

1 Εγκατάσταση Spark 2.4.4 cluster

Στο κείμενο αυτό περιγράφεται η εγκατάσταση του Spark framework στην υποδομή του okeanos ή σε μια υποδομή virtualization όπως το virtual box. Περιγράφεται επίσης πως μπορούμε να εκτελέσουμε ένα job στο Spark cluster που έχουμε δημιουργήσει. Ο οδηγός αυτός είναι βασισμένος στις οδηγίες που δίνονται στην επίσημη σελίδα του Spark (<https://spark.apache.org/docs/2.4.4/index.html>) και μπορεί να χρησιμοποιηθεί για οποιοδήποτε Ubuntu-based Linux σύστημα.

Για να γίνει η εγκατάσταση του Spark cluster στον okeanos, θα χρειαστεί να εργαστούμε σ' ένα απομακρυσμένο (remote) περιβάλλον. Για να το πετύχουμε αυτό, χρησιμοποιούμε τα προγράμματα ssh και scp αν δουλεύουμε σε Linux ή Mac και putty και winscp σε περίπτωση που δουλεύουμε σε Windows. Για τη χρήση αυτών των προγραμμάτων συμβουλευόμαστε το Google και τα man pages του λειτουργικού συστήματος (π.χ. man ssh).

Η εγκατάσταση περιλαμβάνει τα εξής συστήματα:

- HDFS: Κατανεμημένο filesystem, από το οποίο διαβάζουν και γράφουν τα Spark jobs.
- Spark: Open source framework γενικού σκοπού για επεξεργασία δεδομένων, που περιλαμβάνει υλοποίηση του προγραμματιστικού μοντέλου MapReduce.

Οι υπολογιστές (κόμβοι) ενός Spark cluster μπορεί να έχουν έναν ή παραπάνω από τους παρακάτω ρόλους:

| | |
|---------------------------|---|
| HDFS DataNode: | Οι DataNodes περιέχουν κομμάτια (blocks) από τα αρχεία του HDFS. Αναλαμβάνουν να «σερβίρουν» δεδομένα σε κλήσεις εξωτερικών πελατών. |
| HDFS NameNode: | Πρόκειται για τον κεντρικό κόμβο του HDFS. Ο NameNode περιέχει την πληροφορία με την αντιστοίχιση των blocks των αρχείων με τους αντίστοιχους DataNodes (δηλαδή σε ποιον DataNode βρίσκεται κάθε block). |
| Spark Master node: | Ο master είναι ο κεντρικός κόμβος του Spark που μέσω του ενσωματωμένου cluster manager ελέγχει τους διαθέσιμους πόρους και με βάση αυτούς δρομολογεί και διαχειρίζεται τις κατανεμημένες εφαρμογές που τρέχουν στο cluster. |
| Spark Worker node: | Ο worker είναι μια διεργασία του Spark, η οποία τρέχει σε κάθε κόμβο του cluster και διαχειρίζεται τις προς εκτέλεση διεργασίες των κατανεμημένων εφαρμογών στον κόμβο αυτό. |

1.1 Δημιουργία μηχανημάτων

1.1.1 Μέσω του okeanos

Από το site του okeanos knossos (<https://cyclades.okeanos-knossos.grnet.gr/ui/>) δημιουργούμε 2 εικονικές μηχανές (virtual machines) τις οποίες θα ονομάσουμε spark master και spark slave. Ο master θα παίζει τον ρόλο του HDFS NameNode και Spark Master node. Ο slave θα παίζει τον ρόλο του DataNode και Spark Worker Node. Πηγαίνουμε στις Cyclades και πατάμε “New Machine +”. Για το λειτουργικό σύστημα επιλέγουμε “Ubuntu Server LTS”, και στο επόμενο βήμα για το hardware των virtual machines επιλέγουμε 2 πυρήνες, 2GB RAM και 30GB σκληρό δίσκο. Στο επόμενο βήμα για τα δίκτυα δεν επιλέγουμε κάτι και στη συνέχεια ορίζουμε το όνομα του VM και το δημιουργούμε.

