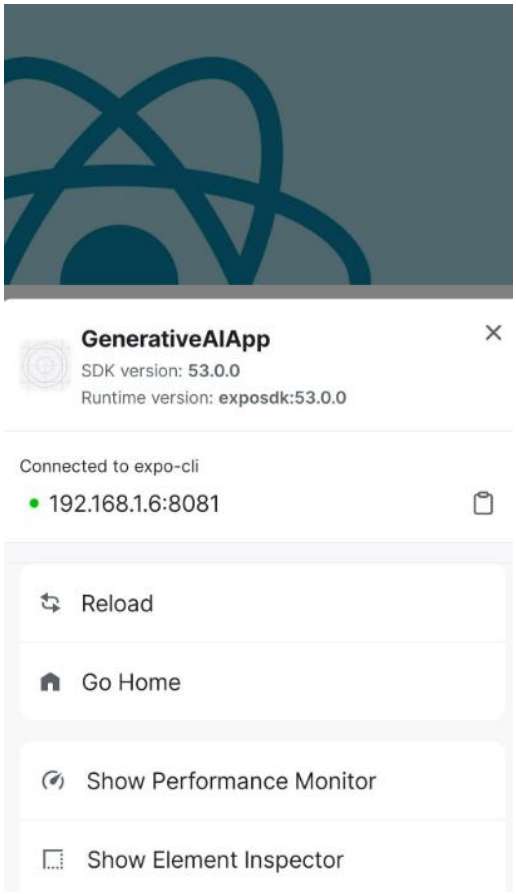
```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\GenerativeAIApp> npx expo start
Starting project at C:\Users\Hoi Yee\Downloads\Generative_AI_System\GenerativeAIApp
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo version:
  react-native@0.79.6 - expected version: 0.79.5
Your project may not work correctly until you install the expected versions of the packages.
```

press w to open a web version.

Install Expo Go on my phone.

Scan the QR code with Expo Go app

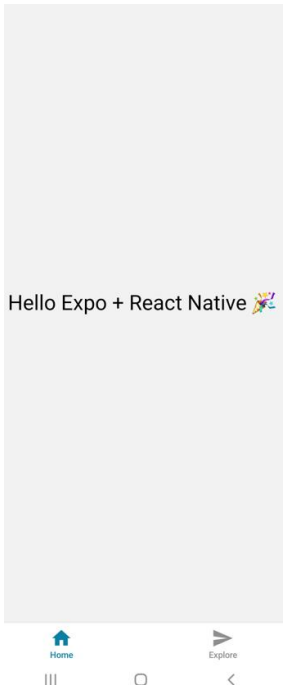
After scanning:

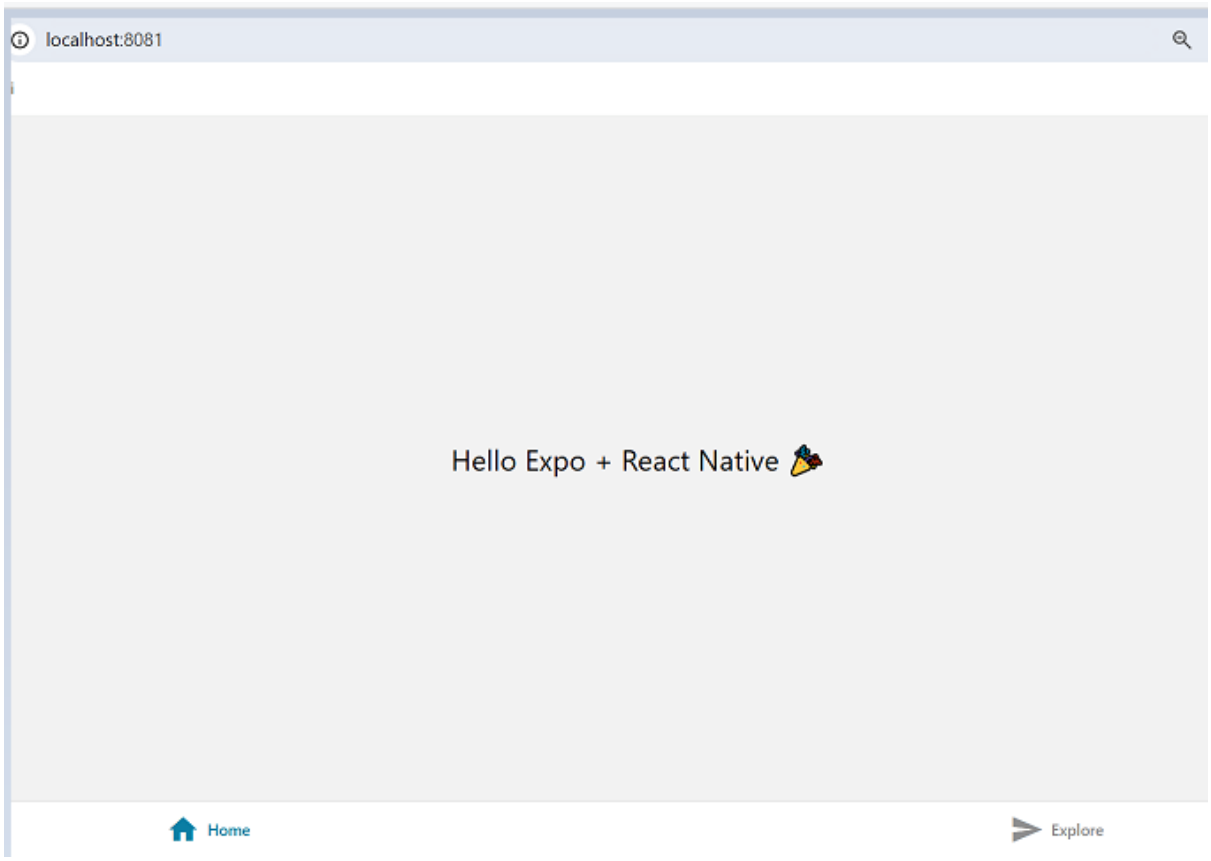


Inside GenerativeAIApp folder, code . (the . means “open current folder” in VS Code).

Expand app/ → index.tsx.

Reload Expo Go on my phone:





To ensure AI generation logic works before UI:

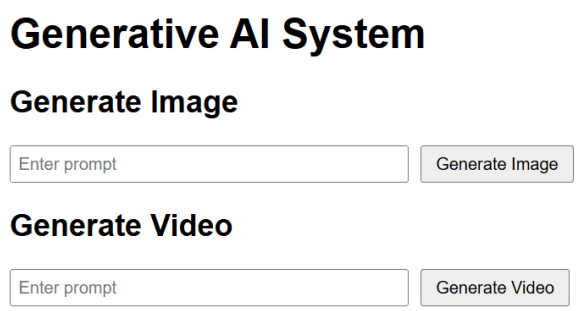
Create Flask API endpoints:

/generate_image → accepts text, returns image (file path or base64).

/generate_video → accepts text, returns video.

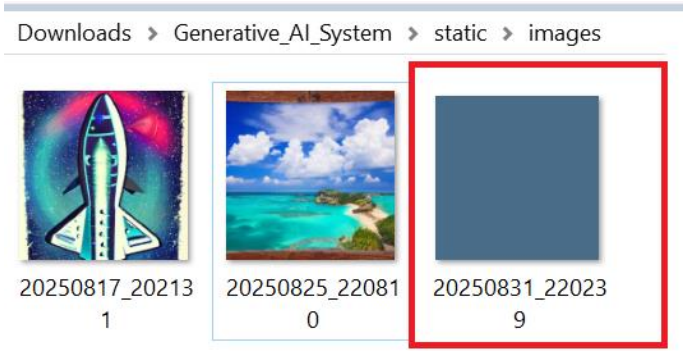
python app.py

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python app.py
>>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a proc
tion WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 108-806-960
```



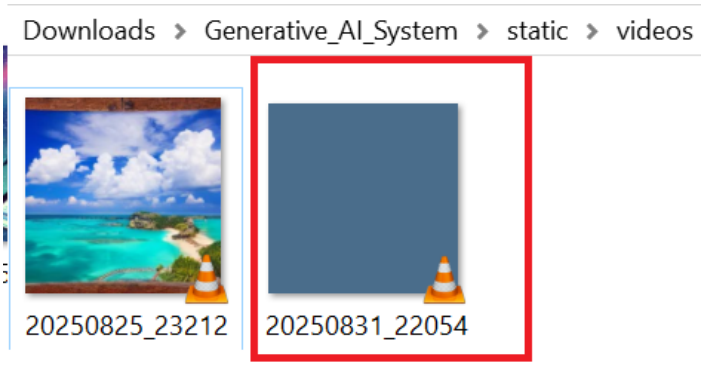
Invoke-WebRequest -Uri "http://127.0.0.1:5000/generate_image" -Method POST -Headers @{ "Content-Type" = "application/json" } -Body '{ "prompt": "a cat on the moon" }' -OutFile "result.png"

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> Invoke-WebRequest -Uri "http://
127.0.0.1:5000/generate_image" -Method POST -Headers @{ "Content-Type" = "application/json" }
 -Body '{ "prompt": "a cat on the moon" }' -OutFile "result.png"
>>
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```



Invoke-WebRequest -Uri "http://127.0.0.1:5000/generate_video" -Method POST -Headers @{ "Content-Type" = "application/json" } -Body '{ "prompt": "a cat on the moon" }' -OutFile "result.mp4"

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> Invoke-WebRequest -Uri "http://
127.0.0.1:5000/generate_video" -Method POST -Headers @{ "Content-Type" = "application/json" }
 -Body '{ "prompt": "a cat on the moon" }' -OutFile "result.mp4"
>>
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```



Both image and video generation endpoints are working.

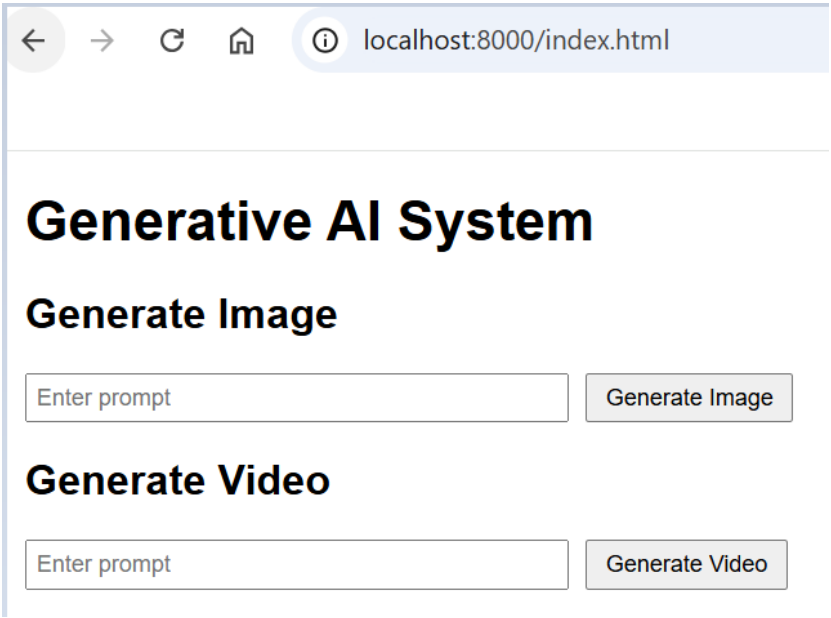
Build a Frontend Interface with:

- A simple form where users enter a prompt.
- Buttons: “Generate Image” / “Generate Video”.
- Display the output on the page.

python -m http.server 8000

Open browser and go to:

http://localhost:8000/index.html



Web frontend is working.

Next step, build a mobile app using Expo **(**interacts with your Flask backend):

npm install -g expo-cli

npx create-expo-app MyGenerativeApp

This will create a new Expo project in MyGenerativeApp folder.

MyGenerativeApp/

├── App.js # Main app file

├── package.json

├── assets/

└── ...

Put code in App.js

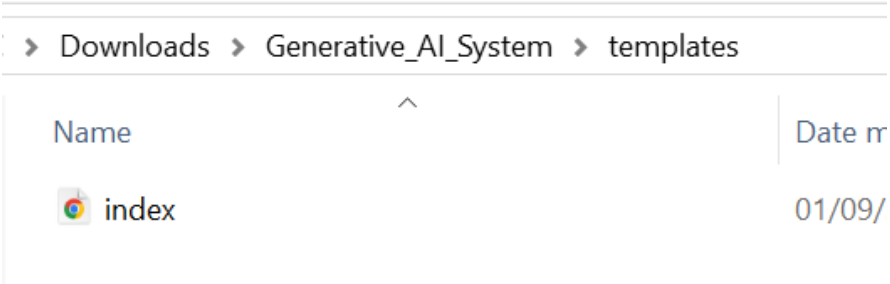
Run your Expo app inside MyGenerativeApp:

npm install

npx expo start

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> npx expo start
>>
Starting project at C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp
Starting Metro Bundler
The following packages should be updated for best compatibility with the installed expo versi
on:
  react-native@0.79.6 - expected version: 0.79.5
Your project may not work correctly until you install the expected versions of the packages.
```

Make new index.html



```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a produc
tion WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 283-858-067
127.0.0.1 - - [01/Sep/2025 16:37:06] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [01/Sep/2025 16:37:07] "GET /favicon.ico HTTP/1.1" 404 -
```

Flask is running 🚀

To test:

(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> curl.exe -X POST "http://127.0.0.1:5000/generate_image" -H "Content-Type: application/json" -d '{\"prompt\":\"test\"}' --output test.png

The file test.png is in:

C:\Users\Hoi Yee\Downloads\Generative_AI_System\test.png

Flask /generate_image endpoint is working.

Next Step: Connect Flask → React Native app

Inside app.js

Change it to your Flask server (your PC's LAN IP, port 5000):

const API_URL = 'http://192.168.1.6:5000'; change this to

const API_URL = 'http://127.0.0.1:5000/';

Install expo-file-system:

(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> npx expo install expo-file-system

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> npx expo instal
l expo-file-system
› Installing 1 SDK 53.0.0 compatible native module using npm
> npm install

up to date, audited 940 packages in 2s

174 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp>
```

Run Flask backend: python app.py

make sure package.json:

change "main": "expo-router/entry",

to

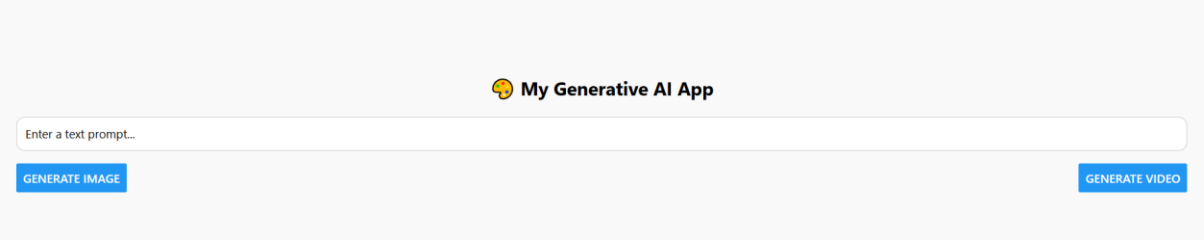
"main": "App.js",
"main": "node_modules/expo/AppEntry.js"

Install dependencies:

npm install

Run Expo:

npx expo start -c



In App.js:

const res = await fetch("http://192.168.1.6:5000/generate-image"

Replace my **computer's LAN IP** to:

http://192.168.1.6:5000

generate_media.py

Text → Image → Video

Combines both steps automatically. Generates image first, then video from it.
Install Expo Video library so react Native app can display and play the generated video inside the app**:**

npx expo install expo-av

Start flask first

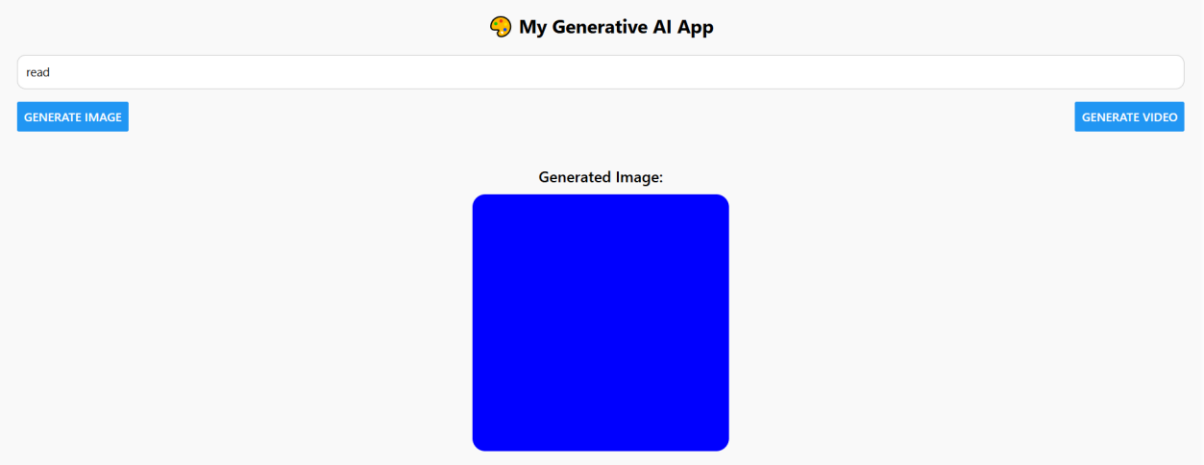
python app.py

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a produ
tion WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 627-255-011
```

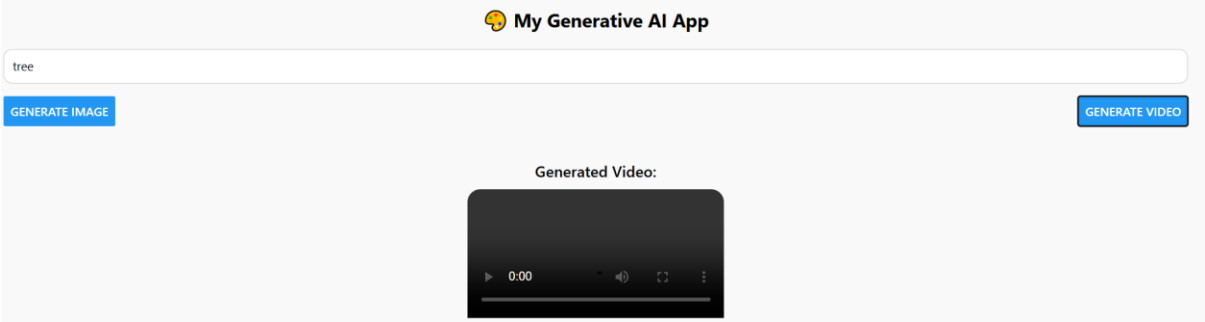
npx expo start -c

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> npx expo start
-c
Starting project at C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp
Starting Metro Bundler
warning: Bundler cache is empty, rebuilding (this may take a minute)
The following packages should be updated for best compatibility with the installed expo versi
on:
  react-native@0.79.6 - expected version: 0.79.5
Your project may not work correctly until you install the expected versions of the packages.
```

typing a prompt and pressing “Generate Image”



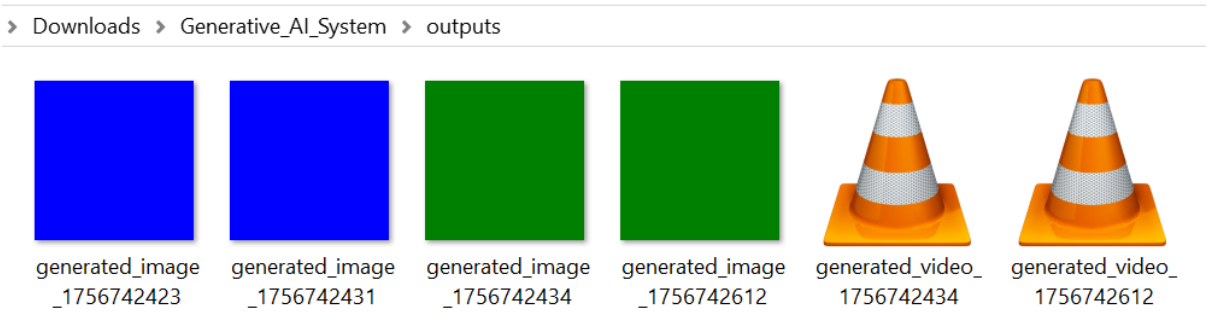
typing a prompt and pressing “Generate Video”



In app.py:

```
192.168.1.6 - - [02/Sep/2025 00:00:23] "OPTIONS /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
 92.168.1.6 - - [02/Sep/2025 00:00:23] "POST /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
 92.168.1.6 - - [02/Sep/2025 00:00:23] "GET /outputs/generated_image_1756742423.png HTTP/1.1'
192.168.1.6 - - [02/Sep/2025 00:00:23] "POST /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
 200 -
192.168.1.6 - - [02/Sep/2025 00:00:23] "POST /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
 200 -
192.168.1.6 - - [02/Sep/2025 00:00:23] "POST /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742423.png
 200 -
192.168.1.6 - - [02/Sep/2025 00:00:23] "POST /generate_image HTTP/1.1" 200 -
192.168.1.6 - - [02/Sep/2025 00:00:23] "GET /outputs/generated_image_1756742423.png HTTP/1.1'
 200 -
192.168.1.6 - - [02/Sep/2025 00:00:31] "OPTIONS /generate_image HTTP/1.1" 200 -
✅ Generated Image URL: http://192.168.1.6:5000/outputs/generated_image_1756742431.png
```

