Table of Contents

Author	Date	Version	Comments
Venks Mantha	2023/12/27	1.0	Initial version
Venks Mantha	2024/01/11	1.1	Updated to add references and some cleanup
Venks Mantha	2024/01/19	1.2	Added Appendix
Venks Mantha	2024/01/21	1.3	Corrections based on further testing/review

Jan 2024

Overview

Many of our customers would like to run LLMs in Snowflake to use them in conjunction with their Snowflake data, without the need to call external APIs. In addition to simply consuming LLMs, our customers are also interested in fine-tuning pretrained LLMs, including models available with the NVIDIA NeMo framework and Meta's Llama models, with their own corporate data.

Our collaboration with NVDIA makes this easy to run NVIDIA accelerated computing workloads directly within Snowflake using Snowpark Container Services!

Note:

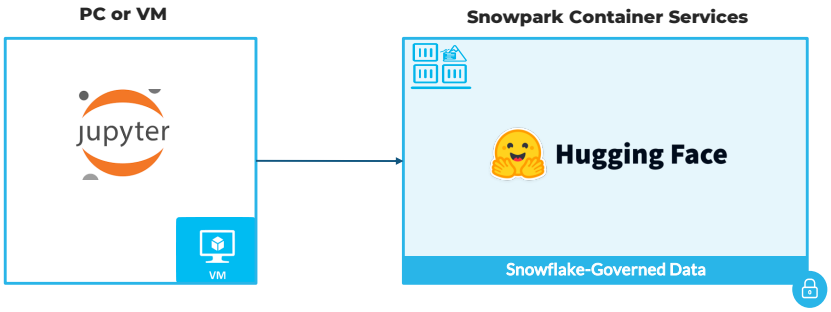
The content contained in this document is informational only and must be validated to ensure it would meet specific implementation needs including the current and feature functionality. The links provided in the document may change or be updated in the future as both Snowflake documentation and the blogs, articles included may be changed or updated by the authors.
This is NOT an official Snowflake document but prepared specifically for the snowflake master class to demonstrate Snowpark Container Services business value.

Objectives

Primary objective is to allow customers to "securely" run a Llama 2 model within Snowflake environment. Further, we finetune and save the finetuned model in Snowpark container Services for inferencing. We achieve this in two steps.

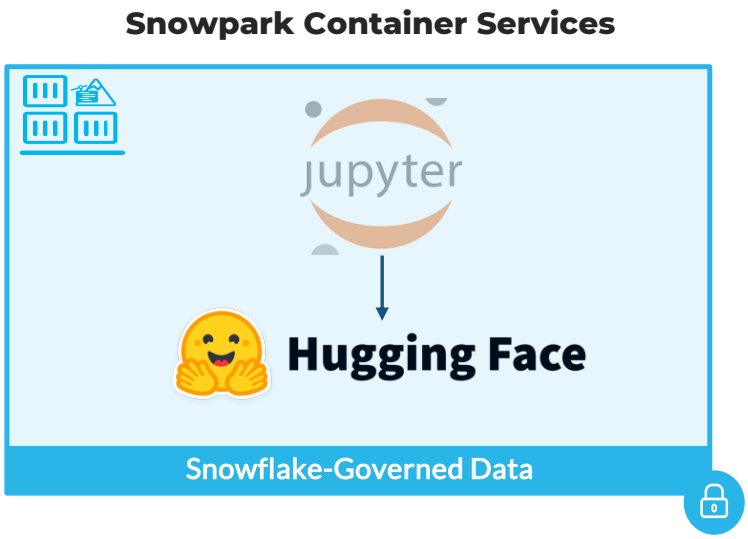
Lab 1:

Deploy a Llama 2 model from Hugging Face in Snowpark Container Services – no finetuning.



Lab 2:

Finetune a Llama 2 model using custom data with Snowpark Container Services and deploy the finetuned model back to Snowpark Container Services.



Both labs are mutually exclusive and so you can try and test each of them independently. To save resources, we will remove the SPCS service from Lab1 before running Lab2. All of it is detailed below.

