1. Coding python code

```
from openai import OpenAI

client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
  api_key = "nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa"
)

completion = client.chat.completions.create(
  model="meta/llama-3.1-405b-instruct",
  messages=[{"role":"user","content":""}],
  temperature=0.2,
  top_p=0.7,
  max_tokens=1024,
  stream=True,

tools=[{"type":"function","function":{"name":"describe_harry_potter_character","descri
ption":"Returns information and images of Harry Potter
characters.","parameters":{"type":"object","properties":{"name":{"type":"string","enum
":["Harry James Potter","Hermione Jean Granger","Ron Weasley","Fred
Weasley","George Weasley","Bill Weasley","Percy Weasley","Charlie Weasley","Ginny
Weasley","Molly Weasley","Arthur Weasley","Neville Longbottom","Luna
Lovegood","Draco Malfoy","Albus Percival Wulfric Brian Dumbledore","Minerva
McGonagall","Remus Lupin","Rubeus Hagrid","Sirius Black","Severus Snape","Bellatrix
Lestrange","Lord Voldemort","Cedric Diggory","Nymphadora Tonks","James
Potter"],"description":"Name of the Harry Potter
character"}},"required":["name"]}}},{"type":"function","function":{"name":"name_a_col
or","description":"A tool that returns a bunch of color names for a given
color_hex.","parameters":{"type":"object","properties":{"color_hex":{"type":"string","de
scription":"A hexadecimal color value which must be represented as a
string."}},"required":["color_hex"]}}}],
  tool_choice="auto"
)

for chunk in completion:
```

```
  if not chunk.choices:
    continue
  if chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")

  if hasattr(chunk.choices[0].delta, 'tool_calls') and chunk.choices[0].delta.tool_calls:
    for tool_call in chunk.choices[0].delta.tool_calls:
      print(f"\nTool Call: {tool_call.function.name}")
      print(f"Arguments: {tool_call.function.arguments}")

node for coding
import OpenAI from 'openai';

const openai = new OpenAI({
  apiKey: 'nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa',
  baseURL: 'https://integrate.api.nvidia.com/v1',
})
async function main() {
  const completionParams = {
    model: "meta/llama-3.1-405b-instruct",
    messages: [{"role":"user","content":""}],
    temperature: 0.2,
    top_p: 0.7,
    max_tokens: 1024,
    stream: true
  };


  completionParams.tools =
[{"type":"function","function":{"name":"describe_harry_potter_character","description"
:"Returns information and images of Harry Potter
characters.","parameters":{"type":"object","properties":{"name":{"type":"string","enum
":["Harry James Potter","Hermione Jean Granger","Ron Weasley","Fred
Weasley","George Weasley","Bill Weasley","Percy Weasley","Charlie Weasley","Ginny
Weasley","Molly Weasley","Arthur Weasley","Neville Longbottom","Luna
Lovegood","Draco Malfoy","Albus Percival Wulfric Brian Dumbledore","Minerva
McGonagall","Remus Lupin","Rubeus Hagrid","Sirius Black","Severus Snape","Bellatrix
```

```
Lestrange","Lord Voldemort","Cedric Diggory","Nymphadora Tonks","James
Potter"],"description":"Name of the Harry Potter
character"}},"required":["name"]}}},{"type":"function","function":{"name":"name_a_col
or","description":"A tool that returns a bunch of color names for a given
color_hex.","parameters":{"type":"object","properties":{"color_hex":{"type":"string","de
scription":"A hexadecimal color value which must be represented as a
string."}},"required":["color_hex"]}}}];
  completionParams.tool_choice = "auto";


  const completion = await openai.chat.completions.create(completionParams);


  for await (const chunk of completion) {
    process.stdout.write(chunk.choices[0]?.delta?.content || '');

    if (chunk.choices[0]?.delta?.tool_calls) {
      chunk.choices[0].delta.tool_calls.forEach(toolCall => {
        console.log(`\nTool Call: ${toolCall.function.name}`);
        console.log(`Arguments: ${toolCall.function.arguments}`);
      });
    }

  }

}

main();
kimik2 python

import requests, base64

invoke_url = "https://integrate.api.nvidia.com/v1/chat/completions"
stream = True


headers = {
```

```
  "Authorization": "Bearer nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa",
  "Accept": "text/event-stream" if stream else "application/json"
}

payload = {
  "model": "moonshotai/kimi-k2.5",
  "messages": [{"role":"user","content":""}],
  "max_tokens": 16384,
  "temperature": 1.00,
  "top_p": 1.00,
  "stream": stream,
  "chat_template_kwargs": {"thinking":True},
}

payload["tools"] =
[{"type":"function","function":{"name":"describe_harry_potter_character","description"
:"Returns information and images of Harry Potter
characters.","parameters":{"type":"object","properties":{"name":{"type":"string","enum
":["Harry James Potter","Hermione Jean Granger","Ron Weasley","Fred
Weasley","George Weasley","Bill Weasley","Percy Weasley","Charlie Weasley","Ginny
Weasley","Molly Weasley","Arthur Weasley","Neville Longbottom","Luna
Lovegood","Draco Malfoy","Albus Percival Wulfric Brian Dumbledore","Minerva
McGonagall","Remus Lupin","Rubeus Hagrid","Sirius Black","Severus Snape","Bellatrix
Lestrange","Lord Voldemort","Cedric Diggory","Nymphadora Tonks","James
Potter"],"description":"Name of the Harry Potter
character"}},"required":["name"]}}},{"type":"function","function":{"name":"name_a_col
or","description":"A tool that returns a bunch of color names for a given
color_hex.","parameters":{"type":"object","properties":{"color_hex":{"type":"string","de
scription":"A hexadecimal color value which must be represented as a
string."}},"required":["color_hex"]}}}]


payload["tool_choice"] = "auto"


response = requests.post(invoke_url, headers=headers, json=payload)
```

```
if stream:
   for line in response.iter_lines():
      if line:
         print(line.decode("utf-8"))
else:
   print(response.json())
```

