

How to Use Meta Llama 3 Large Language Model on Vultr Cloud GPU

Learn how to deploy and use Meta Llama 3 Large Language Model on Vultr Cloud GPU with step-by-step instructions for setup, configuration, and optimization.

Contents

01	Introduction	3
02	Prerequisites	3
03	Infer Meta Llama 3 8B	3
04	Infer Meta Llama 3 8B Instruct	5
05	Conclusion	7

Introduction

[Llama 3](#) is a family of Large Language Models released by Meta, It is a collection of pre-trained and instruction-tuned generative text models. Llama 3 is pre-trained over 15T tokens collected from publically available sources, it is an auto-regressive family of models that uses an optimized transformer architecture. The instruction-tuned models use supervised fine-tuning (SFT) and reinforcement learning with human feedback (RLHF). The Llama 3 models have a custom limited commercial license which means models can used for commercial and research purposes.

In this article, you are to run inference workloads on the Llama3 8B and Llama 3 8B Instruct models. You also going to be exploring the model benchmarks to evaluate the model performance.

Prerequisites

- Deploy a [Vultr Cloud GPU instance](#).
- Access the Jupyter Lab environment from the Instance Page.
- Have access to [Meta Llama 3](#) model weights on HuggingFace

Infer Meta Llama 3 8B

In this section, you are to run inference workloads on the Llama 3 8B base model with 8 billion parameters.

1. Install the required dependencies.

```
PYTHON
```

```
!pip install pytorch transformer huggingface-hub
```

2. Log in to the HuggingFace Hub.

```
PYTHON
```

```
huggingfacehub-cli --login
```

On the execution of this command, you will be prompted for an access token. Where you have to provide the token to access the Llama 3 Models.

3. Import necessary modules.

```
PYTHON
```

```
import transformers  
import torch
```

4. Declare the Model.

```
PYTHON
```

```
model_id = "meta-llama/Meta-Llama-3-8B"
```

5. Declare the model pipeline.

```
PYTHON
```

```
pipeline = transformers.pipeline(  
    "text-generation", model=model_id,  
    model_kwargs={"torch_dtype": torch.bfloat16},  
    device_map="auto"
```

In the above code, you declared the model pipeline with the `task-generation` task. By default, Llama 3 8B is set to 16-bit precision.

6. Provide a prompt.

```
PYTHON
```

```
pipeline("Hey how are you doing today?")
```

Infer Meta Llama 3 8B Instruct

In this section, you are to run inference workloads on the Llama 3 8B Instruct model with 8 billion parameters. This model is fine-tuned for dialogue-based use cases.

1. Import necessary modules.

PYTHON

```
import transformers
import torch
```

2. Declare the model.

PYTHON

```
model_id = "meta-llama/Meta-Llama-3-8B-Instruct"
```

3. Declare the model pipeline.

PYTHON

```
pipeline = transformers.pipeline(
    "text-generation",
    model=model_id,
    model_kwargs={"torch_dtype": torch.bfloat16},
    device="auto",
)
```

4. Set roles for the model.

PYTHON

```
messages = [
    {"role": "system", "content": "You are a pirate chatbot who
    always responds in pirate speak!"},
```

```
{ "role": "user", "content": "Who are you?" },  
]
```

In the above code, you are setting the roles informing the chatbot that it is a pirate chatbot.

5. Make a prompt template.

PYTHON

```
prompt = pipeline.tokenizer.apply_chat_template(  
messages,  
tokenize=False,  
add_generation_prompt=True  
)
```

6. Set the terminators for text generation.

PYTHON

```
terminators = [  
pipeline.tokenizer.eos_token_id,  
pipeline.tokenizer.convert_tokens_to_ids("<|eot_id|>")  
]
```

7. Set the model parameters.

PYTHON

```
outputs = pipeline(  
prompt,  
max_new_tokens=256,  
eos_token_id=terminators,  
do_sample=True,  
temperature=0.6,  
top_p=0.9,  
)
```

In the above code, you are setting up the model parameters that can be adjusted to improve the response generated by the model.

8. Print the generated output.

```
PYTHON
```

```
print(outputs[0]["generated_text"][len(prompt):])
```

Conclusion

In this article, you used Meta Llama 3 models on a Vultr Cloud GPU Server. And ran inference workloads with the latest Llama 3 80B in 16-bit mode with its fine-tuned Instruct version also in the 16-bit mode.



VULTR

