temporary password for CyStorm is:  NDgxMGJjN2

 one time verification code of: 37247814

Your Google Authenticator code: IHFXQY47HERBBD4K

Now pw: wkelong…….

CprE 419 Lab 1: Using the Cluster, and Introduction to HDFS

**Department of Electrical and Computer Engineering**

**Iowa State University**

**Spring 2015**

# Purpose

# We have setup an environment on the “cystorm” cluster for large-scale data analysis. The first goal of this lab is get familiar with this environment, called “Hadoop”. The second goal is to get familiar with a distributed file system for storing large data sets, called “HDFS”, which is short for “Hadoop Distributed File System”. Specifically, at the end of this lab, you will know how to:

* Login to cystorm cloud with your user id
* Transfer files between the local machine and the cloud
* Use HDFS and the HDFS Programming Interface
* Write, compile and execute a Hadoop program on the Cloud

# Submission

# Create a zip archive with the following and hand it in through blackboard.

* A write-up answering questions for each experiment in the lab. For output, cut and paste results from your terminal and summarize when necessary.
* Commented Code for your program. Include all source files needed for compilation.

# Resources

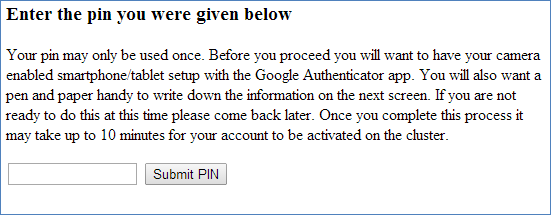
You will need the following before you start the lab. If you don’t have these, please contact the instructors.

* Address of CyStorm cloud resource:  cystorm.its.iastate.edu
* Password to CyStorm. Follow below instructions to get CyStorm password
* Address of Hadoop Master Node: n0
* An email containing a PIN required to create CyStorm access.

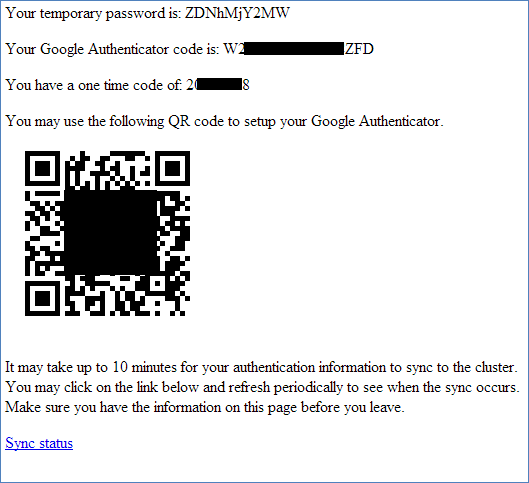
**Logging into cystorm**

In order to set up access in CyStorm, please follow the steps given below (from ISU Network or VPN):

* Go to <https://hpc-ga.its.iastate.edu/cystorm>
* Authenticate with your ISU Net ID / Password
* Enter the PIN given in the received email



* In the next page, you will receive a **temporary password** and a **QR code**.



* Use the QR code to add an account on Google authenticator
* Ssh to cystorm.its.iastate.edu using Putty.
* Use the temporary password and Google authenticator to login
* Change your password after first login using passwd

**You should now be logged into CyStorm and have changed your password.**

# Logging Into the Cloud

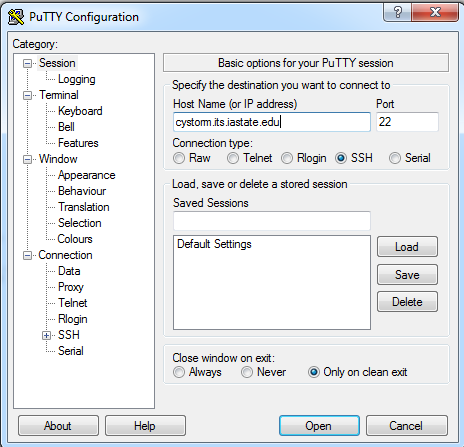
**For Windows:**

You can use Putty to login to the cloud. In case you do not have Putty installed in your system, you can download it from the link below:

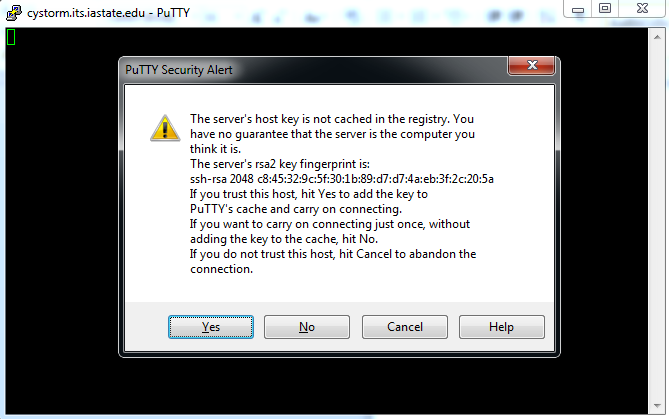
<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Now use Putty.exe to connect to the Cloud as follows (from ISU Network or VPN):

