A.D. 1308
unipg
UNIVERSITÀ DEGLI STUDI
DI PERUGIA

Relazione Laboratorio HPC

Cluster High Availability

Esercitazione n°1

Studente: Ludovico Guercio
Matricola: 340036

Dipartimento di Matematica e Informatica,
Università degli Studi di Perugia

Indice

1 Introduzione

La relazione tratta la prima esercitazione/progetto di laboratorio in cui si è stato assegnato il compito di realizzare un cluster ad alta affidabilità. Ciò è stata guidata tramite la spiegazione degli strumenti e software da utilizzare da parte del prof. Gervasi e il dott. Damiano Perri. Il mio operato, a causa dell'emergenza covid, non è stato svolto fisicamente il laboratorio.

2 Obbiettivi

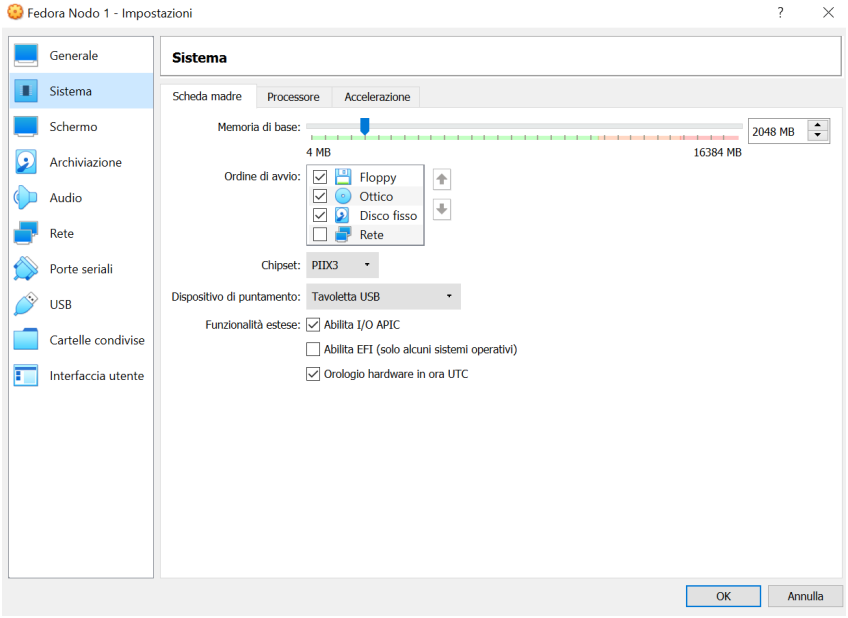
L'obbiettivo di questa esercitazione è implementare un cluster ad alta affidabilità. La realizzazione di ciò viene fatta in modo "semplificato" sia per scopo didattico, sia per le configurazioni hardware e software che si ha a disposizione. Il cluster è formato da due nodi nei quali verrano installati dei software specifici per la sua realizzazione:

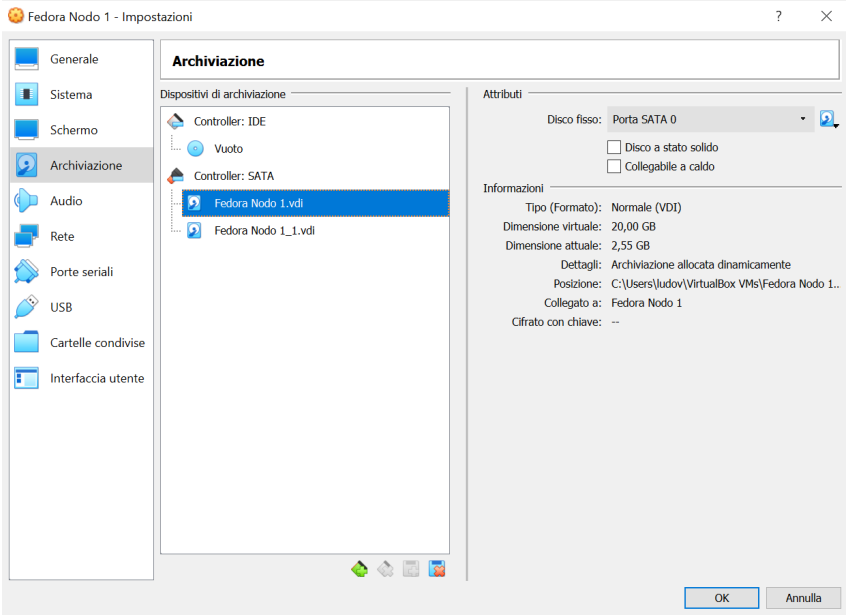
- Pacemaker come CRM(Cluster Resource Manager)
- Corosync come Cluster Engine
- Apache per la creazione e gestione di servizi lato web
- DRDB per creare una risorsa e replicarla nell'altro nodo