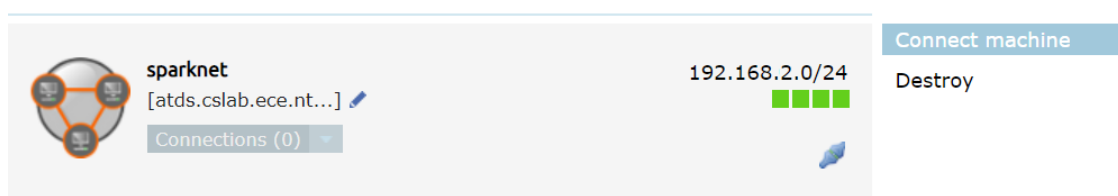
Μετά από την επιτυχημένη δημιουργία των μηχανημάτων ο okeanos επιστρέφει το password για να μπορέσουμε να συνδεθούμε την πρώτη φορά. Φροντίζουμε να κρατήσουμε τους κωδικούς του ‘user’ χρήστη για τα μηχανήματα.

Στη συνέχεια πηγαίνουμε στο tab του site του okeanos με το όνομα «IP» και δημιουργούμε μια καινούρια IPv4 διεύθυνση πατώντας “New IP Address +” και “create IP address”. Αφού δημιουργηθεί, πατάμε “Attach” και επιλέγουμε το spark master μηχανήμα που δημιουργήσαμε για να τη συνδέσουμε στο master.

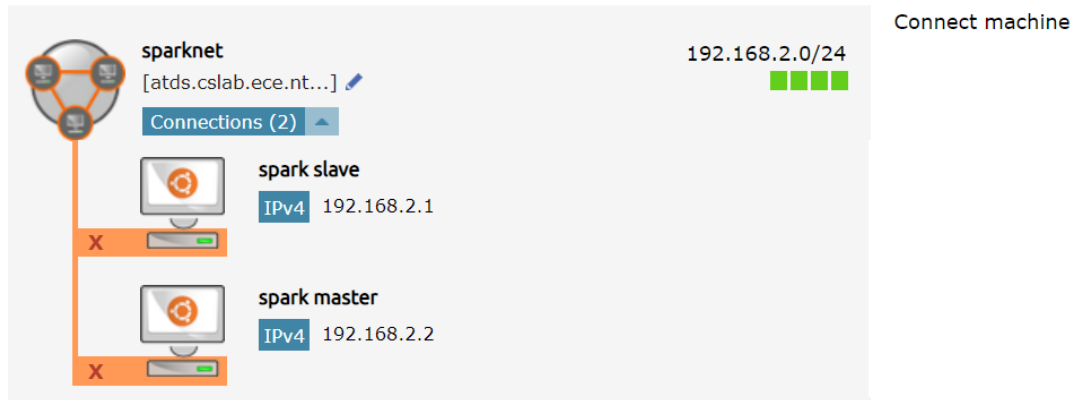


The screenshot displays the Okeanos interface for managing IP addresses. It features two main cards. The top card represents an IP address 83.212.73.176, which is currently 'Available'. It includes a globe icon with 'IP' text, the IP address itself, and a link to the console. To the right, there are three grey squares indicating availability and a blue cloud icon. Action buttons 'Attach' and 'Destroy' are visible on the right side. The bottom card shows the same IP address but with the status 'In use - Running', indicated by four green squares. It also shows details for the 'spark master' machine, including its MAC address 'ce:00:01:eb:26:cb'. A 'Detach' button is present on the right.

Μετά πηγαίνουμε στο tab με τα networks (τριγωνάκι αριστερά από το «IP») και δημιουργούμε ένα καινούριο ιδιωτικό δίκτυο πατώντας “New Network”. Συμπληρώνουμε ένα δικό μας Network name και πατάμε “create network”. Μόλις δημιουργηθεί, πατάμε “Connect machine” και επιλέγουμε και τα δύο VMs που φτιάξαμε για να τα συνδέσουμε στο δίκτυο αυτό. Έτσι θα εμφανίζονται 2 connections κάτω από το όνομα του δικτύου. Με αυτό τον τρόπο καταφέραμε να έχουν και τα δύο εικονικά μηχανήματα private IPv4. Τις private IPv4 τις χρειαζόμαστε γιατί είναι απαραίτητες για τη λειτουργία του Hadoop και του Spark και συγκεκριμένα για την επικοινωνία του master με τους slaves.



