Juan Antonio Cantarero

Ingeniero Informático y Jefe de Equipo en Proyectos para Telcos.

Puedes saber más sobre mí en LinkedIn: es.linkedin.com/in/juancantarero/

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2015-01-30

Instalación de un entorno Hadoop con Ambari en AWS

Contenidos.

Prerrequisitos.

Si quieres aprovechar bien el contenido de este tutorial, deberás tener:

- conocimientos de la arquitectura de Hadoop. Nivel medio.
- conocimientos de administración de Linux. Nivel medio.
- conocimientos de administración de Redes/Seguridad. Nivel básico.

Sobre todo el primer punto es importante. En este tutorial no se explica la arquitectura Hadoop. ;-)

Introducción y objetivos.

El objetivo de este tutorial es aprender a instalar, configurar y monitorizar un cluster Hadoop en modo distribuido, mediante el framework Ambari, todo ello usando máquinas virtuales de AWS. Espero que os sea útil.

¿Qué es Hadoop?

Hadoop es el nombre del peluche que tenía el hijo de Doug Cutting, uno de los dos creadores de Hadoop junto con Mike Cafarella. Aparte de esto, Hadoop es un framework de código abierto creado en 2005 usado para almacenar, procesar y analizar grandes volúmenes de datos, enfocado en el paradigma MapReduce.

Hadoop se encuadra dentro del frameork Apache y posee su propio sistema de archivos o filesystem (HDFS: Hadoop Distributed File System) y un entorno de utilidades adicionales (aunque opcionales) que facilitan ciertas tareas (ecosistema Hadoop: Flume, Nagios, HBase, Ganglia, Scoop, ZooKeeper, Pig, Flume. El nombre de “ecosistema” es obvio como deduciréis). Actualmente tanto Hadoop como su ecosistema están en contínua evolución debido a sus evidentes ventajas: gratuito y efectivo.

TODO: en otros tutoriales explicaré más en detalle qué es eso de MapReduce (a mí la primera vez me costó entenderlo), HDFS, y el ecosistema Hadoop.

Últimas Noticias

» 2015: ¡Volvemos a la oficina!

» Curso JBoss de Red Hat

» Si eres el responsable o líder técnico, considérate desafortunado. No puedes culpar a nadie por ser gris

» Portales, gestores de contenidos documentales y desarrollos a medida

» Comentando el libro Start-up Nation, La historia del milagro económico de Israel, de Dan Senor & Salu Singer

Histórico de noticias

Últimos Tutoriales

» Política de reintentos con Spring Retry

» Guía para entender y usar expresiones regulares

» Instalación de Alfresco 5.0.c sobre Vagrant

» Filtros en AngularJS

» Menos es más. Filosofía detrás de los principios YAGNI, KISS, DRY, DIE, hijos de la navaja de Ockham.

¿Qué es Ambari?

Para aprovechar bien las capacidades del entorno Hadoop conviene disponer de un mainframe distribuido (cluster) y proceder a la instalación y configuración de toda la arquitectura de los nodos maestro/esclavo/backups en distintas redes y/o servidores. Hacer esto manualmente es un trabajo complicado y bastante propenso a errores. Pensar en todos los pasos que hay que seguir en todos los nodos: instalación del paquete de hadoop, configuración de variables de entorno, definición de archivos de configuración, definir permisos, levantar demonios, etc.

Además, la posterior administración y monitorización de un entorno distribuido como Hadoop es oscura.

> TODO: en otro tutorial explicaré en detalle cómo instalar una arquitectura Hadoop manualmente y desde cero.

Para aliviar todos estos inconvenientes surge Ambari: Ambari es un framework open source de la empresa HortonWorks. Consiste en un conjunto de herramientas diseñadas para simplificar la instalación y monitorización de un cluster Hadoop.

¿Qué es AWS?

Amazon Web Services (AWS) es una suite de herramientas basadas en la nube que ofrece un conjunto de servicios globales de informática, almacenamiento, bases de datos, análisis, aplicaciones, etc. Para el contexto de nuestro objetivo, AWS ofrece varios servicios basados en la nube, incluida una serie de instancias de máquinas virtuales cuyo escalado se puede aumentar y reducir automáticamente, todo esto gratuitamente (siempre que esté correctamente gestionado. Ya lo veremos más adelante).

Como es evidente poca gente (salvo perfiles tipo administradores de Entornos Productivos o Centros de Procesamiento en empresas o instituciones públicas) tiene acceso a un cluster real para poder experimentar o trabajar con una arquitectura real distribuida. Considerar que un entorno medianamente “real” de Hadoop es a partir de 4-8 nodos (actualmente hay empresas con más de 1500 nodos activos). Podemos montarnos nuestro propio cluster compartiendo varias máquinas físicas en red, de compañeros por ejemplo, o bien lanzar varias máquinas virtuales en la misma máquina (si tu equipo lo soporta claro).

Pero la mejor alternativa es usar AWS lo que nos permite definir mediante máquinas virtuales (o instancias) un cluster y poder trabajar con él de forma sencilla (más o menos), e independiente. AWS permite definir N máquinas virtuales con grandes capacidades hardware/software, todo ello en la nube por lo que localmente no necesitaremos de un gran equipo.

Entorno.

Para seguir los pasos en este tutorial necesitaremos:

- Sistema Operativo Linux: lo usaremos para conectarnos a cada una de las instancias y poder configurarlas. Yo he usado una máquina virtual Linux CentOS v6 (64 bits, requerido por Hadoop) y 8Gb RAM. Os recomiendo la distribución CentOS ya que es la más popular dentro de la comunidad Hadoop.

Nuestro trabajo estará en la nube por lo que básicamente esto es todo lo que necesitas. Si no tienes un Linux en local también puedes trabajar desde Windows aunque algunas cosas se complicarán.

Definición de las instancias en Amazon Web Services.

Manos a la obra. Empezaremos definiendo nuestra cuenta en Amazon web services. Si ya tienes una, sáltate esta primera parte. Para ello:

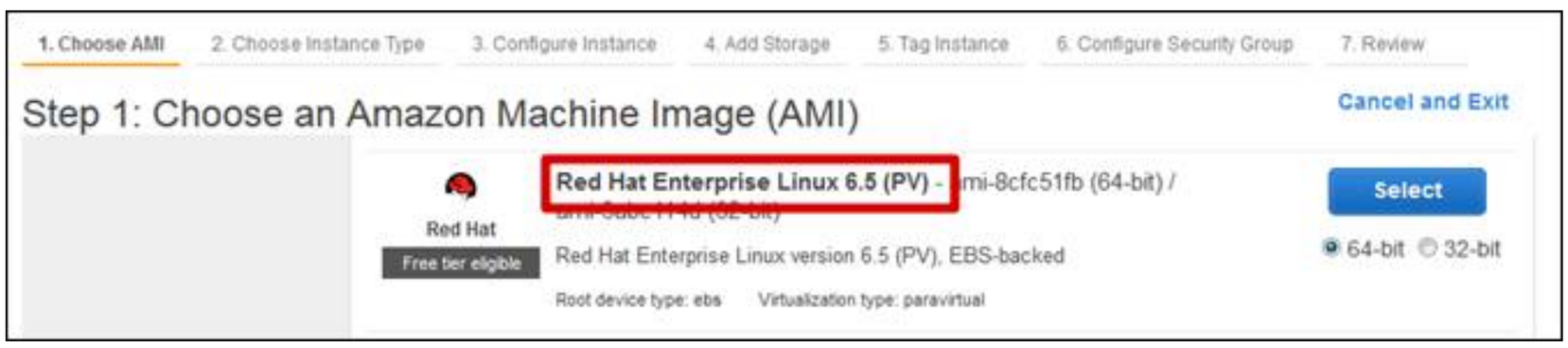
> NOTA: el equipo de Amazon cambia constantemente la interfaz de AWS, por lo que no puedo asegurar que las capturas que aparezcan en este manual se correspondan con la versión que estéis usando en vuestras pruebas.

1. Accede a http://aws.amazon.com/es/ y dar en “Cree una cuenta Gratuita”. Rellena los diferentes formularios con tus datos:

