

Relazione  
Creazione di un cluster  
Calcolo Distribuito e Sistemi ad alte Prestazioni

Manuel Severi

27 Novembre 2019



Università di Perugia



Autore: Severi Manuel

# Indice

<b>1</b>	<b>Introduzione</b>	<b>1</b>
<b>2</b>	<b>Fase iniziale progetto</b>	<b>4</b>
<b>3</b>	<b>Configurazione del Firewall</b>	<b>6</b>
<b>4</b>	<b>Configurazione e run Corosync e Pacemaker</b>	<b>6</b>
<b>5</b>	<b>configurazione DRDB</b>	<b>8</b>
<b>6</b>	<b>Creazione cluster</b>	<b>9</b>
<b>7</b>	<b>Verifica delle configurazioni</b>	<b>11</b>
7.1	Corosync . . . . .	11
7.2	Pacemaker . . . . .	12
<b>8</b>	<b>Creazione di un IP pubblico</b>	<b>14</b>
<b>9</b>	<b>Instalazione Apache2 e sue risorse</b>	<b>15</b>
<b>10</b>	<b>configurazione CRM</b>	<b>16</b>
<b>11</b>	<b>Prove finali di comunicazione</b>	<b>17</b>
<b>12</b>	<b>Conclusioni</b>	<b>21</b>

# 1 Introduzione

In informatica un computer cluster, o più semplicemente un cluster (dall'inglese grappolo), è un insieme di computer connessi tra loro tramite una rete telematica. Scopo del cluster è distribuire un'elaborazione molto complessa tra i vari computer, aumentando la potenza di calcolo del sistema e/o garantendo una maggiore disponibilità di servizio, a prezzo di un maggior costo e complessità di gestione dell'infrastruttura: per essere risolto il problema che richiede molte elaborazioni viene infatti scomposto in sottoproblemi separati i quali vengono risolti ciascuno in parallelo. Inoltre un altro beneficio che se ne può trarre è la sensibile diminuzione di time consuming computazionale, cosicché il costo computazionale totale viene suddiviso a tutti i computer interconnessi tra di loro.

I cluster hanno le seguenti caratteristiche: i vari computer risultano come una singola risorsa computazionale e le varie componenti sono risorse dedicate al funzionamento dell'insieme; il server cluster è quindi un server ad altissime prestazioni poiché, invece di gravare su un'unica macchina standalone, suddivide il carico di lavoro (quindi, ad esempio, funzioni di mail server, web server, database server e file server) su più macchine venendo ad essere di fatto una forma di sistema distribuito. Nell'architettura cluster un nodo è una macchina elaborativa ovvero un server fisico o virtuale che prende parte al grappolo. Per l'utente o i client, il cluster è assolutamente trasparente: tutta la notevole complessità hardware e software è mascherata; i servizi vengono erogati, i dati sono resi accessibili e le applicazioni elaborate come se fossero tutte provenienti da un solo mega computer centrale.

Si può chiaramente notare che in cluster i nodi hanno un ruolo fondamentale, infatti il cluster è proprio composto da diversi nodi.

All'interno del cluster, ogni nodo è un sistema indipendente, con un proprio sistema operativo e una memoria privata. Ad oggi i cluster vengono utilizzati dalle più importanti aziende tra le quali *Amazon*, *Ebay*, *Oracle*.

Diversi sono i tipi di cluster. Nello specifico abbiamo:

- **High Availability or Failover Cluster** il funzionamento delle macchine è continuamente monitorato e quando uno dei due host smette di funzionare un'altra macchina subentra in attività. Lo scopo è garantire dunque un servizio continuativo garantendo cioè alta disponibilità di servizio grazie all'alta affidabilità dovuta alla tolleranza ai guasti del sistema cluster per effetto della ridondanza di apparati;
- **Load Balancing Clusters** è un sistema nel quale le richieste di lavoro sono inviate alla macchina con meno carico di elaborazione distribuendo/bilanciando così il carico di lavoro sulle singole macchine. Questo garantisce tempi minori di processamento di un servizio e minore affaticamento di una macchina;
- **High Performance Computing (HPC Cluster)** i computer sono configurati per fornire prestazioni estremamente alte. Le macchine suddividono i processi di un job su più macchine, al fine di guadagnare in prestazioni. La peculiarità saliente è che i processi sono parallelizzati e che le routine che possono girare separatamente saranno distribuite su macchine differenti invece di aspettare di essere eseguite sequenzialmente una dopo l'altra. GLI HPC sono diffusi specialmente nei Centri di Elaborazione Dati (CED);