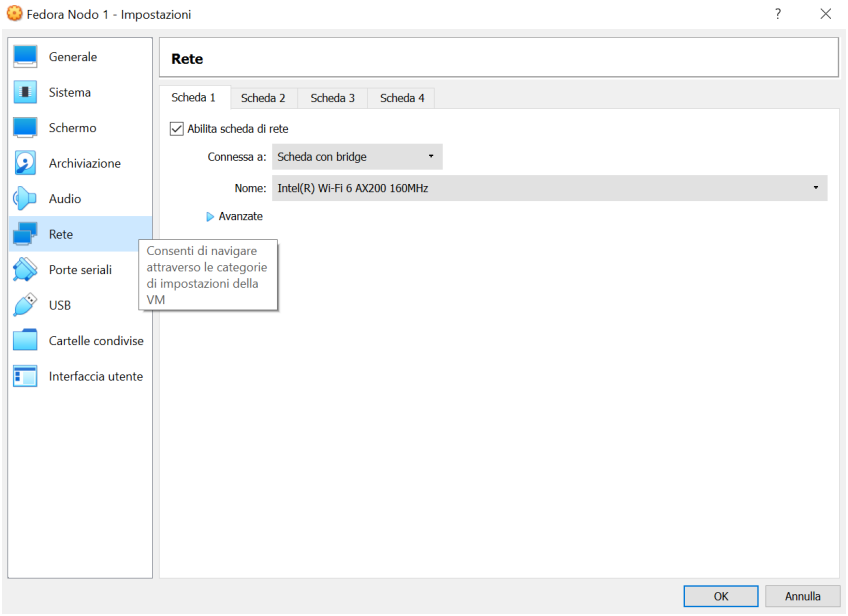
Infine, dopo aver settato correttamente tutto il cluster, bisogna verificare il suo funzionamento, ossia simulare la disconnessione dei nodi (in modo alternato) e controllare che i servizi forniti non cessano di funzionare.

3 Configurazioni Iniziali

La prova di laboratorio è stata eseguita nel mio laptop personale con sistema operativo Windows 10; come programma di virtualizzazione delle macchine è stato utilizzato VirtualBox. Il carico da supportare dell'esercitazione non è stato un problema per la mia macchina avendo a disposizione 16 GB di RAM e un processore di nuova generazione.
Il cluster poteva essere realizzato tramite 2 macchine virtuali Ubuntu Server, oppure 2 macchine Fedora Server. Per questa esercitazione ho optato per la seconda opzione, la quale era anche stata consigliata durante la lezione di laboratorio.
Dopo aver scaricato l'iso della macchina dal sito ufficiale, è stata creata una macchina con seguente la configurazione:


Fedora Nodo 1 - Impostazioni
Generale
Sistema
Schermo
Archiviazione
Audio
Rete
Porte seriali
USB
Cartelle condivise
Interfaccia utente
Sistema
Scheda madre
Processore
Accelerazione
Memoria di base:
2048 MB
4 MB
16384 MB
Ordine di avvio:
Floppy
Ottico
Disco fisso
Rete
Chipset:
PIIX3
Dispositivo di puntamento:
Tavoletta USB
Funzionalità estese:
Abilita I/O APIC
Abilita EFI (solo alcuni sistemi operativi)
Orologio hardware in ora UTC
OK
Annulla


Fedora Nodo 1 - Impostazioni
Generale
Sistema
Schermo
Archiviazione
Audio
Rete
Porte seriali
USB
Cartelle condivise
Interfaccia utente
Archiviazione
Dispositivi di archiviazione
Controller: IDE
Vuoto
Controller: SATA
Fedora Nodo 1.vdi
Fedora Nodo 1_1.vdi
Attributi
Disco fisso:
Porta SATA 0
Disco a stato solido
Collegabile a caldo
Informazioni
Tipo (Formato): Normale (VDI)
Dimensione virtuale: 20,00 GB
Dimensione attuale: 2,55 GB
Dettagli: Archiviazione allocata dinamicamente
Posizione: C:\Users\ludov\VirtualBox VMs\Fedora Nodo 1..
Collegato a: Fedora Nodo 1
Cifrato con chiave: --
OK
Annulla


Fedora Nodo 1 - Impostazioni
Generale
Sistema
Schermo
Archiviazione
Audio
Rete
Porte seriali
USB
Cartelle condivise
Interfaccia utente
Rete
Scheda 1
Scheda 2
Scheda 3
Scheda 4
Abilita scheda di rete
Connessa a:
Scheda con bridge
Nome:
Intel(R) Wi-Fi 6 AX200 160MHz
Avanzate
Consenti di navigare attraverso le categorie di impostazioni della VM
OK
Annulla

La macchina, come riporta gli screenshots, ha le seguenti caratteristiche: 2 GB di RAM, 20 GB di memoria per il disco, 1 GB di memoria del disco per la condivisione, una scheda di rete bridge e processore di 1 core (non riportato nello screenshot). Una volta finito, per semplicità e comodità, basta effettuare una "duplicazione" della macchina così avere due macchine con le stesse caratteristiche. Ad esempio, In VirtualBox per duplicare una macchina si utilizza il comando "Clona".

3.1 Settings delle Macchine

Avviando per la prima volta le macchine, si procede con l'installazione di Fedora Server. Nel menù di installazione è importante settare il disco di destinazione per il sistema operativo. Sempre nel menù di installazione, l'impostazione utenti è disabilitata: dunque si procede a configuare l'utente root. Successivamente, sempre tramite il menù, si possono creare altri user: questo passaggio è stato fatto dopo nel secondo avvio tramite linea di comando. Settato l'utente root si prosegue con l'installazione(può richiedere un pò di tempo). Alla fine del processo, Fedora è finalmente installato nelle macchine: dunque, si eseguono i reboot e si è pronti per lavorare con le due macchine.

3.2 Aggiornamento delle Macchine

Riavviate le macchina, si effettua il login tramite l'utente root. Una volta dentro sono andato subito a creare un nuovo utente in entrambe le macchine tramite i seguenti comandi:

```
useradd nodo1
passwd nodo1
usermod -aG wheel nodo1

useradd nodo2
passwd nodo2
usermod -aG wheel nodo2
```

Un aspetto importante appena si è installato una nuova macchina è quello effettuare l'aggiornamento del sistema operativo e dei pacchetti: in questo modo si scaricano gli ultimi aggiornamenti utilizzabili dalla macchina; questa pasaggio, che può sembrare banale, è operazione molto importante che anche nella realtà pratica è sempre fatta, soprattuto per scopi di sicurezza.

Eseguendo alcuni comandi ho notato l'assenza di alcuni tools importanti per poter lavorare: dunque ho installato il pacchetto relativo ai comandi di rete **net-tools**, e il pacchetto per l'editor testuale **nano** (si può installare anche l'editor **vim**, come si preferisce).

4 Realizzazione del Cluster

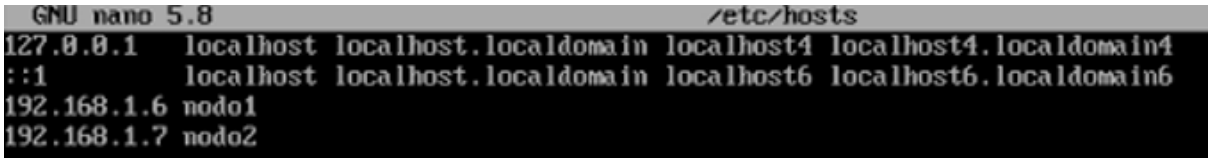
4.1 Installazione dei Tools

Una volta che le due macchine sono pronte ed aggiornate, si è proseguito con l'installazione dei software che saranno utilizzati per la realizzazione del cluster; si sono eseguiti, in entrambe le macchine, i seguenti comandi:

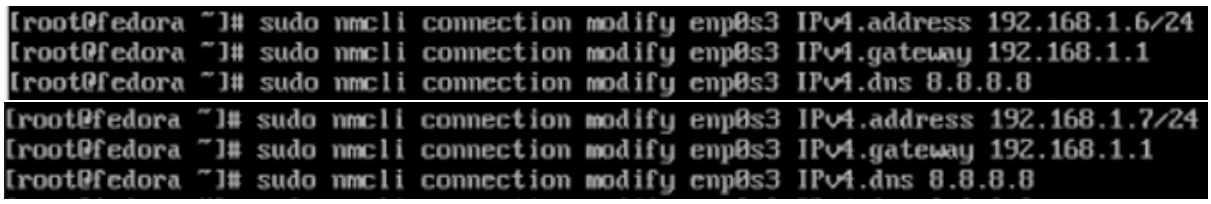
```
sudo dnf -y install pacemaker corosync pcs
sudo dnf -y install drbd-pacemaker drbd-udev
sudo dnf -y install httpd
sudo dnf -y install iptables-services
```

4.2 Ulteriori configurazione per le macchine

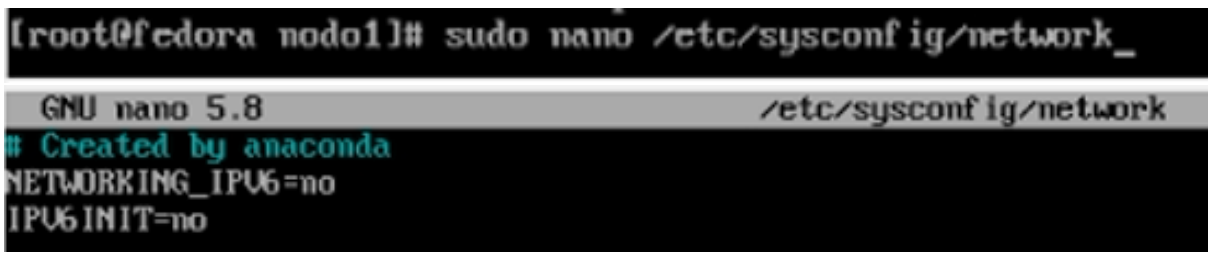
Ho assegnato un nome alle macchine nel file che si trova in */etc/hosts*: questo file è quello che viene interrogato per primo quando si effettua una richiesta **dns** per qualsiasi servizio web; con i nomi presenti nel file sarà concessa una connessione all'indirizzo IP definito.



Per mantenere sempre una comunicazione tra le macchine, è consigliato configurare gli IP statici così evitando un riassegnamento diverso degli indirizzi con il DHCP. La prima macchina avrà indirizzo **192.168.1.6**, mentra la seconda avrà **192.168.1.7**.



Una volta fatto, si effettua di nuovo un riavvio dell'interfaccia di rete per confermare le nuove modifiche degli indirizzi. Inoltre, ho anche disabilitato l'IPV6.



Infine, è consigliabile modificare la configurazione del firewall per evitare eventuali problemi o warning di comunicazione fra i nodi. In questo caso, ho optato per disabilitare direttamente il servizio:

```
sudo systemctl disable firewalld
```

4.3 Configurazione Pacemaker e Corosync

La prima cosa da fare per impostare un cluster è avviare il proprio pacemaker, ossia il gestore delle risorse; questo è fatto semplicemente con i comandi:

```
sudo systemctl enable pcsd
sudo systemctl start pcsd
```

Corosync è responsabile dello scambio dei messaggi tra i nodi interni assicurando sempre una copia delle informazioni distribuite ne cluster. Il suo funzionamento è configurato tramite un file specifico chiamato **corosync.conf**. Per compilare il file, ho preso come riferimento un esempio di file e modificato diversi parametri per l'esercitazione: Dunque, ho inserito le seguenti istruzioni per farlo lavorare.

```
totem {
    version: 2
    cluster_name: testCluster
    transport:knet
    crypto_chiper: aes256
    crypto_hash: sha256
}

nodelist {
    node {
        ring0_addr: nodo1
        name: node1
        nodeid: 1
    }

    node {
        ring0_addr: nodo2
        name: node2
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
    timestamp: on
}
```

Questi settings di Corsync e Pcsd sono state fatte, ovviamente, in entrambe le macchine. Nella directive **totem** ho lasciato invariato i paramatri di *version*, *crypto chiper*,*crypto hash* e*transport*; in cluster name va a piacere il nome da assegnare al cluster. La directive **nodelist** permette di far capire quali sono i nodi del cluster e i loro relativi indirizzi: su *ring addr* vanno inseriti gli indirizzi delle due macchine oppure, come nel mio caso, i

nomi host delle macchine. Per finire, nel file è presente una directive per **quorum** (che verrà poi disabilitata) e una directive per i **logging**.

4.4 Configurazione del Cluster

4.4.1 Autenticazione e avvio

Avendo ora configurato correttamente il Pacemaker e Corsync, possiamo avviare il cluster. Prima però, bisogna effettuare il login nei nodi tramite le rispettive macchine. L'utente, di default, è chiamato **hacluster**, la password può essere facilmente settata tramite il comando **passwd**.

```
sudo passwd hacluster
sudo pcs client local-auth -u hacluster
sudo pcs cluster auth -u hacluster
```

Una volta autenticati in entrambe le macchine, bisogna scegliere tra i due nodi quale sia quello "Master": quindi si configura l'avvio il cluster sulla macchina scelta. Nel mio caso, ho scelto la prima macchina corrispondente al **nodo1**.

```
sudo pcs cluster setup testCluster node1 node2 --force
```

```
Sending 'corosync.conf' to 'node1', 'node2'
node1: successful distribution of the file 'corosync.conf'
node2: successful distribution of the file 'corosync.conf'
Cluster has been successfully set up.
```

Subito dopo avvio il cluster(comandi *start* ed *enable*) e tramite il comando **sudo pcs status** faccio un controllo per vedere se tutto funziona correttamente.

```
[root@fedora nodo1]# sudo pcs cluster start --all
node1: Starting Cluster...
node2: Starting Cluster...
[root@fedora nodo1]# sudo pcs cluster enable --all
node1: Cluster Enabled
node2: Cluster Enabled
[root@fedora nodo1]# sudo pcs status
Cluster name: testCluster

WARNINGS:
No stonith devices and stonith-enabled is not false

Cluster Summary:
  * Stack: corosync
  * Current DC: node1 (version 2.1.1-9.fc35-77db578727) - partition with quorum
  * Last updated: Fri Nov 12 10:59:34 2021
  * Last change:  Fri Nov 12 10:59:26 2021 by hacluster via crmd on node1
  * 2 nodes configured
  * 0 resource instances configured

Node List:
  * Online: [ node1 node2 ]

Full List of Resources:
  * No resources

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
```

In questa esercitazione, non avendo un cluster contente molti nodi si è quindi disabilitato Quorum:

```
sudo pcs property set no-quorum-policy=ignore
```

Inoltre, è necessario disabilitare stonith perchè non si avrà un inconsistenza tra i dati, e nodi non sono effettivamente fisici (quindi con un alimentazione) ma virtualizzati.

```
sudo pcs property set stonith-enabled=flase
```

4.4.2 Web Service

Per far si che la nostra risorsa web sia raggiungibile dal cluster, bisogna configurare in un indirizzo di floating a quest'ultimo. Ovviamente l'indirizzo IP del cluster deve essere valido per la nostra rete. Nel comando bisogna specificare il file di configurazione Corosync, un intervallo di monitoring e un nome al gruppo dei nodi. Poi si aggiunge la risorsa httpd.conf al cluster

```
[root@fedora nodo1]# sudo pcs resource create floating_ip ocf:heartbeat:IPaddr2 ip=192.168.1.9 cidr_
netmask=24 op monitor interval=30s --group gruppoNodi
[root@fedora nodo1]# sudo pcs resource create webServer ocf:heartbeat:apache configfile="/etc/httpd/
conf/httpd.conf" op monitor timeout="15sec" interval="30s" --group gruppoNodi
[root@fedora nodo1]#
```

In questo modo avremo un server web sempre attivo anche con la disattivazione del nodo Master. Il secondo nodo facendo parte del gruppo dei nodi, può raggiungere la risorsa web.

4.5 Configurazione e avvio DBRB

Nella sezione 3 avevo riportato uno screenshot delle configurazioni delle macchine virtuali: lì è presente anche quello dei settaggi dell'archiviazione del disco. Difatti, per entrambe le macchine, si è creata una nuova partizione da 1 GB dedicata al DBRB. Grazie al DBRB si può replicare, ad esempio come nel nostro caso, contenuto Web verso gli altri nodi di un cluster con continuità e consistenza dei dati. In caso di falla, un secondo nodo prende il ruolo di "Master", e con il DBRB legge i contenuti e continua a prestare i servizi del nodo disconnesso. Dunque, dopo aver aggiunto un nuovo disco dalle impostazioni delle macchine di VirtualBox, si può procede alla partizione (per entrambe le macchine) tramite il comando:

```
\sudo fdisk /dev/sdb
```

Questo comando riporta all'esecuzione di una sequenza di sottocomandi: qui è possibile eseguire molteplici funzioni di partizione; in questo caso, ci serve una sola nuova partizione.

```
Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x18f185a7.

Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097151, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097151, default 2097151):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[nodo2@fedora ~]$
```

Ora finita la partizione dei dischi, si procede con la configurazione del DBRB: nella directory DBRB bisogna creare un file di risorse e inserendo una semplice configurazione.

```
sudo nano /etc/drbd.d/<nomefile>.res
```

```
 GNU nano 5.8                    /etc/drbd.d/wwwdata.res                    Modified
resource wwwdata {
  protocol C;

  on nodo1 {
    device /dev/drbd0;
    disk /dev/sdb1;
    address 192.168.1.6:5555;
    meta-disk internal;
  }
  on nodo2 {
    device /dev/drbd0;
    disk /dev/sdb1;
    address 192.168.1.7:5555;
    meta-disk internal;
  }
}
```

- **device /dev/drbd0** indica il nome drbd comune per i due nodi
- **disk /dev/sdb1** specifica la partizione locale usata per la replicazione
- **address:port** è per gli indirizzi IP delle macchine e porte di comunicazione comune
- **meta-disk internal** è definito per usare meta dati interni

Lo stesso file deve essere creato e popolato nella seconda macchina: questo passaggio può essere fatto manualmente, oppure effettuando una copia diretta tramite il comando:

```
scp /etc/drbd.d/<nomefile>.res node2:/etc/drbd.d/
```

Si verifica poi che il file sia incluso dentro */etc/drbd.conf*, altrimenti bisogna inserire al suo interno:

```
include "drbd.d/*.res
```

Successivamente, sempre su entrambe le macchine, si inizializza la risorsa per i nodi con i comandi:

```
drbdadm create-md wwwdata
sudo modprobe drbd
sudo drbdadm up wwwdata
sudo drbdadm -- --overwrite-data-of-peer primary all
sudo drbdadm primary --force wwwdata
```

Infine si può finalmente attivare e avviare il DRBD su i due nodi

```
[root@nodo1 nodo1]# sudo systemctl start drbd
[root@nodo1 nodo1]# sudo systemctl enable drbd
Created symlink /etc/systemd/system/multi-user.target.wants/drbd.service → /usr/lib/systemd/system/
rbd.service.
```

4.5.1 Configurazione risorse DRBD

A questo punto, il cluster può già funzionare. Per completare, configuriamo la risorsa DRBD: formatto inizialmente la risorsa con il comando **mkfs.xfs**, poi si esegue il mount del volume e si scrive aggiunge la risorsa (nel nostro caso una pagina web con semplice testo).

```
mkfs.xfs /dev/drbd0
sudo mount /dev/drbd0 /mnt
echo "contenuto html" | sudo tee /mnt/index.html
sudo umount /dev/drbd0
```

Una volta fatto, basta aggiungere la risorsa creata all'interno del cluster:

```
sudo pcs cluster cib drbd_cfg
sudo pcs -f drbd_cfg resource create WebData ocf:linbit:drbd
drbd_resource=wwwdata op monitor interval=30s

sudo pcs -f drbd_cfg resource promotable WebData promoted-max=1
promoted-node-max=1 clone-max=2 clone-node-max=1 notify=true

sudo pcs cluster cib-push drbd_cfg --config
```

Infine per avere la replicazione negli altri nodi (nel nostro caso solo nel nodo2), si aggiunge una risorsa WebFS:

```
sudo pcs cluster cib fs_cfg
sudo pcs -f fs_cfg resource create WebFS Filesystem device=
"/dev/drbd0" directory="/var/www/html" fstype="xfs"
```

```
sudo pcs -f fs_cfg constraint colocation add WebFS with
WebData-clone INFINITY with-rsc-role=Master

sudo pcs -f fs_cfg constraint order promote WebData-clone
then start WebFS

sudo pcs -f fs_cfg constraint colocation add WebServer
with WebFS INFINITY

sudo pcs -f fs_cfg constraint order WebFS then WebServer
sudo pcs cluster cib-push fs_cfg --config
```

5 Test di Funzionamento

Con il cluster completo e funzionante, si prova quindi a disconnettere il nodo Master (nodo1) per verificare che il funzionamento venga continuato tramite il nodo Slave (nodo2). Sempre tramite il comando **sudo pcs status** si può controllare lo stato dei servizi:

```
[nodo1@nodo1 ~]$ sudo pcs status
[sudo] password for nodo1:
Cluster name: testCluster
Cluster Summary:
  * Stack: corosync
  * Current DC: node2 (version 2.1.1-9.fc35-77db578727) - partition with quorum
  * Last updated: Mon Dec 27 12:06:27 2021
  * Last change:  Mon Dec 27 11:23:29 2021 by root via cibadmin on node1
  * 2 nodes configured
  * 5 resource instances configured

Node List:
  * Online: [ node1 node2 ]

Full List of Resources:
  * Resource Group: gruppoNodi:
    * floating_ip       (ocf::heartbeat:IPaddr2):       Started node1
    * webServer (ocf::heartbeat:apache):        Started node1
  * Clone Set: webData-clone [webData] (promotable):
    * Masters: [ node1 ]
    * Slaves: [ node2 ]
  * webFS       (ocf::heartbeat:Filesystem):     Started node1

Failed Resource Actions:
  * webFS_monitor_20000 on node1 'not running' (7): call=27, status='complete', exitreason='', last-
rc-change='2021-12-27 11:44:36 -05:00', queued=0ms, exec=0ms

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[nodo1@nodo1 ~]$
```

Con:

```
sudo pcs node standby <nomenodo>
sudo pcs node unstandby <nomenodo>
```

Si può stoppare e riavviare un nodo del cluster. Con lo stop del nodo1 , il nodo2 diventa il nuovo Master: