Running Generative LLMs with RunPod | A Serverless Platform

GENERATIVE AI INTERMEDIATE LLMS MACHINE LEARNING PYTORCH

Introduction

Serverless emerges as a game-changing strategy in cloud computing. Allowing developers to concentrate entirely on creating their applications while leaving the underlying infrastructure to cloud providers to take care of. Generative AI Large Language Models have fueled the growth of Serverless GPUs as most developers cannot run them locally due to the high GPU VRAM used by these language models. RunPod is one such platform that's gaining popularity in remote GPU services. RunPod provides access to powerful GPUs for building and testing out applications with large language models by providing various computing services, such as GPU Instances, Serverless GPUs, and API Endpoints. Learn LLMs with RunPod for executing resource-intensive large language models because of affordable pricing and various GPU possibilities.

Learning Objectives

- Learning the concept of Serverless and why it's useful for developers working on LLMs
- Understanding the need for high GPU VRAMto run Large Language Models
- Creating GPU Instances in the Cloud to Run Language Models
- Learning how to allocate GPU VRAM based on the LLM size

This article was published as a part of the Data Science Blogathon.

Table of contents

- Introduction
- What is Serverless?
- About RunPod
- Setting Up RunPod Account
- Run LLMs with RunPod
- Conclusion
- Frequently Asked Questions

What is Serverless?

Serverless is a service/method in Cloud Platforms that allows you to have an infrastructure on demand to carry out our developments and deploy our applications. With serverless, one can solely concentrate on the

development of the application and leave it to the cloud provider to manage the underlying infrastructure. Many cloud platforms like AWS, Azure, GCP, and others provide these offerings.

In recent times, Serverless GPUs have been becoming popular. Serverless GPUs is about renting GPU compute power on the cloud, when you do not have enough memory. These services have been rising since the introduction of large language models. As large language models require huge GPU VRAM, these serverless platforms have been rising one after another, providing better GPU services than others, and one such service is RunPod.

About RunPod

RunPod is a Cloud Platform offering compute services like GPU instances, Serverless GPUs, and even AI endpoints, thus allowing Machine Learning AI developers to leverage large GPUs for building applications with large language models. The prices offered by RunPod for the GPU instances are way less than what the big cloud providers like GCP, Azure, and AWS provide. RunPod has got a wide range of GPUs from RTX 30 series to 40 series and even the Nvidia A series, which have VRAM greater than 40+GB, thus allowing us to run 13Billion and 60Bliion parameters to run easily on it.

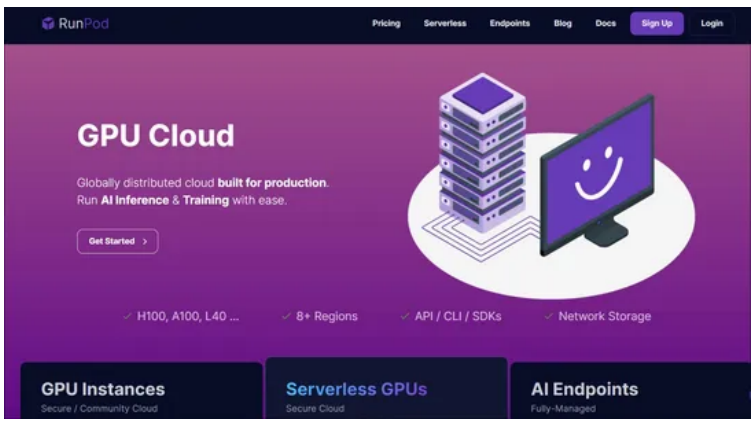
RunPod offers GPU services in two types:

- Community Cloud service when the GPUs you rent are the ones belonging to a single Individual and are incredibly cheaper.
- Secure Cloud service, where the GPUs we use belong to the RunPod themselves and are a bit more costly than the Community Cloud. Service Cloud is more suitable when we want to cluster huge amounts of GPUs to train very large language models.

