Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Dalla Virtualizzazione alla Containerizzazione: Implementazione di LXD e Apache2

Emanuele Damiano, Pier Giuseppe Meo, Giovanni Massafra, Mario Sicuranza

Rapporto Tecnico N: RT-FOSSR-2025-04 Data: Aprile 2025

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
Sede di Napoli, Via P. Castellino 111, I-80131 Napoli,
Tel: +39-0816139508, Fax: +39-0816139531,
e-mail: napoli@icar.cnr.it, URL: www.na.icar.cnr.it

Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Dalla Virtualizzazione alla Containerizzazione: Implementazione di LXD e Apache2

Emanuele Damiano, Pier Giuseppe Meo, Giovanni Massafra, Mario Sicuranza

Istituto di Calcolo e Reti ad Alte Prestazioni del Consiglio Nazionale delle Ricerche
Via Pietro Castellino, 111 – 80131 Napoli, Italia

E-mail: {emanuele.damiano, piergiuseppe.meo, giovanni.massafra, mario.sicuranza}@icar.cnr.it

Abstract

Questo rapporto tecnico illustra dettagliatamente il processo di implementazione di un ambiente di virtualizzazione efficiente, scalabile e sicuro per il progetto FOSSR. Viene descritta l'installazione e la configurazione di LXD per la gestione di container applicativi e di Apache2 come reverse proxy su un server dedicato. La guida operativa copre l'installazione di LXD tramite Snap, la configurazione iniziale dei container e della rete con Netplan e bridge br0, l'installazione e configurazione di Apache2, l'impostazione del reverse proxy per indirizzare il traffico ai container LXD e la configurazione di LXD-UI con certificati SSL/TLS per una gestione web sicura. L'approccio descritto mira a ottimizzare l'utilizzo delle risorse hardware, garantire un elevato isolamento applicativo e fornire una soluzione flessibile per la gestione della rete containerizzata e la distribuzione del traffico web.

Keywords: *LXD, Containerizzazione, Apache2, Reverse Proxy, Virtual Host, Netplan*

Sommario

1. Introduzione

Il presente rapporto tecnico documenta l'implementazione di un prototipo di infrastruttura di rete ad alte prestazioni, sicura e ridondante, sviluppata nell'ambito del progetto *Fostering Open Science in Social Science Research* (FOSSR). Il documento fornisce una descrizione esaustiva delle procedure di installazione e configurazione di un ambiente di virtualizzazione avanzato basato sulla piattaforma LXD (Linux Container Daemon), finalizzato alla gestione efficiente e scalabile di container applicativi. In parallelo, viene illustrata la procedura per l'installazione e la configurazione di Apache2, un server web open-source modulare e versatile, impiegato in qualità di reverse proxy per ottimizzare la distribuzione del traffico web, centralizzare la gestione della sicurezza e garantire un punto di accesso unificato alle applicazioni containerizzate.

Il progetto mira a realizzare un'infrastruttura di rete performante, resiliente e sicura, capace di supportare in modo affidabile le attività di elaborazione, analisi e condivisione dei dati, assicurando al contempo la scalabilità orizzontale e la continuità operativa dei servizi. In questo contesto, l'adozione della containerizzazione mediante LXD consente di isolare rigorosamente applicazioni e servizi in ambienti autonomi, ottimizzando l'utilizzo delle risorse hardware, semplificando il deployment e il ciclo di vita delle applicazioni, e aumentando la portabilità e la scalabilità complessiva delle soluzioni implementative. Parallelamente, Apache2, configurato come reverse proxy, risponde all'esigenza di ottimizzare il flusso del traffico verso i container backend e di implementare meccanismi di sicurezza centralizzati, offrendo al contempo un'interfaccia di accesso performante e sicura per gli utenti finali.

Il rapporto tecnico si propone di fungere da guida operativa e metodologica, fornendo indicazioni dettagliate circa la configurazione, nonchè esempi pratici e soluzioni implementative per replicare l'ambiente di virtualizzazione e reverse proxy descritto.

La scelta delle tecnologie chiave – LXD e Apache2 – è stata ponderata e motivata da criteri di efficienza, flessibilità e sicurezza, e si articola nei seguenti aspetti:

- **LXD – Piattaforma di Container Management Avanzata:** LXD rappresenta la soluzione di riferimento per la gestione dei container Linux, garantendo un elevato grado di isolamento applicativo e un'ottimizzazione nell'utilizzo delle risorse hardware. Grazie alla sua flessibilità, permette una configurazione granulare della rete containerizzata e dello storage, offrendo un'interfaccia intuitiva e funzionalità di gestione avanzate, ideali per ambienti di produzione complessi.
- **Apache2 – Server Web Modulare e Reverse Proxy:** Apache2 si distingue per la sua architettura modulare, la stabilità comprovata e l'affidabilità intrinseca. In qualità di reverse proxy, semplifica l'accesso alle applicazioni containerizzate, centralizza la gestione della sicurezza e abilita funzionalità avanzate, quali il supporto al traffico HTTPS tramite SSL/TLS e il bilanciamento del carico tra più server backend, ottimizzando le performance complessive dell'infrastruttura.

Gli obiettivi principali del presente rapporto tecnico sono:

- **Descrivere dettagliatamente il processo di installazione e configurazione di LXD:** Illustrare le procedure per la creazione, gestione e monitoraggio dei container applicativi, la configurazione avanzata della rete containerizzata e la gestione delle soluzioni di storage, includendo le best practices per la sicurezza.
- **Fornire una guida operativa per la configurazione di Apache2 come reverse proxy:** Presentare esempi concreti di configurazione dei Virtual Host, evidenziando le direttive fondamentali per il proxying e la gestione del traffico HTTP/HTTPS, e illustrando l'eventuale implementazione del bilanciamento del carico.

Il presente documento tecnico si articola in diverse sezioni, ciascuna delle quali analizza in maniera approfondita e sistematica un aspetto specifico dell'implementazione dell'infrastruttura tecnologica. In particolare:

- **Capitolo 2 – Fondamenti Tecnologici:** Questo capitolo introduce le tecnologie alla base dell'infrastruttura, illustrando dettagliatamente i principi operativi, i vantaggi intrinseci e le motivazioni che hanno portato alla scelta di LXD (Linux Container Daemon) per la containerizzazione e di Apache2 come reverse proxy. Vengono inoltre esaminate le soluzioni complementari di networking e sicurezza, elementi fondamentali per la realizzazione di un sistema robusto ed efficiente.
- **Capitolo 3 – LXD:** In questo capitolo vengono descritti in maniera esaustiva i passaggi operativi necessari per l'installazione e la configurazione della piattaforma LXD. Si approfondiscono aspetti quali la creazione e gestione dei container applicativi, la configurazione avanzata della rete containerizzata e delle soluzioni di storage, oltre alle best practices per garantire un elevato livello di sicurezza nell'ambiente LXD.
- **Capitolo 4 – Apache2: Web Server e Reverse Proxy:** Questo capitolo offre una guida dettagliata per configurare Apache2 in qualità di reverse proxy. Vengono spiegate le impostazioni necessarie per la gestione del traffico HTTP e HTTPS, l'abilitazione del protocollo SSL/TLS per comunicazioni crittografate, la configurazione di Virtual Host dedicati e l'eventuale implementazione del bilanciamento del carico tra i container backend.
- **Capitolo 5 – Conclusioni:** In questa sezione vengono sintetizzati i risultati ottenuti, evidenziando come l'infrastruttura sia stata progettata per garantire scalabilità, resilienza e sicurezza. Il documento, fondato su standard consolidati e best practices, rappresenta un valido strumento di riferimento operativo e una guida metodologica per l'implementazione di ambienti di virtualizzazione e reverse proxy, applicabili a contesti di ricerca, sviluppo software e infrastrutture IT complesse.

L'intera configurazione infrastrutturale è stata progettata e realizzata con l'obiettivo primario di assicurare scalabilità orizzontale, elevata resilienza ai guasti e robustezza in termini di sicurezza. Attraverso l'adozione di soluzioni tecnologiche modulari, approcci configurativi standardizzati e best practices di sicurezza consolidate, questo rapporto tecnico si propone come una guida di riferimento per la realizzazione di ambienti similari in ambito IT.

