

LLM supporting tools that make the ecosystem

Paul Rodriguez
San Diego Supercomputer Center

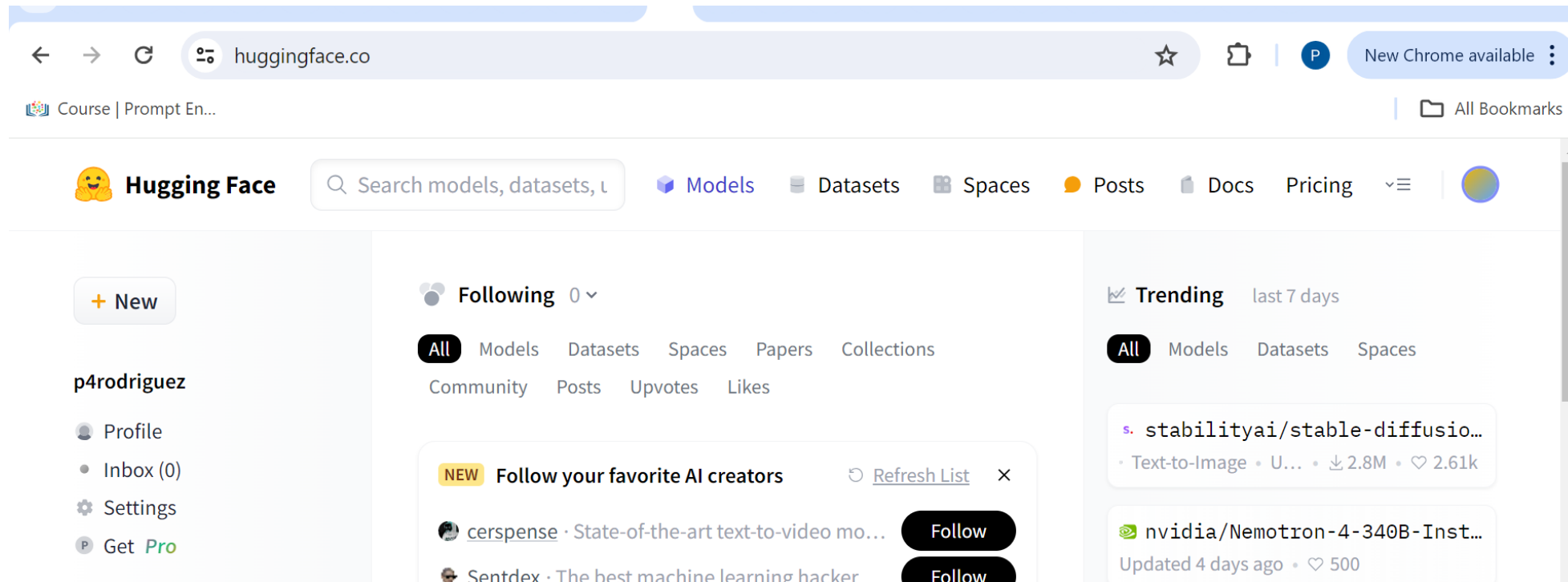
2024 CIML Summer Institute

Outline

- **The Hugging Face ecosystem**
- **Langchain**
- **Hugging Face and Langchain tutorial discussion/demo**

Hugging Face Hub

- **huggingface.com** is a hub of models, data, tutorials for using AI models



Using Hugging Face

- huggingface provides python packages to make it (relatively) easy to run models
- Using models, data require an authentication token
- Some of the main packages are:
 - pipeline: to run inferencing
 - diffusers: for diffusion models
 - transformers: for LLMs
 - accelerate: for efficient and/or parallel execution
 - datasets: to access data from hub

Hugging Face Abstractions

- The pipeline function is built on more basic functions:

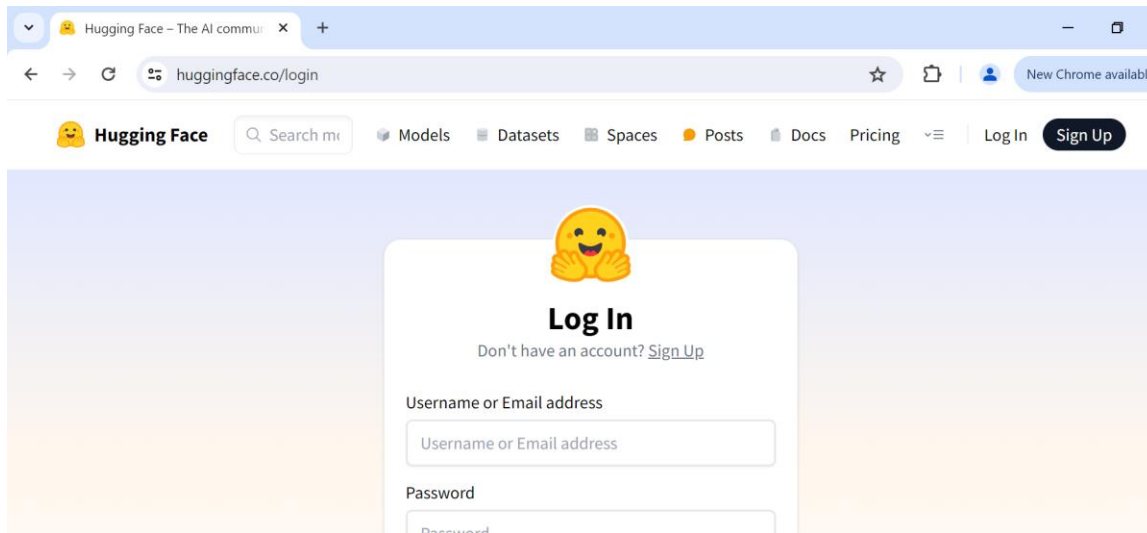
get “config” (to get the model architecture)
get “model” (to download a pretrained model)
get “tokenizer” (to get the appropriate tokenizer)

- These can be invoked more directly if you want to do a pre-training or fine-tuning implementation, or experimentation

Getting account set up

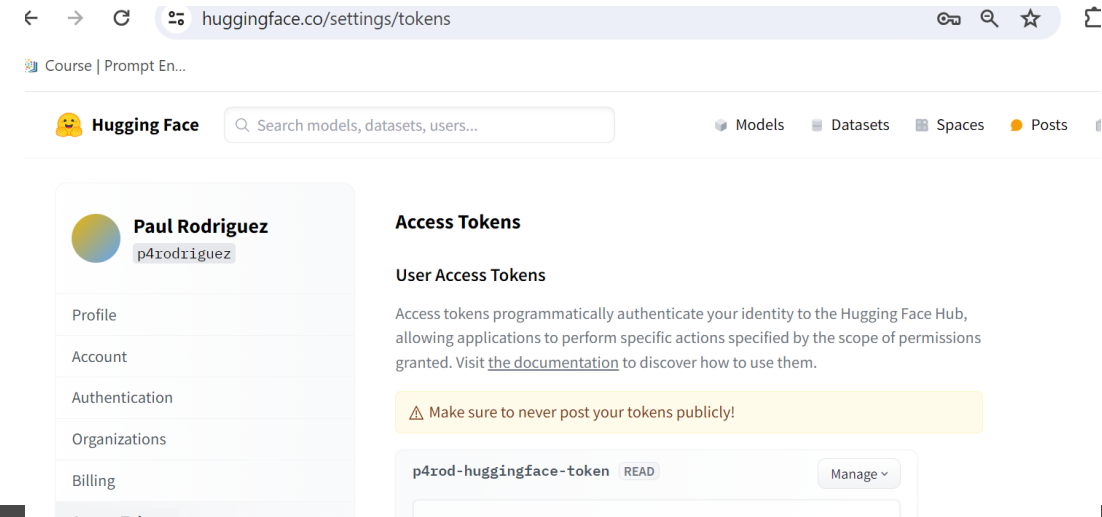
- Create an account on huggingface and get authentication token

huggingface.co/login



The screenshot shows the Hugging Face login page. At the top, there's a navigation bar with the Hugging Face logo, a search bar, and links for Models, Datasets, Spaces, Posts, Docs, Pricing, Log In, and Sign Up. The main content area features a large 'Log In' button with a smiling emoji icon. Below the button, there's a link for 'Don't have an account? Sign Up'. The login form has two input fields: 'Username or Email address' and 'Password', both with placeholder text.