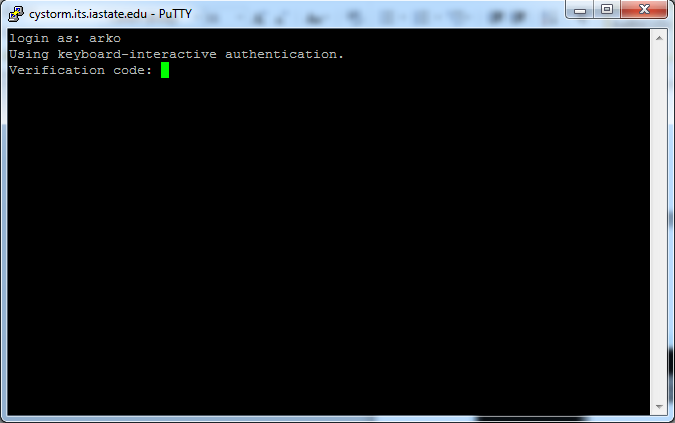
* Start PuTTY and give cystorm.its.iastate.edu as the Host Name, then click Open.



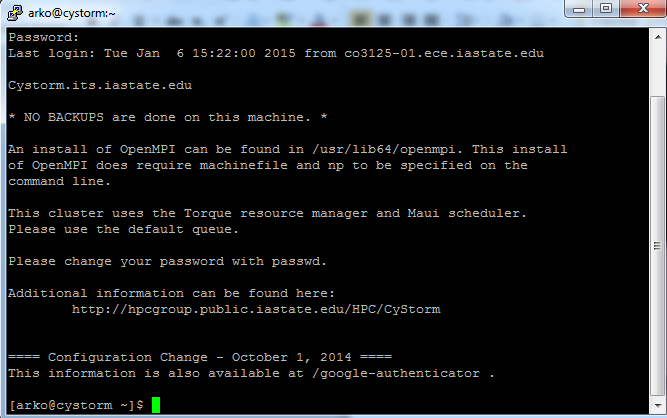
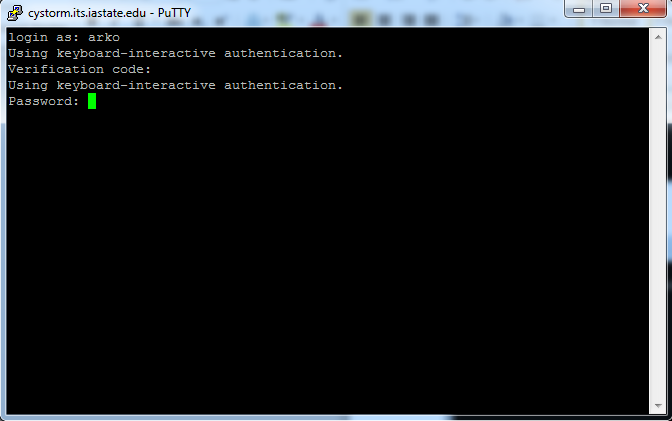
* Click “Yes” to trust the server as shown below



* Provide your ISU Net ID as the User ID and use the Google authenticator app to provide the verification code.



* Give the password and login



* CyStorm is a cluster of computers. We need to access the Hadoop name node on CyStorm. The next step is to ssh into namenode. Type in “ssh n0” to access the Hadoop namenode for CyStorm.



# **HDFS Usage**

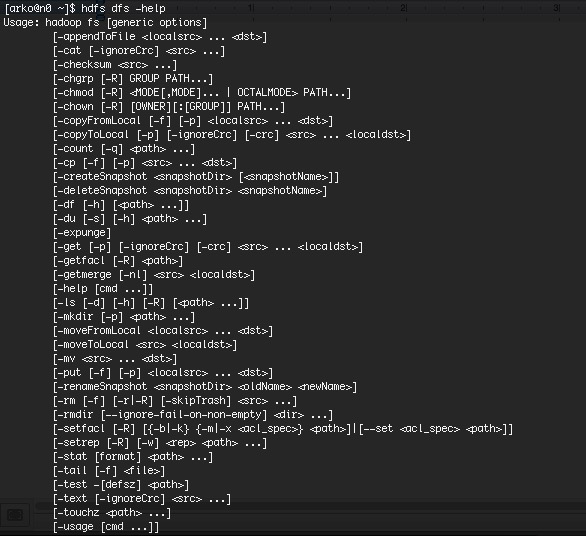
HDFS is a distributed file system that is used by Hadoop to store large files. It automatically splits a file into many “blocks” of no more than 64MB and stores the blocks on separate machines. This helps the system to store huge files, much larger than any single machine can store. The HDFS also creates replicas for each block (by default 3). This helps to make the system more fault-tolerant. HDFS is an integral part of the Hadoop eco-system. HDFS is an Open Source implementation of the Google File System.

First log in to **n0** as described in the above sections.

To try out different HDFS utilities, try the commands:

*$ hdfs dfs –help*

You should see something like the screenshot given below:



You will notice that the HDFS utilities are very similar to that of the default UNIX filesystem utilities and hence should be easy to learn.

# Experiment 1 (10 points):

Create a directory called /scr/<your login id>/lab1 under HDFS. Create a new file called “afile.txt” on your local machine with some text (make me laugh). Then use WinSCP (or ssh on linux) to move the file to your home directory in Cystorm. Use a *hdfs dfs* command to move this file to the directory you created. Show this to the instructor.

# Writing, Compiling, and Executing Programs using Hadoop

Next we will see how to compile a Hadoop program in the Cloud. We will use the Java programming language, as that is the default language used with Hadoop.

**import java.io.\*;**

**import java.lang.\*;**

**import java.util.\*;**

**import java.net.\*;**

**import org.apache.hadoop.fs.\*;**

**import org.apache.hadoop.conf.\*;**

**import org.apache.hadoop.io.\*;**

**import org.apache.hadoop.util.\*;**

**public class HDFSWrite {**

**public static void main ( String [] args ) throws Exception {**

**// The system configuration**

**Configuration conf = new Configuration();**

**// Get an instance of the Filesystem**

**FileSystem fs = FileSystem.get(conf);**

**String path\_name = "/scr/<Your ID>/lab1/newfile";**

**Path path = new Path(path\_name);**

**// The Output Data Stream to write into**

**FSDataOutputStream file = fs.create(path);**

**// Write some data**

**file.writeChars("The first Hadoop Program!");**

**// Close the file and the file system instance**

**file.close();**

**fs.close();**

**}**

**}**

The above Java program uses the HDFS Application Programming Interface (API) provided by Hadoop to write into a HDFS file.

Paste the code into your favorite Java editor and change the following:

String path\_name = *"/scr/<Your ID>/lab1/newfile*"; to reflect your actual User ID.

Note that this is a HDFS path and will not be visible on the local filesystem.

We will transfer this program to the Cloud, compile it and then run it to check the output.

You can also compile the program on your local machine if you have the Hadoop libraries downloaded.

# Transferring Files from/to the Cloud

**For Windows:**

WinSCP is an open source SFTP and FTP client for Microsoft Windows. It can be used to transfer files securely between a remote and a local machine. You can get the WinSCP client from <http://winscp.net/eng/index.php>

To connect to the Hadoop Master node:

* Start WinSCP.
* Type the IP address or host name of the Hadoop Master.
* Click **Login**.
* During the first login, you might get a warning message if the server's key is not cached. Click **Yes** to continue.
* You will need the Google authenticator (verification code) and your password to login

**For UNIX-like systems:**

Issue the commands as described below from your local UNIX-like system to transfer files between the Cloud and your local system.

To transfer a file from your local machine to the Hadoop Master Server:

*$ scp <filename> <user ID>@<server\_IP>:/home/<user ID>/<filename>*

To transfer a file from the server to your local machine:

*$ scp <user ID>@<server\_IP>:/home/<user ID>/<filename> <filename>*

For both cases, you will be prompted for a verification code and password.

**Compiling and Running Hadoop Programs**  
Compile the Java program on the server as follows.*$ mkdir class*

*$ javac -classpath /hadoop/share/hadoop/common/hadoop-common-2.4.1.jar:/hadoop/share/hadoop/common/lib/hadoop-annotations-2.4.1.jar:/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.4.1.jar -d class/ HDFSWrite.java*

*$ jar -cvf HDFSWrite.jar -C class/ .*

*$ rm -rf class*

Ececute the Hadoop program on the server (n0) as follows:

*$ hadoop jar HDFSWrite.jar HDFSWrite*Check the HDFS path *"/scr/<Your ID>/lab1/newfile”* to verify the output of the program.

You should see the file created with the contents as given in the program.

You can also compile the program and generate the jar file on a local machine using Eclipse. For further instructions, see the manual on “Compiling Hadoop With Eclipse” that we have linked from the class website.

**HDFS API**

As seen in the last section, HDFS can also be accessed using a Java API, which allows us to read/write into the file system. For documentation, see:

<http://hadoop.apache.org/docs/r2.4.1/api/>  
Look at Package org.apache.hadoop.fs

The [FSDataOutputStream](http://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/fs/FSDataOutputStream.html)and [FSDataInputStream](http://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/fs/FSDataInputStream.html" \o "class in org.apache.hadoop.fs) can be used to read / write files in HDFS respectively. Check out their class documentations as well.

Read the source code of the program that you just compiled. That would help you to better understand how the HDFS API works. Then, look at the different classes that were used and learn more about the various methods they provide. All the methods can be found in the Hadoop API documentation. Once you have browsed through the HDFS API, go onto the following experiment.

**Experiment 2 (40 points):**

Write a program using the Java HDFS API that reads the contents of the HDFS file “*/class/s15419x/lab1/bigdata*” and computes the 8-bit XOR checksum of all bytes whose offsets range from 5000000000 till 5000000999, both endpoints inclusive. Print out the 8-bit checksum.

Attach the Java code in your submission, as well as the XOR output.

For instance, the XOR checksum of the bitstring “00000000111111110000000011111111” is “00000000”.