Università di Perugia
Dipartimento di Matematica e Informatica

Esame HPC

Esercitazione 1 - Pacemaker Cluster

Professore
Prof. Osvaldo Gervasi

Studente
Nicolò Vescera

Anno Accademico 2021-2022

1 Obbiettivo

L'obiettivo di questa esercitazione è quello di realizzare un cluster di 2 nodi nel quale si dovrà avere *Pacemaker* come Cluster Resource Manager (*CRM*), *Corosync* come *Cluster Engine*, *Apache* come *Web Server* e *DRBD* per creare una risorsa replicata in tutti i nodi del cluster (*Distributed Replicated Storage System*).

2 Ambiente di Lavoro

Questa esercitazione è stata svolta all'interno del seguente ambiente di lavoro:

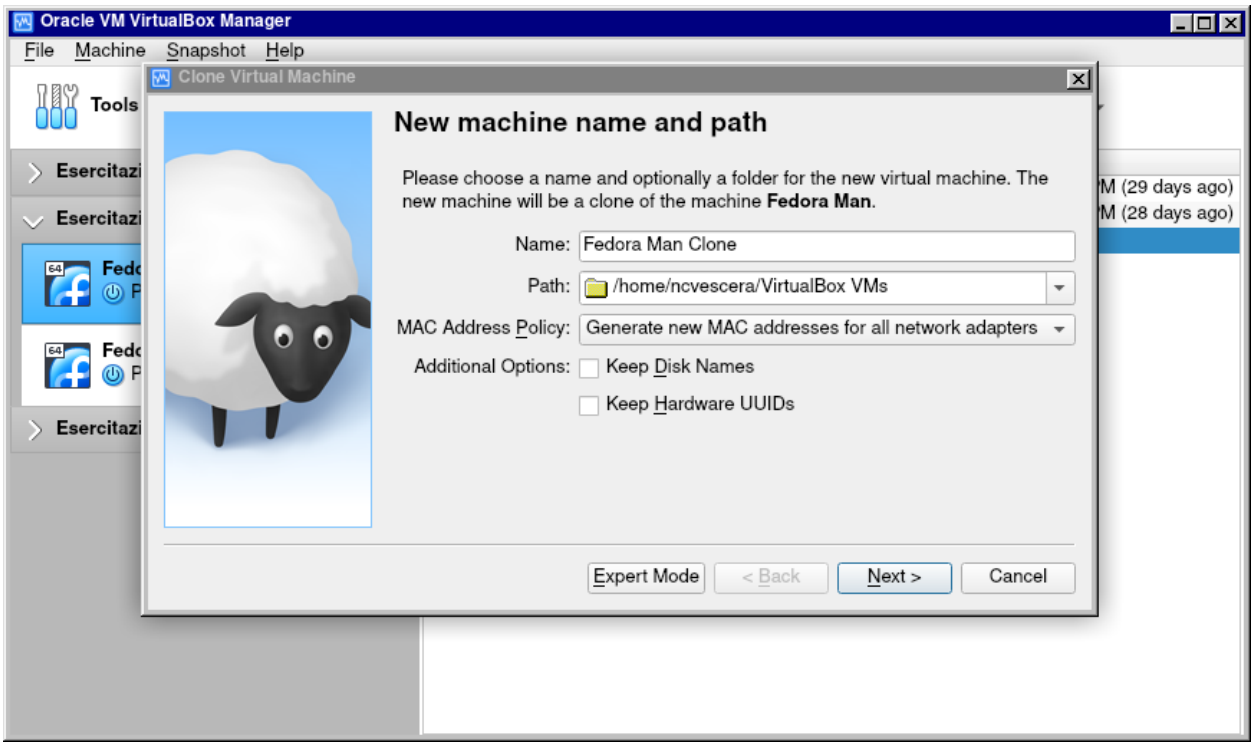
- **Hardware**:
  - **CPU**: AMD Ryzen 9 5900x
  - **RAM**: 32 GB DDR4 @3200 MHz
- **Software**:
  - **Host OS:** Arch Linux
  - **Guest OS**: Fedora 34 Server
  - **Virtualization Software**: VirtualBox 6.1