```
192.168.1.6 - - [02/Sep/2025 00:00:34] "POST /generate_video HTTP/1.1" 200 -
192.168.1.6 - - [02/Sep/2025 00:00:34] "GET /outputs/generated_video_1756742434.mp4 HTTP/1.1"
 206 -
192.168.1.6 - - [02/Sep/2025 00:03:32] "OPTIONS /generate_video HTTP/1.1" 200 -
✅ Generated Video URL: http://192.168.1.6:5000/outputs/generated_video_1756742612.mp4
192.168.1.6 - - [02/Sep/2025 00:03:32] "POST /generate_video HTTP/1.1" 200 -
192.168.1.6 - - [02/Sep/2025 00:03:32] "GET /outputs/generated_video_1756742612.mp4 HTTP/1.1"
 206 -
```



Step 4-Mobile Application Development: Done

backend/

├─ app.py

├─ requirements.txt

└─ models/

Go to backend/ folder:

Create virtual environment and install dependencies:

python -m venv

.\venv\Scripts\activate

pip install -r requirements.txt

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> cd backend
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python
-m venv venv
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> .\venv\
Scripts\activate
(venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> pip instal
l -r requirements.txt
Collecting Flask==2.3.3 (from -r requirements.txt (line 1))
  Downloading flask-2.3.3-py3-none-any.whl.metadata (3.6 kB)
Collecting flask-cors==3.0.10 (from -r requirements.txt (line 2))
  Downloading Flask_Cors-3.0.10-py2.py3-none-any.whl.metadata (5.4 kB)
Collecting Pillow==10.0.0 (from -r requirements.txt (line 3))
  Downloading Pillow-10.0.0-cp311-cp311-win_amd64.whl.metadata (9.6 kB)
Collecting moviepy==1.0.3 (from -r requirements.txt (line 4))
  Downloading moviepy-1.0.3.tar.gz (388 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 388.3/388.3 kB 4.8 MB/s eta 0:00:00
  Installing build dependencies ... done
```

Start Flask server:

python app.py

```
Successfully built moviepy
Installing collected packages: urllib3, Six, Pillow, numpy, MarkupSafe, itsdangerous, imageio
_ffmpeg, idna, decorator, colorama, charset_normalizer, certifi, blinker, Werkzeug, tqdm, req
uests, Jinja2, imageio, click, proglog, Flask, moviepy, flask-cors
Successfully installed Flask-2.3.3 Jinja2-3.1.6 MarkupSafe-3.0.2 Pillow-10.0.0 Six-1.17.0 Wer
kzeug-3.1.3 blinker-1.9.0 certifi-2025.8.3 charset_normalizer-3.4.3 click-8.2.1 colorama-0.4.
6 decorator-4.4.2 flask-cors-3.0.10 idna-3.10 imageio-2.37.0 imageio_ffmpeg-0.6.0 itsdangerou
s-2.2.0 moviepy-1.0.3 numpy-2.3.2 proglog-0.1.12 requests-2.32.5 tqdm-4.67.1 urllib3-2.5.0

[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python app
.py
>>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a produc
tion WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 810-547-622
```

Access endpoints locally:

- Text-to-image: POST http://127.0.0.1:5000/generate-image
- Text-to-video: POST http://127.0.0.1:5000/generate-video

backend is now running

Connect the React Native app to your Flask backend (to display generated images and videos):

Install Axios in React Native (for HTTP requests):
Axios is a JavaScript library used to make HTTP requests (like GET, POST, PUT, DELETE). It's one of the most popular libraries for working with APIs.

From MyGenerativeApp folder:

npm install axios

App.js -updated

- The endpoints return URLs:

  {"image_url": "images/output.png"}
  {"video_url": "videos/output.mp4"}

- In React Native, full URL will be:

  http://192.168.1.6:5000/images/output.png
  http://192.168.1.6:5000/videos/output.mp4

In backend folder: python app.py

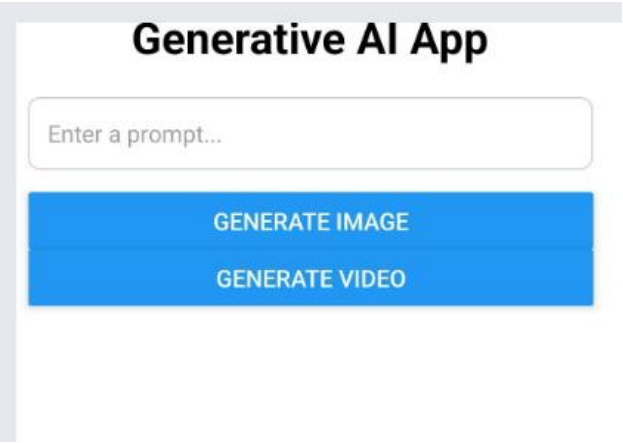
Output:

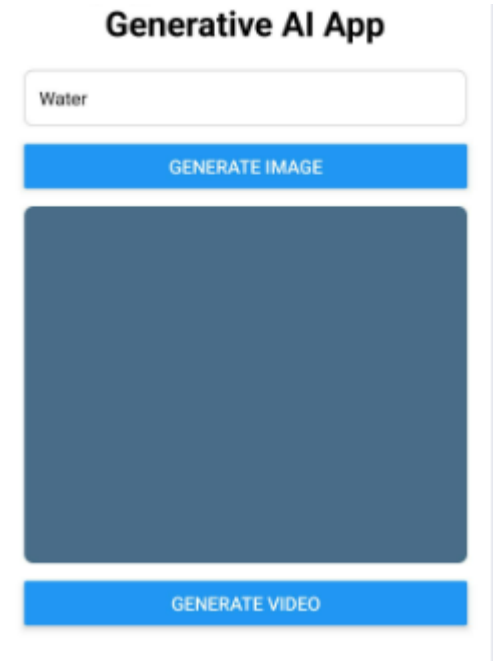
* Running on http://127.0.0.1:5000
* Running on http://192.168.1.6:5000

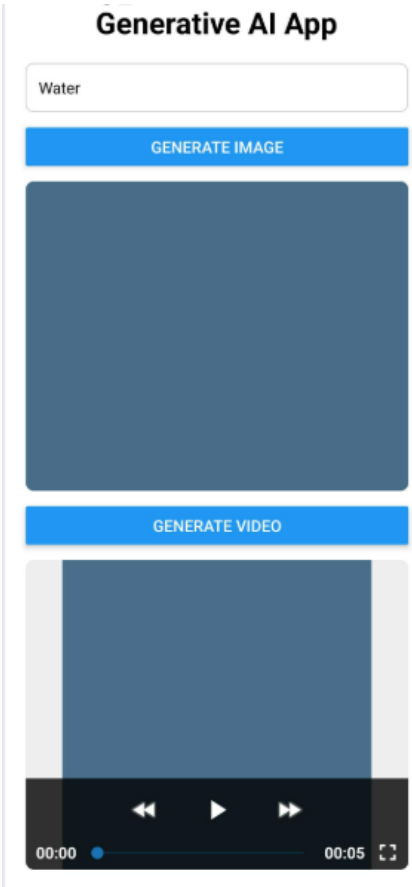
phone will use the IP 192.168.1.6:5000.

Start React Native app: npx expo start

open up Expo Go on phone
scan Metro Bundler QR code

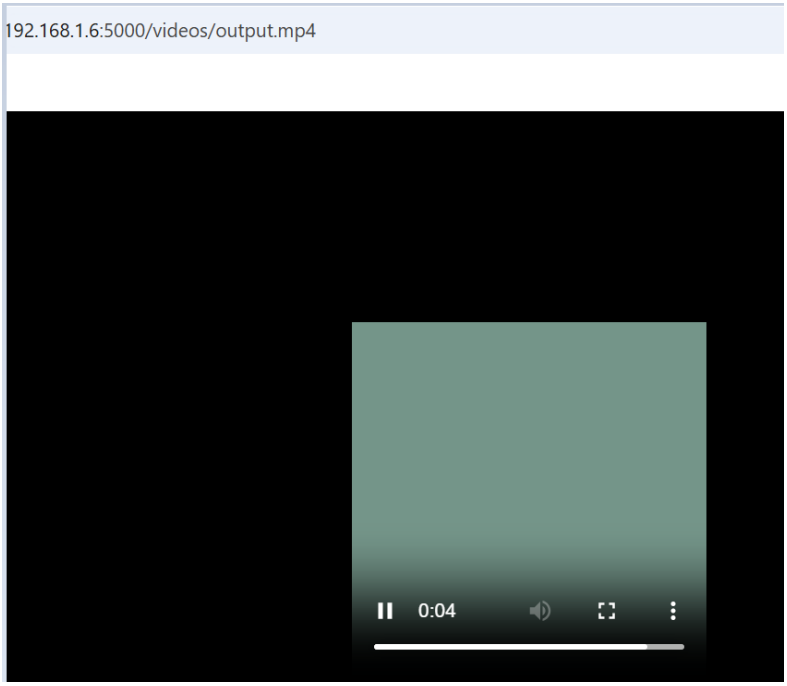






Access generated media → you can view images/videos in a browser, e.g.:

http://192.168.1.6:5000/images/output.png
http://192.168.1.6:5000/videos/output.mp4



Replace the placeholder image/video generation with actual AI models to produces real generative content.

```
backend/
├── app.py
├── requirements.txt
├── models/
│   ├── text_to_image_model.py
│   └── text_to_video_model.py
└── outputs/
    ├── images/
    └── videos/
```

- models/ → contains the actual AI model code.
- outputs/ → will store generated images/videos.

cd backend, Install PyTorch:

```
_ai_system\mygenerativeapp\backend\venv\lib\site-packages (from torchvision) (10.0.0)
Collecting mpmath<1.4,>=1.1.0 (from sympy>=1.13.3->torch)
  Using cached https://download.pytorch.org/whl/mpmath-1.3.0-py3-none-any.whl (536 kB)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\hoi yee\downloads\generative_ai_sy
stem\mygenerativeapp\backend\venv\lib\site-packages (from jinja2->torch) (3.0.2)
Using cached https://download.pytorch.org/whl/cpu/torch-2.8.0%2Bcpu-cp311-cp311-win_amd64.whl
 (619.4 MB)
Using cached https://download.pytorch.org/whl/cpu/torchvision-0.23.0%2Bcpu-cp311-cp311-win_am
d64.whl (1.6 MB)
Using cached https://download.pytorch.org/whl/cpu/torchaudio-2.8.0%2Bcpu-cp311-cp311-win_amd6
4.whl (2.5 MB)
Using cached https://download.pytorch.org/whl/sympy-1.13.3-py3-none-any.whl (6.2 MB)
Using cached https://download.pytorch.org/whl/typing_extensions-4.12.2-py3-none-any.whl (37 k
B)
Using cached https://download.pytorch.org/whl/filelock-3.13.1-py3-none-any.whl (11 kB)
Using cached https://download.pytorch.org/whl/fsspec-2024.6.1-py3-none-any.whl (177 kB)
Using cached https://download.pytorch.org/whl/networkx-3.3-py3-none-any.whl (1.7 MB)
Installing collected packages: mpmath, typing-extensions, sympy, networkx, fsspec, filelock,
torch, torchvision, torchaudio
```

pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cpu

Verify installation:

python -c "import torch; print(torch.__version__)"

```
[notice] A new release of pip is available: 24.0 -> 25.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python -c
"import torch; print(torch.__version__)"
2.8.0+cpu
(venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend>
```

cd backend
python app.py

```
(venv) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python app
.py
>>
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a produc
tion WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.6:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 810-547-622
```

To test

For generating an image:

Use Invoke-RestMethod (recommended in PowerShell)

Invoke-RestMethod -Uri http://127.0.0.1:5000/generate-image `

-Method POST `

-ContentType "application/json" `

-Body '{"text":"Hello world"}'

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> Invoke
RestMethod -Uri http://127.0.0.1:5000/generate-image `
>>                    -Method POST `
>>                    -ContentType "application/json" `
>>                    -Body '{"text":"Hello world"}'
>>
   633;C
image_url
---------
images/output.png


(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend>
```

Open this image in a browser:

http://127.0.0.1:5000/images/output.png

127.0.0.1:5000/images/output.png

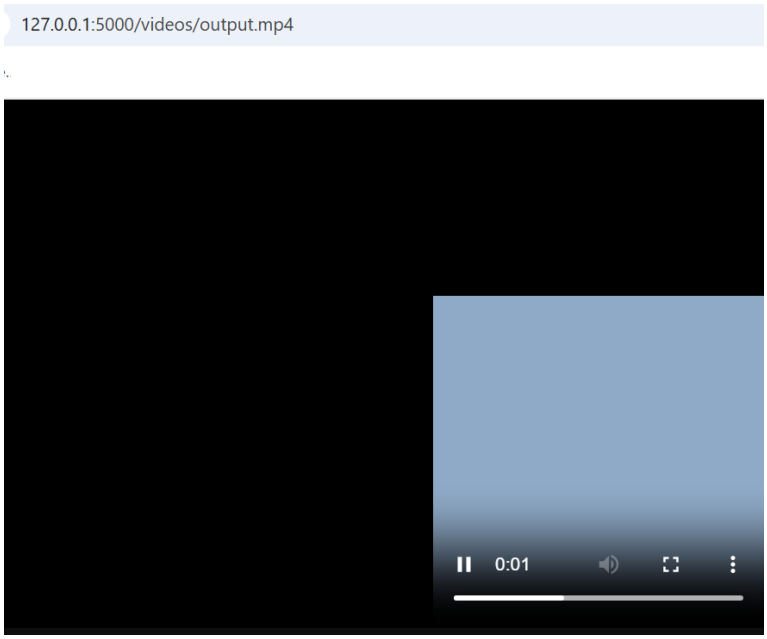
And for video:

```
Invoke-RestMethod -Uri http://127.0.0.1:5000/generate-video `
                  -Method POST `
                  -ContentType "application/json" `
                  -Body '{"text":"Hello world"}'
```

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> Invoke
RestMethod -Uri http://127.0.0.1:5000/generate-video `
>>                   -Method POST `
>>                   -ContentType "application/json" `
>>                   -Body '{"text":"Hello world"}'
>>
   633;C
video_url
---------
videos/output.mp4


(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend>
```

Open this video in a browser:

http://127.0.0.1:5000/videos/output.mp4



Integrate real AI models

In backend venv, run:

pip install moviepy
python -m venv venv313

Activate the new venv:

.\venv313\Scripts\Activate.ps1

Install dependencies inside the new venv

python.exe -m pip install --upgrade pip

pip install flask flask-cors pillow moviepy torch diffusers transformers

```
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python.
exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hoi yee\downloads\generative_ai_system\mygener
ativeapp\backend\venv313\lib\site-packages (24.0)
Collecting pip
  Using cached pip-25.2-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.2-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-25.2
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> pip ins
tall flask flask-cors pillow moviepy torch diffusers transformers
Collecting flask
  Using cached flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting flask-cors
  Using cached flask_cors-6.0.1-py3-none-any.whl.metadata (5.3 kB)
Collecting pillow
  Using cached pillow-11.3.0-cp311-cp311-win_amd64.whl.metadata (9.2 kB)
Collecting moviepy
```


```
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python.
exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\hoi yee\downloads\generative_ai_system\mygener
ativeapp\backend\venv313\lib\site-packages (24.0)
Collecting pip
  Using cached pip-25.2-py3-none-any.whl.metadata (4.7 kB)
Using cached pip-25.2-py3-none-any.whl (1.8 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 24.0
    Uninstalling pip-24.0:
      Successfully uninstalled pip-24.0
Successfully installed pip-25.2
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> pip ins
tall flask flask-cors pillow moviepy torch diffusers transformers
Collecting flask
  Using cached flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting flask-cors
  Using cached flask_cors-6.0.1-py3-none-any.whl.metadata (5.3 kB)
Collecting pillow
  Using cached pillow-11.3.0-cp311-cp311-win_amd64.whl.metadata (9.2 kB)
Collecting moviepy
```

```
Downloading networkx-3.5-py3-none-any.whl (2.0 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 18.9 MB/s  0:00:00
Using cached requests-2.32.5-py3-none-any.whl (64 kB)
Using cached charset_normalizer-3.4.3-cp311-cp311-win_amd64.whl (107 kB)
Using cached idna-3.10-py3-none-any.whl (70 kB)
Using cached urllib3-2.5.0-py3-none-any.whl (129 kB)
Using cached certifi-2025.8.3-py3-none-any.whl (161 kB)
Installing collected packages: mpmath, zipp, urllib3, typing-extensions, sympy, safetensors,
regex, pyyaml, python-dotenv, pillow, packaging, numpy, networkx, markupsafe, itsdangerous, i
mageio_ffmpeg, idna, fsspec, filelock, decorator, colorama, charset_normalizer, certifi, blin
ker, werkzeug, tqdm, requests, jinja2, importlib_metadata, imageio, click, torch, proglog, hu
ggingface-hub, flask, tokenizers, moviepy, flask-cors, diffusers, transformers
Successfully installed blinker-1.9.0 certifi-2025.8.3 charset_normalizer-3.4.3 click-8.2.1 co
lorama-0.4.6 decorator-5.2.1 diffusers-0.35.1 filelock-3.19.1 flask-3.1.2 flask-cors-6.0.1 fs
spec-2025.7.0 huggingface-hub-0.34.4 idna-3.10 imageio-2.37.0 imageio_ffmpeg-0.6.0 importlib_
metadata-8.7.0 itsdangerous-2.2.0 jinja2-3.1.6 markupsafe-3.0.2 moviepy-2.2.1 mpmath-1.3.0 ne
tworkx-3.5 numpy-2.3.2 packaging-25.0 pillow-11.3.0 proglog-0.1.12 python-dotenv-1.1.1 pyyaml
-6.0.2 regex-2025.9.1 requests-2.32.5 safetensors-0.6.2 sympy-1.14.0 tokenizers-0.22.0 torch-
2.8.0 tqdm-4.67.1 transformers-4.56.0 typing-extensions-4.15.0 urllib3-2.5.0 werkzeug-3.1.3 z
ipp-3.23.0
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend>
```

Verify MoviePy

python -c "from moviepy import VideoClip; print('MoviePy works!')"

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend> python -c "f
om moviepy import VideoClip; print('MoviePy works!')"
>>
MoviePy works!
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\backend>
```

It prints MoviePy works!, so the installation succeeded.

Integrate React Native frontend with Flask backend for video generation

Create frontend folder

MyGenerativeApp/
├── backend/ <-- Flask code
└── frontend/ <-- React Native code

Inside frontend folder:

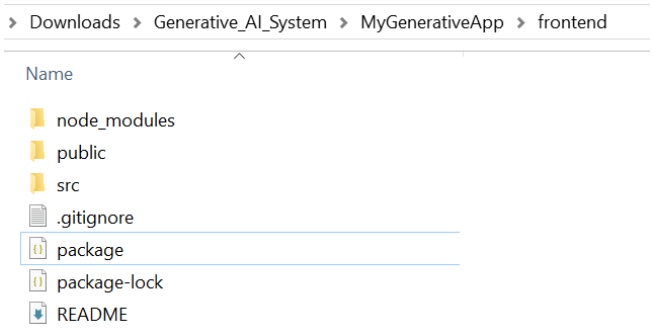
npm install axios (to make HTTP requests)

Connect frontend to the backend endpoints:

Create a new React app:

npx create-react-app .

```
Success! Created frontend at C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\
frontend
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp\frontend
  npm start

Happy hacking!
```



Create api.js

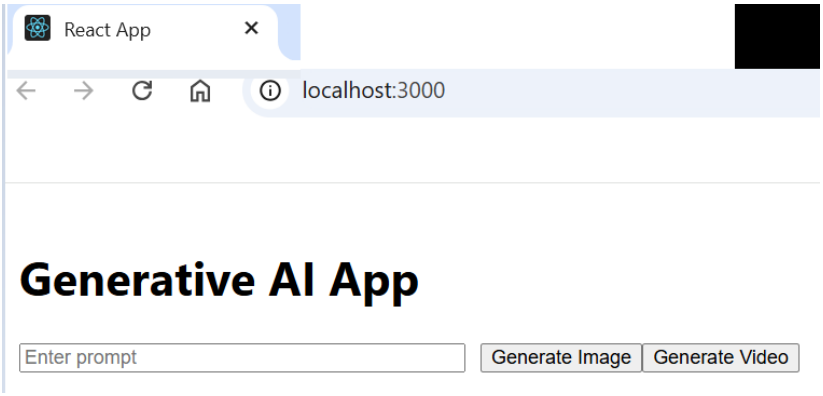
Inside frontend/src/, create api.js (to centralize all API calls.)

Run the frontend:

npm start

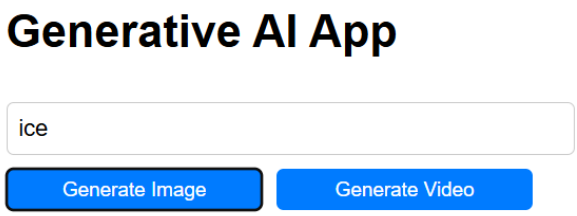
```
  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!
Compiled successfully!

