

POC 1 :

From data integration to Big data distributed processing and storage

Rapport technique

Refka MEJRI
Wouroud GUEDDICH
Ibrahim ROUIS
Abdallah HAMZA

I- Setting up Ambari

1- Create instances

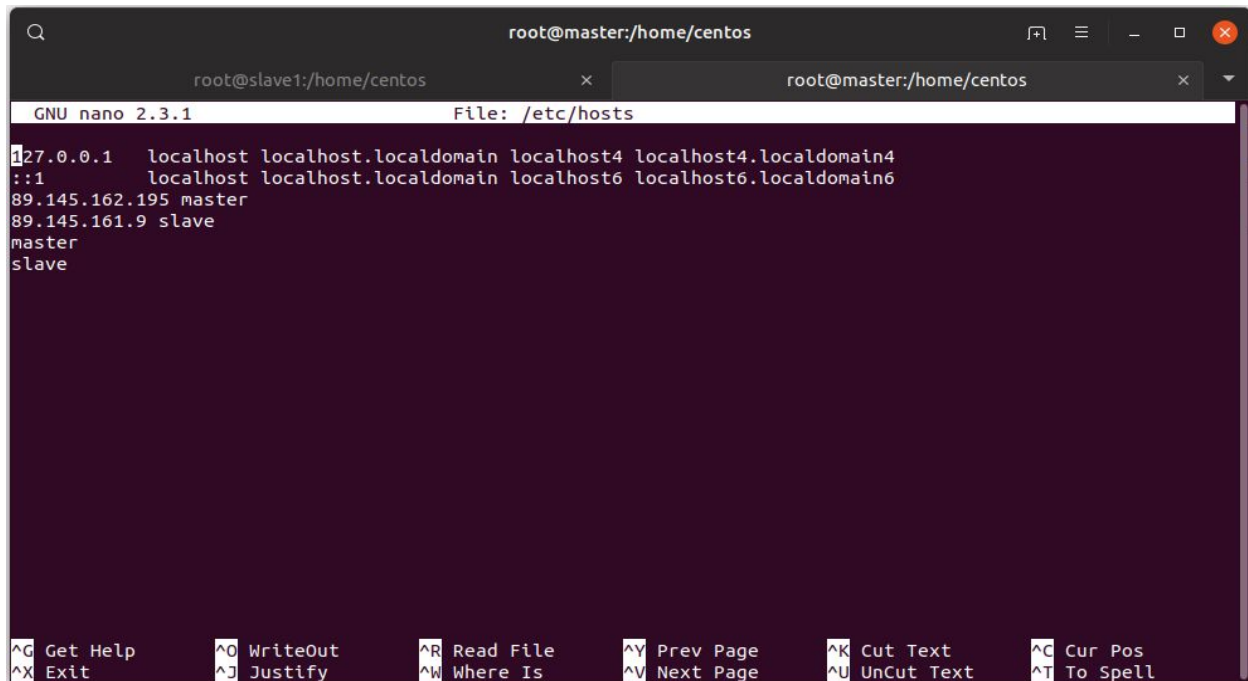
Instances

[ADD](#)

START STOP REBOOT DESTROY				
<input type="checkbox"/> Instance Display Name ▾	Security Group	IP Address ▴	Status ▴	
OS Template ▴ / Disk	Instance Type ▴	Zone ▴		
<input type="checkbox"/> master Linux CentOS 7.6 64-bit / 51 GB	default Medium	89.145.162.195 DE-FRA-1	RUNNING	▶ ■ ↺ ✕
<input type="checkbox"/> slave1 Linux CentOS 7.6 64-bit / 50 GB	default Medium	89.145.161.9 DE-FRA-1	RUNNING	▶ ■ ↺ ✕
<input type="checkbox"/> slave2 Linux CentOS 7.6 64-bit / 87 GB	default Medium	89.145.162.234 DE-FRA-1	STOPPED	▶ ■ ↺ ✕

```
centos@master:~  
centos@slave1:~  
centos@89.145.162.195's password:  
[centos@master ~]$ ssh-keygen  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/centos/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/centos/.ssh/id_rsa.  
Your public key has been saved in /home/centos/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:h/VFL0afCjl1v0iXXKtea2v6QP6yauTDLDYwiAknuHY centos@master  
The key's randomart image is:  
+---[RSA 2048]---+  
|                 .+. |  
|                oo+. |  
|.               .+ =.+ |  
|.o .           o .o+.o. |  
|. + o . S . oo. + |  
|.. E . o . .o. = |  
|. .           o = oo. |  
|                + * o+o. |  
|                . +.++B=. |  
+-----[SHA256]-----+  
[centos@master ~]$
```