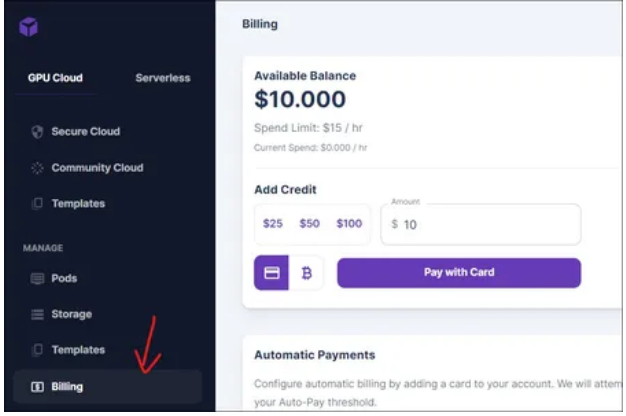
Also, RunPod provides both Spot and On-Demand instances. Spot Instances are the ones that can be interrupted any time while using and hence very cheap, whereas On-Demand instances are uninterruptable. In this article, we will go through RunPod and set a GPU instance to run a text generation web UI, where we will download a large language model from the hugging face and then chat with it

Setting Up RunPod Account

Firstly, we will begin with setting up a RunPod account, to do so click here, which will take you to the RunPod's home screen and you can see the pic below. Then we click on the signup button

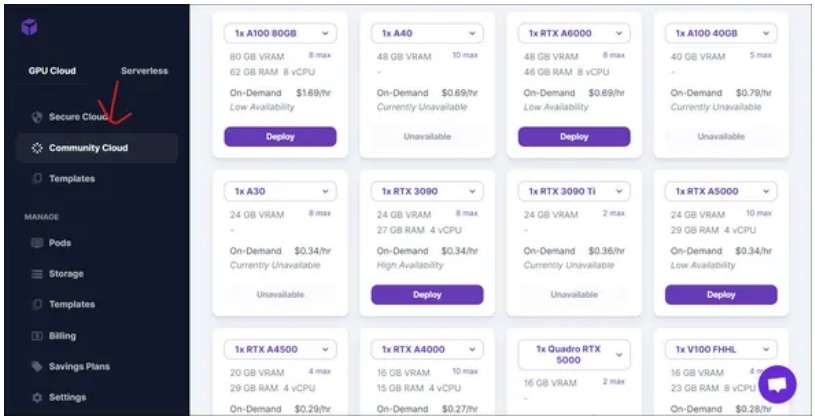


After signing up, we now need to add in credits to get started with using the Cloud GPU Instances. We can start with a minimum deposit of 10$ and can do it either through a debit card or credit card. To buy credits you need to click on the billing section on the left



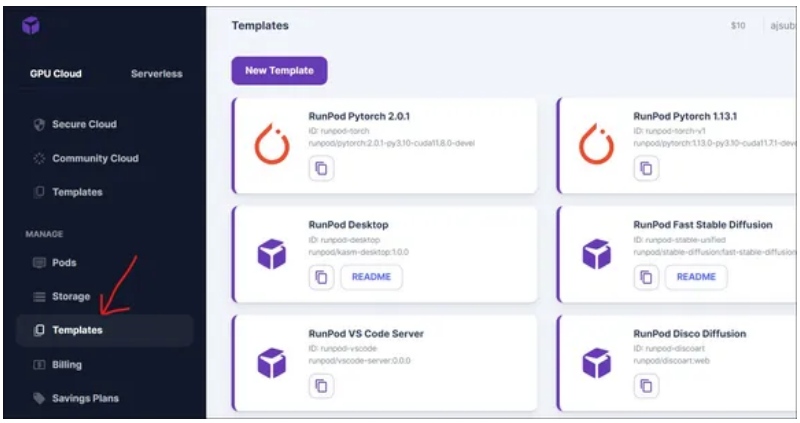
Here, I have bought $10, i.e., my available balance is $10. And this is only a one-time payment. I won't be charged anything after my $10 is exhausted. The Pods we create will automatically shut down when the available balance hits $0. RunPod has automatic payment options, but we will go through a one-time payment setup as we don't have to worry about money being deducted.

GPU Instances



Here, when we click on the Community Cloud on the left, we see that it lists all the available GPUs, their specifications, and how much they charge for them. The Security Cloud is also the same, but the only difference is the GPUs in the Security Cloud are maintained by the RunPod team, and the GPUs in the Community Cloud belong to the Community, i.e. individuals all over the world.