The screenshot shows the Okeanos interface for managing networks. A single card is visible for a network named 'sparknet', which has the IP range '192.168.2.0/24'. The card includes a network diagram icon, the network name, the IP range, and a link to the console. Below the network name, it shows 'Connections (0)'. On the right side, there are three green squares indicating the network status and a blue cloud icon. Action buttons 'Connect machine' and 'Destroy' are located on the right edge of the card.



Μετά από αυτή τη διαδικασία μπορούμε να συνδεθούμε και στα δύο VMs στην IPv6 που τους ανατέθηκε αρχικά εφόσον το δίκτυο μας υποστηρίζει IPv6 (όπως του Πολυτεχνείου) ή αν δεν υποστηρίζει μπορούμε να συνδεθούμε στο spark master μηχανήμα μέσω της public IPv4 που του δώσαμε, και στην private IPv4 του slave μέσα από τον master.

1.1.2 Μέσω virtual box

Χρειάζεται να έχουμε το virtualbox που τρέχει εικονικές μηχανές τόσο από Windows όσο και από Linux μηχανήματα. Κατεβάζουμε την σωστή έκδοση για το λειτουργικό μας από εδώ: <https://www.virtualbox.org/wiki/Downloads>

Από τον wizard επιλέγουμε New machine->Linux-Ubuntu. Του δίνουμε 384MB RAM (default) και στην επόμενη οθόνη επιλέγουμε Startup Disk και create new hard disk (8GB), τύπο VDI, και dynamically allocated χώρο. Η παρεχόμενη RAM και ο ελεύθερος χώρος μπορούν αργότερα να αλλάξουν εάν ο χρήστης το επιθυμεί.

Κατόπιν κατεβάζουμε το cd εγκατάστασης netinst iso του Ubuntu.

```
wget http://archive.ubuntu.com/ubuntu/dists/xenial/main/installer-i386/current/images/netboot/mini.iso
```

Κατά την πρώτη εκκίνηση του virtual machine θα ζητηθεί ένα bootable image για να ξεκινήσει το λειτουργικό. Του δίνουμε το iso που κατεβάσαμε από το ubuntu site, και συνεχίζουμε την εγκατάσταση. Στο τέλος θα έχουμε ένα virtual machine με παρόμοιο software με αυτό του okeanos. Το καλό με αυτό το virtual machine είναι ότι μπορούμε να το μεταφέρουμε σε οποιοδήποτε σύστημα τρέχει virtual box (ανεξάρτητα εάν είναι Linux ή Windows).

1.2 Προαιρετικό: ενημέρωση του αρχείου hosts του μηχανήματός μας.

Με αυτό το βήμα δεν χρειάζεται να θυμόμαστε απ' έξω τα IP addresses των μηχανημάτων του cluster που στήσαμε στον okeanos για να συνδεθούμε σε αυτά.

Το αρχείο hosts υπάρχει σε κάθε υπολογιστή, και στην ουσία αποτελεί μια τοπική βάση DNS την οποία συμβουλευεται πρώτη κάθε φορά που προσπαθεί να κάνει resolve ένα όνομα σε διεύθυνση IP. Το αρχείο hosts περιέχει αντιστοιχίσεις ονομάτων σε IPs. Το αρχείο αυτό σε συστήματα Windows βρίσκεται στην τοποθεσία

```
C:\Windows\System32\drivers\etc\hosts
```

Ενώ σε συστήματα Linux είναι στην τοποθεσία

```
/etc/hosts
```

Ανοίγουμε το αρχείο hosts σαν root χρήστης (ή με χρήση sudo) και συμπληρώνουμε στο τέλος:

```
<oceanos_master_public_ipv4>  master  
<oceanos_slave_public_ipv6>    slave
```

Από εδώ και πέρα, ο υπολογιστής μας κάνει resolve τα DNS names 'master' και 'slave' στα IPs των μηχανημάτων του oceanos. Έτσι, δεν χρειάζεται να θυμόμαστε απ'έξω το IP του κάθε μηχανήματος.

