

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

Luca Casadei
Matricola: 0001069237

14 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 container NFS-SERVER	4

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:

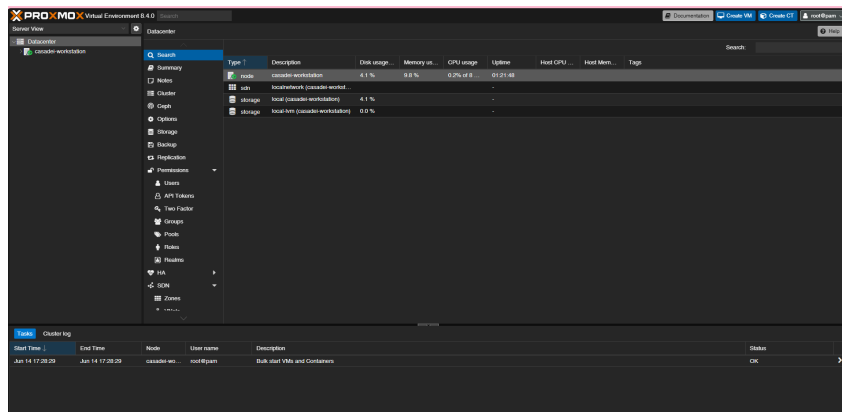


Figura 1: Proxmox installato su workstation

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:

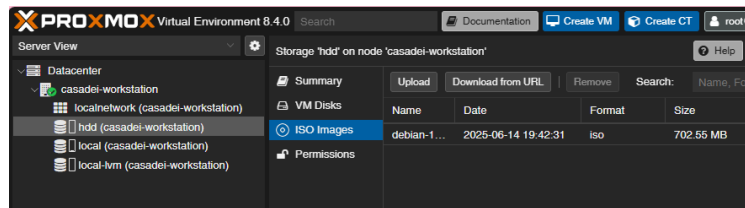


Figura 2: Aggiunta ISO Debian 12 a Proxmox

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-12.1.0-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)
        ↪ signed-by=/etc/apt/keyrings/docker.asc]
        ↪ https://download.docker.com/linux/debian \
11     $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
12     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
13     sudo apt-get update

```

E installiamo i pacchetti:

```

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

```

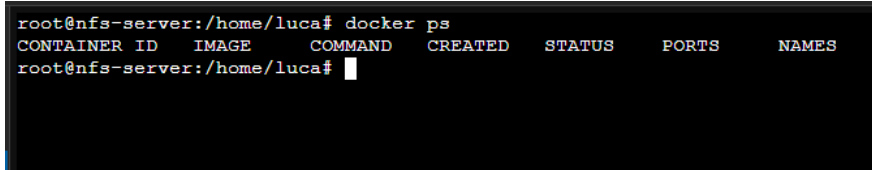
provando ad eseguire il comando:

```

1      docker ps

```

otteniamo:



```

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

```

Figura 4: Output di Docker PS dopo averlo installato

1.2.4 Creazione container NFS-SERVER

Abilitiamo i moduli NFS e NFSD dal kernel linux:

```

1      sudo modprobe {nfs,nfsd}

```

in alternativa si possono aggiungere questi due moduli nel file `/etc/modules`
 Come immagine per docker verrà utilizzata **nfs-server** e verrà configurata
 mediante il seguente script compose:

```

1      services:
2          nfs-server:
3              image: erichough/nfs-server
4              container_name: nfs-server
5              restart: unless-stopped
6              # Necessario perché il protocollo
7              # usa delle direttive a livello di kernel
8              privileged: true
9              ports:
10             - "2049:2049" # NFS (Vedi relazione)
11             # - "111:111" # RPC (Queste porte sarebbero
        ↪ necessarie per supportare V3, non ci interessa
        ↪ per questo progetto)

```

```

12 # - "32765:32765" # mountd
13 # - "32767:32767" # statd
14 environment:
15 # Questo è un argument del server nfs
16 # che indica la cartella dentro al container
17 # da esportare con il protocollo, potrebbero
18 # essercene diverse.
19 - NFS_EXPORT_0=/exports
20 volumes:
21 # Espone la cartella interna esportata prima
22 # fuori dal container con una mappatura bind
23 - /srv/nfs:/exports

```

Dopo aver creato lo script ed eseguito "docker compose up" per farlo partire, nfs-server dovrebbe darci il seguente output:

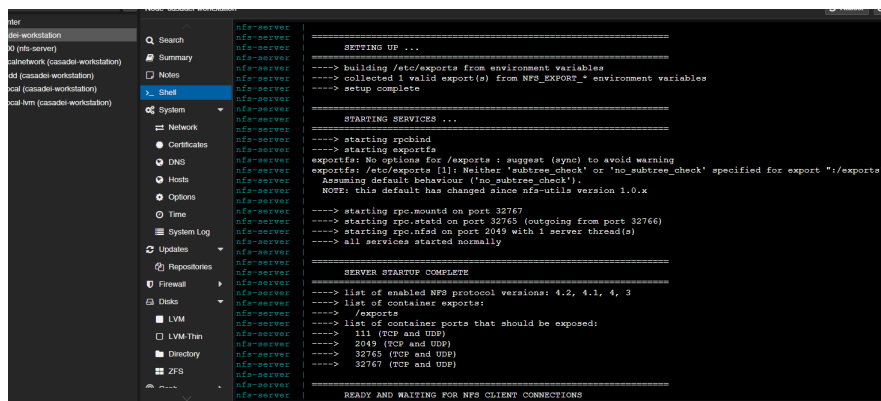


Figura 5: Dopo docker compose up

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>.