You can now view frontend in the browser.

  Local:            http://localhost:3000
  On Your Network:  http://192.168.1.6:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

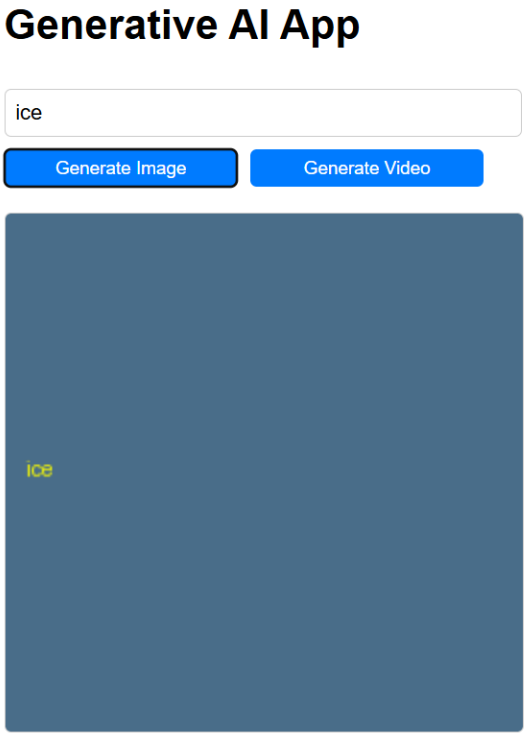
http://localhost:3000/

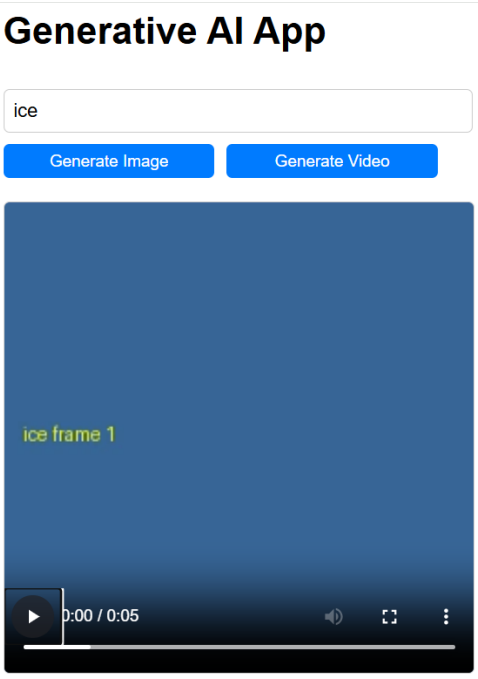


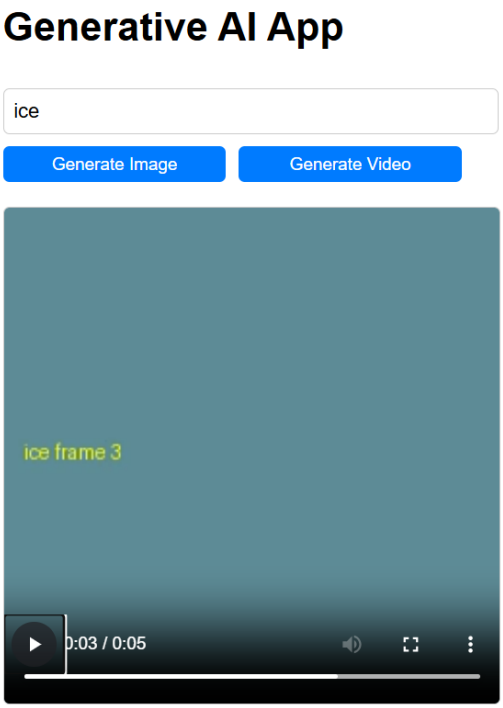
Backend Flask server runs on http://localhost:5000.

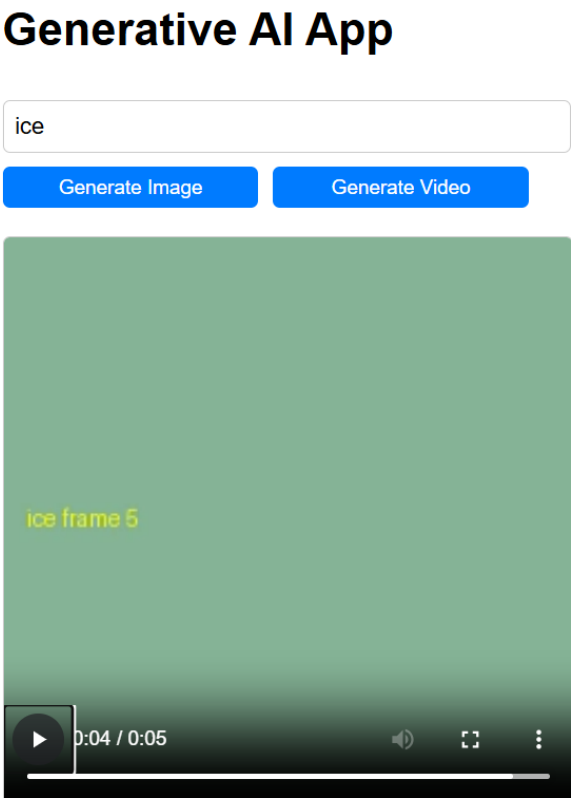
Open browser → input prompt → generate image or video.




Generative AI App
ice
Generate Image
Generate Video
ice


Generative AI App
ice
Generate Image
Generate Video
ice frame 1
0:00 / 0:05


Generative AI App
ice
Generate Image
Generate Video
ice frame 3
0:03 / 0:05


Generative AI App
ice
Generate Image
Generate Video
ice frame 5
0:04 / 0:05

Step 5-CI/CD Pipeline Setup:
□ GitHub Actions:

- Create a pipeline for building, testing, and deploying the generative AI models.

□ Configuration:

- Define steps for code checkout, dependency installation, testing, and deployment to AWS.

CI setup:

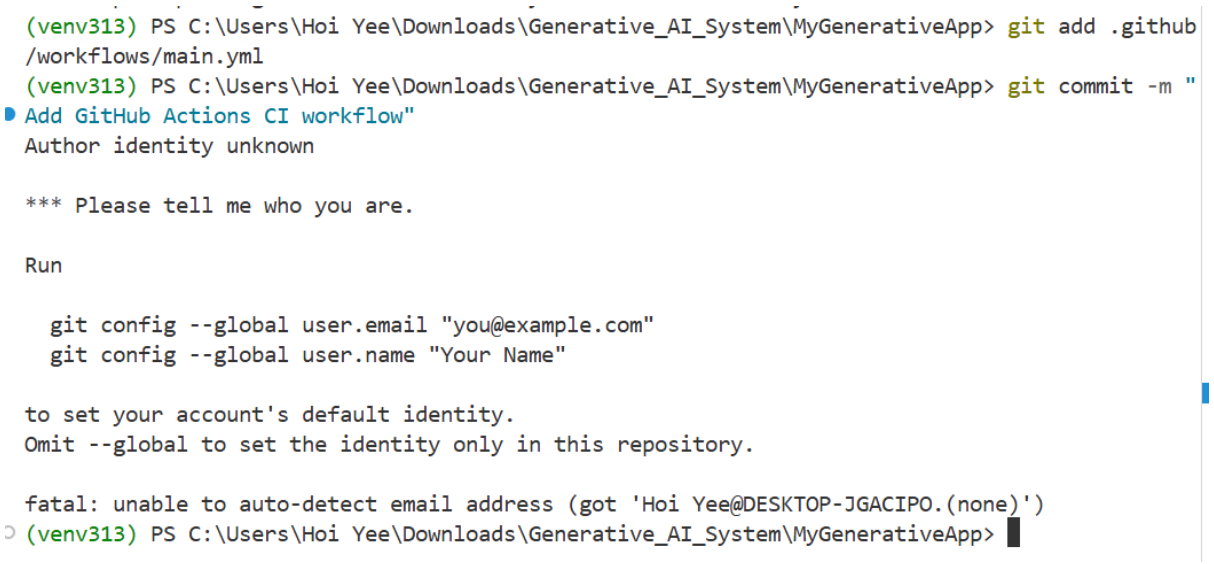
Inside MyGenerativeApp → Create a folder named .github → workflows → main.yml

Add minimal workflow content:

git add .github/workflows/main.yml

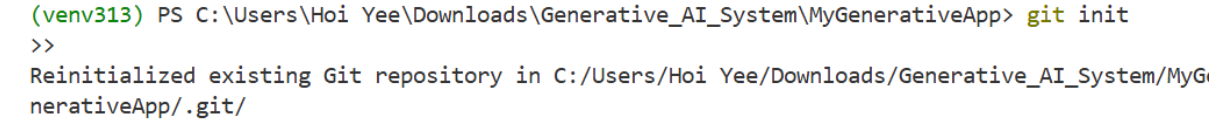
Commit and push:

git commit -m "Add GitHub Actions CI workflow"



Initialize Git repository:

git init



Add files:

git add .

Make the first commit:

git commit -m "Initial commit with workflow"

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> git branch
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> git add .
>>
warning: LF will be replaced by CRLF in app.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package-lock.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in package.json.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in tsconfig.json.
The file will have its original line endings in your working directory
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> git commit -m "
Initial commit with workflow"
>>
Author identity unknown

*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.
```

git config --global user.name "Ng Hoi Yee"

git config --global user.email " nhoiyee@yahoo.com.sg"

Verify configuration:

git config --list

```
global user.email   nhoiyee@yahoo.com.sg
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> git config --li
st
>>
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.name=Ng Hoi Yee
user.email= nhoiyee@yahoo.com.sg
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
core.symlinks=false
core.ignorecase=true
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp>
```

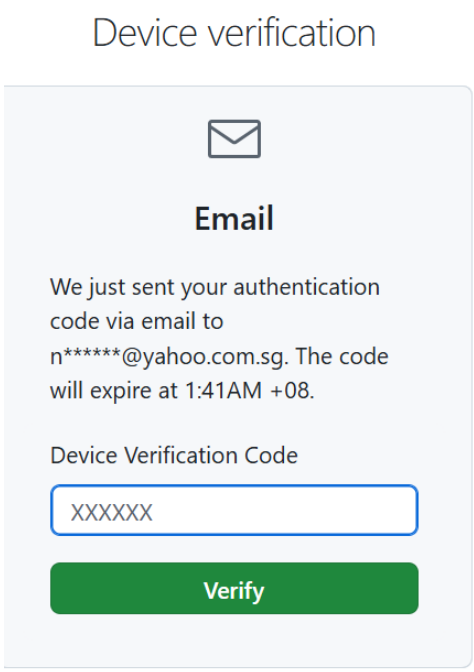
Commit your changes

git commit -m "Initial commit with workflow"

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\MyGenerativeApp> git commit -m "
Initial commit with workflow"
>>
[master (root-commit) 783aa18] Initial commit with workflow
 36 files changed, 13674 insertions(+)
 create mode 100644 .github/workflows/main.yml
 create mode 100644 .gitignore
 create mode 100644 .vscode/settings.json
 create mode 100644 App.js
 create mode 100644 README.md
 create mode 100644 app.json
 create mode 100644 app.py
 create mode 100644 assets/fonts/SpaceMono-Regular.ttf
 create mode 100644 assets/images/adaptive-icon.png
 create mode 100644 assets/images/favicon.png
 create mode 100644 assets/images/icon.png
 create mode 100644 assets/images/partial-react-logo.png
 create mode 100644 assets/images/react-logo.png
 create mode 100644 assets/images/react-logo@2x.png
 create mode 100644 assets/images/react-logo@3x.png
 create mode 100644 assets/images/splash-icon.png
 create mode 100644 components/Collapsible.tsx
 create mode 100644 components/ExternalLink.tsx
 create mode 100644 components/HapticTab.tsx
 create mode 100644 components/HelloWave.tsx
```

Go to GitHub and create a repository

Log in GitHub



Click New repository:

Name it GenerativeAIApp.

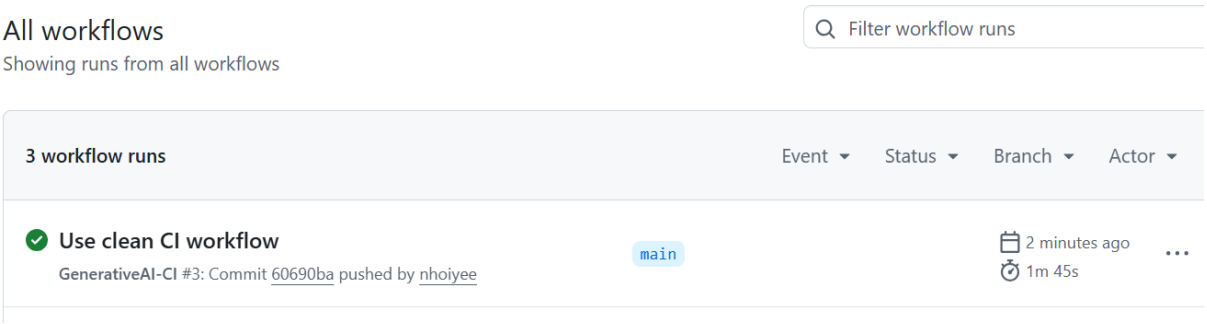
Click **Create repository**

git add .github/workflows/main.yml

git commit -m "Use clean CI workflow"

git push origin main

Go to GitHub → Actions tab → the run starts



my CI set up is done.

Step 6 Cloud Infrastructure:
□ AWS Services:

- Use Amazon ECS for container orchestration.

- Use Amazon S3 for storing generated media.

- Optionally use Amazon RDS for logs and metadata.

□ Terraform Configuration:

- Write Terraform scripts to define and provision the infrastructure

Terraform defines the AWS infrastructure (ECS cluster, S3 bucket, maybe RDS).

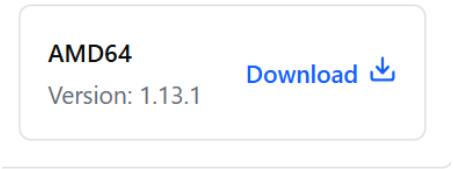
Write Terraform starter files → create AWS infra.

```
└── terraform/
    ├── main.tf
    ├── s3.tf
    ├── ecs.tf
```

Download terraform

https://developer.hashicorp.com/terraform/install?utm_source=chatgpt.com

cd C:\Users\Hoi Yee\Downloads\Generative_AI_System\terraform



Check it is installed:

.\terraform.exe -version

```
for more details.
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\terraform_win> .\terraform.exe
version
>>
Terraform v1.13.1
on windows_amd64
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\terraform_win>
```

Create a folder:

C:\Program Files\Terraform

Move terraform.exe to this folder

terraform -version

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> terraform -version
>>
Terraform v1.13.1
on windows_amd64
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Terraform is installed.

Initialize Terraform:

terraform init

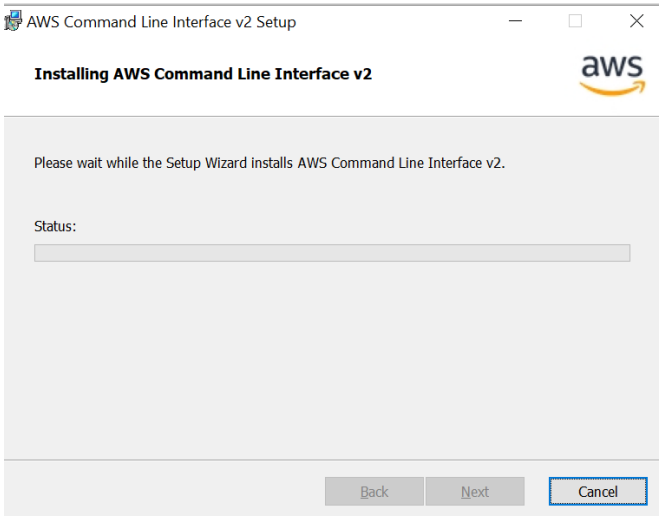
```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System\terraform> terraform init
>>
Initializing the backend...
Initializing modules...
Downloading registry.terraform.io/terraform-aws-modules/vpc/aws 5.0.0 for vpc...
- vpc in .terraform\modules\vpc
Initializing provider plugins...
- Finding hashicorp/aws versions matching ">= 5.0.0"...
- Finding latest version of hashicorp/random...
- Installing hashicorp/aws v6.11.0...
- Installed hashicorp/aws v6.11.0 (signed by HashiCorp)
- Installing hashicorp/random v3.7.2...
- Installed hashicorp/random v3.7.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

Install AWS CLI on Windows:

https://aws.amazon.com/cli/?utm_source=chatgpt.com

Download Link**:**

https://awscli.amazonaws.com/AWSCLIV2.msi



C:\Program Files\Amazon\AWSCLIV2\

Add AWS CLI to PATH:

C:\Program Files\Amazon\AWSCLIV2\

Verify the Installation

aws –version

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws --version
>>
aws-cli/2.28.21 Python/3.13.7 Windows/10 exe/AMD64
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Configure AWS CLI Credentials

Run the configuration command:

aws configure

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws configure
AWS Access Key ID [None]:
```

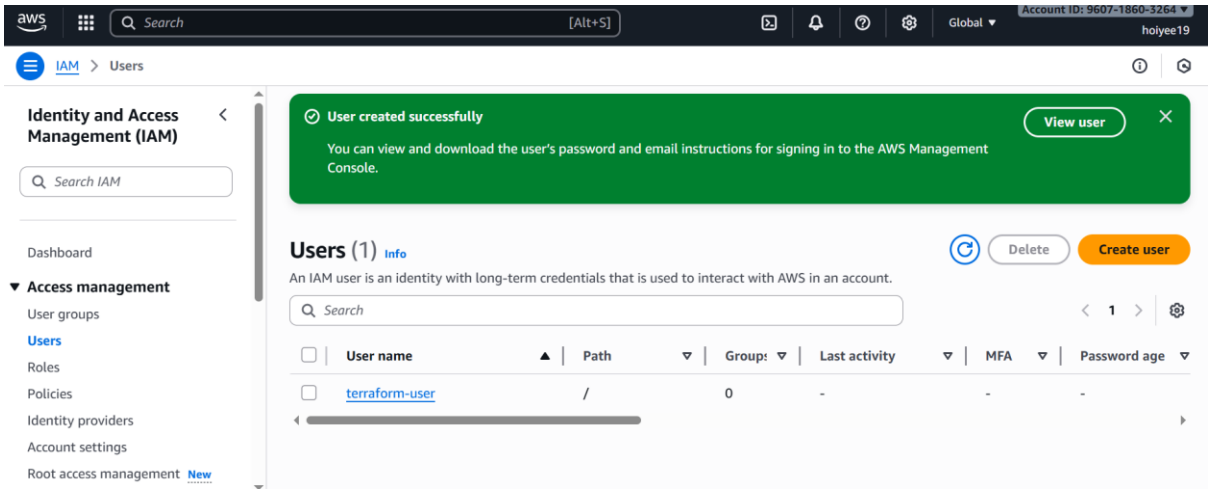
To get AWS Access Key ID:

https://aws.amazon.com/console/?utm_source=chatgpt.com

Sign in with my AWS account

Create user:

terraform-user



Access Key : AKIA57L2KQQA----------W

Secret Access Key : ypsOI----------x12I7eeOXjhIxdLz8

Default region name [None]: ap-southeast-1

Default output format [None]: json

Verify AWS CLI setup:

aws s3 ls

Create an S3 bucket (to store generated images/videos and accessing them from Python app):