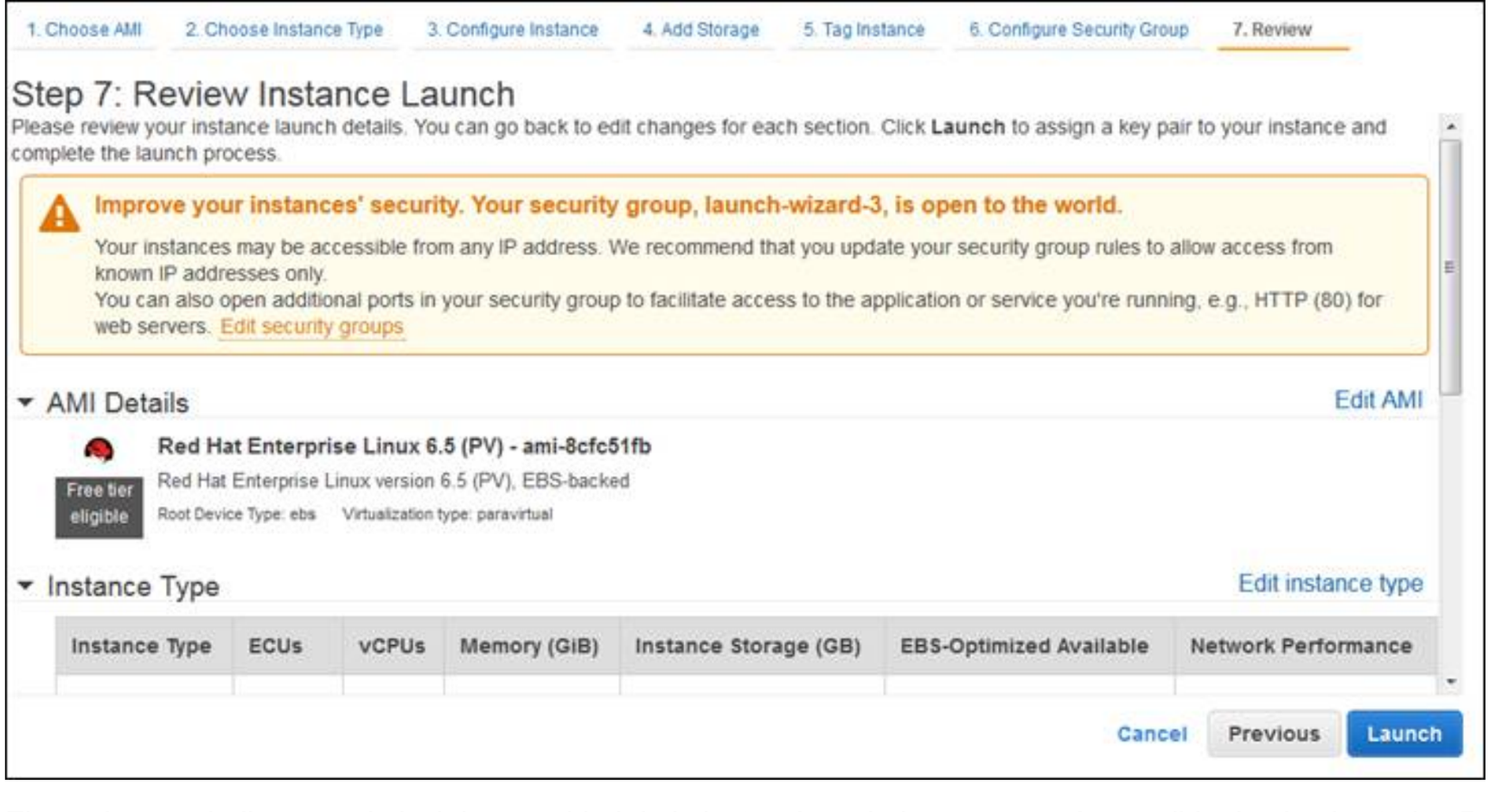
2. Lo importante a tener en cuenta viene en la siguiente pantalla. Amazon te ofrece 1 año gratuito de servicios pero tendrás que indicar tu tarjeta de crédito/débito (debe ser real). Tranquilo, no te harán cargo alguno siempre que uses servicios gratuitos, como los de este tutorial. Puedes obtener más información aquí https://aws.amazon.com/es/free/ aunque Amazon te informa muy claramente, de cuándo un servicio que has seleccionado es o no de pago y cuánto te va a facturar:

3. Una vez verificados los datos proporcionados de la tarjeta y elegido el plan que deseas (AWS Support (Basic)), recibirás varios correos de bienvenida y ya tendrás la cuenta de AWS creada. Accede a ella.
   Lo siguiente será crear las máquinas virtuales o instancias como las llama Amazon. No es difícil pero hay que dar una serie de pasos que no son intuitivos, pero hay que darlos para que todo encaje posteriormente.
4. Accede a tu nueva cuenta AWS. No te marees por la cantidad de servicios que nos ofrecen. Sólo vamos a necesitar uno: EC2. Lo primero que yo haría es definir la zona horaria. La zona disponible más acorde a nuestro uso horario (UTC+01) es Irlanda. Cámbialo desde la vista general de Servicios:

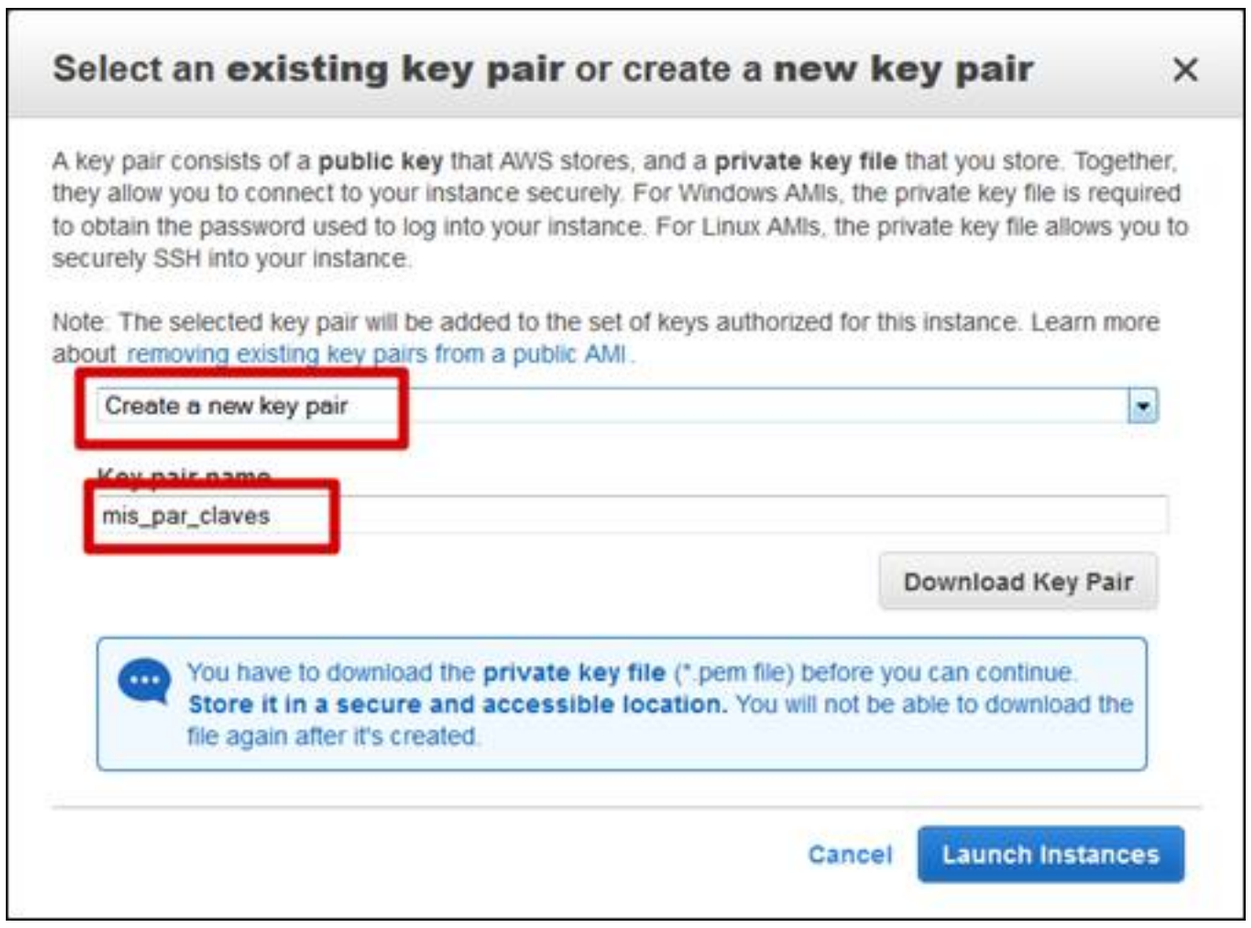
5. Accede a “EC2 Virtual Servers” y luego a instancias. Aquí es donde crearemos las máquinas virtuales. Pulsa en “Launch Instance”. De todas las que salen, selecciona la versión **Red Hat Enterprise Linux 6.5** para 64 bits (1 core, 512 Mb RAM). Recuerda que para aprovechar Hadoop hay que usar entornos de 64 bits. Fíjate también en el subtítulo “Free tier Elegible”: así estamos seguros de usar un servicio gratuíto.
   Aunque lo más lógico hubiera sido elegir la última versión 7.0 de Red Hat (la primera de todas en la lista), posteriormente nos hubieran aparecido errores en la instalación de Ambari:



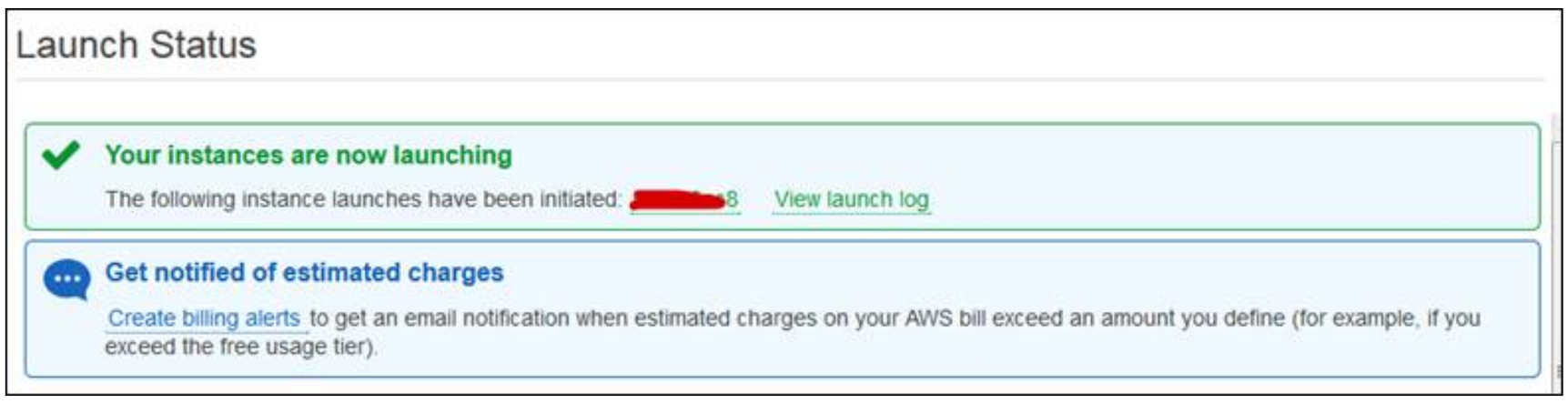
6. Pulsaremos en “Next” en las sucesivas pantallas de configuración dejando por defecto todos los valores hasta pulsar en “Review and Launch”. Nos aparecerá algo como esto:



El warning nos indica que el nivel de seguridad de la instancia es bajo ya que será accesible desde el exterior. De momento lo obviamos y pulsamos en “Launch”.



