For this project, a hadoop cluster had to be set up which consists of a master and at least two worker nodes that are interconnected together by a network, be it private or internet. Hadoop is popular for being able to store large amount of data in distributed fashion called HDFS and they can also process huge amount of data in parallel using MapReduce. For this project, we are going to deploy a Hadoop cluster in Google Cloud using *Dataproc* that has one master and two slaves. We are also going to run two jobs, one being counting number of words in a large text file, and another being estrimating value of PI using Monte Carlo methods.

**Setting Up Hadoop/Spark Service using DataProc:**

First step in creating a Hadoop cluster is creating a yaml file which describes the properties of our cluster. Since I already did have a cluster running, I wanted to simply copy the configuration yaml from the running cluster to my local machine.

#Copying already running cluster configuration into local

gcloud dataproc clusters export cluster-898a --destination cluster.yaml --region us-central1

#Config file for our cluster

config:

configBucket: dataproc-staging-us-central1-475931246002-ptsog0z7

encryptionConfig: {}

endpointConfig: {}

gceClusterConfig:

networkUri: https://www.googleapis.com/compute/v1/projects/codingrant/global/networks/default

serviceAccountScopes:

- https://www.googleapis.com/auth/bigquery

- https://www.googleapis.com/auth/bigtable.admin.table

- https://www.googleapis.com/auth/bigtable.data

- https://www.googleapis.com/auth/cloud.useraccounts.readonly

- https://www.googleapis.com/auth/devstorage.full\_control

- https://www.googleapis.com/auth/devstorage.read\_write

- https://www.googleapis.com/auth/logging.write

shieldedInstanceConfig: {}

zoneUri: https://www.googleapis.com/compute/v1/projects/codingrant/zones/us-central1-a

masterConfig:

diskConfig:

bootDiskSizeGb: 500

bootDiskType: pd-standard

imageUri: https://www.googleapis.com/compute/v1/projects/cloud-dataproc/global/images/dataproc-1-5-ubu18-20210311-093551-rc01

machineTypeUri: https://www.googleapis.com/compute/v1/projects/codingrant/zones/us-central1-a/machineTypes/n1-standard-4

minCpuPlatform: AUTOMATIC

numInstances: 1

preemptibility: NON\_PREEMPTIBLE

securityConfig:

kerberosConfig: {}

softwareConfig:

imageVersion: 1.5.34-ubuntu18

optionalComponents:

- HIVE\_WEBHCAT

- ANACONDA

- DOCKER

properties:

capacity-scheduler:yarn.scheduler.capacity.root.default.ordering-policy: fair

core:fs.gs.block.size: '134217728'

core:fs.gs.metadata.cache.enable: 'false'

core:hadoop.ssl.enabled.protocols: TLSv1,TLSv1.1,TLSv1.2

distcp:mapreduce.map.java.opts: -Xmx768m

distcp:mapreduce.map.memory.mb: '1024'

distcp:mapreduce.reduce.java.opts: -Xmx768m

distcp:mapreduce.reduce.memory.mb: '1024'

hdfs:dfs.datanode.address: 0.0.0.0:9866

hdfs:dfs.datanode.http.address: 0.0.0.0:9864

hdfs:dfs.datanode.https.address: 0.0.0.0:9865

hdfs:dfs.datanode.ipc.address: 0.0.0.0:9867

hdfs:dfs.namenode.handler.count: '20'

hdfs:dfs.namenode.http-address: 0.0.0.0:9870

hdfs:dfs.namenode.https-address: 0.0.0.0:9871

hdfs:dfs.namenode.secondary.http-address: 0.0.0.0:9868

hdfs:dfs.namenode.secondary.https-address: 0.0.0.0:9869

hdfs:dfs.namenode.service.handler.count: '10'

hive:hive.fetch.task.conversion: none

mapred-env:HADOOP\_JOB\_HISTORYSERVER\_HEAPSIZE: '3840'

mapred:mapreduce.job.maps: '21'

mapred:mapreduce.job.reduce.slowstart.completedmaps: '0.95'

mapred:mapreduce.job.reduces: '7'

mapred:mapreduce.jobhistory.recovery.store.class: org.apache.hadoop.mapreduce.v2.hs.HistoryServerLeveldbStateStoreService

mapred:mapreduce.map.cpu.vcores: '1'

mapred:mapreduce.map.java.opts: -Xmx2457m

mapred:mapreduce.map.memory.mb: '3072'

mapred:mapreduce.reduce.cpu.vcores: '1'

mapred:mapreduce.reduce.java.opts: -Xmx2457m

mapred:mapreduce.reduce.memory.mb: '3072'

mapred:mapreduce.task.io.sort.mb: '256'

mapred:yarn.app.mapreduce.am.command-opts: -Xmx2457m

mapred:yarn.app.mapreduce.am.resource.cpu-vcores: '1'

mapred:yarn.app.mapreduce.am.resource.mb: '3072'

spark-env:SPARK\_DAEMON\_MEMORY: 3840m

spark:spark.driver.maxResultSize: 1920m

spark:spark.driver.memory: 3840m

spark:spark.executor.cores: '2'

spark:spark.executor.instances: '2'

spark:spark.executor.memory: 5586m

spark:spark.executorEnv.OPENBLAS\_NUM\_THREADS: '1'

spark:spark.scheduler.mode: FAIR

spark:spark.sql.cbo.enabled: 'true'

spark:spark.ui.port: '0'

spark:spark.yarn.am.memory: 640m

yarn-env:YARN\_NODEMANAGER\_HEAPSIZE: '3840'

yarn-env:YARN\_RESOURCEMANAGER\_HEAPSIZE: '3840'

yarn-env:YARN\_TIMELINESERVER\_HEAPSIZE: '3840'

yarn:yarn.nodemanager.address: 0.0.0.0:8026

yarn:yarn.nodemanager.resource.cpu-vcores: '4'

yarn:yarn.nodemanager.resource.memory-mb: '12288'

yarn:yarn.resourcemanager.nodemanager-graceful-decommission-timeout-secs: '86400'

yarn:yarn.scheduler.maximum-allocation-mb: '12288'

yarn:yarn.scheduler.minimum-allocation-mb: '1024'

tempBucket: dataproc-temp-us-central1-475931246002-1uwyswuw

workerConfig:

diskConfig:

bootDiskSizeGb: 500

bootDiskType: pd-standard

imageUri: https://www.googleapis.com/compute/v1/projects/cloud-dataproc/global/images/dataproc-1-5-ubu18-20210311-093551-rc01

machineTypeUri: https://www.googleapis.com/compute/v1/projects/codingrant/zones/us-central1-a/machineTypes/n1-standard-4

minCpuPlatform: AUTOMATIC

numInstances: 2

preemptibility: NON\_PREEMPTIBLE

#Running a cluster using the yaml file

gcloud dataproc clusters import my-hadoop-cluster --source cluster.yaml --region us-central1

This command enabled us to successfully run a spark/Hadoop cluster with one master and 2 slaves. It also has HDFS and spark service running in all the nodes in the cluster.

**Running my first spark Job- Counting Words:**

Since, master and slaves we created earlier has everything set up for us, we can now start working on trying to get out first job running in the pyspark. First let’s SSH into the master node and then download the file whose texts we will count by the end of this section.

#Downloading the text version of our text file

pawanbhatta178@cluster-898a-m:~$ wget <https://www.gutenberg.org/files/1342/1342-0.txt>

#Copying the file into a folder that I want to share with my HDFS cluster

pawanbhatta178@cluster-898a-m:~$ sudo cp 1342-0.txt ./hadoopFiles/prideAndPrejudice.txt

#Checking if the file was successfully created

pawanbhatta178@cluster-898a-m:~$ cd hadoopFiles/

pawanbhatta178@cluster-898a-m:~/hadoopFiles$ ls

prideAndPrejudice.txt

Now it is time to upload the folder from the local file system in master node