DEEPSEEK R1 PYTHON CODE

```
from openai import OpenAI

client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
  api_key = "nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa"
)

completion = client.chat.completions.create(
  model="deepseek-ai/deepseek-v3.2",
  messages=[{"role":"user","content":""}],
  temperature=1,
  top_p=0.95,
  max_tokens=16384,
  extra_body={"chat_template_kwargs": {"thinking":True}},
  stream=True,

tools=[{"type":"function","function":{"name":"describe_harry_potter_character","descri
ption":"Returns information and images of Harry Potter
characters.","parameters":{"type":"object","properties":{"name":{"type":"string","enum
":["Harry James Potter","Hermione Jean Granger","Ron Weasley","Fred
Weasley","George Weasley","Bill Weasley","Percy Weasley","Charlie Weasley","Ginny
Weasley","Molly Weasley","Arthur Weasley","Neville Longbottom","Luna
Lovegood","Draco Malfoy","Albus Percival Wulfric Brian Dumbledore","Minerva
McGonagall","Remus Lupin","Rubeus Hagrid","Sirius Black","Severus Snape","Bellatrix
Lestrange","Lord Voldemort","Cedric Diggory","Nymphadora Tonks","James
Potter"],"description":"Name of the Harry Potter
character"}},"required":["name"]}}},{"type":"function","function":{"name":"name_a_col
or","description":"A tool that returns a bunch of color names for a given
```

```
color_hex.","parameters":{"type":"object","properties":{"color_hex":{"type":"string","de
scription":"A hexadecimal color value which must be represented as a
string."}},"required":["color_hex"]}}}],
  tool_choice="auto"
)

for chunk in completion:
  if not getattr(chunk, "choices", None):
    continue
  reasoning = getattr(chunk.choices[0].delta, "reasoning_content", None)
  if reasoning:
    print(reasoning, end="")
  if chunk.choices and chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
  if chunk.choices[0].delta.tool_calls:
    print(chunk.choices[0].delta.tool_calls)
```

NODE DEEPSEEK R1

```
import OpenAI from 'openai';

const openai = new OpenAI({
  apiKey: 'nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa',
  baseURL: 'https://integrate.api.nvidia.com/v1',
})

async function main() {
  const completion = await openai.chat.completions.create({
    model: "deepseek-ai/deepseek-v3.2",
    messages: [{"role":"user","content":""}],
    temperature: 1,
    top_p: 0.95,
    max_tokens: 16384,
    chat_template_kwargs: {"thinking":true},
    stream: true,
```

```
    tools:
[{"type":"function","function":{"name":"describe_harry_potter_character","description"
:"Returns information and images of Harry Potter
characters.","parameters":{"type":"object","properties":{"name":{"type":"string","enum
":["Harry James Potter","Hermione Jean Granger","Ron Weasley","Fred
Weasley","George Weasley","Bill Weasley","Percy Weasley","Charlie Weasley","Ginny
Weasley","Molly Weasley","Arthur Weasley","Neville Longbottom","Luna
Lovegood","Draco Malfoy","Albus Percival Wulfric Brian Dumbledore","Minerva
McGonagall","Remus Lupin","Rubeus Hagrid","Sirius Black","Severus Snape","Bellatrix
Lestrange","Lord Voldemort","Cedric Diggory","Nymphadora Tonks","James
Potter"],"description":"Name of the Harry Potter
character"}},"required":["name"]}}},{"type":"function","function":{"name":"name_a_col
or","description":"A tool that returns a bunch of color names for a given
color_hex.","parameters":{"type":"object","properties":{"color_hex":{"type":"string","de
scription":"A hexadecimal color value which must be represented as a
string."}},"required":["color_hex"]}}}],
    tool_choice: "auto",
  })

  for await (const chunk of completion) {
        const reasoning = chunk.choices[0]?.delta?.reasoning_content;
    if (reasoning) process.stdout.write(reasoning);
        process.stdout.write(chunk.choices[0]?.delta?.content || '')
    if (chunk.choices[0]?.delta?.tool_calls)
console.log(chunk.choices[0]?.delta?.tool_calls)
  }

}

main();
```