- 2- SSH key generation
\$ ssh-keygen
- 3- Edit Hosts file
\$ nano /etc/hosts

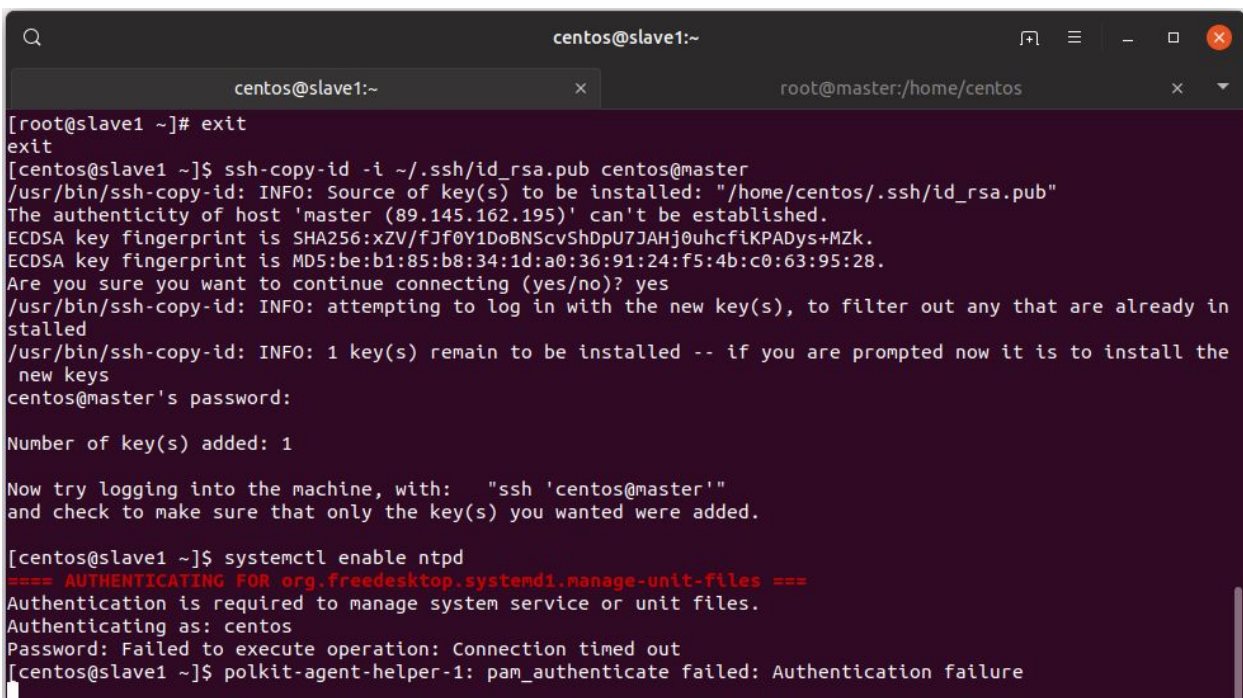


The screenshot shows a terminal window with two tabs. The active tab is titled 'root@master:/home/centos' and displays the GNU nano 2.3.1 editor editing the file /etc/hosts. The file content is as follows:

```
127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
89.145.162.195 master
89.145.161.9 slave
master
slave
```

The bottom status bar of the nano editor shows various keyboard shortcuts: ^G Get Help, ^O WriteOut, ^R Read File, ^Y Prev Page, ^K Cut Text, ^C Cur Pos, ^X Exit, ^J Justify, ^W Where Is, ^V Next Page, ^U UnCut Text, and ^T To Spell.

- 4- Copy SSH key in different hosts
\$ ssh-copy-id -i ~/.ssh/id_rsa.pub centos@master



The screenshot shows a terminal window with two tabs. The active tab is titled 'centos@slave1:~' and displays the following output:

```
[root@slave1 ~]# exit
exit
[centos@slave1 ~]$ ssh-copy-id -i ~/.ssh/id_rsa.pub centos@master
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/centos/.ssh/id_rsa.pub"
The authenticity of host 'master (89.145.162.195)' can't be established.
ECDSA key fingerprint is SHA256:xZV/fJf0Y1DoBNScvShDpU7JAHj0uhcfIKPADys+MZk.
ECDSA key fingerprint is MD5:be:b1:85:b8:34:1d:a0:36:91:24:f5:4b:c0:63:95:28.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already in
stalled
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
new keys
centos@master's password:

Number of key(s) added: 1

Now try logging into the machine, with:  "ssh 'centos@master'"
and check to make sure that only the key(s) you wanted were added.

[centos@slave1 ~]$ systemctl enable ntpd
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-unit-files ====
Authentication is required to manage system service or unit files.
Authenticating as: centos
Password: Failed to execute operation: Connection timed out
[centos@slave1 ~]$ polkit-agent-helper-1: pam_authenticate failed: Authentication failure
```

5- Disable firewall and security configuration

**** Configuring iptables

\$ systemctl disable firewalld

\$ service firewalld stop

***** Disable SELinux

\$ setenforce 0

```
root@slave1:/home/centos
centos@slave1:~ x root@master:/home/centos x root@slave1:/home/centos x
Installed:
ntp.x86_64 0:4.2.6p5-29.el7.centos

Dependency Installed:
autogen-libopts.x86_64 0:5.18-5.el7 ntpdate.x86_64 0:4.2.6p5-29.el7.centos

Complete!
[root@slave1 centos]# systemctl enable ntpd
Created symlink from /etc/systemd/system/multi-user.target.wants/ntpd.service to /usr/lib/systemd/system/ntpd.service.
[root@slave1 centos]# nano /etc/sysconfig/network
[root@slave1 centos]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@slave1 centos]# service firewalld stop
Redirecting to /bin/systemctl stop firewalld.service
[root@slave1 centos]# setenforce 0
[root@slave1 centos]#
```

