INSTALLING AND RUNNING GOFFISH

Installation:

We can use Ubuntu (14.04) as the platform for installing and running Goffish though it works on other platforms as well with few changes. For our project, we choose Ubuntu as the platform.

Given below are the steps we use for building goffish from scratch in a freshly installed Ubuntu machine.

Step 1:

The tar file ("scripts.tar.gz") provided is firstly copied in the home directory. Untar the file. We will get the folder "scripts" after we untar the file. Make sure that the folder is present in home as some scripts refer to ~/scripts for copying some resources.

Give necessary permissions to all the files inside the scripts folder.

>> chmod 777 <Filename> will do the job.

e.g. >>chmod 777 install.sh inside scripts folder

Step 2:

Go inside the scripts folder and run install.sh.

>>cd scripts

>>sudo bash install.sh

The script will download and install following softwares in order:

- Java 1.7
- Maven
- unzip
- openssh-server
- cmake
- g++
- Floe
- Goffish

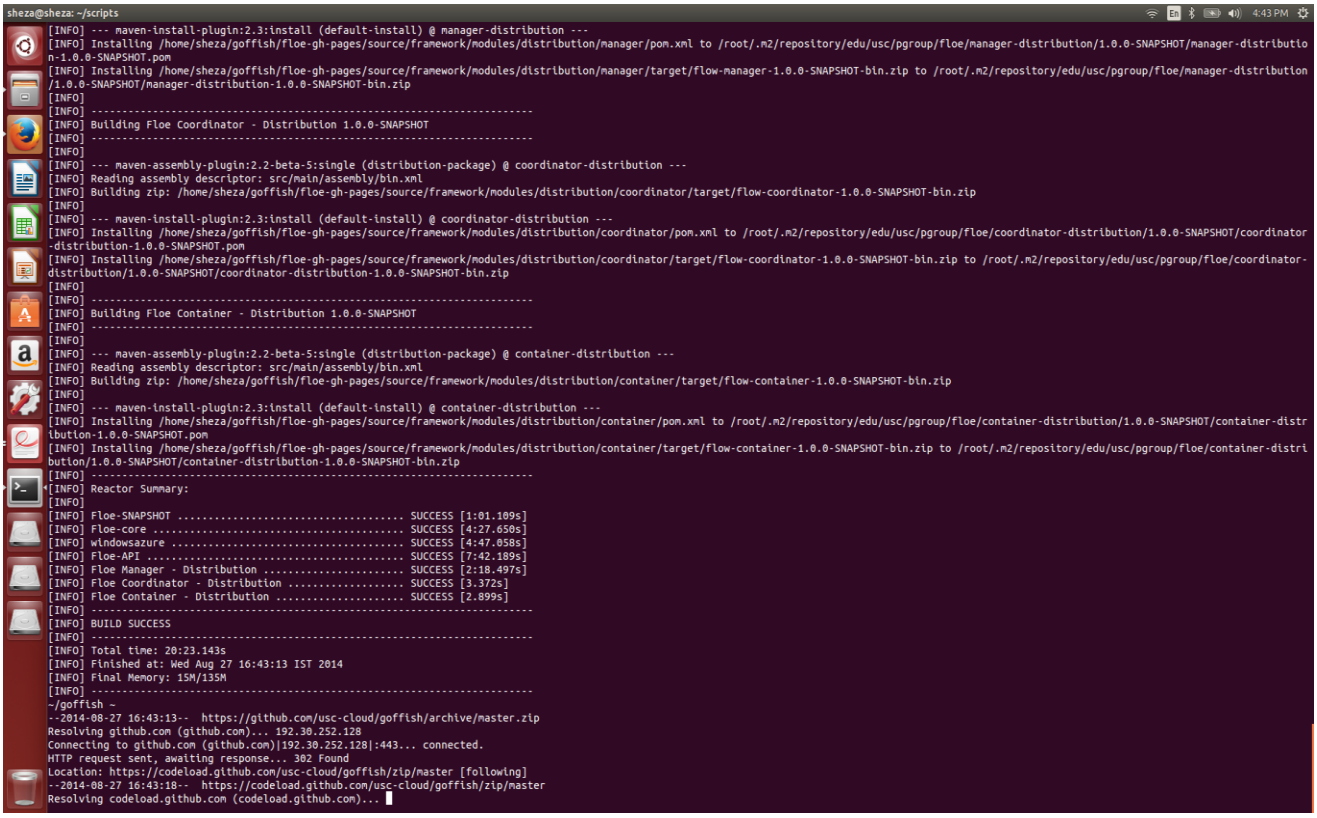
We use this script to install all necessary software in a freshly installed Ubuntu system.