La scelta di adoperare e utilizzare un cluster per smaltire il calcolo computazionale è dettato da alcune esigenze:

- un ambiente distribuito è scalabile, e può essere facilmente incrementato per venire incontro a nuove esigenze computazionali;
- la richiesta di risorse computazionali può essere distribuita fra un insieme di macchine invece di essere confinata ad un singolo sistema, eliminando i cosiddetti "colli di bottiglia" e fornendo, quindi, migliori prestazioni;
- la disponibilità delle risorse e dei servizi viene fortemente incrementata.
- Lo sfruttamento della cooperazione per risolvere problemi complessi.
- Disponibilità di un gran numero di software Open Source per i cluster, come MOSIX, openMosix e Beowulf.

Insieme ai vantaggi che si percepiscono con queste architetture, nascono anche una serie di svantaggi da tenere in considerazione:

- l'amministrazione di decine o centinaia di sistemi è molto più onerosa di quella del singolo sistema;
- la gestione della sicurezza cresce esponenzialmente all'aumentare degli host presenti nella rete;
- la gestione di risorse condivise via rete, è più complessa rispetto a quella locale.

## 2 Fase iniziale progetto

L'obiettivo del progetto è quello di creare un cluster attivo-passivo. Per fare ciò ho installato nel mio computer **VMvirtualbox** che mi ha permesso di creare all'interno due macchine virtuali: una attiva (*chiamata ubuntu-server1*), una passiva (*chiamata ubuntu-server2*). Non avendo a disposizione un computer molto potente, ho installato in entrambe le macchine virtuali ubuntu 18.04.03 live server, proprio perchè rispetto ad altri sistemi operativi è molto più leggero.

Una volta installato Ubuntu su entrambe le macchine, ho controllato gli ip associati ad esse tramite il comando *ifconfig*:

### Macchina virtuale 1

Nome: *server1*

Indirizzo ip: *192.168.56.105*

### Macchina virtuale 2

Nome: *server2*

Indirizzo ip: *192.168.56.106*

Successivamente ho installato i pacchetti necessari per la creazione del cluster e per il funzionamento dello stesso.

Nello specifico si installa il pacemaker che è un gestore di risorse del cluster, il corosync, e il drbd (Distributed Replicated Block Device) che è un device a blocchi replicato sulla rete.

Di seguito vengono mostrati i comandi che mi hanno permesso di installare i vari pacchetti e lo screen che fa riferimento agli stessi.

**NB: i seguenti pacchetti sono stati installati in entrambe le macchine**

```
sudo apt-get install pacemaker corosync pcs
```

```
sudo apt-get install drbd8-utils
```

```
sudo apt-get install apache2
```

```
root@server2:/home/manuel# sudo apt-get install pacemaker corosync pcs
Reading package lists... Done
Building dependency tree
Reading state information... Done
pcs is already the newest version (0.9.164-1).
corosync is already the newest version (2.4.3-0ubuntu1.1).
corosync set to manually installed.
pacemaker is already the newest version (1.1.18-0ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
root@server2:/home/manuel# sudo apt-get install drbd8-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
drbd8-utils is already the newest version (2:8.9.10-2).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
root@server2:/home/manuel# sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.29-1ubuntu4.11).
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
root@server2:/home/manuel# _
```

nella seguente schermata il download è già avvenuto infatti controlla se sono disponibili aggiornamenti

ora sincronizzo i tempi tra i vari server tramite i comandi:

```
sudo dpkg-reconfigure tzdata
```

```
sudo apt-get update
```

```
sudo apt-get -y install ntp
```

### 3 Configurazione del Firewall

dopo aver installato tutti i pacchetti necessari procedo a configurare il firewall aprendo le porte 5404, 5405, 5406 sia in input che in output. A seguire l'esempio con il codice utilizzato per la porta 5404:

```
sudo ufw allow 5404
sudo iptables -A INPUT -i eth1 -p udp -m multiport --dports 5404
-m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
sudo iptables -A OUTPUT -o eth1 -p udp -m multiport --sports 5404
-m conntrack --ctstate ESTABLISHED -j ACCEPT
```

