Running NVIDIA Parabricks on GCP

Table of contents

What is NVIDIA Parabricks?

Starting a Compute Instance

Installing Parabricks

Testing Parabricks

Closing Remarks

List of Figures

Figure 0. Image Vm Instances

Figure 1. Image Instance Settings

Figure 2. Image Machine Configuration

Figure 3. Image Boot Disk

Figure 4. Image Create

Figure 5. Image Ssh

Figure 6. Image Install Drivers

Figure 7. Image Docker Ps

Figure 8. Image Ngc1

Figure 9. Image Docker Pull1

Figure 10. Image Tree

Figure 11. Image Startup

Figure 12. Image File List

This guide shows how to run Parabricks on a compute instance on Google Cloud Platform (GCP).

What is NVIDIA Parabricks?

Parabricks is an accelerated compute framework that supports applications across the genomics industry, primarily supporting analytical workflows for DNA, RNA, and somatic mutation detection applications. With industry leading compute times, Parabricks rapidly converts a FASTQ file to a VCF using multiple, industry validated variant callers and also includes the ability to QC and annotate those variants. As Parabricks is based upon publicly available tools, results are easy to verify and combine with other publicly available data sets.

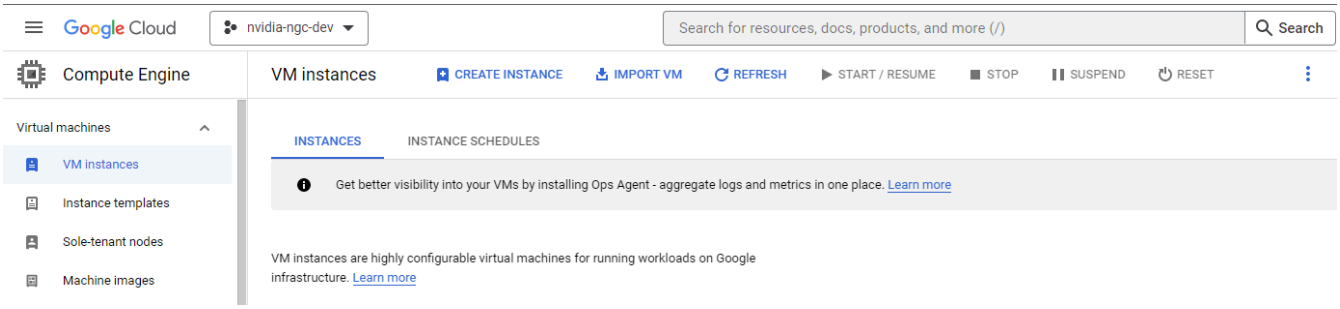
More information is available on the Parabricks Product Page.

Detailed installation, usage, and tuning information is available in the Parabricks user guide.