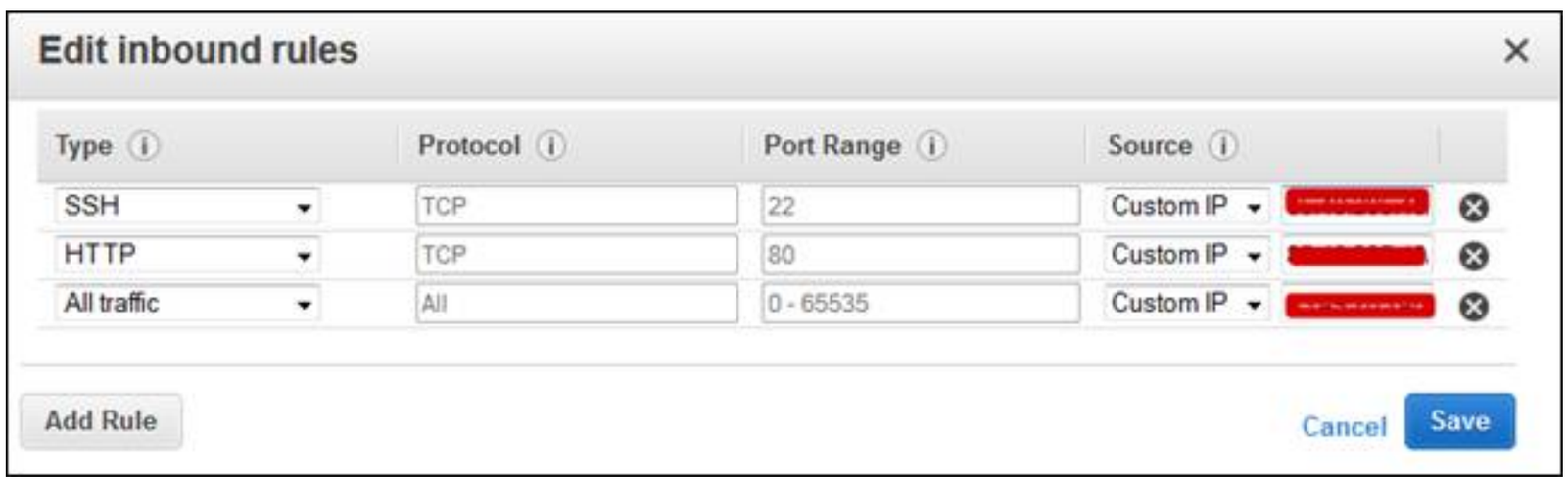
7. Aquí definiremos el fichero .pem con las claves para permitir el acceso a nuestros nodos Linux. Generaremos un nuevo fichero de claves y lo guardamos en local ya que lo necesitaremos posteriormente (no perder este archivo). Pulsamos en “Launch Instances”:



La segunda recomendación es opcional pero te recomiendo que la sigas. Sirve para configurar una alerta por correo cuando esté previsto realizar un cargo en la tarjeta.

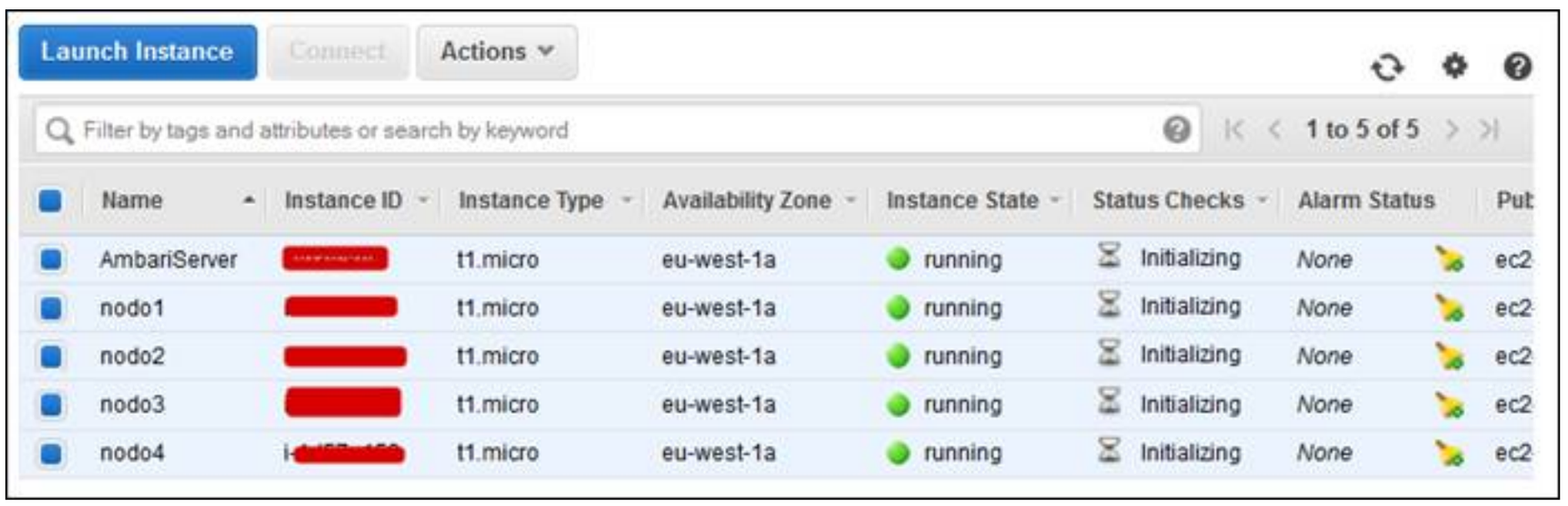
8. En la vista de instancias, verás tu nueva instancia corriendo (por defecto al crear una instancia se levanta automáticamente. Esto no debería ser así). Las propiedades que se muestran en cada columna y que merecen ser comentadas son:
    - Name : nombre nemotécnico que quieras darle. Es un TAG AWS. AWS te permite gestionar TAGs en entidades (BBDD, instancias, volúmenes, etc.) para poder definir valores personalizados. El TAG “Name” de Instances viene creado por defecto.
    - Instance type : debe ser la gratuita: t1.micro.
    - Availibity zone : zona horaria que hemos configurado: eu-west.
    - Instance state : estado. Aquí es importante tener claro los conceptos:
        1. Stopped: equivale a un shutdown. Cuando veas que no vas a seguir trabajando más, deja las instancias en este estado.
        2. Terminated: cuidado porque esto borra la instancia definitivamente. No usar nunca salvo para limpieza. Una vez en este estado, tarda tiempo en desaparecer definitivamente la instancia de la lista.
        3. Running: pues eso. Aunque te salgas de la sesión AWS, las máquinas en este estado no se pararán.
    - Alarm status : si tienes configuradas alarmas, aparecerán aquí.
    - Public IP : por defecto tienes una IP dinámica que cambiará cada vez que levantes la instancia. Más adelante veremos cómo resolver esto.
    - Key name : nombre del fichero .pem usado. Por comodidad debería ser el mismo para todas las instancias.
    - Security groups: define la seguridad del tráfico de la red in/out. Luego lo veremos. Por defecto asigna un grupo “launch-wizard-N” sin restricciones para el tráfico entrante.
9. A continuación definiremos la seguridad del tráfico en la red para la instancia. Nos vamos a la vista de “Security Groups” y pulsamos en “Create Security Groups”. Indicamos un nombre y pulsamos en “Add Rule” para definir las reglas inbound:

Debemos definir reglas para los protocolos SSH y HTTP. Lo más simple es no indicar restricciones de IPs (Source=Anywere) pero lo mejor es restringir a nuestra red (Source=Custom IP):



Volvemos a la vista de instancias, “ActionsàNetworkingàChange Security Groups” y seleccionamos el nuevo grupo creado.

10. Ahora “clonaremos” esta instancia para tener finalmente el número de nodos deseado. Para ello, en la vista de instancia, “ActionsàLaunch more like this” y repetiremos el proceso para cada nodo salvo que ahora indicamos el fichero de claves .pem previamente creado (y tildar la opción “I acknowledge that I have access to the selected prívate file….”). Por defecto, AWS asociará una misma imagen (Amazon Machine Image AMI) para cada instancia con toda la configuración definida, así que no tenemos que preocuparnos de esta gestión.
    En el modelo AWS, cualquier instancia procede de una AMI. Es como lo tienen montado.
    Otra forma de proceder hubiera sido al comienzo, al crear la primera instancia indicamos “Number of instances=N” pero entonces el proceso de configuración posterior tendría que haberse repetido N veces.
11. Este paso es opcional pero recomendable por la comodidad que posteriormente ofrece.
    A continuación definiremos las IPs de los nodos para que sean estáticas entre reinicios. A esto Amazon lo llama “Elastic IPs” (EIP). Nos vamos a la vista de “Elastic IPs” y pulsamos en “Allocate New Address”, y seleccionamos la instancia. Repite el proceso para cada instancia.
    Sólo podremos definir/asignar 5 IPs como máximo (importante limitación). Cuidado, el uso de EIPs a instancias que no están en ejecución (Running) tiene asociado un cargo mensual en tarjeta ($0.005 por Elastic IP).
    Al final nos debería quedar algo como lo siguiente:



Configuración de los nodos Hadoop.