### 4 Configurazione e run Corosync e Pacemaker

dopo aver installato i pacchetti come sopra descritto, si passa alla loro configurazione andando a modificare il file *corosync.conf* nella directory */etc/corosync/* andando a scrivere quello riportato qua sotto:

```
totem {
  version: 2
  cluster_name: lbcluster
  transport: udpu
  interface {
    ringnumber: 0
    bindnetaddr: 192.168.56.255
    broadcast: yes
    mcastport: 5405
  }
}
quorum {
  provider: corosync_votequorum
  two_node: 1
}
nodelist {
  node {
    ring0_addr: 192.168.56.102
    name: serverone
    nodeid: 1
  }
  node {
    ring0_addr: 192.168.56.103
    name: servertwo
    nodeid: 2
  }
}
logging {
  to_logfile: yes
  logfile: /var/log/corosync/corosync.log
  to_syslog: yes
  timestamp: on
}
```



dopo averlo configurato correttamente lo rendo *ENABLE* passando alla configurazione del Pacemaker. Esclusivamente sul server1 eseguo i seguenti programmi per ottenere un indirizzo ip pubblico:

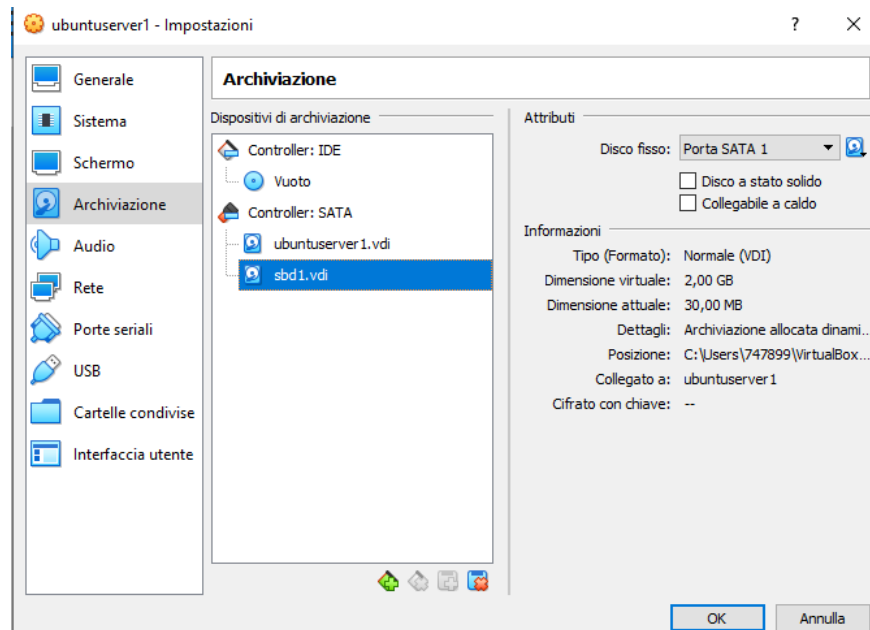
```
sudo crm configure property stonith-enabled=false
sudo crm configure property no-quorum-policy=ignore

sudo crm configure primitive virtual_public_ip
ocf:heartbeat:IPAddr2 params ip="192.168.56.255"
cidr_netmask="32" op monitor interval="10s"
meta migration-threshold="2" failure-timeout="60s"
resource-stickiness="100"
```

una volta eseguiti i seguenti programmi e messi online entrambi i server tramite il comando(come nell esempio per il server1):

```
sudo crm node online serverone
```

bisogna andare a creare due partizioni (della capacità di 2GB) in entrambi i server, andando a modificare le impostazioni della macchina virtuale come in figura.



la partizione è stata creata per salvarci le configurazioni del DRDB

## 5 configurazione DRDB

una volta create le partizioni e spostatoci sulla nuova partizione tramite il comando

```
sudo nano /etc/drbd.conf
```

vado a modificare il file drbd.conf cancellando tutti gli include presenti e andando a scrivere la seguente configurazione

```
global { usage-count no; }
common { syncer { rate 100M; } }
resource r0 {
    protocol C;
    startup {
        wfc-timeout 15;
        degr-wfc-timeout 60;
    }
    net {
        cram-hmac-alg sha1;
        shared-secret "secret";
    }
    on server1{
        device /dev/drbd0;
        disk /dev/sdb1;
        address 192.168.56.102:7788;
        meta-disk internal;
    }
    on server2 {
        device /dev/drbd0;
        disk /dev/sdb1;
        address 192.168.56.103:7788;
        meta-disk internal;
    }
}
```

se tutto è andato a buon fine si otterrà il seguente risultato

```
Every 1.0s: cat /proc/drbd                                server2: Tue Nov 26 13:54:49 2019
version: 8.4.10 (api:1/proto:86-101)
srcversion: 151110056BF899E7D986DDD
0: cs:SyncTarget ro:Secondary/Primary ds:Inconsistent/UpToDate C r-----
   ns:0 nr:1539772 dw:1539772 dr:0 al:16 bm:0 lo:0 pe:0 ua:0 ap:0 ep:1 wo:f oos:556256
   [=====>.....] sync'ed: 73.7% (556256/2096028)K
   finish: 0:00:33 speed: 16,764 (16,920) want: 9,120 K/sec
```

