

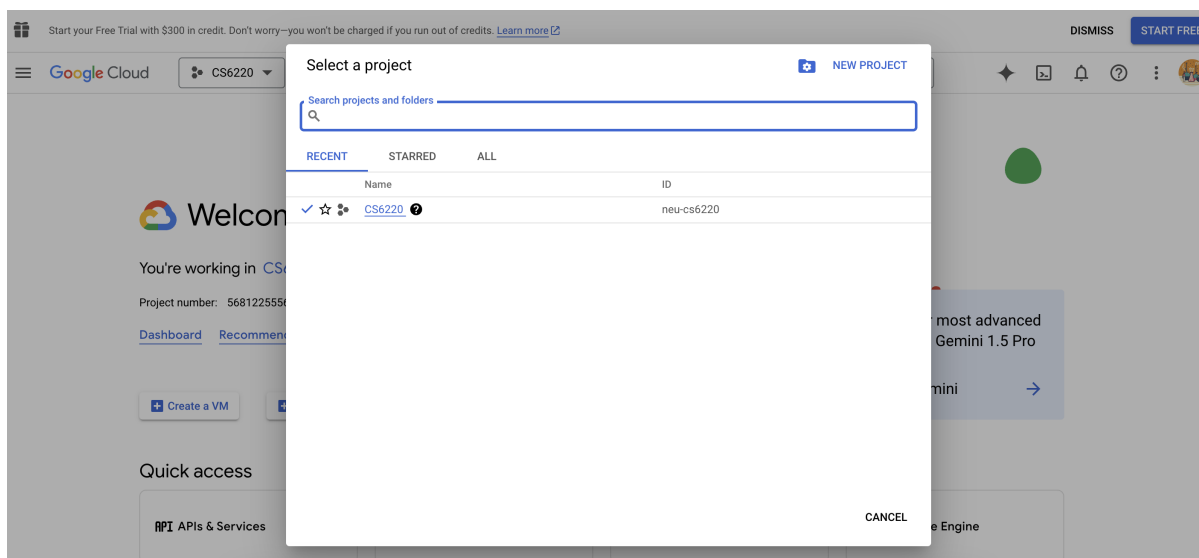


CS 6120 Natural Language Processing — Lab 3

January 25, 2025 (Week 3)

YOUR NAME + LDAP

In this lab, we're going to work in [Google's Workbench](#), an alternative to Google Colab, when you'd like more flexibility beyond Python notebooks. Workbench is a cloud offering in Google Cloud Platform (GCP), which operates in JupyterLab and the notebook environment. In Workbench, you can install packages as you see fit in the terminals and execute Python code in Jupyter Notebooks. It is also a lighter version of procuring a virtual machine, which we will be doing later in the semester by serving Large Language Models.



1 Setup - Google Cloud

If you do not have access to your \$50 Google Cloud credits, please refer to [Lab 1](#) from Week 1. By now for those of you who have enrolled in the class, you should have received an e-mail from me. If you have not, I will provide you with a URL you will need to access in order to request a Google Cloud coupon. You will be asked to provide your school email address and name (you

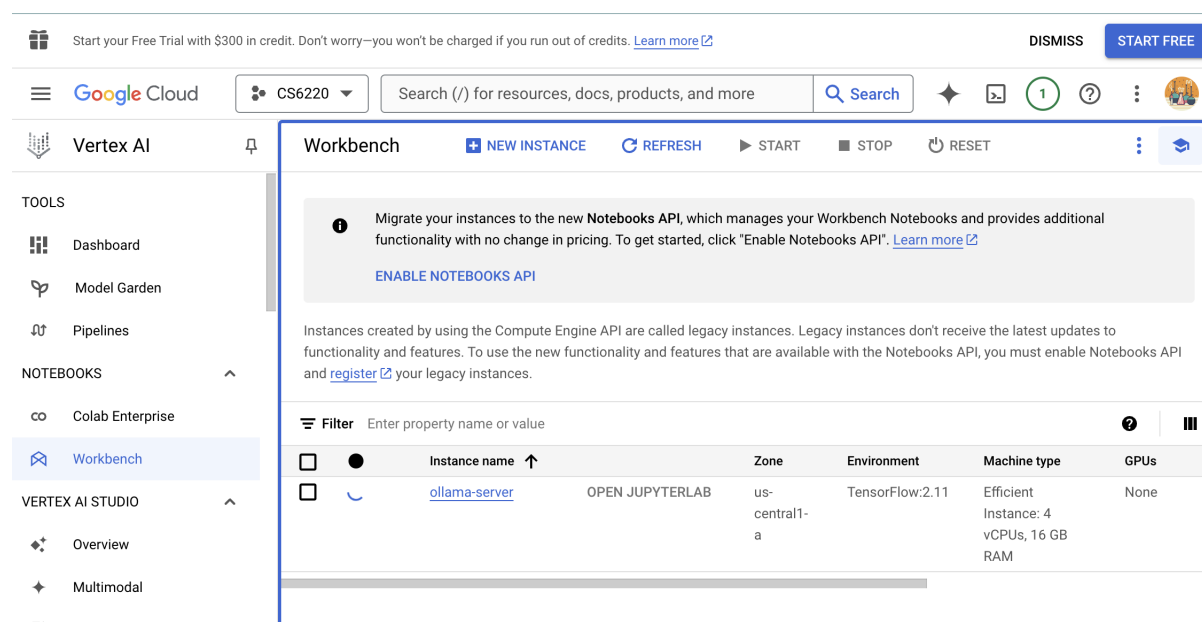
can use the domains: @northeastern.edu or @ccs.neu.edu). An email will be sent to you to confirm these details before a coupon is sent to you. You can redeem this coupon until May 1, 2025 and use it until January 1, 2026. **Go ahead and claim your credit.**