1.3 Ενημέρωση του hostname των spark machines

Εφόσον έχουμε εκτελέσει το 1.2, στα επόμενα βήματα του οδηγού μας για να συνδεθούμε π.χ. στο master θα κάνουμε:

```
ssh user@master
```

αντί για:

```
ssh user<oceanos_master_public_ipv4>
```

Σε περίπτωση αδυναμίας σύνδεσης στην IPv6 του slave αντίστοιχα (επειδή χρειάζεται να έχουμε ipv6 δίκτυο), μπορούμε να συνδεθούμε για τα βήματα 1.3 και 1.4 στον slave εκτελώντας μέσα από τον master:

```
ssh user<oceanos_slave_private_ip>
```

Συνδεόμαστε επομένως με ssh στον spark master (δίνουμε σαν κωδικό τον κωδικό που μας έδωσε ο oceanos ή αυτόν που επιλέξαμε κατά την εγκατάσταση του virtual machine):

```
ssh user@master
```

και τρέχουμε:

```
sudo hostname master
```

για να ονομάσουμε το μηχάνημα master και δίνουμε το password του χρήστη user όταν μας μας ζητηθεί. Αν αποσυνδεθούμε και συνδεθούμε ξανά στο μηχάνημα θα δούμε ότι θα έχει αλλάξει το hostname. Κάνουμε το ίδιο και για το μηχάνημα slave αφού συνδεθούμε σε αυτό:

```
ssh user@slave  
sudo hostname slave
```

1.4 Ενημέρωση αρχείου hosts των spark machines

Στο μηχάνημα master:

```
ssh user@master
sudo vim /etc/hosts
```

και στο τέλος του αρχείου βάζουμε τις private IPv4 των μηχανημάτων

```
<okeanos_master_private_ip>    master
<okeanos_slave_private_ip>     slave
```

Κάνουμε το ίδιο και για το μηχάνημα slave:

```
ssh user@slave
sudo vim /etc/hosts
```

και στο τέλος του αρχείου βάζουμε

```
<okeanos_master_private_ip>    master
<okeanos_slave_private_ip>     slave
```

1.5 Ρύθμιση για passwordless ssh

Το Spark αλλά και το HDFS για να λειτουργήσουν θα πρέπει οι υπολογιστές που αποτελούν το Spark ή το HDFS cluster να μπορούν να συνδέονται μεταξύ τους με την χρήση ssh χωρίς κωδικό. Αυτό επιτυγχάνεται με την χρήση ζεύγους ιδιωτικού/δημόσιου κλειδιού¹. Η βασική ιδέα της τεχνικής αυτής είναι ότι ο κάθε χρήστης μπορεί να έχει ένα ιδιωτικό κλειδί (αρχείο) το οποίο έρχεται ζεύγος με ένα δημόσιο κλειδί. Ο χρήστης τοποθετεί το δημόσιο κλειδί στους υπολογιστές τους οποίους θέλει να έχει πρόσβαση, και με την χρήση του ιδιωτικού μπορεί να συνδεθεί σε αυτούς χωρίς κωδικό. Η πιστοποίηση του χρήστη γίνεται καθώς μόνο ο χρήστης έχει στην κατοχή του το ιδιωτικό κλειδί. Στο μηχάνημα master τρέχουμε

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

Η εντολή δημιουργεί ένα ιδιωτικό κλειδί id_rsa και ένα δημόσιο κλειδί id_rsa.pub. Στη συνέχεια τρέχουμε

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

η παραπάνω εντολή βάζει το δημόσιο κλειδί id_rsa.pub στον κατάλογο με τα αποδεκτά δημόσια κλειδιά του χρήστη user του master μηχανήματος. Με αυτόν τον τρόπο, όποιος έχει στην κατοχή του το ιδιωτικό κλειδί id_rsa μπορεί να συνδεθεί στο user@master.