Templates

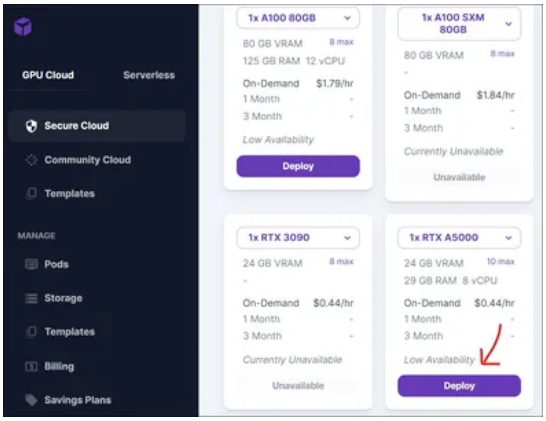


In the above pic, we see predefined templates available. We can run a GPU instance within minutes with these templates. Many templates, like the Stable Diffusion template, allow us to start a GPU instance with stable diffusion to generate images with it. The RunPod VS Code template allows us to write and utilize the GPU from the GPU Instance.

The PyTorch template of different versions, where a GPU instance comes ready with the latest PyTorch library, which we can use to build Machine Learning models. We can also create our custom templates, which we can even share with others so they can spin up a GPU instance with the same template

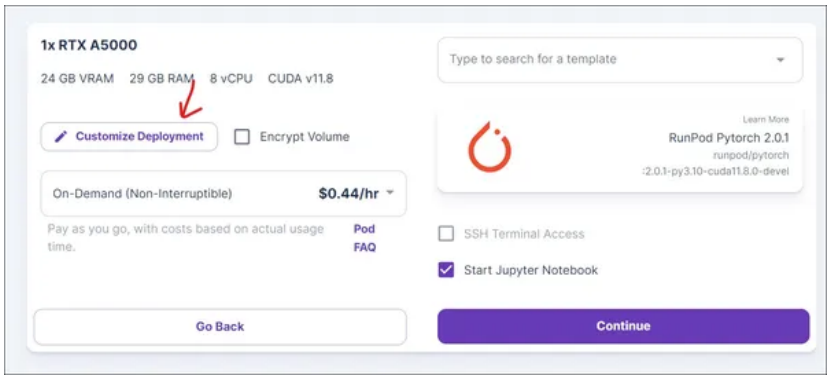
Run LLMs with RunPod

This section will spin up a GPU instance and install the Oobabooga text-generation-web-ui. This can be used to download any model available from the hugging face, whether in the original float16 version or the quantized form. For this, we will select the Nvidia A5000 GPU instance containing 24GB of VRAM, which might be sufficient for our application. So, I select the A5000 and click on Deploy.

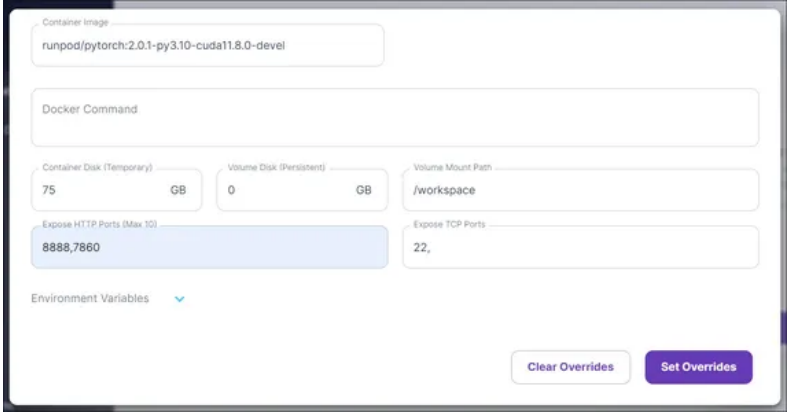


PyTorch Template

Then, as Large Language Models require Pytorch to run, we have chosen the PyTorch template. When we create an Instance from this template, the template will come loaded with the PyTorch libraries. But for this instance, we will be making some changes. So, we click on the custom deployment.