Starting a Compute Instance

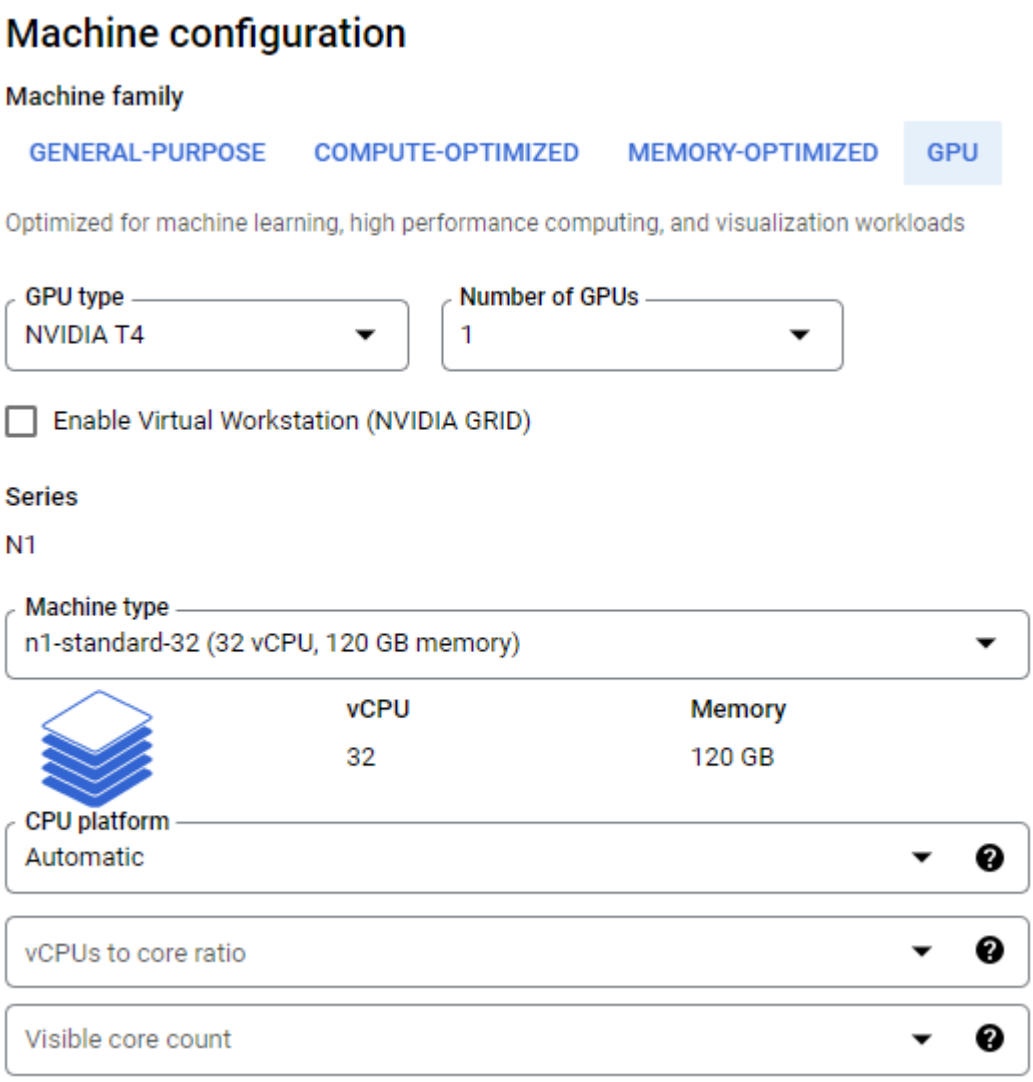
In this section, we will show how to start a Compute Instance on GCP.

Begin by navigating to the Google Cloud homepage and selecting Compute Engine from the left sidebar. This will take us to the VM instances page.