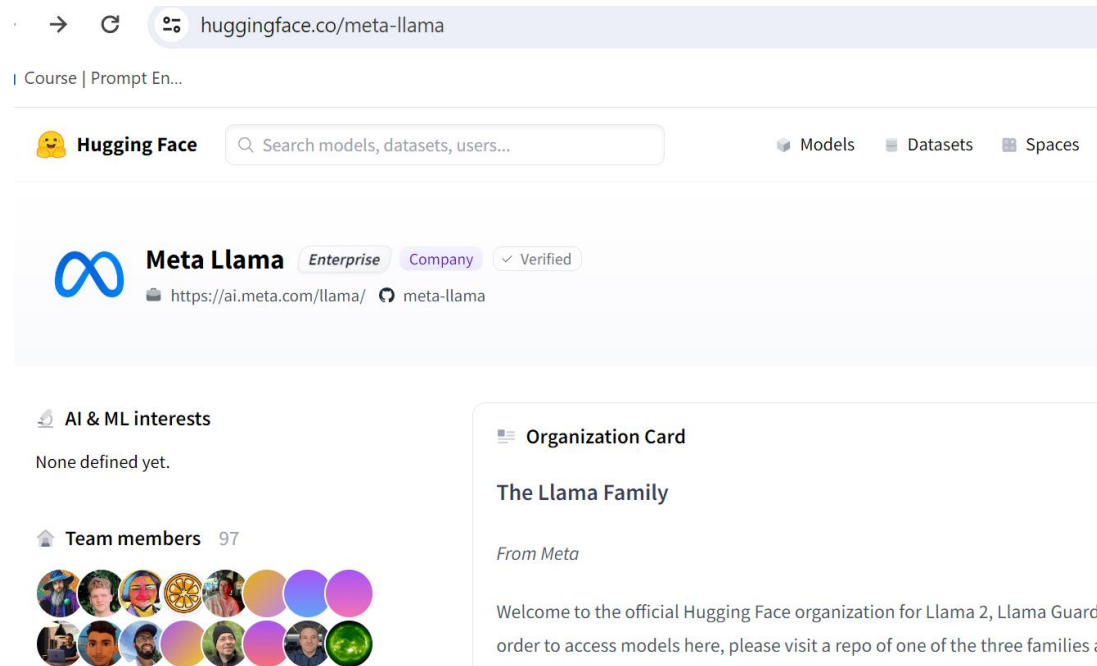
Account -> settings -> get token



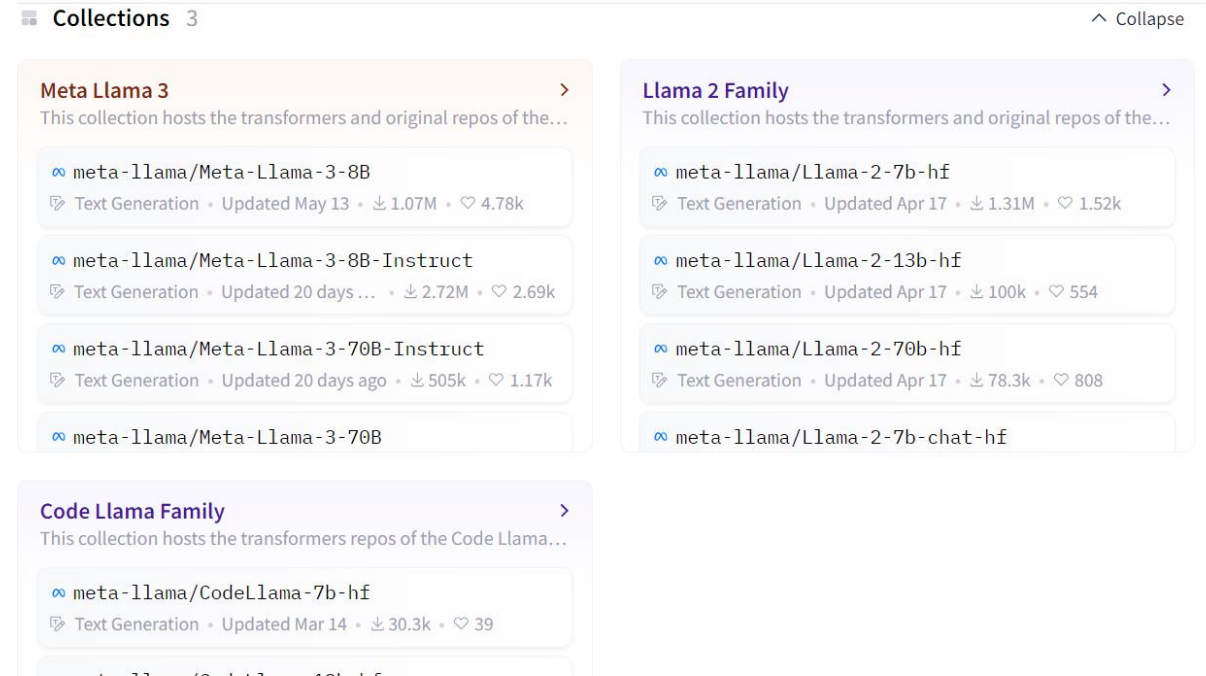
The screenshot shows the Hugging Face settings page, specifically the 'Access Tokens' section. The user's profile is visible on the left, showing the name 'Paul Rodriguez' and the username 'p4rodriguez'. The 'Access Tokens' section on the right explains that access tokens are used for programmatic authentication. It includes a warning: 'Make sure to never post your tokens publicly!'. Below this, there's a table of access tokens. The first token is 'p4rod-huggingface-token' with a 'READ' scope and a 'Manage' button.

Hugging Face model repo

- **Example:** Meta has a family of Llama models that vary by size, response training, 'hf' format, release date, etc..