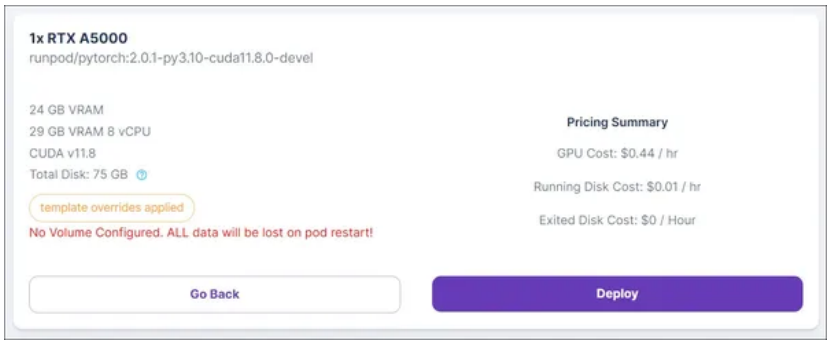


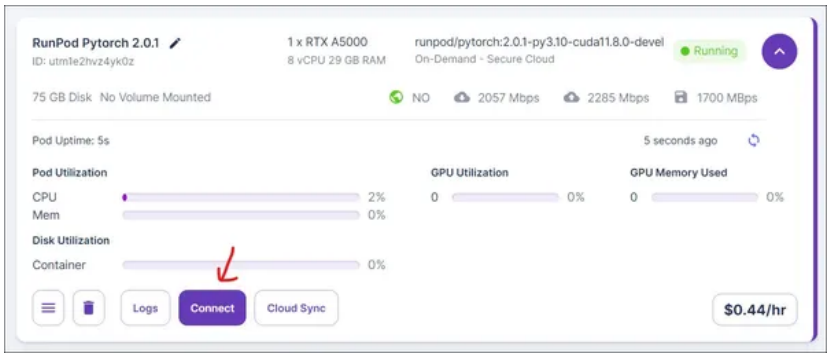
Here, we will be assigning Container Disk to 75GB, so in case we download a big large language model, it will fit in. And in this case, I do not want to store any data for later. So, Volume Disk to zero. When this is set to zero, we will lose all the information when the GPU instance is deleted and for this example case, I'm fine with it. And the application that we run will need access to port 7860. Hence, we expose the 7860 Port. And finally, we click on override.



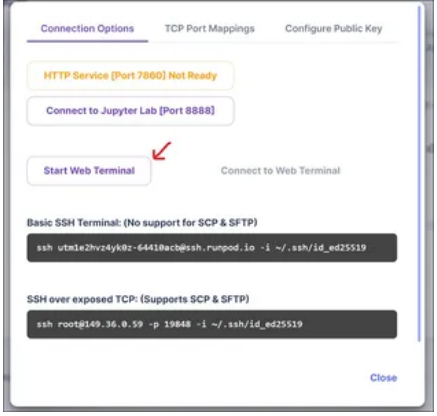
Override

After clicking on the override, we can see the estimated per-hour cost for the GPU instance in the below image. So a 24GB VRAM GPU along with 29GB RAM and 8vCPU will cost around $0.45 per hour, which is very cheap to what many large Cloud Providers provide. Now, we click on the deploy button.





After clicking on deploy above, an Instance will be created within a few seconds. Now we can connect to this GPU instance through SSH via the **Connect** Button shown in the above pic. After clicking on the **Connect** button, a pop-up will appear, where we click on the **Start Web Terminal** and then **Connect to the Web Terminal,** as shown in the below pic, to access our GPU Instance.



Now, a new tab in the web browser will appear, which we can access. Now, in the web terminal, type the below commands to download text-generation-web-ui, allowing us to download any large language model from HuggingFace and use it for inference.

```
git clone https://github.com/oobabooga/text-generation-webui cd text-generation-webui pip install -r requirements.txt
```

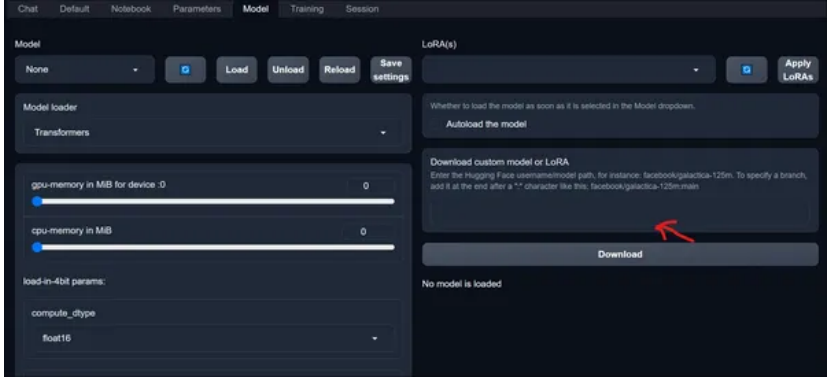
Text Generation Webui

Now, the first command will pull the text-generation-webui GitHub repository that contains the Python code to use large language models locally. The next two lines will go into the directory and install all the necessary libraries for running the Python program. To start the web-ui, we use the below code

```
python server.py --share
```