2. Fondamenti Tecnologici

Il seguente capitolo descrive l'architettura di un'infrastruttura IT avanzata, ingegnerizzata per ottimizzare efficienza, sicurezza e scalabilità. Al centro di tale architettura si pone l'integrazione sinergica di tecnologie di containerizzazione e reverse proxy, implementate su un'infrastruttura di rete robusta e protette da rigorose misure di sicurezza. L'intento di questo capitolo è definire le fondamenta tecnologiche che abilitano un ambiente operativo dinamico, resiliente e intrinsecamente sicuro.

2.1. Architettura

L'architettura del sistema proposto è strutturata su tre pilastri fondamentali: i) la containerizzazione per l'isolamento e la gestione efficiente delle applicazioni; ii) il reverse proxy per l'accesso unificato e sicuro ai servizi; iii) tecnologie complementari per networking e sicurezza che rafforzano l'intera infrastruttura.

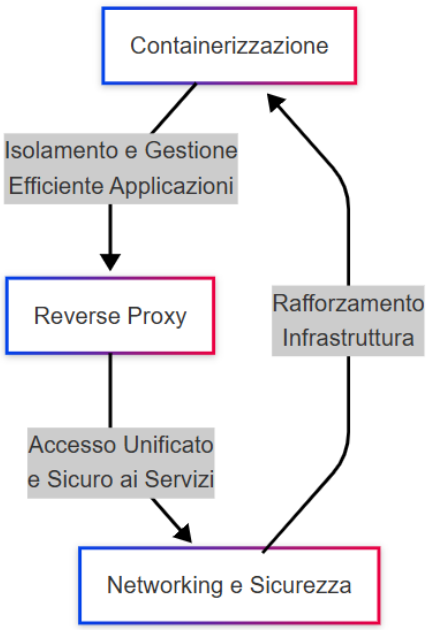

Figura 1 - Architettura del sistema implementato

L'analisi strutturale del diagramma rivela una gerarchia e interdipendenza funzionale tra i tre elementi architetturali cardine del sistema. In primis, la containerizzazione, rappresentata dal blocco apicale, si configura come lo strato fondamentale, il cui obiettivo primario è l'isolamento e la gestione efficiente delle applicazioni, capitalizzando sui vantaggi di portabilità, densità e ottimizzazione delle risorse intrinseci a tale tecnologia. In secondo luogo, il reverse proxy, posizionato centralmente nel diagramma, si pone come intermediario strategico e punto di accesso unificato ai servizi, assolvendo a funzioni critiche quali la terminazione SSL/TLS e il filtraggio applicativo, con la potenziale implementazione di meccanismi di bilanciamento del carico. Infine, le tecnologie di networking e sicurezza, rappresentate dal blocco inferiore, costituiscono un elemento di rinforzo trasversale all'intera infrastruttura. Questi componenti non configurano entità isolate, bensì un insieme integrato di soluzioni che permeano tutti gli strati del sistema, garantendo robustezza, resilienza e un adeguato livello di protezione, in un ciclo di retroazione positiva che influenza e supporta, a sua volta, l'efficacia della containerizzazione e viceversa.

2.1.1. Containerizzazione e LXD

La containerizzazione emerge come paradigma fondamentale per l'isolamento e la gestione delle applicazioni in unità autonome, i *container*. Condividendo il kernel del sistema operativo ospitante, i container offrono un ambiente di esecuzione leggero e ad avvio rapido, apportando vantaggi significativi:

- **Efficienza Operativa e Ottimizzazione delle Risorse:** I container si avviano in pochi secondi e richiedono risorse computazionali limitate rispetto alle macchine virtuali tradizionali, consentendo un utilizzo più efficiente dell'hardware sottostante e una maggiore densità di applicazioni per server fisico.
- **Portabilità e Coerenza Ambientale:** Ogni container incapsula l'applicazione e tutte le sue dipendenze in un'unità autonoma. Questa caratteristica garantisce una coerenza applicativa tra ambienti di sviluppo, test e produzione, semplificando il ciclo di vita del software e riducendo i problemi legati alle differenze ambientali.
- **Scalabilità Dinamica e Reattività:** La natura modulare e leggera dei container abilita una scalabilità orizzontale dinamica. È possibile scalare le applicazioni in risposta alle variazioni del carico di lavoro creando, replicando e distruggendo container in modo rapido e automatizzato, assicurando elevata reattività e resilienza.

In questo contesto, **LXD (Linux Container Daemon)** è stato selezionato come piattaforma di container management, in virtù delle sue caratteristiche distintive:

- **Installazione e Configurazione Semplificate:** LXD si distingue per la sua facilità di installazione e configurazione, offrendo pacchetti precompilati per le principali distribuzioni Linux. Una documentazione esaustiva, un'interfaccia a riga di comando (CLI) intuitiva e un'API RESTful completa facilitano l'amministrazione e l'automazione della piattaforma.
- **Networking e Storage Avanzati:** LXD integra un supporto avanzato per la configurazione di reti complesse e soluzioni di storage performanti. La capacità di gestire reti bridge, VLAN, overlay network e di integrarsi con soluzioni di storage come ZFS e LVM, permette una facile integrazione di LXD in infrastrutture IT esistenti e la creazione di ambienti containerizzati sofisticati.
- **Scalabilità e Performance Elevate:** Grazie al suo design modulare e ottimizzato, LXD è in grado di gestire migliaia di container su un singolo host con elevata efficienza. Questa caratteristica lo rende particolarmente adatto per scenari applicativi che richiedono alta densità di container e performance elevate.

All'interno dell'infrastruttura ivi descritta, LXD è impiegato strategicamente per:

- **Isolamento Applicativo e Servizi:** Garantire l'isolamento di applicazioni e servizi differenti all'interno di container separati, migliorando la sicurezza e la stabilità dell'infrastruttura complessiva.
- **Ambienti di Test On-Demand:** Creare ambienti di test isolati e riproducibili in modo rapido e automatizzato, accelerando i cicli di sviluppo e test del software.
- **Integrazione con la Rete Fisica Esistente:** Integrare i container in modo trasparente con la rete fisica aziendale, facilitando l'accesso ai servizi containerizzati e la comunicazione con altre componenti infrastrutturali.

2.1.2. Apache2 come Reverse Proxy

Apache HTTP Server (Apache2), rinomato server web open source, è stato scelto per implementare la funzione di reverse proxy. La selezione di Apache2 è motivata dalla sua comprovata stabilità, modularità e dalle robuste funzionalità che offre in questo ruolo specifico:

- **Unificazione dell'Accesso e Semplificazione dell'Infrastruttura:** Apache2 funge da punto di ingresso unico per tutte le richieste web provenienti dall'esterno. Agendo come reverse proxy, riceve le richieste e le inoltra in modo trasparente ai container backend che ospitano le applicazioni, semplificando notevolmente l'esposizione dei servizi e l'instradamento del traffico.

- **Bilanciamento del Carico e Alta Disponibilità:** Apache2 è in grado di distribuire in modo intelligente il traffico web in ingresso tra un pool di server backend, implementando strategie di bilanciamento del carico (load balancing). Questa funzionalità incrementa significativamente la resilienza dell'infrastruttura, garantendo la disponibilità dei servizi anche in caso di guasti parziali o picchi di traffico.
- **Sicurezza Centralizzata e Rafforzata:** Apache2 centralizza la gestione dei certificati SSL/TLS, semplificando la configurazione HTTPS e garantendo comunicazioni cifrate per tutti i servizi esposti tramite il reverse proxy. Inoltre, l'ampia disponibilità di moduli di sicurezza per Apache2 (come mod_security e mod_evasive) permette di implementare contromisure efficaci contro un'ampia gamma di attacchi web comuni, rafforzando la postura di sicurezza dell'intera infrastruttura.

La configurazione di Apache2, basata su una struttura modulare, consente di attivare selettivamente solo le funzionalità strettamente necessarie e di integrare agevolmente nuove estensioni e misure di sicurezza in risposta all'evoluzione del panorama delle minacce informatiche. Questa flessibilità architetturale rende Apache2 una scelta solida e adattabile per il ruolo di reverse proxy in ambienti moderni e dinamici.