3 Configurazione Macchine

Per questa esercitazione sono necessarie 2 macchine virtuali che saranno i due nodi del nostro cluster. Ogni macchina è stata configurata come segue:

- **Cores:** 5 Core
- **RAM:** 5GB
- **Dischi di archiviazione:**
  - Disco Principale da 25 GB
  - Disco per risorsa condivisa da 1 GB
- **Scheda di Rete:** Scheda di rete con Bridge

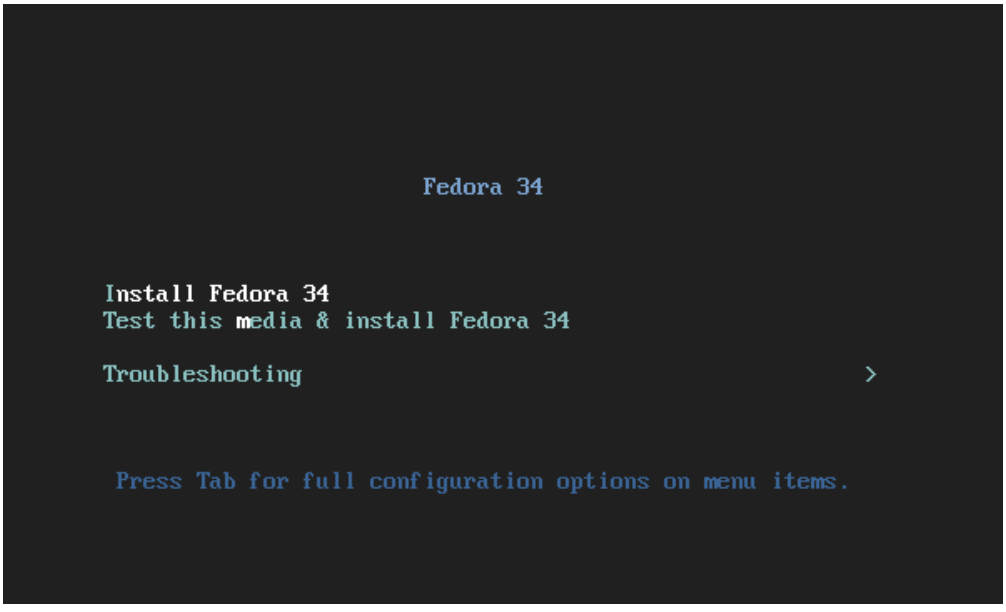
È consigliato configurare una sola macchina, installare e configurare il sistema operativo e i software necessari, per poi utilizzare la funzione 'Clona' di VirtualBox per generare una copia identica senza dover ripetere tali operazioni nuovamente. Durante la fase di clonazione della macchina è necessario selezionare l'opzione "Generare nuovi Mac Address per ogni Network Adapter" come policy per la gestione dei Mac Address.