Antes de instalar Ambari, debemos dejar los nodos del cluster Hadoop configurados para que Ambari los pueda reconocer. Para ello:

- En nuestro Linux en local debemos tener copiado (en el $HOME por ejemplo) el fichero de claves .pem que hemos generado. A continuación y con permisos de root, registramos las IPs de cada nodo en el fichero /etc/hosts de nuestro Linux local. Tener en cuenta que las instancias creadas en AWS tienen un usuario por defecto “ec2-user” sin permisos de root, pero con “sudo” podemos resolver este inconveniente. Los nombres deberán seguir la nomenclatura que se muestra a continuación:

```
54.154...    AmbariServer              AmbariServer
54.154...    nodo1.hdp.hadoop          nodo1
54.154...    nodo2.hdp.hadoop          nodo2
54.154...    nodo3.hdp.hadoop          nodo3
54.154...    nodo4.hdp.hadoop          nodo4
```

- Los siguientes puntos habrá que repetirlos para cada nodo hadoop: nos logamos en el nodo con ssh. Respondemos

con “yes” para agregar el host en la lista de conocidos y que no nos vuelva a preguntar:

```
ssh -i mi_par_claves_AWS.pem ec2-user@AmbariServer
The authenticity of host 'nodo1 (*******)' can't be established.
RSA key fingerprint is f0:c6:1f:ce:b7:44:70:cf:d8:4a:47:2e:7c:04:23:48.
Are you sure you want to continue connecting (yes/no)? yes
```

Configuramos el fichero de hosts de la misma forma que en nuestro Linux local. Podemos editarlo manualmente (siempre como root) o bien copiarlo desde nuestro Linux local mediante:

```
scp -i mi par claves AWS.pem /etc/hosts ec2-user@AmbariServer:/home/ec2-user
```

Cambiamos el nombre del host para que coincida con el configurado en el fichero de hosts:

```
sudo hostname nodo1.hdp.hadoop
```

Este cambio lo podemos hacer permanente editando la propiedad HOSTNAME en el fichero:

```
sudo vi /etc/sysconfig/network
```

Durante la instalación de Ambari, necesitamos tener abiertos ciertos puertos por lo que desactivaremos las tablas IP:

```
sudo chkconfig iptables off
sudo chkconfig ip6tables off
sudo /etc/init.d/iptables stop
```

En la versión Red Hat que nos proporciona AWS el servicio ntpd está por defecto caído. Lo levantamos con:

```
sudo service ntpd start
```

Instalación del entorno Ambari.

A continuación podemos ya instalar el paquete Ambari (última versión 1.7.0) en uno de los nodos (o instancia siguiendo la nomenclatura Amazon). Para ello:

1. Nos logamos en el nodo que hará de servidor Ambari con ssh. Respondemos con “yes” para agregar el host en la lista de conocidos y que no nos vuelva a preguntar:

   ```
   ssh -i mi_par_claves_AWS.pem ec2-user@AmbariServer
   The authenticity of host 'nodo1 (*******)' can't be established.
   RSA key fingerprint is f0:c6:1f:ce:b7:44:70:cf:d8:4a:47:2e:7c:04:23:48.
   Are you sure you want to continue connecting (yes/no)? yes
   ```

   Configuramos de la misma forma el fichero de hosts. Podemos editarlo manualmente (siempre como root) o bien copiarlo desde nuestro Linux local mediante:

   ```
   scp -i mi_par_claves_AWS.pem /etc/hosts ec2-user@AmbariServer:/home/ec2-user
   ```

   Cambiamos el nombre del host para que coincida con el configurado en el fichero de hosts:

   ```
   sudo hostname ambariserver
   ```

   Hacerlo permanente editando la propiedad HOSTNAME en el fichero:

   ```
   sudo vi /etc/sysconfig/network
   ```

   Generamos una clave pública sin password:

   ```
   ssh-keygen -t rsa
   Generating public/private rsa key pair.
   Enter file in which to save the key (/root/.ssh/id_rsa):
   Enter passphrase (empty for no passphrase):
   Enter same passphrase again:
   Your identification has been saved in /root/.ssh/id_rsa.
   Your public key has been saved in /root/.ssh/id_rsa.pub.
   The key fingerprint is:
   b5:cb:81:e9:c3:2f:ac:72:98:e0:53:71:23:30:e1:f9 root@ip-*******.eu-west-1.compu
   The key's randomart image is:
   +--[ RSA 2048]----+
   ```

2. Ya estamos listos para instalar el paquete Ambari. Para ello nos descargamos el software de HortonWorks mediante HTTP (también podemos usar RPM). Da lo mismo. Aseguraros que la versión es la centos6:

   ```
   wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.4.3.38/
   http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.4.3.38/ambari.
   Resolving public-repo-1.hortonworks.com... 54.192.29.221, 54.192.30.186, 54.192.
   Connecting to public-repo-1.hortonworks.com|54.192.29.221|:80... connected.
   HTTP request sent, awaiting response... 200 OK
   Length: 770 [binary/octet-stream]
   Saving to: “ambari.repo”
   100%[======================================================================
   2014-12-15 03:44:23 (46.8 MB/s) - “ambari.repo” saved [770/770]
   ```

   El fichero “ambari.repo” que se ha descargado con la descripción de los paquetes se copia posteriormente a:

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

   …para que yum tome los cambios. A continuación instalamos los paquetes y dependencias. Aceptar todas las opciones con el valor por defecto (copio sólo algunas trazas):

   ```
   sudo yum install ambari-server
   Loaded plugins: amazon-id, rhui-lb, security
   Setting up Install Process
   Resolving Dependencies
   --> Running transaction check
   ---> Package ambari-server.noarch 0:1.4.3.38-1 will be installed
   ```

```
--> Processing Dependency: postgresql-server >= 8.1 for package: ambari-server-
--> Running transaction check
---> Package postgresql-server.x86_64 0:8.4.20-1.el6_5 will be installed
--> Processing Dependency: postgresql-libs(x86-64) = 8.4.20-1.el6_5 for package
--> Processing Dependency: postgresql(x86-64) = 8.4.20-1.el6_5 for package: pos
--> Processing Dependency: libpq.so.5()(64bit) for package: postgresql-server-8
--> Running transaction check
---> Package postgresql.x86_64 0:8.4.20-1.el6_5 will be installed
---> Package postgresql-libs.x86_64 0:8.4.20-1.el6_5 will be installed
--> Finished Dependency Resolution
Dependencies Resolved
```

Si tenéis problemas de dependencias (puede que con python o postgresql) es porque las versiones (de SO, de paquete, etc.) que estáis usando no son las correctas. Revisarlo bien.
Hacemos que Ambari se autoconfigure (aceptamos todos los valores por defecto):

```
sudo ambari-server setup
```

Iniciamos el servidor:

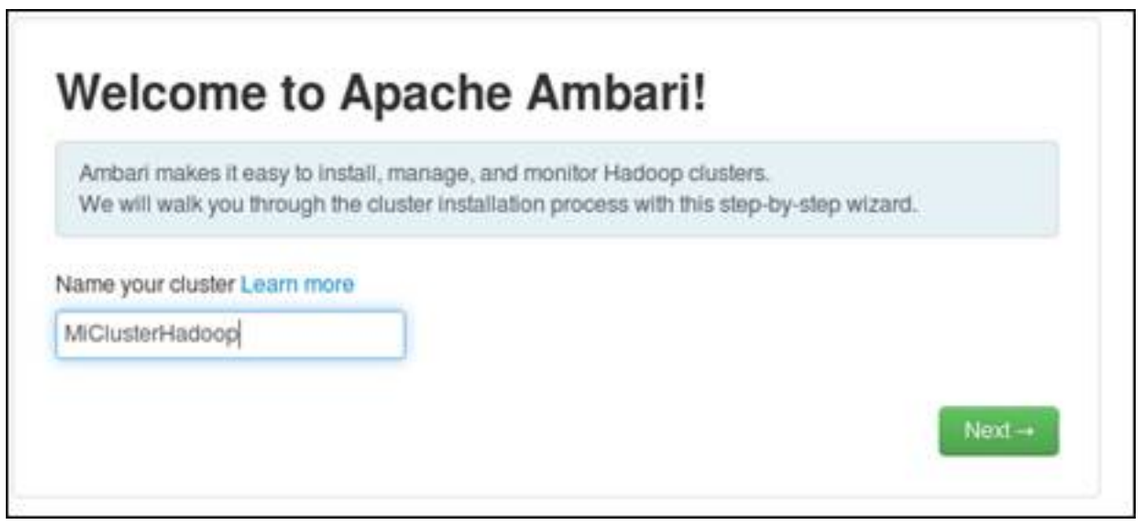
```
sudo ambari-server start
Using python  /usr/bin/python2.6
Starting ambari-server
Ambari Server running with 'root' privileges.
Server PID at: /var/run/ambari-server/ambari-server.pid
Server out at: /var/log/ambari-server/ambari-server.out
Server log at: /var/log/ambari-server/ambari-server.log
Ambari Server 'start' completed successfully.
```

Comprobamos que está corriendo con:

```
ps -aef | grep ambari
root      1926     1  4 04:00 pts/0    00:00:22 /usr/jdk64/jdk1.6.0_31/bin/java
```