High Level Steps

1. Pre-requisites
2. Environment setup

Lab1:

3. Deploy Llama 2 base model from Hugging face to Snowpark Container Services.

Lab2:

4. Build and push (store) the image to Snowflake stage.
5. Finetune Llama 2 model using custom data from the image.
6. Deploy the finetuned model to Snowpark Container Services.

Step 1: Pre-requisites.

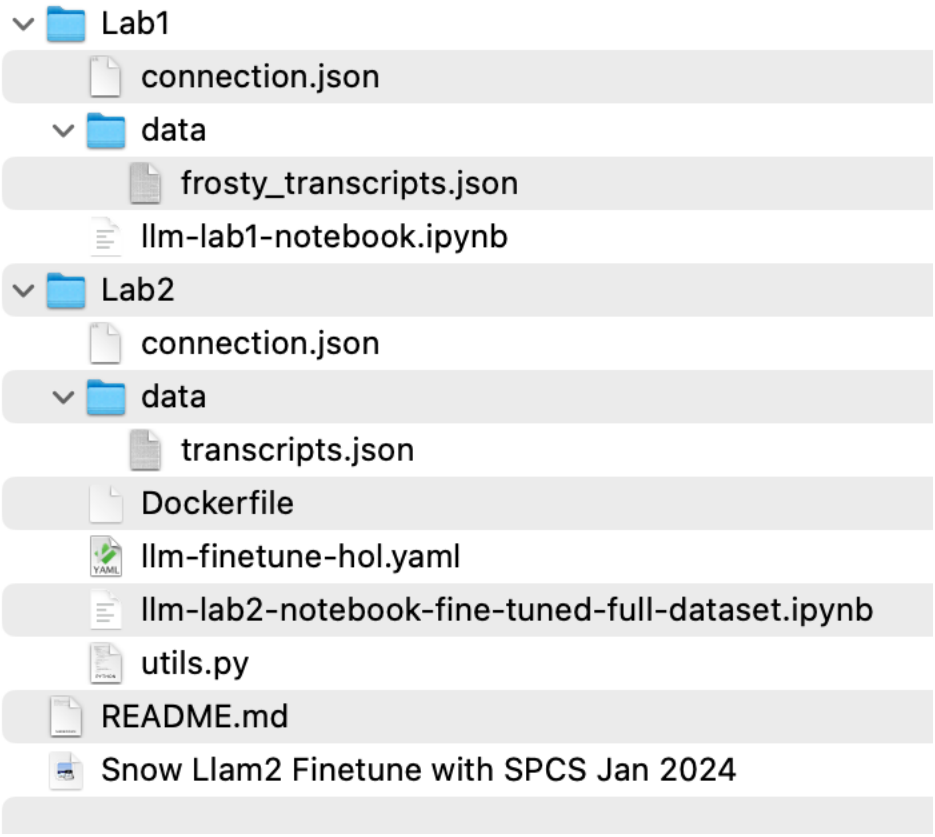
- A PC or a VM where you can download libraries and Python packages from the web.
  - **Note**: This PC or VM would also then push local docker images to Snowflake.
  - You need a good network connection to push the image to Snowflake (upload bandwidth)
- Docker Desktop installed.
  - https://docs.docker.com/desktop/
- Any IDE to run notebooks (VS Code, Jupyter etc.)

- A Hugging Face account.
  - https://huggingface.co/
- Completed Llama 2 request form.
  - https://ai.meta.com/resources/models-and-libraries/llama-downloads/
  - **Note**: Your Hugging Face account email address MUST match the email you provide on the Meta website, or your request will not be approved
- After approval, submit the form to access Llama 2 on Hugging Face to unlock access to the model.
  - https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
  - **Note**: This could take several hours.
- Snowflake account with Snowpark Container Services enabled.
  - As of this masterclass, Snowpark Container Services (aka SPCS) is available in public preview in certain AWS regions.
  - https://docs.snowflake.com/en/developer-guide/snowpark-container-services/overview
- Hugging face token.
  - Login into your account and access your Hugging Face token by browsing to Settings -> Access Tokens -> New token.
  - https://huggingface.co/settings/tokens
- Optional:
  - If you want to know what privileges are required to set up and run images in Snowpark container services, please check this quick start:
  - https://quickstarts.snowflake.com/guide/intro_to_snowpark_container_services/index.html#1

Step 2: Environment Setup.