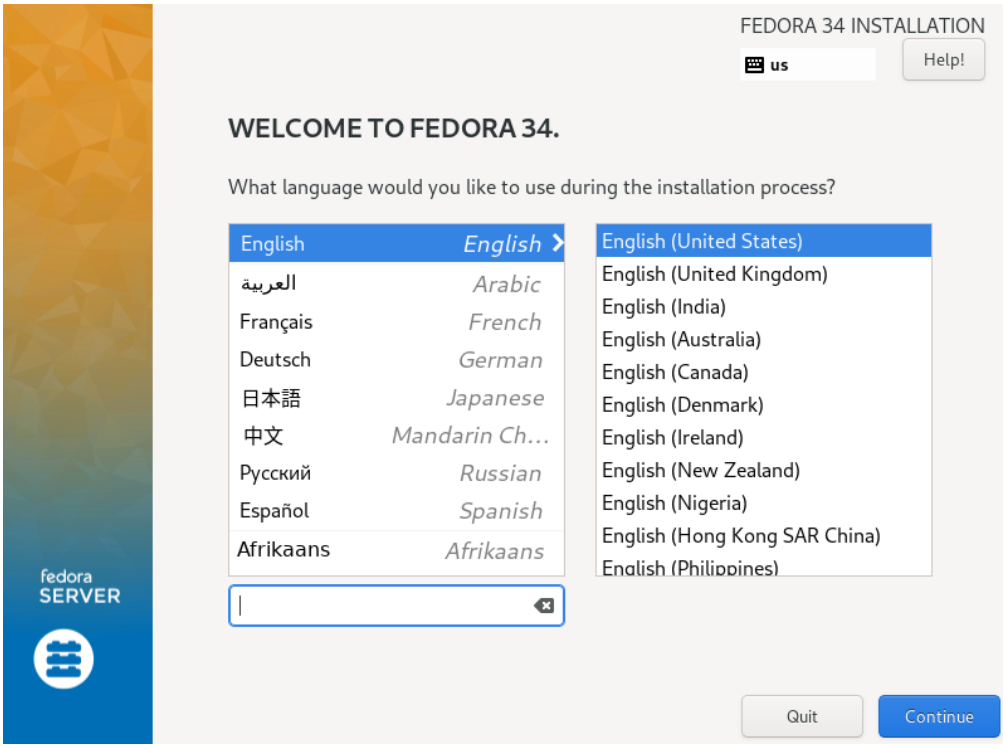
4 Configurazione Software

4.1 Sistema Operativo

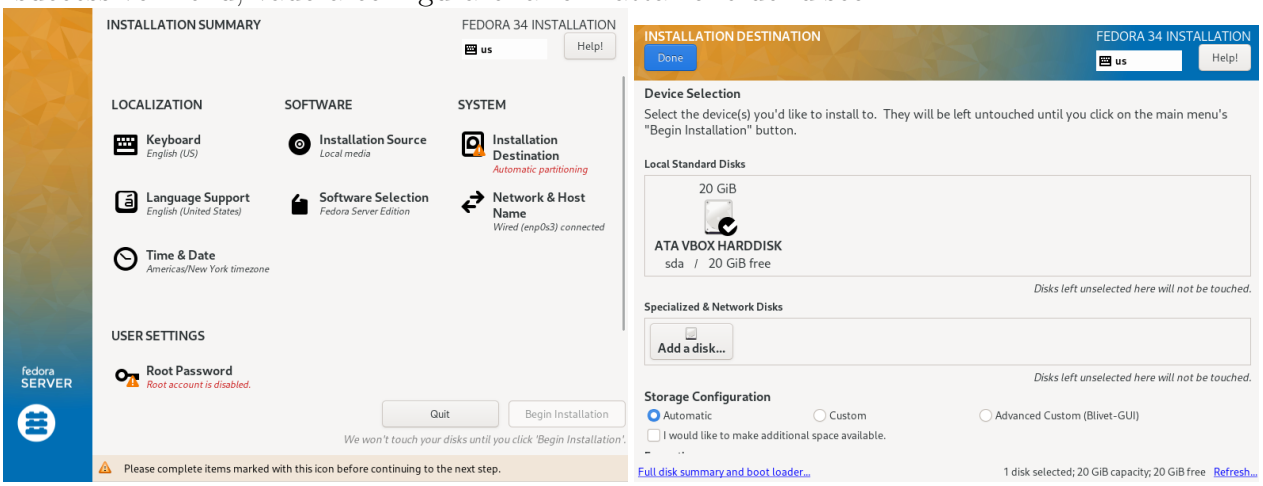
Una volta configurata la macchina virtuale e aggiunta la iso di Fedora, avvio la macchina e scelgo, dal menu, la voce `Install Fedora 34`.