2.1.3. Tecnologie Complementari per Networking e Sicurezza

Per garantire una connettività efficiente e una protezione robusta dei dati, l'infrastruttura integra un insieme di tecnologie complementari che operano a livello di networking e sicurezza:

- **Networking Avanzato**
  - **Netplan: Configurazione Dichiarativa della Rete:** Netplan è impiegato per la configurazione dichiarativa delle interfacce di rete a livello di sistema operativo. Attraverso file di configurazione YAML, Netplan semplifica la gestione di indirizzi IP, gateway predefiniti, server DNS e l'implementazione di reti bridge. Le reti bridge sono essenziali per connettere i container LXD alla rete fisica esistente, consentendo loro di comunicare con altre risorse di rete e con l'esterno.
  - **LACP (Link Aggregation Control Protocol): Aggregazione e Ridondanza della Banda:** LACP è implementato per aggregare interfacce di rete fisiche multiple in un'unica interfaccia logica (*bond* o *team*). Questa tecnica incrementa la banda di rete disponibile per il server e, contemporaneamente, fornisce ridondanza in caso di guasti hardware a livello di singola interfaccia fisica. LACP assicura continuità operativa e prestazioni di rete elevate.
- **Sicurezza Multilivello**
  - **Autenticazione SSH Basata su Chiavi Crittografiche: Sicurezza degli Accessi Amministrativi:** L'accesso amministrativo al sistema è rigorosamente limitato all'autenticazione SSH basata su chiavi crittografiche. L'eliminazione dell'autenticazione tramite password tradizionale riduce drasticamente i rischi associati ad attacchi di *password cracking* e *credential stuffing*, elevando significativamente la sicurezza degli accessi amministrativi.
  - **Isolamento di Rete per Container tramite VLAN:** Per garantire un isolamento efficace del traffico di rete tra i container LXD e tra i container e l'infrastruttura host, vengono implementate VLAN (Virtual LAN). Configurazioni di rete specifiche per ciascun container, eventualmente associate a VLAN dedicate, assicurano una segmentazione logica della rete, limitando la possibilità di propagazione di attacchi da container compromessi e rafforzando la sicurezza complessiva dell'ambiente containerizzato.

2.2. Vulnerabilità e Mitigazioni

Nonostante la robustezza e l'efficacia intrinseche delle tecnologie selezionate per l'infrastruttura FOSSR, si riconosce la necessità di un approccio proattivo alla gestione della sicurezza, imperniato sull'identificazione e la mitigazione delle vulnerabilità potenziali.

In relazione a LXD, si evidenzia primariamente il rischio di *container escape*, ove un container compromesso potrebbe teoricamente sfruttare vulnerabilità a livello del kernel Linux per ottenere accesso non autorizzato al

sistema host o ad altri container. Le strategie di mitigazione per tale rischio includono l'implementazione di strumenti di *hardening* a livello di kernel e container, come AppArmor o SELinux, l'adozione di un regime di aggiornamenti di sicurezza costanti e tempestivi, e l'applicazione rigorosa del principio del minimo privilegio nella configurazione dei container, unitamente a politiche di *resource limiting*.

Parallelamente, la dipendenza di LXD dal kernel Linux sottostante rende imperativo un monitoraggio proattivo degli aggiornamenti di sicurezza del kernel e l'esecuzione di audit di sicurezza periodici e *penetration test* volti a verificare la conformità delle configurazioni e a identificare eventuali aree di miglioramento.

In merito ad Apache2, configurato come *reverse proxy*, si individuano vulnerabilità connesse a potenziali errori di configurazione, che potrebbero inavvertitamente condurre ad accessi non autorizzati o a *information disclosure*. Le contromisure raccomandate in tal senso prevedono l'esecuzione di *audit* regolari e sistematici della configurazione, la verifica periodica dell'implementazione e dell'aggiornamento dei moduli di sicurezza essenziali, quali `mod_ssl`, `mod_security` e `mod_evasive`.

Inoltre, si pone l'accento sul rischio di attacchi Denial of Service (DoS) e Distributed Denial of Service (DDoS), per i quali si suggerisce l'integrazione di moduli di protezione dedicati, quali `mod_evasive` e `mod_security`, e l'implementazione di soluzioni avanzate di bilanciamento del carico.

Infine, in relazione alla gestione dei certificati SSL/TLS, una configurazione inadeguata o obsoleta può esporre l'infrastruttura a rischi di intercettazione e compromissione dei dati. Le mitigazioni in questo ambito comprendono la gestione centralizzata e l'automazione dei processi di rinnovo e rotazione dei certificati, nonchè l'adozione esclusiva di protocolli e algoritmi crittografici riconosciuti come sicuri e aggiornati, in linea con le *best practices* di settore e le raccomandazioni delle autorità competenti.

3. LXD

Questo capitolo presenta la piattaforma LXD come soluzione avanzata per la containerizzazione, illustrando il flusso operativo dalla sua installazione alla configurazione per la gestione di ambienti applicativi isolati e performanti. L'obiettivo è offrire una visione d'insieme delle scelte progettuali e delle best practices adottate, lasciando i dettagli tecnici più specifici alle appendici.

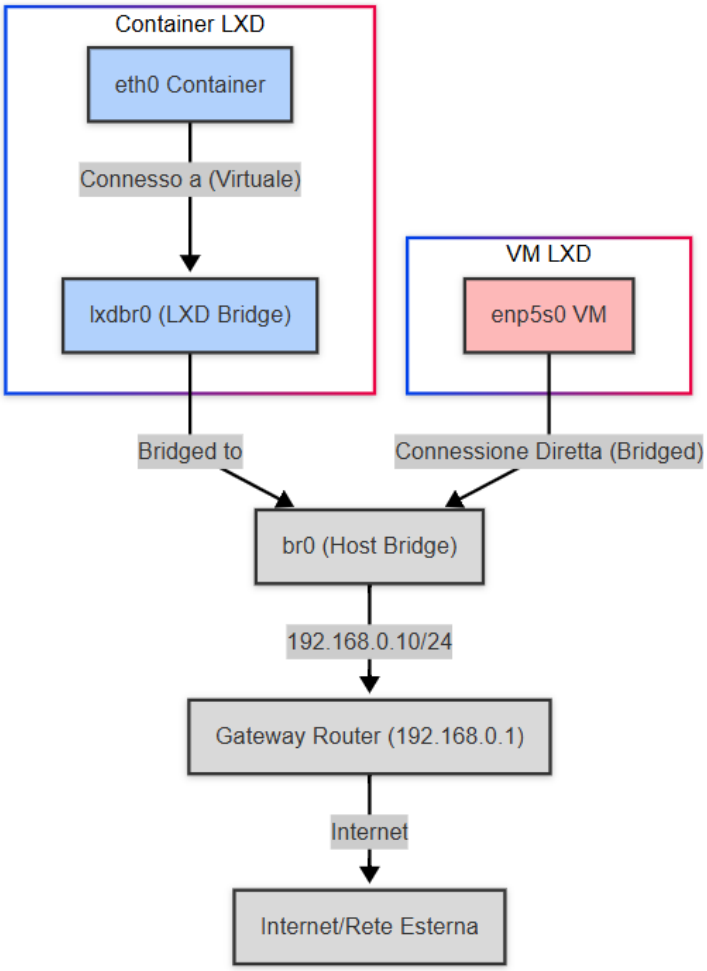

Figura 2 – Rete LXD e VM

Nella Figura 2 è presente il Diagramma di rete che illustra i percorsi di traffico per i container LXD (linea con label "bridged to") e una Virtual Machine (VM) (linea con label "Connessione diretta (Bridged)") nella configurazione descritta. Entrambi i percorsi convergono sul bridge host br0 per la connettività alla rete locale e a Internet.

L'architettura di rete illustrata rappresenta un ambiente di virtualizzazione basato su LXD, in cui container e macchine virtuali (VM) comunicano attraverso una rete bridge configurata con Netplan.