Create an S3 bucket:

aws s3 mb s3://my-generative-ai-bucket --region ap-southeast-1

```
>>
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3 mb s3://my-generative-
-bucket --region ap-southeast-1
>>
make_bucket: my-generative-ai-bucket
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Verify it was created:

aws s3 ls

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3 ls
>>
2025-09-02 22:33:48 my-generative-ai-bucket
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Set Bucket Permissions (for Public Access):

C:\Users\Hoi Yee\Downloads\Generative_AI_System

Create policy.json (for public read access) then run:

aws s3api put-bucket-policy --bucket my-generative-ai-bucket --policy file://policy.json

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3 mb s3://my-generative-ai
-bucket --region ap-southeast-1
>>
make_bucket: my-generative-ai-bucket
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3 ls
>>
2025-09-02 22:33:48 my-generative-ai-bucket
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3api put-bucket-policy --b
ucket my-generative-ai-bucket --policy file://policy.json
>>

Error parsing parameter '--policy': Unable to load paramfile file://policy.json: [Errno 2] No
 such file or directory: 'policy.json'
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws s3api put-bucket-policy --b
ucket my-generative-ai-bucket --policy file://policy.json
>>

An error occurred (AccessDenied) when calling the PutBucketPolicy operation: User: arn:aws:ia
m::960718603264:user/terraform-user is not authorized to perform: s3:PutBucketPolicy on resou
rce: "arn:aws:s3:::my-generative-ai-bucket" because public policies are blocked by the BlockP
ublicPolicy block public access setting.
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Integrate policy with Python app to upload and retrieve generated files

Boto3 allows you to write Python software to control and automate AWS services such as S3, EC2, and Lambda.

pip install boto3

Connect your Python code to the S3 bucket to allows backend to upload generated images/videos

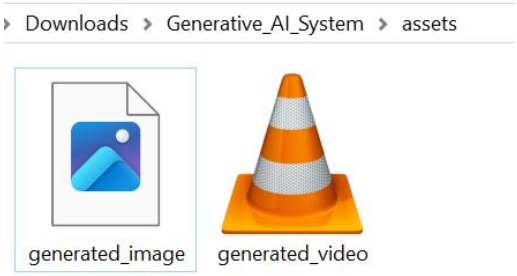
s3_upload.py, generate_image.py

create assets folder under Generative_AI_System

now test

python main.py

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python main.py
assets/generated_image.png uploaded to my-generative-ai-bucket/images/generated_image.png
assets/generated_video.mp4 uploaded to my-generative-ai-bucket/videos/generated_video.mp4
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```



assets/generated_image.png uploaded to my-generative-ai
bucket/images/generated_image.png

S3 integration is working.

Integrate this with main app

A pre-signed URL is a temporary web link that lets anyone access the file without making the bucket public. You generate it using Python.

pre-signed URLs from Python:

print("Image URL:", image_url)
print("Video URL:", video_url)

Replace "YOUR_IMAGE_URL_HERE" and "YOUR_VIDEO_URL_HERE" in the <script> section.

Open index.html in a browser —see uploaded image and video should appear.

call upload_to_s3() to upload the file and gives a temporary URL for frontend.

python main.py

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> python main.py
assets/generated_image.png uploaded to my-generative-ai-bucket/images/generated_image.png
Image URL: https://my-generative-ai-bucket.s3.amazonaws.com/images/generated_image.png?AWS
AccessKeyId=AKIA57L2KQQAHZ2O26RW&Signature=B0EetQtjiArH4yCGmrBJucUdbNI%3D&Expires=17568329
90
assets/generated_video.mp4 uploaded to my-generative-ai-bucket/videos/generated_video.mp4
Video URL: https://my-generative-ai-bucket.s3.amazonaws.com/videos/generated_video.mp4?AWS
AccessKeyId=AKIA57L2KQQAHZ2O26RW&Signature=pBX%2BjWi9JwQbBni1n06pninbsn0%3D&Expires=175683
2990
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Copy the URLs from the console and paste them into index.html for display.

assets/generated_image.png uploaded to my-generative-ai-bucket/images/generated_image.png

Image URL: https://my-generative-ai-bucket.s3.amazonaws.com/images/generated_image.png?AWSAccessKeyId=AKIA57L2KQQAHZ2O26RW&Signature=B0EetQtjiArH4yCGmrBJucUdbNI%3D&Expires=1756832990

assets/generated_video.mp4 uploaded to my-generative-ai-bucket/videos/generated_video.mp4

Video URL: https://my-generative-ai-bucket.s3.amazonaws.com/videos/generated_video.mp4?AWSAccessKeyId=AKIA57L2KQQAHZ2O26RW&Signature=pBX%2BjWi9JwQbBni1n06pninbsn0%3D&Expires=1756832990

index.html > html > body > script

```
2   ıtml>
7   ɔody>
16      <video id="generated-video" width="400" controls>
17          <source id="video-source" src="" type="video/mp4">
18          Your browser does not support the video tag.
19      </video>
20
21      <script>
22          // Replace these with the pre-signed URLs returned from your Python scri
23          const imageUrl = "https://my-generative-ai-bucket.s3.amazonaws.com/image
24          const videoUrl = "https://my-generative-ai-bucket.s3.amazonaws.com/video
25
26          // Set the src attributes dynamically
27          document.getElementById('generated-image').src = imageUrl;
28          document.getElementById('video-source').src = videoUrl;
29
30          // Reload the video element to reflect new source
31          document.getElementById('generated-video').load();
32      </script>
```

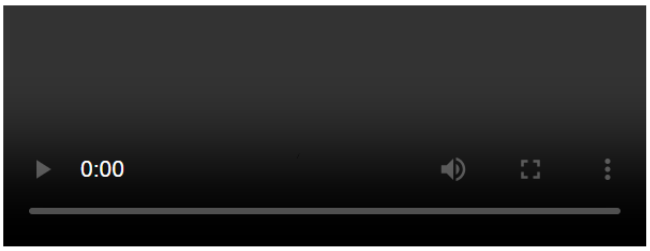
Open index.html in a browser

Uploaded Generative AI Files

Image

Generated Image

Video



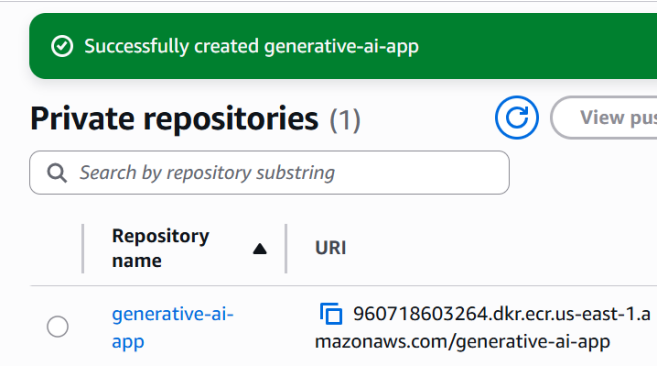
Use Amazon ECS for container orchestration.

Deploy backend services (Flask API, possibly model workers) into ECS so they can run in containers and scale.

Push Image to Amazon ECR

Go to AWS Console → ECR (Elastic Container Registry)**.**

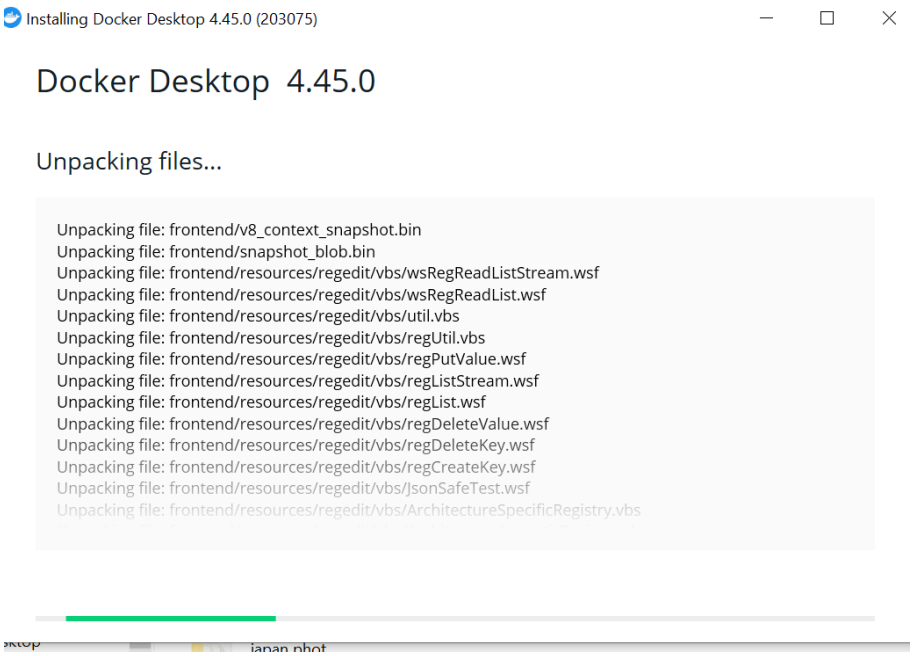
Click Create repository → generative-ai-app.



960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app

Install Docker on Windows

https://www.docker.com/products/docker-desktop/?utm_source=chatgpt.com



Verify Installation:

docker –version

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker --version
>>
Docker version 28.3.3, build 980b856
```

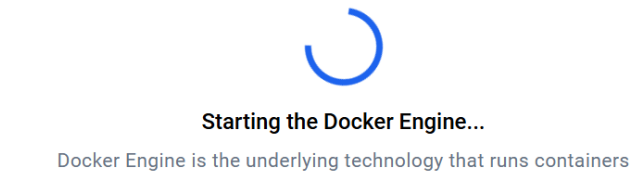
Open Docker Desktop

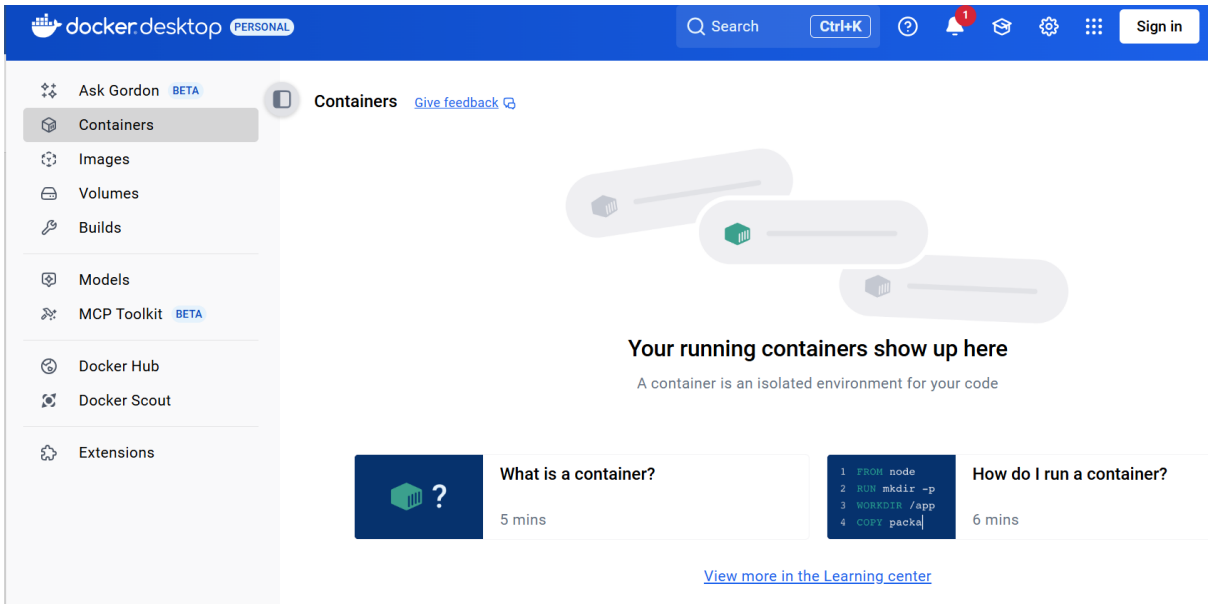
wsl –update

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> wsl --update
Installing: Windows Subsystem for Linux
[==========================53.0%                          ]
```

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> wsl --update
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
```

Click the Restart button in Docker Desktop



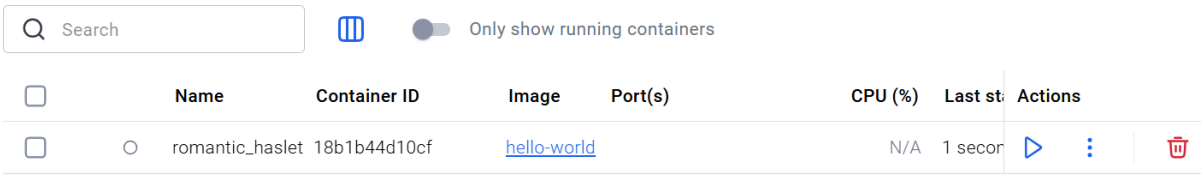


Test Docker

Run a test container:

docker run hello-world

```
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker run hello-world
>>
docker: request returned 500 Internal Server Error for API route and version http://%2F%2F
.%2Fpipe%2FdockerDesktopLinuxEngine/_ping, check if the server supports the requested API
version

Run 'docker run --help' for more information
(venv313) PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
17eec7bbc9d7: Pull complete
Digest: sha256:a0dfb02aac212703bfcb339d77d47ec32c8706ff250850ecc0e19c8737b18567
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
```



inside my Generative_AI_System folder :

Create .dockerignore

To avoid copying huge folders (like venv or node_modules) into the image, which can slow builds and sometimes cause errors

Moved my venv313 folder outside as too large.

inside my Generative_AI_System folder

create utils, models folder and app.py

Build the Docker image locally to make sure it works:

docker build -t generative-ai-app .

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker build -t generative-ai-app .
>>
[+] Building 1197.5s (14/15)                                    docker:desktop-linux
 => [internal] load metadata for docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-ru  2.0s
 => [internal] load .dockerignore                                                    0.0s
 => => transferring context: 125B                                                    0.0s
 => [internal] load build context                                                    0.0s
 => => transferring context: 489B                                                    0.0s
 => [ 1/10] FROM docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-runtime@sha256:e5c  0.1s
 => => resolve docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-runtime@sha256:e5c32  0.1s
 => CACHED [ 2/10] WORKDIR /app                                                      0.0s
 => CACHED [ 3/10] COPY requirements.txt .                                           0.0s
 => CACHED [ 4/10] RUN pip install --no-cache-dir -r requirements.txt                0.0s
 => [ 5/10] COPY requirements_ai.txt .                                               0.1s
 => [ 6/10] RUN pip install --no-cache-dir -r requirements_ai.txt                  378.8s
 => [ 7/10] COPY app.py .                                                            2.5s
 => [ 8/10] COPY utils/ ./utils/                                                     1.0s
 => [ 9/10] COPY models/ ./models/                                                   0.7s
 => [10/10] COPY assets/ ./assets/                                                   0.8s
 => exporting to image                                                             810.9s
 => => exporting layers                                                            544.1s
 => => exporting manifest sha256:b7c7759a330f0276f9b01cf21cc547dbadc4b60d93ae9655ad  0.1s
 => => exporting config sha256:708c888b54adc67f6d0387189bcbe7f065a7275035be892da987  0.1s
 => => exporting attestation manifest sha256:754f283d6bce0e7fb51a0d3d5ec575bf07696b  0.2s
 => => exporting manifest list sha256:4d1789ce585743a98cdcd74cbf7e464533d251db3ed5a  0.1s
```

```
 => => transferring dockerfile: 962B                                          0.0s
 => [internal] load metadata for docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-ru  2.0s
 => [internal] load .dockerignore                                             0.0s
 => => transferring context: 125B                                             0.0s
 => [internal] load build context                                             0.0s
 => => transferring context: 489B                                             0.0s
 => [ 1/10] FROM docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-runtime@sha256:e5c  0.1s
 => => resolve docker.io/pytorch/pytorch:2.2.0-cuda12.1-cudnn8-runtime@sha256:e5c32  0.1s
 => CACHED [ 2/10] WORKDIR /app                                               0.0s
 => CACHED [ 3/10] COPY requirements.txt .                                    0.0s
 => CACHED [ 4/10] RUN pip install --no-cache-dir -r requirements.txt         0.0s
 => [ 5/10] COPY requirements_ai.txt .                                        0.1s
 => [ 6/10] RUN pip install --no-cache-dir -r requirements_ai.txt           378.8s
 => [ 7/10] COPY app.py .                                                     2.5s
 => [ 8/10] COPY utils/ ./utils/                                              1.0s
 => [ 9/10] COPY models/ ./models/                                            0.7s
 => [10/10] COPY assets/ ./assets/                                            0.8s
 => exporting to image                                                      860.9s
 => => exporting layers                                                     544.1s
 => => exporting manifest sha256:b7c7759a330f0276f9b01cf21cc547dbadc4b60d93ae9655ad  0.1s
 => => exporting config sha256:708c888b54adc67f6d0387189bcbe7f065a7275035be892da987  0.1s
 => => exporting attestation manifest sha256:754f283d6bce0e7fb51a0d3d5ec575bf07696b  0.2s
 => => exporting manifest list sha256:4d1789ce585743a98cdcd74cbf7e464533d251db3ed5a  0.1s
 => => naming to docker.io/library/generative-ai-app:latest                   0.3s
 => => unpacking to docker.io/library/generative-ai-app:latest              315.2s
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Docker build succeeded

Run the Docker container locally - to Make sure your app works inside the container.
docker run -it -p 5000:5000 generative-ai-app

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker run -it -p 5000:5000 generative
-ai-app
>>
Hello from Docker!
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Test the backend

After updating your Dockerfile:

docker build -t generative-ai-app .

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker build -t generative-ai-app .

2025/09/04 15:57:30 http2: server: error reading preface from client //./pipe/dockerDesktc
[+] Building 160.0s (3/11)                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.1s
 => => transferring dockerfile: 1.09kB                                              0.1s
 => [internal] load metadata for docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-ru  3.9s
 => [internal] load .dockerignore                                                   0.2s
 => => transferring context: 125B                                                   0.0s
 => [1/7] FROM docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:116  155.6s
 => => resolve docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:11691  0.1s
 => => sha256:42a8919f2ed9b95dfec4eeccf36b869098966254946e137778fbe04025e 99B / 99B  0.4s
 => => sha256:3d552c93e873f61562c7bc8d1e0a9436c890cb44d1a9c97c9a6 1.47GB / 3.62GB  154.8s
 => => sha256:3ad96c6a423ac845ca1c88befd37f33e5cee0f9c46f6a8c8e7300 7.23MB / 7.23MB  2.0s
 => => sha256:d66d6a6a368713979f9d00fad193991ae1af18b8efd3abf4d70 30.45MB / 30.45MB  4.3s
 => => extracting sha256:d66d6a6a368713979f9d00fad193991ae1af18b8efd3abf4d70ade1928  6.9s
```

		Name	Container ID	Image	Port(s)	CPU (%)	Last st	Actions
☐	○	vibrant_lichterm	7db13bca69a4	generative-;	5000:5000	N/A	7 secor	▷ ⋮

```
[+] Building 2023.4s (11/12)                                        docker:desktop-linux
 => => sha256:3d552c93e873f61562c7bc8d1e0a9436c890cb44d1a9c97c9a 3.62GB / 3.62GB  2018.2s
 => => extracting sha256:3d552c93e873f61562c7bc8d1e0a9436c890cb44d1a9c97c9a69b864  268.2s
 => => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38  0.4s
 => => extracting sha256:42a8919f2ed9b95dfec4eeccf36b869098966254946e137778fbe04025  0.4s
 => [internal] load build context                                                  482.2s
 => => transferring context: 3.25GB                                                466.8s
 => [2/7] WORKDIR /app                                                               3.3s
 => [3/7] COPY requirements.txt .                                                    0.8s
 => [4/7] RUN pip install --no-cache-dir -r requirements.txt                        36.6s
 => [5/7] COPY requirements_ai.txt .                                                 3.1s
 => [6/7] RUN pip install --no-cache-dir -r requirements_ai.txt                     96.2s
 => [7/7] COPY . .                                                                 279.9s
 => exporting to image                                                             856.8s
 => => exporting layers                                                            551.1s
 => => exporting manifest sha256:47ce8c52a838d130ae69d34ceb318b1487b8c5beaf28972c0a  0.2s
 => => exporting config sha256:b9c1922c67bdd0766a3aff8947edac6d1e2b79f9e51727486601  0.2s
 => => exporting attestation manifest sha256:65266036b7b2d1b26c0d7b1f62386cd3dfee42  0.8s
 => => exporting manifest list sha256:f138c28d15f84ce52d13e051230182c5cdc62fb12ac96  0.2s
 => => naming to docker.io/library/generative-ai-app:latest                          0.3s
 => => unpacking to docker.io/library/generative-ai-app:latest                     303.5s

=> exporting to image                                                             1070.0s
=> => exporting layers                                                             551.1s
=> => exporting manifest sha256:47ce8c52a838d130ae69d34ceb318b1487b8c5beaf28972c0a  0.2s
=> => exporting config sha256:b9c1922c67bdd0766a3aff8947edac6d1e2b79f9e51727486601  0.2s
=> => exporting attestation manifest sha256:65266036b7b2d1b26c0d7b1f62386cd3dfee42  0.8s
=> => exporting manifest list sha256:f138c28d15f84ce52d13e051230182c5cdc62fb12ac96  0.2s
=> => naming to docker.io/library/generative-ai-app:latest                          0.3s
=> => unpacking to docker.io/library/generative-ai-app:latest                     516.4s
```

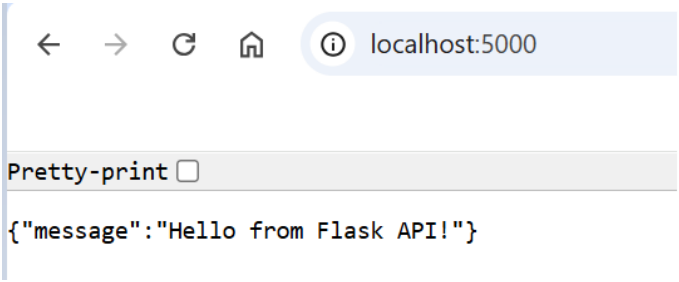
docker build -t generative-ai-app .

Run container:

docker run -p 5000:5000 generative-ai-app

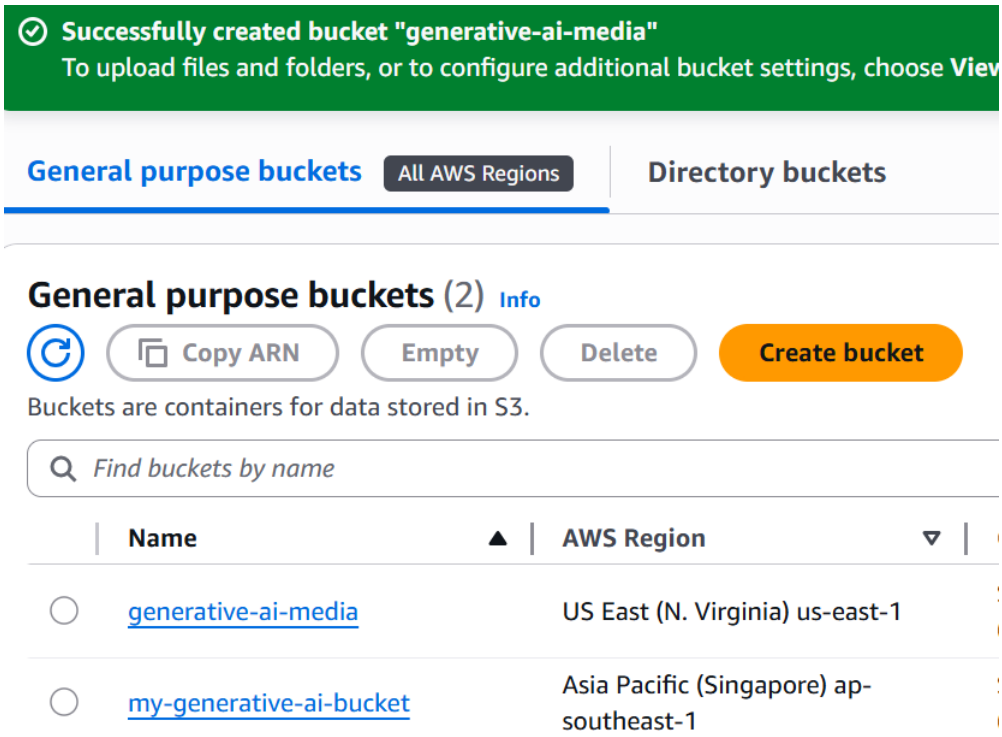
```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker run -p 5000:5000 generative-ai-
app
 * Serving Flask app 'backend/app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a pro
duction WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

Open in browser:
http://localhost:5000



API working.