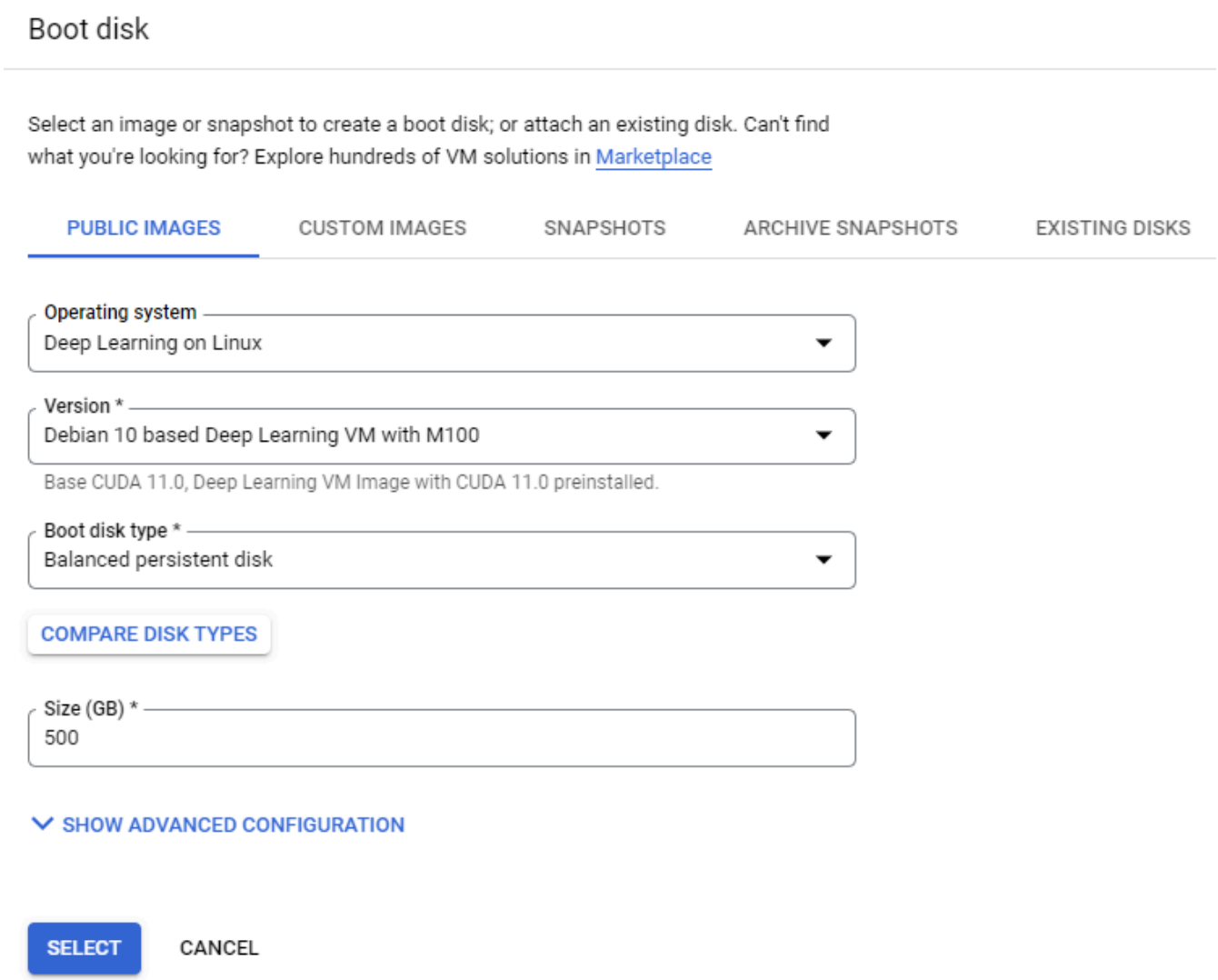


At the top of the page, select Create Instance. Here we can configure all the settings for our VM instance. Under Name let's call the instance "parabricks" and select an appropriate region. For the purpose of this guide, the region can be anything.

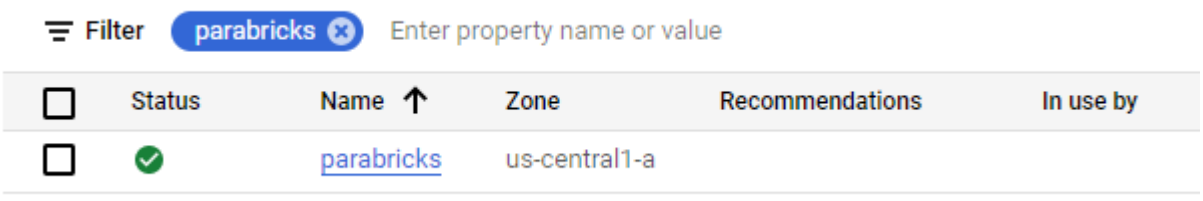
Under Machine Configuration we will select hardware details for the VM instance. Select GPU at the top and select "NVIDIA T4" for GPU Type and "1" for Number of GPUs. Under Machine Type, select "n1-standard-32". This machine type meets the minimum CPU and memory requirements for Parabricks, and is all we need for the purpose of this guide. The Machine Configuration section should now look like this:



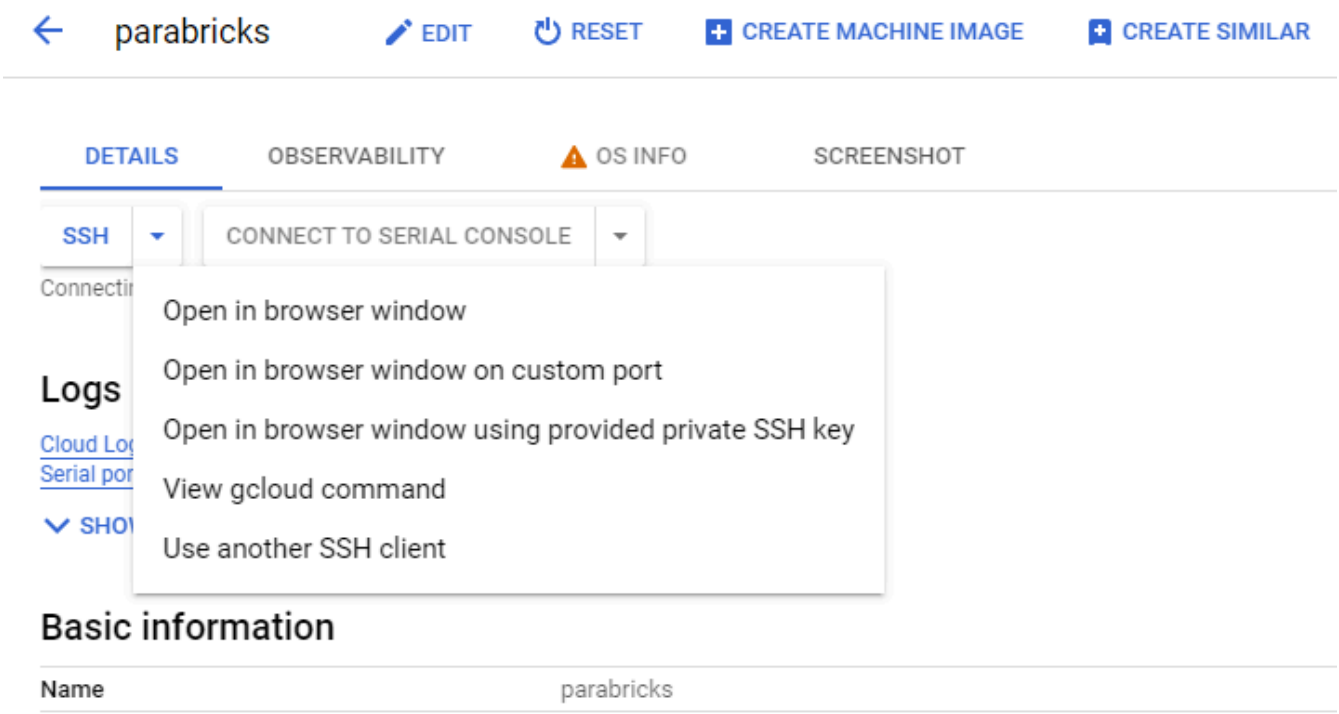
We will make sure our VM instance has the proper GPU drivers by switching our base image from the default image to a base image that already has drivers installed. Under the Boot disk section, select Change. Under Operating System, select “Deep Learning on Linux” and under Version select “NVIDIA GPU-Optimized VMI”. While we are on this page, let’s also increase the disk size from the default value up to 500 GB. This will ensure we have enough space for the test dataset when we test our Parabricks installation. The Boot disk page will look like this:



Now we have everything we need to launch the instance. At the bottom of the page click Create. After a few minutes, we can see the instance is running and ready to be used.



Let's click on the instance to go to the instance page and then under SSH in the top right, select a method to connect. Read more about connecting to GCP instances in their documentation.