DEEPSEEK R1 WITHOUT EXAMPLE from openai import OpenAI

```
client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
```

```
  api_key = "nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa"
)

completion = client.chat.completions.create(
  model="deepseek-ai/deepseek-v3.2",
  messages=[{"role":"user","content":""}],
  temperature=1,
  top_p=0.95,
  max_tokens=16384,
  extra_body={"chat_template_kwargs": {"thinking":True}},
  stream=True
)

for chunk in completion:
  if not getattr(chunk, "choices", None):
    continue
  reasoning = getattr(chunk.choices[0].delta, "reasoning_content", None)
  if reasoning:
    print(reasoning, end="")
  if chunk.choices and chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

QWEN 3 CODER

```
from openai import OpenAI

client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
  api_key = "nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa"
)

completion = client.chat.completions.create(
  model="qwen/qwen3-coder-480b-a35b-instruct",
  messages=[{"role":"user","content":""}],
  temperature=0.7,
  top_p=0.8,
  max_tokens=4096,
```

```
  stream=True
)

for chunk in completion:
  if chunk.choices and chunk.choices[0].delta.content is not None:
    print(chunk.choices[0].delta.content, end="")
```

NODE

```
import OpenAI from 'openai';

const openai = new OpenAI({
  apiKey: 'nvapi-kJ0wf7syd-5VhM-
yieG7AwFgd6xP2lLBC90oBRQiqFwErk6ZIFxpQr2FhqOq9zSa',
  baseURL: 'https://integrate.api.nvidia.com/v1',
})

async function main() {
  const completion = await openai.chat.completions.create({
    model: "qwen/qwen3-coder-480b-a35b-instruct",
    messages: [{"role":"user","content":""}],
    temperature: 0.7,
    top_p: 0.8,
    max_tokens: 15713,
    stream: true
  })

  for await (const chunk of completion) {
    process.stdout.write(chunk.choices[0]?.delta?.content || '')
  }

}

main();
```

IMAGE GENERATION

API KEY

```
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
```

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
Follow the steps below to download and run the NVIDIA NIM inference microservice for this model on your infrastructure of choice.
Step 1Get Credentials
Get API Key
To access Stable Diffusion 3.5 Large model read and accept Stable Diffusion 3.5 Large, Stable Diffusion 3.5 Large TensorRT and Stable Diffusion 3.5 Large ControlNet TensorRT License Agreements and Acceptable Use Policy.
Create a new Hugging Face token with *Read access to contents of all public gated repos you can access* permission.
Export your personal credentials as environment variables:
Bash
Copy

```
export NGC_API_KEY=<PASTE_API_KEY_HERE>
export HF_TOKEN=<PASTE_HUGGING_FACE_TOKEN_HERE>
```

Step 2Pull and Run the NIM
Login to NVIDIA NGC so that you can pull the NIM container:
Bash
Copy

```
echo "$NGC_API_KEY" | docker login nvcr.io --username '$oauthtoken' --password-stdin
```

Pull and run the NIM with the command below.
Bash
Copy

```
# Create the cache directory on the host machine.
export LOCAL_NIM_CACHE=~/.cache/nim
mkdir -p "$LOCAL_NIM_CACHE"
chmod 777 $LOCAL_NIM_CACHE

docker run -it --rm --name=nim-server \
  --runtime=nvidia --gpus='"device=0"' \
  -e NGC_API_KEY=$NGC_API_KEY \
  -e HF_TOKEN=$HF_TOKEN \
  -p 8000:8000 \
  -v "$LOCAL_NIM_CACHE:/opt/nim/.cache/" \
  nvcr.io/nim/stabilityai/stable-diffusion-3.5-large:latest
```

You can specify the desired variant of Stable Diffusion 3.5 Large by adding **-e NIM_MODEL_VARIANT=<your variant>**. Available variants are **base**, **base+canny**, **base+depth** and **base+canny+depth**.

When you run the preceding command, the container downloads the model, initializes a NIM inference pipeline, and performs a pipeline warm up. A pipeline warm up typically requires up to three minutes. The warm up is complete when the container logs show **Pipeline warmup: start/done**.

Step 3Test the NIM

Bash

Copy