6- Install ambari agent in all the hosts

\$ wget -nv

<http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.7.3.0/ambari.repo> -O

/etc/yum.repos.d/ambari.repo

\$ yum install ambari-agent

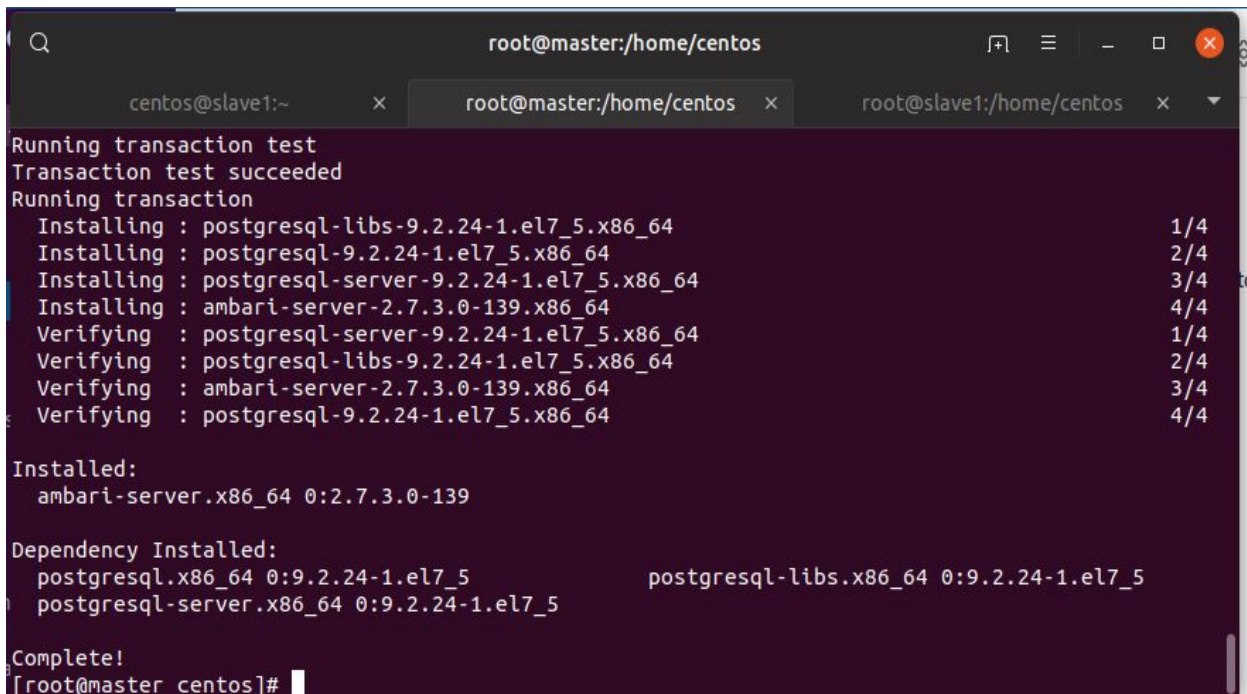
```
root@master:/home/centos
centos@slave1:~ x root@master:/home/centos x root@slave1:/home/centos x
ambari-agent-2.7.3.0-139.x86_64.rpm | 36 MB 00:00:01
Retrieving key from http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.7.3.0/RPM-GPG-KEY-RPM-GPG-KEY-Jenkins
Importing GPG key 0x07513CAD:
  Userid : "Jenkins (HDP Builds) <jenkins@hortonworks.com>"
  Fingerprint: df52 ed4f 7a3a 5882 c099 4c66 b973 3a7a 0751 3cad
  From : http://public-repo-1.hortonworks.com/ambari/centos7/2.x/updates/2.7.3.0/RPM-GPG-KEY-RPM-GPG-KEY-Jenkins
Is this ok [y/N]: y
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : ambari-agent-2.7.3.0-139.x86_64 1/1
  Verifying : ambari-agent-2.7.3.0-139.x86_64 1/1

Installed:
ambari-agent.x86_64 0:2.7.3.0-139

Complete!
[root@master centos]#
```


7- Install Ambari server

\$ yum install ambari-server



```
root@master:/home/centos
Running transaction test
Transaction test succeeded
Running transaction
  Installing : postgresql-libs-9.2.24-1.el7_5.x86_64      1/4
  Installing : postgresql-9.2.24-1.el7_5.x86_64         2/4
  Installing : postgresql-server-9.2.24-1.el7_5.x86_64   3/4
  Installing : ambari-server-2.7.3.0-139.x86_64         4/4
  Verifying  : postgresql-server-9.2.24-1.el7_5.x86_64   1/4
  Verifying  : postgresql-libs-9.2.24-1.el7_5.x86_64     2/4
  Verifying  : ambari-server-2.7.3.0-139.x86_64         3/4
  Verifying  : postgresql-9.2.24-1.el7_5.x86_64         4/4

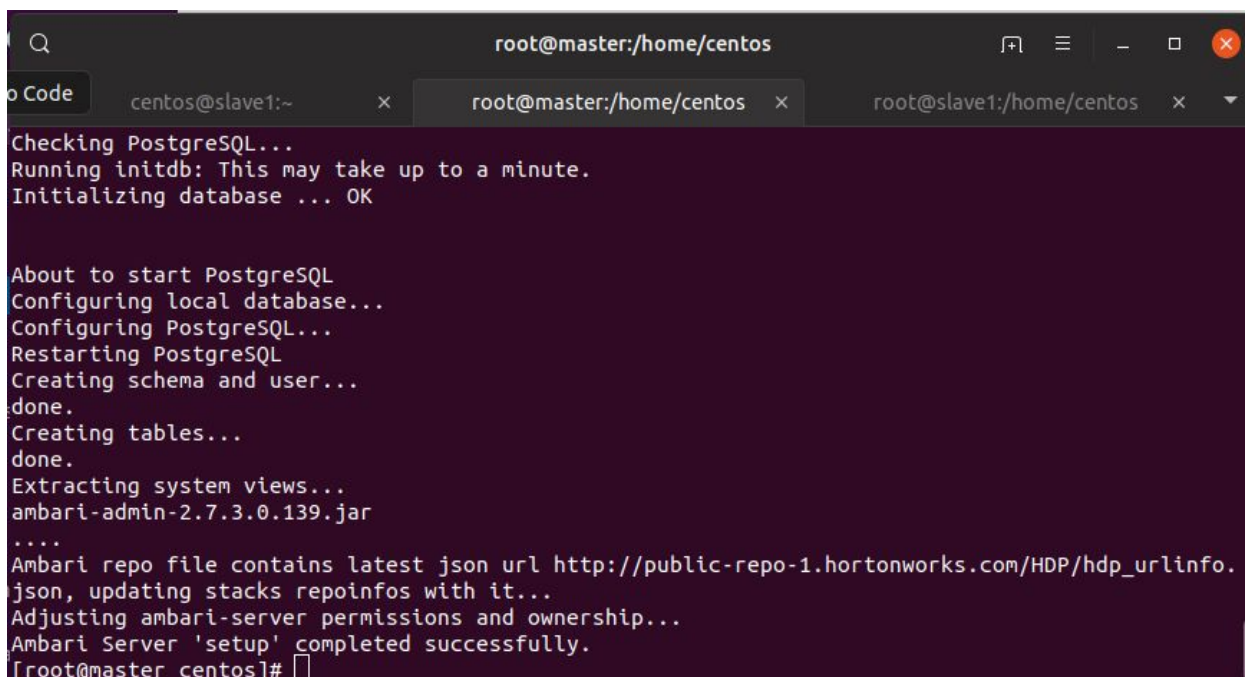
Installed:
  ambari-server.x86_64 0:2.7.3.0-139

Dependency Installed:
  postgresql.x86_64 0:9.2.24-1.el7_5      postgresql-libs.x86_64 0:9.2.24-1.el7_5
  postgresql-server.x86_64 0:9.2.24-1.el7_5

Complete!
[root@master centos]#
```

