Big Data Technologies   
in Google Cloud Platform

# Accessing the Educational Credits

## Receive the voucher

To receive your educational credits please follow this link and use your e-uvt account to receive the voucher. The voucher will be received via this email address.

**Important!** The e-uvt account cannot be used to create billing accounts in Google Cloud.   
**You will need to use your personal account**. You don’t need to add your credit card.

## Apply the voucher

Log in using your **personal** Google account at <https://console.cloud.google.com/education>

Insert your name and voucher code. A billing account for education will be created automatically. Make sure you have this billing account selected.

## Create a Project

Use the side bar Menu (click on the 3 lines in the top left if not shown).

Go to Cloud Overview -> Dashboard

In the top bar you should see a projects button. Clicking on it will show the following interface:

A screenshot of a computer

Description automatically generated

Click on New Project give it a name.

# Google Compute Engine

Compute Engine offers Infrastructure as a Service and allows the creation of Virtual Machines over which we have full control. We can select the operating system, connect via SSH and modify the state of the virtual machine as we like.

## Task 1

Access the Compute Engine interface from the Menu. You will be asked to Enable this API. Wait for the API to get activated. Now you are able to create virtual machines, but this is not the scope of the lab.

# Google Cloud Storage

Cloud Storage is a service that allows for Object Storage. Objects are stored in buckets. You will pay for the amount of space you use.

## Task 2

Access the Storage interface and create a Storage bucket. Make sure you allow public access to the bucket. The bucket can be used by Spark to read data directly from the Cloud bucket.

# Google DataProc

DataProc is a service that allows the full deployment of a Big Data platform consisting of the Hadoop Distributed File System (HDFS), Yet Another Resource Negotiator (YARN) and Spark.

We will use Compute Engine to deploy this cluster, but if you are acquinted with Google Kubernetes Engine, then you can use this option.

## Task 3

Access the DataProc interface. If it does not appear in your side-bar Menu, then go to the dashboard and use the search bar (at the top) to search for DataProc.

Activate the DataProc API.

# Creating a cluster

## Task 4

Access the DataProc interface. Click on Create Cluster from the top bar. Select Compute Engine.

Select a location in Europe. Select the Standard version of deployment (1 master, N workers).

Scroll down and make sure you **Enable Component Gateway** and select the **Jupyter Notebook**.

From the left menu Click on **Configure nodes**. Because the limits on the educational account, I recommend the following configuration:

* Educational Account Quota:
  + Limited to 8 vCPUs, 500 GB SSD
  + You can choose n2-highmem-2 (2vCPU, 16GB memory) for both Master and Worker nodes
  + You can use standard persistent disk instead of Balanced persistent disk.

From the left menu Click on **Customize Custer**. Disable the check on **Internal IP only**.

Click **CREATE**. When ready, you can access the **Jupyter Lab** interface from the Cluster details, as in the following Figure.

A screenshot of a computer

Description automatically generated

# Running Spark jobs using Cloud Storage data

Once you open the Jupyter interface you will find two folders:

* GCS (A Cloud Storage bucket created for this cluster)
* Local Disk (The local disk of the master VM)

If you save your notebook in GCS, then they will be accessible in the corresponding bucket instantiated for this cluster. The storage bucket remains active and can be reused for multiple instantiations of DataProc clusters.

If you save your notebook in the Local Disk then the disk will be erased when deleting the cluster and your notebook wil not be available in a new cluster.

Open a notebook in the GCS folder and write the following code to instantiate a spark session:

* from pyspark.sql import SparkSession
* spark = SparkSession.builder \
* .appName("MyApp") \
* .master("yarn") \
* .config("spark.executor.memory", "6g") \
* .config("spark.executor.cores", "2") \
* .config("spark.executor.instances", "2") \
* .getOrCreate()
* print(spark)

To read data from the Bucket Storage created in Chapter 3 you can use the following code:

df = spark.read.json("gs://bdt-bucket-2024/data")

where bdt-bucket-2024/data should point to the bucket your created and the folder with data in your bucket.

We can apply a simple processing to assess the performance of our system. The following figure shows an example processing:

A screenshot of a computer code

Description automatically generated

# Storing data in HDFS

To store the data in HDFS we must first download it to the master and then store it in the Hadoop Distributed File System. To achieve this, you can open a Terminal in the Jupyter interface. Then we can use the following commands:

1. Copy data to the master machine

gsutil cp -r gs://your-bucket/your/data ./

1. Create a folder for your data in the HDFS system

hdfs dfs -mkdir -p /user/yourName/data

hdfs dfs -put ./data /user/yourName/data

The data is now stored in HDFS.

To read data from HDFS you just use the hdfs protocol and use your data path created using the above steps.

A screenshot of a computer code

Description automatically generated

# Which is faster?

Compare the time required to execute the simple processing using Cloud Storage and HDFS. Which is better?