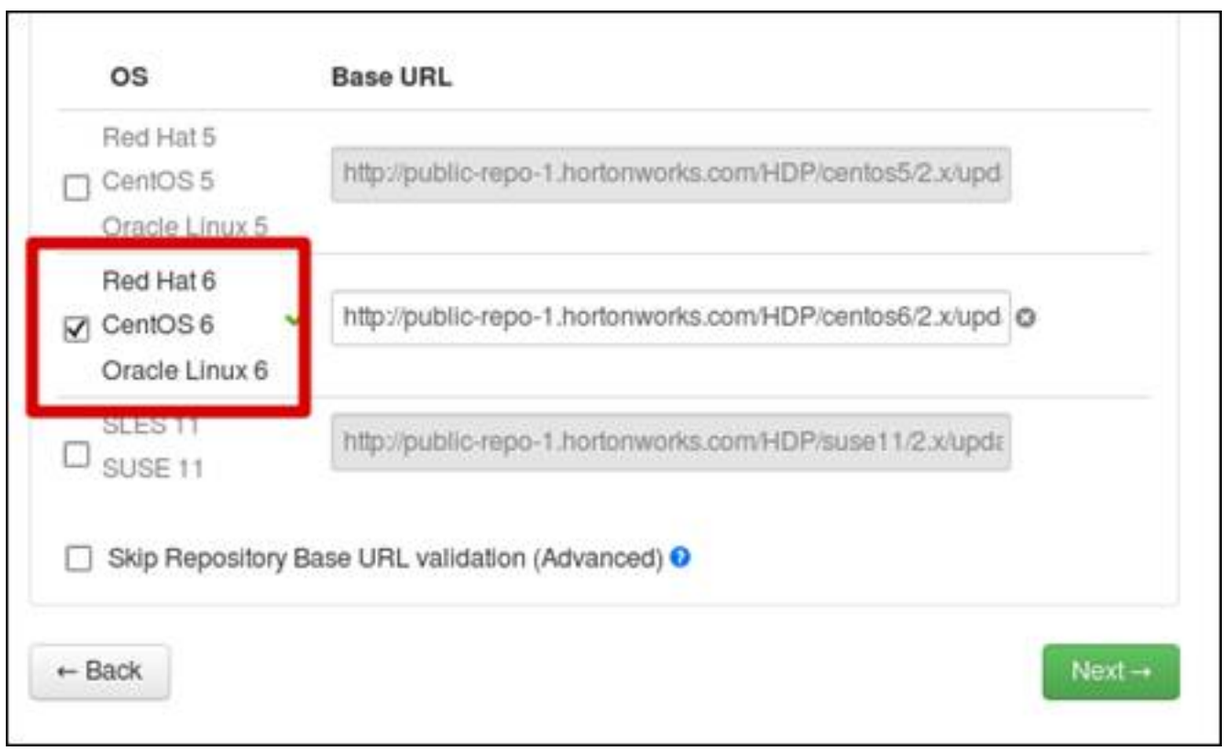
Este proceso se levantará cada vez que arranquemos la instancia por lo que nos olvidamos de él. Ya podemos entrar en la ventana de configuración desde cualquier navegador en: http://****:8080/#/login (usuario y pass: admin/admin luego se podrá cambiar), donde en lugar de los asteriscos tendrás que indicar la IP del nodo Ambari. Si en la política de grupos de seguridad de la instancia has puesto “Source=Anywere”, podrás acceder también desde Windows.

3. Una vez en este paso configurar Ambari es sencillo siguiendo el asistente web que se nos proporciona desde el navegador. Antes de seguir, aseguraros de tener todas las instancias EC2 levantadas.
   Proporcionamos un nombre para el cluster (cualquiera vale):



Seleccionamos la versión de Hadoop a usar. Recomiendo por sencillez la 1.3.3, aunque la 2.0.6 exige menos requisitos hardware:

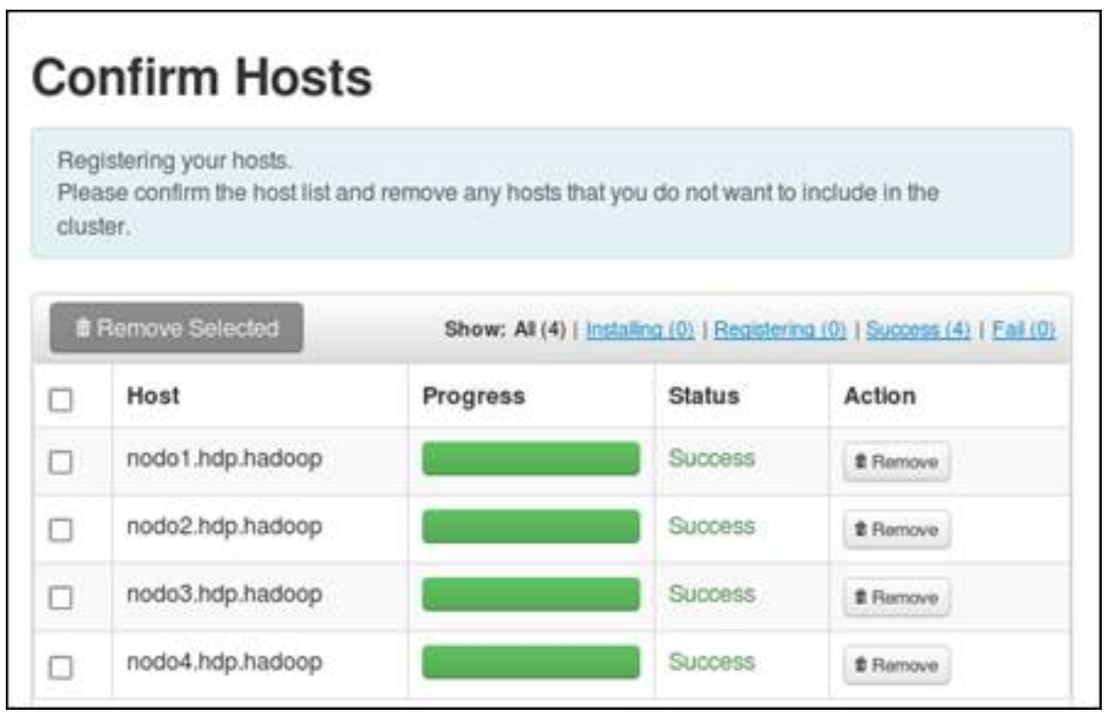
También ayudamos a Ambari indicando el tipo de distribución Linux que tenemos instalada (así las cosas irán más rápidas):



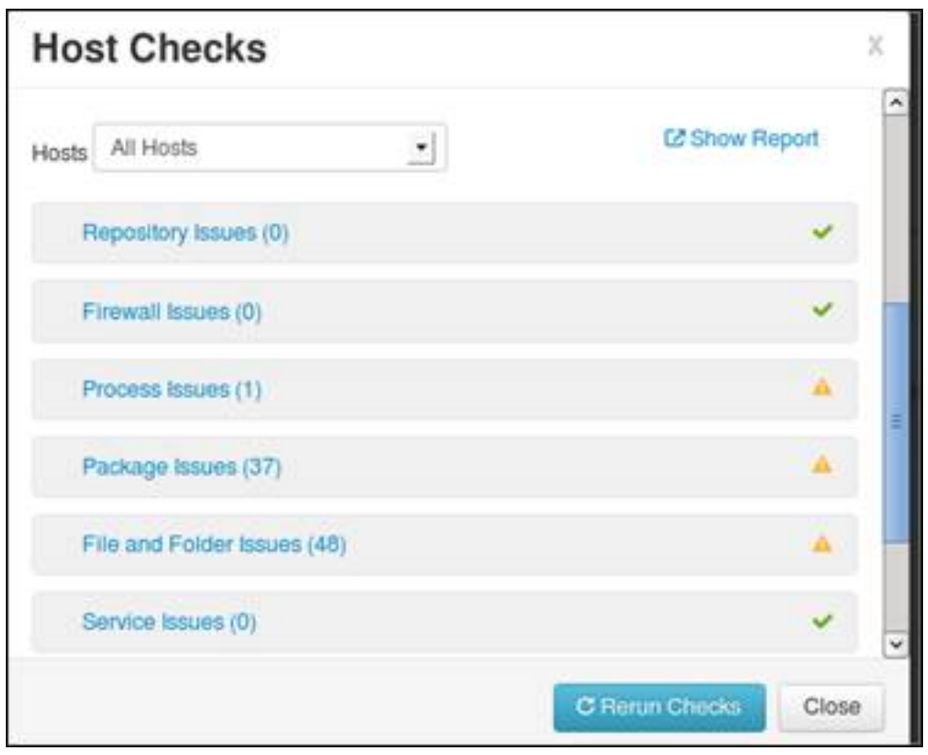
Indicamos la lista de nodos que formarán nuestro cluster (sin incluir el nodo donde corre Ambari). El formato del nombre debe ser el mostrado en la imagen (xxx.hdp.hadoop):

Cargamos el fichero .pem con la clave e indicamos la cuenta con permisos de root:

A continuación se procederá a registrar cada nodo automáticamente:



Si habéis tenido que repetir el proceso más de una vez puede que no tengáis limpios los nodos de instalaciones anterioes por lo que os saldrán warnings:



Lanzando este comando en los nodos con warnings y dando a “Rerun Checks” se soluciona el problema:

```
1 sudo python /usr/lib/python2.6/site-packages/ambari_agent/HostCleanup.py --ski
```

Seleccionamos los servicios a instalar. Como mínimo: HDFS + YARN (o MapReduce1 según la versión que elijáis).

El servicio ZooKeeper no es necesario instalarlo en la versión 1.3.3 pero sí los servicios de monitorización Nagios y Ganglia. En MapReduce2 (YARN) sólo Zookeeper es obligatorio instalarlo.
No os preocupéis por estas combinaciones, ya que Ambari nos alertará sugiriendo las opciones más adecuadas según el caso:

Choose Services

Choose which services you want to install on your cluster.