## 6 Creazione cluster

Una volta finito di configurare il Firewall, si è installato il cluster software in entrambe le macchine tramite il comando

```
sudo apt-get install pcs -y
```

Viene creato un utente cluster al quale viene assegnata una password; la password scelta Root(pur sapendo che è considerata una password debole, ma siamo in ambito accademico e verrà usata la stessa ogni volta viene chiesta una password) deve essere la stessa in entrambi i nodi, ovvero la stessa sia su *server1* che su *server2*

Una volta impostata la password bisognerà eseguire questi comandi:

```
systemctl start pcsd.service
```

```
systemctl enable pcsd.service
```

Nello specifico questi due comandi vanno a startare e ad abilitare il pcs. Adesso si dovranno autorizzare entrambi i nodi tramite il comando:

```
pcs cluster auth 192.168.56.105 192.168.56.106 --force
```

Verrà chiesto username e password. Basterà utilizzare come username *hacluster* e la password.

Se il tutto è stato fatto correttamente dovremo avere autorizzato entrambi i nodi, come in figura.

```
root@server1:/home/manuel# pcs cluster auth 192.168.56.105 192.168.56.106 --force
Username: hacluster
Password:
192.168.56.106: Authorized
192.168.56.105: Authorized
root@server1:/home/manuel#
```

esempio eseguito su server1

Successivamente bisognerà creare un cluster, specificando un nome (nel mio caso si chiama *UABLR*) e aggiungere i nostri nodi al cluster appena creato specificando gli indizzi ip dei due nodi.

Questo l'ho fatto tramite il comando

```
pcs cluster setup --start --name UABLR 192.168.56.105
192.168.56.106
```

Adesso bisognerà abilitare il nuovo cluster appena creato ai due nodi. Questo lo si può fare tramite il comando:

```
pcs cluster enable --all
```

Se è stato effettuato tutto correttamente, il comando tornerà due stringhe come riportate di seguito:

```
192.168.56.105: Cluster Enabled
192.168.56.106: Cluster Enabled
```

Adesso, se si va a controllare lo status del cluster, tramite il comando

```
pcs status
```

il risultato deve essere simile a quello in figura perchè la figura rappresente il risultato a fine configurazione.

```
Last change: Wed Nov 27 11:00:25 2019 by root via cibadmin on server1
2 nodes configured
8 resources configured

Online: [ server1 server2 ]

Full list of resources:

Resource Group: gruppo_risorse
ClusterVirtualIp (ocf::heartbeat:IPaddr2):      Stopped
website (ocf::heartbeat:apache):              Stopped
Master/Slave Set: cln_web [webdata]
webdata (ocf::linbit:drbd):
webdata (ocf::linbit:drbd):
web_filesystem (ocf::heartbeat:Filesystem):
Master/Slave Set: drbd_master_slave [drbd_res]
drbd_res (ocf::linbit:drbd):
drbd_res (ocf::linbit:drbd):
fs_res (ocf::heartbeat:Filesystem):      Stopped

Daemon Status:
corosync: active/enabled
pacemaker: active/enabled
pcsd: active/enabled
root@server1:/home/manuel#
```

le risorse sono volutamente stopped in questo caso l'importante è notare che sono presenti due nodi configurati ed entrambi i server sono online

Si nota che i due nodi sono correttamente configurati al cluster.

## 7 Verifica delle configurazioni

### 7.1 Corosync

Ora si controlla che funzioni il *Corosync* grazie al comando:

```
systemctl status corosync
```

ed il risultato deve essere come quello dello screen che segue:

```
root@server1:/home/manuel# systemctl status corosync
● corosync.service - Corosync Cluster Engine
   Loaded: loaded (/lib/systemd/system/corosync.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2019-11-27 15:29:11 CET; 24min ago
     Docs: man:corosync
            man:corosync.conf
            man:corosync_overview
  Main PID: 1025 (corosync)
    Tasks: 2 (limit: 1108)
   CGroup: /system.slice/corosync.service
           └─1025 /usr/sbin/corosync -f
```

se per qualche motivo non risulta in running eseguire questi passaggi in ordine controllando ad ogni passaggio il nuovo stato del Corosync.