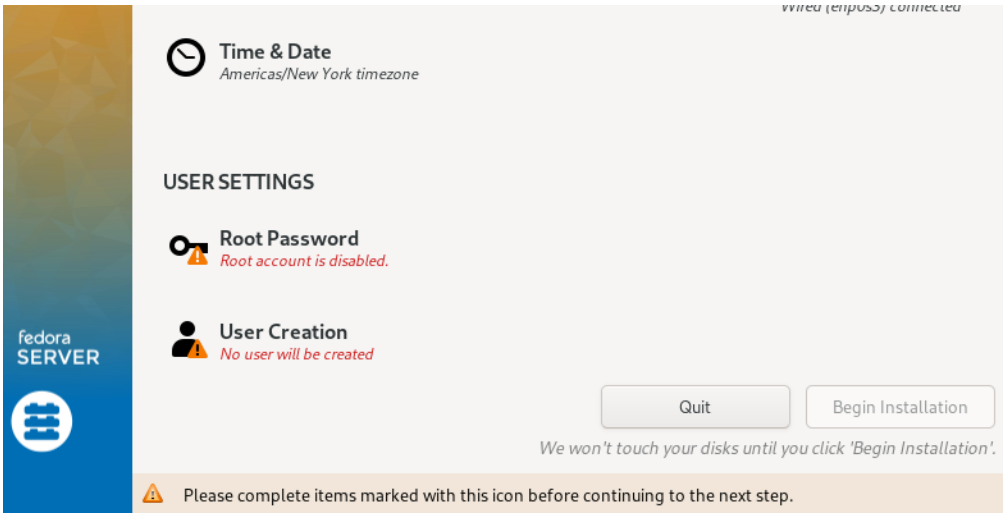
Seleziono la lingua del sistema. In Fedora 34 c'è un bug per cui se la lingua italiana viene selezionata non viene mostrata la sezione per configurare un nuovo utente nel menu successivo. è anche per questo motivo che scelgo la lingua inglese.

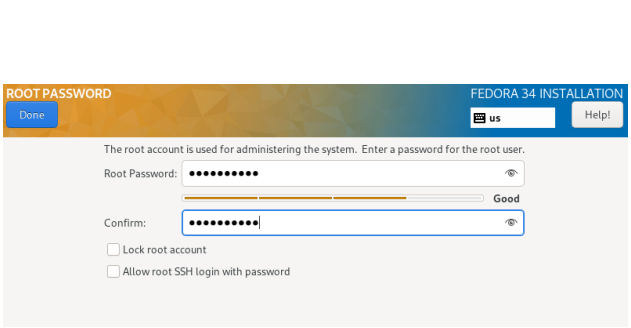


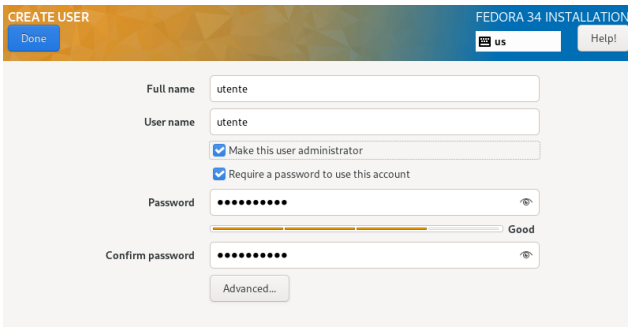
Dal successivo menu, vado a configurare la formattazione del disco.



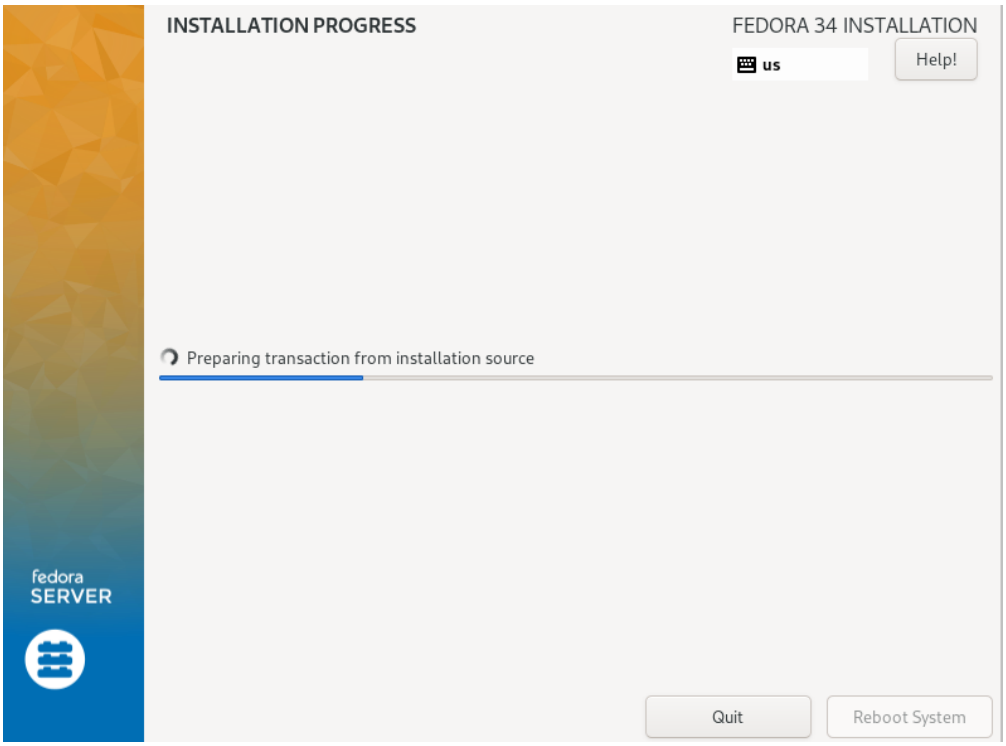
A questo punto comparirà una nuova voce per configurare anche l'utente normale. Procedo prima a scegliere una password per `root` e poi imposto l'utente base.