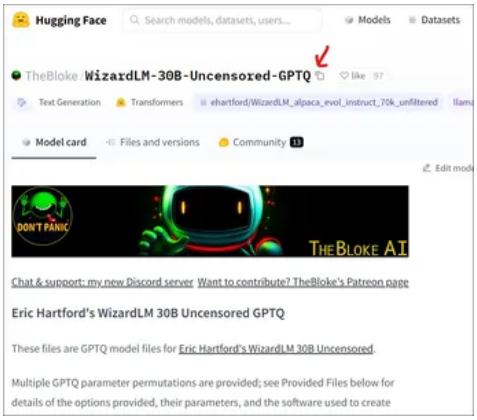
The above command will start the web-ui. This will start web-ui in localhost. But as we run the application in the remote GPU instance, we need to use a Public URL to access the website. The –share option will create a Public URL, which we can click on to access the text-generation-web-ui.

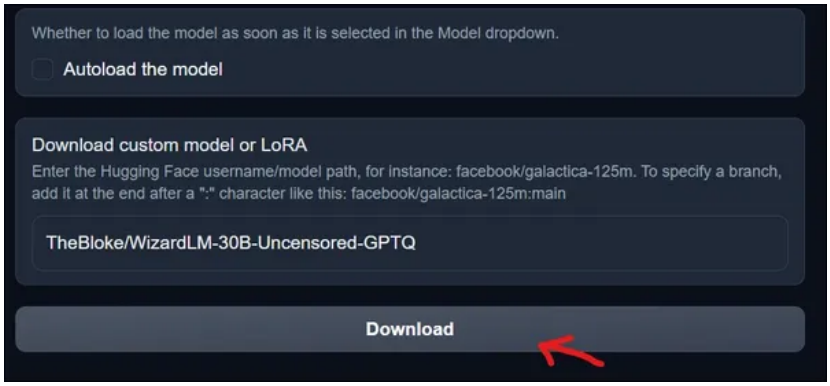
Click on the gradio.live link, as shown in the above Image, to access the UI. In that UI, go to the Model section in the top menu. Here in the below pic, we see towards the right; we need to provide a link for the model that we want to use.



WizardLM 30B

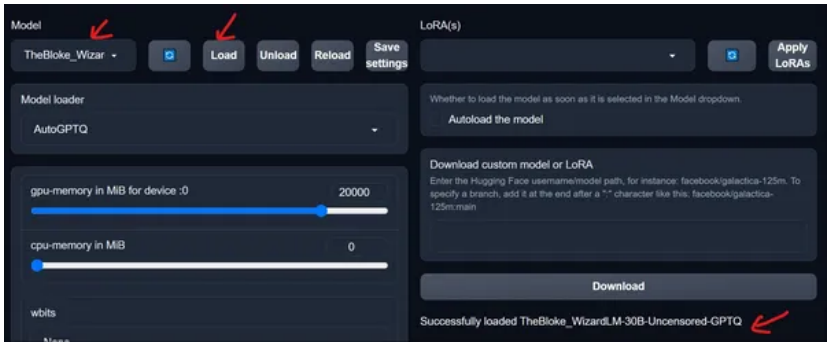
For this, let's go to Hugging Face to a model named WizardLM 30B, a 30Billion Parameter model. We will click on the copy button to copy the link to this model and then paste it into the UI, and then click on the download button to download the model.





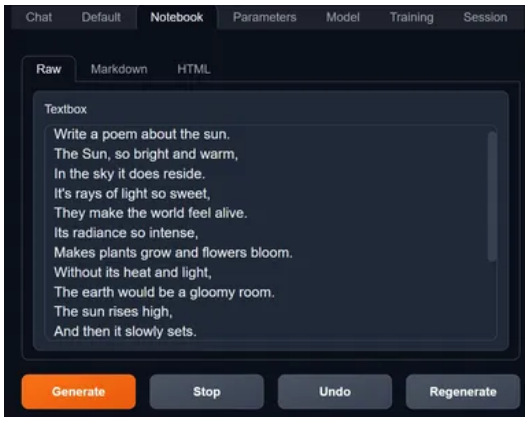
Select UI

After the large language model is downloaded, we can select it from the left part of the UI under the Model. Click the refresh button next to it if you cannot find the downloaded model. Now select the model that we have just downloaded. The model we have downloaded is a 16GB model. So allocate around 20GB of GPU VRAM to it to run the model completely on GPU. Then click on the load button. This will load the model to the GPU, and you can see a success message towards the right part of the UI.