Nella parte superiore dello schema, si trova un *Container LXD*, dotato di un'interfaccia di rete virtuale denominata *eth0 Container*. Questo container è connesso alla *Rete Bridge LXD* (*lxdbr0*), un bridge virtuale che funge da switch software per i container, consentendo la loro interconnessione e l'accesso alla rete esterna. La connessione tra *eth0 Container* e *lxdbr0* è rappresentata da una linea etichettata "*Connesso a (Virtuale)"*, mentre il collegamento tra *lxdbr0* e il bridge dell'host (*br0*) è indicato come *Bridged to*, segnalando che il traffico dei container transita attraverso questa rete di transito prima di raggiungere l'infrastruttura principale.

Il *Bridge Host* (*br0*), configurato tramite Netplan, rappresenta il punto centrale della rete, gestendo il traffico sia dei container LXD che delle macchine virtuali. Esso possiede un indirizzo IP *192.168.0.10/24* e appartiene alla rete locale *192.168.0.0/24*. Accanto ad esso, è presente una *Virtual Machine (VM)*, la quale dispone di una propria interfaccia di rete virtuale denominata *enp5s0 VM*. A differenza dei container, la VM non utilizza la rete di transito *lxdbr0*, ma è direttamente connessa a *br0*, come indicato dalla linea con etichetta "*Connessione Diretta (Bridged)"*. La configurazione di rete della VM prevede un indirizzo IP statico *192.168.0.151/24*, un gateway predefinito *192.168.0.1* e server DNS *8.8.8.8* e *8.8.4.4*.

Dal punto di vista della connettività esterna, il *Gateway Router* (*192.168.0.1*) rappresenta il punto di uscita della rete locale verso Internet. Tutto il traffico generato dai container e dalle VM transita attraverso *br0* per raggiungere il router, il quale si occupa di instradare i pacchetti verso la rete esterna.

In sintesi, l'architettura presentata distingue due percorsi di rete principali:

- **Percorso dei container LXD:** il traffico passa attraverso *eth0 Container* → *lxdbr0* → *br0* → *Gateway Router* → *Internet*.
- **Percorso della VM:** il traffico segue il percorso diretto *enp5s0 VM* → *br0* → *Gateway Router* → *Internet*.

La differenza chiave risiede nel fatto che i container utilizzano una rete di transito (*lxdbr0*), mentre le VM si connettono direttamente al bridge centrale (*br0*). Tuttavia, entrambi i percorsi convergono su *br0*, che funge da snodo principale prima di raggiungere la rete locale e, successivamente, Internet.

3.1 Installazione di LXD

La procedura di installazione di LXD si articola nei seguenti passaggi chiave:

- **Preparazione dell'Ambiente:** Aggiornamento del sistema operativo e installazione dei pacchetti necessari (ad es. snapd).
- **Installazione tramite Snap:** Utilizzo del comando `sudo snap install lxd` per scaricare e installare l'ultima versione stabile.
- **Verifica dell'Installazione:** Esecuzione di `lxd --version` per confermare il corretto funzionamento.

L'installazione di LXD è stata eseguita seguendo una procedura standardizzata, articolata in fasi propedeutiche e nell'installazione vera e propria del software. Tutti i dettagli operativi e i relativi comandi sono descritti in modo approfondito nell'Appendice A.

3.2. Configurazione Iniziale di LXD

Successivamente all'installazione, è stata eseguita la configurazione iniziale di LXD, definendo i parametri fondamentali per il funzionamento dell'ambiente di containerizzazione.

3.2.1 Inizializzazione di LXD

La configurazione iniziale di LXD è stata avviata tramite il comando interattivo:

```
sudo lxd init
```

Il comando `sudo lxd init` avvia una procedura guidata interattiva che richiede all'utente di specificare diversi parametri di configurazione, tra cui:

- **Backend di storage:** Il prompt di configurazione richiede la selezione del backend di storage per LXD. Le opzioni principali sono `dir` e `zfs`. L'opzione `dir` configura LXD per utilizzare una directory all'interno del filesystem host come storage pool, rappresentando la configurazione più semplice e

adatta ad ambienti di test o con basse esigenze prestazionali. L'opzione `zfs`, invece, configura LXD per utilizzare ZFS (Zettabyte File System) come storage pool, offrendo funzionalità avanzate quali performance elevate, integrità dei dati, snapshot efficienti e funzionalità avanzate come la compressione e la deduplica, rendendolo più appropriato per ambienti di produzione esigenti. La scelta del backend di storage dipende dai requisiti specifici in termini di prestazioni, affidabilità e funzionalità desiderate per l'ambiente LXD.

- **Configurazione della rete:** Il prompt di configurazione richiede di specificare la configurazione di rete per LXD. In particolare, è possibile configurare LXD per utilizzare un bridge di rete preesistente, integrandosi con l'infrastruttura di rete esistente. La configurazione del bridge di rete è fondamentale per consentire ai container LXD di comunicare con la rete fisica e con l'esterno.
- **Abilitazione dell'accesso remoto:** Il prompt di configurazione offre la possibilità di abilitare la gestione remota di LXD, consentendo l'amministrazione dell'ambiente LXD da postazioni remote. L'abilitazione dell'accesso remoto può essere utile in scenari di gestione centralizzata o di amministrazione da remoto dell'infrastruttura HPC.

La procedura guidata `lxd init` consente di configurare in modo interattivo i parametri fondamentali di LXD, adattando l'ambiente di containerizzazione alle specifiche esigenze dell'infrastruttura sottostante.

Per una comprensione completa della configurazione di rete adottata nel prototipo HPC, si rimanda alle Appendici B e C. In Appendice B viene illustrata la configurazione iniziale tramite Netplan, mentre in Appendice C sono dettagliate le modifiche apportate per integrare il bridge di rete destinato all'ambiente LXD. Queste sezioni forniscono il file YAML completo, l'analisi delle scelte implementative e le procedure di applicazione e verifica, elementi fondamentali per replicare o modificare la topologia di rete in ambienti simili.

3.3. LXD-UI: Interfaccia di Gestione Web Sicura

Questa sezione fornisce una panoramica sull'implementazione e configurazione di LXD-UI, la console web che permette agli amministratori di gestire in modo intuitivo e sicuro l'ambiente containerizzato basato su LXD. LXD-UI viene installato tramite Snap, garantendo aggiornamenti automatici e isolamento applicativo, e configurato per operare in modalità HTTPS su una porta dedicata, con una politica di Cross-Origin che facilita la gestione delle richieste web.

Per assicurare la sicurezza delle comunicazioni, è stata adottata una soluzione basata su certificati SSL/TLS. Il processo include la creazione di un'autorità di certificazione (CA) autofirmata, la generazione e la firma del certificato server, e la conversione dei certificati nel formato PFX per l'importazione nel browser, garantendo così una connessione crittografata e affidabile.

Tutti i dettagli operativi, inclusi i comandi specifici e le procedure passo-passo per l'installazione, la configurazione iniziale e la gestione dei certificati, sono riportati nell'Appendice D.

3.4. Architettura e Configurazione della Rete in LXD

L'architettura di rete in un ambiente LXD si basa su bridge virtuali, che consentono una gestione centralizzata della connettività per container e macchine virtuali (VM). Questo modello garantisce la comunicazione tra le istanze virtualizzate e l'integrazione con la rete fisica sottostante, facilitando l'accesso a risorse esterne.

3.4.1 Creazione della Rete in LXD

Per integrare l'ambiente LXD con la rete fisica preesistente, è stata creata una nuova rete LXD di tipo bridge denominata `lxdbr0`. La creazione della rete è stata eseguita tramite il seguente comando:

```
lxc network create lxdbr0 bridge.external_interfaces=br0 \
          ipv4.address=192.168.71.1/24 ipv4.nat=false \
          ipv6.address=none
```

Di seguito si analizzano i parametri adottati per la configurazione della rete **lxdbr0**:

- **lxc network create lxdbr0**: Crea una nuova rete LXD denominata `lxdbr0`. Il nome è arbitrario e può essere personalizzato.
- **bridge.external_interfaces=br0** : Collega `lxdbr0` al bridge preesistente `br0`, configurato tramite Netplan, garantendo l'integrazione con la rete fisica.
- **ipv4.address=192.168.71.1/24**: Assegna alla rete `lxdbr0` l'indirizzo IP `192.168.71.1/24`, che funge da gateway per i container connessi.
- **ipv4.nat=false**: Disabilita il Network Address Translation (NAT), permettendo ai container di ottenere indirizzi IP direttamente accessibili dalla rete esterna.
- **ipv6.address=none**: Disabilita il supporto per IPv6, semplificando la configurazione in contesti in cui non è richiesto.