Service all | none	Version	Description
☑ HDFS	2.2.0.2.0.6.0	Apache Hadoop Distributed File System
☑ YARN + MapReduce2	2.2.0.2.0.6.0	Apache Hadoop NextGen MapReduce (YARN)
☐ Nagios	3.5.0	Nagios Monitoring and Alerting system
☐ Ganglia	3.5.0	Ganglia Metrics Collection system
☑ Hive + HCat	0.12.0.2.0.6.1	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
☐ HBase	0.96.1.2.0.6.1	Non-relational distributed database and centralized service for configuration management & synchronization
☑ Pig	0.12.0.2.0.6.1	Scripting platform for analyzing large datasets
☑ Sqoop	1.4.4.2.0.6.1	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases

Distribuimos los servicios maestros en los distintos nodos que disponemos. Los servicios Namenode y SNameNode deben ir en el mismo nodo, si no tendréis muchos problemas. Para el resto de servicios se aconseja tener 1 servicio por nodo para no saturar el mismo nodo:

Assign Masters

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

NameNode: nodo1.hdp.hadoop (590.3 MB,

SNameNode: nodo3.hdp.hadoop (590.3 MB,

JobTracker: nodo2.hdp.hadoop (590.3 MB,

nodo1.hdp.hadoop (590.3 MB, 1 cores)
NameNode

nodo2.hdp.hadoop (590.3 MB, 1 cores)
JobTracker

nodo3.hdp.hadoop (590.3 MB, 1 cores)
SNameNode

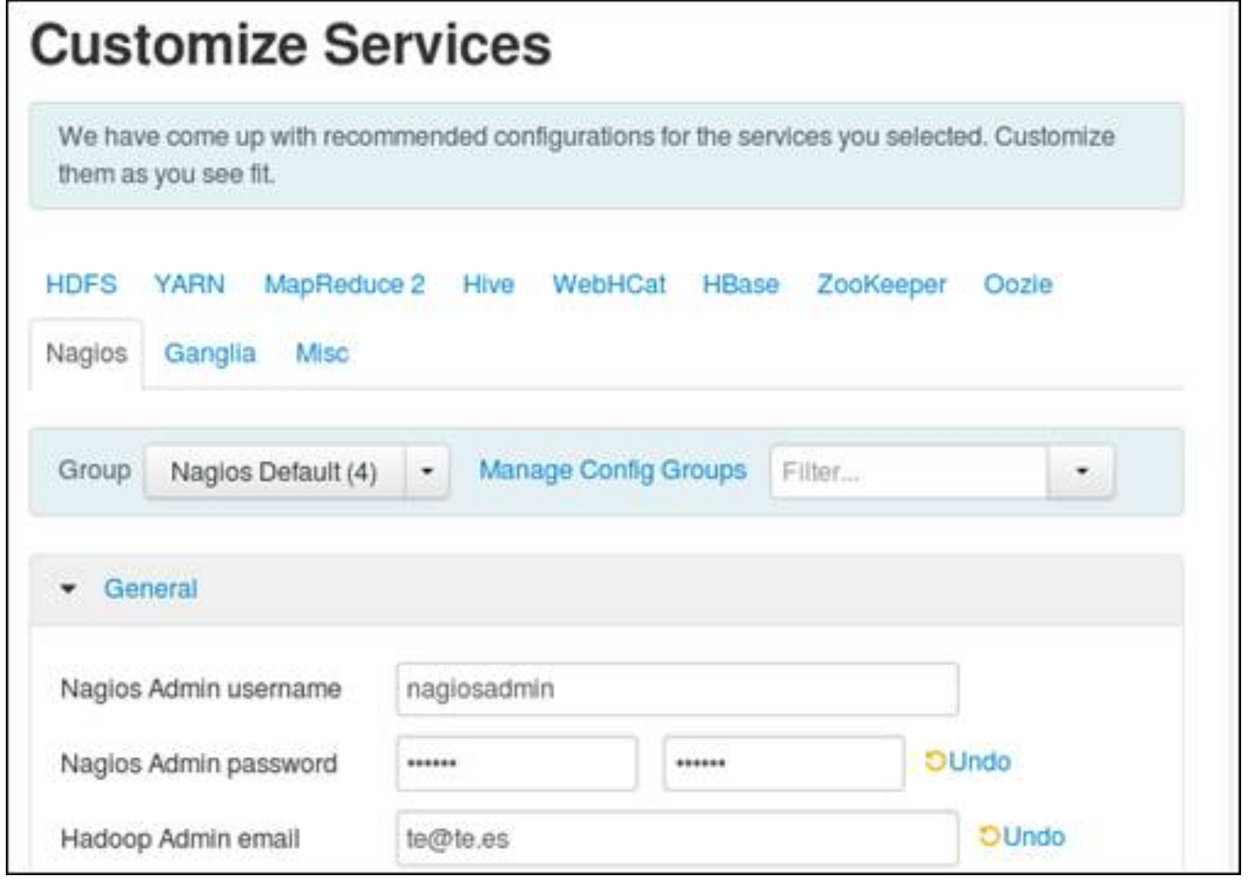
Distribuimos los nodos esclavos en el cluster. Realmente con 1 esclavo de cada tipo es suficiente para comenzar. Posteriormente se pueden agregar más esclavos. Recomiendo no abusar de las pretensiones de nuestro cluster, ya que las instancias EC2 están bastante limitadas en recursos hardware.

Assign Slaves and Clients

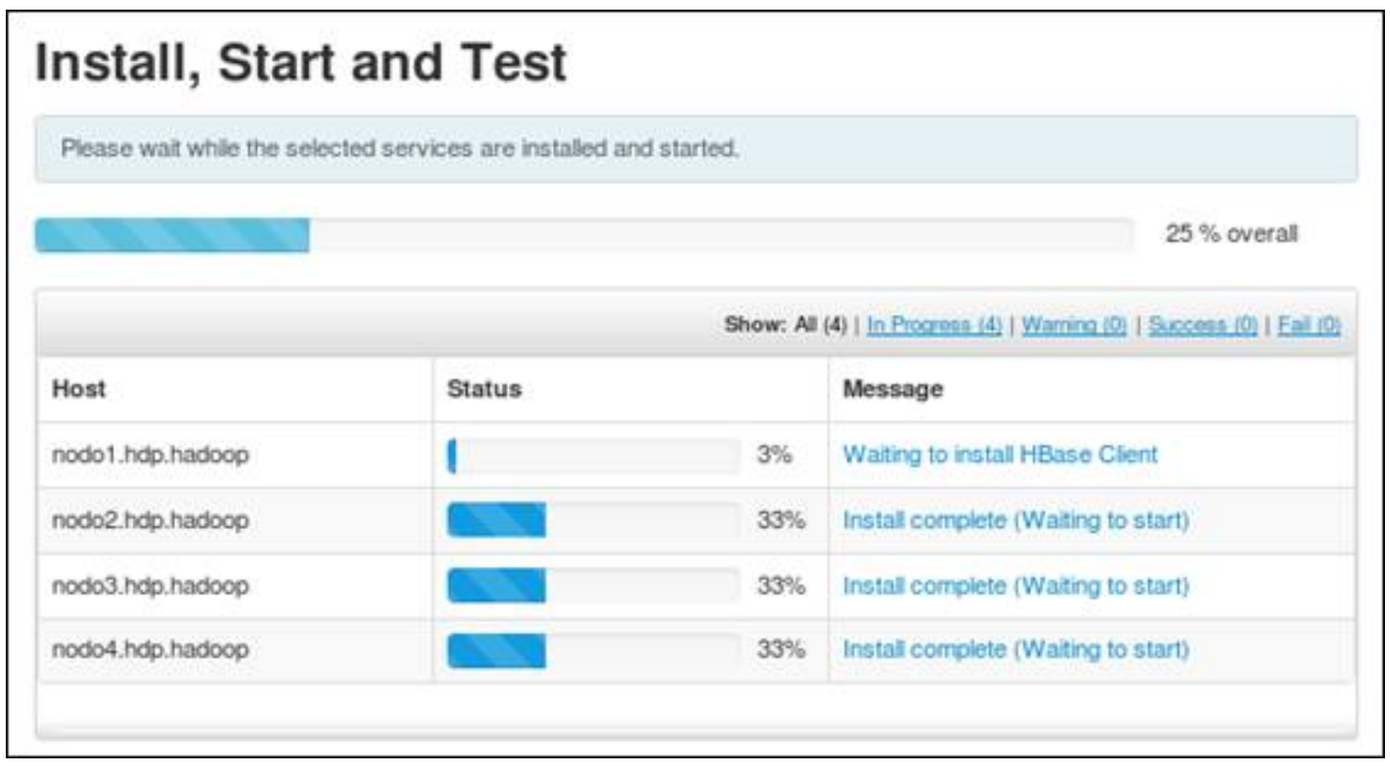
Assign slave and client components to hosts you want to run them on.
Hosts that are assigned master components are shown with *.
"Client" will install HDFS Client and MapReduce Client.

Host	all | none	all | none	all | none
nodo1.hdp.hadoop *	☑ DataNode	☑ TaskTracker	☐ Client
nodo2.hdp.hadoop *	☑ DataNode	☑ TaskTracker	☐ Client
nodo3.hdp.hadoop *	☑ DataNode	☑ TaskTracker	☐ Client
nodo4.hdp.hadoop	☑ DataNode	☑ TaskTracker	☑ Client

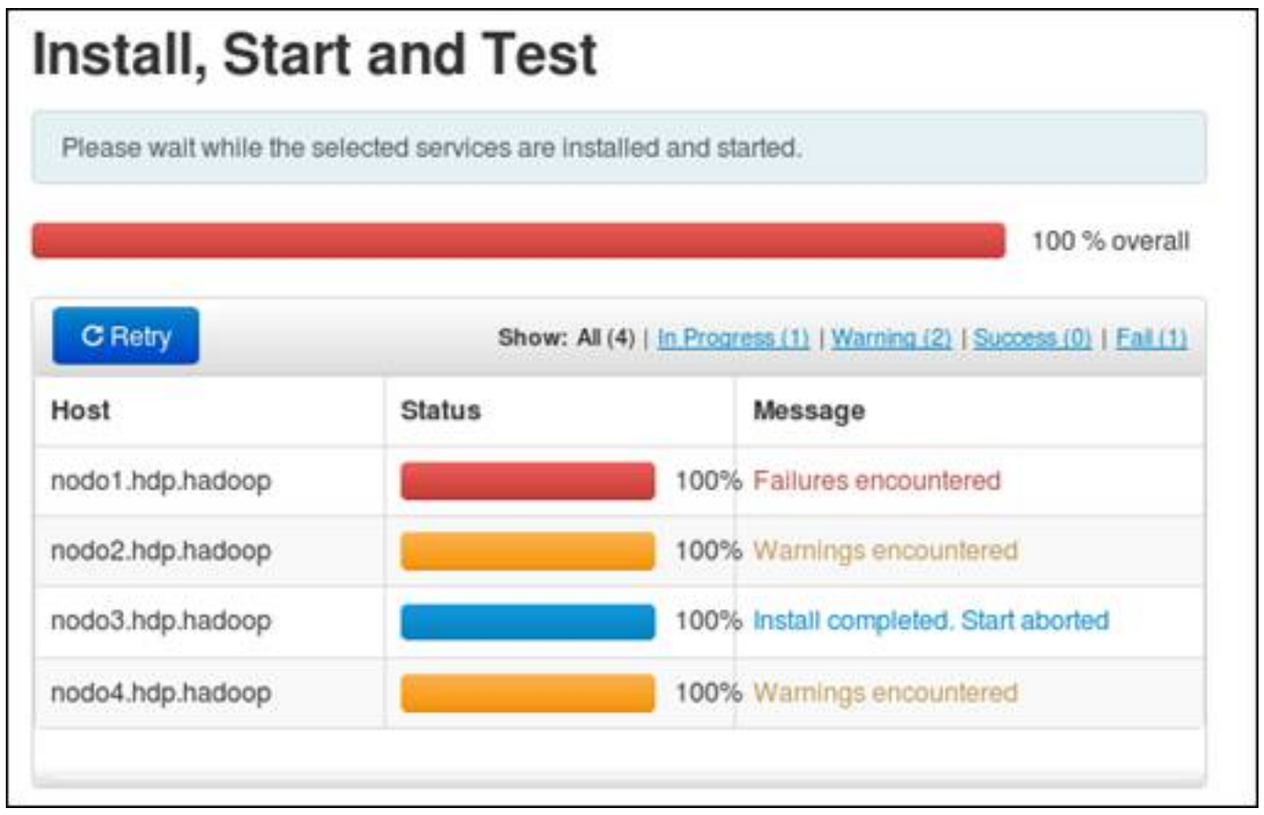
Asignar usuario y clave para los servicios Nagios, Hive y Ozzie (si procede):