Τώρα, αρκεί να μεταφέρουμε από το μηχάνημα master τον κατάλογο .ssh στον χρήστη user του μηχανήματος slave

```
scp -r .ssh/ user@slave:~/
```

Στη εντολή αυτή θα χρειαστεί να βάλουμε το password του μηχανήματος slave.

¹ http://en.wikipedia.org/wiki/Public-key_cryptography

Μετά από αυτή τη διαδικασία μπορούμε πλέον να κάνουμε εύκολα ssh από το μηχάνημα master στο slave με την εντολή `ssh user@slave` και το ανάποδο χωρίς password.

1.6 Δημιουργία NAT

Από τα δύο VMs που φτιάξαμε και τα δύο έχουν private IPv4, αλλά μόνο ο master έχει public IPv4. Αυτό σημαίνει ότι στο ισχύον setup μόνο ο master μπορεί να βλέπει σε όλο το δίκτυο του «έξω κόσμου». Επειδή θα χρειαστεί να εγκαταστήσουμε διάφορα πακέτα και στον slave, θέλουμε να δώσουμε και σε αυτόν πλήρη πρόσβαση στο εξωτερικό δίκτυο.

Για να το πετύχουμε αυτό δημιουργούμε και τρέχουμε στον master το παρακάτω script:

```
#!/bin/bash
echo "Enabling ipv4 forwarding (cleaning old rules)"
# flushing old rules -- USE WITH CARE
iptables --flush
iptables --table nat --flush
# MASQUERADE each request form the inside to the outer world
iptables -t nat -A POSTROUTING -j MASQUERADE
# enable IPv4 packet forwarding in the kernel
echo 1 > /proc/sys/net/ipv4/ip_forward
echo "Master is now operating as router"
```

και στον slave αντίστοιχα:

```
#!/bin/bash
ENDPOINT_INTERFACE=$(cat /etc/hosts | grep master | awk '{print $1}')
route add default gw $ENDPOINT_INTERFACE
echo "Gateway now points to $ENDPOINT_INTERFACE"
```

Για τη δημιουργία και την εκτέλεση των scripts χρησιμοποιούμε της εξής εντολές:

```
vim nat.sh
chmod +x nat.sh
sudo ./nat.sh
```

1.7 Εγκατάσταση Java 8

Για να εγκαταστήσουμε την OpenJDK Java 8, πρέπει πρώτα να ενημερώσουμε τα apt repositories των πακέτων.

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk
```

Μόλις ενημερώσουμε τα repositories κι εγκαταστήσουμε την java (πατάμε yes σε ότι ζητηθεί), ελέγχουμε αν η εγκατάσταση ολοκληρώθηκε με επιτυχία.

```
user@master:~$ java -version
openjdk version "1.8.0_222"
OpenJDK Runtime Environment (build 1.8.0_222-8u222-b10-
lubuntu1~16.04.1-b10)
OpenJDK 64-Bit Server VM (build 25.222-b10, mixed mode)
```

Η παραπάνω διαδικασία πρέπει να ακολουθηθεί και για το slave μηχάνημα.

1.8 Εγκατάσταση HDFS

Τα παρακάτω βήματα πρέπει να γίνουν σε όλους τους υπολογιστές του cluster.

Κατεβάζουμε το installation package από ένα repository και το αποσυμπιέζουμε στο home folder μας.

```
cd ~
wget http://www-eu.apache.org/dist/hadoop/common/hadoop-2.7.7/hadoop-2.7.7.tar.gz
tar -xzf hadoop-2.7.7.tar.gz
```

Κάνουμε edit to ~/.bashrc ώστε να κάνουμε export τις παρακάτω μεταβλητές περιβάλλοντος. Προσθέτουμε τις παρακάτω γραμμές στο τέλος του bashrc.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_INSTALL=/home/user/hadoop-2.7.7
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export HADOOP_CONF_DIR=$HADOOP_INSTALL/etc/hadoop
```