2 Creating a JupyterLab Instance

In your new project, we'll now start up Jupyterlab, install some libraries, and then do some processing. The first step is to create your own Workbench instance by clicking on **New Instance**, which will subsequently allow us to open JupyterLab. When you click **New Instance**, you will be presented with a multitude of options. We won't worry too much about that now; the defaults are OK for now. However, please do change the name of your instance so that you can find it later.

When we start coding with LLMs, you will notice that they will run painfully slow without GPU's. GCP automatically sets a quota for your GPU use to safeguard against cost overruns and resourcing issues. In our case, we would explicitly like to use GPU's, and may need to increase our quota. To do so, in your left panel, select IAM & Admin and from there, Quotas & System Limits to appropriately set GPU's.

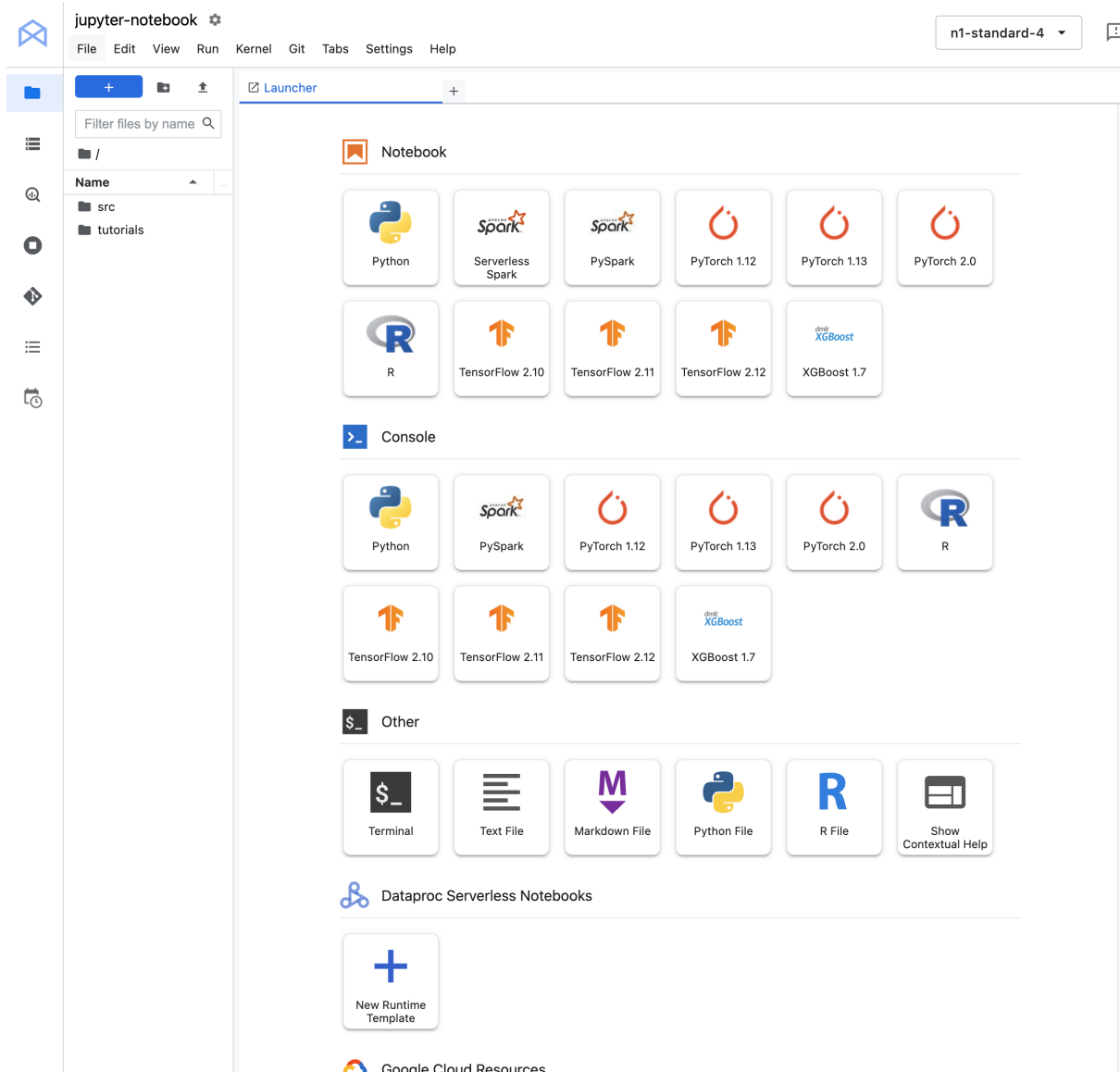
Go ahead and select a machine with a T4 (or any available) GPU to attach to your machine. When you've selected the appropriate options for you, **go ahead and submit** to create your instance. Depending on the GPU resources (and its expense), you may have to try several times before successful obtaining a Jupyter VM. After you've submitted, this may take a little while.



The screenshot shows the Google Cloud Workbench console. On the left, there's a sidebar with navigation options: Vertex AI, TOOLS (Dashboard, Model Garden, Pipelines), NOTEBOOKS (Colab Enterprise, Workbench), and VERTEX AI STUDIO (Overview, Multimodal). The 'Workbench' option is selected. The main panel shows a 'Workbench' header with buttons for 'NEW INSTANCE', 'REFRESH', 'START', 'STOP', and 'RESET'. Below this is a notification about migrating to the new Notebooks API. A table lists the instances:

Instance name	Zone	Environment	Machine type	GPUs
ollama-server	us-central1-a	TensorFlow:2.11	Efficient Instance: 4 vCPUs, 16 GB RAM	None

When you've started your Jupyter GCP instance, you can access it via clicking on the instance. Click on **Open JupyterLab**, and you'll have access to your instance's dashboard, where there are multiple applications, including a file navigator (and downloader) on the side panel. Note that JupyterLab will stop after a few hours of inactivity to avoid unnecessary charges, but you should generally practice good hygiene and shut down whenever you're not using the resource.



The terminal gives you access to the underlying VM, which can also be used to install packages and manipulate your hardware (e.g., NVIDIA GPU's). You can type in commands to see what files you have access to. In particular, we would like to check the GPU status. We can do so with the `nvidia-smi` command, which should give you the following screen:

```
(base) jupyter@jupyter-gpu:~$ nvidia-smi
Mon Dec 30 23:04:22 2024
```

NVIDIA-SMI 550.90.07		Driver Version: 550.90.07		CUDA Version: 12.4	
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	On	00000000:00:04.0	Off	0
N/A	44C	P8	9W / 70W	1MiB / 15360MiB	0% Default
					N/A

```

+-----+
| Processes:

```

GPU	GI	CI	PID	Type	Process name	GPU Memory
	ID	ID				Usage
No running processes found						

```
(base) jupyter@jupyter-gpu:~$
```

If you see the above, congratulations! You have managed to secure a JupyterLab Launcher with a GPU. Sometimes they are short in supply.

3 Running a Jupyter Notebook with GPUs

We will now try out our notebooks in JupyterLabs. Go ahead and open a Tensorflow 2-11 Notebook and enter the below into the first cell and run it.

```
import tensorflow as tf

print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

If the above shows that it recognizes a GPU, we know that the notebook kernel has access to the appropriate hardware. Now we must ensure that the code uses it appropriately. For this, there are several neural networks at keras.io. We're going to start with a miniature Transformer Network: the FNet Model. Let's download the [FNet Notebook Training Model](#) in a terminal with "curl" or "wget" commands. There, we will have instructions to download the data in the terminal. Go ahead and run through the instructions in the lab.

4 What to Submit

Here are the artifacts to submit:

- Screenshot of the downloaded data in your terminal. (You can type `ls`) or a screenshot of it downloading in the terminal.
- A picture of the results of `nvidia-smi`, but making sure that **GPU usage is non-zero**. This is easiest done while the F-Net is *training*