Once we are connected, the VM will ask if we want to install the NVIDIA Driver. Select yes, and allow it to install the drivers automatically.

```
Warning: Permanently added 'compute.4604189263571679143' (ECDSA) to the list of known hosts.
=====================================
Welcome to the Google Deep Learning VM
=====================================

Version: common-cu110.m100
Based on: Debian GNU/Linux 10 (buster) (GNU/Linux 4.19.0-22-cloud-amd64 x86_64\n)

Resources:
 * Google Deep Learning Platform StackOverflow: https://stackoverflow.com/questions/tagged/google-dl-platform
 * Google Cloud Documentation: https://cloud.google.com/deep-learning-vm
 * Google Group: https://groups.google.com/forum/#!forum/google-dl-platform

To reinstall Nvidia driver (if needed) run:
sudo /opt/deeplearning/install-driver.sh
Linux parabricks 4.19.0-22-cloud-amd64 #1 SMP Debian 4.19.260-1 (2022-09-29) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Creating directory '/home/gburnett_nvidia_com'.

This VM requires Nvidia drivers to function correctly.   Installation takes ~1 minute.
Would you like to install the Nvidia driver? [y/n] y
```

Once the driver installation finishes, there we need to set up our Docker environment. At this point, Docker is already installed however it requires sudo access to run. We can get around this by running the following commands:

The first command adds our user to the Docker group, allowing us to run Docker commands without using sudo. The second command refreshes Docker to make sure these changes take effect. We can test that this worked by running *docker ps*. This command should run without any errors:

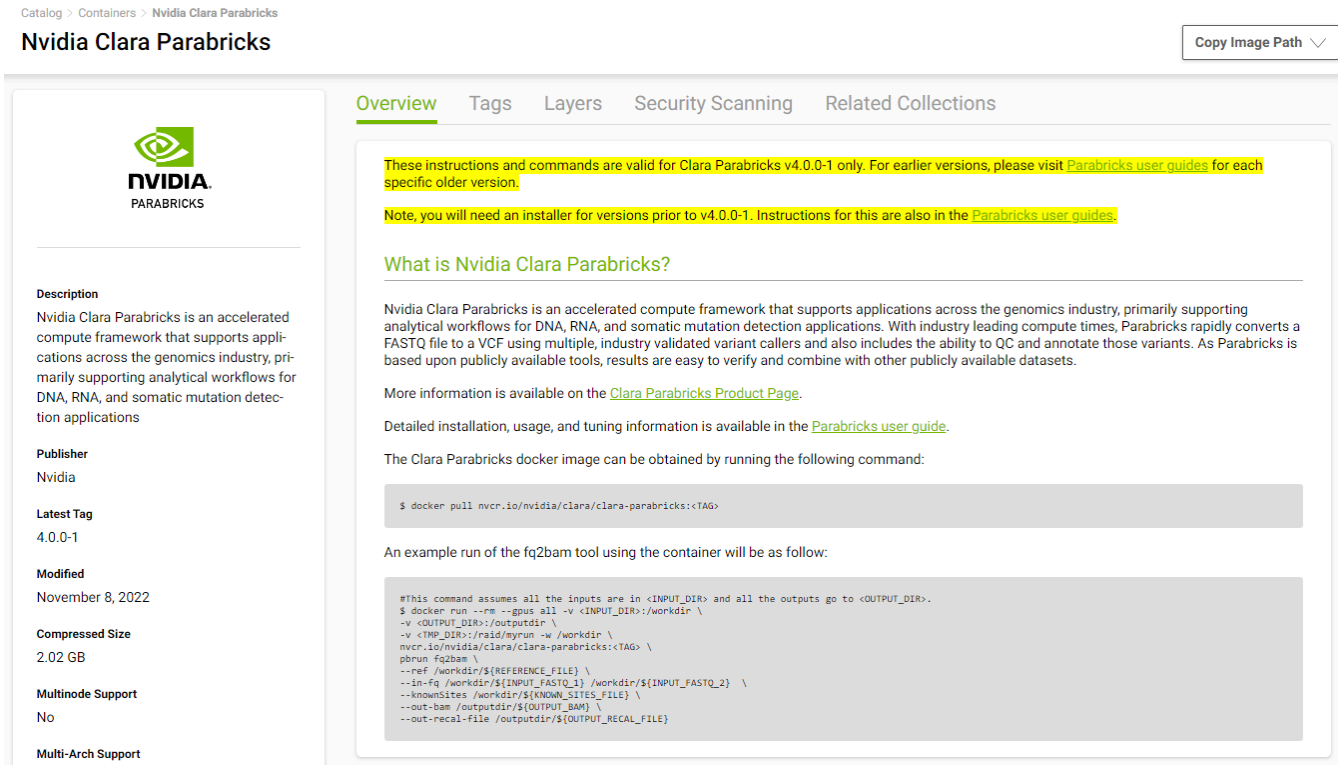
```
$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
$
```

Now we are ready to start the Parabricks installation!

Installing Parabricks

We will install Parabricks into our instance that we just created. To do this, we will use the NVIDIA GPU Cloud (NGC) to download the Parabricks Docker image.

Visit the Parabricks page on NGC to get the Docker pull command for the latest version of Parabricks.



Back in our EC2 instance, let's run the docker pull command:

```
$ docker pull nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1
```

```
gburnett_nvidia_com@parabricks:~$ docker pull nvcr.io/nvidia/clara/clara-parabricks:4.0.0-1
4.0.0-1: Pulling from nvidia/clara/clara-parabricks
d7bfe07ed847: Pull complete
bbbbd451a669: Pull complete
773163705c35: Pull complete
d6949fcf1aef: Pull complete
3eb73064088b: Pull complete
a3ac3ab0ee35: Pull complete
8d88682a5e1d: Pull complete
Digest: sha256:0170beef24131a23bb63bc36ec059e493df1f04a4a78f9d2c5df9bce1d5d9a35
Status: Downloaded newer image for nvcr.io/nvidia/clara/clara-parabricks:4.0.0-1
nvcr.io/nvidia/clara/clara-parabricks:4.0.0-1
```

Now Parabricks is installed! Let's run some sample data to test it.

Testing Parabricks

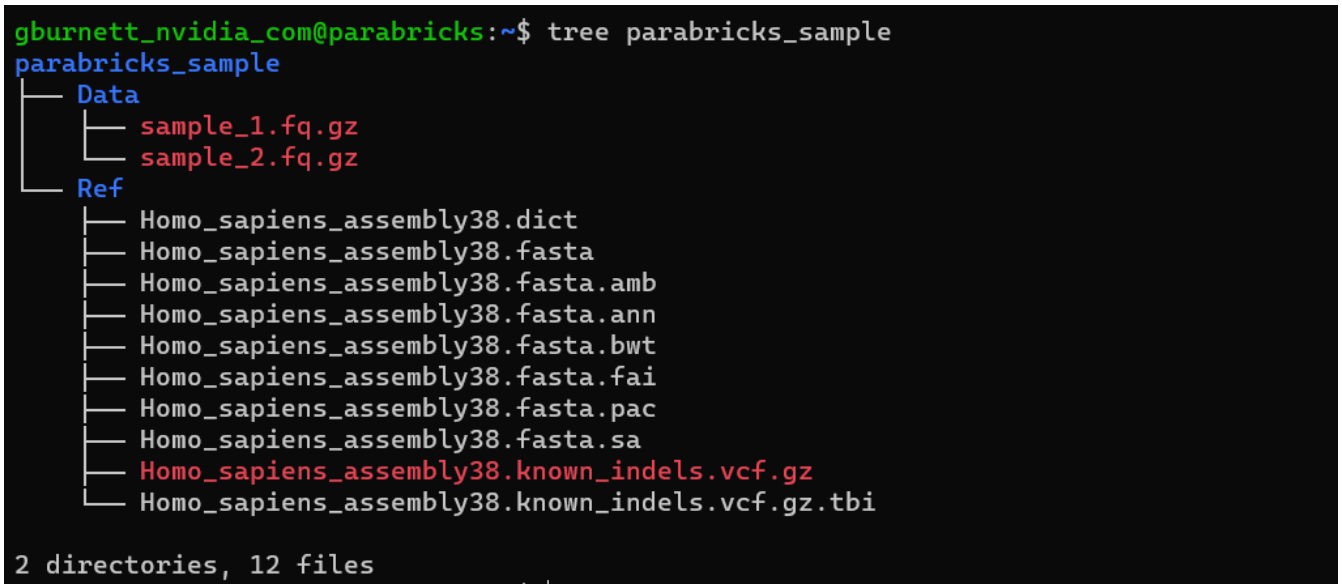
Parabricks provides a small sample dataset as a test for the installation and hardware which can be downloaded using:

```
$ wget -O parabricks_sample.tar.gz \
"https://s3.amazonaws.com/parabricks.sample/parabricks_sample.tar.gz"
```