However if java 1.7 is already installed in the system and JAVA_HOME variable is properly set in .bashrc file then comment out the following lines in install.sh script to avoid repetition.

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
echo 'export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64 ' >>~/.bashrc
export PATH=$JAVA_HOME:$PATH
echo 'export PATH=$JAVA_HOME:$PATH' >>~/.bashrc
```

Downloading and installing so many softwares will take some time.

Along with the software installation the script also downloads and builds floe and goffish using maven. Make sure at the end all the goffish modules are successfully bulid.



```
sheza@sheza: ~/scripts
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:testCompile (default-testCompile) @ road-network-gml-generator ---
[INFO] No sources to compile
[INFO]
[INFO] --- maven-surefire-plugin:2.10:test (default-test) @ road-network-gml-generator ---
[INFO] No tests to run.
[INFO] Surefire report directory: /home/sheza/goffish/goffish-master/goffish-trunk/tools/road-network-gml-generator/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------

Results :

Tests run: 0, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:2.2:jar (default-jar) @ road-network-gml-generator ---
[INFO] Building jar: /home/sheza/goffish/goffish-master/goffish-trunk/tools/road-network-gml-generator/target/road-network-gml-generator-2.0.jar
[INFO]
[INFO] --- maven-install-plugin:2.3:install (default-install) @ road-network-gml-generator ---
[INFO] Installing /home/sheza/goffish/goffish-master/goffish-trunk/tools/road-network-gml-generator/target/road-network-gml-generator-2.0.jar to /root/.m2/repository/edu/usc/goffish/road-network-gml-genera
tor/2.0/road-network-gml-generator-2.0.jar
[INFO] Installing /home/sheza/goffish/goffish-master/goffish-trunk/tools/road-network-gml-generator/pom.xml to /root/.m2/repository/edu/usc/goffish/road-network-gml-generator/2.0/road-network-gml-generator
-2.0.pom
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Goffish ........................................... SUCCESS [13.649s]
[INFO] GoFS .............................................. SUCCESS [0.005s]
[INFO] GoFS-API .......................................... SUCCESS [2:00.249s]
[INFO] GoFFish-Core ...................................... SUCCESS [44.745s]
[INFO] GoFS-Distribution ................................. SUCCESS [1:16.906s]
[INFO] Gopher ............................................ SUCCESS [0.003s]
[INFO] Gopher-API ........................................ SUCCESS [7.650s]
[INFO] Gopher-Core ....................................... SUCCESS [33.370s]
[INFO] Gopher - Distribution ............................. SUCCESS [20.815s]
[INFO] Gopher - client ................................... SUCCESS [4.517s]
[INFO] Gopher samples .................................... SUCCESS [0.003s]
[INFO] Gopher-Connected-Components ....................... SUCCESS [0.396s]
[INFO] Gopher-Sample-Vert-count .......................... SUCCESS [0.438s]
[INFO] Gopher-Page-Rank .................................. SUCCESS [0.487s]
[INFO] Gopher-Shortest-Path .............................. SUCCESS [0.452s]
[INFO] BlockRank ......................................... SUCCESS [0.356s]
[INFO] Gopher-Licence Plate Tracer ....................... SUCCESS [0.329s]
[INFO] Gopher-N_Hop_Stat_Collector ....................... SUCCESS [6.780s]
[INFO] Goffish tools ..................................... SUCCESS [0.003s]
[INFO] GoFFish-GML Generator ............................. SUCCESS [1.768s]
[INFO] GoFFish-Road Network GML Generator ................ SUCCESS [1.899s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5:35.348s
[INFO] Finished at: Wed Aug 27 16:51:06 IST 2014
[INFO] Final Memory: 32M/120M
[INFO] ------------------------------------------------------------------------
~/goffish ~
sheza@sheza:~/scripts$
```

After the installation do:

>>`source ~/.bashrc` to make all the changes visible in current shell session.

If everything works fine at the end of step 2 we will have a folder “goffish” in the home directory containing two folders namely “floe-gh-pages” and “goffish-master”.

Step 3(Deploying):

Change the permissions for deploy.sh and run the script

>>chmod 777 deploy.sh

>>sudo bash deploy.sh

Purpose of this script is to create a working environment for goffish by creating folder structure and copying necessary jar files into appropriate locations.

After running this script if everything works correctly we can see a third folder namely “deployment” in the earlier “goffish” directory.

