# **Setting up Environment in GCP**

**End Goal**

To setup a distributed scalable environment on GCP for DGA Feed Training and Inference Engine that looks like this.

Cisco 1M 
Dataset 
(GOOD) 
Bambenek 
DGA Feed 
(BAD) 
Preprocess 
Cleanup & 
Merge 
Kafka 
Producer 
Data Ingest 
Kafka 
Processed 
Dataset 
DNS Server/lnference Engine 

## Prerequisites

* Setup GCP account
* Installed Google Cloud SDK on local machine (<https://cloud.google.com/sdk/install>)
* Created a project

Google Cloud Platform 
Compute Engine 
VM instances 
Instance groups 
Instance templates 
Sole tenant nodes 
Disks 
Snapshots 
Images 
TPUS 
Committed use discounts 
Metadata 
Health checks 
Zones 
Network endpoint groups 
Operations 
VM instances 
Filter VM instances 
a CREATE INSTANCE 
IMPORT VM 
shaped-icon-217910 
w266-191606 
70968413100 
qualified-smile-191606 
C REFRESH 
NEW PROJECT 
Select from BERKELEY.EDU 
Search projects and folders 
RECENT 
Name 
210-1 0 
w266 0 
berkeley.edu O 
My First Project O 

## Kubernetes setup

* On your browser go to kubernetes engine page and select the project (This will take several mins to start kubernetes engine) <https://console.cloud.google.com/projectselector/kubernetes?_ga=2.201928657>.- 1707404544.1516051830

* While kubernetes engine gets ready, on your local machine, pull the docker images

$ docker pull confluentinc/cp-zookeeper:latest

$ docker pull confluentinc/cp-kafka:latest

$ docker pull midsw205/cdh-minimal:latest

$ docker pull midsw205/spark-python:0.0.5

$ docker pull midsw205/base:0.1.9

* Check the images

$ docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

confluentinc/cp-kafka latest 373a4e31e02e 6 weeks ago 558MB

confluentinc/cp-zookeeper latest 18b57832a1e2 4 months ago 562MB

midsw205/spark-python latest a554c3520502 6 months ago 3.87GB

midsw205/base latest 03fff049d97a 6 months ago 2.7GB

* Install kubectl

$ gcloud components install kubectl

* Tag the images to be pushed to GCP by using the image ids seen in the docker images output above.

$ docker tag a554c3520502 gcr.io/w210-1/spark-python

$ docker tag 03fff049d97a gcr.io/w210-1/mids

$ docker tag 373a4e31e02e gcr.io/w210-1/kafka

$ docker tag 18b57832a1e2 gcr.io/w210-1/zookeeper

* Push the tagged images to gcloud

$ gcloud docker --push gcr.io/w210-1/spark-python

$ gcloud docker --push gcr.io/w210-1/mids

$ gcloud docker --push gcr.io/w210-1/kafka

$ gcloud docker --push gcr.io/w210-1/zookeeper

* Create a cluster in gcloud

$ gcloud container clusters create kafka --num-nodes=5 --zone northamerica-northeast1-a

This creates a cluster with 5 nodes below (5 nodes is the maximum in n1-standard-1 flavor). You can change the zone based on where you live.

* Check if the cluster is created and the computes are operational:

$ gcloud compute instances list

NAME ZONE MACHINE\_TYPE PREEMPTIBLE INTERNAL\_IP EXTERNAL\_IP STATUS

gke-kafka-default-pool-6a5787d6-299m northamerica-northeast1-a n1-standard-1 10.162.0.6 35.203.30.177 RUNNING

gke-kafka-default-pool-6a5787d6-4gl4 northamerica-northeast1-a n1-standard-1 10.162.0.2 35.203.93.16 RUNNING

gke-kafka-default-pool-6a5787d6-l11c northamerica-northeast1-a n1-standard-1 10.162.0.4 35.203.7.23 RUNNING

gke-kafka-default-pool-6a5787d6-pwq6 northamerica-northeast1-a n1-standard-1 10.162.0.5 35.203.47.5 RUNNING

gke-kafka-default-pool-6a5787d6-vczj northamerica-northeast1-a n1-standard-1 10.162.0.3 35.203.80.232 RUNNING

**Deploying the containers**

* We will use the following files in gcp directory

$ ls -1 \*.yaml

kafka-deployment.yaml

kafka-service.yaml

mids-claim0-persistentvolumeclaim.yaml

mids-deployment.yaml

mids-service.yaml

myhdfs-deployment.yaml

myhdfs-service.yaml

myspark-deployment.yaml

myspark-service.yaml

zookeeper-deployment.yaml

zookeeper-service.yaml

* Use kubectl to bringup service and deployments

kubectl create --filename zookeeper-deployment.yaml

kubectl create --filename kafka-deployment.yaml

kubectl create --filename mids-deployment.yaml

kubectl create --filename myspark-deployment.yaml

kubectl create --filename zookeeper-service.yaml

kubectl create --filename kafka-service.yaml

kubectl create --filename mids-service.yaml

kubectl create --filename myspark-service.yaml

* Check if all the containers are deployed

$ kubectl get pods -o wide

NAME READY STATUS RESTARTS AGE IP NODE

kafka-5c7bb56cbd-8ndmk 1/1 Running 0 4d 10.20.0.4 gke-kafka-default-pool-6a5787d6-4gl4

mids-545f9676c-z2jvf 1/1 Running 0 4d 10.20.2.5 gke-kafka-default-pool-6a5787d6-l11c

myspark-cd94d8765-vvdf8 1/1 Running 0 2m 10.20.1.7 gke-kafka-default-pool-6a5787d6-pwq6

zookeeper-6cbcdd499f-wmsnq 1/1 Running 0 5d 10.20.1.5 gke-kafka-default-pool-6a5787d6-pwq6

**Verifying the message flow**

* Login to the Kafka container

$ kubectl exec -it kafka-5c7bb56cbd-8ndmk bash

root@kafka-5c7bb56cbd-8ndmk:/#

* Create a topic 210test

root@kafka-5c7bb56cbd-8ndmk:/# kafka-topics --create --topic **210test** --partitions 1 --replication-factor 1 --if-not-exists --zookeeper zookeeper:32181

Created topic "210test".

root@kafka-5c7bb56cbd-8ndmk:/#

* Produce a random sequence and publish to the topic 210test

root@kafka-5c7bb56cbd-8ndmk:/# seq 42 | kafka-console-producer --request-required-acks 1 --broker-list localhost:29092 --topic **210test**

>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

root@kafka-5c7bb56cbd-8ndmk:/#

* Login to the Spark container
* fetch messages through pyspark

$ kubectl exec -it myspark-cd94d8765-vvdf8 bash

root@myspark-cd94d8765-vvdf8:/spark-2.2.0-bin-hadoop2.6#

* Launch pyspark

root@myspark-cd94d8765-vvdf8:/spark-2.2.0-bin-hadoop2.6# pyspark

Python 3.6.1 |Anaconda custom (64-bit)| (default, May 11 2017, 13:09:58)

[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux

:::: <Some ignorable warnings>

Welcome to

\_\_\_\_ \_\_

/ \_\_/\_\_ \_\_\_ \_\_\_\_\_/ /\_\_

\_\ \/ \_ \/ \_ `/ \_\_/ '\_/

/\_\_ / .\_\_/\\_,\_/\_/ /\_/\\_\ version 2.2.0

/\_/

Using Python version 3.6.1 (default, May 11 2017 13:09:58)

SparkSession available as 'spark'.

>>>

* Fetch messages from kafka with topic 210test

>>> numbers = spark \

... .read \

... .format("kafka") \

... .option("kafka.bootstrap.servers", "kafka:29092") \

... .option("subscribe","210test") \

... .option("startingOffsets", "earliest") \

... .option("endingOffsets", "latest") \

... .load()

>>> numbers.show()

+----+-------+-------+---------+------+--------------------+-------------+

| key| value| topic|partition|offset| timestamp|timestampType|

+----+-------+-------+---------+------+--------------------+-------------+

|null| [31]|210test| 0| 0|2018-10-07 19:31:...| 0|

|null| [32]|210test| 0| 1|2018-10-07 19:31:...| 0|

|null| [33]|210test| 0| 2|2018-10-07 19:31:...| 0|

|null| [34]|210test| 0| 3|2018-10-07 19:31:...| 0|

|null| [35]|210test| 0| 4|2018-10-07 19:31:...| 0|

|null| [36]|210test| 0| 5|2018-10-07 19:31:...| 0|

|null| [37]|210test| 0| 6|2018-10-07 19:31:...| 0|

|null| [38]|210test| 0| 7|2018-10-07 19:31:...| 0|

|null| [39]|210test| 0| 8|2018-10-07 19:31:...| 0|

|null|[31 30]|210test| 0| 9|2018-10-07 19:31:...| 0|

|null|[31 31]|210test| 0| 10|2018-10-07 19:31:...| 0|

|null|[31 32]|210test| 0| 11|2018-10-07 19:31:...| 0|

|null|[31 33]|210test| 0| 12|2018-10-07 19:31:...| 0|

|null|[31 34]|210test| 0| 13|2018-10-07 19:31:...| 0|

|null|[31 35]|210test| 0| 14|2018-10-07 19:31:...| 0|

|null|[31 36]|210test| 0| 15|2018-10-07 19:31:...| 0|

|null|[31 37]|210test| 0| 16|2018-10-07 19:31:...| 0|

|null|[31 38]|210test| 0| 17|2018-10-07 19:31:...| 0|

|null|[31 39]|210test| 0| 18|2018-10-07 19:31:...| 0|

|null|[32 30]|210test| 0| 19|2018-10-07 19:31:...| 0|

+----+-------+-------+---------+------+--------------------+-------------+

only showing top 20 rows

All Set!!!!

**References:**

* https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app <https://kubernetes.io/docs/admin/cluster-large/>
* Special thanks to Pavan Kurapati for publishing the original procedure during his 205, that helped me setup this.