When the download completes, we can untar the data using:

```
$ tar xzvf parabricks_sample.tar.gz
```

The parabricks_sample folder should look like this when we're done:



Finally, we can run any of the Parabricks pipelines on it. Let's run the germline pipeline using the following command:

```
$ docker run \ --rm \ --gpus all \ --volume `pwd`:`pwd` \ --workdir
`pwd`/parabricks_sample \ nvcr.io/nvidia/clara/clara-parabricks:4.3.1-1 \ pbrun
germline \ --ref Ref/Homo_sapiens_assembly38.fasta \ --in-fq Data/sample_1.fq.gz
Data/sample_2.fq.gz \ --knownSites
Ref/Homo_sapiens_assembly38.known_indels.vcf.gz.tbi \ --out-bam output.bam \ --
out-variants germline.vcf \ --out-recal-file recal.txt
```

We can tell that Parabricks started correctly when we see the Parabricks banner and the ProgressMeter begins to populate with values:

```
Please visit https://docs.nvidia.com/clara/#parabricks for detailed documentation


[Parabricks Options Mesg]: Automatically generating ID prefix
[Parabricks Options Mesg]: Read group created for /home/gburnett_nvidia_com/parabricks_sample/Data/sample_1.fq.gz and
/home/gburnett_nvidia_com/parabricks_sample/Data/sample_2.fq.gz
[Parabricks Options Mesg]: @RG\tID:HK3TJBCX2.1\tLB:lib1\tPL:bar\tSM:sample\tPU:HK3TJBCX2.1

[Parabricks Options Mesg]: Checking argument compatibility
[Parabricks Options Mesg]: Read group created for /home/gburnett_nvidia_com/parabricks_sample/Data/sample_1.fq.gz and
/home/gburnett_nvidia_com/parabricks_sample/Data/sample_2.fq.gz
[Parabricks Options Mesg]: @RG\tID:HK3TJBCX2.1\tLB:lib1\tPL:bar\tSM:sample\tPU:HK3TJBCX2.1
[PB Info 2022-Nov-18 00:54:13] ------------------------------------------------------------------------------
[PB Info 2022-Nov-18 00:54:13] ||                Parabricks accelerated Genomics Pipeline                ||
[PB Info 2022-Nov-18 00:54:13] ||                           Version 4.0.0-1                              ||
[PB Info 2022-Nov-18 00:54:13] ||                      GPU-BWA mem, Sorting Phase-I                      ||
[PB Info 2022-Nov-18 00:54:13] ------------------------------------------------------------------------------
[M::bwa_idx_load_from_disk] read 0 ALT contigs
[PB Info 2022-Nov-18 00:54:18] GPU-BWA mem
[PB Info 2022-Nov-18 00:54:18] ProgressMeter    Reads           Base Pairs Aligned
[PB Info 2022-Nov-18 00:54:51] 5043564          580000000
[PB Info 2022-Nov-18 00:55:20] 10087128 1160000000
[PB Info 2022-Nov-18 00:55:49] 15130692 1740000000
[PB Info 2022-Nov-18 00:56:19] 20174256 2320000000
[PB Info 2022-Nov-18 00:56:48] 25217820 2900000000
```

This should take ~10 minutes to finish running. When it's done, we should see the output files in the sample data directory:

```
gburnett_nvidia_com@parabricks:~$ ls output/
output.bam  output.bam.bai  output_chrs.txt  output.vcf
```

Congratulations, we've just run our first Parabricks job!

Closing Remarks

We encourage you to expand on the demo in this guide by using your own data, trying other pipelines, and generally exploring what Parabricks has to offer. Check out the documentation for more information about the different pipelines available. You can also find our online developer community on the Parabricks forum, where you can ask questions and search through answers while you are learning how to use Parabricks.