Questa configurazione permette di creare una rete bridge `lxdbr0` che funge da intermediario tra i container e la rete fisica, garantendo flessibilità e scalabilità nella gestione della connettività.

3.4.2 Associazione dei Container al Bridge di Rete LXD lxdbr0

Dopo aver creato e configurato la rete `lxdbr0`, è possibile collegare i container a questa rete al momento della loro creazione. Per farlo, si utilizza l'opzione `-n` (o `--network`) del comando `lxc launch`. Ad esempio, per avviare un container denominato `mycontainer` basato sull'immagine Ubuntu 20.04 e associarlo alla rete `lxdbr0`, si esegue il seguente comando:

```
lxc launch ubuntu:20.04 mycontainer -n lxdbr0
```

Di seguito, si analizzano i parametri utilizzati nel comando:

- **`lxc launch ubuntu:20.04 mycontainer`**: Crea e avvia un nuovo container basato sull'immagine Ubuntu 20.04. Il nome del container, in questo caso `mycontainer`, è arbitrario e può essere personalizzato a discrezione dell'amministratore. L'immagine Ubuntu 20.04 viene scaricata dal repository predefinito di LXD (`ubuntu`).
- **`-n lxdbr0`**: Specifica la rete LXD a cui il container deve essere connesso. L'uso di `lxdbr0` garantisce che il container ottenga un indirizzo IP dalla sottorete `192.168.71.0/24` e possa comunicare con altri container sulla stessa rete, nonché con la rete fisica tramite `br0`.

L'uso di lxdbr0 garantisce che il container ottenga un indirizzo IP dalla sottorete `192.168.71.0/24`, con `192.168.71.1` configurato come gateway fornito dall'interfaccia lxdbr0 stessa. Questa configurazione consente di integrare in modo efficace i container con l'infrastruttura di rete preesistente, garantendo la connettività necessaria per le applicazioni ospitate all'interno dei container.

3.5. Migrazione e Configurazione di Macchine Virtuali in Ambiente LXD

Il presente paragrafo si focalizza sulla migrazione e configurazione di VM all'interno dell'ambiente LXD, illustrando il processo tecnico adottato per trasferire una VM da un server di origine alla nuova infrastruttura LXD. La procedura dettagliata nei paragrafi seguenti assicura una transizione fluida e la piena integrazione della VM nell'ambiente di destinazione, preservandone la funzionalità e la connettività di rete.

3.5.1. Configurazione dell'Ambiente di Destinazione

Elementi fondamentali per la corretta configurazione dell'ambiente LXD sono i profili di rete e gli storage pool, che definiscono rispettivamente le impostazioni di networking e le modalità di allocazione dello storage per le VM. I profili di rete in LXD rappresentano modelli predefiniti per la configurazione delle interfacce di rete virtuali delle VM, consentendo di standardizzare e semplificare l'assegnazione di impostazioni di rete coerenti. Parallelamente, gli storage pool definiscono le modalità con cui LXD gestisce lo spazio di storage

per le VM, offrendo diverse opzioni in termini di prestazioni, funzionalità e gestione dello spazio disco. I paragrafi successivi, 3.5.1.1. Profilo di Rete "PublicIP" e 3.5.1.2. Storage Pool LVM, descriveranno nel dettaglio la configurazione specifica adottata per il profilo di rete e lo storage pool nell'ambiente di destinazione, evidenziandone le motivazioni tecniche e le scelte implementative.

3.5.1.1. Profilo di Rete "PublicIP"

Per garantire connettività di rete diretta e prestazioni ottimali per le Virtual Machine (VM) FOSSR, è stato definito il profilo di rete denominato "PublicIP". Tale profilo è stato progettato per implementare un modello di networking di tipo "bridged", le cui specifiche tecniche sono delineate nella seguente configurazione YAML:

```
name: PublicIP
description: Profilo per le VM FOSSR
devices:
  eno8403:
    nictype: bridged
    parent: br0
    type: nic
  root:
    path: /
    pool: lxdpool
    type: disk
config: {}
```

Il dispositivo di rete `eno8403`, configurato con `'nictype: bridged'` e `'parent: br0'`, definisce l'interfaccia di rete virtuale della VM. La modalità "bridged" consente alla VM di connettersi direttamente alla rete fisica dell'host attraverso un bridge di livello 2, in questo caso `br0`. In questo modo, la VM acquisisce un indirizzo IP direttamente dal network fisico, risultando pienamente integrata nella rete locale e accessibile dall'esterno senza necessità di Network Address Translation (NAT). Questa configurazione semplifica notevolmente la migrazione delle VM, in quanto l'indirizzamento IP e la connettività di rete vengono preservati inalterati tra host diversi, eliminando la necessità di complesse riconfigurazioni del networking a seguito di una migrazione.

Parallelamente alla configurazione di rete, il profilo "PublicIP" definisce anche il dispositivo disco `root` per le VM. Il parametro `pool: lxdpool` specifica che il disco root delle VM associate a questo profilo sarà allocato all'interno dello storage pool LVM `lxdpool`, precedentemente configurato e descritto nel paragrafo 3.5.1.2. Questa impostazione garantisce la coerenza con l'infrastruttura di storage definita e sfrutta i vantaggi del thin provisioning e della gestione efficiente dello spazio offerti da LVM per i dischi delle VM.

In sintesi, il profilo di rete "PublicIP" fornisce alle VM FOSSR una connettività di rete di tipo "bridged" che le integra pienamente nella rete fisica, semplificando la gestione del networking e la migrazione, unitamente all'allocazione del disco root all'interno dello storage pool LVM `lxdpool` per una gestione efficiente dello spazio disco.

3.5.1.2. Storage Pool LVM

LXD offre una vasta gamma di opzioni per la gestione dello storage, supportando driver nativi come **dir**, **lvm**, **zfs** e **btrfs**, ognuno con caratteristiche specifiche. Il driver **dir**, che utilizza semplici directory nel filesystem dell'host, è l'opzione più immediata e adatta per ambienti di test e sviluppo.

Per questa configurazione, è stato scelto LVM (Logical Volume Manager) per via dei suoi numerosi vantaggi: un efficiente meccanismo di *thin provisioning*, grande flessibilità nella gestione dello spazio disco e un equilibrio ottimale tra prestazioni e funzionalità avanzate, come gli snapshot. Sebbene ZFS e btrfs offrano vantaggi aggiuntivi in termini di robustezza e funzionalità avanzate, LVM si distingue per la sua versatilità e facilità di implementazione, rendendolo adatto a una vasta gamma di scenari applicativi.

La selezione del disco fisico per la configurazione dello storage pool LVM richiede la massima cautela. Un errore in questa fase può causare conseguenze gravi, inclusa la perdita permanente di dati sull'host e nei container. Utilizzare un device a blocchi già montato nel filesystem host può portare a corruzione dei dati e instabilità del sistema. È quindi fondamentale assicurarsi che il disco selezionato sia dedicato e non montato nel filesystem del sistema operativo host.

La configurazione dello storage pool basato su LVM, denominato `lxdpool`, è definita nel seguente codice YAML:

```
name: lxdpool
driver: lvm
config:
  lvm.thinpool_name: LXDThinPool
  lvm.vg_name: lxdpool
  volatile.initial_source: /dev/sda
  source: lxdpool
```

Come indicato nella configurazione, l'utilizzo di un device a blocchi come sorgente (`source: lxdpool`), combinato con il *thin provisioning* implementato tramite `LXDThinPool`, consente un'allocazione ottimale e dinamica dello spazio di storage. Tuttavia, è cruciale identificare correttamente un disco fisico non montato per evitare i rischi menzionati.

Notare che `source: lxdpool` in questo contesto non si riferisce al disco fisico, ma è un nome logico interno a LXD per indicare che lo storage pool userà LVM e il thin pool LXDThinPool. La sorgente fisica**,** ovvero il disco, è indicata in `volatile.initial_source: /dev/sda` (o `/dev/sdb` nell'esempio successivo).

