

Relazione di progetto di Virtualizzazione e integrazione di sistemi Server NFS e volumi di container

Luca Casadei
Matricola: 0001069237

17 giugno 2025

Indice

1 Creazione delle macchine virtuali	1
1.1 Ambiente di virtualizzazione	1
1.2 Creazione server NFS	2
1.2.1 Cenni su NFS e NFSv4	2
1.2.2 Effettiva creazione	2
1.2.3 Installazione di Docker Engine e Compose	3
1.2.4 Creazione e configurazione del container nfs-server	4
1.3 Creazione della seconda macchina virtuale (WebServer)	5
1.3.1 Sistema operativo	5
1.3.2 Comunicazione tra macchine virtuali	6
1.3.3 Prova specifica NFS	8
2 Creazione di un servizio che si appoggia su volumi NFS per la persistenza	8
2.1 Creazione del sito web di prova e database	8
2.2 Test conclusivo	9
3 Conclusioni	10
3.1 Struttura del progetto e password	10

1 Creazione delle macchine virtuali

In questa sezione verrà indicata la modalità utilizzata per la creazione delle macchine virtuali, in particolar modo verrà descritto il modo usato per metterle in comunicazione.

1.1 Ambiente di virtualizzazione

Come ambiente di virtualizzazione è stato scelto *Proxmox*, un software debian-based open source che permette di gestire macchine virtuali basato sull'infrastruttura KVM che fornisce già un repository di immagini LXC (Linux Containers).

Procedo quindi ad un'installazione "fresca" di Proxmox su una workstation, a seguito di tutte le fasi di installazione, mi trovo nella seguente situazione:

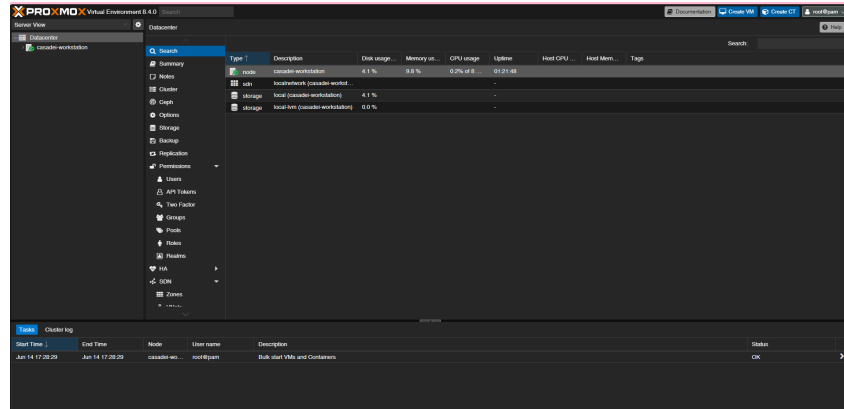


Figura 1: Proxmox installato su workstation

1.2 Creazione server NFS

1.2.1 Cenni su NFS e NFSv4

Il protocollo NFS (Network File System) versione 4 descritto dalla [1, rfc7530] è un'evoluzione dello stesso protocollo in versione 3 e 2 descritti rispettivamente dalle [2, rfc1813] e [3, rfc1094], l'idea alla base del protocollo è quella di rendere accessibile delle risorse (file) all'interno della rete, indipendentemente dal sistema operativo o architettura di rete. Per realizzare questa astrazione il protocollo fa uso di primitive chiamate RPC (Remote Procedure Call) ed una rappresentazione dei dati esterna a quella dei vari sistemi operativi, ma interpretabile da tutti attraverso la rete, la XDR (eXternal Data Representation).

NFS era (originariamente) pensato per essere un protocollo stateless, quindi non manteneva informazioni riguardanti i client che ne facevano uso. La versione 2 introduce la possibilità di gestire file di dimensioni notevolmente superiori rispetto alla versione precedente, oltre all'introduzione di nuove primitive e migliorie a livello di integrità dei dati in caso di operazioni asincrone.

Dalla versione 4, con l'introduzione del file locking (modalità per irrobustire l'integrità dei file trasferiti) il protocollo è necessariamente diventato stateful, rendendo quindi incompatibili il protocollo in versione 4 e le versioni precedenti, nonostante questo gran parte dei server nfs consentono di configurare il protocollo in più versioni diverse.