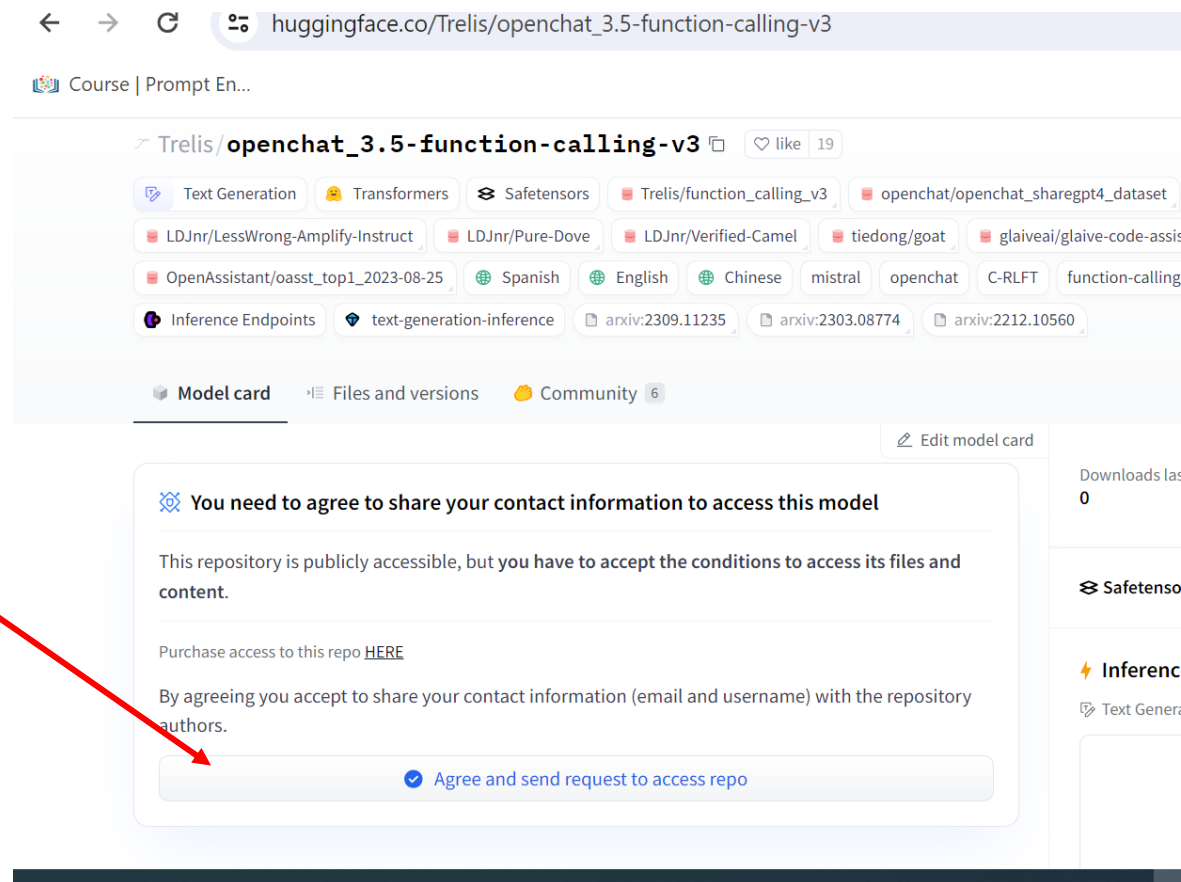
The screenshot shows the Hugging Face profile page for Meta Llama. The browser address bar displays 'huggingface.co/meta-llama'. The page header includes 'Course | Prompt En...'. The main navigation bar features the Hugging Face logo, a search bar, and links for 'Models', 'Datasets', and 'Spaces'. The profile card for 'Meta Llama' is visible, showing the organization's name, a verified status, and links to their website and GitHub. Below the profile card, there are sections for 'AI & ML interests' (None defined yet) and 'Team members' (97 members). An 'Organization Card' titled 'The Llama Family' is also present, with a welcome message for the official Hugging Face organization for Llama 2, Llama Guard, and Llama Guard.



The screenshot shows the 'Collections' page on Hugging Face, specifically for the 'Meta Llama 3' and 'Llama 2 Family' collections. The 'Meta Llama 3' collection is highlighted, showing a list of models including 'meta-llama/Meta-Llama-3-8B', 'meta-llama/Meta-Llama-3-8B-Instruct', 'meta-llama/Meta-Llama-3-70B-Instruct', and 'meta-llama/Meta-Llama-3-70B'. The 'Llama 2 Family' collection is also visible, showing models like 'meta-llama/Llama-2-7b-hf', 'meta-llama/Llama-2-13b-hf', 'meta-llama/Llama-2-70b-hf', and 'meta-llama/Llama-2-7b-chat-hf'. The 'Code Llama Family' collection is partially visible at the bottom, showing 'meta-llama/CodeLlama-7b-hf'.

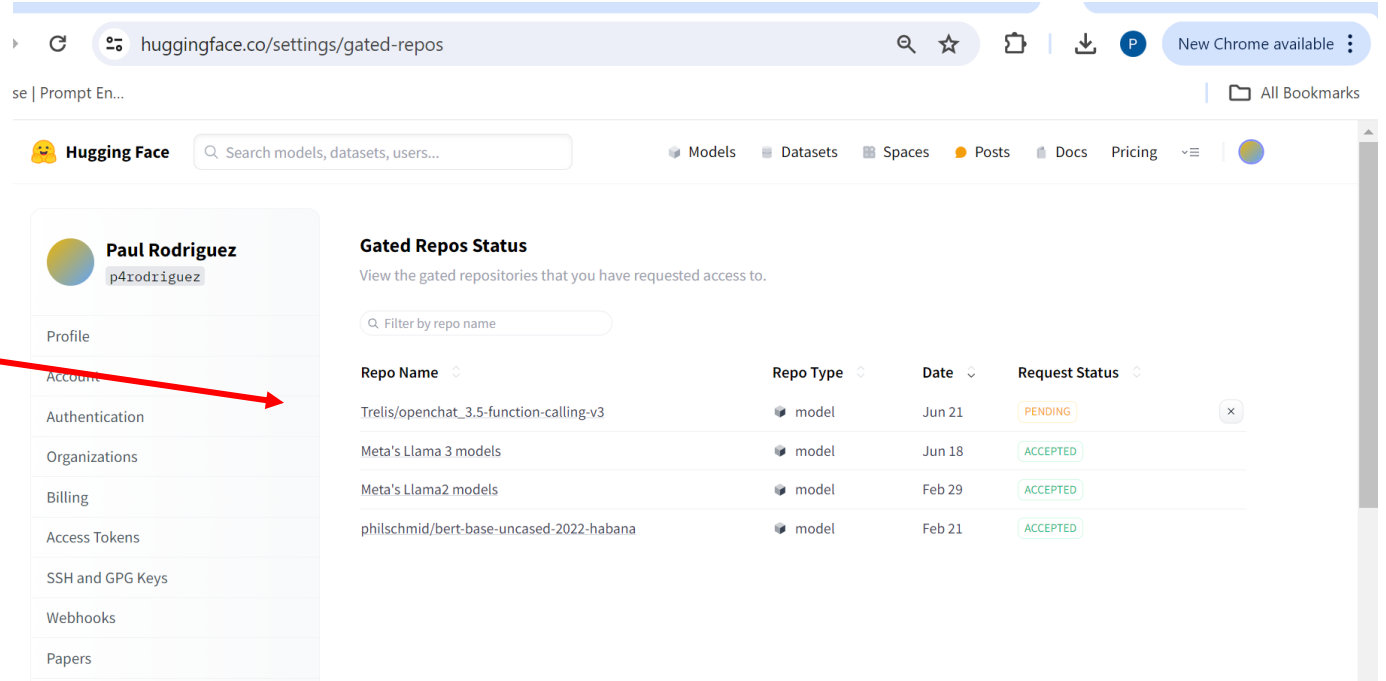
Hugging Face model repo

- Some models or data require that you request access here



Hugging Face model repo

- Some models or data require that you request access
- Then go to account->setting->gated repo