1. riavviare tutti i servizi prestando maggior attenzione proprio a Corosync
2. eseguire il reboot della macchina, avendo modificato i file di configurazione il Corosync si potrebbe arrestare
3. controllare che tutti i file di configurazione siano stati correttamente scritti ed effettuare il reboot

## 7.2 Pacemaker

iniziamo controllando i processi di pacemaker in esecuzione tramite il comando sul server 1:

```
ps axf |grep pacemaker
```

```
root@server1:/home/manuel# ps axf |grep pacemaker
12969 tty1      S+   0:00      \_ grep --color=auto pacemaker
1191 ?         Ss   0:00 /usr/lib/pacemaker/lrmd
1193 ?         Ss   0:00 /usr/lib/pacemaker/pengine
```

inoltre controllo che non ci siano problemi nel cluster, e abilito sia il corosync e il pacemaker ad aabilitarsi ed avviarsi ad ogni riavvio del sistema tramite i comandi:

```
systemctl enable corosync
systemctl enable pacemaker
```

così che all'avvio del clustr, registro automaticamente il numero e i dettagli dei nodi del cluster nonché lo stack utilizzato e la versione di Pacemaker utilizzata. per visualizzare tali dati in formato xml usare il comando:

```
pcs cluster cib
```

che restituisce una cosa simile a quella sottostante con piccole variazioni in base alla versione ed al pc su cui gira la macchina

```
root@server1:/home/manuel# pcs cluster cib
<cib_crm_feature_set="3.0.14" validate-with="pacemaker-2.10" epoch="4" num_updates="6" admin_epoch="0" cib-last-written="Tue Nov 26 14:47:24 2019" update-origin="server1" update-client="crmd" update-user="hacluster" have-quorum="1" dc-uuid="1">
  <configuration>
    <crm_config>
      <cluster_property_set id="cib-bootstrap-options">
        <nvpair id="cib-bootstrap-options-have-watchdog" name="have-watchdog" value="false"/>
        <nvpair id="cib-bootstrap-options-dc-version" name="dc-version" value="1.1.18-2b07d5c5a9"/>
        <nvpair id="cib-bootstrap-options-cluster-infrastructure" name="cluster-infrastructure" value="corosync"/>
        <nvpair id="cib-bootstrap-options-cluster-name" name="cluster-name" value="UABLR"/>
      </cluster_property_set>
    </crm_config>
    <nodes>
      <node id="1" uname="server1"/>
      <node id="2" uname="server2"/>
    </nodes>
    <resources/>
    <constraints/>
  </configuration>
  <status>
    <node_state id="1" uname="server1" in_ccm="true" crmd="online" crm-debug-origin="do_state_transition" join="member" expected="member">
      <lrmd id="1">
        <lrmd_resources/>
      </lrmd>
    </node_state>
    <node_state id="2" uname="server2" in_ccm="true" crmd="online" crm-debug-origin="do_state_transition" join="member" expected="member">
      <lrmd id="2">
        <lrmd_resources/>
      </lrmd>
    </node_state>
  </status>
</cib>
```

ora verifichiamo le informazioni del cluster con il seguente comando:

```
crm_verify -L -V
```

notiamo come il terminale ci mostri dei errori facilmente "aggirabili" disabilitando *STONITH* tramite il comando

```
pcs property set stonith-enabled=false
```

ora riprovando il comando per verificare le informazioni molto probabilmente, se non sicuramente, gli errori sono scomparsi.

## 8 Creazione di un IP pubblico

questa parte di relazione va ad analizzare come creare un ip pubblico così da poter essere raggiunti da altre macchine e , nel nostro caso, va eseguita solo sulla macchina principale, ovvero il server 1. Per fare quello appena descritto uso il comando:

```
pcs resource create nome_scelto ocf:heartbeat:IPaddr2  
    ip="192.168.1.200" op monitor interval="30s"
```

seguito subito da questo comando per attivare URL, dare un massimo di tempo per la risposta e specificare il file di configurazione

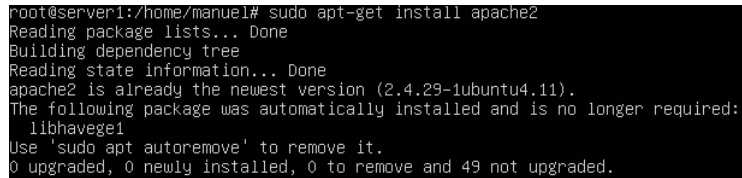
```
sudo pcs di nome-casuale ocf:heartbeat:apache  
    configfile=/etc/apache2.conf  
statusurl="http://127.0.0.1/server-status" op monitor  
    interval="30s"
```