Το configuration του Hadoop βρίσκεται στο path: \$HADOOP_INSTALL/etc/hadoop. Στον κατάλογο αυτό θα ρυθμίσουμε τα εξής αρχεία:

- core-site.xml
- hadoop-env.sh
- hdfs-site.xml: Ρυθμίσεις για την παραμετροποίηση του HDFS
- slaves: τα host names των κόμβων που λειτουργούν ως datanodes

Περισσότερες λεπτομέρειες για τα αρχεία παραμετροποίησης μπορούν να βρεθούν στη διεύθυνση: <http://hadoop.apache.org/docs/r2.7.7/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html>

Κάνουμε edit τα αρχεία(vim <file-name>) και προσθέτουμε τα εξής configuration properties:

core-site.xml

```
...
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://master:9000</value>
  </property>
</configuration>
```

hdfs-site.xml

```
...
<!-- Put site-specific property overrides in this file. -->
<configuration>
  <property>
    <name>dfs.replication</name>
```

```

    <value>2</value>
    <description>Default block replication.</description>
  </property>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/user/hdfsname</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>/home/user/hdfsdata</value>
  </property>
  <property>
    <name>dfs.blocksize</name>
    <value>64m</value>
    <description>Block size</description>
  </property>
  <property>
    <name>dfs.webhdfs.enabled</name>
    <value>true</value>
  </property>
  <property>
    <name>dfs.support.append</name>
    <value>true</value>
  </property>
</configuration>

```

slaves: αφαιρούμε το localhost και βάζουμε

```
slave
```

hadoop-env.sh: τροποποιούμε το JAVA_HOME

```

...
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
...

```

1.9 Εγκατάσταση Spark

Τα παρακάτω βήματα πρέπει να γίνουν σε όλους τους υπολογιστές του cluster.

Κατεβάζουμε το installation package από ένα repository και το αποσυμπιέζουμε στο home folder μας.

```

cd ~
wget https://archive.apache.org/dist/spark/spark-2.4.4/spark-2.4.4-bin-hadoop2.7.tgz
tar -xzf spark-2.4.4-bin-hadoop2.7.tgz

```

Κάνουμε edit to ~/.bashrc ώστε να κάνουμε export τις παρακάτω μεταβλητές περιβάλλοντος. Προσθέτουμε τις παρακάτω γραμμές στο τέλος του bashrc.

```

export SPARK_HOME=/home/user/spark-2.4.4-bin-hadoop2.7
export PATH=$PATH:$SPARK_HOME/bin
alias start-all.sh='$SPARK_HOME/sbin/start-all.sh'
alias stop-all.sh='$SPARK_HOME/sbin/stop-all.sh'

```


Το configuration του Spark βρίσκεται στο path: \$SPARK_HOME/conf. Στον κατάλογο αυτό υπάρχουν κάποια αρχεία με την κατάληξη '.template'. Εμείς θα δημιουργήσουμε (σαν αντίγραφα των αντίστοιχων .template) και στη συνέχεια θα ρυθμίσουμε τα εξής αρχεία:

- spark-env.sh
- spark-defaults.conf: Ρυθμίσεις για την παραμετροποίηση του Spark
- slaves: τα host names των κόμβων που λειτουργούν ως compute slaves

Περισσότερες λεπτομέρειες για τα αρχεία παραμετροποίησης μπορούν να βρεθούν στη διεύθυνση: <https://spark.apache.org/docs/2.4.4/spark-standalone.html#installing-spark-standalone-to-a-cluster>

Κάνουμε edit τα αρχεία (vim <file-name>) και προσθέτουμε τα εξής configuration properties:

spark-env.sh

```
...  
SPARK_WORKER_CORES=2  
SPARK_WORKER_MEMORY=1g
```

spark-defaults.conf

```
...  
spark.master                spark://master:7077  
spark.submit.deployMode     client  
spark.executor.instances    2  
spark.executor.cores        1  
spark.executor.memory        512m  
spark.driver.memory         512m
```

slaves: αφαιρούμε το localhost και βάζουμε

```
slave
```

1.10 Εκκίνηση του HDFS

Το πρώτο πράγμα που έχουμε να κάνουμε είναι format του HDFS. Συνδεόμαστε στον master και δίνουμε την παρακάτω εντολή:

```
ssh user@master  
hdfs namenode -format
```

Format κάνουμε την πρώτη φορά που εγκαθιστούμε το HDFS.

