Fall 2024- I535
Mgmt. Access Use Big Data

Vaishnavi Vishwas Pawar
vpawar@iu.edu

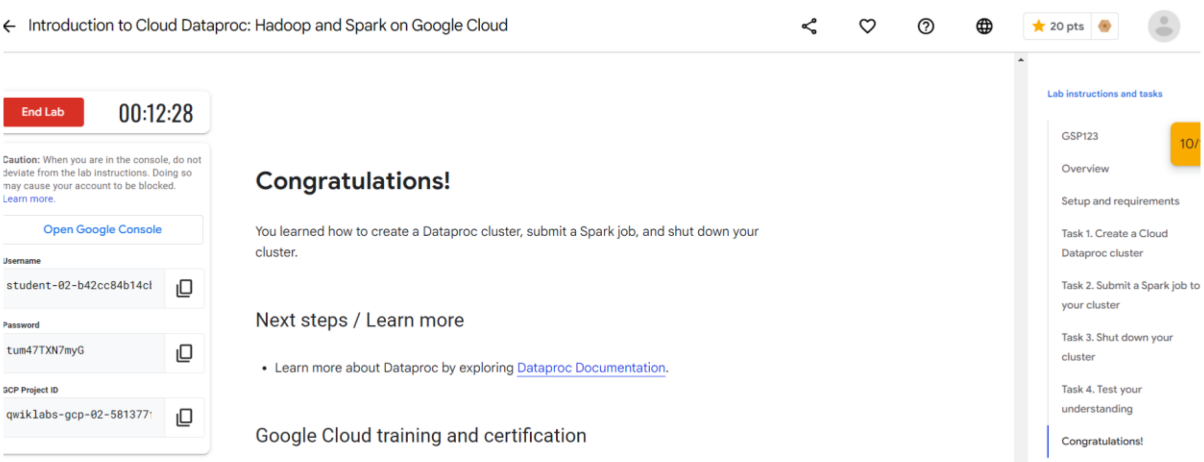
Distributed Computing with Dataproc Assignment

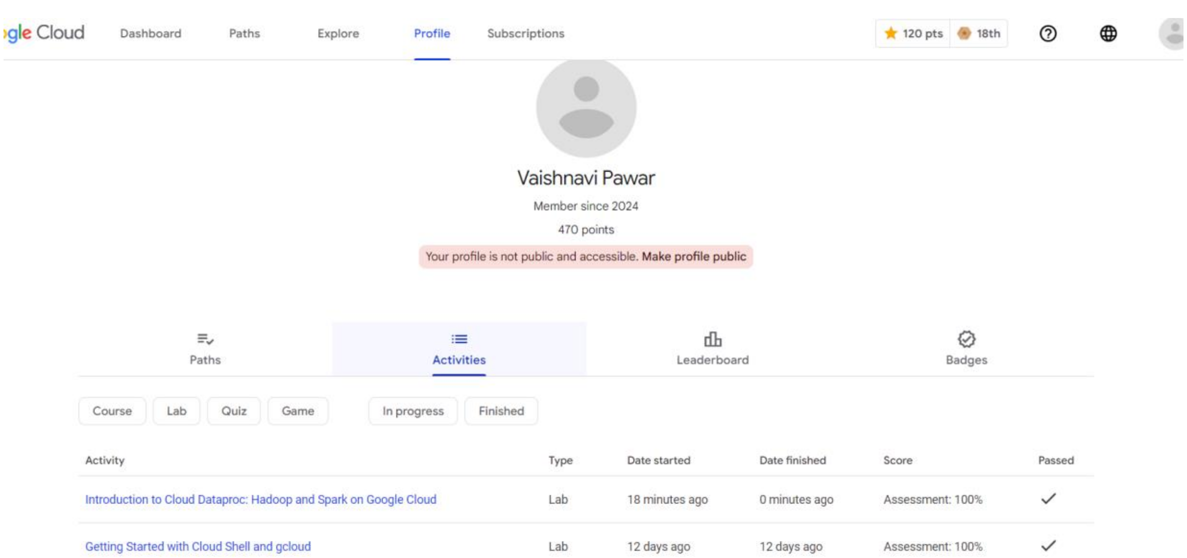
As part of my Distributed Computing with Dataproc assignment, I completed two hands-on labs to gain practical experience with Google Cloud services. I worked on the following labs:

- Introduction to Cloud Dataproc: Hadoop and Spark on Google Cloud
- Dataproc: Qwik Start Console.

These labs allowed me to dive into Google Cloud's Dataproc platform and learn how to set up and manage clusters while performing distributed computing tasks. I investigated the platform's capabilities and learned how its services and tools perform in real-world scenarios.

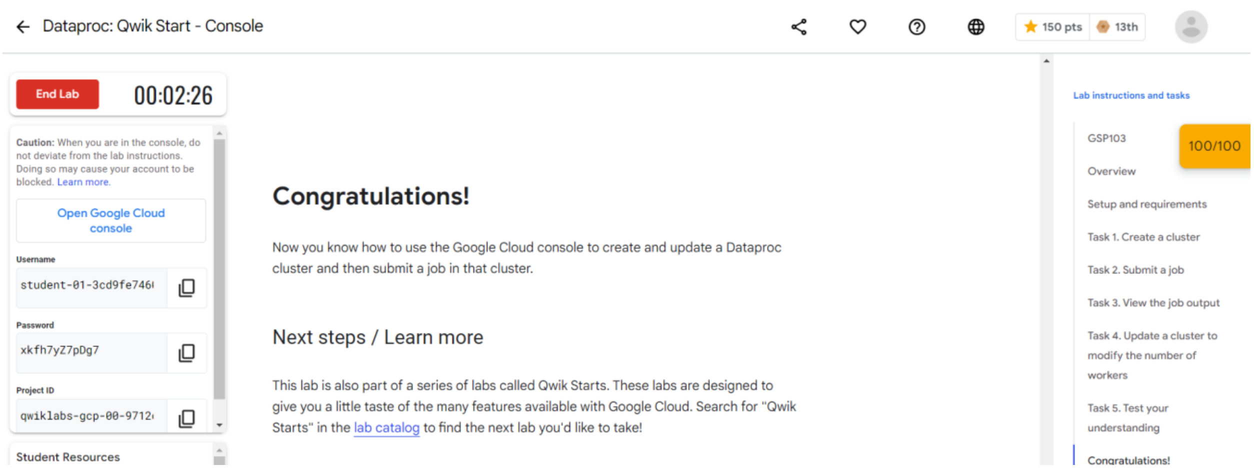
Introduction to Cloud Dataproc: Hadoop and Spark on Google Cloud

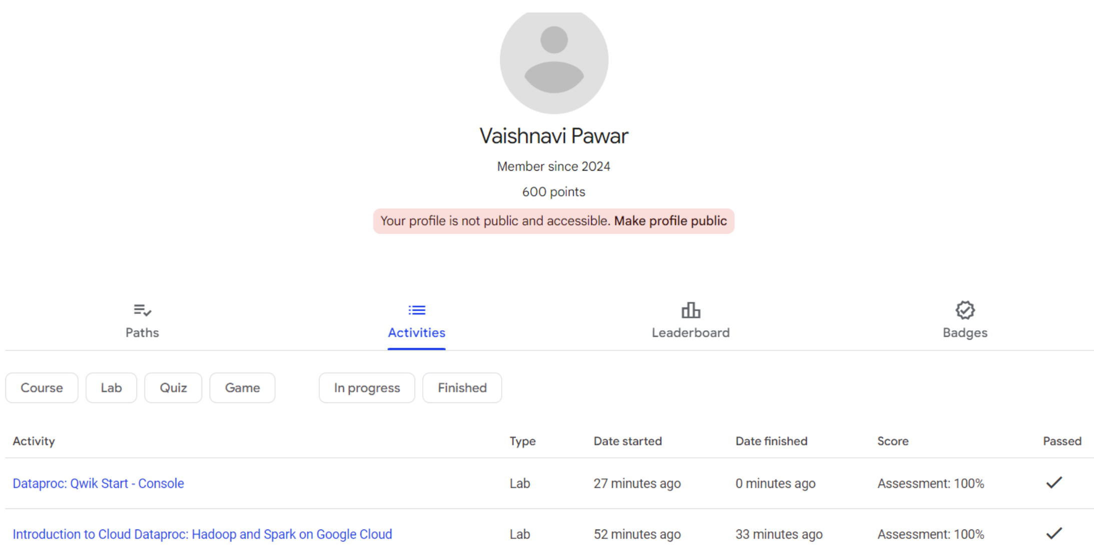




Fall 2024- I535 Vaishnavi Vishwas Pawar
Mgmt. Access Use Big Data vpawar@iu.edu

Dataproc: Qwik Start - Console





Fall 2024- I535 Vaishnavi Vishwas Pawar
Mgmt. Access Use Big Data vpawar@iu.edu

Report:

My Understanding of Cloud Dataproc Capabilities

According to what I've learned, Cloud Dataproc is essentially a tool for simplifying Hadoop and Spark cluster management. Dataproc takes care of everything for you, so you don't have to worry about manually configuring each node or balancing workloads. It's like having an automated assistant who not only configures your cluster but also ensures that it's compatible with other GCP services such as Google Cloud Storage and BigQuery.

The coolest part for me is how adaptable Dataproc is; you can scale up or down depending on your job requirements. This means you're not committing resources (or costs) that you don't need. It's also very convenient that everything is integrated into the GCP ecosystem, making it easier to manage data flows without switching between platforms.

What I Learned in Each Lab

<u>Introduction to Cloud Data Dataproc: Hadoop and Spark on Google Cloud.</u>

This lab gave me firsthand experience with creating and configuring a Dataproc cluster. What caught my interest the most was how easy it was to run jobs using both Hadoop and Spark without getting bogged down in configuration details. I saw firsthand how Spark jobs can be distributed across multiple nodes, which was eye-opening when considering how large datasets are handled.

Main Takeaways:

- I learned how to configure a Dataproc cluster and select the appropriate machine types and worker nodes for each task.
- I ran a sample Spark job, and it was fascinating to see how the cluster handled it.
- The integration with Google Cloud Storage made it simple to pull in data, process it, and save the results.

<u>Dataproc: Qwik Start - Console.</u>

This lab focused more on using the GCP console, which I appreciated because I didn't have to use the command line to complete tasks. Navigating the console was simple, and I was able to quickly launch a Dataproc cluster and manage jobs directly from the website. I spent a lot of time looking through the logs and tracking how jobs were progressing, which helped me understand how Dataproc manages jobs behind the scenes.

Vaishnavi Vishwas Pawar
vpawar@iu.edu

Main Takeaways:

- The GCP console is extremely user-friendly for managing Dataproc clusters.
- I learned how to launch jobs directly from the interface and monitor their status in real time.
- The lab demonstrated how easy and manageable logs and error tracking are with GCP's built-in tools.
- I also studied into what happens when you change the number of worker nodes in the cluster. Adding more nodes increased workload distribution efficiency, which improved job performance, particularly when dealing with large datasets.

How Changing the Number of Worker Nodes Affects Performance

My experience in these labs taught me that the number of worker nodes has a significant impact on performance. More worker nodes result in more parallel processing, which speeds up the job. It's pretty simple: if you have a large dataset or a complex Spark job, adding more worker nodes will speed up and smooth out the workload by distributing it across more resources.

However, there is a balance. While adding nodes boosts performance, it also raises costs, so it's critical to find the sweet spot. I discovered that Dataproc's autoscaling feature is a lifesaver here; it automatically adjusts the number of nodes based on the job's requirements, allowing you to avoid paying for resources that aren't required. In short, more nodes equal faster processing, but you must consider cost efficiency.