AWS Management Console → S3 → Create bucket.



generative-ai-media - US East (N. Virginia) us-east-1

my-generative-ai-bucket- Asia Pacific (Singapore) ap-southeast-1

S3_BUCKET = generative-ai-media

S3_BUCKET = S3 bucket to use when it uploads or downloads files.

On Windows (PowerShell):

$env:S3_BUCKET = "generative-ai-media"
python app.py

Building the Docker image, pushing it to ECR, and deploying it to ECS (Fargate)

make app live in the cloud, connected to S3 for storage.

Prepare Docker Image

Start Menu → Docker Desktop →

Run as Administrator to initialize WSL 2 backend properly.

Check Docker status: docker version

Shows Docker is running.

Build Docker image:

docker build -t generative-ai-app .

```
=> [internal] load metadata for docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-ru  4.2s
=> [internal] load .dockerignore                                                    0.0s
=> => transferring context: 102B                                                    0.0s
=> [1/5] FROM docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:11691  0.4s
=> => resolve docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:11691  0.4s
=> [internal] load build context                                                    4.8s
=> => transferring context: 3.26MB                                                  4.2s
=> CACHED [2/5] WORKDIR /app                                                        0.0s
=> CACHED [3/5] COPY backend/requirements.txt /app/                                 0.0s
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                  0.0s
=> CACHED [5/5] COPY backend /app/backend                                           0.0s
=> exporting to image                                                               0.2s
=> => exporting layers                                                              0.0s
=> => exporting manifest sha256:4de211c78b90e89bf720fc9c523cf591b2ebcaae12fa1b7224  0.0s
=> => exporting config sha256:a07b5b298c02a01ca8f69c3df09db0794efa7bdc79fd6c603673  0.0s
=> => exporting attestation manifest sha256:4e06d025b3851050ef3b880937e24701dab1cc  0.1s
=> => exporting manifest list sha256:bb4645e00733ec992b37ccf78274afdfe540cc2f070df  0.0s
=> => naming to docker.io/library/generative-ai-app:latest                          0.0s
=> => unpacking to docker.io/library/generative-ai-app:latest                       0.0s
```

Test locally:

docker run -p 5000:5000 generative-ai-app

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker run -p 5000:5000 generative-ai-
app
>>
 * Serving Flask app 'backend/app.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a pro
duction WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

Push Docker Image to ECR

Authenticate Docker with ECR**:**

aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin

< aws_account_id>.dkr.ecr.us-east-1.amazonaws.com

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> aws ecr get-login-password --region us
-east-1 | docker login --username AWS --password-stdin 9[redacted].dkr.ecr.us-east-1.amaz
onaws.com
Login Succeeded
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

push image to ECR:

AWS Console → ECR (Elastic Container Registry)

Repository name: generative-ai-app

Region: us-east-1 (same as ECS)

repository URL like:

960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app

Tag your Docker image

So Docker knows where to push:

docker tag generative-ai-app:latest 960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest

Push the image to ECR

docker push 123456789012.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker push 123456789012.dkr.ecr.us-ea
st-1.amazonaws.com/generative-ai-app:latest
The push refers to repository [123456789012.dkr.ecr.us-east-1.amazonaws.com/generative-ai-
app]
tag does not exist: 123456789012.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

Fix the tagging

docker tag generative-ai-app:latest 960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest

Push again

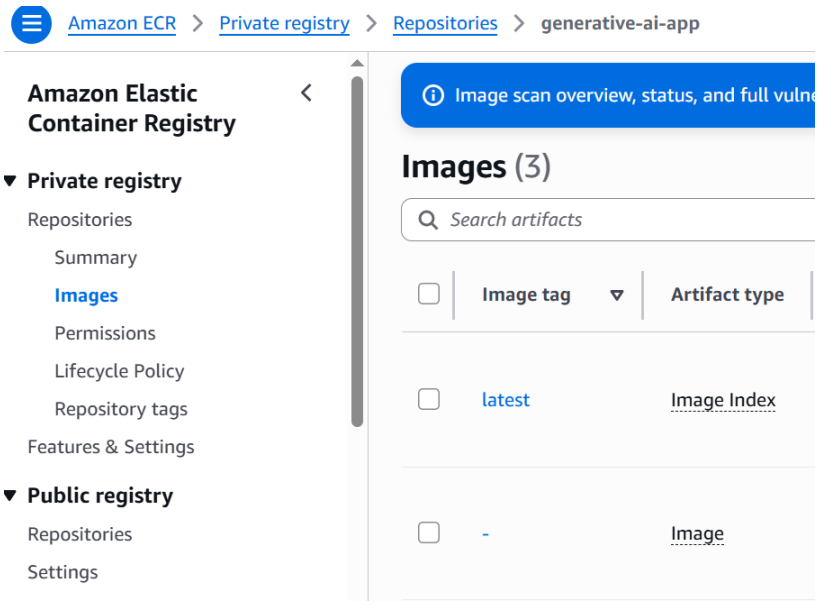
docker push 960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker push 960718603264.dkr.ecr.us-ea
st-1.amazonaws.com/generative-ai-app:latest
The push refers to repository [960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-
app]
4f4fb700ef54: Pushed
42a8919f2ed9: Pushed
f4940f41e93c: Pushed
e5e2f0b78cf1: Pushed
27eab9a661b6: Pushed
3d552c93e873: Pushed
a4fabb5a85c5: Pushed
f35558b59226: Pushed
d66d6a6a3687: Pushed
3ad96c6a423a: Pushed
latest: digest: sha256:bb4645e00733ec992b37ccf78274afdfe540cc2f070dfc1db56863eaf0853b72 si
ze: 856
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

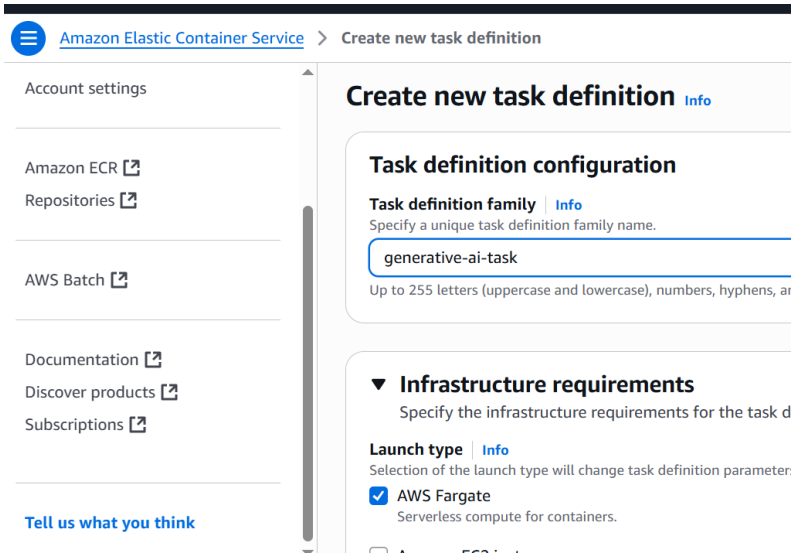
Verify

AWS Console → ECR → Repositories → generative-ai-app

see your image with the latest tag.



AWS Console → ECS → Task Definitions → Create new task definition



Task size:

CPU: 0.5 vCPU

Memory: 1 GB

Add container

Container name → generative-ai-container

Image URI**:** 123456789012.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app:latest

Port mappings: Container port: 5000

Protocol: TCP

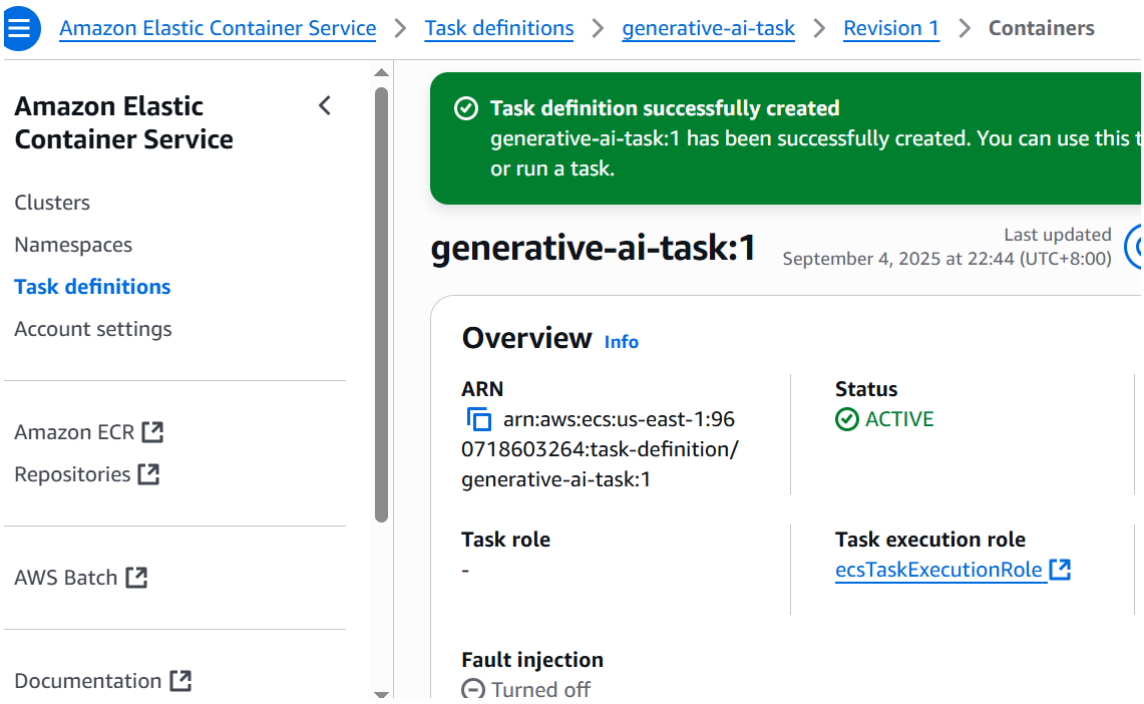
Add environment variables

Environment variables → Add:

Key: S3_BUCKET

Value: generative-ai-media

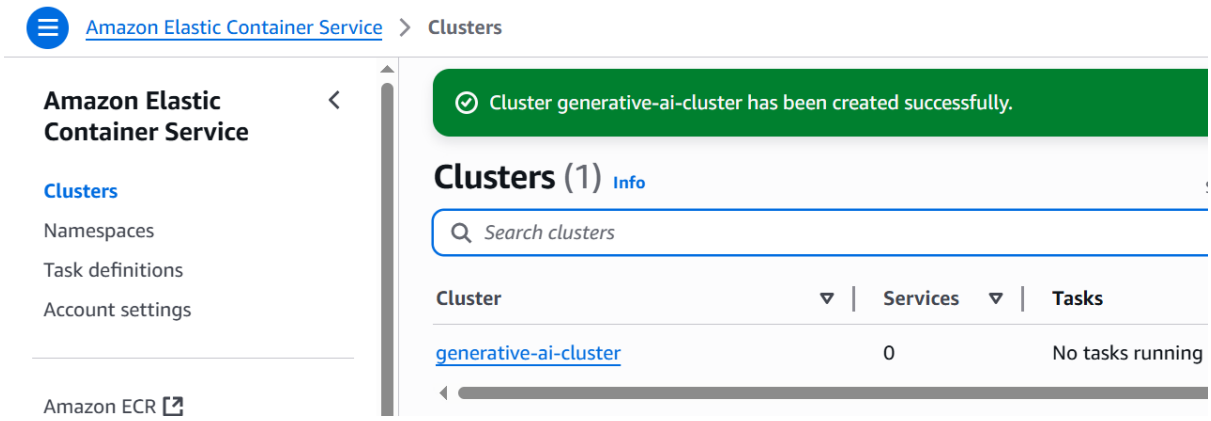
Create task definition:



Task Definition **(**generative-ai-task:1**)** inside ECS Cluster + Service so my app is live

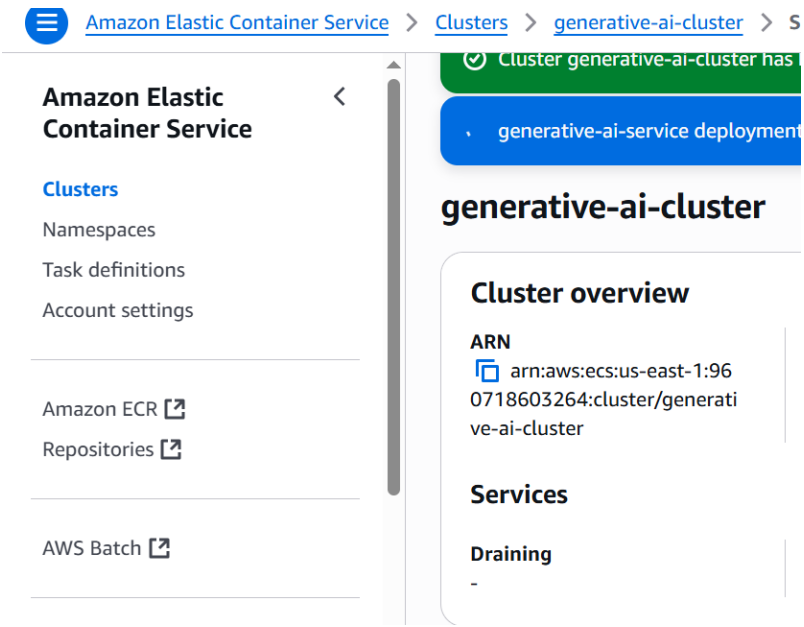
AWS Console → ECS → Clusters → Create Cluster

Cluster name: generative-ai-cluster



Create an ECS Service that runs your task (generative-ai-task:1) on Fargate.

ECS → Clusters → generative-ai-cluster → Services → Create



Identifies your ECS cluster**:**

arn:aws:ecs:us-east-1:960718603264:cluster/generative-ai-cluster

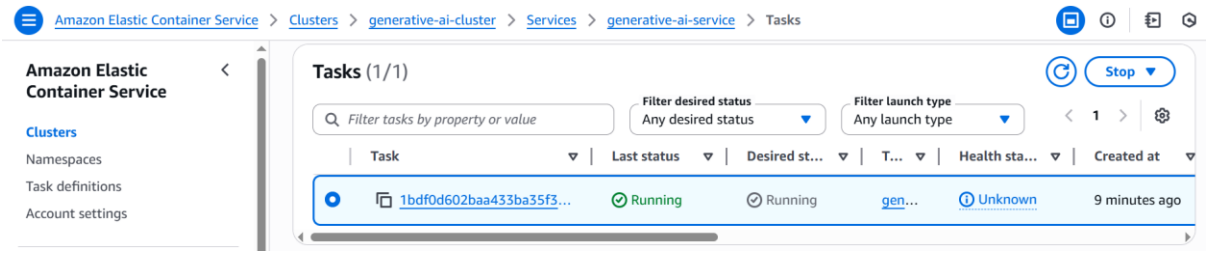
Create a Service in this cluster so the task definition (generative-ai-task:1) actually runs.

ECS → Clusters → generative-ai-cluster

Create → Service

Service name**:** generative-ai-service-v2

Check status: Cluster → generative-ai-cluster → Tasks



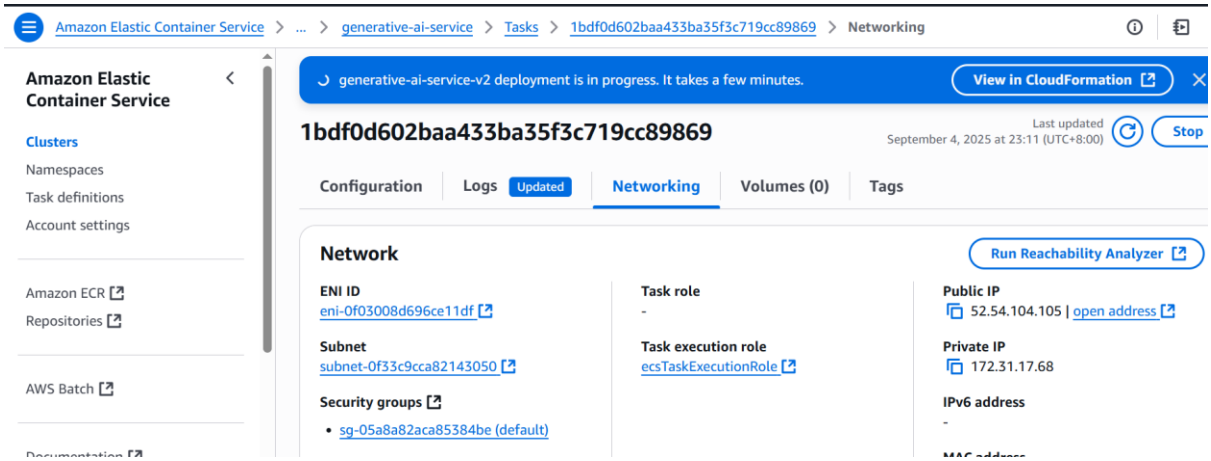
Task status is RUNNING

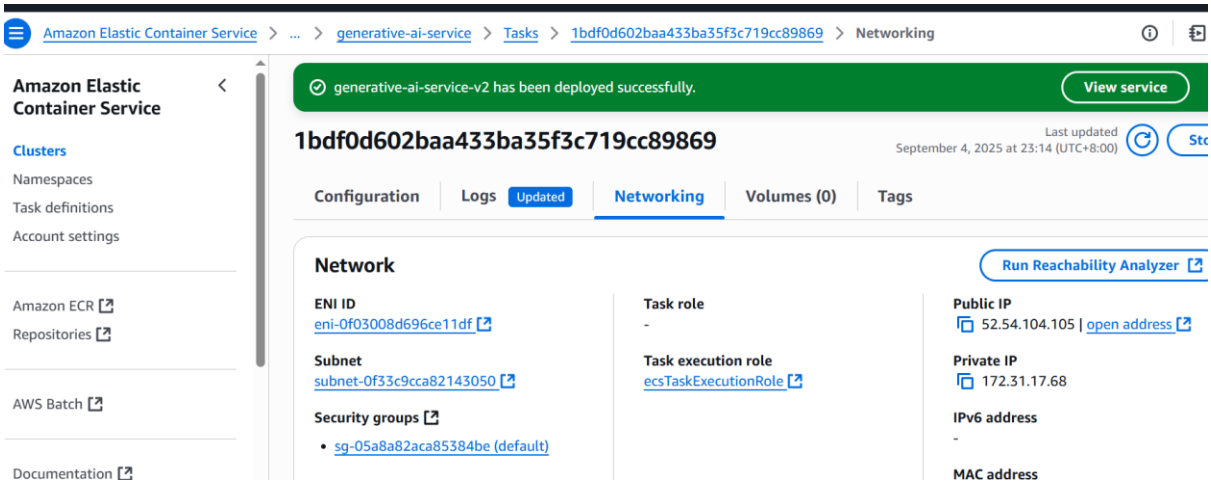
Get Public IP:

- Click on the running task → Networking → ENI → Public IP

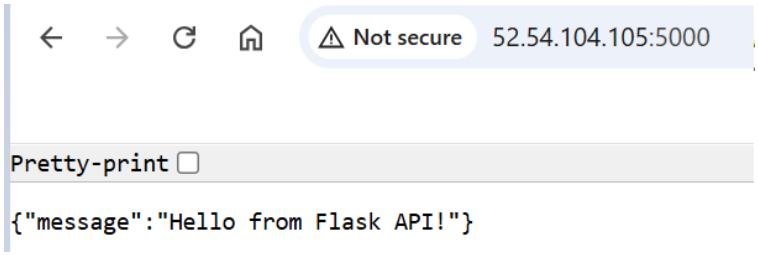
- Open in browser:

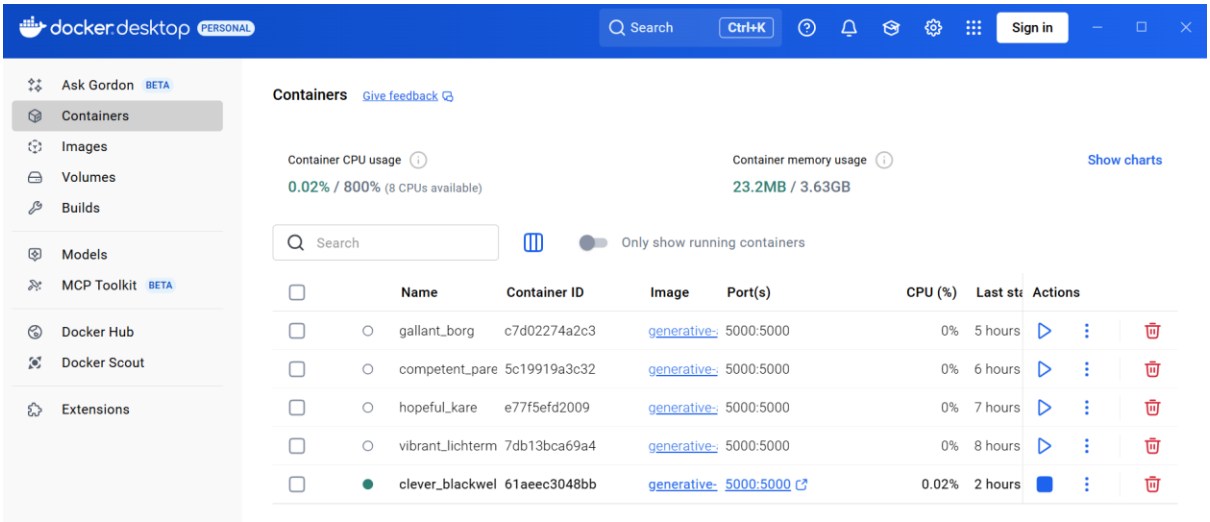
  http://52.54.104.105/





app is accessible at: http://52.54.104.105:5000





Test S3 uploads from app to make sure container can read/write to bucket

Add a simple S3 test route in Flask app

Rebuild Docker image:

docker build -t nhoiyee/generative-ai-app .

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker build -t nhoiyee/generative-ai-
app .
>>
[+] Building 5.2s (11/11) FINISHED                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.0s
 => => transferring dockerfile: 472B                                                0.0s
 => [internal] load metadata for docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-ru  1.9s
 => [auth] pytorch/pytorch:pull token for registry-1.docker.io                      0.0s
 => [internal] load .dockerignore                                                   0.1s
 => => transferring context: 102B                                                   0.0s
 => [1/5] FROM docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:11691  0.1s
 => => resolve docker.io/pytorch/pytorch:2.2.1-cuda12.1-cudnn8-runtime@sha256:11691  0.1s
 => [internal] load build context                                                   2.5s
 => => transferring context: 3.26MB                                                 2.1s
 => CACHED [2/5] WORKDIR /app                                                       0.0s
 => CACHED [3/5] COPY backend/requirements.txt /app/                                0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                 0.0s
 => CACHED [5/5] COPY backend /app/backend                                          0.0s
 => exporting to image                                                              0.3s
 => => exporting layers                                                             0.0s
 => => exporting manifest sha256:fea1b55afb209c5841985cb1609b79843bcd7c6ade953f5f4d  0.0s
```

Then push again:

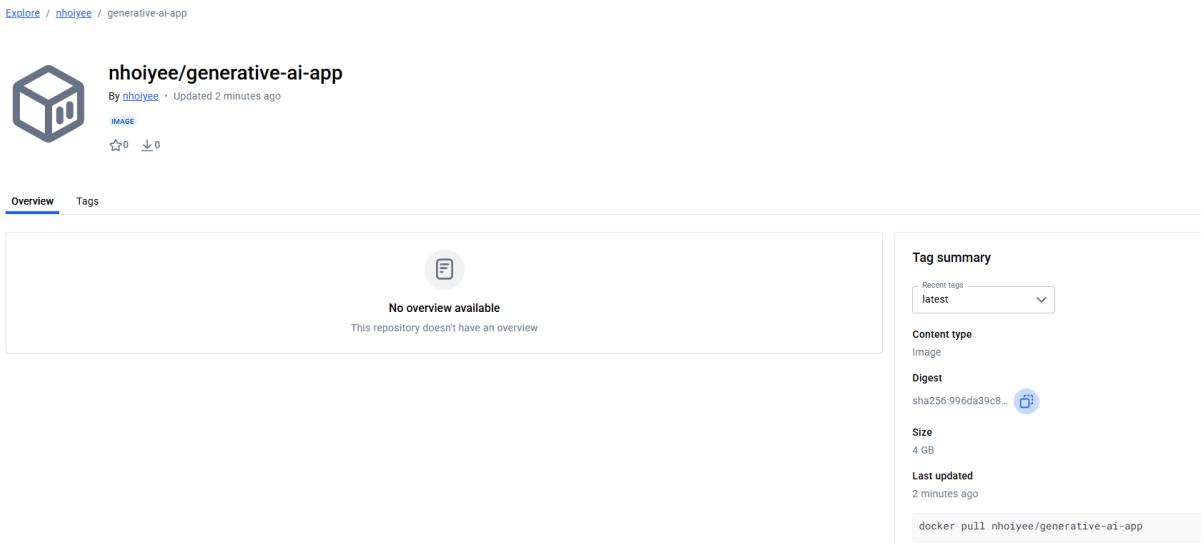
docker push nhoiyee/generative-ai-app

```
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System> docker push nhoiyee/generative-ai-app
Using default tag: latest
The push refers to repository [docker.io/nhoiyee/generative-ai-app]
3ad96c6a423a: Pushing  2.097MB/7.232MB
a4fabb5a85c5: Pushed
d66d6a6a3687: Pushing  1.049MB/30.45MB
b2fc22e9ab62: Pushing  2.097MB/397MB
4f4fb700ef54: Pushed
f35558b59226: Pushing  2.097MB/270.7MB
e5e2f0b78cf1: Pushed
a6034e3a92cb: Pushed
42a8919f2ed9: Pushed
3d552c93e873: Pushing  3.146MB/3.622GB
```