## 9 Instalazione Apache2 e sue risorse

se non fatto in precedenza, si installa apache2 tramite il comando:

```
sudo apt-get install apache2
```



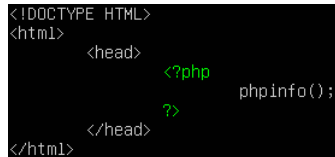
```
root@server1:/home/manuel# sudo apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
apache2 is already the newest version (2.4.29-1ubuntu4.11).
The following package was automatically installed and is no longer required:
  libhavege1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 49 not upgraded.
```

come nelle installazioni precedenti anche questo screen è preso a fine progetto quindi in realtà in questa schermata si vede il computer che cerca aggiornamenti

ora si va a creare un file PHP che stampa le info del PHP stesso sulla directory `/var/www/html/` tramite il comando `sudo` come nell'esempio sottostante

```
sudo nano /var/www/html/index.php
```

il file è decisamente molto semplice ed ha la seguente struttura, creata tramite terminale:



```
<!DOCTYPE HTML>
<html>
  <head>
    <?php      phpinfo();
    ?>
  </head>
</html>
```

e vado constatare come è raggiungibile da l'altro server lo stesso file tramite l'ip pubblico

## 10 configurazione CRM

nel seguente capitolo si va a descrivere come configurare il CRM(Customer Relationship Management), che migliora l'esperienza dei vari client. Iniziamo entrando nella configurazione del crm con il comando

```
crm configure
```

noteremo che il terminale ha subito dei cambiamenti grafici, ma è del tutto normale, visto che si è entrati direttamente nella configurazione di tale servizio. Il primo comando da usare è il seguente

```
primitive drbd_res ocf:linbit:drbd params drbd_resource=0 op
monotor interval=29s role=Master op monitor interval=31s
role=Slave
```

così da specificare il master e lo slave e relativi tempi di risposta. Si prosegue con una serie di comandi quali

```
primitive fs_res ocf:heartbeat:Filesystem params
device=/dev/drbd0 directory=var/www/html fstype=ext4 collocation
fs_drbd_colo INFINITY: fs_res drbd_master_slave:Master order
fs_after_drbd mandatory: drbd_master_slave:promote fs_res:start
```

sono tutti i comandi volti ad ottenere una configurazione ottima del CRM ora andiamo a commitare, tramite la parola chiave `commit` il tutto con le configurazione vecchie, se non ci sono errori, il commit andrà a buon fine altrimenti durante il commit il software controlla se tutti i comandi sono scritti bene e se nessun comando va in conflitto con le impostazioni del server. Contolliamo quello appena fatto tramite il comando

```
crm_mon
```

che permette di monitorare lo stato e la configurazione del cluster. Il suo output include il numero di nodi, uname, uuid, lo stato, le risorse configurate nel cluster e lo stato corrente di ciascuno. L'output di *crm\_mon* può essere visualizzato sulla console o stampato in un file HTML. Di seguito possiamo notare come si presenta la schermata in caso si scelga di stampare su terminale il risultato del comando.

```

node 1: server1
node 2: server2
primitive ClusterVirtualIp IPAddr2 \
    params cidr_netmask=25 ip=10.0.2.15 nic=enp0s3 \
    op monitor interval=30s \
    op start interval=0s timeout=20s \
    op stop interval=0s timeout=20s
primitive drbd_res ocf:linbit:drbd \
    params drbd_resource=0 \
    op monitor interval=29s role=Master \
    op monitor interval=31s role=Slave
primitive fs_res Filesystem \
    params device="/dev/drbd0" directory="/var/www/html/fstype=ext4"
primitive web_filesystem Filesystem \
    params device="/dev/drbd0" directory="/var/www/html" fstype=ext3 \
    op monitor interval=20 timeout=40 \
    op notify interval=0s timeout=60 \
    op start interval=0s timeout=60 \
    op stop interval=0s timeout=60
primitive webdata ocf:linbit:drbd \
    params drbd_resource=webdata \
    op demote interval=0s timeout=90 \
    op monitor interval=10s \
    op notify interval=0s timeout=90 \
    op promote interval=0s timeout=90 \
    op reload interval=0s timeout=30 \
    op start interval=0s timeout=240 \
    op stop interval=0s timeout=100
primitive website apache \
    params configfile="/etc/apache2/apache2.conf" statusurl="http://localhost/server-status" \
    op monitor interval=10s \
    op start interval=0s timeout=40s \
    op stop interval=0s timeout=60s
group gruppo_risorse ClusterVirtualIp website
ms c1n_web webdata \
    meta master-max=1 master-node-max=1 clone-max=2 clone-node-max=1 notify=true
:~

```

Infine andiamo a restartare sia il Corosync che il pacemaker per essere sicuri di applicare le modifiche appena realizzate e controlliamo che lista dei file presenti in `var/www/html/` nottando appunto che sono presenti i nostri file, altrimenti vanno ricreati.