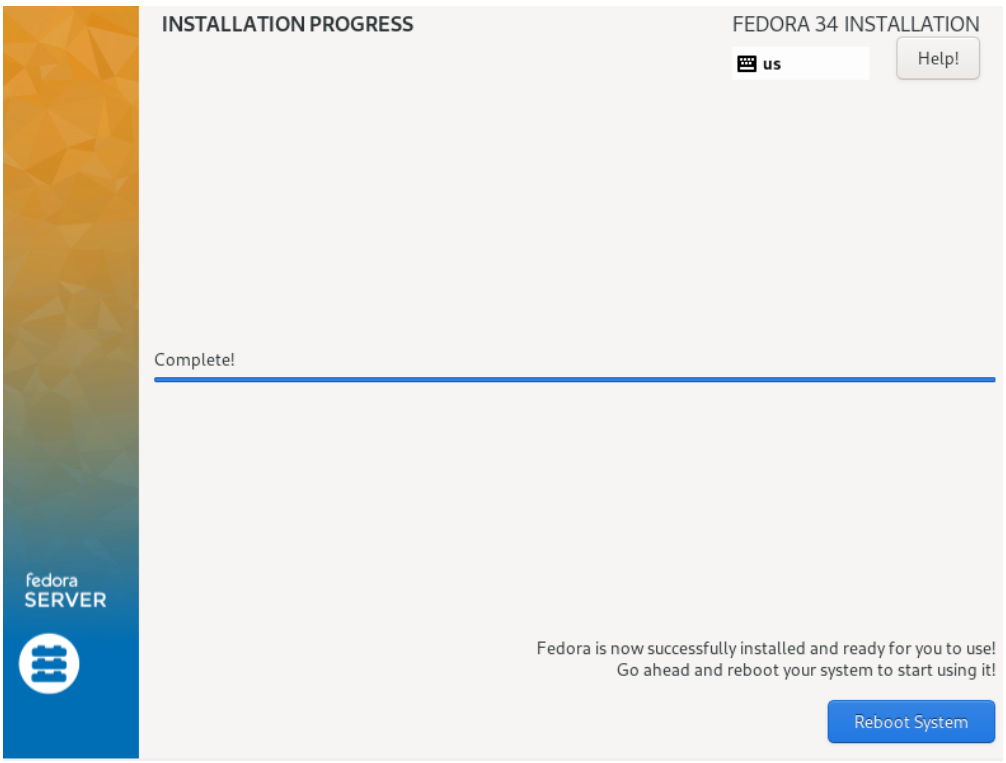




Avvio il processo di installazione del SO.



L'installazione è completata e riavvio il sistema.



4.2 Software Necessario

Appena ho terminato la fase di installazione del SO, ho provveduto ad aggiornarlo con il seguente comando, per evitare problemi di compatibilità e software obsoleto:

```
sudo dnf -y upgrade
```

Poi ho installato i pacchetti necessari (*Pacemaker*, *Corosync*, *Apache* e *DRBD*) con:

```
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
```

4.3 IP

Ho assegnato IP statici alle macchine per essere sicuro di poterle sempre raggiungere e che il DHCP del mio router non gli assegni indirizzi diversi col passare del tempo. Per farlo ho utilizzato i seguenti comandi:

- Macchina 1:

```
sudo nmcli connection modify enp0s3 IPv4.address
   192.168.178.52/24
sudo nmcli connection modify enp0s3 IPv4.gateway
   192.168.178.1
sudo nmcli connection modify enp0s3 IPv4.dns 8.8.8.8
sudo nmcli connection modify enp0s3 IPv4.method manual
```

- Macchina 2:

```
sudo nmcli connection modify enp0s3 IPv4.address
   192.168.178.53/24
sudo nmcli connection modify enp0s3 IPv4.gateway
   192.168.178.1
sudo nmcli connection modify enp0s3 IPv4.dns 8.8.8.8
sudo nmcli connection modify enp0s3 IPv4.method manual
```

Con la prima riga di ogni set di istruzioni vado a specificare l'indirizzo IP che voglio assegnare alla macchina con tanto di subnet mask, la seconda riga indica l'indirizzo del default gateway e la terza specifica il DNS che voglio utilizzare. `enps0s3` è il nome della scheda di rete che sto configurando.
Per rendere effettive le modifiche basta riavviare il sistema: `sudo reboot`.
Si può controllare il successo di questa operazione analizzando l'output del comando

```
route -n
```

se restituisce qualcosa vuol dire che il tutto è andato a buon fine.

```
Kernel IP routing table
Destination     Gateway         Genmask         ...   Iface
0.0.0.0         192.168.178.1   0.0.0.0         ...   enp0s3
192.168.178.0   0.0.0.0         255.255.255.0   ...   enp0s3
```

Un possibile esempio di output.

4.4 hosts & hostname

Ho modificato il file `/etc/hostname` definendo un nome diverso per ogni macchina in modo tale da poterle distinguere più facilmente durante le sessioni SSH:

- Nome Macchina 1: fedoraman
- Nome Macchina 2: fedoragirl

In entrambe le macchine, alla fine del file `/etc/hosts` ho aggiunto le seguenti righe per facilitare poi la configurazione degli altri servizi:

```
192.1168.178.52 Fedoraman
192.1168.178.53 Fedoragirl
```

4.5 Firewall

Per evitare problemi di comunicazione tra le varie macchine ho deciso di disabilitare il firewall:

```
sudo systemctl stop firewalld
sudo systemctl disable firewalld
```

4.6 PCSD

le operazioni di questa sezione sono state eseguite in entrambe le macchine

Per prima cosa ho cambiato la password dell'utente `hacluster` in quanto servirà per l'autenticazione dei vari nodi in alcuni passaggi successivi:

```
sudo passwd hacluster
```

Successivamente ho abilitato il servizio e l'ho avviato con:

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

4.7 Corosync

In entrambe le macchine ho provveduto alla configurazione di `corosync` andando a popolare il file `/etc/corosync/corosync.conf` con il seguente contenuto:

```
totem {
  version: 2
  cluster_name: ExampleCluster
  transport: knet
  crypto_cipher: aes256
  crypto_hash: sha256
}

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

  node {
     ring0_addr: Fedoragirl
     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
}
```

Nella sezione `totem` ho impostato il nome del cluster modificando l'attributo `cluster_name`. La sezione `nodelist` conterrà l'elenco di tutti i nodi del cluster, per ogni nodo ho aggiunto la sottosezione `node{}` con gli attributi:

- `ring0_addr`: indica l'indirizzo del nodo, nel mio caso è `Fedoraman/Fedoragilr` per via della configurazione in sottosezione 4.4
- `name`: il nome da assegnare al nodo
- `nodeid`: un numero progressivo che identifica il nodo

4.8 Autenticazione dei Nodi

In ogni macchina ho effettuato l'autenticazione del nodo al cluster con i seguenti comandi:

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

La password da utilzzare in questa fase è quella dell'utente `hacluster` scelta al punto sottosezione 4.6

4.9 Avvio Cluster

Configuro l'avvio automatico del cluster e lo faccio partire con i seguenti comandi:

```
sudo pcs cluster setup ExampleCluster node1 node2 --force
sudo pcs cluster start --all
sudo pcs cluster enable --all
```

Questa operazione deve essere eseguita in una sola macchina, non importa quale, io per comodità ho scelto `Fedoraman(node1)`.
Controllo il corretto funzionamento del cluster tramite il comando:

```
sudo pcs status
```

Se ho un output come il seguente vuol dire che il tutto è andato a buon fine (è importante che i nodi risultino `Online`):

```
Cluster name: ExampleCluster

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

Cluster Summary:
   * Stack: corosync
   * Current DC: node1 (version 2.1.1-9.fc34-77db578727) -
      partition with quorum
   * Last updated: Wed Nov  3 18:17:32 2021
   * Last change:  Wed Nov  3 18:02:47 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
```

4.10 Cluster Property

Ho disabilitato le property `stonith` e `quorum` in quanto la prima non è necessaria ai fini di questa esercitazione e la seconda dato che non ho un numero sufficiente di macchine per utilizzare questa proprietà (ne servono minimo 3).

```
sudo pcs property set stonith-enabled=false
sudo pcs property set no-quorum-policy=ignore
```

4.11 Cluster IP

Assegno un IP al cluster in modo tale da raggiungere il server web, che verrà configurato in seguito, e altri possibili servizi:

```
sudo pcs resource create floating_ip ocf:heartbeat:IPaddr2 ip
   =192.168.178.55 cidr_netmask=24 op monitor interval=60s
```

L'indirizzo viene specificato con il parametro `ip=` e la subnet mask con `cidr_netmask=`. È importante notare che l'IP scelto deve appartenere alla stessa rete delle macchine e non deve essere utilizzato da nessun altro dispositivo !

4.12 Risorsa HTTP

Aggiungo la risorsa HTTP al cluster in modo tale da avere un server web sempre attivo anche se il nodo principale cade:

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

4.13 Partizionamento Disco Condiviso

Procedo al partizionamento del secondo disco presente in tutte e due le macchine.
Con il seguente comando posso controllare il suo nome che gli è stato assegnato (di solito `/dev/sdb`):

```
sudo fdisk -l
```

Partiziono effettivamente il disco con:

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

Si aprirà un menu interattivo e dovrò digirare i seguenti comandi:

- `n`: nuova partizione
- `p`: partizione primaria
- `1`: numero di partizioni
- `ENTER`: primo settore (utilizzare il valore di default)
- `ENTER`: ultimo settore (utilizzare il valore di default)
- `w`: scrive le modifiche
- `q`: esce dal programma

Oppure posso effettuare il tutto in maniera autoamtica con:

```
sed -e 's/\s*\([\+0-9a-zA-Z]*\).*/\1/' << EOF | fdisk /dev/sdb
   n # new partition
   p # primary partition
   1 # partition number 1
     # default - start at beginning of disk
     # default - stop at ending of disk
   w # write the partition table
   q # and we're done
EOF
```

4.14 Configurazione DRBD

Prima di tutto, devo andare a modificare la policy di sicurezza di SELinux per DRBD, dato che ne impedisce (di defualt) il corretto funzionamento, tramite il seguetne comando:

```
sudo semanage permissive -a drbd_t
```

Fatto ciò, posso andare a configurare DRBD popolando il file di configurazione `/etc/drbd.d/wwwdata.res` con il seguente contenuto:

```
resource wwwdata {
   protocol C;
   device /dev/drbd0;

   syncer{
       verify-alg sha1;
   }

   net {
       cram-hmac-alg sha1;
       shared-secret "barisoni";
   }
   on fedoraman {
       disk /dev/sdb1;
       address 192.168.178.52:7788;
       meta-disk internal;
   }
   on fedoragirl {
       disk /dev/sdb1;
       address 192.168.178.53:7788;
       meta-disk internal;
   }
}
```

Ho scelto una chiave per l'algoritmo di crittografia tramite la proprietà `shared-secret`. È importante notare che `on fedoraman` e `on fedoragirl` sono i nomi che ho impostato nel punto sottosezione 4.4 e nelle relative sezioni ho specificato il nome del disco da utilizzare (sarà sempre `/dev/sdb1` per via della configurazione in sottosezione 4.13), l'indirizzo IP della macchina (non sembra possibile utilizzare gli alias definiti in sottosezione 4.4) e una porta a scelta per la comunicazione (ho utilizzato la `7788`)

Creo dunque la risorsa drbd:

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

Abilito il modulo kernel per rendere sempre utilizzabile la risorsa, anche ai prossimi riavvii:

```
sudo modprobe drbd
echo "drbd" | sudo tee -a /etc/modules-load.d/drbd.conf
```

Completo la configurazione della risorsa con:

```
sudo drbdadm up wwwdata
sudo drbdadm -- --overwrite-data-of-peer primary all

sudo drbdadm primary --force wwwdata
```

Il terzo comando deve essere eseguito solo sulla prima macchina !
Per monitorare l'avanzamento del processo ho utilizzato il seguente comando: `watch cat /proc/drbd`.

Infine la abilito per far si che si avvii in automatico:

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

4.15 Formattazione Risorsa DRBD

Procedo con la formattazione della risorsa DRBD creata in precedenza e con l'aggiunta di una pagina web per controllare il corretto funzionamento:

```
mkfs.xfs /dev/drbd0
sudo mount /dev/drbd0 /mnt
echo "<h1>Hello World</h1>" | sudo tee /mnt/index.html
sudo umount /dev/drbd0
```

4.16 Risorsa DRBD

Aggiungo al cluster la risorsa DRBD creata in precedenza:

```
sudo pcs cluster cib drbd_cfg
sudo pcs -f drbd_cfg resource create WebData ocf:linbit:drbd
   drbd_resource=wwwdata 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
```

Controllo il risultato di questa operazione con il comando:

```
sudo pcs status
```

Mi aspetto un outpit simile al seguente:

```
Cluster name: ExampleCluster
Cluster Summary:
   * Stack: corosync
   * Current DC: node1 (version 2.1.1-9.fc34-77db578727) -
      partition with quorum
   * Last updated: Tue Dec 14 10:39:46 2021
   * Last change:  Mon Nov  8 10:35:32 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: Fedora_group:
       * floating_ip   (ocf::heartbeat:IPaddr2):
          Started node1
       * http_server   (ocf::heartbeat:apache):
          Started node1
   * Clone Set: WebData-clone [WebData] (promotable):
       * Masters: [ node1 ]
       * Slaves: [ node2 ]

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

4.17 Risorsa WebFS

Per ultimo aggiungo al cluster la risorsa WebFS per far si che i dati del Web Server siano replicati in tutti i nodi:

```
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
```

Nei comandi dove compare la clausula `order` vado a specificare l'ordine con cui le risorse devono essere avviate, per esempio con `order WebFS then http_server` sto dicendo al cluster di avviare prima la risorsa WebFs e poi il server web.

4.18 Failover Test

Come test finale ho simulato un failover del nodo principale con il seguente comando:

```
sudo pcs node standby node1
```

Controllo quindi che `node2` venga promosso a master e si avvii la risorsa WebFs:

```
sudo pcs status
```

```
Cluster name: ExampleCluster
Cluster Summary:
   * Stack: corosync
   * Current DC: node1 (version 2.1.1-9.fc34-77db578727) -
      partition with quorum
   * Last updated: Tue Dec 14 10:53:27 2021
   * Last change:  Tue Dec 14 10:53:10 2021 by root via
      cibadmin on node1
   * 2 nodes configured
   * 5 resource instances configured

Node List:
   * Node node1: standby
   * Online: [ node2 ]

Full List of Resources:
   * Resource Group: Fedora_group:
       * floating_ip   (ocf::heartbeat:IPaddr2):
          Started node2
       * http_server   (ocf::heartbeat:apache):
          Started node2
   * Clone Set: WebData-clone [WebData] (promotable):
       * Masters: [ node2 ]
       * Stopped: [ node1 ]
   * WebFS     (ocf::heartbeat:Filesystem):     Started node2

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

Una volta assicurato il successo di questa operazione riporto in vita `nodo1` con:

```
sudo pcs node unstandby node1
```