```
sheza@sheza: ~/scripts
  inflating: gopher-client-2.0/lib/jargs-1.0.jar
  inflating: gopher-client-2.0/lib/jaxen-1.1.4.jar
  inflating: gopher-client-2.0/lib/log4j-1.2.9.jar
  inflating: gopher-client-2.0/lib/apfloat-1.6.3.jar
  inflating: gopher-client-2.0/lib/apfloat-samples-1.6.3.jar
  inflating: gopher-client-2.0/lib/slf4j-api-1.6.1.jar
  inflating: gopher-client-2.0/lib/bcprov-jdk15on-1.47.jar
  inflating: gopher-client-2.0/lib/bcpkix-jdk15on-1.47.jar
  inflating: gopher-client-2.0/lib/axiom-api-1.2.14.jar
  inflating: gopher-client-2.0/lib/geronimo-activation_1.1_spec-1.1.jar
  inflating: gopher-client-2.0/lib/geronimo-javamail_1.4_spec-1.7.1.jar
  inflating: gopher-client-2.0/lib/geronimo-stax-api_1.0_spec-1.0.1.jar
  inflating: gopher-client-2.0/lib/apache-mime4j-core-0.7.2.jar
  inflating: gopher-client-2.0/lib/axiom-impl-1.2.14.jar
  inflating: gopher-client-2.0/lib/woodstox-core-asl-4.1.4.jar
  inflating: gopher-client-2.0/lib/stax2-api-3.1.1.jar
  inflating: gopher-client-2.0/lib/jsch-0.1.50.jar
~/goffish/deployment/samples ~/goffish/deployment ~/goffish ~
~/goffish/deployment/samples/gofs-samples ~/goffish/deployment/samples ~/goffish/deployment ~/goffish ~
~/goffish/deployment/samples ~/goffish/deployment ~/goffish ~
~/goffish/deployment/samples/gopher-jars ~/goffish/deployment/samples ~/goffish/deployment ~/goffish ~
~/goffish/deployment/samples ~/goffish/deployment ~/goffish ~
~/goffish/deployment ~/goffish ~
~/goffish/deployment/gopher-client-2.0/bin ~/goffish/deployment ~/goffish ~
~/goffish/deployment ~/goffish ~
~/goffish/deployment/goffish_conf ~/goffish/deployment ~/goffish ~
~/goffish/deployment ~/goffish ~
{u'class_path': u'edu.usc.pgroup.goffish.gopher.sample.VertCounter', u'arguments': u'NILL', u'graph_id': u'fbgraph', u'iterations': u'0', u'jar_name': u'vert-count-2.0.jar'}
# Config File for formatting a GoFS cluster
# ----------------------------------------
# the name node url
gofs.namenode.type = edu.usc.goffish.gofs.namenode.RemoteNameNode
gofs.namenode.location = http://localhost:9998
# list of data nodes to format and add to the name node
# repeat for each data node to include
gofs.datanode =file://localhost/home/sheza/goffish/deployment/goffish_home/gofs-data/
# full class name of the serializer to use at every data node

# gofs.serializer = edu.usc.goffish.gofs.slice.JavaSliceSerializer
gofs.serializer = edu.usc.goffish.gofs.slice.KryoSliceSerializer