8- Set up Ambari server

\$ ambari-server setup



```
root@master:/home/centos
Checking PostgreSQL...
Running initdb: This may take up to a minute.
Initializing database ... OK

About to start PostgreSQL
Configuring local database...
Configuring PostgreSQL...
Restarting PostgreSQL
Creating schema and user...
done.
Creating tables...
done.
Extracting system views...
ambari-admin-2.7.3.0.139.jar
....
Ambari repo file contains latest json url http://public-repo-1.hortonworks.com/HDP/hdp_urlinfo.
json, updating stacks repoinfos with it...
Adjusting ambari-server permissions and ownership...
Ambari Server 'setup' completed successfully.
[root@master centos]#
```

9- Edit ambari-agent config file

```
$ nano /etc/ambari-agent/conf/ambari-agent.ini
```

```
*****  
  
[server]  
  
hostname=<ambari.server.hostname: master>  
  
url_port=8440  
  
secured_url_port=8441  
*****
```

10- start ambari server and agent

```
$ ambari-agent start
```

```
$ ambari-server start
```

11- Enter to web ambari server web interface

Master:8080

Set cluster name

Installer

admin

0 Get Started

1 Select Version

2 Install Options

3 Confirm Hosts

4 Choose Services

5 Assign Masters

6 Assign Slaves and Clients

7 Customize Services

8 Review

9 Install, Start and Test

10 Summary

Get Started

This wizard will walk you through the cluster installation process. First, start by naming your new cluster.

Name your cluster [Learn more](#)

POC

CANCEL

NEXT →

12- Install Ambari: select version

Installer

admin

Get Started

Select Version

Install Options

Confirm Hosts

Choose Services

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

Select Version

Select the software version and method of delivery for your cluster.

HDP-3.1

HDP-3.0

HDP-3.14.0

Accumulo	1.7.0
Infra Solr	0.1.0
Ambari Metrics	0.1.0
Atlas	1.1.0
Druid	0.12.1
URPaaS	2.0.2

Repositories

Using a Public Repository requires Internet connectivity. Using a Local Repository requires you have configured the software in a repository available in your network.

Use Public Repository

Use Local Repository

Provide Base URLs for the Operating Systems you are configuring.

+ADD

13- Set ambari hosts

Installer

admin

1Get Started

2Select Version

3Install Options

4Confirm Hosts

5Choose Services

6Assign Masters

7Assign Slaves and Clients

8Customize Services

9Review

10Install, Start and Test

11Summary

Install Options

Enter the list of hosts to be included in the cluster and provide your SSH key.

Target Hosts

Enter a list of hosts using the Fully Qualified Domain Name (FQDN), one per line. Or use [Pattern Expressions](#)

master
slave

Host Registration Information

☐ Provide your [SSH Private Key](#) to automatically register hosts

☒ Perform [manual registration](#) on hosts and do not use SSH

CHOOSE FILE

No file selected

ssh private key

SSH User Account

root

SSH Port Number

22

14- register hosts

Visual Studio Code

Select Version

Install Options

Confirm Hosts

Choose Services

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

admin

Confirm Hosts

Registering your hosts.

Please confirm the host list and remove any hosts that you do not want to include in the cluster.

Show: All (2) | Installing (0) | Registering (0) | Success (2) | Fail (0)

Host	Progress	Status	Action
master	<div></div>	Success	
slave1	<div></div>	Success	

Items per page: 25 | 1 - 2 of 2

Some warnings were encountered while performing checks against the 2 registered hosts above [Click here to see the warnings.](#)

← BACK

CANCEL

NEXT →

15- select services to install

Install Options

Confirm Hosts

Choose Services

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

Choose Services

Choose which services you want to install on your cluster.

Service	Version	Description
<input checked="" type="checkbox"/> HDFS	3.1.1	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	3.1.1	Apache Hadoop NextGen MapReduce (YARN)
<input type="checkbox"/> Tez	0.9.1	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	3.1.0	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
<input checked="" type="checkbox"/> HBase	2.0.2	Non-relational distributed database and centralized service for configuration management & synchronization
<input type="checkbox"/> Pig	0.16.0	Scripting platform for analyzing large datasets
<input type="checkbox"/> Sqoop	1.4.7	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
<input type="checkbox"/> Oozie	4.3.1	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExtJS Library.
<input checked="" type="checkbox"/> ZooKeeper	3.4.6	Centralized service which provides highly reliable distributed coordination
<input type="checkbox"/> Storm	1.2.1	Apache Hadoop Stream processing framework
<input type="checkbox"/> Accumulo	1.7.0	Robust, scalable, high performance distributed key/value store.
<input type="checkbox"/> Infra Solr	0.1.0	Core shared service used by Ambari managed components.
<input checked="" type="checkbox"/> Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster

8

16- assign masters

Installer admin

Assign Masters

Assign master components to hosts you want to run them on.