1. Clone or download the repository to your local PC or VM.
   https://github.com/VenksMantha/spcs_llm_finetune

2. Unzip the contents of the repository if did you a download.
3. Confirm you have two folders as follows:



4. Snowflake Set up:

    a. Use a role that has enough privileges. For e.g. role VENKSADMIN is a custom role I created that has all the necessary privileges. You replace VENKSADMIN with you own custom role. Please note that you MUST create a custom role and please DO NOT use ACCOUNT ADMIN role.

    Please see Appendix B on steps I used to create VENKSADMIN role which has elevated privileges.

    ```
    USE ROLE venksadmin ;
    ```

    b. Create a Snowflake virtual warehouse (or you can use an existing one)

    ```
    CREATE WAREHOUSE llm_finetune_wh
    WAREHOUSE_SIZE = 'xsmall'
    AUTO_SUSPEND = 180
    INITIALLY_SUSPENDED = true ;

    USE WAREHOUSE llm_finetune_wh ;
    ```

    c. Create a database and schema to hold SPCS specific objects.

```
CREATE DATABASE llm_finetune_db ;
CREATE SCHEMA hol ;
USE SCHEMA llm_finetune_db.hol ;
```

d. Create a Security Integration (OAUTH based) which is mandatory and is required to connect SPCS as an end point.

```
CREATE SECURITY INTEGRATION IF NOT EXISTS
snowservices_ingress_oauth
TYPE = oauth
OAUTH_CLIENT = snowservices_ingress
ENABLED = true ;
```

e. Create a Compute Pool (like a virtual warehouse)

```
CREATE COMPUTE POOL llm_finetune_pool
MIN_NODES = 1
MAX_NODES = 1
INSTANCE_FAMILY = GPU_7 ;
```

f. Check the status of the Compute Pool.

```
DESCRIBE COMPUTE POOL llm_finetune_pool ;
```

g. Make sure the Compute Pool is in IDLE or ACTIVE state before proceeding with next steps!

	name	state	min_nodes	··· max_nodes	instance_family
1	LLM_FINETUNE_POOL	ACTIVE	1	1	GPU_7

5. Set up a local Conda environment as follows:

a. In a terminal window, run the following command to create a Conda environment:

```
conda create --name llm-finetune-hol -c
https://repo.anaconda.com/pkgs/snowflake python=3.9
```

b. Check your Conda environments to make sure you have the "`llm-finetune-hol`" env is available.

```
conda env list
```

c. Activate "`llm-finetune-hol`" environment.

```
conda activate llm-finetune-hol
```

d. Run the following command to install required packages from Snowflake Anaconda channel:

```
conda install -c https://repo.anaconda.com/pkgs/snowflake
snowflake-snowpark-python pandas notebook
```

e. Edit and update connection.json file with your credentials including that Hugging face token you acquired as part of the pre-requisites step.

Note 1:
Below, the role, warehouse, database, schema, and compute pool are left as such just to align with what I have used for my deployment. Please make sure you update "all" of them appropriate to your credentials.

Note 2:
Also, I had issue with case sensitivity with role, database, schema, and compute pool. So, I made sure I specify upper case in my connection information. But please note that when creating these objects, I don't enforce any case as you can see above in the SQL meta-data commands.

```
{
  "account"   : "<locator>",
  "user"      : "<Username>",
  "password"  : "<Password>",
  "role"      : "VENKSADMIN",
  "warehouse" : "LLM_FINETUNE_WH",
  "database"  : "LLM_FINETUNE_DB",
  "schema"    : "HOL",
  "compute_pool" : "LLM_FINETUNE_POOL",
  "huggingface_token": "<Token>"
}
```

Note 3:
This is simple and easy way to authenticate to Snowflake for this lab but may not a preferred choice since password is put in plain text. If you want more secure way, you could follow the security set up as described in out SPCS 101 QuickStart using Snow CLI:

https://quickstarts.snowflake.com/guide/intro_to_snowpark_container_services/index.html#2

Step 3: Deploy Llama 2 base model to SPCS.

1. Open a terminal window.

```
vmantha@C02D60TJMD6R ~ % pwd
/Users/vmantha
vmantha@C02D60TJMD6R ~ %
```

2. Go to the Lab1 Folder

```
vmantha@C02D60TJMD6R Lab1 % pwd
/Users/vmantha/d1/Work/SnowML/spcs_llm_finetune/Lab1
vmantha@C02D60TJMD6R Lab1 %
```

3. Make sure you activated your Conda environment that you installed above.

```
vmantha@C02D60TJMD6R ~ % conda activate llm-finetune-hol
(llm-finetune-hol) vmantha@C02D60TJMD6R ~ %
```

4. Run your local Jupyter notebook environment.

```
(llm-finetune-hol) vmantha@C02D60TJMD6R Lab1 % jupyter notebook
```

5. Run the notebook "**`llm-lab1-notebook.ipynb`**" end-2-end.

   Please take time to read through each cell if you are not familiar to running a notebook or new to Python, notebook, Snowpark ML etc. The goal behind this SPCS hands-on-lab is for anybody at the least finish the lab "as such" and then identify the use cases that help your company drive custom/fine-tuned LLM models.

Congratulations! If you successfully completed each cell, then you completed the Lab1. In case you are wondering where and who of building an image and deployment, Snowflake took care of it. This is the easy button of deploying a Snowpark ML model to Snowpark Container Services using Snowpark ML Model Registry package.

The following "deploy" code will do all the image building behind the scenes, creates a service, and deploys the service to container services for model inference.

```
llama_model_ref.deploy(
    deployment_name="llama_predict",
    platform=deploy_platforms.TargetPlatform.SNOWPARK_CONTAINER_SERVICES,
    permanent=True,
    options={"compute_pool": COMPUTE_POOL, "num_gpus": 1})
```

The notebook has logging enabled so you will see the detailed information on all things happening behind the scenes.

Step 4: Build and push an image to Snowflake Stage.

This is the first step for Lab2 where we will build an image using local Docker. The image includes all the necessary components (software, meta-data, and instructions) for us to fine-tune the Llama 2 model in Snowpark container services with data in Snowflake. This allows us to:

Interact with Snowflake using Llama2 model running inside Snowpark container services which is secure and performant as data never leaves Snowflake. We bring the process and compute to data.

1. Create Snowflake stages and Image repository to hold image and its associated data and specification files including Jupyter notebook libraries. For finetuning we will run the Jupyter notebook within Snowpark container services. In Lab1 we used our local version of the Jupyter notebook server.

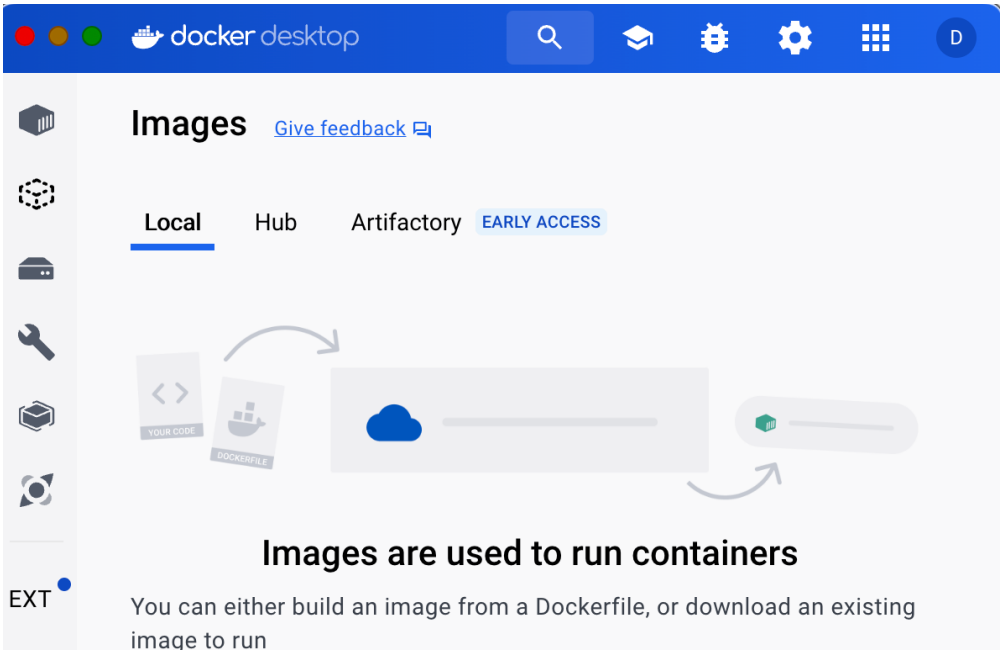
```
USE ROLE venksadmin ;
USE WAREHOUSE llm_finetune_wh ;
USE SCHEMA llm_finetune_db.hol ;

CREATE STAGE yaml_stage   DIRECTORY = ( ENABLE = true )
ENCRYPTION = ( TYPE = 'SNOWFLAKE_SSE' );

CREATE STAGE llm_workspace_stage DIRECTORY = ( ENABLE = true )
ENCRYPTION = ( TYPE = 'SNOWFLAKE_SSE' );

CREATE IMAGE REPOSITORY llm_repo ;
```

2. Get your Docker up and running!



3. Open a terminal window as we will be using Docker command line to build the image. Go to the Lab2 folder.

```
vmantha@C02D60TJMD6R Lab2 % pwd
/Users/vmantha/d1/Work/SnowML/spcs_llm_finetune/Lab2
vmantha@C02D60TJMD6R Lab2 %
```

4. Edit and update "`connection.json`" file with your credentials including that Hugging face token you acquired as part of the pre-requisite's steps. Or you can just copy and paste the same file from Lab1 folder to the Lab2 folder. The "`connection.json`" file for both the labs is exactly the same.

5. Edit and update "`llm-finetune-hol.yaml`" file. This is the specification file that you use to create the service within a compute pool. Any typos or mis-matches in this file will cause issues creating and running the service successfully.

6. Open a terminal window as we will be using Docker command line to build the image.

    a. Check if any images exist already. I generally clean up all my images (assume I am not using them)

```
vmantha@C02D60TJMD6R Lab2 % docker image list
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
vmantha@C02D60TJMD6R Lab2 %
```

    b. Build the image running this command:

```
docker build -no-cache --platform linux/amd64 -t llm-finetune-hol .
```

    Here is the output on my terminal.

```
vmantha@C02D60TJMD6R Lab2 % docker build --no-cache --platform linux/amd64 -t llm-finetune-hol .
[+] Building 135.3s (14/14) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 847B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/rapidsai/rapidsai:23.04-cuda11.8-runtime-ubuntu22.04-py3.8
 => [auth] rapidsai/rapidsai:pull token for registry-1.docker.io
 => [internal] load build context
 => => transferring context: 654B
 => CACHED [1/8] FROM docker.io/rapidsai/rapidsai:23.04-cuda11.8-runtime-ubuntu22.04-py3.8@sha256:4588236a2fa78ac4b6d0112d5eee977e47ce85fe5d76dc4a9aeb2d5f5fdf16ff
 => [2/8] RUN apt-get update && apt-get install -y --no-install-recommends
 => [3/8] WORKDIR /notebooks
 => [4/8] COPY llm-lab2-notebook-fine-tuned-full-dataset.ipynb .
 => [5/8] COPY utils.py .
 => [6/8] COPY connection.json .
 => [7/8] COPY data/transcripts.json .
 => [8/8] RUN conda install -n rapids -c https://repo.anaconda.com/pkgs/snowflake snowflake-snowpark-python pandas jupyterlab
 => exporting to image
 => => exporting layers
 => => writing image sha256:cfd5f2d87b462a44a308c2b169070462c4ad816f9b2765204ea42ff81adeef5b
 => => naming to docker.io/library/llm-finetune-hol

What's Next?
  View a summary of image vulnerabilities and recommendations → docker scout quickview
vmantha@C02D60TJMD6R Lab2 %
```

    c. Check to make sure the image now exists in your local Docker.

```
vmantha@C02D60TJMD6R Lab2 % docker image list
REPOSITORY          TAG       IMAGE ID       CREATED          SIZE
llm-finetune-hol    latest    cfd5f2d87b46   17 minutes ago   17.6GB
```

    d. Tag the image to include the FQDN.

```
docker tag llm-finetune-hol:latest <Org Name>-<Account Name>.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo/llm-finetune-hol:latest
```

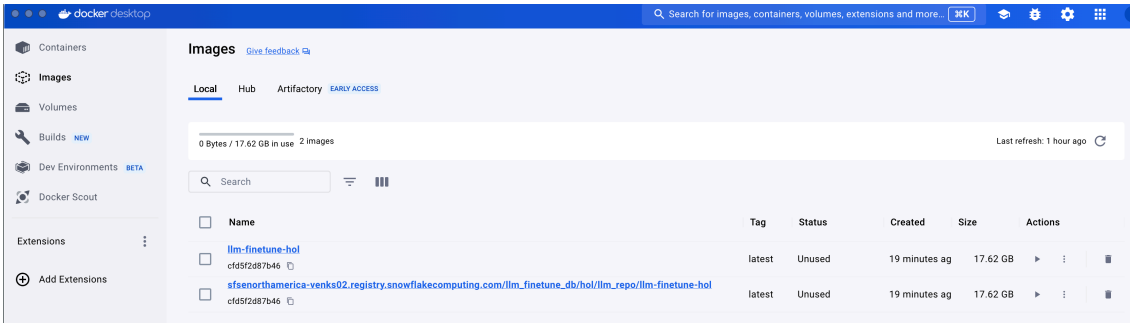
Note: When building and referring to docker images in Snowflake, we use the standard URL as follows:

```
https://<orgname>-<account_name>.snowflakecomputing.com
```

e. Check to make sure now you have both the images in your local Docker. Example below:

```
vmantha@C02D60TJMD6R Lab2 % docker image list
REPOSITORY                                                                                                TAG       IMAGE ID       CREATED          SIZE
llm-finetune-hol                                                                                          latest    cfd5f2d87b46   19 minutes ago   17.6GB
sfsenorthamerica-venks02.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo/llm-finetune-hol   latest    cfd5f2d87b46   19 minutes ago   17.6GB
vmantha@C02D60TJMD6R Lab2 %
```

f. You can also check the images using your local Docker GUI.



g. Authenticate to Snowflake so the before you push the image docker knows the correct credentials are stored and used. If authenticating for the first time, you will need to enter your Snowflake username and password.

```
docker login <Org Name>-<Account Name>.registry.snowflakecomputing.com
```

h. Finally push the image to Snowflake stage. This step requires a good bandwidth between your local PC or VM to your Snowflake instance. If for some reason you don't succeed for the first time (time-out issues) re-run the same command multiple times to push all the layers of the docker image to Snowflake.

```
docker push <Org Name>-<Account Name>.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo/llm-finetune-hol:latest
```

Following error may occur which basically means the image took too long to push to Snowflake. If so, just rerun the command again as Docker will pick-up where it left off.

```
error parsing HTTP 408 response body: invalid character 'r'
looking for beginning of value: "request timeout"
```

i. You can check the images in the Snowflake repositories as:

```
SHOW IMAGE REPOSITORIES ;
```

There should be two repositories. The first repository is created automatically when we did the Snowpark ML Deploy function. The second repository is manually uploaded using the steps above.

created_on	name	database_name	schema_name	repository_url	owner
2024-01-11 19:22:49.404 -0800	LLM_REPO	LLM_FINETUNE_DB	HOL	sfsenorthamerica-venks02.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo	VENKSADMIN
2024-01-11 17:02:29.492 -0800	SNOWML_REPO	LLM_FINETUNE_DB	HOL	sfsenorthamerica-venks02.registry.snowflakecomputing.com/llm_finetune_db/hol/snowml_repo	VENKSADMIN

Following command lists the images in a specific repository.

```
SELECT
SYSTEM$REGISTRY_LIST_IMAGES('/llm_finetune_db/hol/llm_repo');
```

Step 5: Finetune Llama 2 using custom data.

In Lab1, we ran the Jupyter notebook using our local and VM based Jupyter server. For Lab2 we will run the Jupyter server within Snowpark container services that allows us to securely interact with Snowflake. In short, we are bringing the process and software to data which is the key value proposition of Snowpark container services.

Before we create a service in our compute pool, let's clean-up the previous service that was created automatically with Snowpark ML Model registry deploy function.

Run the following command to check the services running in out compute pool.

```
SHOW SERVICES IN COMPUTE POOL llm_finetune_pool ;
```

Drop any service that is running to give all the compute pool resources for our Lab2.

```
DROP SERVICE llm_finetune_db.hol.<Service_Name> ;
```

Before we create the service, let's upload the spec file to the stage as we will point the service to the spec file.

Using Snowsight we will upload "`llm-finetune.yaml`" to "`yaml_stage`". Once uploaded let's make sure the file exists in the stage:

```
LIST @yaml_stage ;
```

name	size	md5
yaml_stage/llm-finetune-hol.yaml	611	bfc0249d71c0d389a1a31959d4f7c817

Let's make sure the compute pool is ready. If not, use the RESUME command to bring it back to IDLE state.

```
SHOW COMPUTE POOLS ;

ALTER COMPUTE POOL llm_finetune_pool RESUME ;
```

Wait for the compute pool to be in IDLE state before executing the following command:

```
CREATE SERVICE llm_finetune_service
IN COMPUTE POOL LLM_FINETUNE_POOL
FROM @yaml_stage
SPEC = 'llm-finetune-hol.yaml' ;
```

Once the service is created, run the following SQL command to make sure the service is in READY state. If for any reason, something is not typed or updated, the service will fail to start!

```
SELECT
       v.value:containerName::varchar container_name
      ,v.value:status::varchar status
      ,v.value:message::varchar message
FROM
      (SELECT
PARSE_JSON(system$get_service_status('LLM_FINETUNE_SERVICE'))) t,
      LATERAL FLATTEN(input => t.$1) v;
```

Once the service in READY state, run the following command to retrieve the URL.

```
SHOW ENDPOINTS IN SERVICE llm_finetune_service ;
```

Copy the value from the INGRESS_URL column and past it in the browser which should something like this:

```
bex4byi-sfsenorthamerica-venks02.snowflakecomputing.app
```

This is the endpoint to the Jupyter Service running in Snowpark container services that we just created! Once you paste the URL in the browser, you will be asked for your username and password for the Snowflake. Enter the same credentials you have been using so far to authenticate to Snowflake.

Step 6: Deploy finetuned model to SPCS.

You should see the Jupyter notebook interface that you might be familiar with. Open the notebook and you are ready now to finetune the Llama2 model!

You need to supply your hugging face token in a cell in this notebook. Look for the following cell:

Load Base Model and Tokenizer

```
[4]: # TODO: Set your Hugging Face token
     !huggingface-cli login --token <token_name>
```

Replace <token_name> with your Hugging face token.

Run each cell reading through the notes.

Congratulations again! If you successfully completed each cell, then you completed the Lab 2.

Step 7: Cleanup

```
-- ================
-- Clean up
-- ================

-- Set the context to any default Warehouse, DB, Schema

USE ROLE venksadmin ;
USE WAREHOUSE venks_default_wh ;
USE SCHEMA snowflake_sample_data.tpch_sf1 ;

-- Drop the objects

DROP WAREHOUSE llm_finetune_wh ;
DROP SERVICE llm_finetune_db.hol.llm_finetune_service ;
DROP COMPUTE POOL llm_finetune_pool ;
DROP DATABASE llm_finetune_db ;
DROP SECURITY INTEGRATION snowservices_ingress_oauth ;
```

References

Snowpark Container Services – Tech Primer:
https://medium.com/snowflake/snowpark-container-services-a-tech-primer-99ff2ca8e741

Intro to Snowpark Container Services – Quick Start:
https://quickstarts.snowflake.com/guide/intro_to_snowpark_container_services/index.html#0

Documentation:
https://docs.snowflake.com/en/developer-guide/snowpark-container-services/overview

Connecting to Snowflake from Snowpark Container Services:
https://medium.com/snowflake/connecting-to-snowflake-from-snowpark-container-services-cfc3a133480e

Unlocking the Power of GPU in Snowflake with Snowpark Container Services – Blog: https://www.snowflake.com/blog/unlock-gen-ai-snowpark-gpu-powered-compute/

Appendix A

Following are the names of all the Snowflake objects specified in this document and the code. You are welcome to use the exact name if you happened to be working on an account where you have access and all the right privileges and don't have any of the following names in use.

If you chose to use your own names outside of the account locator and account name, then please make sure you search and change the names at all the locations and references.

Account Locator:	`<account-locator>`
Account Name:	`<Org Name>-<Account Name>`
User:	`<Username>`
Role:	`VENKSADMIN`
Database:	`LLM_FINETUNE_DB`
Schema:	`HOL`
Virtual Warehouse:	`LLM_FINETUNE_WH`
Internal Stages:	`YAML_STAGE, LLM_WORKSPACE_STAGE`
Conda Environment:	`LLM-FINETUNE-HOL`
Compute Pool:	`LLM_FINETUNE_POOL`
Service:	`LLM_FINETUNE_SERVICE`
Image:	`LLM-FINETUNE-HOL`
Container:	`LLM-FINETUNE-HOL`
Image Repository:	`LLM_REPO`

Here are the files where the above object references are made outside of SQL commands.

- `connection.json`
- `llm-finetune-hol.yaml`

Also, please make sure when building the docker image locally, carefully check and update the names. I am showing an image here for readability.

```
-- docker build --no-cache --platform linux/amd64 -t llm-finetune-hol .
-- docker tag llm-finetune-hol:latest <OrgName>-<AccountName>.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo/llm-finetune-hol:latest
-- docker login <OrgName>-<AccountName>.registry.snowflakecomputing.com
-- docker push <OrgName>-<AccountName>.registry.snowflakecomputing.com/llm_finetune_db/hol/llm_repo/llm-finetune-hol:latest
```

As mentioned before, when building the image, you need to use the account name and not the account locator. Please check with your Snowflake admin if you are unsure of your locator and account names.

Appendix B

Here is how I created my custom role VENKSADMIN. Please feel free to create your own custom role with necessary privileges if you don't want to assign elevated privileges to your custom role.

```
USE ROLE SECURITYADMIN ;

CREATE ROLE venksadmin ;
GRANT ROLE venksadmin TO USER venks ;
GRANT ROLE venksadmin TO ROLE SYSADMIN ;

USE ROLE ACCOUNTADMIN ;

GRANT ALL ON ACCOUNT TO ROLE venksadmin ;
-- Ignore warning:
Grant partially executed: privileges [MANAGE LISTING AUTO FULFILLMENT, MANAGE
ORGANIZATION SUPPORT CASES] not granted.

GRANT IMPORTED PRIVILEGES ON DATABASE SNOWFLAKE TO ROLE public ;
CREATE DATABASE snowflake_sample_db FROM SHARE sfc_samples.sample_data ;

ALTER USER venks SET DEFAULT_ROLE = venksadmin ;

USE ROLE venksadmin ;
CREATE OR REPLACE WAREHOUSE venks_basic_wh AUTO_SUSPEND = 300, INITIALLY_SUSPENDED =
true ;

SHOW WAREHOUSES ;
ALTER USER venks SET DEFAULT_WAREHOUSE = venks_basic_wh ;
```

*** The End ***