```
a6034e3a92cb: Pushed
42a8919f2ed9: Pushed
3d552c93e873: Pushed
latest: digest: sha256:996da39c8793cd51a9f10a43b42124629eeb3e104dae03e30391b88e609ff293 si
ze: 856
PS C:\Users\Hoi Yee\Downloads\Generative_AI_System>
```

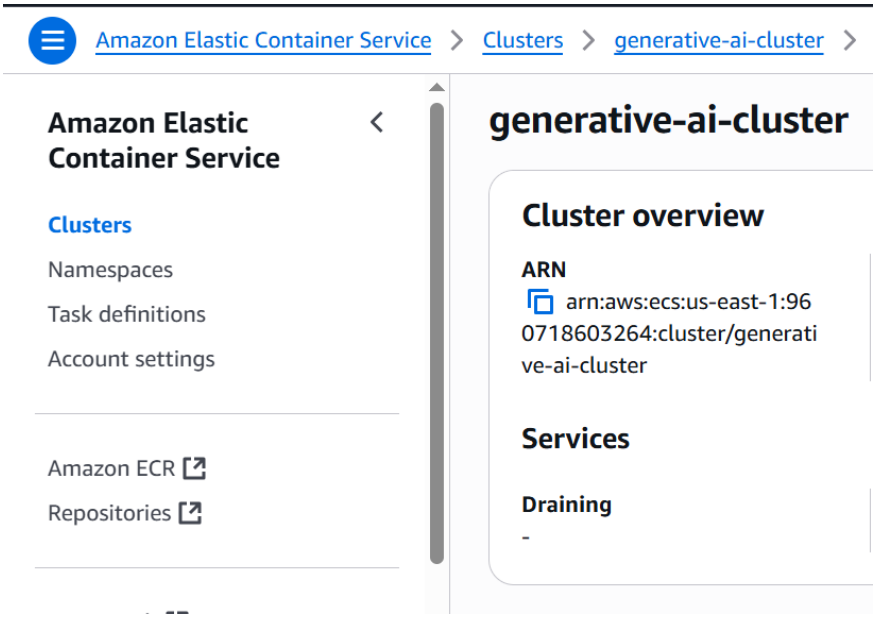
Took a while. 10 minutes.

After that, the image shows on Docker Hub repo page:
https://hub.docker.com/r/nhoiyee/generative-ai-app



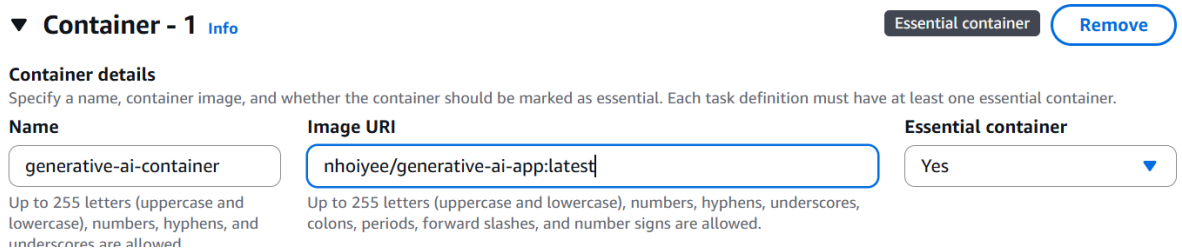
Update ECS to use this new image

ECS to pull this image

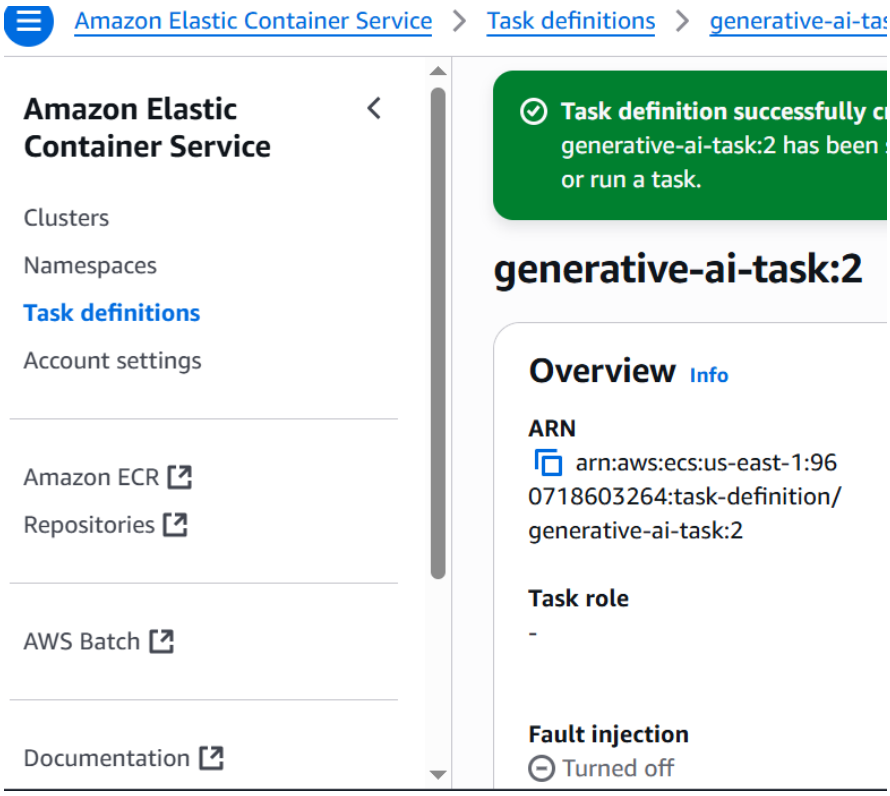


Update Task Definition

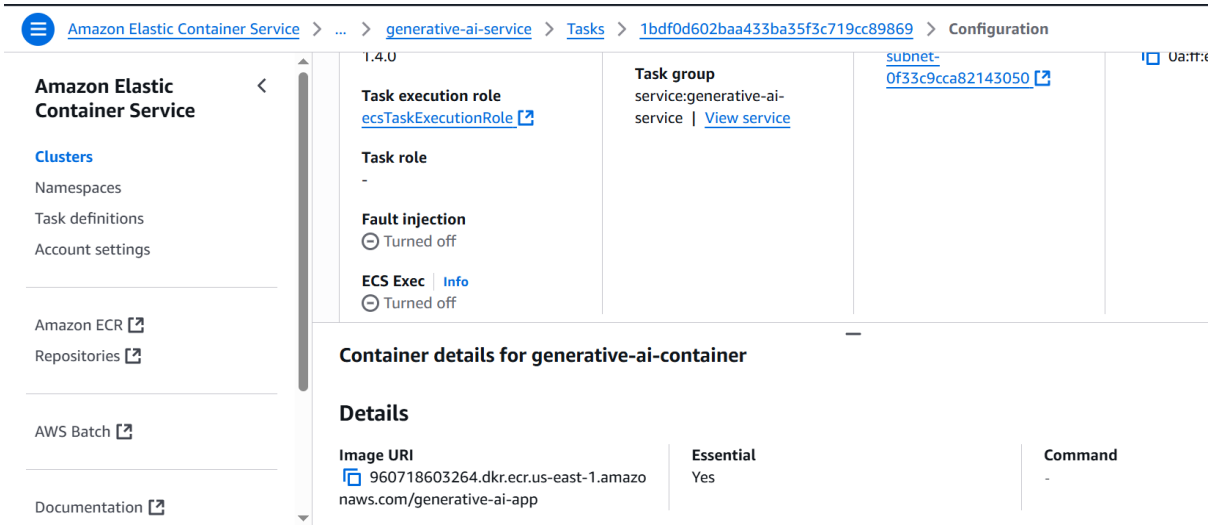
Image URI : change to:
nhoiyee/generative-ai-app:latest



Click Create



Test S3 Upload
Once the new container is running, test my S3 upload route:



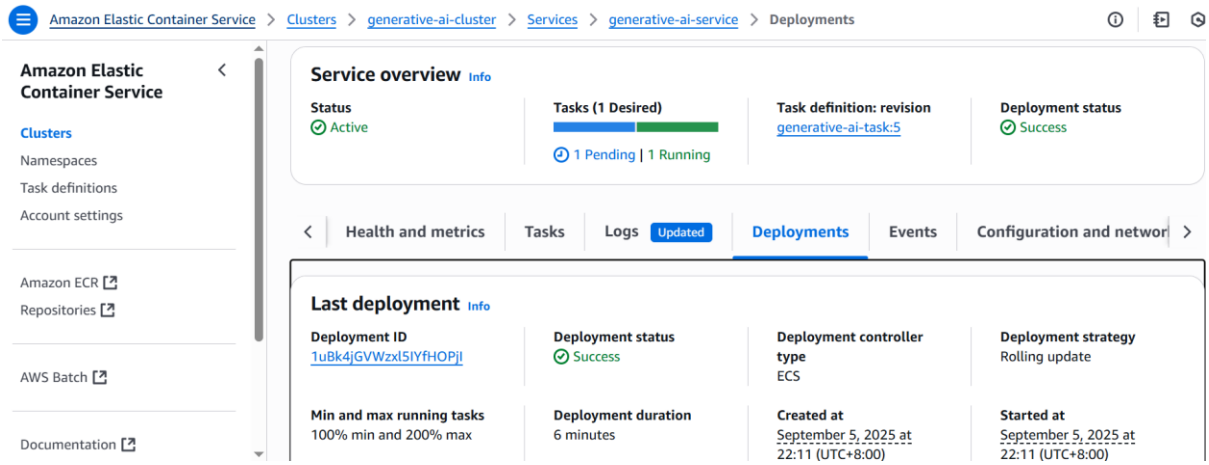
Need to change this
960718603264.dkr.ecr.us-east-1.amazonaws.com/generative-ai-app
To
nhoiyee/generative-ai-app:latest

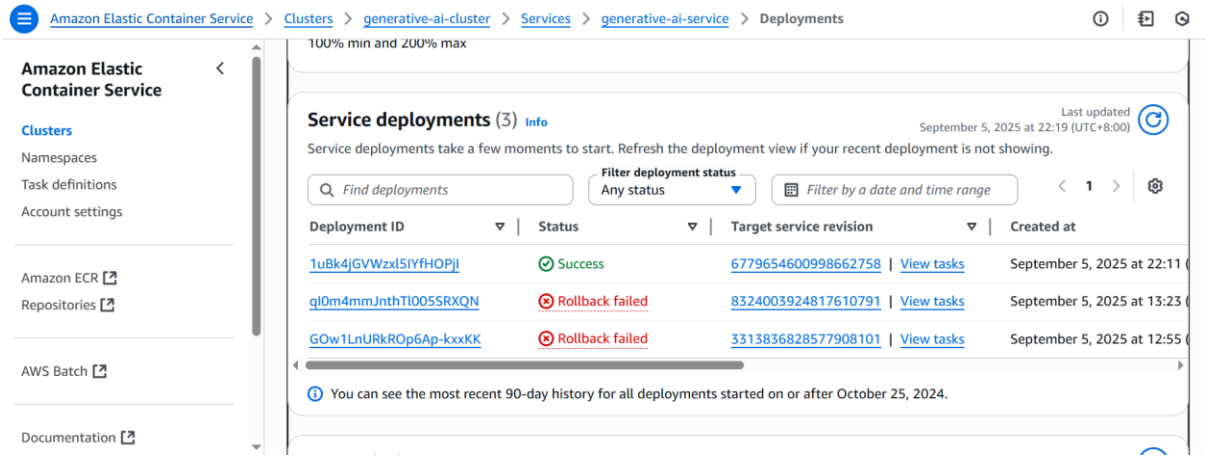
make /test-s3-upload work on ECS

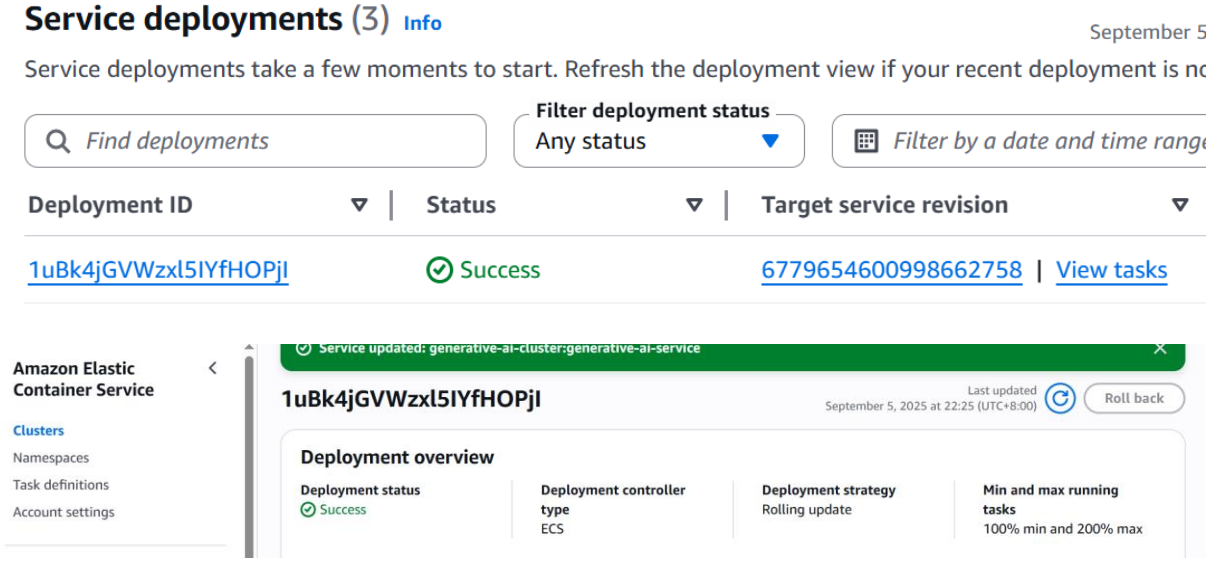
docker build and docker push:
(Docker image: nhoiyee/generative-ai-app:latest)

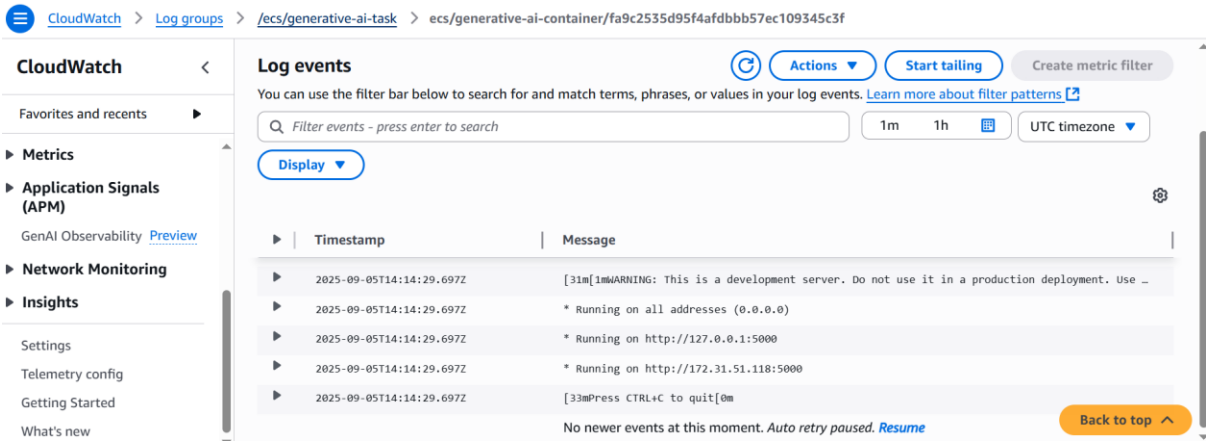
docker buildx build --push -t nhoiyee/generative-ai-app:latest .

Update Service → Run Task









▶	2025-09-05T14:14:29.697Z	* Running on all addresses (0.0.0.0)
▶	2025-09-05T14:14:29.697Z	* Running on http://127.0.0.1:5000
▶	2025-09-05T14:14:29.697Z	* Running on http://172.31.51.118:5000
▶	2025-09-05T14:14:29.697Z	[33mPress CTRL+C to quit[0m

ECS task is now running successfully.

Step 7 - Deployment is done

8-Observability:
□ Amazon CloudWatch:

- Set up monitoring, logging, and alerting.

□ AWS X-Ray:

- Implement tracing for request tracking.

□ SLI/SLO/SLA Indicators:

- Define and monitor service levels, performance objectives, and agreements.

Amazon CloudWatch

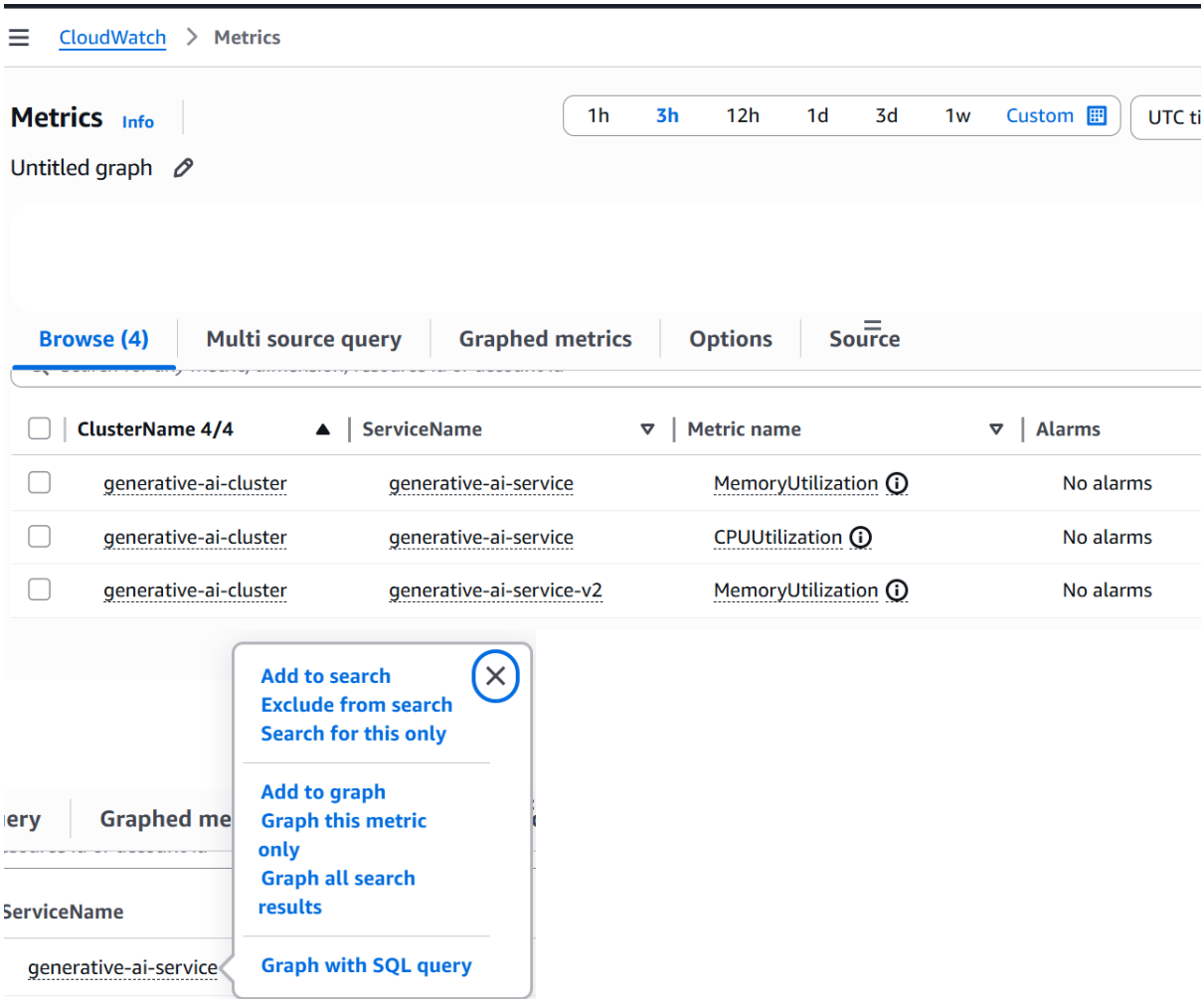
Objective: Set up monitoring, logging, and alerting.

Logs (already partly done):

- CloudWatch is already collecting ECS container logs (/ecs/generative-ai-task).
- Confirm logs show Flask app running successfully.

Metrics / Monitoring:

- Go to CloudWatch → Metrics → ECS
- Monitor CPUUtilization, MemoryUtilization, RunningTaskCount.

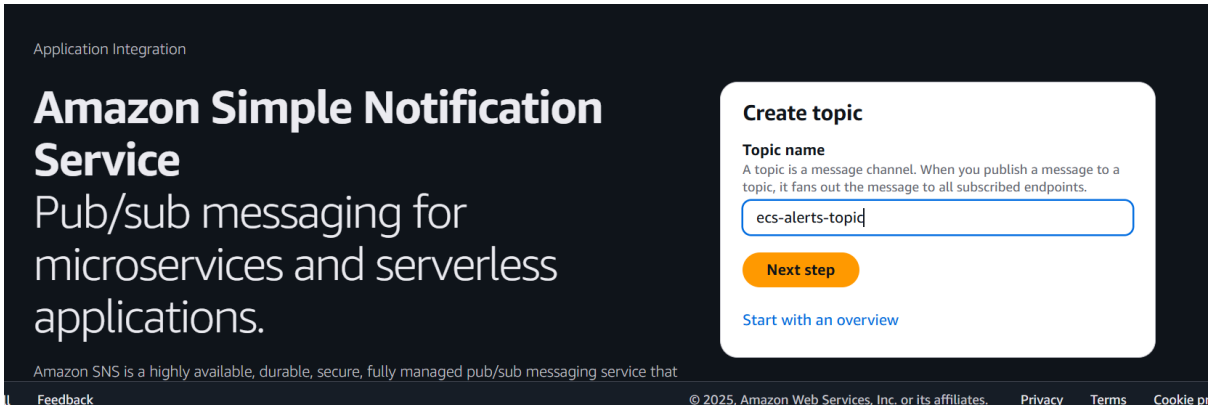


Alarms:

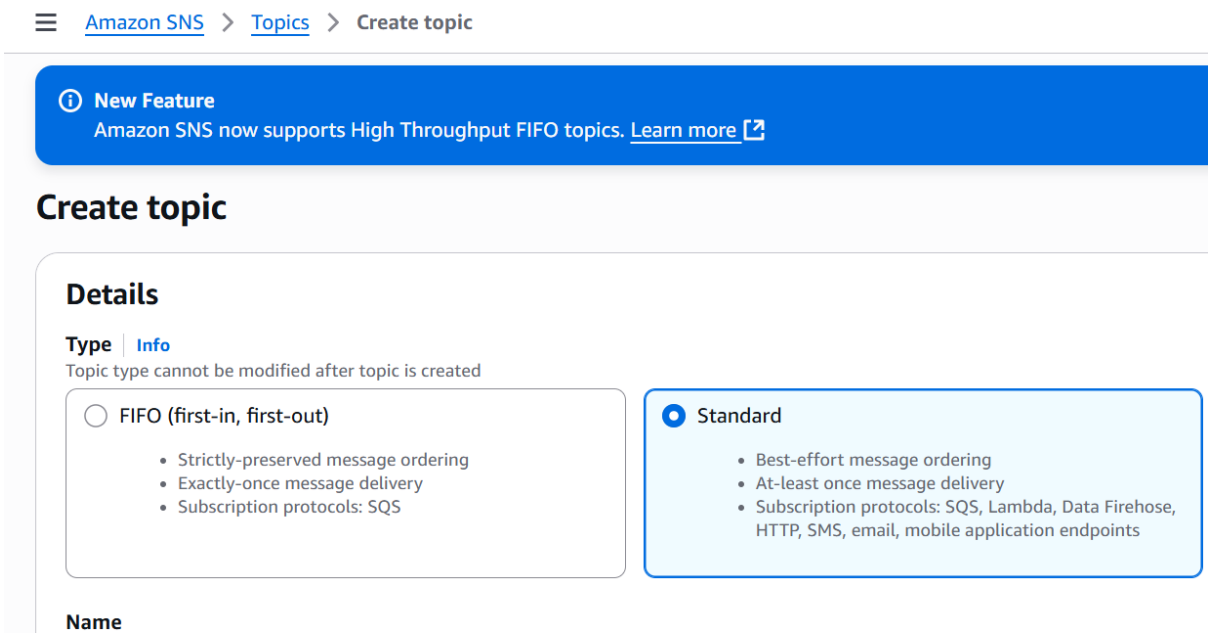
- Create alarms for thresholds, e.g.:
  - CPU > 80% → alert
  - Memory > 80% → alert
- Use SNS to send email alerts (optional but good practice).

Create an SNS Topic (for alerts)

Go to AWS Console → SNS → Topics.

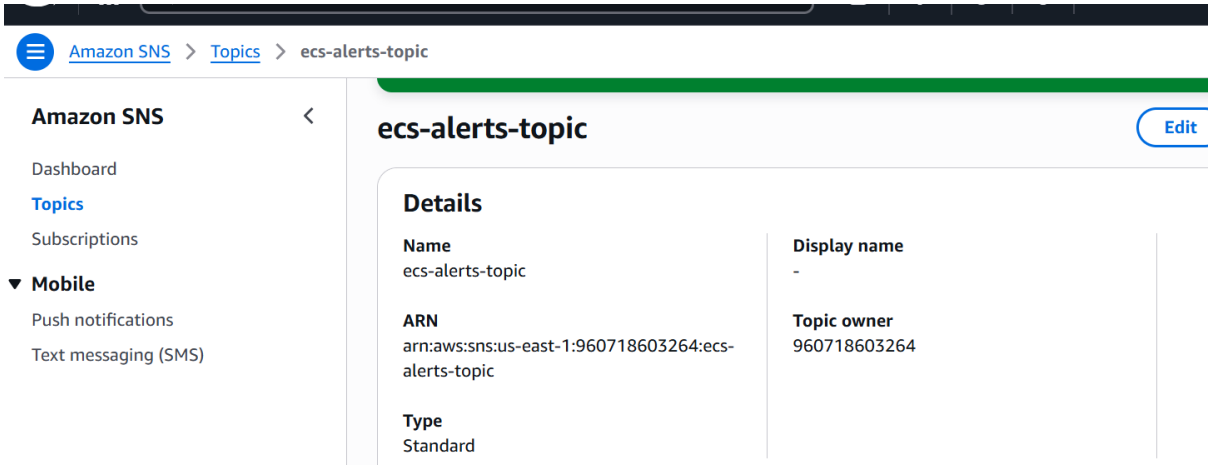


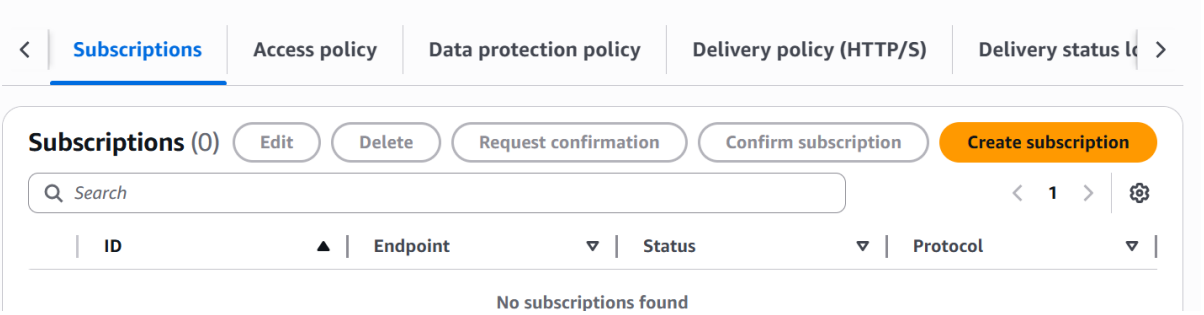
Name it : ecs-alerts-topic.



Click Create topic → choose Standard**.**

Click Create topic**.**

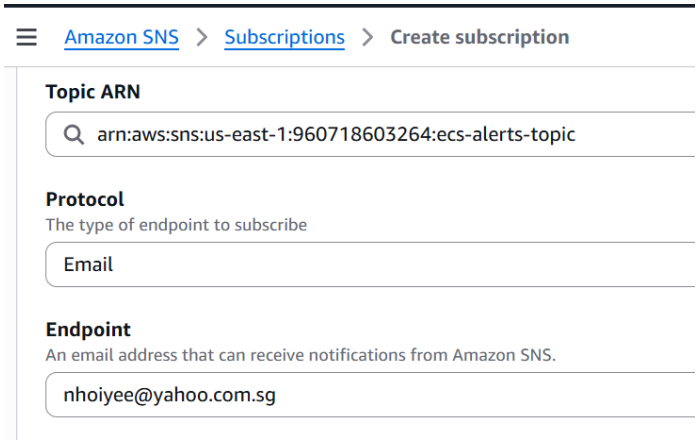




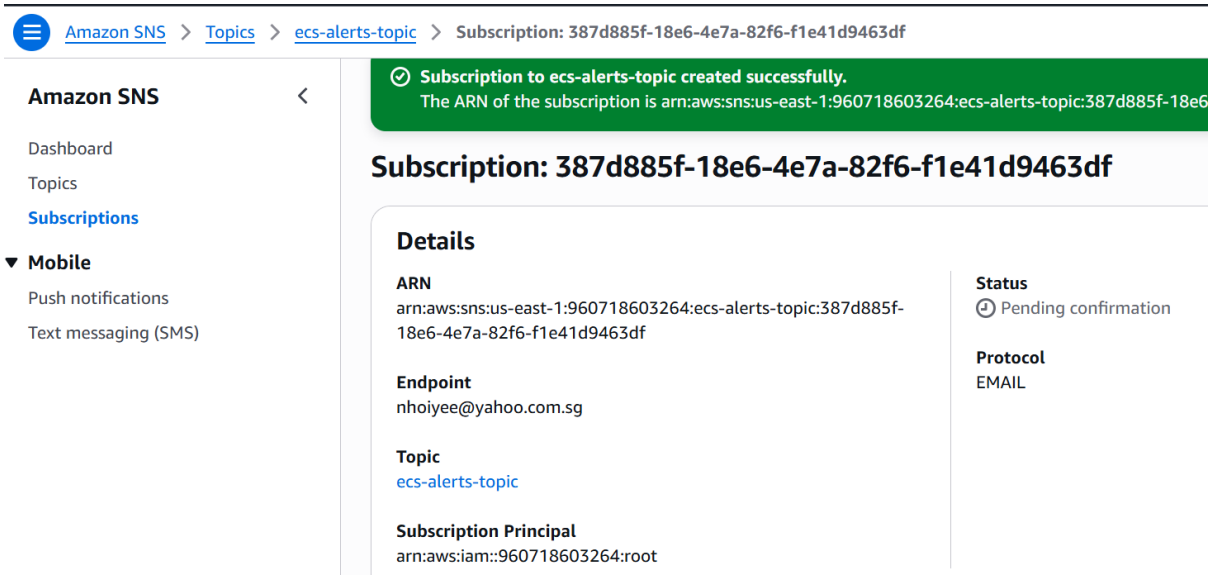
Click Create subscription**.**

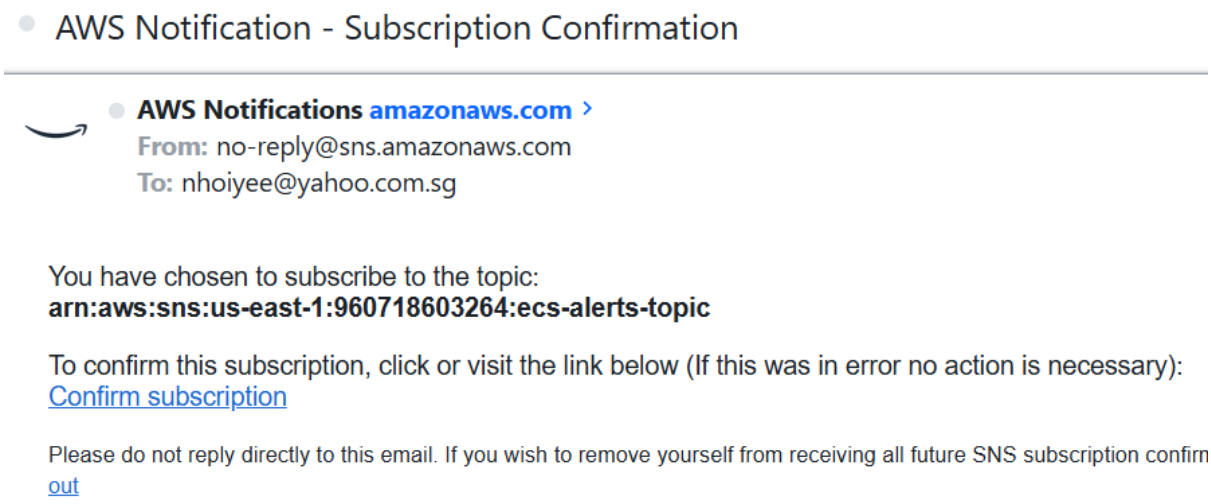
Protocol: Email

Endpoint: your email address



Click Create subscription → you'll receive a confirmation email.





Confirm the subscription by clicking the link in the email.

AWS X-Ray (Request Tracing)

Objective: Implement tracing to track requests through your service.

Enable X-Ray in ECS Task

Go to ECS → Task Definitions → generative-ai-task → Create new revision JSON**.**

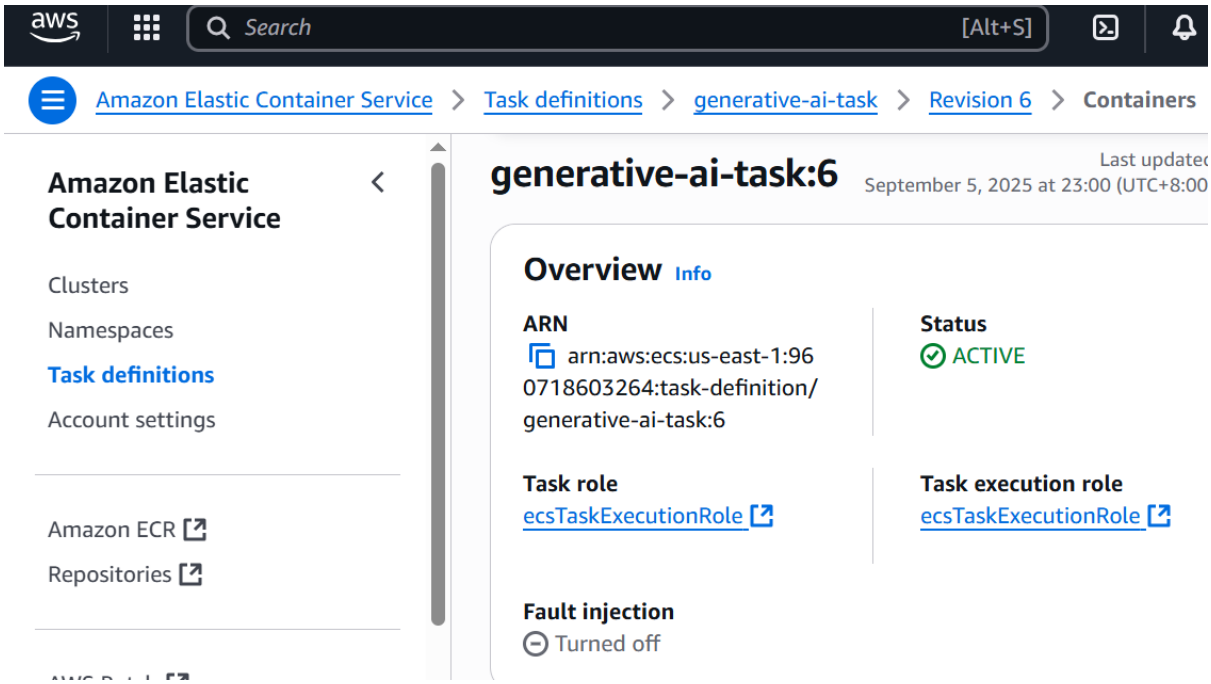
Add a new container definition like this inside the "containerDefinitions" array alongside your Flask container**:**