Visto che siamo arrivati alla conclusione della configurazione è consigliato effettuare un reboot.

## 11 Prove finali di comunicazione

controlliamo subito che le due macchine comunichino tra loro tramite un test *ping* e come possiamo vedere dalle schermate sottostanti le due macchine "si vedono" e riescono a comunicare con ritardi di scarsissima entità.

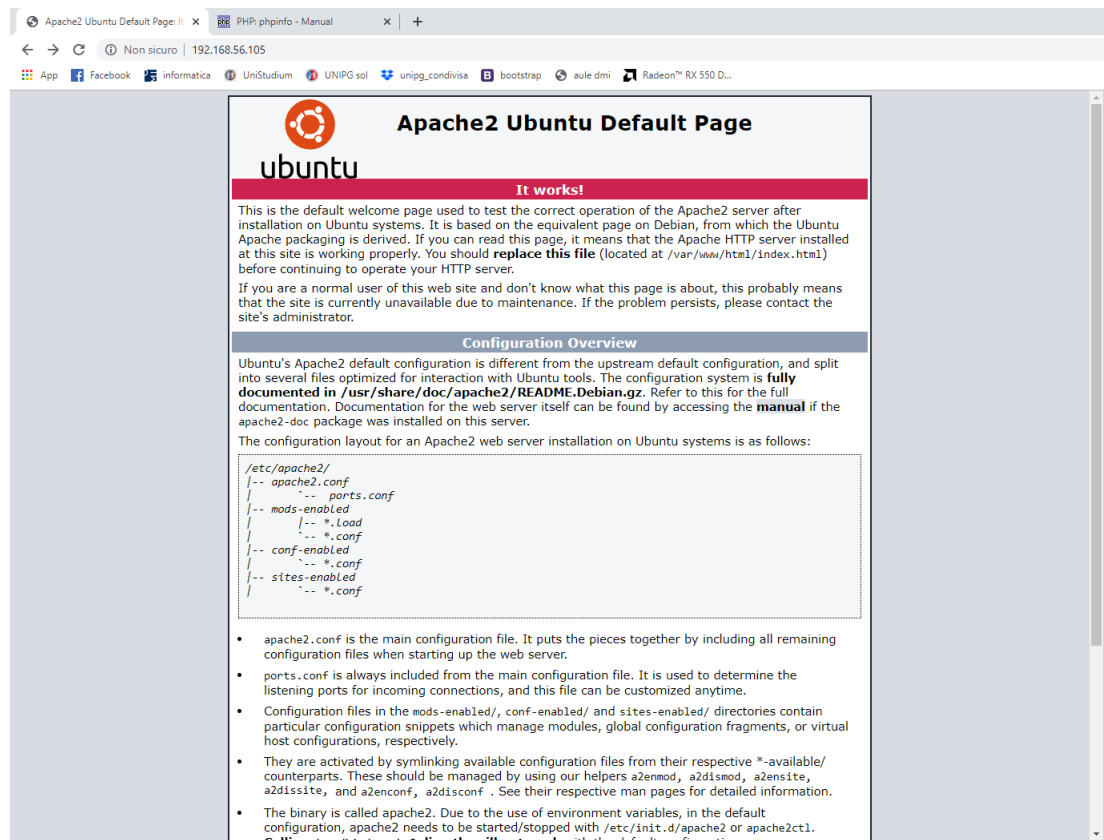
```
root@server1:/home/manuel# ping 192.168.56.106
PING 192.168.56.106 (192.168.56.106) 56(84) bytes of data.
64 bytes from 192.168.56.106: icmp_seq=1 ttl=64 time=0.461 ms
64 bytes from 192.168.56.106: icmp_seq=2 ttl=64 time=0.781 ms
64 bytes from 192.168.56.106: icmp_seq=3 ttl=64 time=0.801 ms
64 bytes from 192.168.56.106: icmp_seq=4 ttl=64 time=0.786 ms
64 bytes from 192.168.56.106: icmp_seq=5 ttl=64 time=0.798 ms
64 bytes from 192.168.56.106: icmp_seq=6 ttl=64 time=0.798 ms
^C
--- 192.168.56.106 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5010ms
rtt min/avg/max/mdev = 0.461/0.737/0.801/0.126 ms
root@server1:/home/manuel#
```

