**# Procedure for the automatic function of creating cluster(code expalnation)**

*1.* "#!/usr/bin/python" shebang is added in order to make the file executable. /usr/bin/python is the path. 'os ' module of python is also imported to use the 'system' function of this module which enables to run linux commands using python scripts .

-------------------------------------------------------------------------------------------------------------------------------

2.operator and command modules are imported from the python library.These are predefined modules of python.

**##code for the manually created modules(namenode:-)**.

the modues are used as libraries after placing them in /usr/lib64/python2.7 this is the location where python keeps all its libraries and modules.

###################################################################

**#!/usr/bin/python**

**import os**

**def nn(s):**

**cmd=" hostnamectl set-hostname namenode"**

**os.system("sshpass -p redhat ssh -l root {} {}".format(s,cmd))**

**os.system("sshpass -p redhat scp /root/Desktop/final\_work/namenode/core-site.xml {}:/etc/hadoop/core-site.xml".format(s))**

**os.system("sshpass -p redhat scp /root/Desktop/final\_work/namenode/hdfs-site.xml {}:/etc/hadoop/hdfs-site.xml".format(s))**

**form=" hadoop namenode -format"**

**service=" hadoop-daemon.sh start namenode"**

**os.system("sshpass -p redhat ssh -l root {} {}".format(s,form))**

**os.system("sshpass -p redhat ssh -l root {} {}".format(s,service))**

**############################################################################**

----------------------------------------------------------------------------------------------------------------------------

**3**. Manually created modules 'datanode' and 'namenode' are also imported.

**i)** In the manually created modules function is created for both operations to be performed at both data-node and name-node seprately.

-- both the functions accept ip of the node as a string

**ii)** operations include

--'format' feature of python is used to place the strings collected in some variables to a desired location which is specified by "{}". this function supports more than one variables at a time to be placed.

--setting hostname using command 'hostname set-hostname namenode' or 'hostname set-hostname datanode' . this is a linux command . and then by using ssh coomand this command is made to run on the diiferent nodes.

-- 'sshpass -p redhat' this enables us to do the ssh command without manually entering the password .in this case the password for the nodes are 'redhat'.

--'ssh -l root ' this enables us to access the nodes with the permission of root.

-- the whole ssh command is run in python using the 'os' module of python and moreover using the system function, which runs command's of linux via python.

**iii)** replacing the core-site.xml file and the hdfs-site.xml file and starting their respective services using ssh.

--the file to be replaced are created on the main machine and placed on the Desktop (final path == /root/Desktop/final\_work/namemode/core-site.xml and /root/Desktop/final\_work/namemode/hdfs-site.xml) for both namenode and datanode respectively.

-- the procedure is completed by using 'os.system' module of python and 'ssh ' commands as used in the previous cases.

**iv)** the respective daemons ie services are started using the command 'hadoop-daemon.sh start namenode' with the help of 'os ' module and 'ssh ' commands as done in the previous cases.

-------------------------------------------------------------------------------------------------------------------------------

**4.** Further code explanation :-

###################################################################

**c=commands.getstatusoutput("nmap -sP 192.168.56.0-255 -n | grep 'Nmap scan' |awk '{print $5}'")**

**y=open('b.txt',mode='wr')**

**y.write(c[1])**

**y.close()**

**fh=open('b.txt')**

##############################################################################

**i)** the 'getstatusoutput' function of 'commands ' module is used to collect the data in string format .

-- 'nmap -sp 192.168.56.0-255 -n' this linux commands goes in to the network and collects the ip ranging from 192.168.56.0 to 192.168.56.255 .

--'grep ' command is used to collect the desired output from a list of various output.In this case we want to extract output lines starting from 'Nmap scan' from the output generated by the command 'nmap -sp 192.168.56.0-255 -n'.

--'awk ' commad is used to cut a string .'print $5' is the parameter passed to the 'awk ' command specifying the criteria for ranging of the characters to be cut out.Here we need the 5th field , thus $5 is passed .

--'"y=open('b.txt',mode='wr')'" this python command creates a file (if not already created) named 'b.txt' with reading and writing permissions given to it .

--"y,write and y.close" functions are used to write data in 'b.txt' and then close it after writing the data.

--"fh=open('b.txt')" this copies the content of b.txt in fh.

-------------------------------------------------------------------------------------------------------------------------------

**5.** Further code explanation :-

###################################################################

**l=[]**

**for x in fh:**

**if (x=='192.168.56.1\n' or (x=='192.168.56.100\n') or (x=='192.168.56.102')or (x=='192.168.56.101')):**

**pass**

**else:**

**p=x.strip()**

**l.append(p)**

###################################################################

**i)**  a empty dictionary is created named 'l' using 'l=[]'

ii) 'for x in fh' for loop now excess the data in variable 'fh' one by one . Ip's 192.168.56.1 and 192.168.56.100 ,192.168.56.102,192.168.56.101 are ip of the main machine and of thevnetwork which need to be excluded and thus they are passed via 'if' and the remaining ip's are appended in the list 'l' after stripping the strings.

##############################################################################

**6.** Further code explanation:

###################################################################

**md=dict()**

**for x in l:**

**if (x=='192.168.56.1\n' or x=='192.168.56.100\n' or x=='192.168.56.102' or x=='192.168.56.101'):**

**pass**

**else:**

**g=commands.getstatusoutput("sshpass -p redhat ssh -l root "+x+" free -m |awk 'NR==2{print $2}'")**

**d=dict()**

**d[x]=int(g[1])**

**md.update(d)**

**md\_list=sorted(md.items(),key=operator.itemgetter(1))**

##############################################################################

**ii)** an empty dictionary is created using the keyword 'dict' and names as ''md'.

iii)now the data in the list 'l' is accessed . using 'ssh ' commands as used earlier 'free -m ' is executed on the nodes . "free -m " collects the RAM data on the machine . Thus all the free available RAM is collected by this command for all the nodes . 'awk ' selects the second fiels of the output of the 'free -m' and then this integer data is saved in the dictionary which is then further added in the md .

**iii)** 'md\_list ' sorts the items if the list 'md' using the key to sort it .the key contains the RAM(free and available ) in descending order.

-------------------------------------------------------------------------------------------------------------------------------

**7.** further code explanation:-

##################################################################

**namenode.nn(md\_list[0][0])**

**for i in md\_list:**

**if i==(md\_list[0][0]):**

**pass**

**else:**

**print md\_list[q][0]**

**datanode.dn(md\_list[q][0])**

**q=q+1**

**##############################################################################**

**i)**from the final list containing ip's and arranged as per the RAM size in descending order . The first ip having the maximum RAM is sent to the 'nn' function of the 'namenode' module . and all the remaining ip's are made datanode via 'dn' function of the datanode module.This is done by accesing the 'md\_list ' using for loop and by increasing the counter after every iteration.

-------------------------------------------------------------------------------------------------------------------------------