TIP: Σε περίπτωση που εμφανιστεί το εξής error:

```
user@master:/opt# hdfs namenode -format  
-bash: hdfs: command not found
```

σημαίνει ότι το σύστημα δεν έχει 'δει' ακόμα τις αλλαγές που κάναμε στο .bashrc αρχείο οπότε εκτελούμε

```
source ~/.bashrc
```

Στη συνέχεια ξεκινάμε το HDFS (NameNode + DataNodes) δίνοντας από τον master την εξής εντολή:

```
start-dfs.sh
```

Παίρνουμε το παρακάτω output:

```
user@master:~# start-dfs.sh
Starting namenodes on [master]
The authenticity of host 'master (83.212.104.253)' can't be
established.
ECDSA key fingerprint is
03:f4:58:65:d0:ca:76:b6:53:50:bf:1f:cd:d9:2a:2b.
Are you sure you want to continue connecting (yes/no)? yes
master: Warning: Permanently added 'master,83.212.104.253' (ECDSA) to
the list of known hosts.
master: starting namenode, logging to /home/user/hadoop-
2.7.7/logs/hadoop-root-namenode-master.out
slave: starting datanode, logging to /home/user/hadoop-
2.7.7/logs/hadoop-root-datanode-slave.out
Starting secondary namenodes [0.0.0.0]
The authenticity of host '0.0.0.0 (0.0.0.0)' can't be established.
ECDSA key fingerprint is
03:f4:58:65:d0:ca:76:b6:53:50:bf:1f:cd:d9:2a:2b.
Are you sure you want to continue connecting (yes/no)? yes
0.0.0.0: Warning: Permanently added '0.0.0.0' (ECDSA) to the list of
known hosts.
0.0.0.0: starting secondarynamenode, logging to /home/user/hadoop-
2.7.7/logs/hadoop-root-secondarynamenode-master.out
```

Την πρώτη φορά πρέπει να πατήσουμε “γ” κατά την σύνδεση του master με τον slave, καθώς ο master ξεκινάει τον slave μέσω ssh και δεν τον γνωρίζει.

Ελέγχουμε να δούμε αν τρέχει το HDFS στον master:

```
user@master:~# jps
20024 NameNode
20310 Jps
20187 SecondaryNameNode
```

Και στον slave:

```
user@slave:~# jps
15458 DataNode
15557 Jps
```

Το σύστημα έχει ξεκινήσει και στα παρακάτω url μπορεί κανείς να βλέπει την κατάσταση του HDFS:

http://<okeanos_master_public_ipv4>:50070 ή <http://master:50070>: Το url αυτό δείχνει την κατάσταση του NameNode, καθώς και τα περιεχόμενα του αποθηκευτικού συστήματος.

1.11 Εκκίνηση Spark

Για να ξεκινήσει στη συνέχεια το Spark (master+workers) εκτελούμε από τον master την εντολή:

```
start-all.sh
```

Παίρνουμε το παρακάτω output:

```
user@master:~# /home/user/spark-2.4.4-bin-hadoop2.7/sbin/start-all.sh
starting org.apache.spark.deploy.master.Master, logging to
/home/user/spark-2.4.4-bin-hadoop2.7/logs/spark-user-
org.apache.spark.deploy.master.Master-1-master.out
slave: starting org.apache.spark.deploy.worker.Worker, logging to
/home/user/spark-2.4.4-bin-hadoop2.7/logs/spark-user-
org.apache.spark.deploy.worker.Worker-1-slave.out
```

Ελέγχουμε να δούμε αν τρέχει το Spark στον master:

```
user@master:~# jps
20024 NameNode
20310 Jps
20187 SecondaryNameNode
17556 Master
```

Και στον slave:

```
user@slave:~# jps
15458 DataNode
15557 Jps
17551 Worker
```

Το σύστημα έχει ξεκινήσει και στα παρακάτω url μπορεί κανείς να βλέπει την κατάσταση του Spark cluster:

http://<oceanos_master_public_ipv4>:8080 ή <http://master:8080>: Το url αυτό δείχνει την κατάσταση του cluster, καθώς και την κατάσταση και το history των jobs.