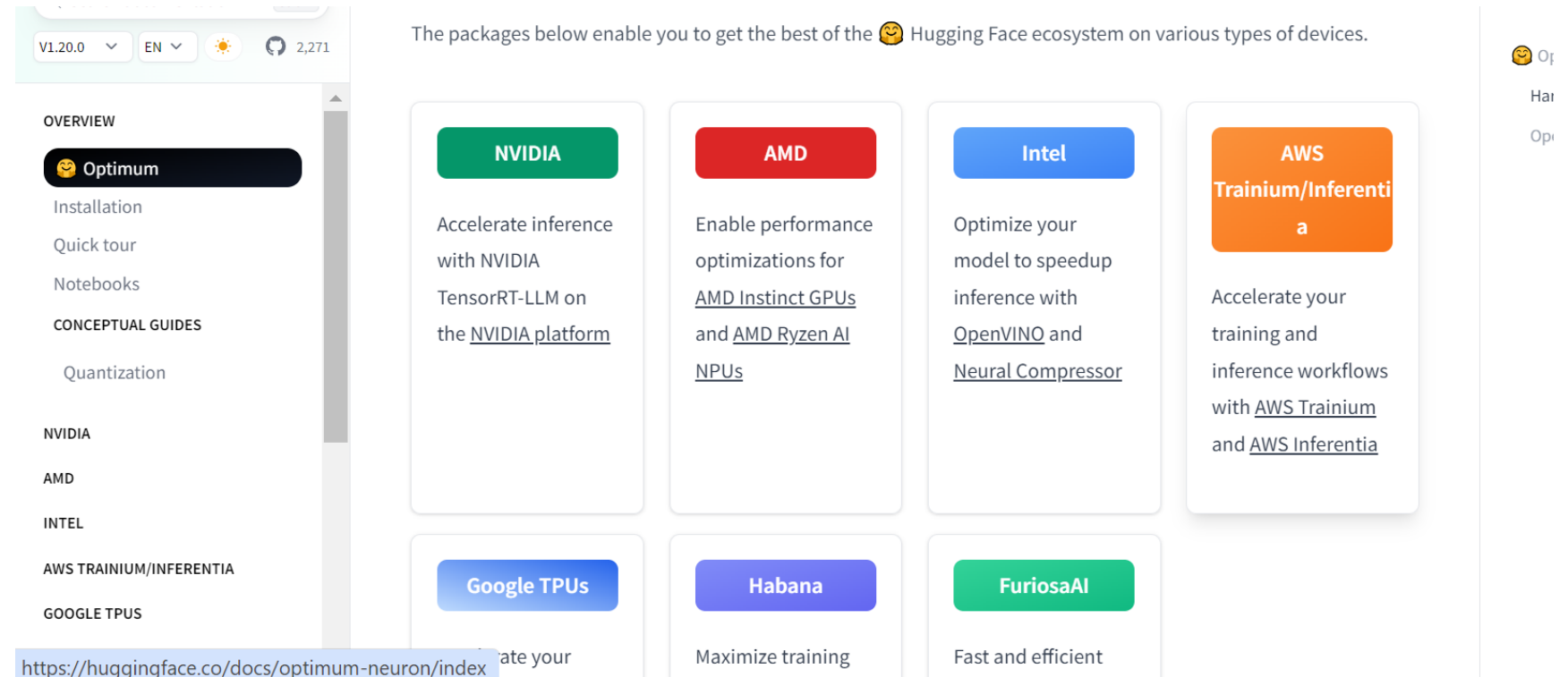


The screenshot shows the Hugging Face website with the user 'Paul Rodriguez' (p4rodriguez) logged in. The left sidebar contains a menu with items: Profile, Account, Authentication, Organizations, Billing, Access Tokens, SSH and GPG Keys, Webhooks, and Papers. The 'Account' item is highlighted, and a red arrow points from it to the 'Gated Repos Status' section on the right. The 'Gated Repos Status' section displays a table of requested access to gated repositories.

Repo Name	Repo Type	Date	Request Status
Trelis/openchat_3.5-function-calling-v3	model	Jun 21	PENDING
Meta's Llama 3 models	model	Jun 18	ACCEPTED
Meta's Llama2 models	model	Feb 29	ACCEPTED
philschmid/bert-base-uncased-2022-habana	model	Feb 21	ACCEPTED

Hardware Partners

- Hugging Face also incorporates accelerator libraries into their packages



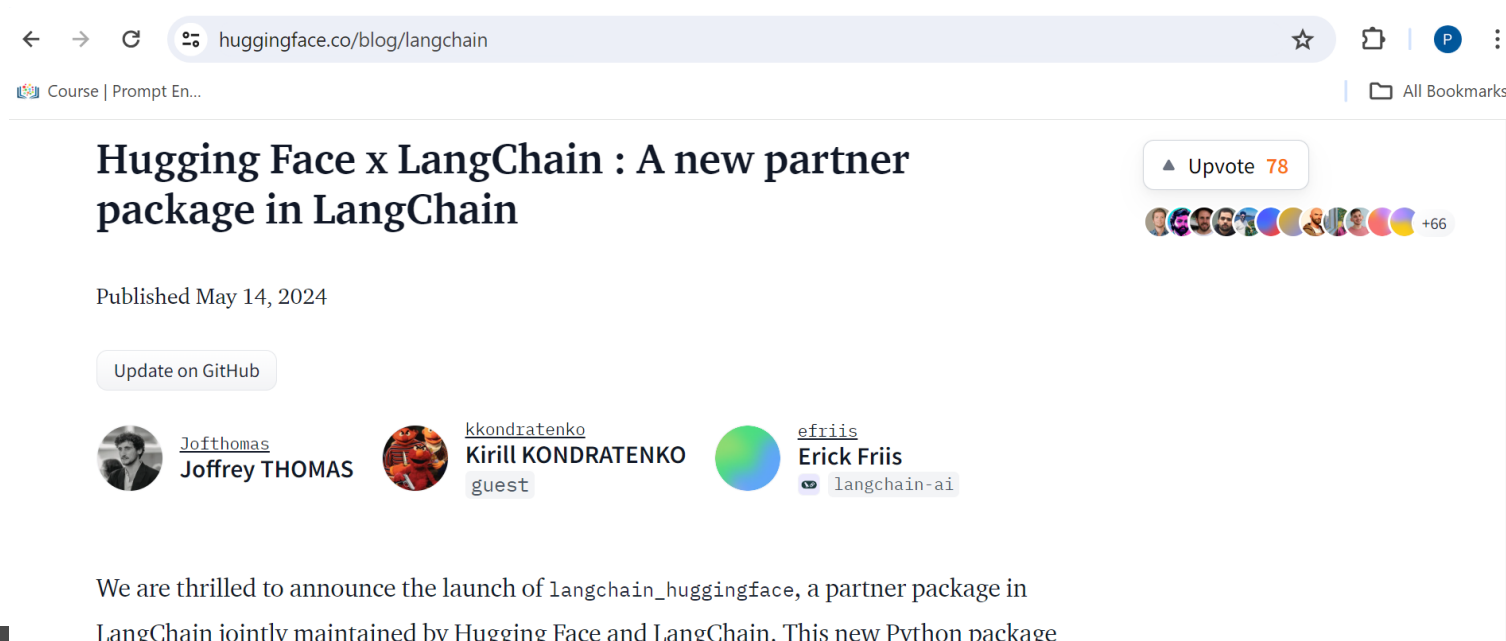
The screenshot displays the Hugging Face Optimum documentation page. The left sidebar contains a navigation menu with sections: OVERVIEW (with 'Optimum' selected), Installation, Quick tour, Notebooks, CONCEPTUAL GUIDES (with 'Quantization' selected), NVIDIA, AMD, INTEL, AWS TRAINIUM/INFERENCEIA, and GOOGLE TPUS. The main content area features a header stating, 'The packages below enable you to get the best of the 🤗 Hugging Face ecosystem on various types of devices.' Below this, there are eight cards representing different hardware partners:

- NVIDIA**: Accelerate inference with NVIDIA TensorRT-LLM on the [NVIDIA platform](#)
- AMD**: Enable performance optimizations for [AMD Instinct GPUs](#) and [AMD Ryzen AI NPU](#)s
- Intel**: Optimize your model to speedup inference with [OpenVINO](#) and [Neural Compressor](#)
- AWS Trainium/Inferentia**: Accelerate your training and inference workflows with [AWS Trainium](#) and [AWS Inferentia](#)
- Google TPUs**: Maximize training
- Habana**: Fast and efficient
- FuriosaAI**: Fast and efficient

A URL bar at the bottom left shows the address: <https://huggingface.co/docs/optimum-neuron/index>.