```
{
  "name": "xray-daemon",
  "image": "amazon/aws-xray-daemon",
  "essential": false,
  "memory": 256,
  "portMappings": []
}
```

In your Flask container definition, add environment variable:

```
{
  "name": "AWS_XRAY_DAEMON_ADDRESS",
  "value": "xray-daemon:2000"
}
```

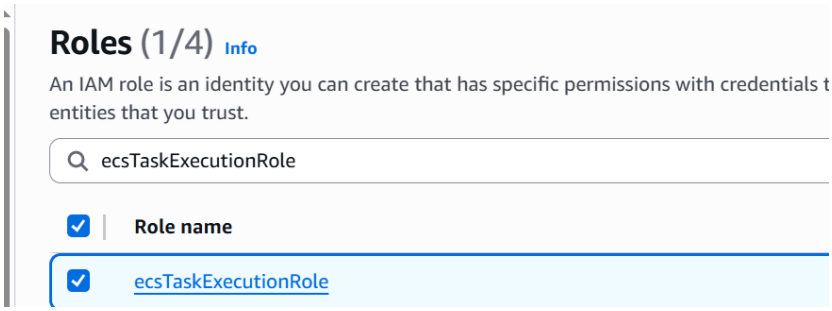
Click **Create** → new revision is created.



task role - ecsTaskExecutionRole.

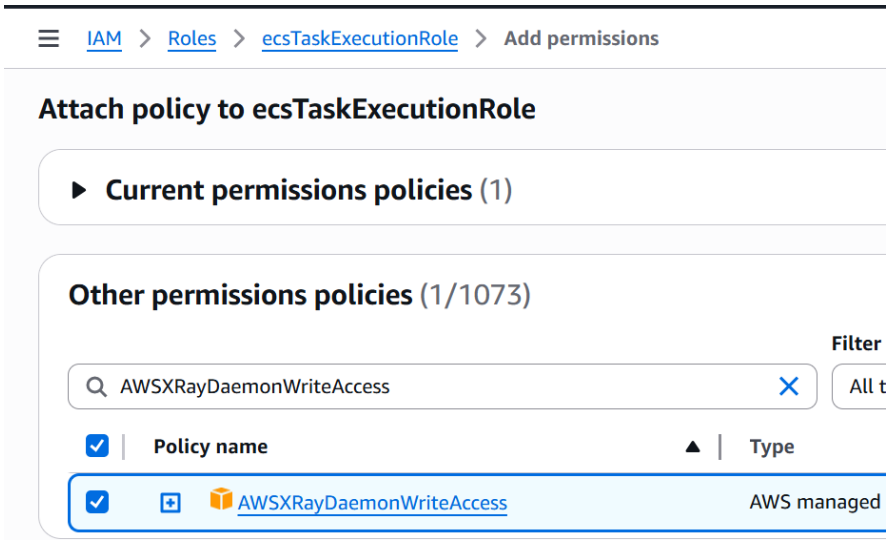
To enable X-Ray:

Open **AWS Console → IAM → Roles**.

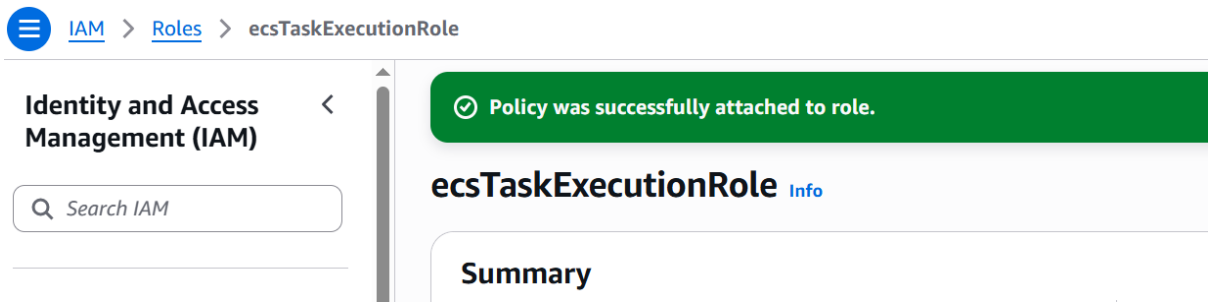


Attach the X-Ray Policy

1. Click Add permissions → Attach policies.
2. Search for AWSXRayDaemonWriteAccess.



Check the box → click Attach policy**.**



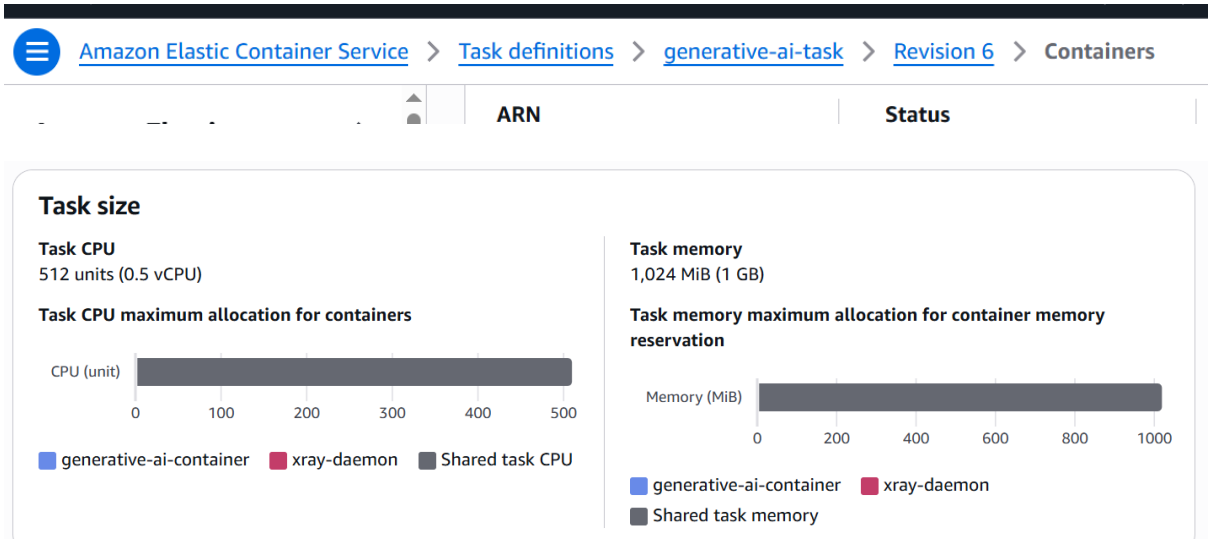
Now this role has permission to write traces to X-Ray.

Flask app can now send traces to AWS X-Ray**,** and verify them in X-Ray → Service Map

SLI/SLO/SLA Indicators

Objective: Define and monitor service performance using CloudWatch metrics and alarms.

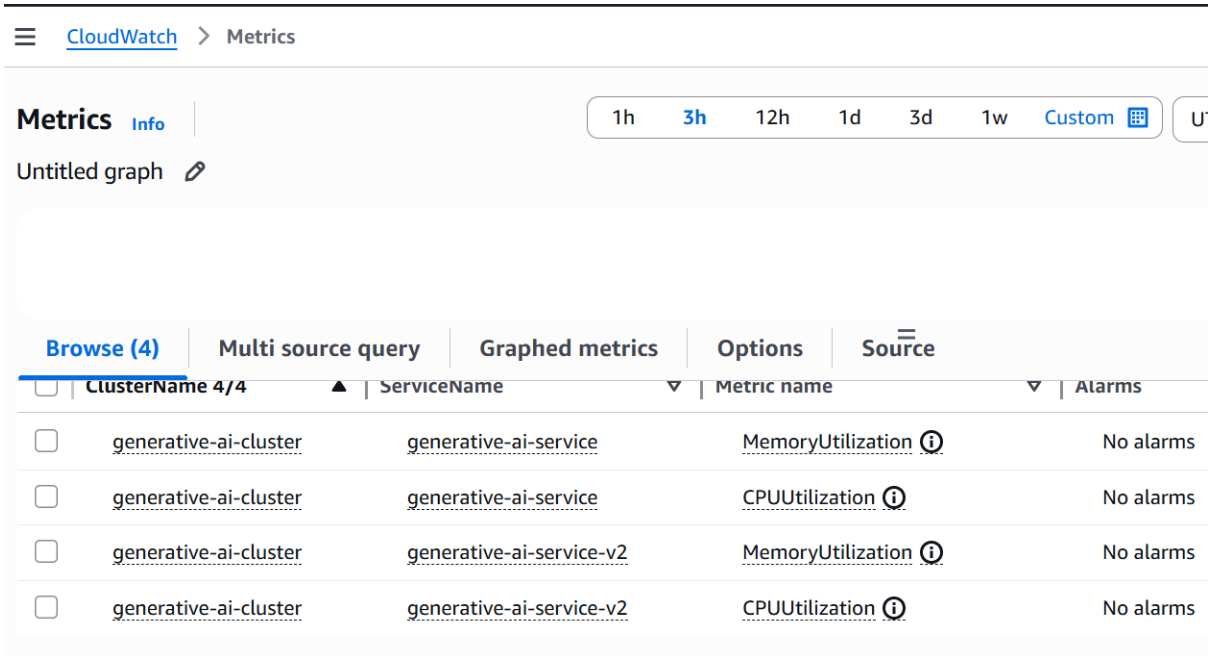
Define Your SLI (Service Level Indicator)



Set SLO (Service Level Objective)

Steps in CloudWatch:

Go to CloudWatch → Metrics → ECS → Per-Service Metrics.



Check CPUUtilization**,** MemoryUtilization**,** RunningTaskCount**.**

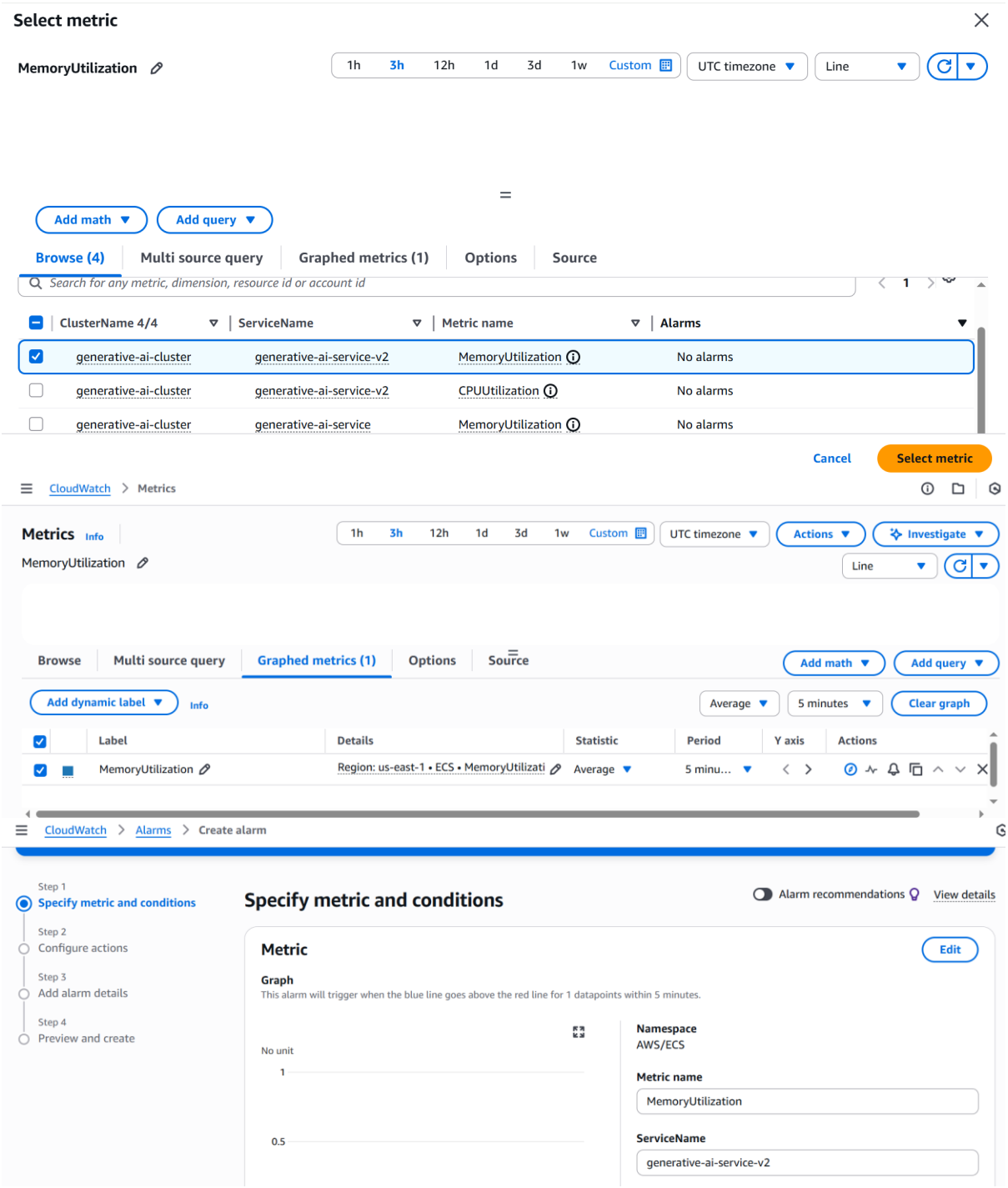
Use these metrics to define thresholds:

- CPU < 80%
- Memory < 80%
- RunningTaskCount = desired task count

Create CloudWatch Alarms for SLO

1. Go to CloudWatch → Alarms → Create alarm**.**
2. Select ECS metric (CPUUtilization, MemoryUtilization, RunningTaskCount).
3. Set Threshold based on SLO:
4. Attach SNS topic to send notifications.

Alarms act as automatic monitoring of SLO violations.


Select metric
MemoryUtilization
1h 3h 12h 1d 3d 1w Custom
UTC timezone
Line
Add math
Add query
Browse (4)
Multi source query
Graphed metrics (1)
Options
Source
Search for any metric, dimension, resource id or account id
ClusterName 4/4
ServiceName
Metric name
Alarms
generative-ai-cluster
generative-ai-service-v2
MemoryUtilization
No alarms
generative-ai-cluster
generative-ai-service-v2
CPUUtilization
No alarms
generative-ai-cluster
generative-ai-service
MemoryUtilization
No alarms
Cancel
Select metric
CloudWatch
Metrics
Metrics
Info
MemoryUtilization
1h 3h 12h 1d 3d 1w Custom
UTC timezone
Actions
Investigate
Line
Browse
Multi source query
Graphed metrics (1)
Options
Source
Add math
Add query
Add dynamic label
Info
Average
5 minutes
Clear graph
Label
Details
Statistic
Period
Y axis
Actions
MemoryUtilization
Region: us-east-1 • ECS • MemoryUtilizati
Average
5 minu...
CloudWatch
Alarms
Create alarm
Step 1
Specify metric and conditions
Step 2
Configure actions
Step 3
Add alarm details
Step 4
Preview and create
Specify metric and conditions
Alarm recommendations
View details
Metric
Edit
Graph
This alarm will trigger when the blue line goes above the red line for 1 datapoints within 5 minutes.
No unit
1
0.5
Namespace
AWS/ECS
Metric name
MemoryUtilization
ServiceName
generative-ai-service-v2

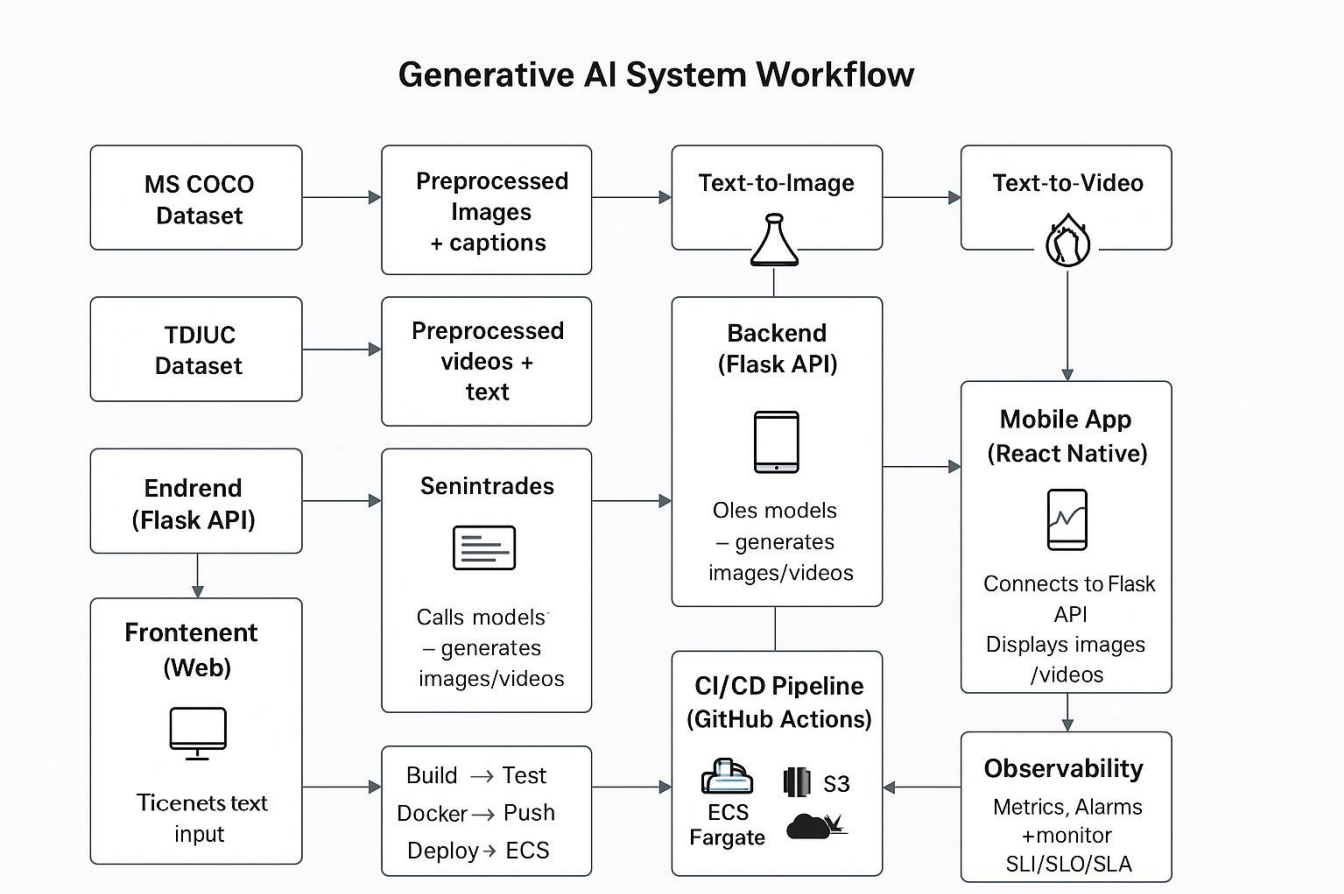
Define SLA (Service Level Agreement)

SLA = formal agreement with client / stakeholders. Examples:

SLA is documented and tracked using the CloudWatch alarms

Monitor and Report

- Regularly check CloudWatch → Alarms.
- View metrics graphs for SLI trends.
- Use alarm history to report SLO/SLA compliance**.**



END