in alto la schermata del server1 in basso la schermata del server2

```
root@server2:/home/manuel# ping 192.168.56.105
PING 192.168.56.105 (192.168.56.105) 56(84) bytes of data.
64 bytes from 192.168.56.105: icmp_seq=1 ttl=64 time=0.476 ms
64 bytes from 192.168.56.105: icmp_seq=2 ttl=64 time=0.799 ms
64 bytes from 192.168.56.105: icmp_seq=3 ttl=64 time=0.761 ms
64 bytes from 192.168.56.105: icmp_seq=4 ttl=64 time=0.314 ms
64 bytes from 192.168.56.105: icmp_seq=5 ttl=64 time=0.812 ms
64 bytes from 192.168.56.105: icmp_seq=6 ttl=64 time=0.787 ms
^C
--- 192.168.56.105 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5021ms
rtt min/avg/max/mdev = 0.314/0.658/0.812/0.193 ms
root@server2:/home/manuel#
```

ora che sappiamo che le due macchine comunicano, sarebbe opportuna controllare che tutti i servizi siano in running ed enabled, come fatto in precedenza, ma ora questo passaggio lo saltiamo.

Andiamo quindi a verificare che il browser del nostro sistema operativo principale, in questo caso Windows 10 con Chrome, riesce a raggiungere l'*index.html* di *apache2* che viene automaticamente generata al momento del download del servizio e si va a posizionare su */var/www/html/index.html*, se non vi sono errori vedremo la schermata sottostante.

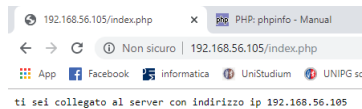


ora andiamo a controllare che sia raggiungibile anche la nostra pagina, ovvero quella in php che stampa le info della macchina. Per far ciò basta digitare sulla barra in alto del browser `192.168.56.105/index.php` così da collegarsi al server e caricare l'html di risposta che apparirà simile a questa riprodotta in basso.

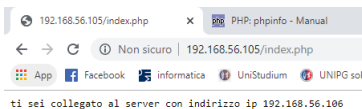
PHP Version 7.2.24-0ubuntu0.18.04.1	
System	Linux server1 4.15.0-70-generic #79-Ubuntu SMP Tue Nov 12 1
Build Date	Oct 28 2019 12:07:07
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-dom.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-imagick.ini, /etc/php/7.2/apache2/conf.d/20-ldap.ini, /etc/php/7.2/apache2/conf.d/20-mbstring.ini, /etc/php/7.2/apache2/conf.d/20-mcrypt.ini, /etc/php/7.2/apache2/conf.d/20-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/20-pdo.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-recode.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718.NTS
PHP Extension Build	API20170718.NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled

ora invece tramite il php andiamo a controllare su quale macchina ci colleghiamo e modifichiamo il file index.php e stampiamo l'ip della macchina, ci aspettiamo che se entrambi i server sono attivi il risultato sarà quello della macchina numero1 ovvero **192.168.56.105**, per fare quello descritto sfruttiamo quello scritto all'interno della variabile `$_SERVER` del php:

```
$_SERVER['SERVER_ADDR'];
```



adesso creando un errore cerchiamo di collegarci alla macchina secondaria. Per generare questa eccezione andiamo a spegnere o mettere in stamby la macchina principale e se le nostre impostazioni sono corrette dopo un attesa maggiore rispetto a prima (il tempo necessario per il client a capire che il primo server non può rispondere) otterremo invece che lo stesso ip del sito web quello della macchina secondaria che nel nostro caso è **192.168.56.106**



Il risultato ottenuto è quello sperato quindi possiamo concludere che il nostro cluster è perfettamente configurato.

## 12 Conclusioni

In conclusione posso dire che cimentarmi nella creazione di un cluster, nella configurazione delle risorse e nella gestione dei cluster mi ha appassionato. Girando per il web ho anche letto che il concetto di cluster non rientra soltanto in ambito informatico ma, al contrario, si utilizza anche in altri campi tra cui quelli medici, fisici e chimici. Infine riporto una frase tratta dal sito "WIKIPEDIA" che mi ha personalmente colpito: "Il progetto SETI@home sembrerebbe essere il più grande cluster distribuito esistente. Utilizza circa tre milioni di personal computer sparsi in tutto il mondo per analizzare i dati provenienti dal radiotelescopio di Arecibo, al fine di trovare la prova dell'esistenza di intelligenza extraterrestre."