./GoFSDeployGraph edu.usc.goffish.gofs.namenode.RemoteNameNode http://localhost:9998 "fbgraph" 1 /home/sheza/goffish/deployment/samples/gofs-samples/list.xml
<gml><template>/home/sheza/goffish/deployment/samples/gofs-samples/graphs/fbin.txt/fbin.txt-template.gml</template><instances /></gml>
/home/sheza/goffish
/home/sheza/goffish/deployment/gopher-client-2.0/bin/CopyArtifacts.sh /home/sheza/goffish/deployment/samples/gopher-jars/vert-count-2.0.jar vert-count-2.0.jar ${USER}  localhost

/home/sheza/goffish/deployment/gopher-client-2.0/bin/StartServers.sh ${USER} localhost localhost fbgraph file://localhost/home/sheza/goffish/deployment/goffish_home/gofs-data/gofs

sleep 5s;

/home/sheza/goffish/deployment/gopher-client-2.0/bin/DeployGopher.sh /home/sheza/goffish/deployment/goffish_home/gofs-2.0/conf/gofs.config localhost localhost


./GopherClient.sh /home/sheza/goffish/deployment/gopher-client-2.0/gopher-config.xml /home/sheza/goffish/deployment/goffish_home/gofs-2.0/conf/gofs.config fbgraph vert-count-2.0.jar edu.usc.pgroup.goffish.
gopher.sample.VertCounter NILL
kill $(ps aux | grep -v [N]ameNode | grep '[l]ocalhost' | awk '{print $2}')

sheza@sheza:~/scripts$
```

Again after the installation do:

`>>source ~/.bashrc` to make all the changes visible in the current shell session.

Step 4(Security):

As goffish is a distributed system, lot of file exchanges between different machines takes place using password less ssh. Even for a single machine deployment we need password less ssh to localhost.

First start the ssh server using:

>>sudo service ssh restart

Now test if we can do password less ssh to localhost using:

>>ssh localhost

If you are asked for the password then do the following steps to do password less ssh to localhost:

1.generate the public private keys

```
>>ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
```

2.authorize the key by adding it to the list of authorized keys

```
>> cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

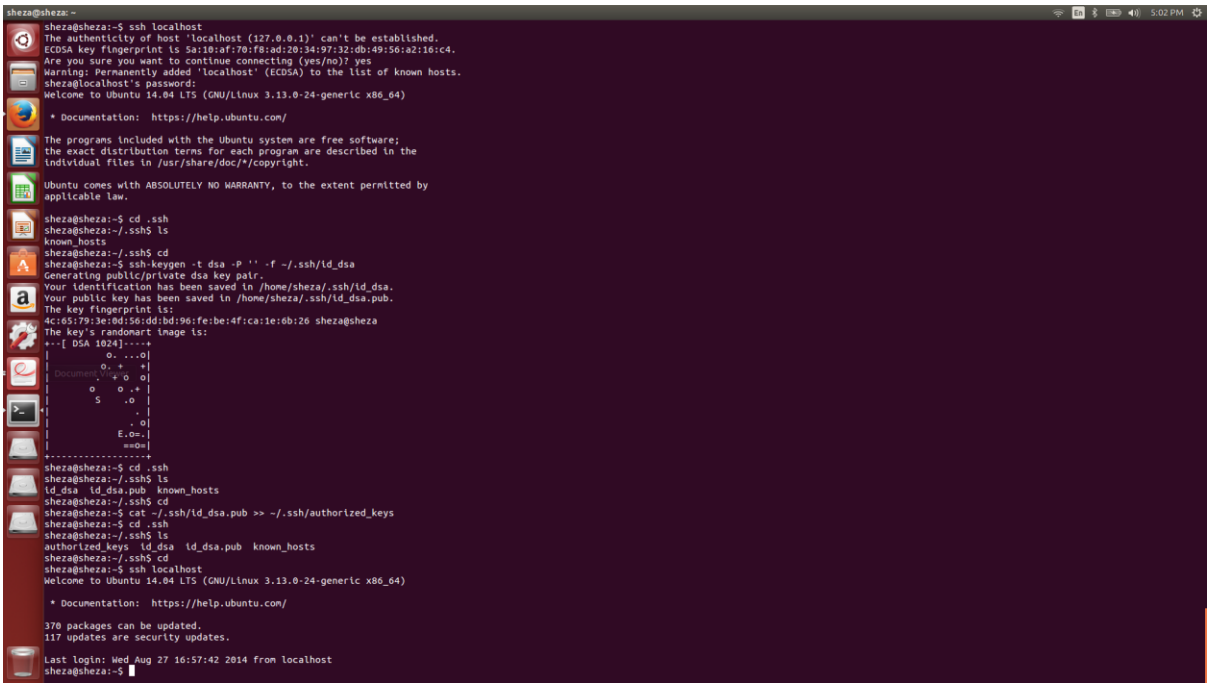
3.test that we can log in with no password

>>ssh localhost

If still it asks for the password then do the following:

```
>>cat .ssh/id_rsa.pub | ssh localhost 'cat >> .ssh/authorized_keys'
```

Now we should be able to do a password less ssh to localhost.



*For running the system in a cluster we should make sure that we can do password less ssh between all machines involved.

GoFFish System Description

Entire goffish platform can be divided into three parts:

GoFS: GoFS is a distributed file system, which contains the graph data. Just like HDFS, GoFS also contains one name-node and other worker-nodes. Name-node contains information about the entire file system, and other worker node refers to name-node for information about other parts of the distributed file system.

Gopher: Gopher is the execution engine for goffish. It takes the subgraph centric program and run it in a distributed fashion over all the hosts. Gopher in turn work on top of another distribution programming framework called floe.

Gopher Client: Gopher Client submits job to gopher along with the information about the graphs it needs to access from GoFS
A rough folder structure of the deployment folder created from step 3 is given below

```
deployment
|
|-goffish_home
| |
| |-gofs-2.0
| |-gofs-data
| |-gopher-server-2.0
|
|-samples
| |
| |-gofs-samples
| | |
| | |-graphs
| |
| |-gopher-jars
|
|-goffish_conf
|
|-gopher-client-2.0
```

Brief description about some of the important folder is given below:

goffish_home : goffish home contains all necessary jar files associated with gofs and gopher. “gofs-2.0” contain necessary script file associated with gofs and gofs-data contain the distributed file system. gopher-server contains jars and script associated with gopher. we can access goffish_home using $GOFFISH_HOME from you shell.

samples: samples folder contain the graph data files and the sample programs to run on gopher. “graphs” folder contains sample graph data files and “gopher-jar” contains the jar for sample programs. Our own graph data files and program jar should also go to these folders.

gopher_client: This folder contains scripts necessary for gopher client. We will use these scripts to start gopher and submit our jobs to gopher.
goffish_conf: This folder contains few json file and one python file. As we can observe that running goffish jobs involves running various script files with different argument. To simplify the process we keep few json file namely

- instace_list.json : Containing information about all host files

- jar_data.json : Containing information about the jar file and class files
- graph_data.json : Containing information about graph data to be loaded

Depending on the data of this json files the python script will make change in all necesseasy script files.

By default the json files are set to run vertex count program on a facebook graph on the localhost. Jars and graph data file is included in the deployment.

Running sample GoFFish job

Here we describe step by step instruction for running the sample program (vertex count) on the sample graph (facebook graph)

To run a goffish job from scratch we need to do the following steps:

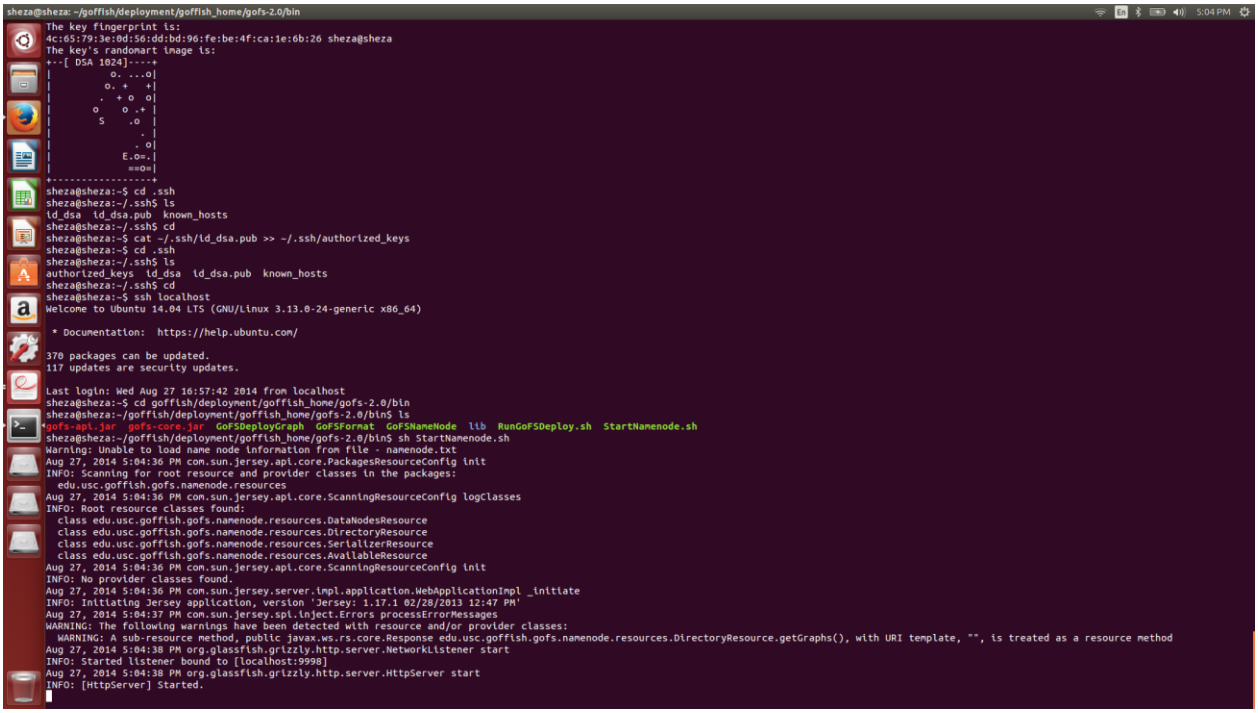
1) Start a namenode server
2) Format the GoFS File System
3) Load the graph in to GoFS
4) Start Gopher engine
5) Submit the gopher job
6) Checking the result

Following steps describes how to perform each step:
Before starting the namenode server give all the necessary permissions to gofs-data folder located inside goffish_home of deployment folder inside goffish folder.
>>cd /goffish/deployment/goffish_home
>>sudo chmod 777 gofs-data
This will do the job.

1.Start a namenode server

Go to ~/goffish/deployment/goffish_home/gofs-2.0/bin

Now run the script “StartNamenode.sh”
It will start the name-node server on the local host.

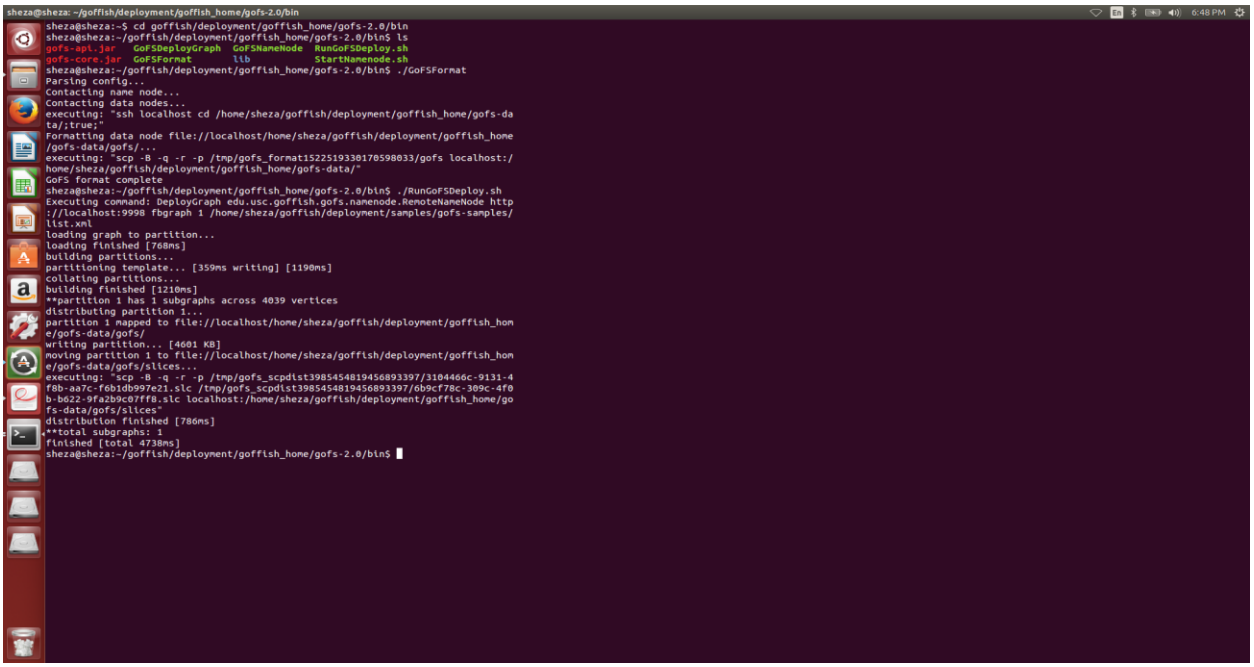


2.Format the GoFS File System

Now open another terminal and In the same folder run “GoFSFormat”. It will format the gofs file system.
>>./GoFSFormat will do the job.

3.Deploying Graph in GoFS

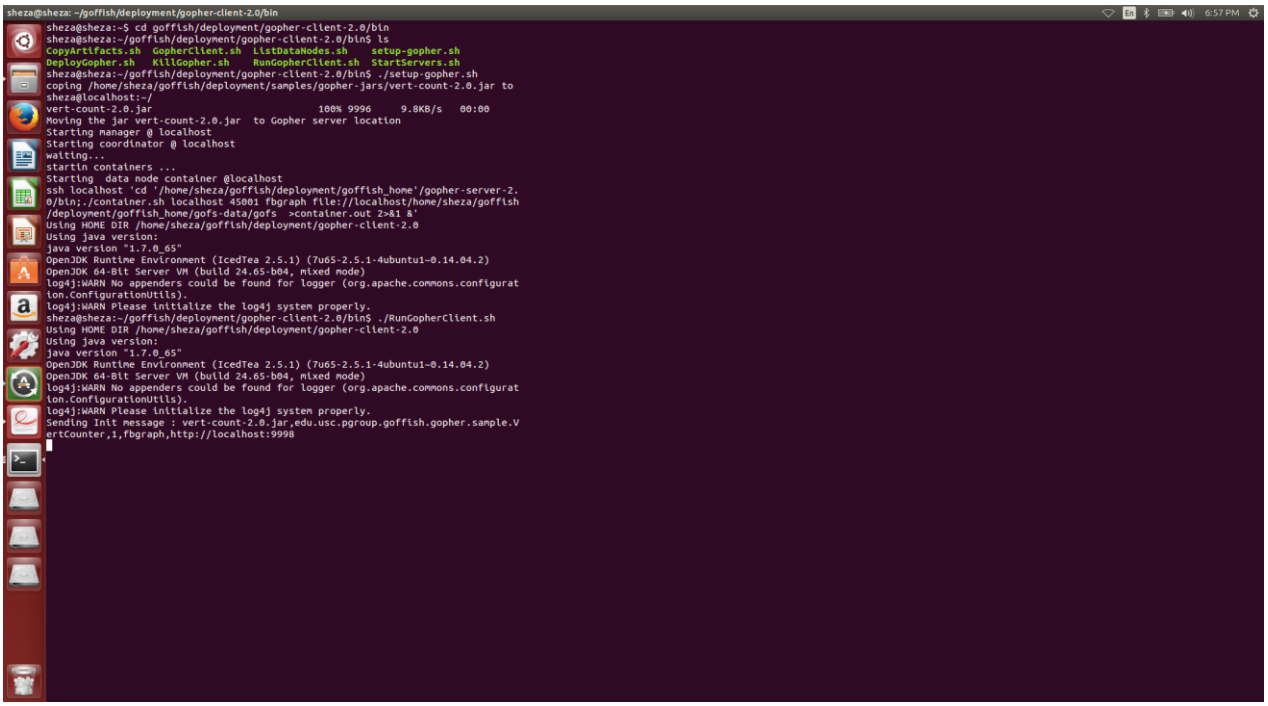
In the same folder run “RunGoFSDeploy.sh”. It will deploy the facebook graph on gofs in the localhost.