SNameNode: slave1 (3.7 GB, 2 cores)

NameNode: master (3.7 GB, 2 cores)

Timeline Service V1.5: slave1 (3.7 GB, 2 cores)

ResourceManager: master (3.7 GB, 2 cores)

Timeline Service V2.0 Reader: master (3.7 GB, 2 cores)

YARN Registry DNS: master (3.7 GB, 2 cores)

History Server: slave1 (3.7 GB, 2 cores)

Hive Metastore: slave1 (3.7 GB, 2 cores)

HiveServer2: slave1 (3.7 GB, 2 cores)

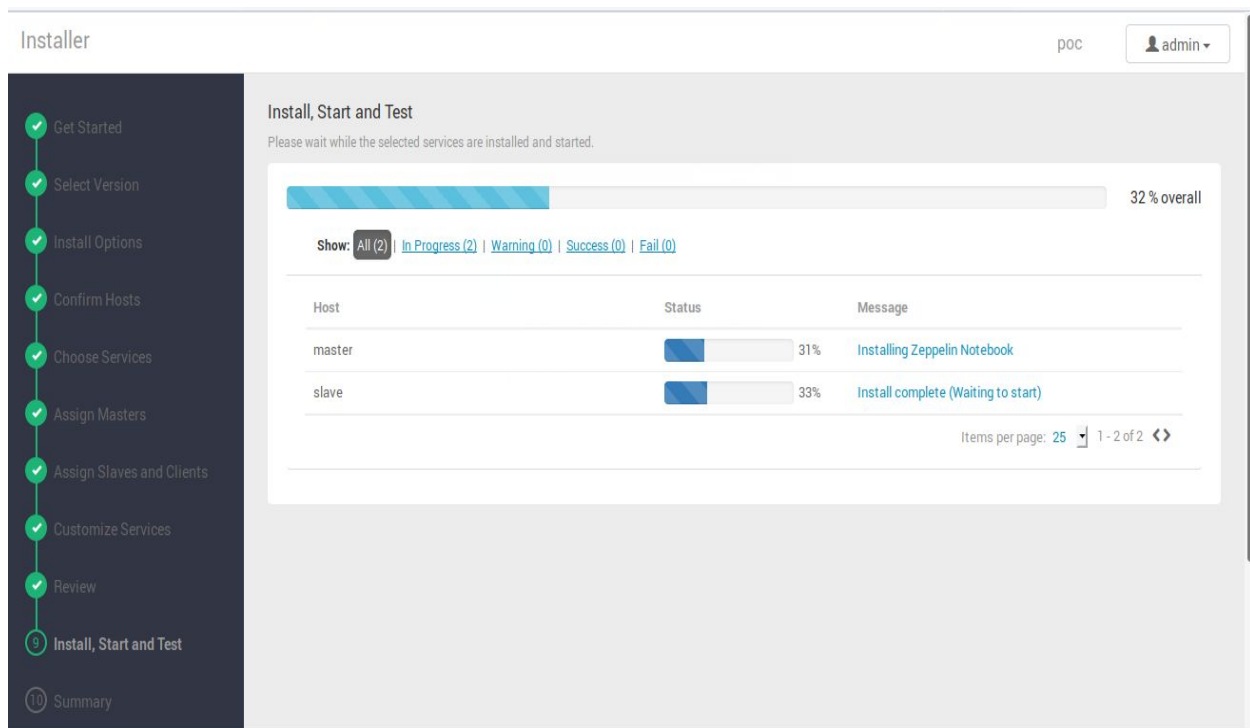
master (3.7 GB, 2 cores)

- NameNode
- ResourceManager
- Timeline Service V2.0 Reader
- YARN Registry DNS
- HBase Master
- ZooKeeper Server
- Metrics Collector
- Grafana
- Atlas Metadata Server
- Kafka Broker
- Activity Explorer
- HST Server
- Activity Analyzer
- Spark2 History Server
- Zeppelin Notebook

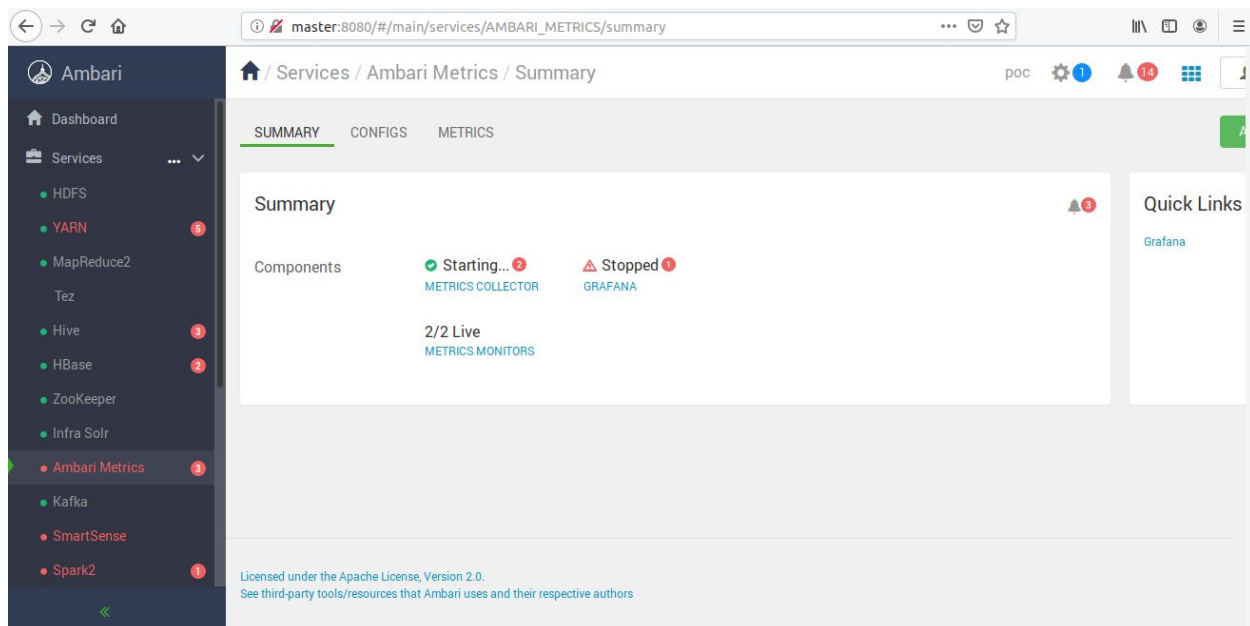
slave1 (3.7 GB, 2 cores)

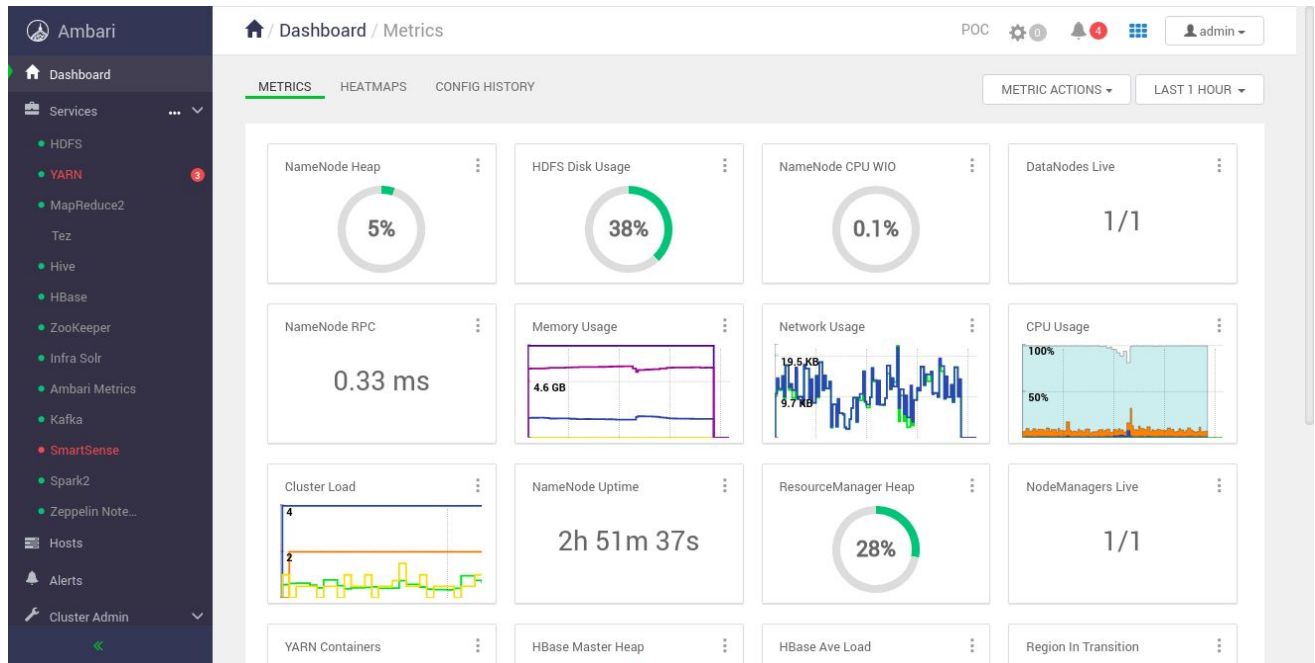
- SNameNode
- Timeline Service V1.5
- History Server
- Hive Metastore
- HiveServer2
- ZooKeeper Server

17- installing services



18- Ambri Dashboard: Hadoop, Spark et compagnies à partir de Ambari (Hortonworks)





II- Setting up Cassandra

Installing Cassandra steps 3.11.4 on CentOS 7.6

- Check the compatibility of the version of Cassandra with your java version
- Installation from Debian packages :

```
echo "deb http://www.apache.org/dist/cassandra/debian 311x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
```

- Update the repositories :
`sudo apt-get update`
- install Cassandra :
`sudo apt-get install cassandra`
- Commands for starting and stopping Cassandra :
`service cassandra start`
- Command to check Cassandra status :
`service cassandra status`

Create a Simple Cassandra Cluster with 3 Nodes

- Cassandra is installed on the 3 V.M (nodes)
- Each node has open communication between the other nodes.
- (disable the firewall on Cassandra)
- The IP addresses of each node are known.
- No data is stored on the 3 Cassandra instances.

(If you've already started your Cassandra instance you'll need to stop it and remove the data it contains. The main reason for this is because the `cluster_name` needs to be the same on all nodes)

```
service cassandra stop
```

```
rm -rf /var/lib/cassandra/data/system/*
```

- Cassandra is configured using various files in the `/etc/cassandra/conf` directory.

```
/etc/cassandra/conf/cassandra.yaml
```

```
-cluster_name : 'poc'
```

```
-seeds : are the IP addresses of the clusters seed servers
```

```
-listen_address : the server IP
```

```
-rpc_address : the server IP
```

```
-endpoint_snitch :GossipingPropertyFileSnitch
```

- Edit the `cassandra-rackdc.properties`

```
/etc/cassandra/conf/cassandra-rackdc.properties
```

- Troubleshooting operating timeouts :

Cassandra imposes timeouts on read and write operations to prevent a given operation from negatively impacting the performance of the cluster. If you encounter an operation timeout, you can take the following actions to promote operation success.

```

sudo nano /etc/cassandra/cassandra.yaml

# How long the coordinator should wait for read operations to complete
read_request_timeout_in_ms: 50000
# How long the coordinator should wait for seq or index scans to complete
range_request_timeout_in_ms: 100000
# How long the coordinator should wait for writes to complete
write_request_timeout_in_ms: 20000
# How long the coordinator should wait for counter writes to complete
counter_write_request_timeout_in_ms: 50000
# How long a coordinator should continue to retry a CAS operation
# that contends with other proposals for the same row
cas_contention_timeout_in_ms: 10000
# How long the coordinator should wait for truncates to complete
# (This can be much longer, because unless auto_snapshot is disabled
# we need to flush first so we can snapshot before removing the data.)
truncate_request_timeout_in_ms: 600000
# The default timeout for other, miscellaneous operations
request_timeout_in_ms: 100000

# How long before a node logs slow queries. Select queries that take longer than
# this timeout to execute, will generate an aggregated log message, so that slow quer
# can be identified. Set this value to zero to disable slow query logging.
slow_query_log_timeout_in_ms: 5000

```

Importing the csv file into the Cassandra Cluster

- Create a keyspace :

```

tion refused"))}
[root@slave1 default.conf]# cqlsh slave1
Connected to poc Cluster at slave1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.4 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> CREATE KEYSPACE weather WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 1};
cqlsh> describe keyspaces

system_schema system_auth system weather system_distributed system_traces

cqlsh> use weather
... ;

```


- Create a table (a column family)

```

cqlsh:weather> CREATE table isd_history (USAF varchar, WBAN varchar, station_name varchar, country varchar, state varchar, ICAO varchar, LAT float,
LON float, ELEV float, begin varchar, end varchar, PRIMARY KEY (USAF, WBAN)) ;
SyntaxException: line 1:152 mismatched input 'begin' expecting ')' (...LON float, ELEV float, [begin]...)
cqlsh:weather> CREATE table gsod (STN varchar, WBAN varchar, YEARMODA varchar, TEMP float, DEWP float, SLP float, STP float, VISIB float, WDSP
float, MXSPD float, GUST float, MAX float, MIN float, PRCP float, SNDP float, F varchar, R varchar, S varchar, H varchar, TH varchar, TOO varchar, P
PRIMARY KEY ((STN, WBAN), YEARMODA));
cqlsh:weather> COPY gsod (STN, WBAN, YEARMODA, TEMP, DEWP, SLP, STP, VISIB, WDSP, MXSPD, GUST, MAX, MIN, PRCP, SNDP, F, R, S, H, TH
, TOO ) from '/home/centos/gsod.csv' with DELIMITER = ',';
Using 1 child processes

Starting copy of weather.gsod with columns [stn, wban, yearmoda, temp, dewp, slp, stp, visib, wdsp, mxspd, gust, max, min, prcp, sndp, f, r, s
h, th, too].
Processed: 35000 rows; Rate: 12184 rows/s; Avg. rate: 9915 rows/s

```

- Copy the data on the table :

```
SyntaxException: line 1:5 no viable alternative at input 'keyspace' [[drop] keyspace...]
cqlsh> drop keyspace weather ;
cqlsh> CREATE KEYSPACE IF NOT EXISTS resto_NY WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 1};
cqlsh> CREATE KEYSPACE weather WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'dc1' : 1 , 'datacenter1' : 1};
cqlsh> use weather
... ;
cqlsh:weather> CREATE table gsoad (STN varchar, WBAN varchar, YEARMODA varchar, TEMP float, DEWP float, SLP float, STP float, VISIB float, WDSF
float, MXSPD float, GUST float, MAX float, MIN float, PRCP float, SNPD float, F varchar, R varchar, S varchar, H varchar, TH varchar, TOO varchar,
PRIMARY KEY ((STN, WBAN), YEARMODA));
cqlsh:weather> COPY gsoad (STN, WBAN, YEARMODA, TEMP, DEWP, SLP, STP, VISIB, WDSF, MXSPD, GUST, MAX, MIN, PRCP, SNPD, F, R, S, H, TH
, TOO ) from '/home/centos/gsoad.csv' with DELIMITER = ',';
Using 1 child processes

Starting copy of weather.gsoad with columns [stn, wban, yearmoda, temp, dewp, slp, stp, visib, wdsf, mxspd, gust, max, min, prcp, snpd, f, r, s
, h, th, too].
Processed: 35000 rows; Rate: 11061 rows/s; Avg. rate: 9266 rows/s
```

- Result after importing data

```

Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 5 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" inf
o={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 5 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" inf
o={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Failed to import 20 rows: Unavailable - Error from server: code=1000 [Unavailable exception] message="Cannot achieve consistency level ONE" in
fo={"required_replicas": 1, 'alive_replicas': 0, 'consistency': 'ONE'}, given up after 5 attempts
Exceeded maximum number of insert errors 1000
Failed to process 1007 rows: failed rows written to import_weather_gsoo.err
Exceeded maximum number of insert errors 1000
Processed: 97580000 rows; Rate: 6203 rows/s; Avg. rate: 11775 rows/s
97580000 rows imported from 0 files in 2 hours, 18 minutes, and 7.289 seconds (0 skipped).
cqlsh:weather> Connection to slave1 closed by remote host.
Connection to slave1 closed.

```

III- Download and clean data

To download and clean the data we used a bash script.

```
#!/bin/bash
for year in {1930..1950}; do
# Create a directory for the year
  if [ ! -d "$year" ]; then
    echo Creating a directory.
    mkdir -p $year
  fi
# Download the data (if we don't already have it)
  if [ ! -f "$year/gsod_$year.tar" ]; then
    wget ftp://ftp.ncdc.noaa.gov/pub/data/gsod/$year/gsod_$year.tar -O
$year/gsod_$year.tar
  fi

# Unzip the data
  if [ -f "$year/gsod_$year.tar" ]; then
    echo Unzipping the downloaded data.
    tar -xvf $year/gsod_$year.tar -C $year/
    rm $year/gsod_$year.tar
    for filename in `ls $year/*.gz`; do
      gunzip $filename
    done
  fi
done
for year in {1930..1950}; do
# Strip the first line of each of the .op files
  for filename in `ls $year/*.op`; do
    tail -n +2 $filename > $filename.header_stripped
```

```

        mv $filename.header_stripped $filename
    done
# Stack the .op files
    cat $year/*.op > $year.op
    rm -rf $year
done

#converting to csv
if [ -f "gsod.csv" ]; then
    rm gsod.csv
fi
for year in {1930..1950}; do
    echo $year
    awk '{printf
"%s;%s;%s;%.1f;%.1f;%.1f;%.1f;%.1f;%.1f;%.1f;%.1f;%.1f;%.2f;%.1f;%s;%s
;%s;%s;%s;%s\n",$1,$2,$3,$4,$6,$8,$10,$12,$14,$16,$17,substr($0,103,6),subs
tr($0,111,6),substr($0,119,5),$21,substr($22,1,1),substr($22,2,1),substr($2
2,3,1),substr($22,4,1),substr($22,5,1),substr($22,6,1)}' OFS=';' $year.op
>> gsod2.csv;
    rm $year.op

```

IV- Data processing with Pyspark:

```

import pandas as pd
import pyspark
from pyspark.context import SparkContext
from pyspark.sql.session import SparkSession
from functools import reduce
from past.builtins import xrange
from datetime import datetime
from pyspark.sql.functions import col, udf
from pyspark.sql.types import DateType
from pyspark.sql.functions import *

sc = SparkContext("local","gsod")
spark = SparkSession(sc)
#import gsod data into dataframe
df =
spark.read.format("csv").option("delimiter",";").load("hdfs:///poc/data/gso
d.csv")
#import station history data into dataframe

```

```

hi =
spark.read.format("csv").option("header","true").load("hdfs:///poc/data/isd
-history.csv")
#import country list data into dataframe
cn =
spark.read.format("csv").option("delimiter",";").option("header","true").lo
ad("hdfs:///poc/data/country-list.csv")
#import continent list into dataframe
co =
spark.read.format("csv").option("delimiter",";").option("header","true").lo
ad("hdfs:///poc/data/continent.csv")

# Add header
oldColumns = df.schema.names
newColumns = ["STN", "WBAN", "YEARMODA", "TEMP", "DEWP", "SLP", "STP",
"VISIB", "WDSP", "MXSPD", "GUST", "MAX", "MIN", "PRCP", "SNDP",
"F", "R", "S", "H", "TH", "TOO"]
df = reduce(lambda df, idx: df.withColumnRenamed(oldColumns[idx],
newColumns[idx]), xrange(len(oldColumns)), df)
# set column types
types=["string","string","string","float","float","float","float","float","
float","float","float","float","float","float","float","string","string","S
tring","string","string","String"]
df = reduce(lambda df, idx: df.withColumn(newColumns[idx],
df[newColumns[idx]].cast(types[idx])),xrange(len(newColumns)), df)
# set date type column
func = udf(lambda x: datetime.strptime(x,'%Y%m%d'), DateType())
df = df.withColumn('YEARMODA', func(col('YEARMODA')))
# change temp from F to C
tocColumns=["TEMP", "DEWP", "MAX", "MIN"]
df= reduce(lambda df, idx:
df.withColumn(tocColumns[idx],when(df[tocColumns[idx]]
<9999.9,round((df[tocColumns[idx]]-32)*5/9,1)),xrange(len(tocColumns)),
df)
#join continent table to country
cn=cn.join(co,on='ID',how='left')
# join country table to station history
df3 = hi.join(cn, hi.CTRY == cn.ID,how='left').drop(cn.ID)
# join all data
df4 = df.join(df3, (df.STN == df3.USAF) & (df.WBAN ==
df3.WBAN),how='left').drop(df3.USAF).drop(df3.WBAN)

# basic data by year
df7=df4.select('STN','WBAN','YEARMODA','COUNTRY').groupby(year('YEARMODA')).
alias('Year')).agg(countDistinct('STN','WBAN').alias('No of
station'),count('STN').alias('No of
lines'),countDistinct('COUNTRY').alias('No of countries')).orderBy('Year')

```

```

df7.toPandas().to_csv('year.csv',index=False)

# mean temp by decade
df5=df4.groupby(floor(year('YEARMODA')/10)
%10).agg(mean('TEMP').alias("Mean
Temp"),count('TEMP')).withColumnRenamed('(FLOOR((year(YEARMODA) / 10)) %
10)', 'decade')
df5.toPandas().to_csv('mean.csv',index=False)
#mean temp by decade per continent
df5c=df4.groupby(floor(year('YEARMODA')/10)
%10,'CONTINENT').agg(mean('TEMP').alias("Mean
Temp"),count('TEMP')).withColumnRenamed('(FLOOR((year(YEARMODA) / 10)) %
10)', 'decade').orderBy('decade')
df5c.toPandas().to_csv('mean_cont.csv',index=False)

# station by year
df8=df4.select('STN','WBAN',year('YEARMODA').alias('year'),'STATION
NAME','COUNTRY','LAT','LON').distinct()
df8.toPandas().to_csv('station.csv',index=False)

# Temp with decade and continent
dfbb=df4.where(year('YEARMODA')>1969)
dfB=dfbb.select((floor(year('YEARMODA')/10)
%10).alias('decade'),'TEMP','CONTINENT')
dfB.write.csv("hdfs:///poc/data/decade_cont.csv")
#dfB.toPandas().to_csv('decade_cont.csv',index=False)

# Min and Max temp every year
df21=df4.select(year('YEARMODA').alias('Year'),'STATION
NAME','COUNTRY','MIN').groupBy('Year').agg(min('MIN').alias('MIN')).orderBy
('Year')
df22=df4.select(year('YEARMODA').alias('Year'),'STATION
NAME','COUNTRY','MAX').groupBy('Year').agg(max('MAX').alias('MAX')).orderBy
('Year')
df11=df4.select(year('YEARMODA').alias('Year'),'YEARMODA','STATION
NAME','COUNTRY','MIN')
df12=df4.select(year('YEARMODA').alias('Year'),'YEARMODA','STATION
NAME','COUNTRY','MAX')
df13=df4.select(year('YEARMODA').alias('Year'),'STATION
NAME','COUNTRY','TEMP')
df31=df21.join(df11,(df21.MIN==df11.MIN)&(df21.Year==df11.Year),how='left')
.drop(df11.Year).drop(df11.MIN).orderBy('Year')
df32=df22.join(df12,(df22.MAX==df12.MAX)&(df22.Year==df12.Year),how='left')
.drop(df12.Year).drop(df12.MAX).orderBy('Year')
df31.toPandas().to_csv('min.csv',index=False)
df32.toPandas().to_csv('max.csv',index=False)

```

```

# rainy days per year
dfr=df4.select(year('YEARMODA').alias('Year'),'R').crosstab('Year','R').ord
erBy('Year_R')
dfr.toPandas().to_csv('rain.csv',index=False)

# snowy days per year
dfs=df4.select(year('YEARMODA').alias('Year'),'S').crosstab('Year','S').ord
erBy('Year_S')
dfs.toPandas().to_csv('snow.csv',index=False)

```