1.12 Χρήσιμες εντολές, αρχεία:

- Όπως είδαμε, με την εντολή jps μπορούμε να δούμε τις java διεργασίες που τρέχουν σ' ένα μηχάνημα. Κάνοντας kill -9 <pid> σκοτώνουμε κάποια διεργασία που δεν αποκρίνεται.
- Ο κατάλογος logs/ μέσα στους φακέλους εγκατάστασης περιέχει τα logfiles των υπηρεσιών του HDFS και του Spark. Είναι χρήσιμος καθώς εκεί μπορούμε να βλέπουμε πιθανά λάθη/προβλήματα που προκύπτουν.
- Στον κατάλογο tmp υπάρχουν τα lock files των προγραμμάτων του HDFS (με κατάληξη .pid). Καλό είναι, εάν έχουμε τερματίσει τον HDFS cluster με βίαιο τρόπο (πχ με kill -9 <pid>), να τα σβήσουμε.

2 Εκτέλεση παραδειγμάτων

Μετά την ολοκλήρωση της εγκατάστασης μπορούμε να τρέξουμε από τον master ένα έτοιμο παράδειγμα του spark υλοποιημένο σε java που υπολογίζει το π.

```
user@master:~# spark-submit --class
org.apache.spark.examples.JavaSparkPi /home/user/spark-2.4.4-bin-
hadoop2.7/examples/jars/spark-examples_2.11-2.4.4.jar
...
Pi is roughly 3.1449
...
```

Στο φάκελο \$SPARK_HOME/examples υπάρχουν γενικά πολλά παραδείγματα σε python και scala εκτός από java.

Ανάλογα με τη γλώσσα που θα επιλέξουμε να υλοποιήσουμε την άσκηση θα χρειαστεί να εγκατασταθεί και στα δύο μηχανήματα η python ή η scala με την εντολή “sudo apt-get install python/scala” (η java υπάρχει ήδη). Συνίσταται το development να γίνεται στους υπολογιστές σας και όταν θέλετε να τρέξετε τον κώδικα να μεταφέρετε το binary αρχείο στο master μηχανήμα. Με χρήση του scp ή του WinScp (ανάλογα με το λειτουργικό μας σύστημα), μεταφέρουμε το αρχείο αυτό στον master του cluster που έχουμε στήσει στον okeanos.

Στη περίπτωση της python μεταφέρουμε αυτούσιο το .py αρχείο με τον κώδικά μας και τον τρέχουμε:

```
me@localpc:~# scp mycode.py user@master:~/
ssh user@master
user@master:~# spark-submit mycode.py
```

Στην περίπτωση της java ή της scala κάνουμε compile των κώδικα στον υπολογιστή μας και μόλις ολοκληρωθεί το compile, στον φάκελο target του project που έχουμε δημιουργήσει έχει δημιουργηθεί ένα jar file. Αυτό το jar file είναι το εκτελέσιμό μας που θα μεταφέρουμε στον master και θα το τρέξουμε με την εντολή:

```
user@master:~# spark-submit mycode.jar
```

Hint: Σε όλες τις περιπτώσεις το spark ψάχνει μέσα στον κώδικα την main class για να την εκτελέσει. Εάν η κύρια κλάση σας λέγεται αλλιώς δε θα μπορέσει να εκτελεστεί και θα επιστρέψει error το spark. Σε αυτή την περίπτωση περνάμε επιπλέον σαν όρισμα με το “--class” το path της κλάσης που θέλουμε να εκτελεστεί:

```
user@master:~# spark-submit --class
org.apache.spark.examples.JavaSparkPi /home/user/spark-2.4.4-bin-
hadoop2.7/examples/jars/spark-examples_2.11-2.4.4.jar
```