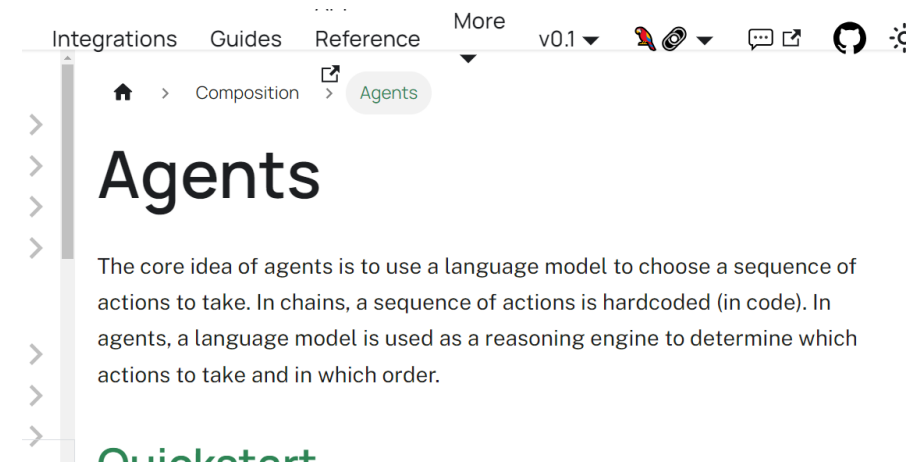
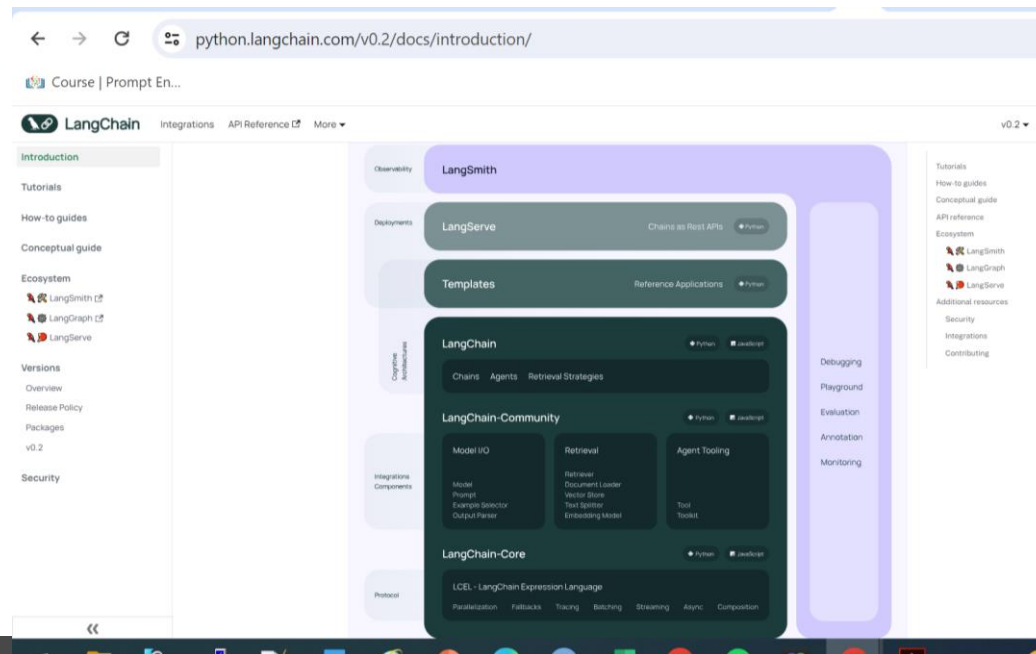
Partner packages

- Hugging Face incorporates other packages
- For example, Langchain helps run RAG applications by:
 - accessing PDF files or URL text as raw documents
 - creating database of documents (split into chunks and vectorized)
 - Setting up prompts that include relevant contexts



Langchain:

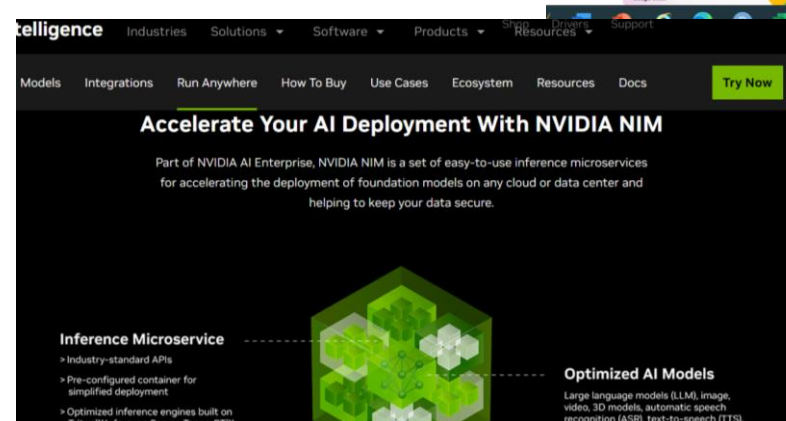
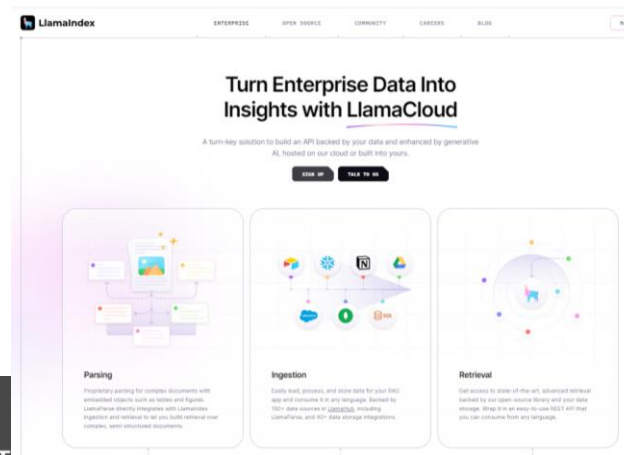
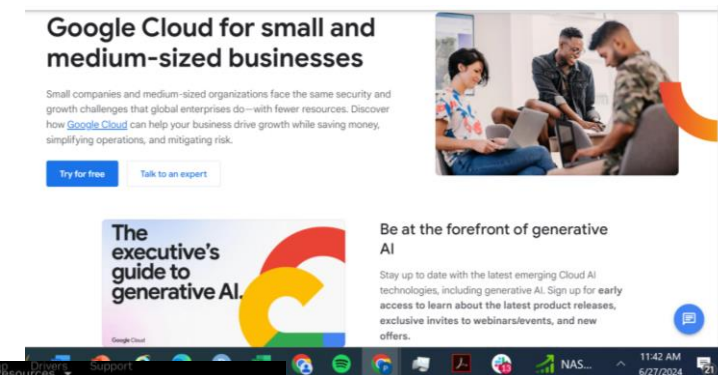
- Not an LLM provider (like openAI, or huggingface) but provides a standard interface
- Has ecosystem of tools beyond RAG – e.g. Agents: “a language model is used as a reasoning engine to determine which actions to take and in which order.”