1.2.2 Effettiva creazione

Ora andrò a creare la macchina virtuale che conterrà il container per il server NFSv4, non uso un container LXC Proxmox perché NFSv4 richiede accesso kernel-level. Come sistema da virtualizzare ho scelto Debian 12, che scarico e aggiungo alle ISO Images di Proxmox (nell'hdd secondario), con il seguente risultato:

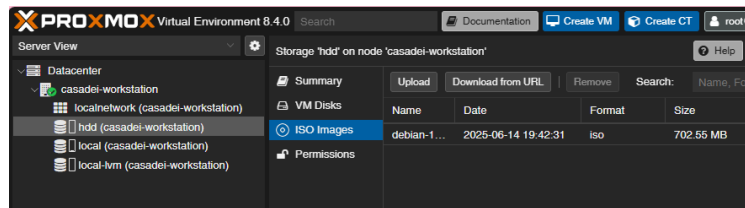


Figura 2: Aggiunta ISO Debian 12 a Proxmox

Ora procedo a creare la macchina virtuale effettiva seguendo tutte le configurazioni di Proxmox, questa è la finestra di creazione finalizzata:

Create: Virtual Machine

General OS System Disks CPU Memory Network **Confirm**

Key ↑	Value
cores	1
cpu	x86-64-v2-AES
ide2	hdd:iso/debian-12.11.0-amd64-netinst.iso,media=cdrom
memory	2048
name	nfs-server
net0	virtio,bridge=vbr0,firewall=1
nodename	casadei-workstation
numa	0
ostype	l26
scsi0	hdd:20,format=qcow2,iotread=on
scsihw	virtio-scsi-single
sockets	1
vmid	100

☐ Start after created

Advanced ☐ **Back** **Finish**

Figura 3: Creazione della macchina virtuale per il server NFSv4

Si procede quindi all'installazione di Debian normale, a seguito dell'avvio della macchina virtuale, dopodiché procedo ad installare tutti i pacchetti necessari, a partire da docker:

1.2.3 Installazione di Docker Engine e Compose

Dalla documentazione ufficiale di Docker, aggiungiamo la chiave GPG (GNU Privacy Guard) per poter installare il pacchetto:

```

1 # Add Docker's official GPG key:
2 sudo apt-get update
3 sudo apt-get install ca-certificates curl
4 sudo install -m 0755 -d /etc/apt/keyrings
5 sudo curl -fsSL https://download.docker.com/linux/debian/gpg
   ↪ -o /etc/apt/keyrings/docker.asc

```

```

6      sudo chmod a+r /etc/apt/keyrings/docker.asc
7
8      # Add the repository to Apt sources:
9      echo \
10     "deb [arch=$(dpkg --print-architecture)
11     ↪ signed-by=/etc/apt/keyrings/docker.asc]
12     ↪ https://download.docker.com/linux/debian \
13     $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
14     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
15     sudo apt-get update

```

```
1 sudo apt-get install docker-ce docker-ce-cli containerd.io
   ↪ docker-buildx-plugin docker-compose-plugin
```

```
1 docker ps
```

```
root@nfs-server:/home/luca# docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
root@nfs-server:/home/luca#
```

1.2.4 Creazione e configurazione del container nfs-server

```
1      sudo modprobe {nfs,nfsd}
```

```

#A-NEWVER =====
#A-NEWVER -----
#A-NEWVER SETTING UP ...
#A-NEWVER -----
#A-NEWVER ----> building /etc/exports from environment variables
#A-NEWVER collected 1 valid export(s) from NFS_EXPORTS_ environment variable
#A-NEWVER ----> setup complete
#A-NEWVER -----
#A-NEWVER STARTING SERVICES ...
#A-NEWVER -----
#A-NEWVER ----> starting rpcbind
#A-NEWVER ----> starting exportsd
#A-NEWVER ----> starting rpc.mountd on port 32767
#A-NEWVER ----> starting rpc.statd on port 32768 (outgoing from port 32766)
#A-NEWVER ----> starting rpc.svc on port 2049 with 1 server thread(s)
#A-NEWVER ----> all services started normally
#A-NEWVER -----
#A-NEWVER SERVER STARTUP COMPLETE
#A-NEWVER -----
#A-NEWVER ----> list of enabled NFS protocol versions: 4.2, 4.1, 4, 3
#A-NEWVER ----> list of container exporters:
#A-NEWVER   /exports (/rw,nfs,no_subtree_check)
#A-NEWVER ----> list of container ports that should be exposed:
#A-NEWVER     111 (TCP and UDP)
#A-NEWVER     2049 (TCP and UDP)
#A-NEWVER     32765 (TCP and UDP)
#A-NEWVER     32767 (TCP and UDP)
#A-NEWVER -----
#A-NEWVER READY AND WAITING FOR NFS CLIENT CONNECTIONS

```

1.3 Creazione della seconda macchina virtuale (WebServer)

1.3.1 Sistema operativo

Come sistema operativo della seconda macchina virtuale ho scelto *Alpine Linux* per due ragioni principali, la prima è per mostrare il funzionamento anche su sistemi leggermente diversi, seppur entrambi linux, e questo è estremamente leggero e minimale, la seconda ragione è una personale avversione ad installare sistemi non open source. Questa è la configurazione della macchina virtuale, con un processore Intel.

cores	1
cpu	Skylake-Client
ide2	hdd:iso/alpine-standard-3.22.0-x86_64.iso,media=cdrom
memory	2048
name	web-server
net0	virtio,bridge=vmb0,firewall=1
nodename	casadei-workstation
numa	0
ostype	l26
scsi0	hdd:32,format=qcow2,iothread=on
scsihw	virtio-scsi-single
sockets	1
vmid	101

Figura 6: Creazione della macchina virtuale per il WebServer

Si procede quindi all'installazione su disco di Alpine Linux mediante il comando *"setup-alpine"*, per verificare la corretta impostazione delle interfacce con dhcp si può usare *"ip a"* e si ottiene:

```

Welcome to Alpine!

The Alpine Wiki contains a large amount of how-to guides and general
information about administrating Alpine systems.
See <https://wiki.alpinelinux.org/>.

You can setup the system with the command: setup-alpine

You may change this message by editing /etc/motd.

web-server:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether bc:24:11:e3:d0:82 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.71/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::be24:11ff:fee3:d082/64 scope link
        valid_lft forever preferred_lft forever
web-server:~# _

```

Figura 7: Alpine Linux installato

1.3.2 Comunicazione tra macchine virtuali

Tornando alla prima macchina virtuale e lanciando un ping, si può vedere che le due macchine riescono a comunicare.

```

luca@nfs-server:~$ ping 192.168.1.71
PING 192.168.1.71 (192.168.1.71) 56(84) bytes of data.
64 bytes from 192.168.1.71: icmp_seq=1 ttl=64 time=0.339 ms
64 bytes from 192.168.1.71: icmp_seq=2 ttl=64 time=0.309 ms
64 bytes from 192.168.1.71: icmp_seq=3 ttl=64 time=0.237 ms
64 bytes from 192.168.1.71: icmp_seq=4 ttl=64 time=0.239 ms
64 bytes from 192.168.1.71: icmp_seq=5 ttl=64 time=0.412 ms
64 bytes from 192.168.1.71: icmp_seq=6 ttl=64 time=0.397 ms
^C
--- 192.168.1.71 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5099ms
rtt min/avg/max/mdev = 0.237/0.322/0.412/0.068 ms

```

Figura 8: I due server comunicano con ping

Se vogliamo usare i nomi di dominio, essendo il bridge di proxmox assegnato alle macchine virtuali di default (vmbr0) condiviso con la rete della macchina host (la workstation), la configurazione di rete viene data dal router a cui è collegata la workstation mediante DHCP, e se il router funge anche da server DNS, ci consente di trovare nella rete locale le macchine (virtuali o no) mediante il loro nome di dominio, ottenendo quindi il seguente risultato con ping:

```

Debian GNU/Linux 12 nfs-server tty1
nfs-server login: root
Password:
Linux nfs-server 6.1.0-37-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.140-1 (2025-05-22) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun 16 15:49:40 CEST 2025 on tty1
root@nfs-server:~# ping web-server
PING web-server.homenet.telecomitalia.it (192.168.1.71) 56(84) bytes of data.
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=1 ttl=64 time=0.387 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=3 ttl=64 time=0.412 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=4 ttl=64 time=0.406 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=5 ttl=64 time=0.389 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=6 ttl=64 time=0.379 ms
64 bytes from web-server.homenet.telecomitalia.it (192.168.1.71): icmp_seq=7 ttl=64 time=0.267 ms
^C
--- web-server.homenet.telecomitalia.it ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 601ms
rtt min/avg/max/mdev = 0.287/0.367/0.412/0.045 ms
root@nfs-server:~#

```

Figura 9: Macchine comunicano anche usando hostname

Se questo non fosse possibile, potremmo modificare il file `/etc/hosts` all'interno delle macchine virtuali ed aggiungere manualmente il nome di dominio, ad esempio, questo potrebbe essere il file hosts all'interno della macchina nfs-server che contiene il record dell'altra macchina virtuale:

```

127.0.0.1    localhost
127.0.1.1    nfs-server.nfs-server.casadei.ddns.net  nfs-server
192.168.1.71 web-server.casadei.ddns.net      web-server

# The following lines are desirable for IPv6 capable hosts
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters

```

Figura 10: File hosts in /etc nella macchina nfs-server

se usiamo ping adesso, in nome di dominio completo sarà diverso da quello precedente perché non è più quello fornito dal server dns del router:

```

root@nfs-server:~# ping web-server
PING web-server.casadei.ddns.net (192.168.1.71) 56(84) bytes of data.
64 bytes from web-server.casadei.ddns.net (192.168.1.71): icmp_seq=1 ttl=64 time=0.412 ms
64 bytes from web-server.casadei.ddns.net (192.168.1.71): icmp_seq=2 ttl=64 time=0.401 ms
64 bytes from web-server.casadei.ddns.net (192.168.1.71): icmp_seq=3 ttl=64 time=0.443 ms
64 bytes from web-server.casadei.ddns.net (192.168.1.71): icmp_seq=4 ttl=64 time=0.279 ms
^C
--- web-server.casadei.ddns.net ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.279/0.383/0.443/0.062 ms
root@nfs-server:~#

```

Figura 11: Nome di dominio diverso se in file hosts

Un'ulteriore alternativa sarebbe quella di creare (virtuale o meno) un server DNS all'interno della stessa rete per evitare di andare ad aggiungere ogni hostname nel file `/etc/hosts` di ogni macchina, sarebbe la soluzione più raffina-

ta ma il procedimento per questa modalità verrà omesso perché non rientrante nell'obiettivo del progetto.

1.3.3 Prova specifica NFS

Ora facciamo però una prova specifica con NFS, dopo aver installato su alpine nfs-utils (solo per prova, non servirà nel sistema finale) possiamo montare un percorso connettendoci al server NFS, con i comandi visibili nell'immagine, e successivamente creare un file in una delle due cartelle predisposte per l'esportazione dal server NFS, ad esempio "db":

```
web-server:~# mount -o rw,nolock -t nfs 192.168.1.165:/exports /mnt
web-server:~# touch /mnt/exports/
db/  web/
web-server:~# touch /mnt/exports/db/prova.txt
web-server:~# _
```

Figura 12: Mount del percorso esportato e creazione di file di prova

Questo file deve ora essere visibile anche dal server, nella corrispondente cartella:

```
root@nfs-server:/home/luca# ls /srv/nfs/exports/db/
prova.txt
root@nfs-server:/home/luca#
```

Figura 13: file visibile nel server NFS

2 Creazione di un servizio che si appoggia su volumi NFS per la persistenza

Questa è la parte cruciale del progetto, la creazione di un insieme di container che adottino persistenza dei dati sul nostro server NFS, partiamo quindi dalla creazione del file compose che metta in esecuzione un semplicissimo e basilare sito web, ed il relativo database, i cui dati devono essere persistenti e appunto, su volumi NFS.

2.1 Creazione del sito web di prova e database

Questi codici sono consultabili dalla repository, non essendo questo progetto incentrato sulla programmazione web verranno omessi i passaggi per realizzarli in questa relazione, elenco brevemente le tecnologie usate:

- **MariaDB**: Database per testare la persistenza dei dati su volumi NFS
- **PHP**: Lato server
- **Javascript + HTML**: con Apache come server http

Il sito presenta un semplice elenco di elementi a caso che si possono aggiungere mediante apposito form, nulla di più. Il database viene inizialmente popolato dallo script `crea-tabella.sql` montato come volume di tipo `bind` durante l'inizializzazione del container, dato che abbiamo la persistenza dei dati tramite NFS, se vogliamo svuotare completamente il database e forzare di nuovo l'inizializzazione dobbiamo rimuovere i file che MariaDB crea all'interno della cartella condivisa, o dal server NFS o dal container stesso. Il container per PHP e Apache viene creato secondo l'apposito Dockerfile corredato assieme alla relazione nella cartella `web/sito`.

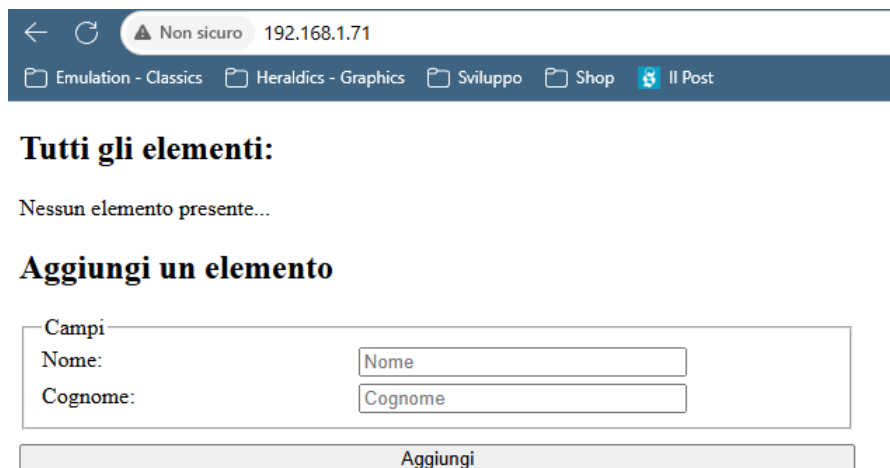
2.2 Test conclusivo

Quando lanciamo `"docker compose up -d"` partendo da `web-docker-compose.yml` si ottiene:

```
[+] Running 3/3server_default          Created
✓ Network web-server_default          Created
✓ Container web-server-database-1     Healthy
✓ Container web-server-sito-1         Started
```

Figura 14: Servizi funzionanti

A questo punto è possibile accedere al sito mediante l'indirizzo della macchina virtuale `web-server`:



The screenshot shows a web browser window with the address bar displaying `192.168.1.71` and a warning icon indicating it is not secure. The browser's tab bar shows several tabs: `Emulation - Classics`, `Heraldis - Graphics`, `Sviluppo`, `Shop`, and `Il Post`. The main content area of the browser displays the following text:

Tutti gli elementi:

Nessun elemento presente...

Aggiungi un elemento

Campi

Nome:

Cognome:

Figura 15: Sito web

Dal form si possono quindi aggiungere diversi elementi che verranno visualizzati nella lista, per testare la persistenza dei dati, si può forzare il build del container del database aggiungendo `-build` a `docker compose up`, e si noterà che gli elementi che erano stati inseriti in precedenza sono ancora visualizzabili nella lista perché MariaDB vede che ci sono degli elementi pre-esistenti nel volume

nfs, se invece si cancellano i file creati da maria db nel volume nfs da un'altra macchina virtuale (o il container stesso) al prossimo avvio MariaDB dovrà riefettuare l'inizializzazione, svuotando di conseguenza tutto il suo contenuto, al che alla riapertura del sito non si vedranno più elementi nella lista.

3 Conclusioni

3.1 Struttura del progetto e password

- **Ambiente di virtualizzazione:** ProxMox
 - **Macchina virtuale 1 (100):** Contiene container per il server NFS, il percorso dei file è /home/luca, la password per gli account (luca, root) è: nfs
 - **Macchina virtuale 2 (101):** Contiene i servizi web e database che usa volume NFS, il percorso dei file è "/root/servizi/*", la password per gli account (luca, root) è: web-server
 - * **web-server:** Container con immagine PHP-Apache creato mediante Dockerfile, il percorso interessato è sito/*
 - * **database:** Container con immagine MariaDB che monta un volume con driver NFS, il percorso interessato è sql/*

Entrambe le macchine hanno accesso ssh abilitato sull'account "luca" con password nfs e web-server

Glossario

GPG GNU Privacy Guard. 3

KVM Tecnologia di virtualizzazione open source integrata nel kernel Linux che gli consente di funzionare da hypervisor in grado di eseguire molteplici macchine virtuali diverse. 1

LXC Linux Containers. 1

NFS Network File System. 2

RPC Remote Procedure Call. 2

XDR eXternal Data Representation. 2

Riferimenti bibliografici

- [1] T. Haynes e D. Noveck, «Network File System (NFS) Version 4 Protocol,» Internet Engineering Task Force, Request for Comments RFC 7530, mar. 2015, Standards Track. indirizzo: <https://www.rfc-editor.org/rfc/rfc7530>.

- [2] B. Callaghan, «NFS Version 3 Protocol Specification,» Internet Engineering Task Force, Request for Comments RFC 1813, giu. 1995, Informational. indirizzo: <https://www.rfc-editor.org/rfc/rfc1813>.
- [3] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh e B. Lyon, «NFS: Network File System Protocol Specification,» Internet Engineering Task Force, Request for Comments RFC 1094, mar. 1989, Informational. indirizzo: <https://www.rfc-editor.org/rfc/rfc1094>.