Pulsar en “Deploy”:



Este paso llevará un tiempo y es el más crítico y delicado de todos. Básicamente lo que hace ahora Ambari es lanzar un conjunto de scripts Python en cada máquina para configurar los servicios maestro/esclavo. Como véis no hay magia, ya que es todo un poco rudimentario.
Esta fase comprende 2 etapas en cascada: la instalación de los servicios y el inicio de los mismos. Si falla la primera en cualquier nodo, no se continuará con la segunda etapa. Si durante el inicio de los servicios falla algún maestro, no arrancarán los esclavos de los que depende (como es lógico). El maestro más importante es el NameNode:



En caso de fallo puedes ver el error por la misma interfaz de Ambari, o mejor yendo a cada directorio de logs del nodo que ha fallado que son:

```
Servidor Ambari      :/var/log/ambari-server/
Nodo X               : /var/log/hadoop/hdfs
```

Si falla el repositorio de paquetes verás el siguiente error:

```
'/usr/bin/yum -d 0 -e 0 -y install XXXXXX' returned 1: Error: database disk im
```

…tendrás que limpiar el repositorio tal y como se indica abajo y pulsar en “Retry”:

```
sudo yum clean all
```

Si al arrancar el NameNode ves en el log este error (puede darse dependiendo de la distribución de maestros elegida):

```
2015-01-20 05:41:49,118 ERROR org.apache.hadoop.hdfs.server.namenode.NameNode:
```

…tienes que cambiar la IP pública del nodo indicado en el error, en el fichero /etc/hosts para que usar la IP privada (Private IPs del EC2).

```
#54.154…..   nodo1.hdp.hadoop    nodo1
172.31…..    nodo1.hdp.hadoop    nodo1
```

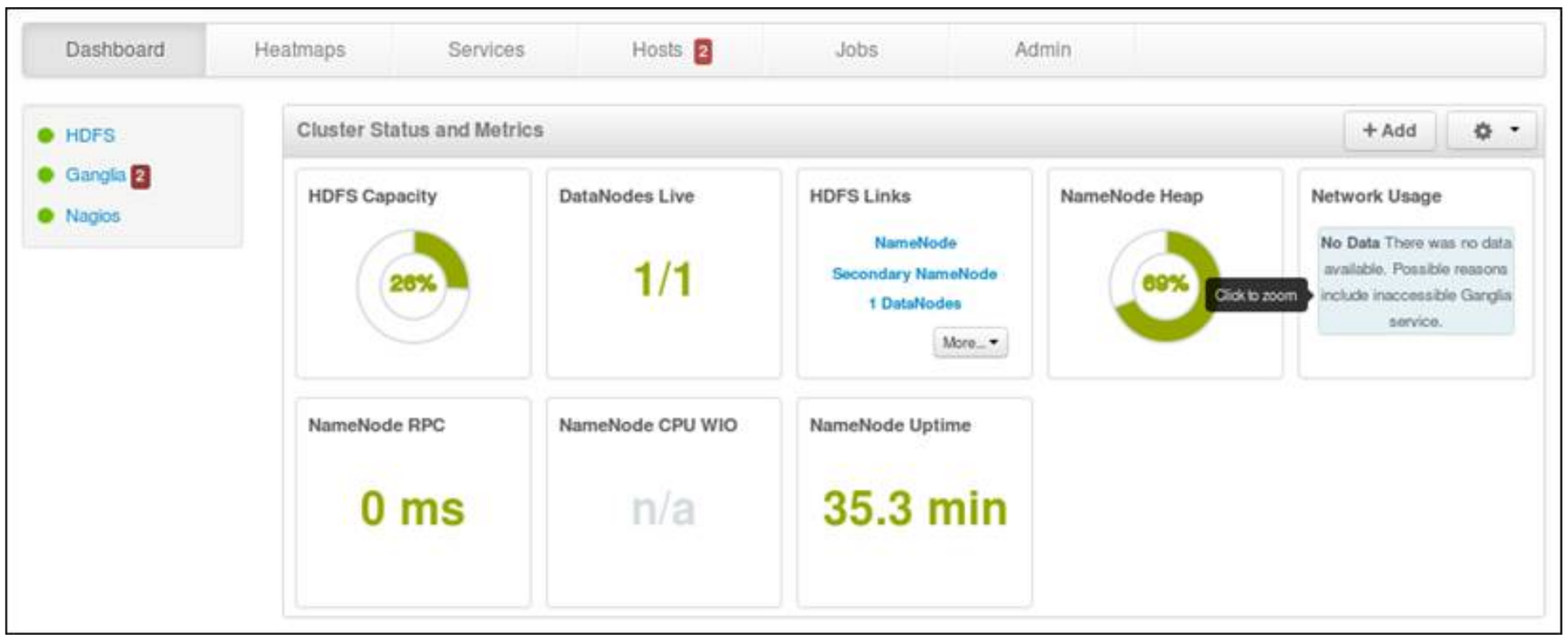
Si estás en un callejón sin salida y quieres volver a empezar la instalación Ambari desde cero, lanza estos comandos que borran toda la configuración en el servidor Ambari:

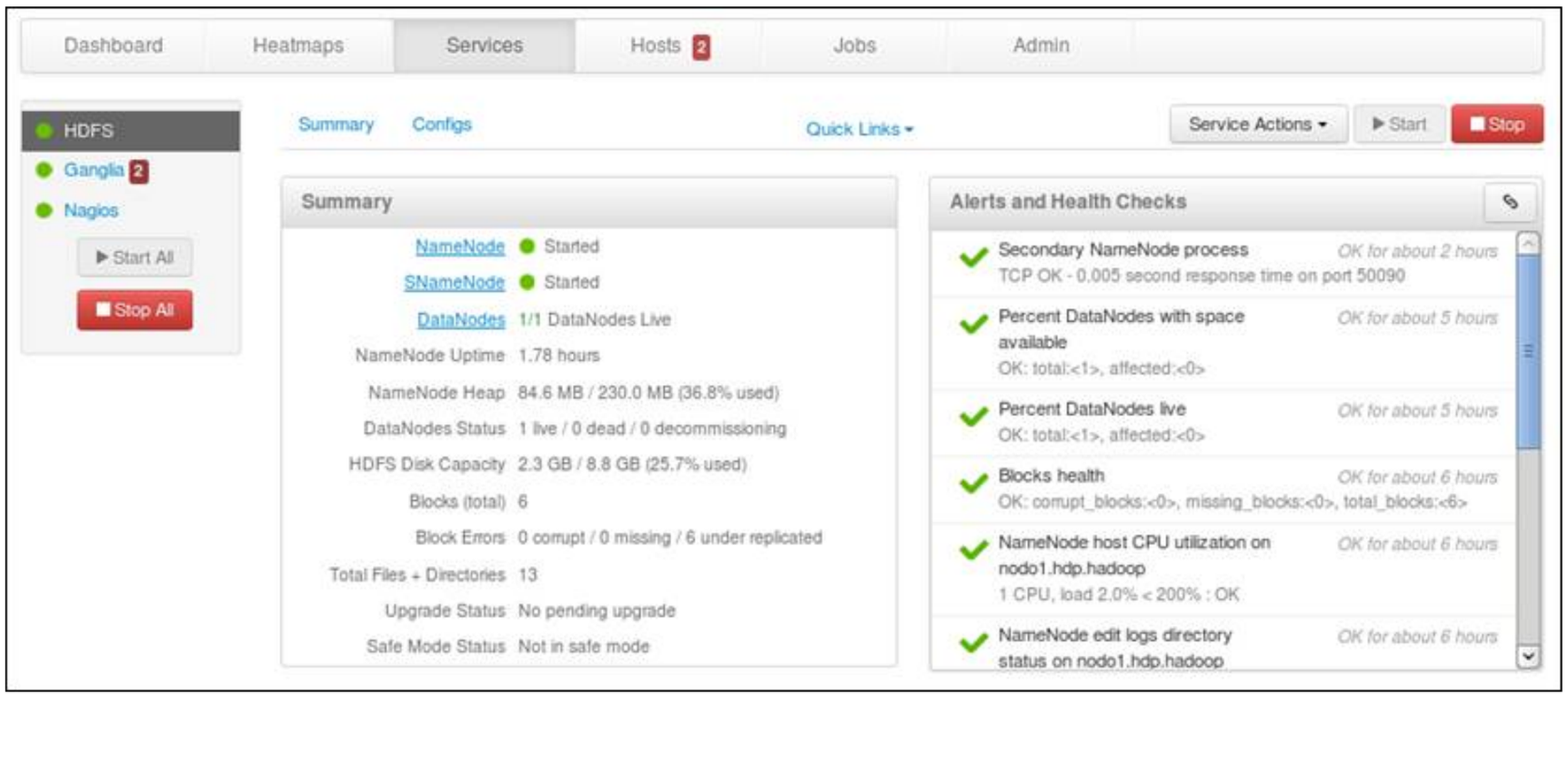
```
sudo ambari-server stop
sudo ambari-server reset
sudo ambari-server start
```

