Building with Foundation Models on Amazon SageMaker Studio

Building with Foundation Models on Amazon SageMaker Studio
Introduction to Workshop Studio Setup
SageMaker Spaces: JupyterLab and Code Editor
Lab 0 - Deploy Llama2 and Embedding Models
Lab 1 - Setup an LLM Playground on Studio
Lab 2 - Prompt Engineering with LLMs
Lab 3 - Retrieval Augmented Generation (RAG) using PySpark on EMR
Lab 4 - Fine-Tune Gen AI Models on Studio
Studio Notebook HuggingFace Fine-Tuning
Studio Notebook HuggingFace Fine-Tuning
Trainium JumpStart Fine-Tuning
Lab 5 - Foundation Model Evaluation

AWS account access
Open AWS console (us-east-1)
Get AWS CLI credentials
Get EC2 SSH key

Exit event

Studio Notebook HuggingFace Fine-Tuning

> **Important**
> Run with **Code Editor**: We're going to use scripts inside `lab-04-finetuning/local-finetune/*.py` notebook for this section

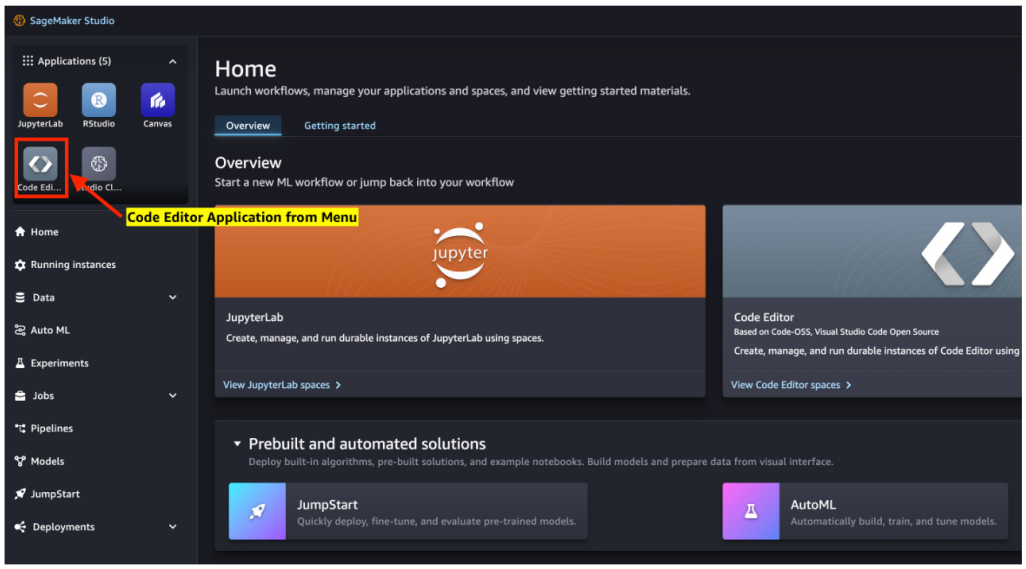
Contents

- Contents
- Code Editor for Fine-Tuning
  - Code Editor Syntax
- Dataset
- Fine-Tuning on ML Instances with GPU, Deploy Model on GPU Instances and Validate Model Inference
  - Fine-Tuning on ML Instances with GPU using Code Editor Terminal
  - Deploy Model on GPU Instances
  - Validate Model Inference

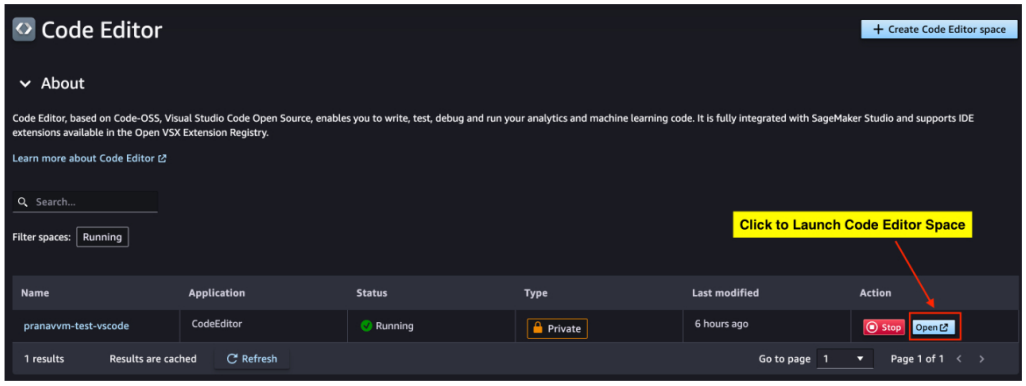
Code Editor for Fine-Tuning

For this lab, we're going to leverage SageMaker's Code Editor Open Source VSCode to fine tune a Llama 2 Model. To get to SageMaker's Code Editor,

1. Let's navigate back to Studio home



2. You should already have a Code Editor Space provisioned and running.



> If you don't see a Code Editor Space, please navigate to the earlier "SageMaker Spaces: JupyterLab and Code Editor" Section of this Workshop and follow the instruction provided under `Create a VSCode Space`

Code Editor Syntax

Unlike JupyterLab, which is very heavily notebook driven and prepared for a visual workflow, Code Editor is meant for a more 'developer' styled workflow. In that spirit, we're going to be running our Fine-Tuning task in a script mode.

> **Ensure Repo is cloned**
> Please ensure you have cloned the repository into your Code Editor space, as per instructions provided under SageMaker Spaces: JupyterLab and Code Editor section

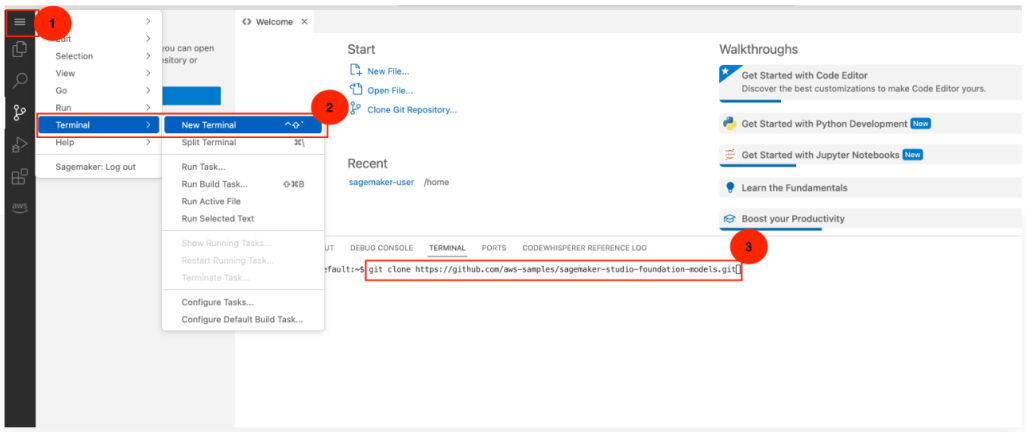
If you haven't cloned the repository into Code Editor Space, the follow the instructions below to do so,

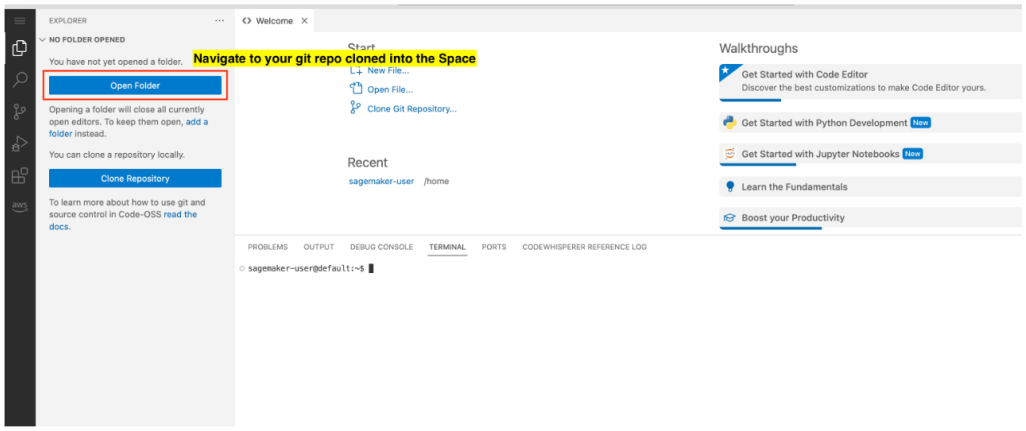
1. Launch a new terminal
2. Paste the git repo into the terminal to clone the repository into your Space

```
git clone https://github.com/aws-samples/sagemaker-studio-foundation-models.git
```

3. Navigate to `lab-04-finetuning/local-finetune/` like below from your File Explorer or from your terminal,

```
cd ~/sagemaker-studio-foundation-models/lab-04-finetuning/local-finetune/
```





Dataset

In this demo, we're going to use Dolly dataset in an instruction tuning format. Dolly dataset contains roughly 15,000 instruction following records for various categories such as question answering, summarization, information extraction etc. It is available under Apache 2.0 license. We will select the summarization examples for fine-tuning. You can extend this lab fine-tuning example to any dataset you prefer.

Dataset is loaded into memory prior to fine-tuning an LLM.

```
# train slice
train_dataset = load_dataset(dataset_name, split="train[:75%]")
# validation slice
validation_dataset = load_dataset(dataset_name, split="train[75%:95%]")
# test slice
test_dataset = load_dataset(dataset_name, split="train[95%:]")
```

Sample dataset is downloaded in a `json` format,

```
{'instruction': 'Who was boxer John Baldwin', 'context': 'Known as "The Mad" Baldwin, he turned pro in 1970 and lost a decision to Ranvi
```

Then, the `json` is structured using dolly format using `### Instruction`, `### Context` and `### Answer` keys,

```
### Instruction
Who was boxer John Baldwin
```

```
### Context
Known as "The Mad" Baldwin, he turned pro in 1970 and lost a decision to Marvin Hagler in 1975. In 1977 he took on Rocky Mosley Jr in th

### Answer
Born in Detroit, John Baldwin was a professional US boxer.  He fought in the Light-Middleweight category and won a bronze medal in the 1
```

The prompt aligned text blob is tokenized and chunked to ready the dataset for model fine-tuning.

Fine-Tuning on ML Instances with GPU, Deploy Model on GPU Instances and Validate Model Inference

We're going to use a Llama 2 variant called `Mikael110/llama-2-7b-guanaco-fp16` for this lab. This model can be found on HuggingFace hub here.

> **Important**
> This model is Llama2 variant and adheres to meta's Llama 2 licensing policy that can be found here : Llama2 License

We are going to be using NVIDIA `Tesla T4` 16 GB GPU for fine-tuning Llama 2 7B. Unfortunately, `Llama2-7B` fp32 model does not fit inside a 16GB GPU, we're going to use bitsandbytes to quantize our model into a 4bit model before fine-tuning on a T4 GPU.

```python
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_use_double_quant=True,
    bnb_4bit_quant_type="nf4",
    bnb_4bit_compute_dtype=torch.bfloat16
)

model = AutoModelForCausalLM.from_pretrained(
    model_id,
    quantization_config=bnb_config,
    device_map="auto"
)
```

This is going to download model weights from HuggingFace hub to your local machine, loads the model and its weights into memory and pins it the GPU. Tracking your GPU usage should show usage as below,

```
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
Wed Jan  3 19:46:14 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 470.57.02    CUDA Version: 11.8     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:1E.0 Off |                    0 |
| N/A   28C    P0    24W /  70W |   4509MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
+-----------------------------------------------------------------------------+
```

Fine-Tuning on ML Instances with GPU using Code Editor Terminal

Navigate to `lab-04-finetuning/local-finetune/` like below from your File Explorer or from your terminal,

```
cd ~/sagemaker-studio-foundation-models/lab-04-finetuning/local-finetune/
```

We are going to be running `local_fine_tune_llama2.py` script from our terminal but before that, we have a setup process to install some pre-requisite packages. Please run the following commands in sequence,

1. Setup

```
sh setup.sh
```

2. Kick off fine tuning job

```
python3 local_fine_tune_llama2.py \
    --dataset_name=databricks/databricks-dolly-15k \
    --hf_model_id=Mikael110/llama-2-7b-guanaco-fp16 \
    --epochs=1 \
    --per_device_train_batch_size=1 \
```

```
    --per_device_valid_batch_size=1 \
    --logging_steps=1
```

3. Wait for training task to complete,

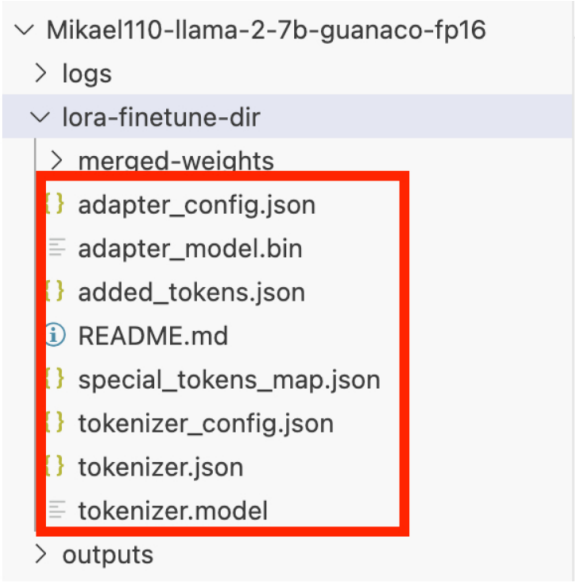
```
Accept Llama 2 EULA? [y/n]: y
Training size: 100 | Validation size: 25

Training sample:

{   'category': 'open_qa',
    'context': '',
    'instruction': 'Who became king of Holland in 1806?',
    'response': 'William I of the Netherlands became king of Holland in 1806.'}

Validation sample:

{   'category': 'open_qa',
    'context': '',
    'instruction': 'Who won the World Chess Championship in 2021?',
    'response': 'Magnus Carlsen defeated Ian Nepomniachtchi 7.5 - 3.5 to '
                'become the 2021 World Chess Champion.'}
### Instruction
Why is it a good idea to walk every day?

### Answer
Walking is a good exercise for burning calories without causing any damage to joints. People of any age can do it and a few miles a day can result in positive effects on cardiovascular health
 and general fitness.</s>
Total number of samples ---> Train: 23, Validation: 5
Loading checkpoint shards:  50%|                                                                                                  | 1/2 [01:18<01:18, 78.08s/it]
```

```
    'response': 'William I of the Netherlands became king of Holland in 1806.'}

Validation sample:

{   'category': 'open_qa',
    'context': '',
    'instruction': 'Who won the World Chess Championship in 2021?',
    'response': 'Magnus Carlsen defeated Ian Nepomniachtchi 7.5 - 3.5 to '
                'become the 2021 World Chess Champion.'}
### Instruction
Why is it a good idea to walk every day?

### Answer
Walking is a good exercise for burning calories without causing any damage to joints. People of any age can do it and a few miles a day can result in positive effects on cardiovascular health
 and general fitness.</s>
Total number of samples ---> Train: 23, Validation: 5
Loading checkpoint shards: 100%|                                                                                                  | 2/2 [01:45<00:00, 52.51s/it]
Found 7 modules to quantize: ['v_proj', 'o_proj', 'up_proj', 'gate_proj', 'k_proj', 'down_proj', 'q_proj']
Beginning to train model for Epochs:1, Train/Val Batch sizes:1/1, Learning Rate:1e-05, Logging Step: 1
/opt/conda/lib/python3.10/site-packages/transformers/optimization.py:411: FutureWarning: This implementation of AdamW is deprecated and will be removed in a future version. Use the PyTorch im
plementation torch.optim.AdamW instead, or set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
{'loss': 1.8143, 'learning_rate': 9.565217391304349e-06, 'epoch': 0.04}
{'loss': 1.1694, 'learning_rate': 9.130434782608697e-06, 'epoch': 0.09}
  9%|                                                                                                                          | 2/23 [00:36<06:21, 18.18s/it]
```

Ongoing Model Tuning

Deploy Model on GPU Instances

Once the local training task is complete, you should find PEFT LoRA adapter weights in your working dir (ex: `Mikael110/llama-2-7b-guanaco-fp16 saved to Mikael110-llama-2-7b-guanaco-fp16/lora-finetune-dir/`).



To deploy your fine-tuned adapter weights as a SageMaker endpoint, you can run,

```
python3 deploy_llama2.py \
    --sm_endpoint_name=ft-meta-llama2-7b-chat-tg-ep \
    --sm_model_name=ft-meta-llama2-7b-chat-tg-model \
    --instance_type=ml.g5.2xlarge \
    --hf_model_id=Mikael110/llama-2-7b-guanaco-fp16
```

Deploy llama2 script does the following,