Other ecosystems to run LLMs

Many Big Tech firms have LLM services that provide ‘turn key’ solutions/tools for RAG, with APIs, open models, interfaces, deployment., etc.. geared for businesses.

- Nvidia (NeMo)
- Google (cloud) AI
- LlamaIndex (meta)
- Etc...



Keras extension for NLP

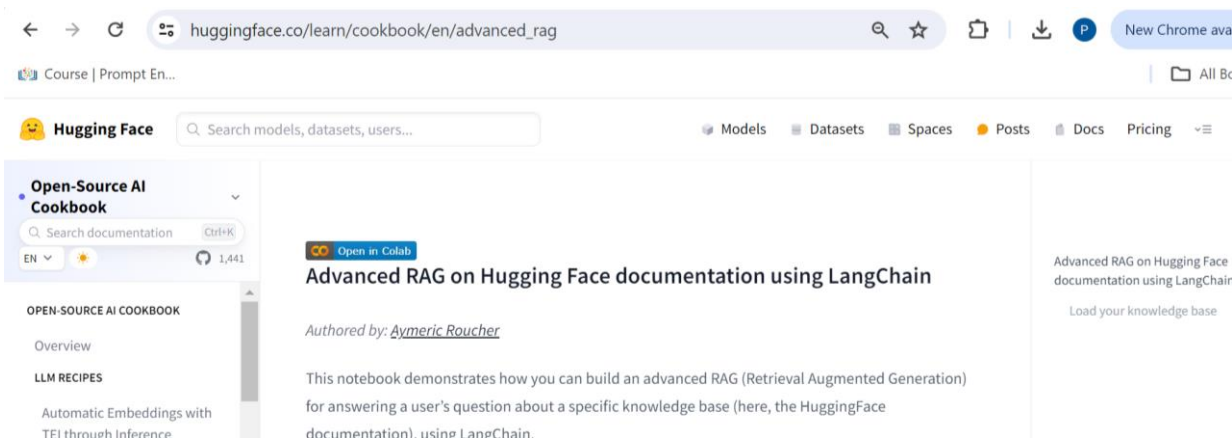
- KerasNLP also has models, data, functions to run different LLMs (no tokens)
- Includes pre-trained LLMs base or full model, for example:
 - GPT2Backbone the model without task specific output layers
 - GPT2CausalLM the model with output predictions
 - GPT2CausalLMPreprocessor the preprocessor that feeds model.fit

Expanse Demo/Exercise setting up HuggingFace and Langchain

Let's combine 2 Jupyter notebook LLM tutorials

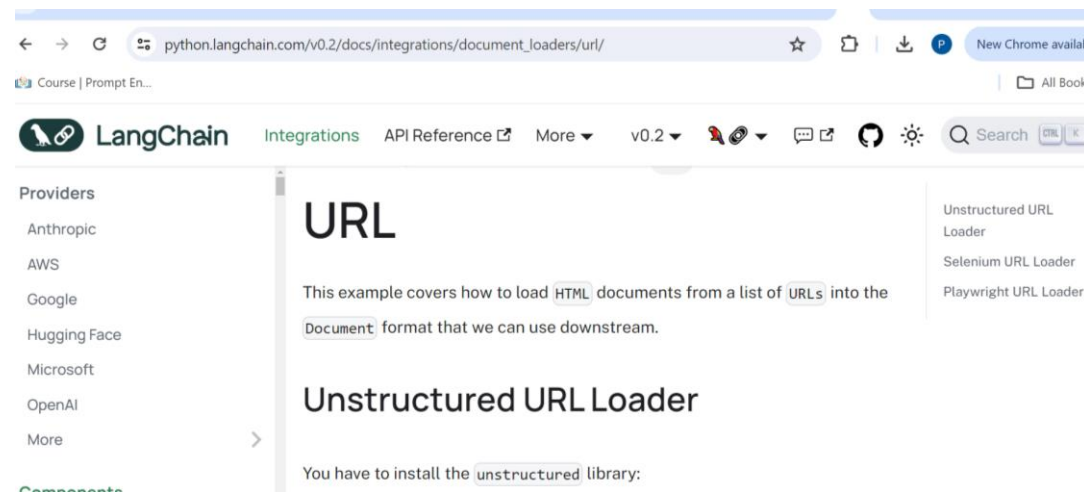
- Start with Huggingface/Langchain for RAG

https://huggingface.co/learn/cookbook/en/advanced_rag

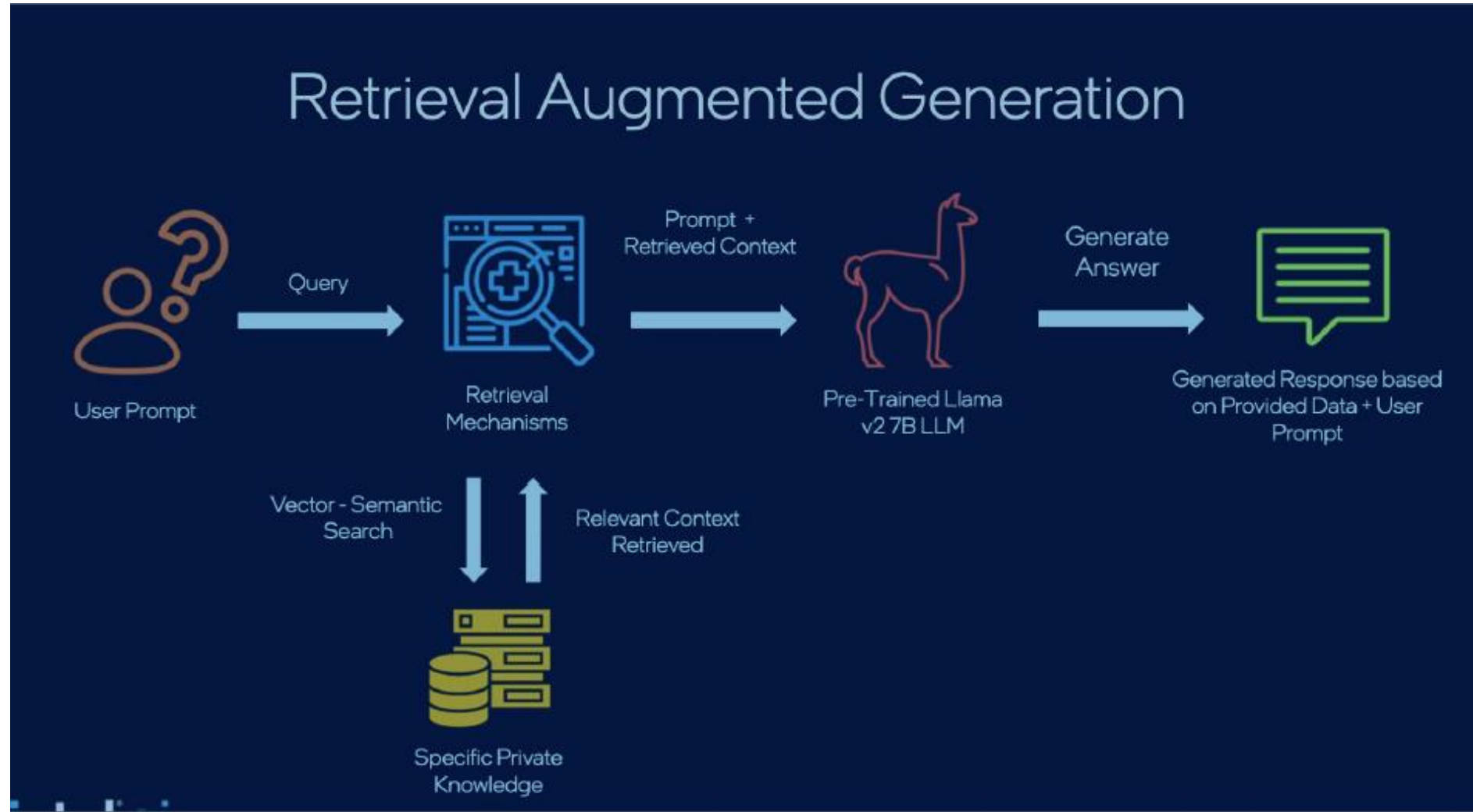


- But use a URL about SLURM for the raw documents

https://python.langchain.com/v0.2/docs/integrations/document_loaders/url/



RAG overview (from the Mai's slide)