```
invoke_url="http://localhost:8000/v1/infer"

output_image_path="result.jpg"

response=$(curl -X POST $invoke_url \
   -H "Accept: application/json" \
   -H "Content-Type: application/json" \
   -d '{
        "prompt": "A simple coffee shop interior",
        "mode": "base",
        "seed": 0,
        "steps": 30
      }')
response_body=$(echo "$response" | awk '/{/,EOF-1')
echo $response_body | jq .artifacts[0].base64 | tr -d '"' | base64 --decode > $output_image_path
```

For more details on getting started with this NIM including configuring using parameters, visit the Visual GenAI NIM docs.

GLM 5

Follow the steps below to download and run the NVIDIA NIM inference microservice for this model on your infrastructure of choice.

Step 1Get Credentials

Get API Key

To access Stable Diffusion 3.5 Large model read and accept Stable Diffusion 3.5 Large, Stable Diffusion 3.5 Large TensorRT and Stable Diffusion 3.5 Large ControlNet TensorRT License Agreements and Acceptable Use Policy.

Create a new Hugging Face token with *Read access to contents of all public gated repos you can access* permission.

Export your personal credentials as environment variables:
Bash
Copy
export NGC_API_KEY=<PASTE_API_KEY_HERE>
export HF_TOKEN=<PASTE_HUGGING_FACE_TOKEN_HERE>
Step 2Pull and Run the NIM
Login to NVIDIA NGC so that you can pull the NIM container:
Bash
Copy
echo "$NGC_API_KEY" | docker login nvcr.io --username '$oauthtoken' --password-stdin
Pull and run the NIM with the command below.
Bash
Copy
Create the cache directory on the host machine.
export LOCAL_NIM_CACHE=~/.cache/nim
mkdir -p "$LOCAL_NIM_CACHE"
chmod 777 $LOCAL_NIM_CACHE

docker run -it --rm --name=nim-server \
  --runtime=nvidia --gpus='"device=0"' \
  -e NGC_API_KEY=$NGC_API_KEY \
  -e HF_TOKEN=$HF_TOKEN \
  -p 8000:8000 \
  -v "$LOCAL_NIM_CACHE:/opt/nim/.cache/" \
  nvcr.io/nim/stabilityai/stable-diffusion-3.5-large:latest
You can specify the desired variant of Stable Diffusion 3.5 Large by adding **-e NIM_MODEL_VARIANT=<your variant>**. Available variants are **base**, **base+canny**, **base+depth** and **base+canny+depth**.
When you run the preceding command, the container downloads the model, initializes a NIM inference pipeline, and performs a pipeline warm up. A pipeline warm up typically requires up to three minutes. The warm up is complete when the container logs show **Pipeline warmup: start/done**.
Step 3Test the NIM
Bash
Copy
invoke_url="http://localhost:8000/v1/infer"

output_image_path="result.jpg"

```
response=$(curl -X POST $invoke_url \
   -H "Accept: application/json" \
   -H "Content-Type: application/json" \
   -d '{
        "prompt": "A simple coffee shop interior",
        "mode": "base",
        "seed": 0,
        "steps": 30
      }')
response_body=$(echo "$response" | awk '/{/,EOF-1')
echo $response_body | jq .artifacts[0].base64 | tr -d '"' | base64 --decode >
$output_image_path
```

For more details on getting started with this NIM including configuring using parameters, visit the Visual GenAI NIM docs.

```
from openai import OpenAI
import os
import sys

_USE_COLOR = sys.stdout.isatty() and os.getenv("NO_COLOR") is None
_REASONING_COLOR = "\033[90m" if _USE_COLOR else ""
_RESET_COLOR = "\033[0m" if _USE_COLOR else ""

client = OpenAI(
  base_url = "https://integrate.api.nvidia.com/v1",
  api_key = "nvapi-Zkilan61gTArhb0MzK-
ul89uKNJOmr6y0BXKJcdzc6oXHmy0p2RJ85hjaJYOqqBO"
)


completion = client.chat.completions.create(
  model="z-ai/glm5",
  messages=[{"role":"user","content":""}],
  temperature=1,
  top_p=1,
  max_tokens=16384,
  extra_body={"chat_template_kwargs":{"enable_thinking":True,"clear_thinking":False}},
```

```
  stream=True
)

for chunk in completion:
  if not getattr(chunk, "choices", None):
    continue
  if len(chunk.choices) == 0 or getattr(chunk.choices[0], "delta", None) is None:
    continue
  delta = chunk.choices[0].delta
  reasoning = getattr(delta, "reasoning_content", None)
  if reasoning:
    print(f"{_REASONING_COLOR}{reasoning}{_RESET_COLOR}", end="")
  if getattr(delta, "content", None) is not None:
    print(delta.content, end="")
```