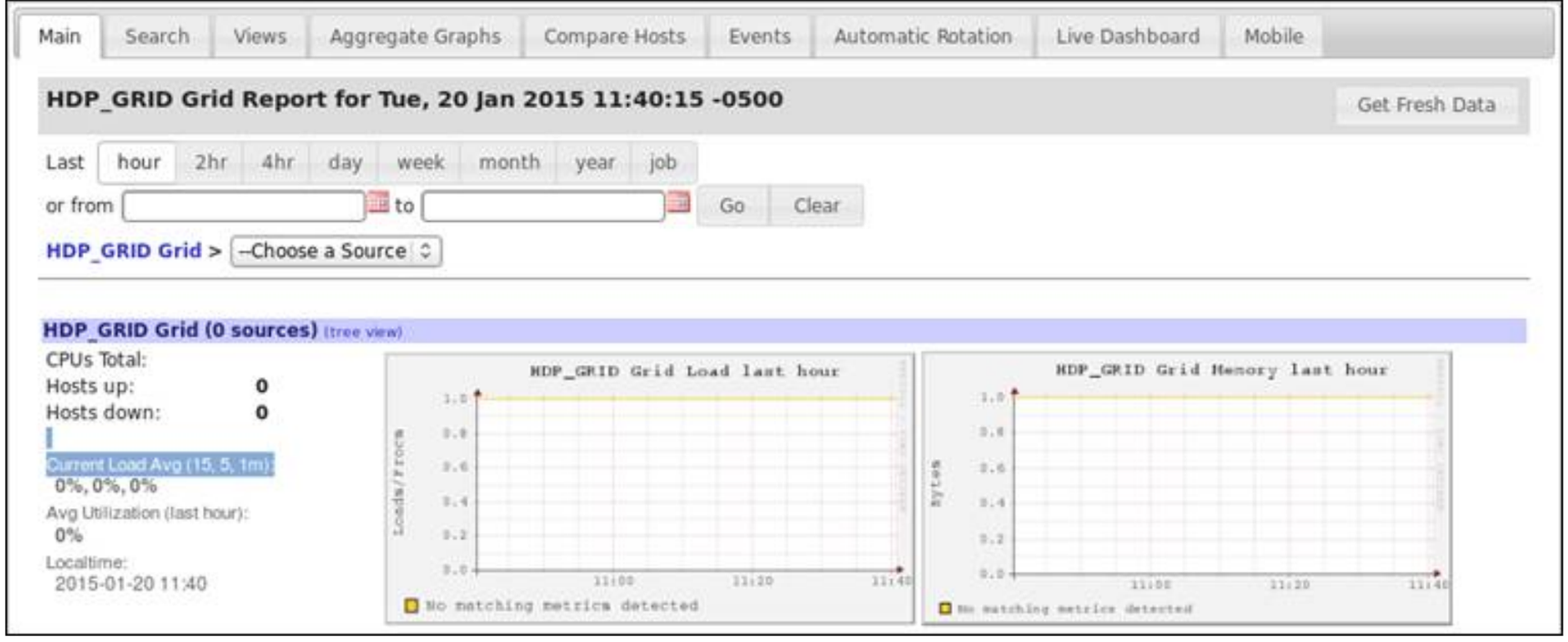
Si todo ha ido bien verás que los servicios se han arrancado correctamente. Realmente el servicio más importante para Ambari es el NameNode ya que controla el HDFS, y permite que el proceso de configuración pueda continuar sin el resto de servicios (claro está, aparecerán warnings en Ambari y además no podrás lanzar trabajos):

Finalmente este será el aspecto de la página principal de administración de Ambari. El trabajo de despliegue ha terminado.
Desde esta pantalla podemos realizar tareas adicionales de monitorizarión/administración como por ejemplo:

- Agregar/eliminar servicios esclavos (tasktracker, datanode, etc.).
- Iniciar/Detener servicios del cluster.
- Agregar/eliminar/mover nodos al cluster.
- Crear alarmas/informes personalizados (uso de disco,memoria,red por nodo,job,tarea, etc.).
- Acceder al sistema HDFS.
- Monitorizar los trabajos de cada usuario en el cluster.
- Acceder a Ganglia-Mobile vía smartphone.





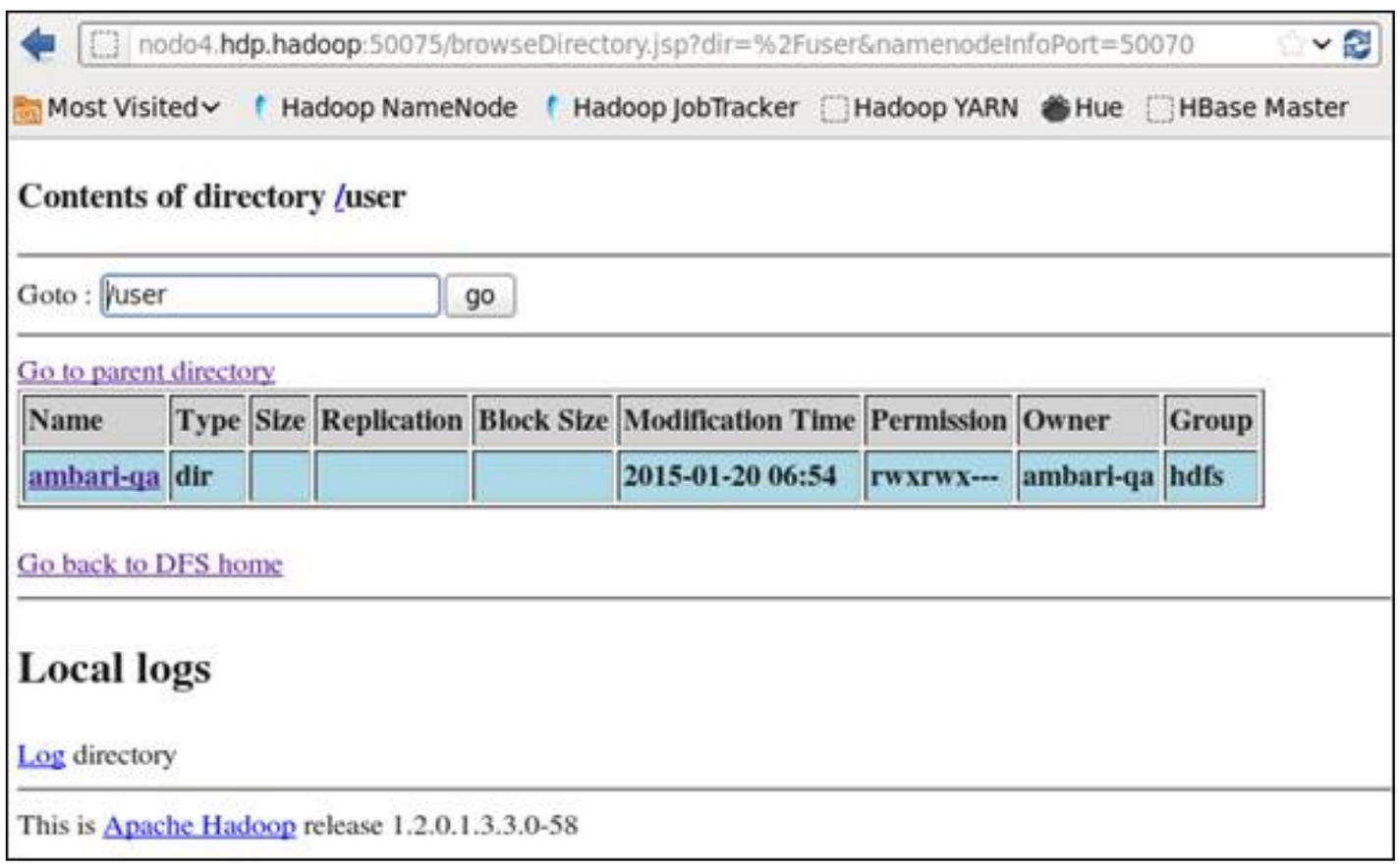


Para poder acceder al servidor de archivos HDFS vía web (Namenode) y ver el contenido sin este error:

```
1 org.apache.hadoop.security.AccessControlException: Permission denied: user=gop
```

…tendrás que dar permisos en HDFS al usuario “gopher” del servidor web:

```
1 sudo -u hdfs hadoop fs -chown gopher /
```



Conclusiones.

Ambari puede ayudar a controlar clusters que cambian continuamente o que deben ser manejamos por equipos sin formación técnica específica. Encuentro interesante el módulo de definición de informes y alertas. También la posibilidad de mover un servicio de nodo lo encuentro útil (aunque no está del todo automatizado).

Como puntos débiles he encontrado el proceso de instalación y despliegue bastante frágil, y requiere conocimientos adicionales para poder salir del problema. También la interfaz presenta algunos fallos relacionados con el estado real de cada servicio así como los warnings mostrados. La documentación oficial publicada en la web de HortonWorks está limitada y algunos de los problemas que la comunidad de usuarios publican en el foro de soporte (http://hortonworks.com/community/forums) siguen sin solución.

Puedes encontrar más información en:

- AWS: https://aws.amazon.com/es/documentation/
- Ambari: http://docs.hortonworks.com/HDPDocuments/
- Hadoop: http://hadoop.apache.org/docs/stable/

Si te ha gustado este tutorial y quieres seguir aprendiendo sobre Hadoop puedes enviarme sugerencias del siguiente tutorial a mi correo. Gracias.

Saludos.

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «