



UNIVERSITÀ DI PERUGIA  
Dipartimento di Matematica e Informatica



LAUREA MAGISTRALE IN INFORMATICA  
CORSO DI HIGH PERFORMANCE COMPUTING

# Esercitazione n.1

## HIGH AVAILABILITY CLUSTER

*Professore*

**Prof. Ovaldo Gervasi**  
**Dott. Damiano Perri**

*Studente*

**Cristian Cosci**  
**(Matricola: 350863)**

---

Anno Accademico 2021-2022

# Indice

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduzione al problema</b>                       | <b>3</b>  |
| <b>2</b> | <b>Descrizione specifiche Hardware e Software</b>     | <b>5</b>  |
| 2.1      | Specifiche Software . . . . .                         | 5         |
| 2.2      | Specifiche Hardware . . . . .                         | 8         |
| <b>3</b> | <b>Preparazione macchine virtuali</b>                 | <b>9</b>  |
| 3.1      | Installazione Sistema Operativo . . . . .             | 9         |
| 3.2      | Installazione Software Necessari . . . . .            | 11        |
| 3.3      | Clonazione Macchina Virtuale . . . . .                | 12        |
| 3.4      | Configurazione IP . . . . .                           | 13        |
| <b>4</b> | <b>Configurazione Sistema</b>                         | <b>18</b> |
| 4.1      | PCSD . . . . .  | 18        |
| 4.2      | Configurazione Corosync . . . . .                     | 18        |
| 4.3      | Autenticazione nodi . . . . .                         | 21        |
| 4.4      | Partizionamento disco per risorsa condivisa . . . . . | 22        |
| 4.5      | Configurazione e avvio del Cluster . . . . .          | 26        |
| 4.6      | Stonith e Quorum . . . . .                            | 26        |
| 4.7      | Configurazione risorsa HTTP e DRBD . . . . .          | 27        |
| 4.8      | Risorsa DRBD . . . . .                                | 30        |
| <b>5</b> | <b>Conclusioni</b>                                    | <b>33</b> |
| 5.1      | Test funzionamento del cluster . . . . .              | 33        |

# Capitolo 1

## Introduzione al problema

Il lavoro svolto ha l'obiettivo di realizzare un **cluster ad alta affidabilità**.

Un **cluster** di computer è costituito da un insieme di computer (nodi) interconnessi tra loro, capaci di lavorare in modo da essere visti come un unico sistema. Il carico di lavoro è quindi distribuito in modo da garantire la fornitura dei servizi forniti anche nel caso uno o più nodi si disconnettano.

Un cluster è composto da:

- Nodi;
- Rete;
- Os;
- Cluster Middleware.

I cluster vengono implementati per migliorare la velocità e l'affidabilità rispetto a quella fornita da un singolo nodo. I nodi all'interno di un cluster sono sistemi indipendenti. I cluster sono in grado di supportare applicazioni che vanno dall'e-commerce a importanti database. Il cluster computing può anche essere utilizzato per dividere il carico di lavoro dei servizi tra i nodi, consentendo di gestire il bilanciamento del carico di lavoro e la tolleranza ai guasti.

In particolare i **cluster ad alta affidabilità** sono progettati per fornire una disponibilità ininterrotta di dati o servizi. In caso di guasto di un nodo, il servizio può essere ripristinato senza compromettere la disponibilità dei servizi forniti dal cluster. Lo

scopo di questi cluster è garantire che una singola istanza di un'applicazione sia sempre in esecuzione su un membro del cluster alla volta, ma se e quando quel membro del cluster non è più disponibile, l'applicazione eseguirà il failover su un altro membro del cluster. La resistenza ai guasti deve essere garantita dalla presenza di diversi nodi con lo stesso contenuto informativo, in grado di subentrare nell'erogazione dei contenuti non appena un nodo abbia problematiche di connessione.

Nelle sezioni successive dell'elaborato verranno presentati e spiegati i vari passaggi, le specifiche hardware e software utilizzate e le configurazioni eseguite.

## Capitolo 2

# Descrizione specifiche Hardware e Software

Il cluster realizzato è composto da 2 nodi in cui si hanno:

- **Pacemaker** come CRM (Cluster Resource Manager);
- **Corosync** come Cluster Engine;
- **Apache** per la gestione dei servizi web;
- **DRBD** per creare una risorsa drbd.

Il cluster verrà realizzato tramite un ambiente virtualizzato (in questo caso Virtualbox è stato scelto come software di virtualizzazione). Il sistema operativo consigliato è Fedora Server (versione 34).

### 2.1 Specifiche Software

#### Cluster Resource Manager

**Pacemaker** è un gestore di risorse ad alta disponibilità open source per cluster di piccole e grandi dimensioni. In caso di errore, i gestori delle risorse come Pacemaker avviano automaticamente il ripristino e si assicurano che l'applicazione sia disponibile da una delle macchine rimanenti nel cluster. Pacemaker ottiene la massima

disponibilità per i servizi cluster rilevando e ripristinando gli errori del nodo e del livello di servizio. Ciò viene raggiunto utilizzando le capacità di messaggistica e di appartenenza fornite dall'infrastruttura cluster preferita. Pacemaker è responsabile del funzionamento delle risorse, permette di controllarne il loro stato, di avviarle o di stopparle, e gestisce il comportamento che queste devono avere nel caso si verifichino malfunzionamenti.

## Cluster Engine

**Corosync** Cluster Engine è un demone che gestisce lo scambio di messaggi e l'appartenenza dei nodi all'interno dei gruppi. È stato implementato come evoluzione di OpenAIS, per risolvere i problemi osservati lavorando con OpenAIS, PeaceMaker e Apache Qpid. Corosync si avvicina all'alta disponibilità garantendo che ogni server ridondante nel sistema mantenga una copia ridondante delle informazioni utilizzate per prendere decisioni per l'applicazione. Questo approccio è chiamato "distributed state machine". In una tipica macchina a stati, i progettisti di software chiamano funzioni che modificano lo stato dell'applicazione. Utilizzando Corosync, i progettisti di software inviano messaggi invece di chiamare funzioni. Quando questi messaggi vengono consegnati, la macchina a stati su tutti i nodi cambia il suo stato in modo ordinato e coerente. Corosync è altamente sintonizzato e progettato per le prestazioni. È stata presa una considerazione speciale per ridurre al minimo il cambio di contesto della fine della memoria.

## DRBD

Il **DRBD** (Distributed Replicated Block Device), viene utilizzato per replicare il contenuto informativo all'interno di un nodo master, sui vari nodi slave. Il DRBD esegue la copia live di un disco tra il server attivo e quello di backup, al fine di garantire la correttezza delle informazioni memorizzate su disco. In combinazione verrà utilizzato **Heartbeat**, con lo scopo di verificare il corretto funzionamento dei server associati a un determinato servizio. In caso di failover il software esegue lo scambio tra un server di backup e il server primario. Il DRBD è un elemento fondamentale per formare un cluster ad alta disponibilità (HA). Questo viene fatto eseguendo il mirroring di un intero dispositivo a tramite una rete assegnata. La funzionalità software di repli-

cazione dei blocchi dell'hard disk del nodo master è completamente trasparente alle applicazioni che ne devono leggere il contenuto. Se ad esempio inseriamo le pagine di un contenuto Web all'interno dello spazio governato dal DRBD, queste verranno replicate su tutti i nodi del cluster garantendo continuità di servizio e consistenza dei dati. Un applicativo software di questo tipo viene installato quando la soluzione che si sta realizzando è fortemente indirizzata all'affidabilità dei dati. Se il nodo primario dovesse incorrere in problematiche di rete, heartbeat, avvierà la procedura che porta uno dei nodi secondari a prendere il posto del primario, garantendo la continuità del servizio offerto. Dopo un'interruzione di un nodo, il DRBD:

- risincronizza automaticamente il nodo temporaneamente non disponibile all'ultima versione dei dati, in background, senza interferire con il servizio in esecuzione. Ovviamente questo funziona anche se il ruolo del nodo sopravvissuto è stato modificato mentre il peer era inattivo.
- Nel caso in cui un'interruzione completa dell'alimentazione disattivi entrambi i nodi, DRBD rileverà quale dei nodi è rimasto inattivo più a lungo ed eseguirà la risincronizzazione nella giusta direzione.
- Dopo un'interruzione della rete di replica, DRBD ristabilirà la connessione ed eseguirà automaticamente la risincronizzazione necessaria.

## Heartbeat

**Heartbeat** è un demone che consente la realizzazione logica del cluster. Il programma Heartbeat è uno dei componenti principali del progetto LinuxHA (High-Availability Linux). Heartbeat è altamente portatile ed è il primo software scritto per il progetto Linux-HA. Heartbeat invia costantemente messaggi ai nodi chiedendo se questi sono ancora a disposizione, nel caso in cui il nodo primario non risponda, uno dei nodi passivi prende in carico l'erogazione dei servizi forniti dal nodo primario gestendo l'indirizzo IP virtuale a cui il cluster fa riferimento.

## 2.2 Specifiche Hardware

Sono state utilizzate due macchine virtuali con il sistema operativo Fedora con la seguente configurazione (la stessa per entrambe le macchine):

- **CORE ASSEGNATI:** 2;
- **RAM:** 2 GB;
- **DISCO 1:** 20 GB;
- **DISCO 2:** (spiegazione utilizzo vedi Sezione 4.4): 2 GB;
- **SCHEDA DI RETE:** impostata in modalità Bridge.

Per comodità è stata prima configurata una singola macchina virtuale e successivamente è stata clonata. È ovviamente necessario clonare la macchina virtuale scegliendo di creare nuovi indirizzi MAC per tutte le schede di rete. I processi e le fasi di installazione sono spiegate nel Capitolo 3.



# Capitolo 3

## Preparazione macchine virtuali

### 3.1 Installazione Sistema Operativo

Per prima cosa è stato necessario installare il sistema operativo Fedora su una macchina virtuale (vedi Figura 3.1). L'installazione è stata eseguita lasciando tutto di default e creando un solo utente "**cristian**".

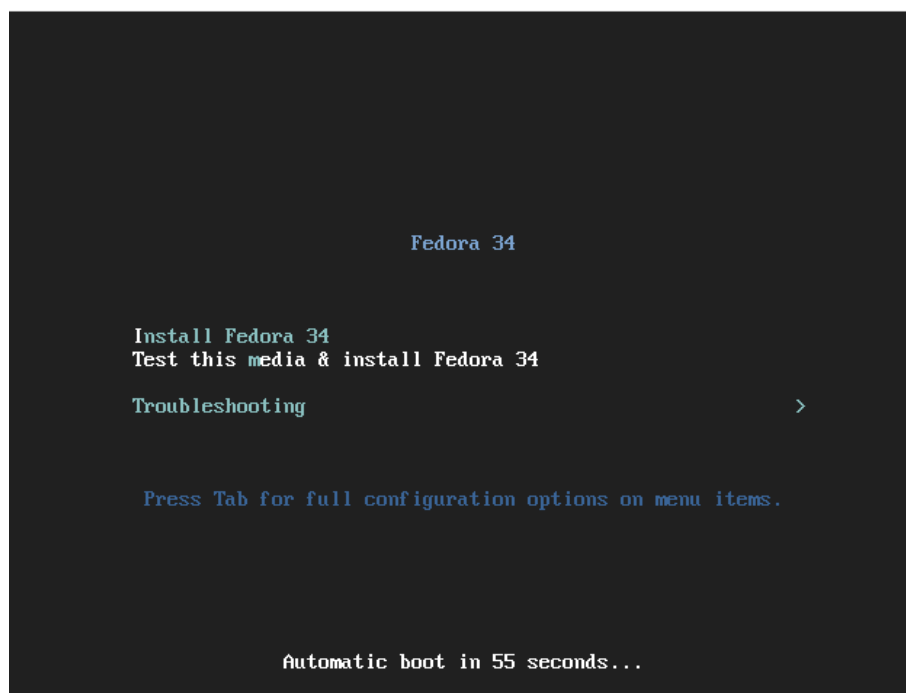


Figura 3.1: Installazione sistema operativo 1.

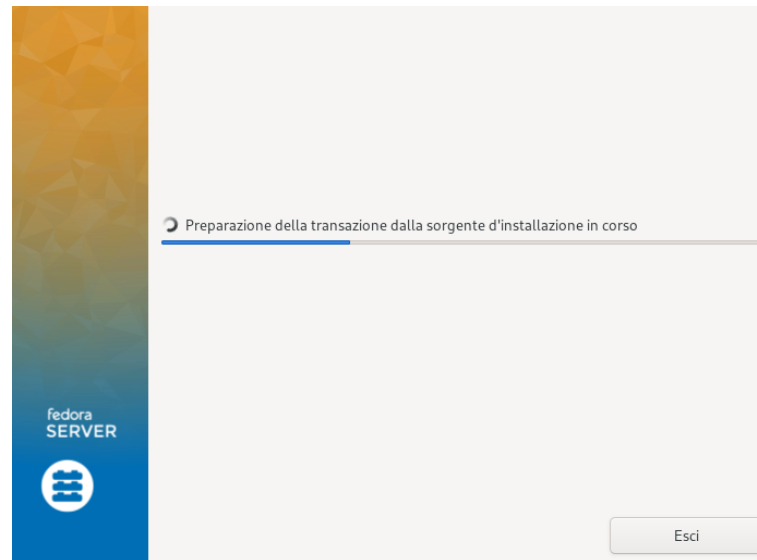


Figura 3.2: Installazione sistema operativo 2.

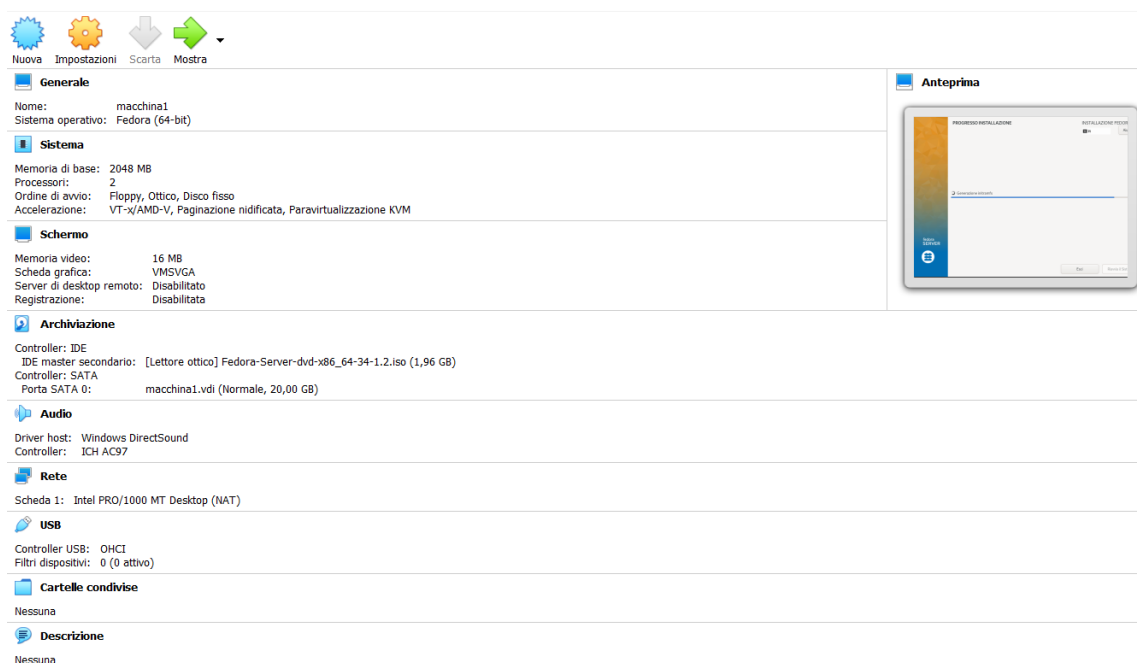
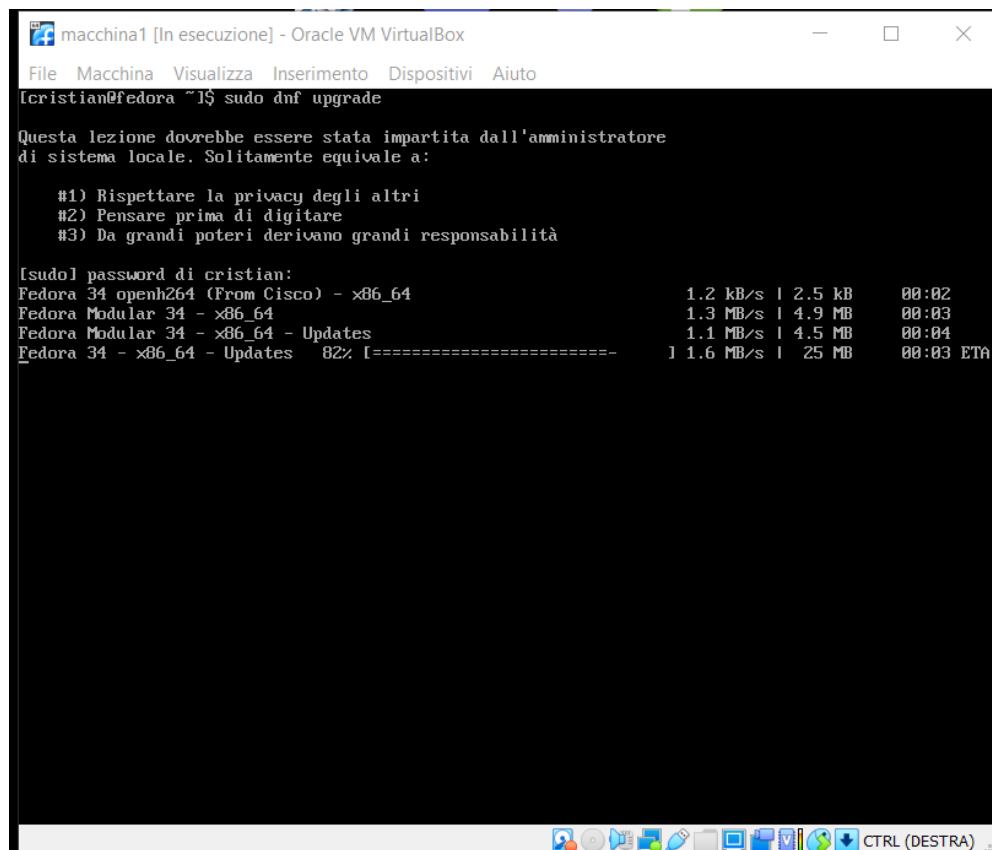


Figura 3.3: Configurazione macchina virtuale con Fedora.

In Figura 3.3 è possibile vedere la configurazione della *Macchina1* con Fedora.

## 3.2 Installazione Software Necessari

Una volta installato il sistema operativo è stato necessario effettuare un aggiornamento del sistema operativo mediante il comando "upgrade".



```
macchina1 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
lcristian@fedora ~]$ sudo dnf upgrade

Questa lezione dovrebbe essere stata impartita dall'amministratore
di sistema locale. Solitamente equivale a:

#1) Rispettare la privacy degli altri
#2) Pensare prima di digitare
#3) Da grandi poteri derivano grandi responsabilità

[sudo] password di cristian:
Fedora 34 openh264 (From Cisco) - x86_64          1.2 kB/s | 2.5 kB    00:02
Fedora Modular 34 - x86_64                      1.3 MB/s | 4.9 MB    00:03
Fedora Modular 34 - x86_64 - Updates             1.1 MB/s | 4.5 MB    00:04
Fedora 34 - x86_64 - Updates  82% [===== 1 1.6 MB/s | 25 MB    00:03 ETA
```

Figura 3.4: Upgrade sistema operativo.

Successivamente sono stati installati tutti i vari software e pacchetti necessari alla creazione e test di un cluster ad alta affidabilità.

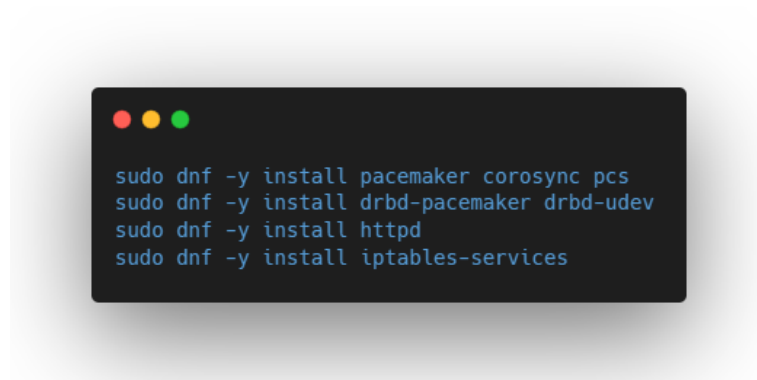


Figura 3.5: Installazione pacchetti fondamentali.

### 3.3 Clonazione Macchina Virtuale

Successivamente, avendo una macchina con tutti i software necessari, ho proceduto alla clonazione di quest'ultima mediante la procedura fornita direttamente da VirtualBox. Una particolare accortezza sta nello scegliere l'impostazione per generare nuovi indirizzi MAC per tutte le schede di rete.

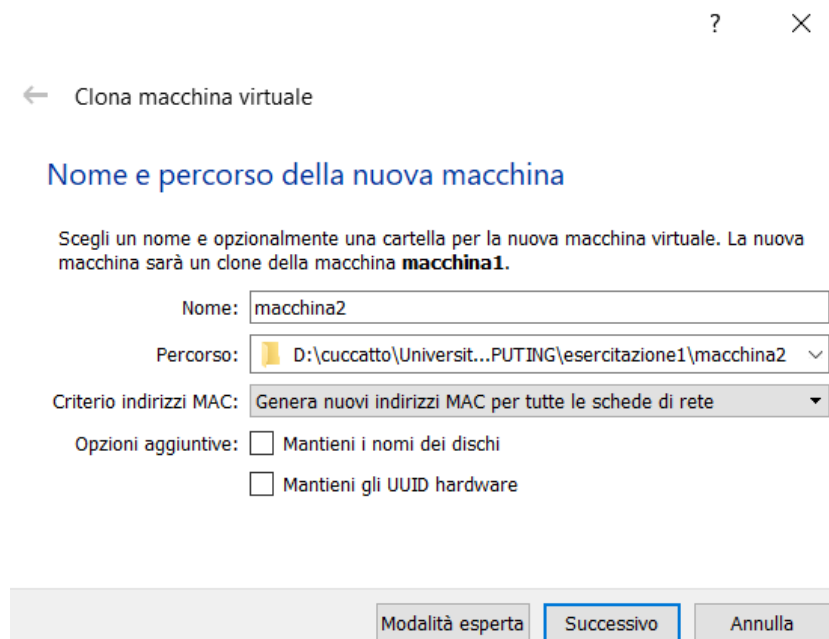


Figura 3.6: Clonazione macchina virtuale.

A questo punto ho impostato le schede di rete delle due macchine (*macchina1* e *macchina2*) in modalità Bridge.

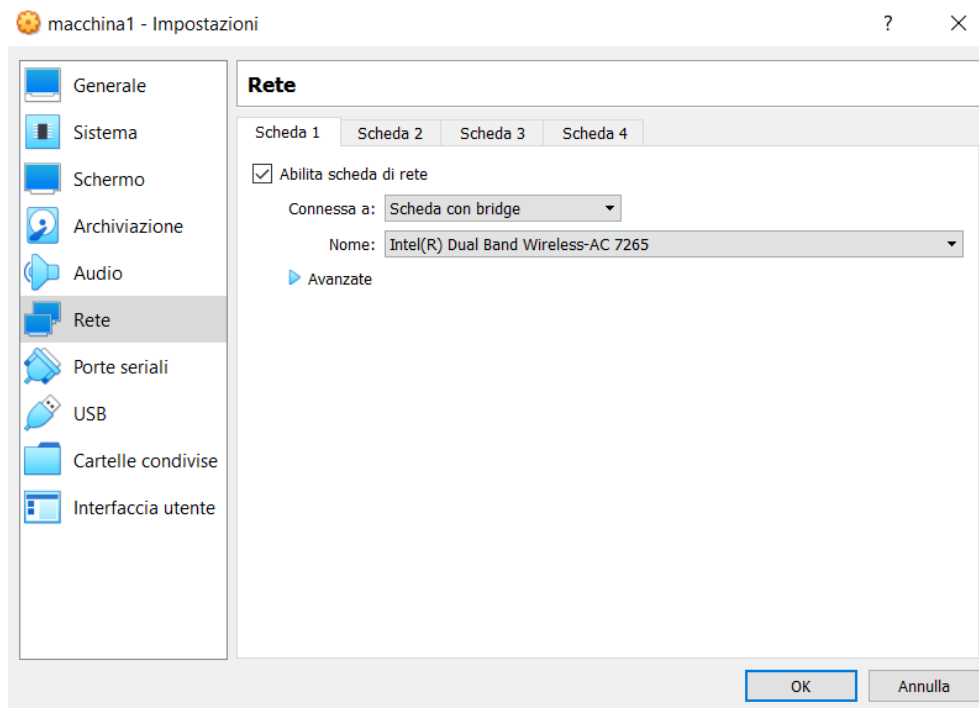
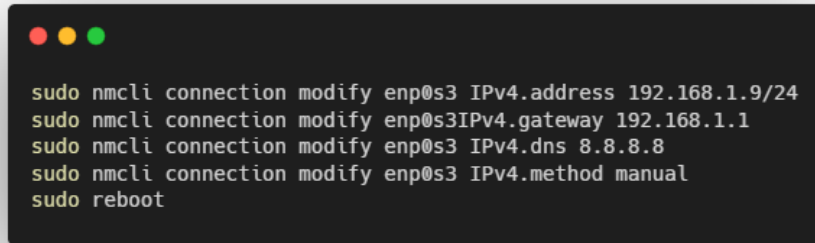


Figura 3.7: Scheda di rete in modalità bridge.

## 3.4 Configurazione IP

Per evitare problemi legati al DHCP del router di casa (riassegnamento di indirizzi diversi alle macchine virtuali nelle successive accensioni) ho deciso di settare degli indirizzi IP statici alle macchine.

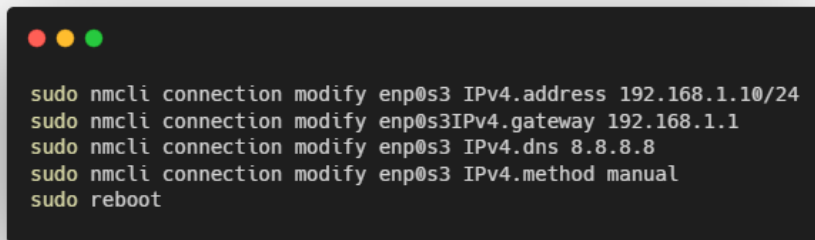
macchina1:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains five lines of terminal commands:

```
sudo nmcli connection modify enp0s3 IPv4.address 192.168.1.9/24
sudo nmcli connection modify enp0s3IPv4.gateway 192.168.1.1
sudo nmcli connection modify enp0s3 IPv4.dns 8.8.8.8
sudo nmcli connection modify enp0s3 IPv4.method manual
sudo reboot
```

Figura 3.8: Configurazione IP macchina1.

macchina2:

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains five lines of terminal commands:

```
sudo nmcli connection modify enp0s3 IPv4.address 192.168.1.10/24
sudo nmcli connection modify enp0s3IPv4.gateway 192.168.1.1
sudo nmcli connection modify enp0s3 IPv4.dns 8.8.8.8
sudo nmcli connection modify enp0s3 IPv4.method manual
sudo reboot
```

Figura 3.9: Configurazione IP macchina1.

In questo caso la scheda di rete da configurare è **enp0s3**. Le istruzioni in successione servono ad effettuare i seguenti passaggi:

- assegno indirizzo IP (con subnet mask) alla macchina;
- assegno il gateway alla macchina;
- assegno DNS;

- l'ultima istruzione permette di evitare incongruenze con i due indirizzi (uno statico e uno assegnato dal DHCP). Con questa istruzione non si ha più l'indirizzo assegnato dal DHCP.

È possibile vedere (dopo il riavvio del sistema) il corretto funzionamento delle modifiche, leggendo la routing table (vedi Figura 3.10).

```

macchina1 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
Icristian@fedora ~]$ route -n
Kernel IP routing table
Destination      Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0          192.168.1.1    0.0.0.0         UG    100    0      0 enp0s3
192.168.1.0      0.0.0.0        255.255.255.0   U     100    0      0 enp0s3
Icristian@fedora ~]$

```

Figura 3.10: Routing table.

Ho successivamente assegnato un nome ad ogni macchina nel relativo file `/etc/hosts`. Per fare ciò ho semplicemente aggiunto le seguenti righe ai rispettivi file delle due macchine.

**macchina1:**



Figura 3.11: Configurazione file hosts macchina1.

**macchina2:**



Figura 3.12: Configurazione file hosts macchina2.

A questo punto il file `/etc/hosts` compare nel seguente modo.



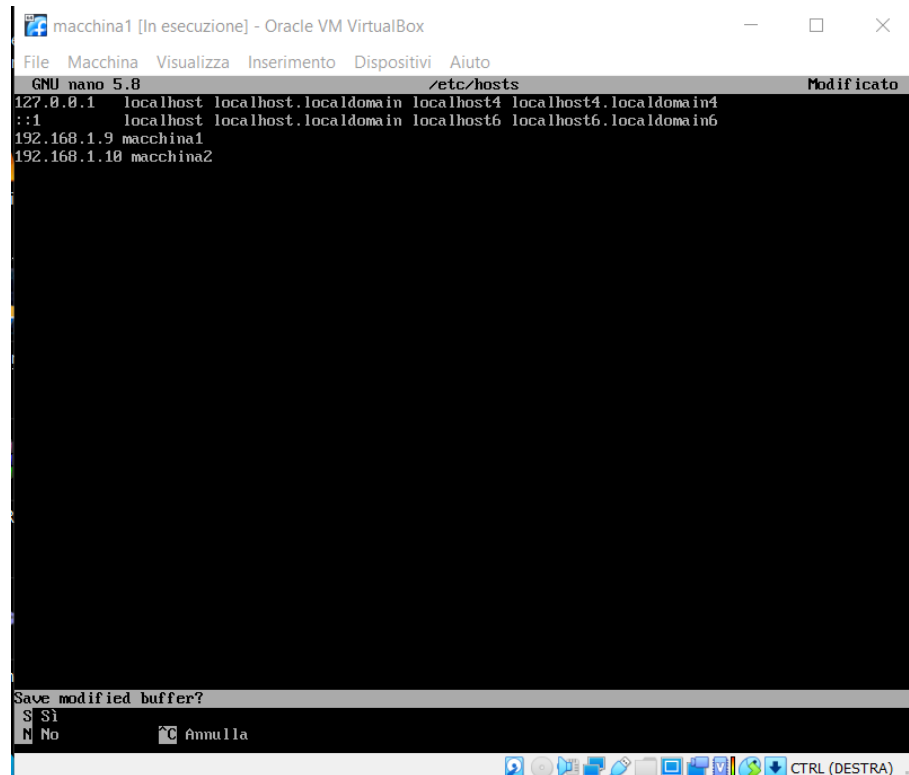


Figura 3.13: File `/etc/hosts` macchina1.

È consigliabile anche disabilitare il firewall, in quanto potrebbe limitare delle comunicazioni e causare errori nella realizzazione del cluster se non appositamente configurato. In questo caso si tratta solo di una configurazione di test per verificare l'effettivo funzionamento del cluster, quindi si può procedere semplicemente disabilitandolo con i seguenti comandi (in entrambe le macchine ovviamente):

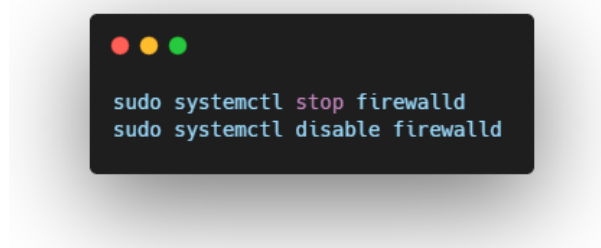


Figura 3.14: Disabilito firewall.

# Capitolo 4

## Configurazione Sistema

### 4.1 PCSD

Per procedere alla creazione di un cluster è necessario avviare Pacemaker su ogni nodo. Per fare ciò ho prima configurato l'utente **hacluster** impostando la relativa password. In questo modo sarà possibile autenticarsi nelle successive fasi di configurazione del cluster. In seguito ho abilitato e attivato il software su ogni nodo.

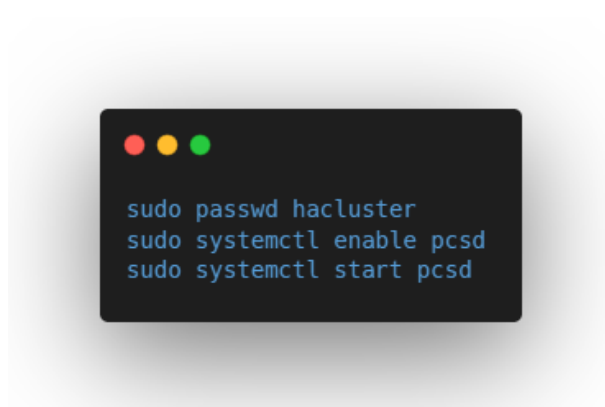
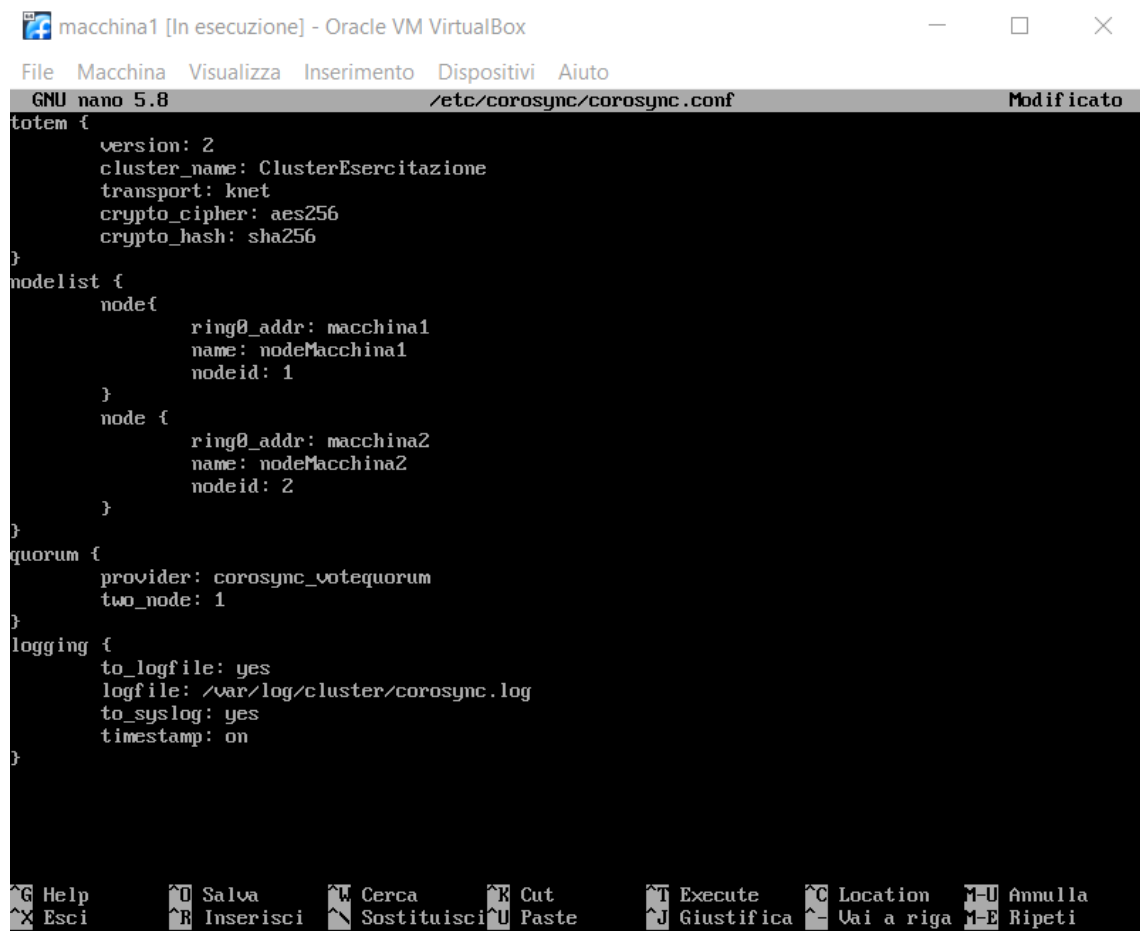


Figura 4.1: Disabilitato firewall.

### 4.2 Configurazione Corosync

Successivamente è necessario popolare il file `/etc/corosync/corosync.conf`. Il file **corosync.conf** fornisce i parametri del cluster utilizzati da corosync, il gestore clu-

ster su cui si basa Pacemaker. In generale, non si dovrebbe modificare direttamente il file `corosync.conf` ma, invece, utilizzare l'interfaccia `pcs` o `pcsd`. Tuttavia, potrebbe verificarsi una situazione in cui è necessario modificare direttamente questo file (una guida alla configurazione <https://www.systutorials.com/docs/linux/man/5-corosync.conf/>). Nel mio caso il file è stato così configurato (in entrambe le macchine):



```
macchina1 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
GNU nano 5.8 /etc/corosync/corosync.conf Modificato
totem {
    version: 2
    cluster_name: ClusterEsercitazione
    transport: knet
    crypto_cipher: aes256
    crypto_hash: sha256
}
nodelist {
    node {
        ring0_addr: macchina1
        name: nodeMacchina1
        nodeid: 1
    }
    node {
        ring0_addr: macchina2
        name: nodeMacchina2
        nodeid: 2
    }
}
quorum {
    provider: corosync_votequorum
    two_node: 1
}
logging {
    to_logfile: yes
    logfile: /var/log/cluster/corosync.log
    to_syslog: yes
    timestamp: on
}
^G Help      ^O Salva     ^W Cerca    ^K Cut       ^T Execute  ^C Location  ^-U Annulla
^X Esci      ^R Inserisci ^_ Sostituisci ^U Paste     ^J Giustifica ^_ Vai a riga ^-E Ripeti
```

Figura 4.2: Configurazione file `/etc/corosync/corosync.conf`.

Qui di seguito ho riportato la descrizione e la spiegazione dei parametri utilizzati e scelti:

- **totem { }**: Questa direttiva di primo livello contiene opzioni di configurazione per il protocollo totem.

- **logging { }**: Questa direttiva di primo livello contiene opzioni di configurazione per la registrazione.
- **quorum { }**: Questa direttiva di primo livello contiene le opzioni di configurazione per il quorum.
- **nodelist { }**: Questa direttiva di primo livello contiene opzioni di configurazione per i nodi nel cluster.

All'interno della direttiva **totem**, ci sono sette opzioni di configurazione di cui una è richiesta, cinque sono facoltative e una è richiesta quando IPV6 è configurato nella sottodirettiva dell'interfaccia. La direttiva richiesta controlla la versione della configurazione del totem. L'opzione facoltativa, a meno che non si utilizzi la direttiva IPV6, controlla l'identificazione del processore. Le altre, opzionali, controllano la segretezza e l'autenticazione, la modalità di funzionamento ad anello ridondante e il campo MTU massimo di rete.

- **version**: Specifica la versione del file di configurazione. Attualmente l'unica versione valida per questa direttiva è la 2.
- **crypto\_hash**: Specifica quale autenticazione HMAC deve essere utilizzata per autenticare tutti i messaggi. I valori validi sono none (nessuna autenticazione), md5, sha1, sha256, sha384 e sha512. L'impostazione predefinita è sha1.
- **crypto\_cipher**: Questo specifica quale cifratura dovrebbe essere usata per cifrare tutti i messaggi. I valori validi sono nessuno (nessuna crittografia), aes256, aes192, aes128 e 3des. L'abilitazione di crypto\_cipher richiede anche l'abilitazione di crypto\_hash. L'impostazione predefinita è aes256.
- **transport**: Questa direttiva controlla il meccanismo di trasporto utilizzato. Se l'interfaccia a cui si lega corosync è un'interfaccia RDMA come RoCEE o Infiniband, è possibile specificare il parametro "iba". Per evitare completamente l'uso del multicast, è possibile specificare un parametro di trasporto unicast "udpu". Ciò richiede di specificare l'elenco dei membri nella direttiva nodelist, che potrebbe potenzialmente costituire l'appartenenza prima della distribuzione. L'impostazione predefinita è udp. Il tipo di trasporto può anche essere impostato su udpu o iba.

- **cluster\_name**: Specifica il nome del cluster ed è utilizzato per la generazione automatica dell'indirizzo multicast.

All'interno della direttiva **logging**, ci sono diverse opzioni di configurazione che sono tutte facoltative.

- **timestamp**: Questo specifica che un timestamp viene inserito su tutti i messaggi di registro. L'impostazione predefinita è disattivata.
- **logfile**: Se la direttiva `to_logfile` è impostata su `yes`, questa opzione specifica il percorso del file di registro. Nessuna impostazione predefinita.

All'interno della direttiva **quorum** è possibile specificare l'algoritmo del quorum da utilizzare:

- **provider**: Al momento della scrittura è supportato solo `corosync_votequorum`.

All'interno della direttiva `nodelist` è possibile specificare informazioni specifiche sui nodi nel cluster. La direttiva può contenere solo la sottodirettiva del nodo, che specifica ogni nodo che dovrebbe essere un membro dell'appartenenza e dove sono necessarie opzioni non predefinite. Ogni nodo deve avere almeno il campo `ring0_addr` riempito. Le opzioni possibili sono:

- **ringX\_addr**: Questo specifica l'indirizzo IP di uno dei nodi. X è il numero del ring. Nel mio caso è `macchina1/macchina2` (vedi configurazione Sezione 3.4).
- **nodeid**: Questa opzione di configurazione è facoltativa quando si utilizza IPv4 e richiesta quando si utilizza IPv6. Questo è un valore a 32 bit che specifica l'identificatore del nodo consegnato al servizio di appartenenza al cluster. Se questo non è specificato con IPv4, l'ID del nodo sarà determinato dall'indirizzo IP a 32 bit del sistema a cui il sistema è associato con l'identificatore dell'anello pari a 0. Il valore dell'identificatore del nodo pari a zero è riservato e non deve essere utilizzato.

## 4.3 Autenticazione nodi

A questo punto effettuo l'autenticazione per ogni nodo al cluster.

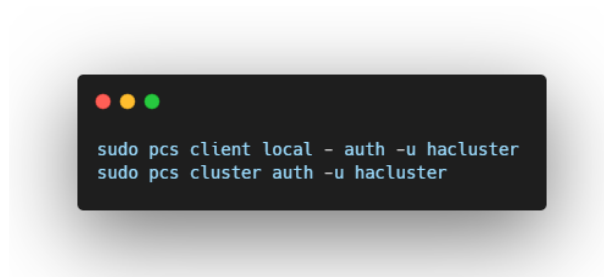


Figura 4.3: Autenticazione nodi.

## 4.4 Partizionamento disco per risorsa condivisa

Successivamente ho deciso di procedere al partizionamento del secondo disco su ciascuna macchina. In particolare ho aggiunto un nuovo disco da 2GB ad ogni macchina e l'ho successivamente partizionato. È stato necessario avviare una nuova partizione primaria, comprendendo l'intero disco, ed ho successivamente riavviato le due macchine virtuali. In Figura 4.4, Figura 4.4, Figura 4.4 e Figura 4.4 è possibile vedere il procedimento da eseguire su VirtualBox per aggiungere un nuovo disco ad una macchina virtuale. Successivamente è necessario il partizionamento, con i comandi in Figura 4.8. Infine in Figura 4.9 si può vedere come il disco aggiuntivo sia correttamente utilizzabile dal sistema operativo. Tutti i seguenti passaggi sono stati eseguiti in entrambe le macchine virtuali.

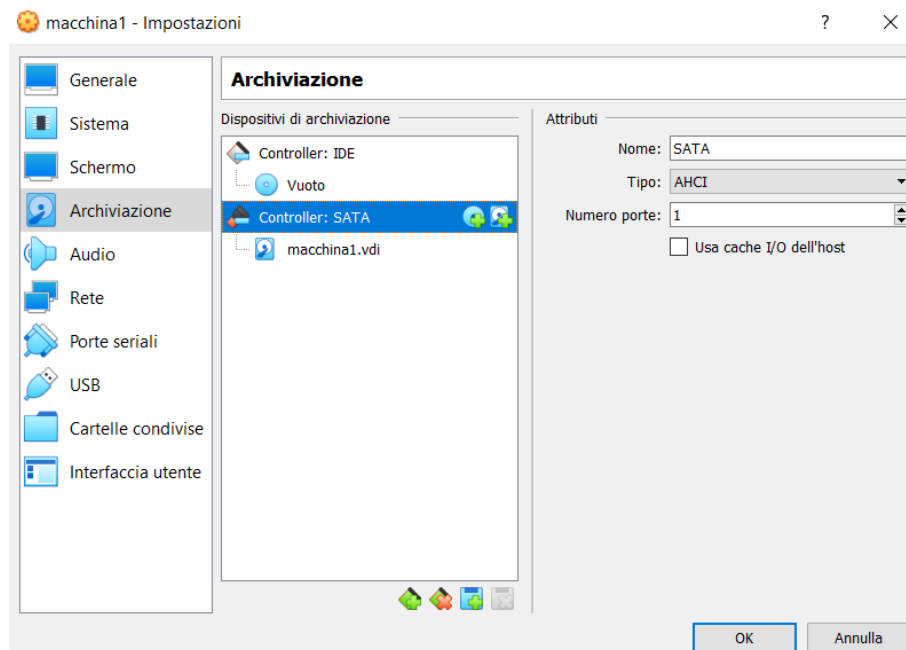


Figura 4.4: Aggiunta nuovo disco su macchina virtuale.

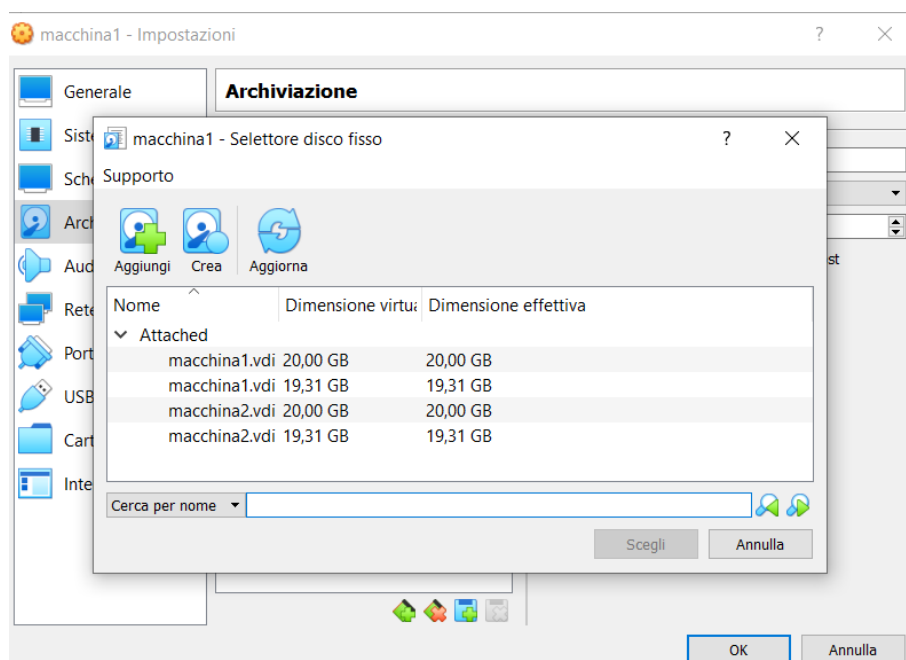


Figura 4.5: Aggiunta nuovo disco su macchina virtuale.

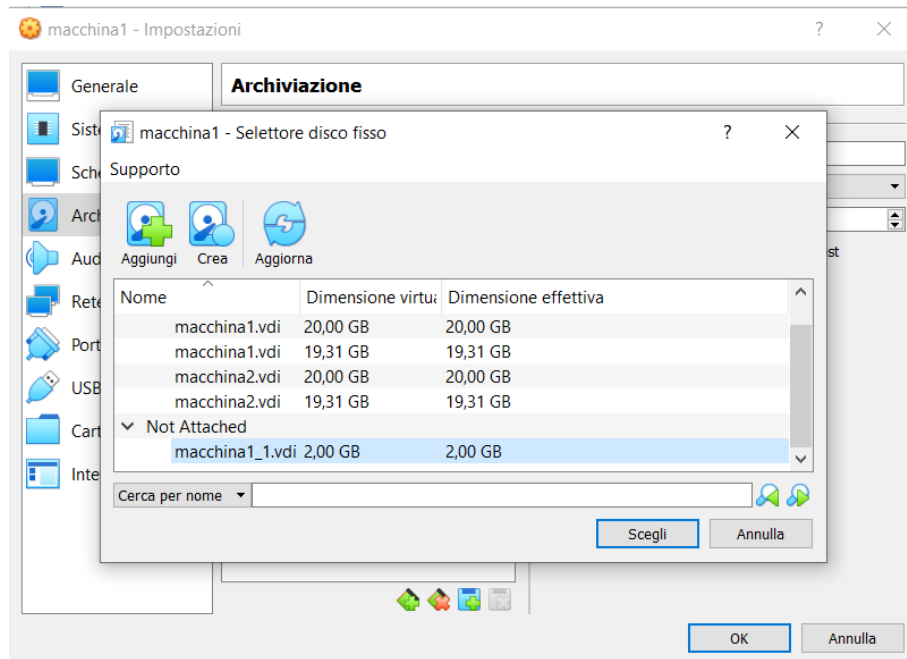


Figura 4.6: Aggiunta nuovo disco su macchina virtuale.

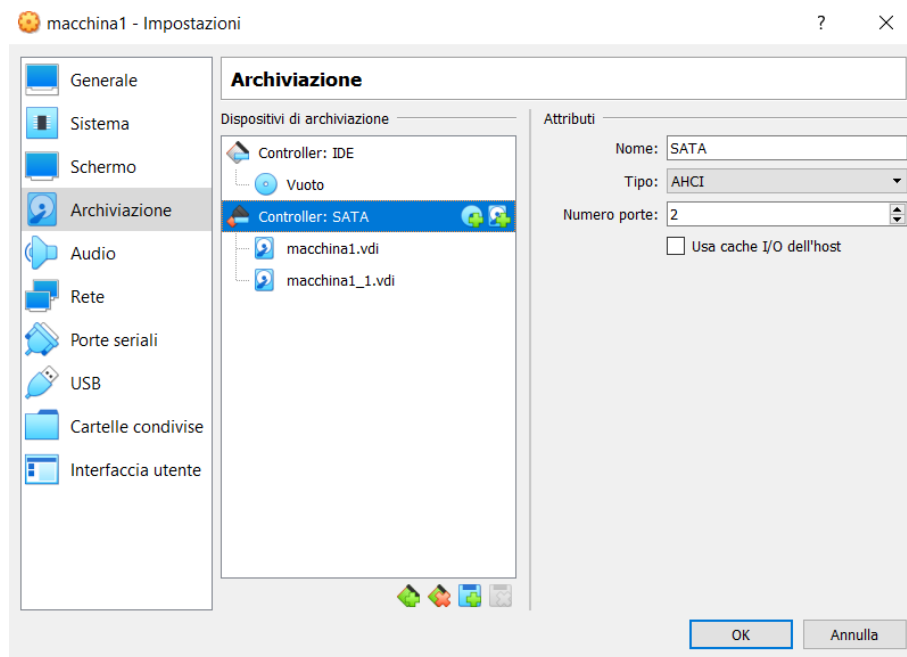


Figura 4.7: Aggiunta nuovo disco su macchina virtuale.



```
sudo fdisk -l      # serve per vedere i dischi collegati, generalmente il nuovo disco sara' /dev/sdb
sudo fdisk /dev/sdb # formattiamo il disco: n, p, 1, enter, enter, w
sudo fdisk -l      # controllo la presenza della nuova partizione /dev/sdb1
```

Figura 4.8: Partizionamento nuovo disco.

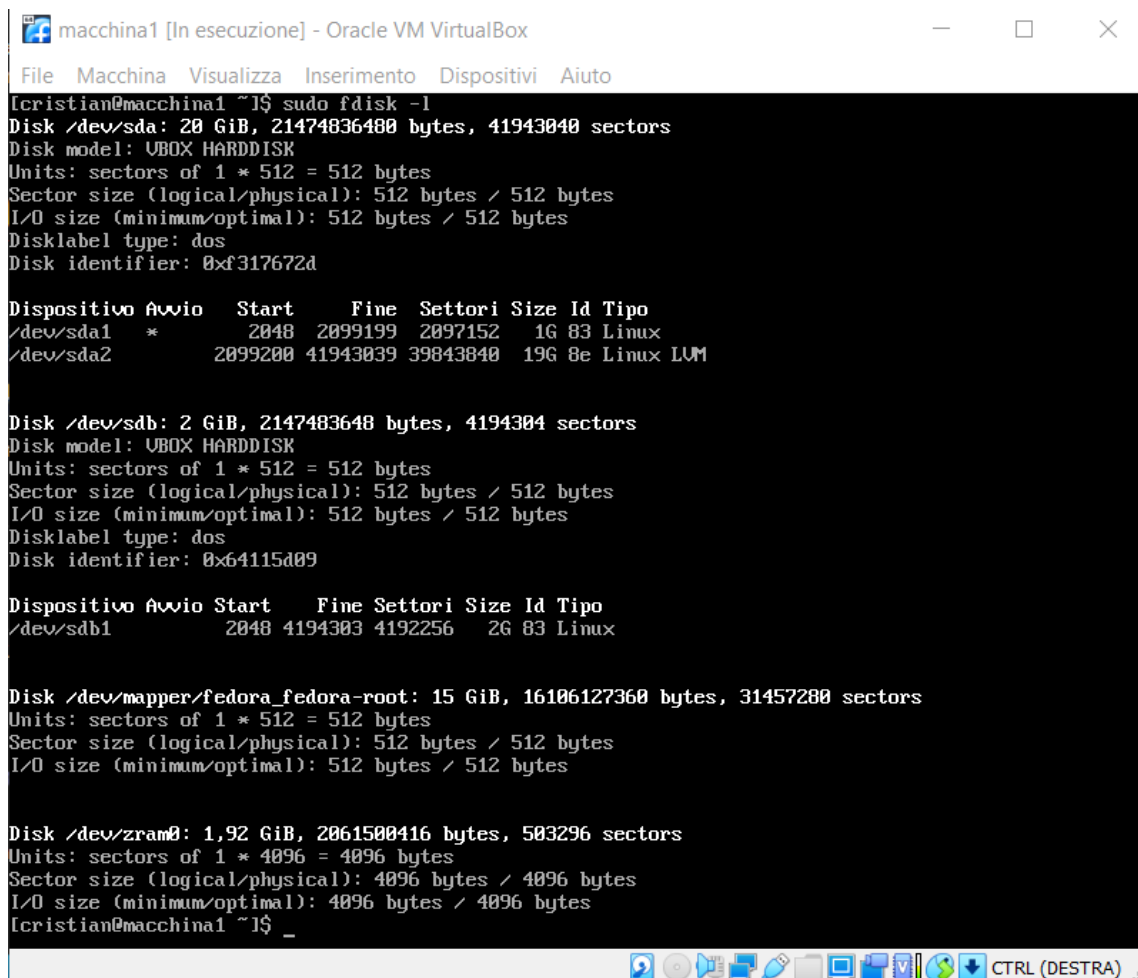
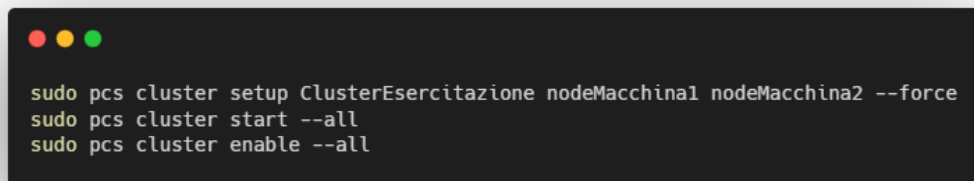


Figura 4.9: Visualizzazione nuovo disco su macchina virtuale.

## 4.5 Configurazione e avvio del Cluster

È ora possibile configurare l'avvio del cluster. Quest'ultimo passaggio deve essere eseguito su una sola macchina (che inizialmente verrà anche scelta come nodo master).

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It displays three lines of terminal commands in a light green monospace font.

```
sudo pcs cluster setup ClusterEsercitazione nodeMacchina1 nodeMacchina2 --force
sudo pcs cluster start --all
sudo pcs cluster enable --all
```

Figura 4.10: Configurazione avvio automatico del cluster e relativo avvio.

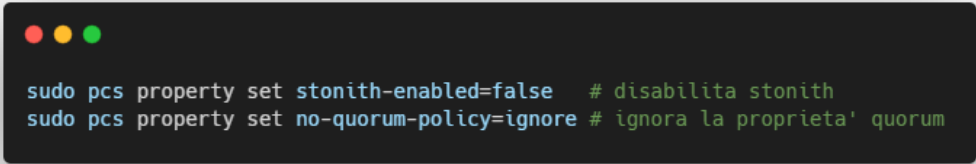
È ora possibile verificare il corretto funzionamento del cluster controllando mediante il comando **pc status**.

## 4.6 Stonith e Quorum

In una situazione reale potrebbe essere possibile che un nodo slave si renda conto che l'attuale master non stia funzionando correttamente, ad esempio per un problema di rete. Se in questa situazione il nodo slave diventasse master, potrebbe creare inconsistenza nei dati in quando potrebbe essere possibile che l'anomalia che era stata analizzata fosse relativa alla rete e non al nodo stesso. Grazie a **Stonith** si renderebbe non operativo il master, evitando di creare inconsistenza nei dati. Ancora meglio sarebbe delegare questa funzionalità ad un componente hardware in grado di spegnere il nodo che presenta problemi di questo tipo. In questo progetto Stonith non verrà configurato in quanto non possono esserci situazioni di inconsistenza dei dati e i nodi non sono alimentati effettivamente da una presa di corrente in quanto virtualizzati all'interno della macchina Host.

**Quorum** permette di disabilitare il cluster nel caso in cui meno della metà dei nodi sono inaccessibili e quindi disconnessi dalla rete. Quorum dovrà essere disattivato

perché con un esempio utilizzando solo 2 nodi rischieremmo che il cluster si disattivi nel caso in cui un solo nodo vada offline.

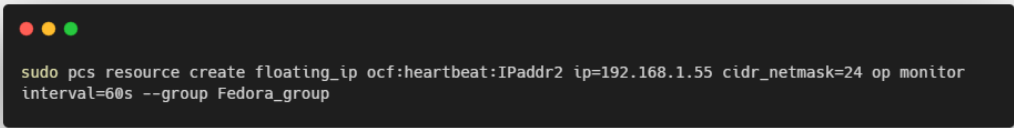


```
sudo pcs property set stonith-enabled=false # disabilita stonith
sudo pcs property set no-quorum-policy=ignore # ignora la proprieta' quorum
```

Figura 4.11: Disabilito Stonith e Quorum

## 4.7 Configurazione risorsa HTTP e DRBD

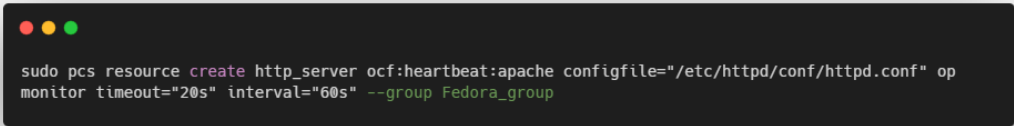
Per prima cosa è necessario assegnare un ip virtuale al cluster, in modo da poter successivamente raggiungere il server web che vi verrà configurato.



```
sudo pcs resource create floating_ip ocf:heartbeat:IPaddr2 ip=192.168.1.55 cidr_netmask=24 op monitor interval=60s --group Fedora_group
```

Figura 4.12: Assegnamento indirizzo IP al cluster.

Aggiungo la risorsa HTTPD. In questo modo si ha un server web sempre attivo, anche nel caso in cui il nodo principale vada offline.



```
sudo pcs resource create http_server ocf:heartbeat:apache configfile="/etc/httpd/conf/httpd.conf" op monitor timeout="20s" interval="60s" --group Fedora_group
```

Figura 4.13: Aggiunta risorsa httpd.

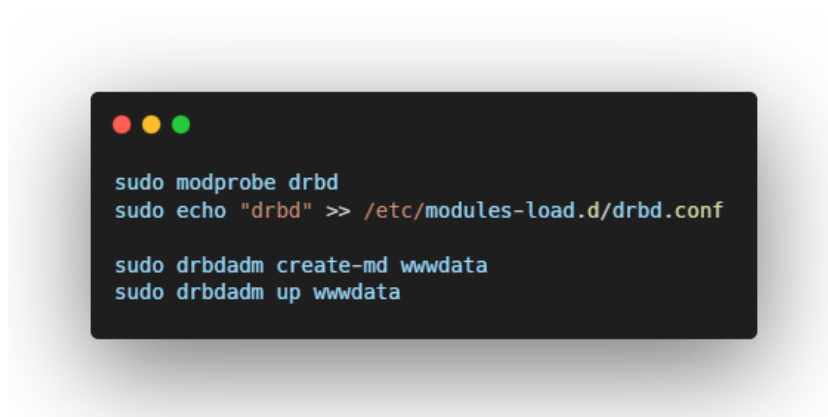
È ora necessario procedere alla configurazione del file `/etc/drbd.d/wwwdata.res` (vedi esempio [https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.1/html/Clusters\\_from\\_Scratch/\\_configure\\_drbd.html](https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/_configure_drbd.html)). Come si può vedere in Figura 4.14 il disco da utilizzare è quello aggiunto in Sezione 4.4. Per variare dalla configurazione del tutorial ho scelto di utilizzare un algoritmo di crittografia basato su chiave segreta condivisa. Questa configurazione deve essere fatta in entrambe le macchine, invertendo opportunamente l'ordine delle due macchine nel file (in `macchina1` apparirà prima la configurazione di `macchina1` e viceversa in `macchina2`). Una volta fatto questo ho ritenuto necessario disabilitare una policy di sicurezza che non mi ha permesso inizialmente di procedere nella configurazione, in quanto i sistemi con SELinux possono bloccare il server del database, impedendone l'avvio o impedendo al nodo di stabilire connessioni con altri nodi del cluster. Per evitare ciò, è necessario configurare le politiche SELinux per consentire al nodo di funzionare (<https://galeracluster.com/library/documentation/selinux.html>).



```
resource wwwdata {  
    protocol C;  
    device /dev/drbd0;  
  
    syncer {  
        verify-alg sha1;  
    }  
  
    net {  
        cram-hmac-alg sha1;  
        shared-secret "prova";  
    }  
  
    on macchina1 {  
        disk /dev/sdb1;  
        address 192.168.1.9:7676;  
        meta-disk internal;  
    }  
    on macchina2 {  
        disk /dev/sdb1;  
        address 192.168.1.10:7676;  
        meta-disk internal;  
    }  
}
```

Figura 4.14: Configurazione file `/wwwdata.res`.

Ho successivamente creato la risorsa drbd e abilitato il modulo kernel per rendere sempre utilizzabile la risorsa nei successivi riavvii del sistema (in entrambe le macchine).

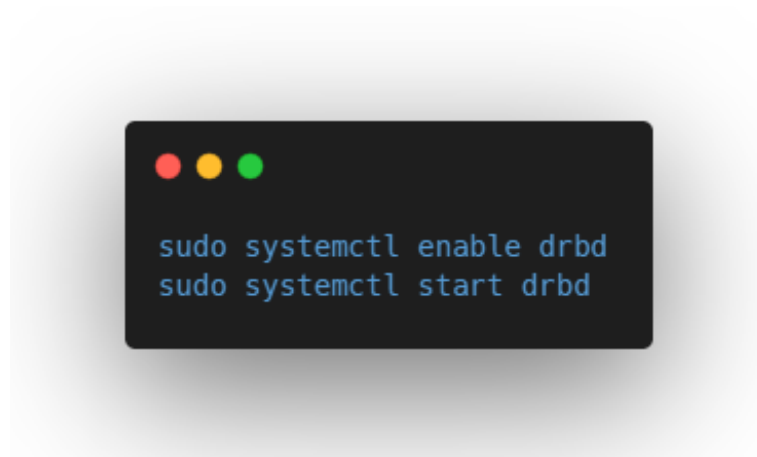
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains the following commands:

```
sudo modprobe drbd
sudo echo "drbd" >> /etc/modules-load.d/drbd.conf

sudo drbdadm create-md wwwdata
sudo drbdadm up wwwdata
```

Figura 4.15: Creazione risorsa drbd e abilito modulo kernel.

È ora possibile abilitare e avviare il drbd.

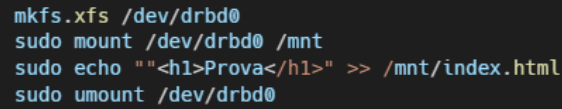
A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains the following commands:

```
sudo systemctl enable drbd
sudo systemctl start drbd
```

Figura 4.16: Avvio drbd.

## 4.8 Risorsa DRBD

Come richiesto ho formattato e aggiunto alla risorsa creata in precedenza una pagina web per controllare il corretto funzionamento.



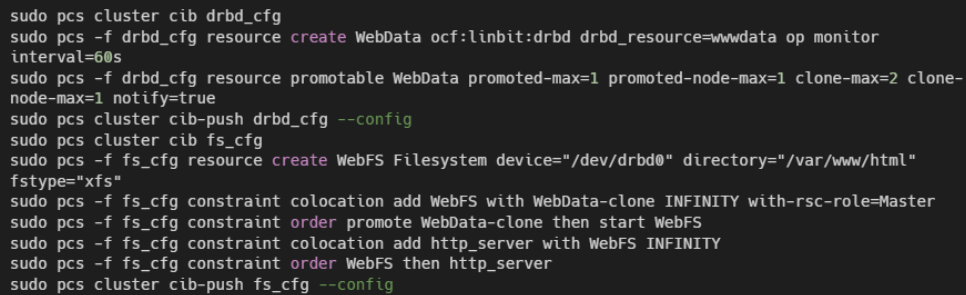
```

mkfs.xfs /dev/drbd0
sudo mount /dev/drbd0 /mnt
sudo echo "<h1>Prova</h1>" >> /mnt/index.html
sudo umount /dev/drbd0

```

Figura 4.17: Formattazione risorsa.

Successivamente l'ho aggiunta direttamente al cluster. Per far sì che i dati del Web Server vengano replicati in tutti i nodi del cluster, garantendone la disponibilità in caso di caduta di uno di essi, è necessario aggiungere al cluster la risorsa WebFS (con la certezza che quest'ultima venga avviata prima del server web stesso).



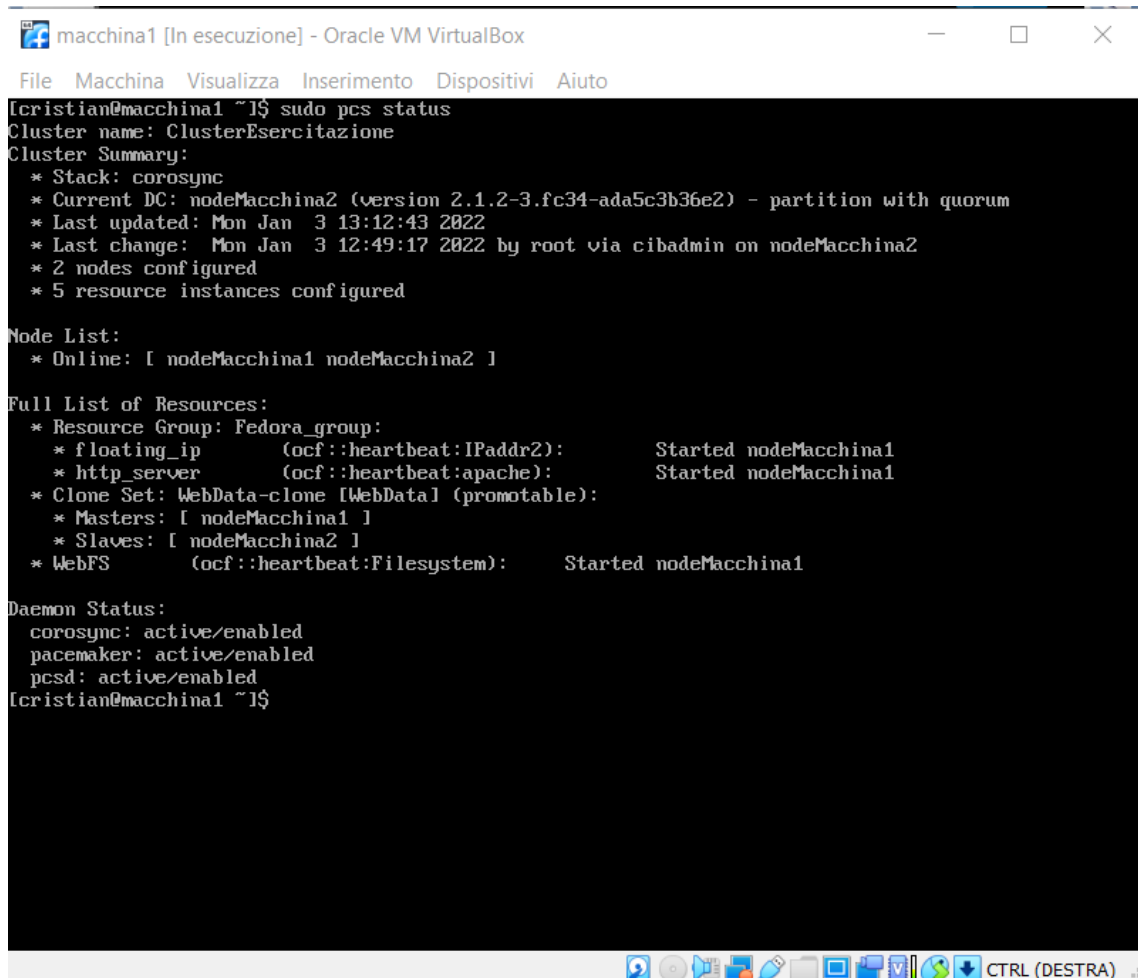
```

sudo pcs cluster cib drbd_cfg
sudo pcs -f drbd_cfg resource create WebData ocf:linbit:drbd drbd_resource=wwwwdata op monitor
interval=60s
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
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 http_server with WebFS INFINITY
sudo pcs -f fs_cfg constraint order WebFS then http_server
sudo pcs cluster cib-push fs_cfg --config

```

Figura 4.18: Risorsa WebFS.

A questo punto è possibile controllare il corretto funzionamento del cluster con il comando **sudo pcs status**. L'output dovrebbe essere simile al seguente.



```
macchina1 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
[cristian@macchina1 ~]$ sudo pcs status
Cluster name: ClusterEsercitazione
Cluster Summary:
  * Stack: corosync
  * Current DC: nodeMacchina2 (version 2.1.2-3.fc34-ada5c3b36e2) - partition with quorum
  * Last updated: Mon Jan  3 13:12:43 2022
  * Last change:  Mon Jan  3 12:49:17 2022 by root via cibadmin on nodeMacchina2
  * 2 nodes configured
  * 5 resource instances configured

Node List:
  * Online: [ nodeMacchina1 nodeMacchina2 ]

Full List of Resources:
  * Resource Group: Fedora_group:
    * floating_ip      (ocf::heartbeat:IPaddr2):      Started nodeMacchina1
    * http_server      (ocf::heartbeat:apache):      Started nodeMacchina1
  * Clone Set: WebData-clone [WebData] (promotable):
    * Masters: [ nodeMacchina1 ]
    * Slaves: [ nodeMacchina2 ]
  * WebFS              (ocf::heartbeat:Filesystem):    Started nodeMacchina1

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
[cristian@macchina1 ~]$
```

Figura 4.19: Controllo lo stato del cluster appena configurato.



# Capitolo 5

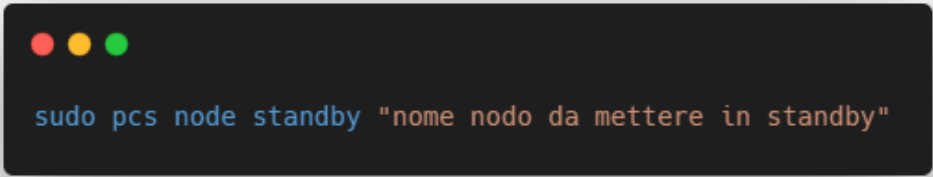
## Conclusioni

### 5.1 Test funzionamento del cluster

Una volta ultimate le configurazioni è necessario testare il corretto funzionamento del cluster. In particolare è richiesto che: nel momento in cui un nodo vada offline, nel caso in cui quest'ultimo sia quello che sta fornendo la risorsa, il cluster deve garantire comunque la completa disponibilità della risorsa richiesta.

Dobbiamo quindi verificare che da una configurazione del cluster in cui un nodo funga da **master** (e fornisca la risorsa), nel momento in cui venga stoppato un altro nodo **slave** subentri da master e continui a fornire la risorsa.

Per fare ciò ho utilizzato il seguente comando (per mettere in standby il nodo master):

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The text inside the terminal is a command to put a node in standby using the 'pcs' tool.

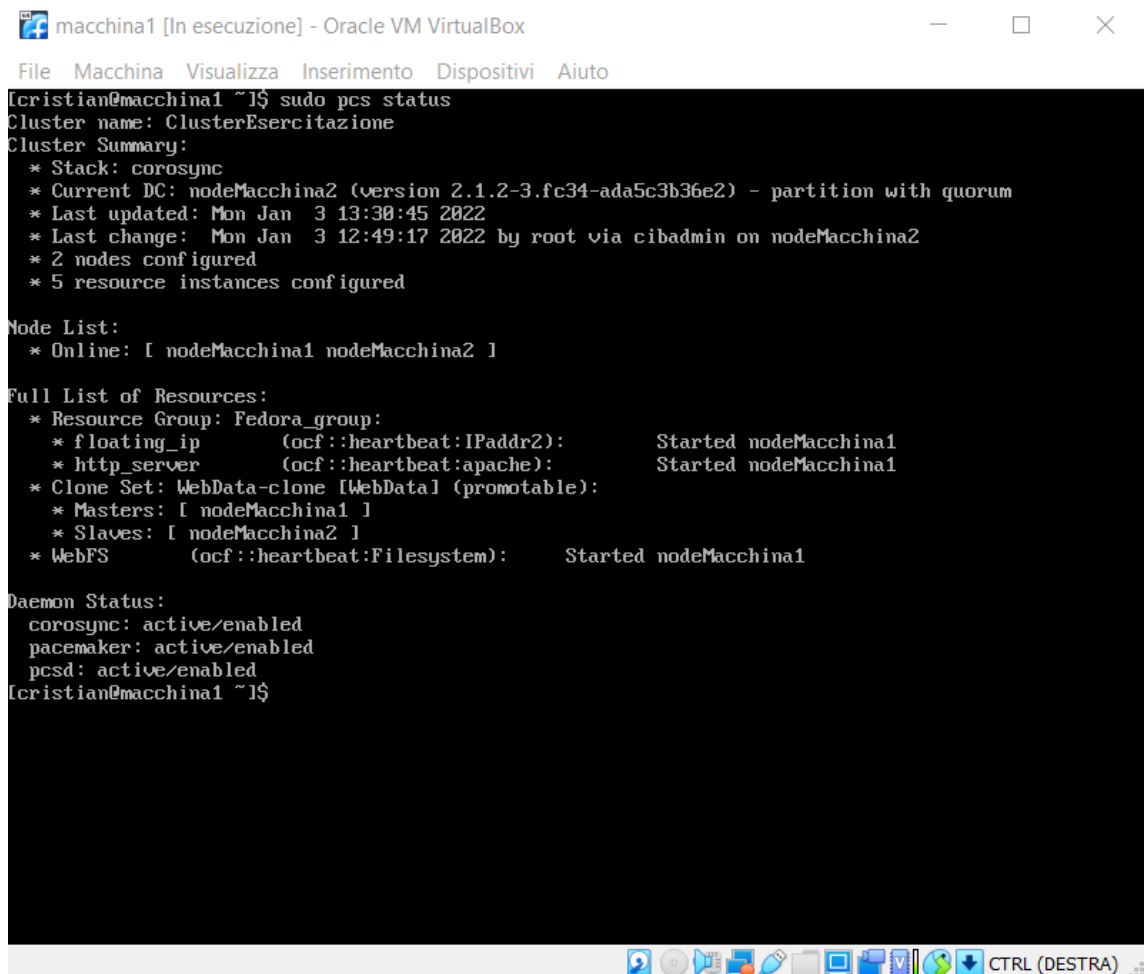
```
sudo pcs node standby "nome nodo da mettere in standby"
```

Figura 5.1: Come mettere in standby un nodo del cluster.

Procediamo ora con il verificare il corretto funzionamento procedendo con le seguenti fasi:

- Controllo lo stato del cluster (controllo chi funge da nodo master e chi da nodo slave, che tutte le risorse siano correttamente disponibili e che tutto sia in funzione);
- metto in standby il nodo master;
- controllo lo stato del cluster e verifico che il nodo slave precedente sia subentrato come nodo master e che tutte le risorse siano ancora disponibili;
- riavvio il nodo in standby;
- verifico che il nodo sia di nuovo online (in questo momento il nodo rimesso online diventa slave, il nodo master resta quello subentrato in precedenza. Gli viene quindi restituito il ruolo solo nel momento in cui ci sia un problema con il nodo attualmente master.)

Nelle figure successive sono riportati in tale ordine i passaggi sopracitati.



```
macchina1 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
[cristian@macchina1 ~]$ sudo pcs status
Cluster name: ClusterEsercitazione
Cluster Summary:
 * Stack: corosync
 * Current DC: nodeMacchina2 (version 2.1.2-3.fc34-ada5c3b36e2) - partition with quorum
 * Last updated: Mon Jan  3 13:30:45 2022
 * Last change: Mon Jan  3 12:49:17 2022 by root via cibadmin on nodeMacchina2
 * 2 nodes configured
 * 5 resource instances configured

Node List:
 * Online: [ nodeMacchina1 nodeMacchina2 ]

Full List of Resources:
 * Resource Group: Fedora_group:
   * floating_ip      (ocf::heartbeat:IPaddr2):      Started nodeMacchina1
   * http_server      (ocf::heartbeat:apache):      Started nodeMacchina1
 * Clone Set: WebData-clone [WebData] (promotable):
   * Masters: [ nodeMacchina1 ]
   * Slaves: [ nodeMacchina2 ]
 * WebFS              (ocf::heartbeat:Filesystem):   Started nodeMacchina1

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
[cristian@macchina1 ~]$
```

Figura 5.2: Controllo lo stato del cluster inizialmente.

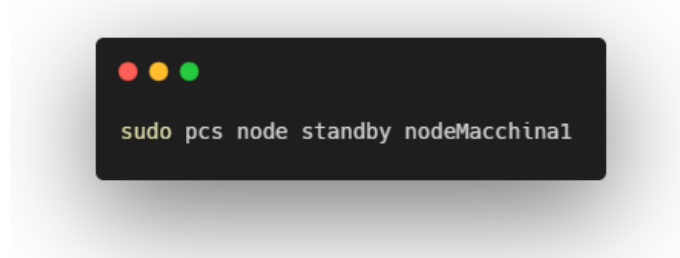
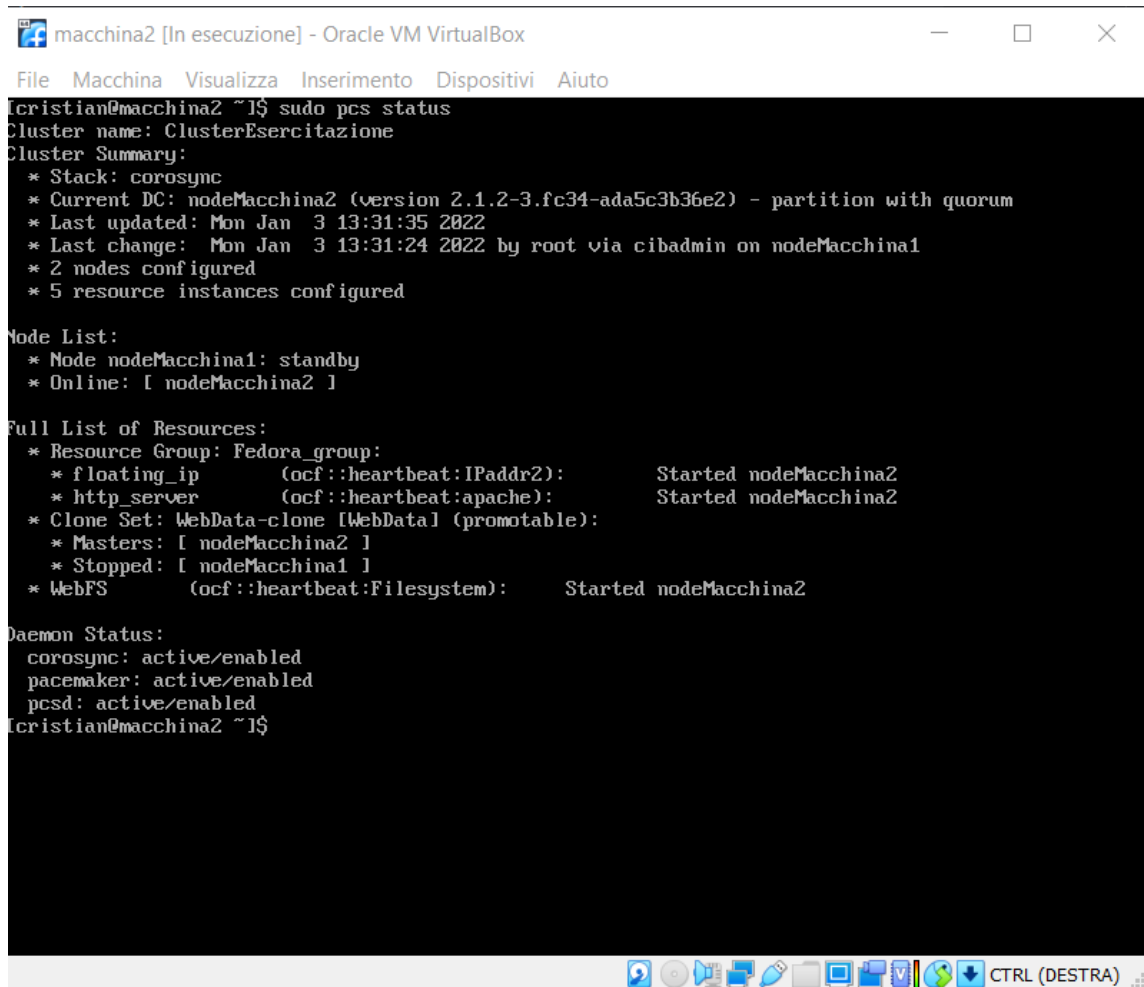


Figura 5.3: Provoco la caduta del nodo master.



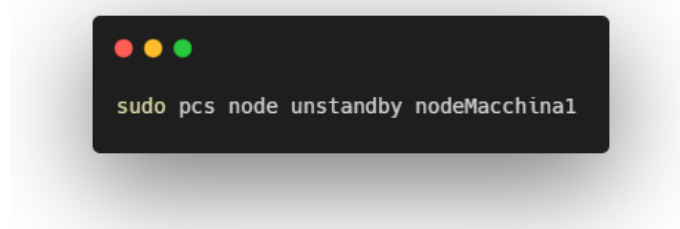
```
macchina2 [In esecuzione] - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
cristian@macchina2 ~1$ sudo pcs status
Cluster name: ClusterEsercitazione
Cluster Summary:
 * Stack: corosync
 * Current DC: nodeMacchina2 (version 2.1.2-3.fc34-ada5c3b36e2) - partition with quorum
 * Last updated: Mon Jan  3 13:31:35 2022
 * Last change:  Mon Jan  3 13:31:24 2022 by root via cibadmin on nodeMacchina1
 * 2 nodes configured
 * 5 resource instances configured

Node List:
 * Node nodeMacchina1: standby
 * Online: [ nodeMacchina2 ]

Full List of Resources:
 * Resource Group: Fedora_group:
   * floating_ip      (ocf::heartbeat:IPAddr2):      Started nodeMacchina2
   * http_server      (ocf::heartbeat:apache):       Started nodeMacchina2
 * Clone Set: WebData-clone [WebData] (promotable):
   * Masters: [ nodeMacchina2 ]
   * Stopped: [ nodeMacchina1 ]
 * WebFS             (ocf::heartbeat:Filesystem):    Started nodeMacchina2

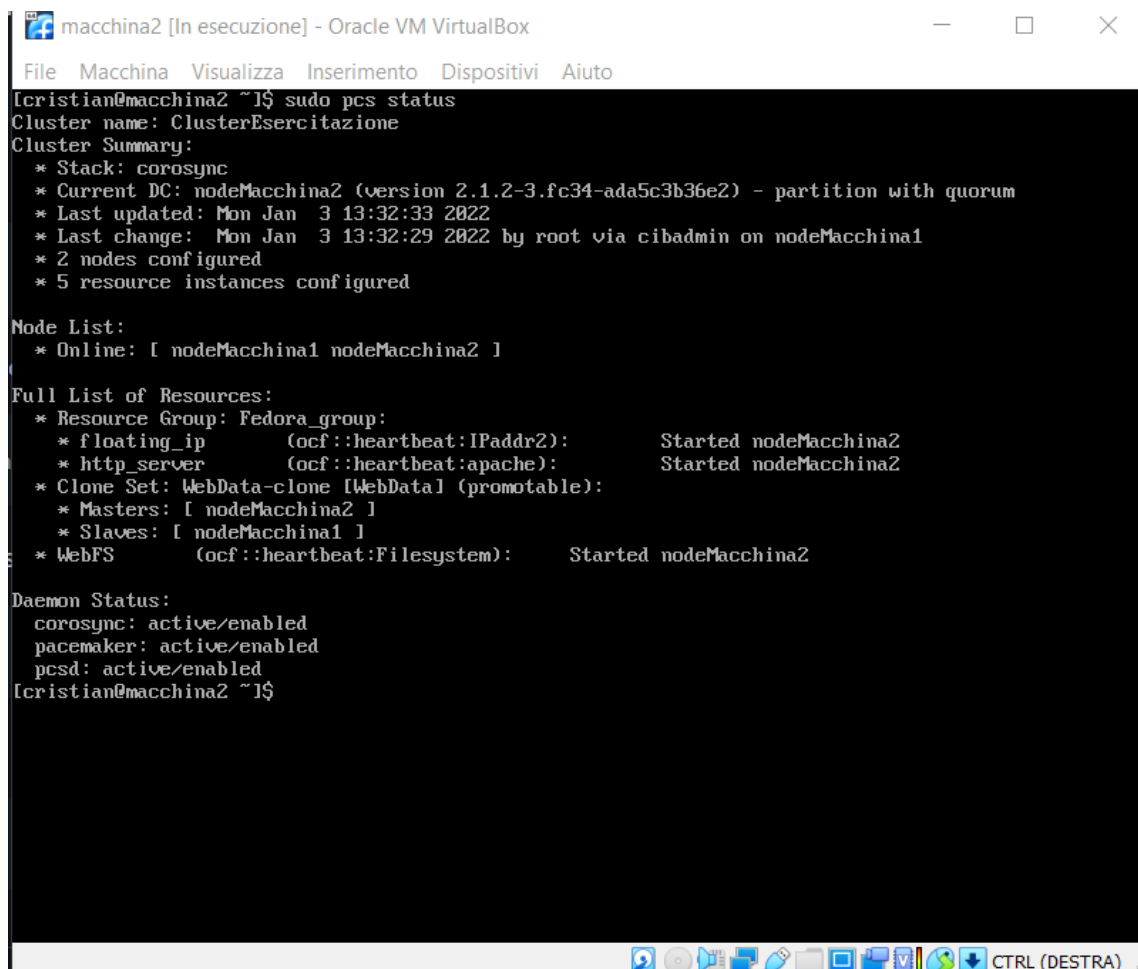
Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
cristian@macchina2 ~1$
```

Figura 5.4: Controllo lo stato del cluster e verifico che sia subentrato il nodo che prima era slave.



```
sudo pcs node unstandby nodeMacchina1
```

Figura 5.5: Riattivo il nodo in standby.



```
macchina2 [In esecuzione] - Oracle VM VirtualBox
File  Macchina  Visualizza  Inserimento  Dispositivi  Aiuto
[cristian@macchina2 ~]# sudo pcs status
Cluster name: ClusterEsercitazione
Cluster Summary:
 * Stack: corosync
 * Current DC: nodeMacchina2 (version 2.1.2-3.fc34-ada5c3b36e2) - partition with quorum
 * Last updated: Mon Jan  3 13:32:33 2022
 * Last change: Mon Jan  3 13:32:29 2022 by root via cibadmin on nodeMacchina1
 * 2 nodes configured
 * 5 resource instances configured

Node List:
 * Online: [ nodeMacchina1 nodeMacchina2 ]

Full List of Resources:
 * Resource Group: Fedora_group:
   * floating_ip      (ocf::heartbeat:IPaddr2):      Started nodeMacchina2
   * http_server      (ocf::heartbeat:apache):      Started nodeMacchina2
 * Clone Set: WebData-clone [WebData] (promotable):
   * Masters: [ nodeMacchina2 ]
   * Slaves: [ nodeMacchina1 ]
 * WebFS              (ocf::heartbeat:Filesystem):    Started nodeMacchina2

Daemon Status:
 corosync: active/enabled
 pacemaker: active/enabled
 pcsd: active/enabled
[cristian@macchina2 ~]#
```

Figura 5.6: Controllo lo stato del cluster e verifico che il nodo riattivato sia slave.

Come appurato il cluster funziona correttamente e riesce a prevenire eventuali problematiche di disponibilità di risorse dovute alla caduta di un nodo. In questo particolare esempio sono stati configurati solo due nodi, ma è ovviamente possibile e consigliato implementarne un numero superiore in un'eventuale applicazione reale.