Running notebook tutorials on Expanse

- On Expanse we can run a singularity container and start up Jupyter notebooks on a GPU

Pro: easy to follow instructions

Con: some things not obvious

Running notebook tutorials on Expanse

- On Expanse we can run a singularity container and start up Jupyter notebooks on a GPU

Pro: easy to follow instructions

Con: some things not obvious

OR

- On Expanse we can get a GPU node and use the command line to run singularity and set up commands

Pro: you get to see how/where everything is set up

Con: more typing

Running notebook tutorials on Expanse

- On Expanse we can run a singularity container and start up Jupyter notebooks on a GPU

Pro: easy to follow instructions

Con: some things not obvious

OR

- On Expanse we can get a GPU node and use the command line to run singularity and set up commands

Pro: you get to see how/where everything is set up

Con: more typing

added benefit: directly translates to a batch script

Hugging Face set up using a notebook

- Many tutorials use Jupyter notebooks with “!pip install” commands

```
In [ ]: ► !pip install --upgrade 'huggingface_hub[pytorch,cli]'  
!pip install transformers  
!pip install accelerate
```

```
In [ ]: ► #restart kernel to reload .local modules after installing hugging face hub  
import huggingface_hub  
from transformers import AutoTokenizer  
import transformers  
import torch
```

*Packages will be in
/home/userid/.local*

*After installing (~5 min)
restart kernel to reset
variables*

Hugging Face set up using a notebook

- Many tutorials use Jupyter notebooks with “!pip install” commands

```
In [ ]: ▶ !pip install --upgrade 'huggingface_hub[pytorch,cli]'  
        !pip install transformers  
        !pip install accelerate
```

```
In [ ]: ▶ #restart kernel to reload .local modules after installing hugging face hub  
        import huggingface_hub  
        from transformers import AutoTokenizer  
        import transformers  
        import torch
```

*Packages will be in
/home/userid/.local*

*After installing (~5 min)
restart kernel to reset
variables*

*BEWARE, you might
create conflicts with other
set ups using .local*

exercise

- jupyter_gpu_shared_pytorch
- Review and run cells
- Try changing chunk size?

The screenshot displays a JupyterLab web interface in a browser. The address bar shows the URL: `unwashed-earwig-motto.expense-user-content.sdsc.edu/lab/tree/HFace/Rag_example_v3.ipynb`. The interface includes a top toolbar with various icons, a left sidebar with a file explorer, and a main editor area.

File Explorer (Left Sidebar):

- Search: Filter files by name
- Path: / HFace /
- Table:

Name	Last Modified
T1	14 seconds ago
Rag_example_v3.ipynb	12 minutes ago

Main Editor Area:

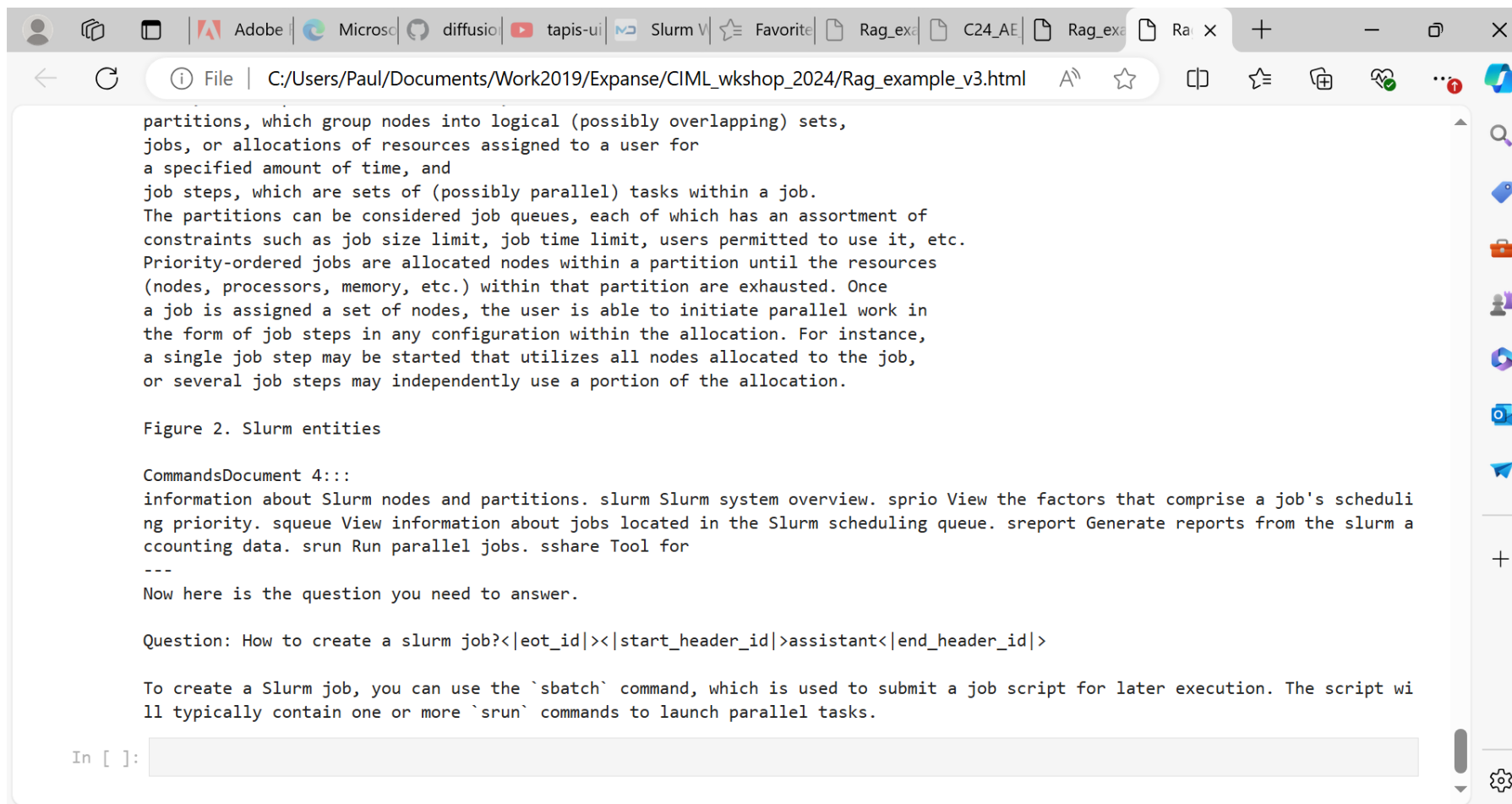
- Tab: Rag_example_v3.ipynb
- Language: Python 3 (ipykernel)
- Code Editor Content:

```
[ ]: #The tutorial would have your run the following
      #But it takes too long for us to wait,
      # so we'll just use pre-installed folders
```

Bottom Status Bar:

- Simple: ☐
- 0 \$ 2
- Python 3 (ipykernel) | Idle
- Mode: Edit
- Ln 3, Col 47
- Rag_example_v3.ipynb
- 1

Sample output



The screenshot shows a web browser window with the address bar displaying 'C:/Users/Paul/Documents/Work2019/Expanse/CIML_wkshop_2024/Rag_example_v3.html'. The page content includes a paragraph about Slurm partitions and job steps, followed by a caption 'Figure 2. Slurm entities'. Below this is a section titled 'CommandsDocument 4:::' listing various Slurm commands and their functions. At the bottom, there is a chat interface with a 'Question:' and an 'assistant:' response.

partitions, which group nodes into logical (possibly overlapping) sets, jobs, or allocations of resources assigned to a user for a specified amount of time, and job steps, which are sets of (possibly parallel) tasks within a job. The partitions can be considered job queues, each of which has an assortment of constraints such as job size limit, job time limit, users permitted to use it, etc. Priority-ordered jobs are allocated nodes within a partition until the resources (nodes, processors, memory, etc.) within that partition are exhausted. Once a job is assigned a set of nodes, the user is able to initiate parallel work in the form of job steps in any configuration within the allocation. For instance, a single job step may be started that utilizes all nodes allocated to the job, or several job steps may independently use a portion of the allocation.

Figure 2. Slurm entities

CommandsDocument 4:::
information about Slurm nodes and partitions. slurm Slurm system overview. sprio View the factors that comprise a job's scheduling priority. squeue View information about jobs located in the Slurm scheduling queue. sreport Generate reports from the slurm accounting data. srun Run parallel jobs. sshare Tool for

Now here is the question you need to answer.

Question: How to create a slurm job?<|eot_id|><|start_header_id|>assistant<|end_header_id|>

To create a Slurm job, you can use the `sbatch` command, which is used to submit a job script for later execution. The script will typically contain one or more `srun` commands to launch parallel tasks.

In []:

Hugging Face set up using the command line

Main Steps

1. Request a GPU node
2. ssh into that node
3. Load modules
4. Run “singularity shell” command
5. Run “pip install” commands
6. Login to hugging face
7. Run your script

After you this a while, you might prefer to work from command line to see all the details

Hugging Face set up using the command line

Main Steps

1. Request a GPU node

```
[train113@login02 ciml-summer-institute-2024]$  
[train113@login02 ciml-summer-institute-2024]$ jupyter-gpu-shared-pytorch
```

2. ssh into that node

```
[p4rodrig@login01 HFace]$ squeue -u p4rodrig  
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)  
      31537549 gpu-debug  hface-gp p4rodrig  R        0:01      1 exp-7-59  
[p4rodrig@login01 HFace]$  
[p4rodrig@login01 HFace]$  
[p4rodrig@login01 HFace]$ ssh exp-7-59
```

Hugging Face set up using the command line

Main Steps

3. Load modules

```
[p4rodrig@exp-7-59 HFace]$ module purge
[p4rodrig@exp-7-59 HFace]$ module load gpu
[p4rodrig@exp-7-59 HFace]$ module load slurm
[p4rodrig@exp-7-59 HFace]$
[p4rodrig@exp-7-59 HFace]$ module load singularitypro/3.11
```

4. Run “singularity shell” command

```
[p4rodrig@exp-7-59 ~]$ singularity shell /cm/shared/apps/containers/singularity/pytorch
h-latest.sif --nv --bind /expance,/scratch
Singularity>
```

Hugging Face set up using the command line

Main Steps

5. Run “pip install” commands (takes ~5minutes)

```
Singularity> pip install --upgrade huggingface_hub[pytorch,cli] transformers  
ts  
pip install --upgrade langchain sentence-transformers langchain-community  
pip install --upgrade bitsandbytes pypdf faiss-gpu pydantic
```

6. Login to hugging face

```
Singularity> /home/p4rodrig/.local/bin/huggingface-cli login --token hf_cx0F
```

7. Run your script

```
Singularity> python3 Rag_example_v1.py > rag_stdout.txt
```

Hugging Face set up using the command line

Other Steps

Review output:

```
Now here is the question you need to answer.  
  
Question: How to create a slurm job?<|eot_id|><|start_header_id|>assistant<|end_header_id|>  
  
According to the provided documents, to create a Slurm job, you can use the `sbatch` command. This command is used to submit a job script for later execution. The script will typically contain one or more `srun` commands to launch parallel tasks.  
  
Here is an example of how to use `sbatch`:  
  
`sbatch [options] script.sh`  
  
Where `script.sh` is the name of the job script file, and `[options]` are optional parameters that can be used to specify various settings for the job.  
  
For more information on how to use `sbatch` and other Slurm commands, you can refer to the Slurm documentation, which is available in the provided documents.
```

Hugging Face set up using the command line

Other Steps

Review cached models in
~/.cach/huggingface/hub

```
[p4rodrig@login02 hub]$ du -sh *
417M    models--bert-base-cased
572K    models--bert-base-uncased
512     models----home--p4rodrig--HW2-tests--Yelp1m--bert-it-
-bert-it-vocab.txt
13G     models--meta-llama--Llama-2-7b-chat-hf
15G     models--meta-llama--Meta-Llama-3-8B-Instruct
512     version.txt
[p4rodrig@login02 hub]$
```

Hugging Face in a batch script

Other Steps

move .local to new folder, and export PYTHONPATH to avoid potential conflicts

Use “singularity exec” to launch commands in a batch script

```
# Set up paths for python to find packages
export PYTHONPATH=/home/$USER/Local_HFgpu-latest/lib/python3.1
export PATH=/home/$USER/Local_HFgpu-latest/local/bin:$PATH
echo "----- paths -----"
echo $PYTHONPATH
echo $PATH

#You can run hugging face login first (it will put the token i
# and also run it within singularity (b/c it was installed tha
singularity exec --nv --bind /exppanse,/scratch /cm/shared/apps
pytorch-latest.sif /home/$USER/Local_HFgpu-latest/bin/huggingf

#Now you can run the rag example
singularity exec --nv --bind /exppanse,/scratch /cm/shared/apps
pytorch-latest.sif python3 Rag_example_v1.py > rag_stdout.txt
```

Text for command lines

- The above instructions are in “CommandLineTexts.txt” file

```
[p4rodrig@login02 HFace]$ more CommandLineTexts.txt
1 Request a GPU node
/cm/shared/apps/sdsc/galyleo/galyleo.sh launch -A use300 -p gpu-debug -n 1 -l
  -G 1 -t 00:30:00 -e singularitypro/3.11 --nv --bind /expance,/scratch -s /c
ared/apps/containers/singularity/pytorch/pytorch-latest.sif
NOTE: change use300 to quest
```

- Also look at the run-pyt-gpu.. batch script

```
[p4rodrig@login02 HFace]$ more run-pyt-gpudebug-latest.sh
#!/usr/bin/env bash
#SBATCH --job-name=hface-gpu
#SBATCH --account=use300
# -----
#SBATCH --partition=cpu-debug
```


Hugging Face set up using the command line

Other details like:

Login to hugging face

```
Singularity> /home/p4rodrig/.local/bin/huggingface-cli login --token hf_cx0E
```

Review cached models in
~/.cach/huggingface/hub

```
[p4rodrig@login02 hub]$ du -sh *
417M    models--bert-base-cased
572K    models--bert-base-uncased
512     models---home--p4rodrig--HW2-tests--Yelp1m--bert-it-
-bert-it-vocab.txt
13G     models--meta-llama--Llama-2-7b-chat-hf
15G     models--meta-llama--Meta-Llama-3-8B-Instruct
512     version.txt
[p4rodrig@login02 hub]$
```


end