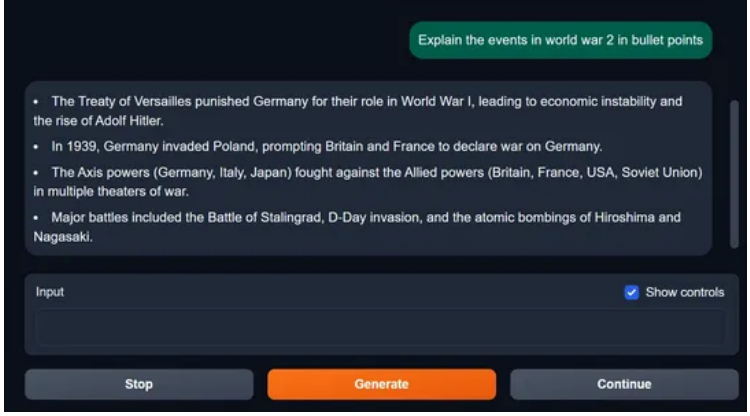


Write a Poem

Now, the large language model is loaded into the GPU, and we can infer it. Go to the Notebook Section of the UI by clicking on the Notebook in the top menu. Here, I test the model by asking it to run a poem on the sun by saying “Write a poem about the Sun” and then click on the Generate button. The following is generated:



The above pic shows that the model has generated a poem based on our query. The best part here is the poem is related to the Sun. Most large language models try to drift apart from the initial query, but here, our WizardLM large language model maintains the query relation until the end. Instead of just text generation, we can also chat with the model. For this, we go to the Chat Section by clicking Chat Present on top of the UI. Here, let’s ask the model some questions.



Here, we asked the model to give information about World War 2 in bullet points. The model was successful in replying with a chat message that was relevant to the query. The model also presented the information in bullet points, as requested in the query chat message. So this way, we can download any open-source large language model and use it through the UI in this GPU instance we have just created.

Conclusion

In this article, we have looked into a Cloud Platform named RunPod that provides GPU Serverless Services. Step by step, we have seen how to create an account with RunPod and then how to create a GPU Instance within it. Finally, in the GPU Instance, we have seen the process of running a text-generation-ui that lets us download open source Generative AI large language model and infer the model.

Key Takeaways

Some of the key takeaways from this article include:

- RunPod is a cloud platform offering GPU services.
- RunPod offers its services in two ways. One is the Community Cloud services, where the GPUs we rent are from an Individual out there and are cheap and the other is the Secure Cloud service, where any GPU Instance we create belongs to the RunPod GPUs.
- RunPod comes with templates containing some boilerplate code we can build, i.e., the GPU instances we create with these templates will come ready with them(libraries/software) installed.
- RunPod offers both automatic and one-time payment services.

Frequently Asked Questions

Q1. What is Serverless?

A. Serverless is an offering provided by Cloud Platforms, where the Cloud Provider maintains the infrastructure, and all we need is to focus on our code and not worry about taking care of the underlying infrastructure.

Q2. What are Serverless GPU/ GPU Instances?

A. These are GPU services provided by Cloud Platforms, where the Cloud Platforms offer you GPU services and charge per hour. The price depends on the type of GPU and the memory used.

Q3. What is RunPod?

A. RunPod is a cloud platform that primarily focuses on GPU services. The services include the provision of GPU Instances, Serverless GPUs, and API Endpoint services. RunPod charges these GPU instances on a per-hour basis. Anyone with an account with RunPod can spin up a GPU Instance within seconds and run applications that use GPUs extensively.

Q4. What type of GPUs are offered by RunPod?

A. A wide range of GPUs with wide memory ranges are offered by the RunPod platform. These include GPUs from consumer-grade to industry-grade GPUs. The memory ranges from 8GB to all the way up to 80GB VRAM These GPUs can be stacked together, and the utmost 8 GPUs can be stacked together depending on the availability of the GPUs.

Q5. What are Spot GPU Instances?

A. Spot GPU Instances are the ones that can be interrupted anytime without notice. If you create a Spot GPU Instance, it is not guaranteed when it will shut down. It can shut down at any time. The Spot GPU Instances are generally cheaper than the On-Demand GPU Instances, where the Instance doesn't shut down and will stay until you stop it or delete it.

Article Url - https://www.analyticsvidhya.com/blog/2023/09/generative-llms-with-runpod/

Ajay Kumar Reddy