4.Start Gopher engine

Now go to ~/goffish/deployment/gopher-client-2.0/bin

Run “setup-gopher.sh”. It will copy the jar file across systems, start gopher server in the namenode.

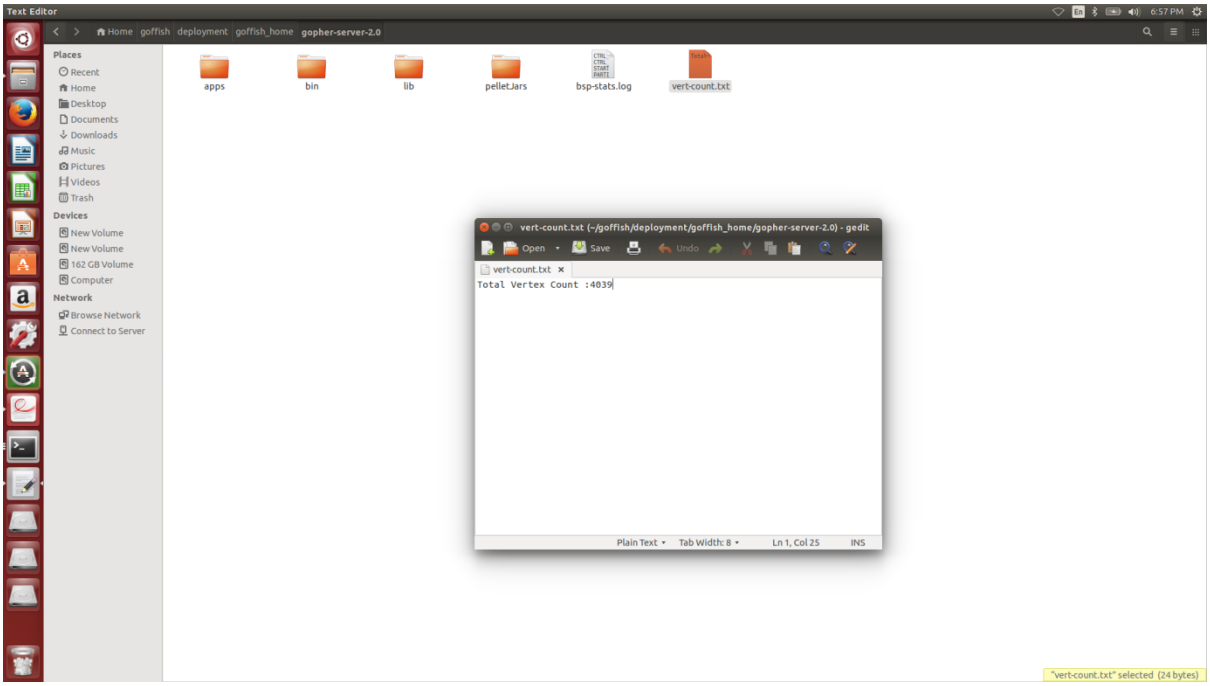


5.Submit the Gopher job

Now in the same folder run “RunGopherClient.sh”. It will submit the vertex count program to gopher for the facebook graph.

6.Checking the result

The above script will keep running.
To check the result go to $GOFFISH_HOME/ gopher-server-2.0
We will see a file “vert-count.txt” having output
Total Vertex Count :4039



If no file is seen go to $GOFFISH_HOME/ gopher-server-2.0/bin

And see the output of container.out. This file actually contains the log of the subgraph centric program. So if any exception occurs then it should be available here.
However if the file “vert-count.txt” is seen then we have successfully run our first goffish job.

As the “RunGopherClient.sh” keep running we can close the script using ^C

This documentation is all about installing and running a sample graph (vertex count program on facebook graph) on GoFFish platform. We can also run other sample graphs on this platform.
GoFFish can also be run in distributed settings.