Hadoop Container with underlying ceph S3-storage

The container enables the user to run ***Hadoop*** client application/frameworks such as ***Hive*** (SQL on map-reduce) , ***TPCDS*** (benchmark ,generating data for later execution) , ***TeraGen***(another benchmark), ***SPARK*** (enable analytic-executions with various tools such as Scala,Python,R)

The container saved a lot of work (I did those installations several times on different machines), hadoop and the other components required some configuration. It's not 'plug-and-play'.

It's for the user who needs to test its ceph-s3 against hadoop-ecosystem.

The following will explain how to use hadoop-container which run hadoop ecosystem including
*) hadoop (3.0.0) - enables to run map-reduce jobs
*) TPCDS
*) teragen -> terasort -> teravalidate.
*) hive (3.0.0)
*) spark(3.0.0)

The container enables all of the above while using ceph as the underlying storage(s3).

The container should run as follows
sudo docker run -h mydocker -it galsl/hadoop:alpha   sh -c '/work/generate_key.bash ; /bin/bash'

Using podman you should use the following

```
podman run --user root -h mydocker -it galsl/hadoop:alpha   sh -c
'/work/generate_key.bash ; /bin/bash'
```

In case it's the first time it's running on a user host, it will first download the image and then run the container.
The container starts with a bash prompt (user root).
It creates ssh-keys , and boot sshd service, it's essential for hadoop processes, and also enables ssh into the running container.

Note: you can use the following command for ssh (it reduce some overhead)
ssh root@<container-id> -y -o "UserKnownHostsFile /dev/null" -o "StrictHostKeyChecking no"
Password: docker

Adding hostname (mydocker) to /etc/hosts is very helpful , it enables browsers (http://mydocker:8088/cluster) retrieves hadoop status/configuration/operations/logs.

The shell(Bash) contains functions to ease the various operations the user needs to do.
show_help print out the usage of those functions.

check_sshd check sshd service status
Before booting hadoop processes, it need to modify configuration, it is done by running
deploy_ceph_s3a_ip <host-ip>

Upon modifying configuration with host-ip , it's better to validate that ceph is reachable (should be up and running …).

check_s3_connect , send simple /ls/ request to ceph, it should respond immediately.

If this is not the case , it may relate to firewalls on the host machine, the following should resolve that.

1. ***sudo systemctl disable firewalld***   //disable firewall
2. f***irewall-cmd --zone=public --add-port=8000/tcp*** // Mark.k suggestion, not to disable firewall but to add rule.

Before starting with anything related to hadoop (hive,spark,etc) , it should be up & running.

start_hadoop will boot hadoop , it also runs jps (java ps) , respond should contain 6 processes.

```
7961 Jps
1340 NodeManager
461 DataNode
1021 ResourceManager
318 NameNode
735 SecondaryNameNode
```

stop_hadoop will shutdown all hadoop processes.
And again …   browse (http://mydocker:8088/cluster) into hadoop, and retrieve status/configuration/operations/logs , it helps a lot to understand what's going on.

Some useful hadoop commands.
mapred job -list // show list of current jobs and their status
mapred job -kill <id> // kill specific job

Running TPCDS
TPCDS is generating data (24 tables), and users can control generated-data size with SCALE env-var.
SCALE=2 is the minimum value, 1000 will generate TeraByte of data.

SCALE=2 start_tpcds // will run for several minutes and spill report on console while execution.

Upon TPCDS is complete, it is possible to run hive queries.
Before getting into hive session, it need to init meta-store-db,define and load tables, by running ***load_s3_data_into_hive_db***

Running Hive

cd_tpcds;hive will open a hive session, there it's possible to run queries on data generated by TPCDS.

```
hive> use tpcds_bench; // switch to database tpcds_bench
hive> show tables;    // list of tables exist in database
hive> select count(*) from call_center; // simple query , will commence map-reduce jobs , can be viewed on browser
```

Running Spark

Upon running hadoop/tpcds , it's worth getting into spark-shell and doing some stuff on S3.
Type ***spark_shell*** and it will open a scala session.
The following command lines enable users to access files on S3 and to do some operations on it.

```
scala> import java.io.File
import java.io.File

scala> sc.hadoopConfiguration.set("fs.s3a.awsAccessKeyId", "b2345678901234567890")

scala> sc.hadoopConfiguration.set("fs.s3a.awsSecretAccessKey",
"b234567890123456789012345678901234567890")

scala> sc.textFile("s3a://tpcds2/2/reason/data-m-00001").count()
res6: Long = 36

scala> sc.textFile("s3a://tpcds2/2/reason/data-m-00001").foreach( println )
16|AAAAAAAAABAAAAAA|Did not fit|
17|AAAAAAAABBAAAAAA|Wrong size|
18|AAAAAAAACBAAAAAA|Lost my job|
19|AAAAAAAADBAAAAAA|unauthoized purchase|
.
.
```

Create parquet file from CSV

Creating parquet using this container reduces a lot of effort.
On /work directory, reside a script-file create_parquet_file.bash (need to source it)
Before creating parquet file, ceph-s3 should be up and running, the container should be connected (as explained above), also hadoop should be up and running (above),
The creation of parquet is done using hive, the bash-function is taking care of all necessary steps.
Just run create_parquet from command-line.
The content of the parquet file is according to the /create_csv/ bash function, thus any CSV file can be used.
should note also that the table-scheme should be altered to CSV file.

Note: as mentioned this container save a lot of time, probably its not fit to everybody needs
Or missing some more installations.
It will be great to change/modify the container and save it in a different name/tag.