Verifica della Configurazione

Dopo aver creato lo storage pool LVM con il comando **lxc storage create lxdpool lvm**, è importante verificare che la configurazione sia corretta e che il disco desiderato sia stato utilizzato come sorgente. Un metodo efficace per farlo è usare il comando **lvs** (Logical Volume Scan), che fornisce dettagli sui volumi logici LVM all'interno del volume group **lxdpool**.

```
LV                              VG      Attr       LSize   Pool
LXDThinPool                     lxdpool twi-aotz-- <10,48t
virtual-machines_iopp           lxdpool Vwi-aotz-k 100,00m LXDThinPool
virtual-machines_iopp-snap0     lxdpool Vri---tz-k 100,00m LXDThinPool
virtual-machines_iopp-snap0.block lxdpool Vri---tz-k 200,00g LXDThinPool
virtual-machines_iopp.block     lxdpool Vwi-aotz-k 200,00g LXDThinPool
```

```
Origin                      Data%  Meta%  Move Log Cpy%Sync Convert
                            2,26   11,08
                            46,15
virtual-machines_iopp
virtual-machines_iopp.block
                            21,00
```

Figura 3 - Output del comando lvs

L'output del comando **lvs** permette di verificare diversi aspetti chiave della configurazione:

1. Volume Group (VG):
   - La colonna "VG" deve mostrare **lxdpool** per tutti i volumi logici elencati. Questo conferma che appartengono al volume group creato per LXD.
2. Logical Volumes (LV):
   - LXDThinPool: Questo thin pool è essenziale per il funzionamento dello storage pool LXD con *thin provisioning.* La sua presenza è un indicatore positivo.

- lxdpool: Questo volume logico thinly provisioned funge da contenitore principale per le istanze LXD.
- Potrebbero essere presenti ulteriori volumi logici dinamici, come quelli associati a istanze (es. **virtual-machines_iopp**) o ai loro snapshot (es. **virtual-machines_iopp-snap0**). Questi volumi dimostrano che lo storage pool è operativo e gestisce correttamente la creazione di istanze.

3. Pool:
   - Per i volumi logici associati alle istanze, la colonna "Pool" deve indicare **LXDThinPool**. Questo conferma che le istanze stanno utilizzando il *thin provisioning* come previsto.
4. Attributi (Attr):
   - Gli attributi dei volumi logici forniscono informazioni sul loro stato:
     - **twi-aotz--** per **LXDThinPool**: Indica un thin pool attivo.
     - **lvo--------** per **lxdpool**: Indica un volume logico thinly provisioned attivo.
     - **Vwi-aotz-k** e **Vri---tz-k** per i volumi delle istanze: Indicano volumi virtuali thinly provisioned, potenzialmente snapshot.

Se l'output di **lvs** mostra una struttura coerente con quanto descritto (presenza di **LXDThinPool**, **lxdpool**, eventuali volumi per istanze e attributi corretti), significa che lo storage pool LVM è stato configurato correttamente ed è funzionante.

Verifica Finale con lxc storage show

Per una verifica riassuntiva della configurazione dello storage pool LVM, è possibile utilizzare il comando:

```
lxc storage show lxdpool
```

Questo comando conferma che la sorgente (**source**) sia impostata correttamente su **lxdpool** e che il driver sia **lvm**. Se i valori sono coerenti, la configurazione è completa e valida.

Identificazione di un Disco Non Montato

Per individuare un disco fisico non montato adatto alla creazione dello storage pool LVM, si possono utilizzare i seguenti comandi:

1. **lsblk**
   Il comando **lsblk** elenca i dispositivi a blocchi disponibili nel sistema in formato ad albero. Un disco non montato non avrà alcun valore nella colonna **MOUNTPOINT**.
   Esempio di output:

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
└─sda1   8:1    0   20G  0 part /
sdb      8:16   0  100G  0 disk
sdc      8:32   0   50G  0 disk
```

   In questo caso, **/dev/sdb** e **/dev/sdc** non hanno punti di mount e possono essere candidati adatti per lo storage pool. Tuttavia, è necessario verificarne ulteriormente il contenuto prima dell'utilizzo.

2. **fdisk –l**
   Il comando fdisk -l mostra le partizioni di tutti i dischi riconosciuti dal sistema. Dischi senza partizioni formattate o informazioni di mount suggeriscono che non sono in uso.

   Esempio di output di `fdisk -l`:

```
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x1234abcd

Device     Boot Start      End  Sectors Size Id Type
/dev/sda1  *     2048 41943039 41940992  20G 83 Linux

Disk /dev/sdb: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/sdc: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Anche qui, **/dev/sdb** e **/dev/sdc** non mostrano partizioni o mountpoint, rendendoli potenzialmente idonei per lo storage pool.

3.5.2. Procedura di Migrazione

La migrazione della macchina virtuale (VM) dal server di origine all'ambiente LXD di destinazione è stata eseguita in due fasi principali: l'importazione dell'immagine della VM e la configurazione di rete per l'adattamento all'infrastruttura di destinazione.

3.5.2.1. Importazione della VM

L'immagine della VM, esportata dal server di origine in formato .tgz (vm.tgz), è stata trasferita all'ambiente LXD e importata tramite il comando:

```
lxc import vm.tgz --storage lxdpool
```

- **lxc import** : Strumento nativo di LXD per importare immagini di container e VM.
- **--storage lxdpool** : Alloca lo storage della VM nello storage pool LVM **lxdpool**, sfruttando funzionalità come il *thin provisioning* e gli snapshot.

Successivamente, è stato applicato il profilo di rete **PublicIP** alla VM tramite:

```
lxc profile add <nome-container> PublicIP
```

Questo profilo garantisce la connettività diretta alla rete fisica tramite interfaccia bridged.

3.5.2.2. Configurazione di Rete

Per assegnare un indirizzo IP statico alla VM, è stata implementata una configurazione Netplan all'interno della VM. Il file YAML (/etc/netplan/) è stato strutturato come segue:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp5s0:
```

```
dhcp4: false
addresses:
  - 192.168.0.151/24
gateway4: 192.168.0.1
nameservers:
  addresses:
    - 8.8.8.8
    - 8.8.4.4
```

Analizzando la configurazione YAML, si evidenzia:

- `version: 2`: Specifica la versione del formato di configurazione Netplan.
- `renderer: networkd`: Indica l'utilizzo di `networkd` come backend di gestione della rete, il renderer predefinito in Ubuntu Server.
- `ethernets:`: Definisce la configurazione per le interfacce di rete Ethernet.
- `enp5s0:`: Identifica l'interfaccia di rete specifica da configurare all'interno della VM. Il nome dell'interfaccia (`enp5s0`) può variare a seconda del sistema operativo guest e della configurazione hardware virtuale.
- `dhcp4: false`: Disabilita l'acquisizione automatica dell'indirizzo IP tramite DHCP per IPv4, indicando l'intenzione di configurare un indirizzo IP statico.
- `addresses: - 192.168.0.151/24`: Assegna l'indirizzo IP statico `192.168.0.151` con subnet mask `/24` (equivalente a `255.255.255.0`). La scelta di un indirizzo IP statico è motivata dalla necessità di garantire un indirizzo IP persistente e prevedibile per la VM all'interno della rete di destinazione.
- `gateway4: 192.168.0.1`: Definisce l'indirizzo IP del gateway predefinito (`192.168.0.1`) per instradare il traffico di rete verso l'esterno della sottorete locale.
- `nameservers: addresses: - 8.8.8.8 - 8.8.4.4`: Configura i server DNS primario (`8.8.8.8`) e secondario (`8.8.4.4`) per la risoluzione dei nomi di dominio. In questo caso, sono stati utilizzati i server DNS pubblici di Google.

Per applicare la configurazione di rete definita nel file Netplan, è stato eseguito il comando:

```
sudo netplan apply
```

3.5.3. Verifica dell'Implementazione

Dopo la migrazione e la configurazione, sono state condotte verifiche per confermare il successo dell'operazione:

- **Integrità dei Dati:** La procedura di importazione tramite lxc import ha garantito il trasferimento completo e integro dei dati della VM.
- **Connettività di Rete:** La configurazione Netplan ha assegnato correttamente l'indirizzo IP statico 192.168.0.151, confermato da test di connettività (ping, SSH, ecc.).
- **Accesso Bridged:** La configurazione "bridged" del profilo PublicIP ha permesso alla VM di operare direttamente sulla rete fisica.
- **Ottimizzazione dello Storage:** L'uso dello storage pool LVM lxdpool ha consentito di sfruttare il thin provisioning, ottimizzando l'utilizzo dello spazio disco.

In conclusione, la VM è stata migrata con successo nell'ambiente LXD, risultando pienamente operativa e beneficiando delle funzionalità offerte da LXD, LVM e del profilo di rete PublicIP.

3.6. Limitazioni e potenziali problemi

Nonostante i numerosi vantaggi offerti da LXD, è importante riconoscere alcune limitazioni e possibili criticità che devono essere attentamente valutate durante la progettazione e l'implementazione di infrastrutture basate su container.

1. **Compatibilità Hardware e Kernel**: LXD dipende strettamente dal kernel Linux e dall'architettura del sistema operativo ospitante. Questa dipendenza può generare problemi di compatibilità o prestazioni su distribuzioni Linux non pienamente supportate o su hardware obsoleto che non implementa le funzionalità avanzate del kernel richieste da LXD.
2. **Livelli di Isolamento**: Sebbene LXD fornisca un isolamento applicativo robusto e sufficiente per molte casistiche, questo differisce concettualmente e tecnicamente dall'isolamento più rigoroso garantito dalle macchine virtuali tradizionali (basate su hypervisor). Per applicazioni che richiedono livelli elevati di sicurezza o isolamento hardware completo, LXD potrebbe non essere la soluzione ideale rispetto a tecnologie di virtualizzazione full-stack.
3. **Scalabilità e Gestione delle Risorse**: Sebbene LXD sia progettato per gestire un numero elevato di container in modo scalabile, ambienti particolarmente densi, con migliaia di container attivi simultaneamente, potrebbero evidenziare colli di bottiglia nella gestione delle risorse. In tali scenari, è fondamentale condurre test di performance specifici e ottimizzare sia la configurazione infrastrutturale che il carico di lavoro applicativo per garantire prestazioni ottimali.

4. Apache2: Web Server e Reverse Proxy

Il presente capitolo è dedicato alla configurazione di Apache2 come *reverse proxy*, un ruolo strategico in molteplici architetture di rete per garantire scalabilità, sicurezza e ottimizzazione delle risorse.

Un reverse proxy agisce da intermediario tra i client esterni (come browser web) e i server backend interni (nel nostro caso, container LXD), offrendo numerosi vantaggi operativi. Dal punto di vista funzionale, il reverse proxy intercetta tutte le richieste provenienti dall'esterno, applica eventuali regole di filtraggio e instradamento, inoltrandole poi ai server backend appropriati. Successivamente, le risposte dei backend vengono ritrasmesse ai client, nascondendo la struttura interna della rete e garantendo un unico punto di accesso per l'infrastruttura web.

L'adozione di Apache2 come reverse proxy per container LXD offre i seguenti vantaggi chiave:

- **Bilanciamento del carico**: Distribuzione intelligente del traffico per evitare il sovraccarico di singoli server.
- **Maggiore sicurezza**: Protezione dei backend mediante filtraggio del traffico e mascheramento della topologia interna.
- **Caching avanzato**: Riduzione della latenza e ottimizzazione dell'uso della banda passante.
- **Gestione centralizzata di SSL/TLS**: Semplificazione della sicurezza crittografica e miglioramento delle performance.
- **Instradamento avanzato**: Possibilità di implementare regole flessibili per il routing del traffico.

L'immagine seguente (Figura 4) illustra il flusso delle richieste in un'architettura basata **su** Apache2 come reverse proxy per LXD**.**

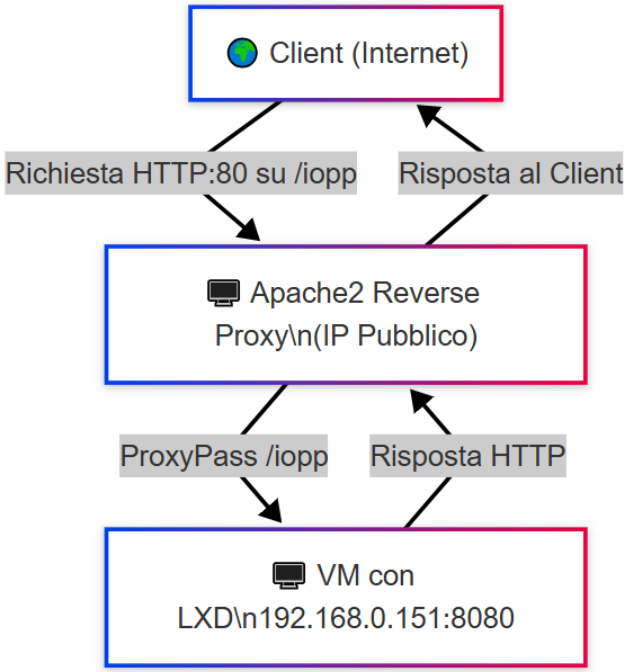

Figura 4 - Reverse Proxy con Apache2

In questo scenario:

1. Un client su Internet effettua una richiesta HTTP al server con IP pubblico, sulla rotta `/iopp`.
2. Apache2, configurato come reverse proxy, intercetta la richiesta e la inoltra alla VM con LXD all'indirizzo `192.168.0.151:8080`.

3. La VM elabora la richiesta e invia la risposta ad Apache2.
4. Apache2 inoltra la risposta al client, fungendo da unico punto di accesso alla rete interna.

Questo meccanismo consente di ottimizzare la sicurezza e la gestione del traffico, proteggendo i backend da accessi diretti e offrendo un livello di controllo avanzato sulla distribuzione delle richieste.

Le sezioni seguenti approfondiranno i dettagli tecnici relativi all'installazione, alla configurazione di base e alla configurazione avanzata di Apache2 come reverse proxy in un ambiente di containerizzazione LXD, con esempi pratici e best practice operative.

4.1. Installazione e Configurazione di Base di Apache2

L'installazione e la configurazione di base di Apache2 costituiscono i passaggi preliminari essenziali per predisporre il server web all'utilizzo come reverse proxy. Questa fase comprende l'installazione del pacchetto Apache2 dai repository ufficiali APT, la verifica dell'integrità dell'installazione e dello stato operativo del servizio, l'attivazione dei moduli necessari per la funzione di reverse proxy e l'applicazione delle configurazioni di sicurezza basilari per migliorare la robustezza del server.

4.1.1. Installazione di Apache2

L'installazione di Apache2 è stata eseguita tramite il sistema di gestione dei pacchetti apt, strumento di riferimento per l'installazione, l'aggiornamento e la rimozione di software su sistemi operativi Debian e derivate, quali Ubuntu. La procedura di installazione di Apache2 si articola nei seguenti comandi da eseguire tramite terminale con privilegi amministrativi (sudo):

```
sudo apt update
sudo apt install apache2
```

- **sudo apt update**: Questo comando aggiorna l'indice locale dei pacchetti disponibili nei repository APT, garantendo l'accesso alle versioni più recenti e sicure del software. L'aggiornamento previene potenziali problemi di dipendenze o incompatibilità durante l'installazione.
- **sudo apt install apache2**: Questo comando installa il pacchetto Apache2 e tutte le dipendenze necessarie, configurando il servizio affinché venga avviato automaticamente all'accensione del sistema.

Terminata l'installazione, è necessario verificare il corretto funzionamento di Apache2 attraverso i seguenti metodi:

- **verifica della versione di Apache2**:

  ```
  apache2 -v
  ```

  Questo comando restituisce la versione del software installata, confermando la disponibilità dell'eseguibile.

- **verifica dello stato del servizio:**

  ```
  sudo systemctl status apache2
  ```

  L'output di questo comando fornisce informazioni dettagliate sullo stato del servizio, confermando se Apache2 è attivo e funzionante.

4.1.2. Configurazione di Base di Apache2

Dopo aver verificato l'operatività del servizio, si procede con la configurazione di base per abilitare le funzionalità di reverse proxy e incrementare la sicurezza del server.

Abilitazione dei Moduli Necessari

L'abilitazione dei moduli Apache2 richiesti per il reverse proxy avviene tramite il comando `a2enmod`, che consente di attivare moduli aggiuntivi. I moduli essenziali sono:

```
sudo a2enmod proxy
sudo a2enmod proxy_http
sudo a2enmod ssl
sudo a2enmod headers
```

- **proxy e proxy_http**: Forniscono il supporto per il proxying delle richieste HTTP.
- **ssl**: Abilita la gestione delle connessioni HTTPS.
- **headers**: Consente la manipolazione degli header HTTP, utile per politiche di sicurezza avanzate.

Dopo l'abilitazione dei moduli, è necessario riavviare Apache2 per rendere effettive le modifiche

```
sudo systemctl restart apache2
```

Configurazione dei Virtual Host

Apache2 utilizza il concetto di Virtual Host per gestire più siti o applicazioni su un unico server. Per configurare un reverse proxy, è necessario creare un file di configurazione specifico all'interno della directory `/etc/apache2/sites-available/` e abilitarlo con il comando:

```
sudo a2ensite nome_virtual_host.conf
sudo systemctl reload apache2
```

La configurazione dettagliata del Virtual Host sarà approfondita nella sezione 4.2.

Disabilitazione del Directory Browser

Per motivi di sicurezza, è consigliabile disabilitare la visualizzazione dell'indice delle directory, prevenendo potenziali esposizioni di informazioni sensibili. Questo viene realizzato modificando il file di configurazione del Virtual Host predefinito (`000-default.conf` o `default.conf`), aggiungendo la direttiva:

```
<Directory /var/www/>
    Options -Indexes +FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

Dopo aver applicato la modifica, è necessario riavviare Apache2 per renderla effettiva:

```
sudo systemctl restart apache2
```

Questa configurazione impedisce l'elenco dei file presenti nelle directory non dotate di file `index`, mitigando il rischio di esposizione di informazioni riservate.

4.2. Configurazione del Reverse Proxy con Apache2

La configurazione avanzata di Apache2 come reverse proxy si articola in una serie di passaggi operativi volti a definire in modo preciso e puntuale il comportamento del reverse proxy, in termini di routing del traffico, gestione della sicurezza e ottimizzazione delle performance.

La configurazione avanzata prevede la creazione e la configurazione di Virtual Host dedicati alla funzione di proxying, la definizione delle direttive di proxying per indirizzare il traffico in ingresso verso i container LXD backend in base a regole specifiche, la gestione del traffico sicuro tramite protocollo HTTPS e configurazione SSL/TLS, e, opzionalmente, l'implementazione del bilanciamento del carico per distribuire il traffico tra più container backend e garantire scalabilità e resilienza.

4.2.1. Creazione dei Virtual Host per il Reverse Proxy

Per configurare Apache2 come reverse proxy per i container LXD, è necessario creare uno o più file di configurazione Virtual Host dedicati specificamente alla funzione di reverse proxy. Come precedentemente menzionato, i file di configurazione dei Virtual Host vengono tipicamente memorizzati nella directory /etc/apache2/sites-available/. È prassi comune creare un nuovo file di configurazione dedicato per il reverse proxy, denominandolo in modo descrittivo, ad esempio, reverse-proxy.conf, e memorizzandolo all'interno della directory /etc/apache2/sites-available/.

L'esempio seguente illustra la configurazione di un Virtual Host per il reverse proxy che indirizza il traffico in ingresso con un determinato percorso URL (/iopp in questo esempio) verso uno specifico container LXD backend:

```
<VirtualHost *:80>
    ServerName reverse-proxy.example.com
    ServerAdmin webmaster@example.com
    DocumentRoot /var/www/html

    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined

    ProxyPass /iopp http://192.168.0.151:8080/
    ProxyPassReverse /iopp http://192.168.0.151:8080/

</VirtualHost>
```

L'analisi delle direttive di configurazione del Virtual Host `reverse-proxy.conf` evidenzia:

- `<VirtualHost *:80>`: Definisce un Virtual Host che ascolta su tutte le interfacce di rete (`*`) sulla porta HTTP standard `80`. È possibile specificare un indirizzo IP specifico al posto di `*` per limitare l'ascolto ad una interfaccia di rete specifica.
- `ServerName reverse-proxy.example.com`: Specifica il nome di dominio o hostname a cui il Virtual Host risponderà. È necessario sostituire `reverse-proxy.example.com` con il nome di dominio desiderato per il reverse proxy. Questo nome di dominio dovrà essere configurato nel DNS per puntare all'indirizzo IP del server Apache2.
- `ServerAdmin webmaster@example.com`: Specifica l'indirizzo email dell'amministratore del server web. Questo indirizzo email viene tipicamente utilizzato per scopi amministrativi e per la visualizzazione di messaggi di errore del server. È necessario sostituire `webmaster@example.com` con un indirizzo email valido.
- `DocumentRoot /var/www/html`: Specifica la directory radice del Virtual Host, ovvero la directory da cui Apache2 servirà i file statici nel caso in cui la richiesta non venga indirizzata ad un server backend tramite proxy. In una configurazione di reverse proxy puro, questa direttiva potrebbe non essere

rilevante, ma è comunque necessario specificare una directory valida. `/var/www/html` è la directory radice di default per i Virtual Host in Apache2.

- `ErrorLog ${APACHE_LOG_DIR}/error.log`: Specifica il file di log per gli errori del Virtual Host. `${APACHE_LOG_DIR}` è una variabile di ambiente Apache2 che punta alla directory di log di Apache2. Il file `error.log` conterrà i messaggi di errore generati dal Virtual Host.
- `CustomLog ${APACHE_LOG_DIR}/access.log combined`: Specifica il file di log per gli accessi al Virtual Host. Il file `access.log` conterrà le informazioni relative alle richieste HTTP ricevute dal Virtual Host. Il formato `combined` specifica un formato di log dettagliato che include informazioni quali l'indirizzo IP del client, la data e l'ora della richiesta, il percorso URL richiesto, il codice di stato HTTP e lo User-Agent del client.
- `ProxyPass /iopp http://192.168.0.151:8080/`: Questa direttiva è la direttiva fondamentale per la configurazione del reverse proxy. `ProxyPass /iopp` indica che tutte le richieste in ingresso con percorso URL che inizia con `/iopp` (ovvero, tutte le richieste) devono essere indirizzate al server backend specificato. `http://192.168.0.151:8080/` specifica l'URL del server backend a cui le richieste devono essere inoltrate. In questo esempio, le richieste vengono indirizzate al server HTTP in ascolto sull'indirizzo IP `192.168.0.151` e sulla porta `8080`. `192.168.0.151` rappresenta l'indirizzo IP del container LXD backend, mentre `8080` rappresenta la porta su cui il servizio web all'interno del container è in ascolto. È necessario sostituire `192.168.0.151:8080` con l'indirizzo IP e la porta effettivi del container LXD backend.
- `ProxyPassReverse /iopp http://192.168.0.151:8080/`: La direttiva `ProxyPassReverse` è complementare alla direttiva `ProxyPass` e è fondamentale per il corretto funzionamento del reverse proxy in scenari in cui il server backend genera redirect HTTP o URL assoluti nelle risposte. `ProxyPassReverse` modifica gli header di risposta HTTP provenienti dal server backend, riscrivendo gli URL di redirect e gli URL assoluti in modo che puntino correttamente al reverse proxy stesso anziché al server backend. In tal modo, il browser web del client seguirà correttamente i redirect e accederà correttamente alle risorse tramite il reverse proxy. La direttiva `ProxyPassReverse` deve essere configurata con lo stesso URL backend della direttiva `ProxyPass`.

Per abilitare il Virtual Host configurato nel file `reverse-proxy.conf`, è necessario creare un link simbolico dal file di configurazione alla directory `/etc/apache2/sites-enabled/` tramite il comando `a2ensite` e riavviare il servizio Apache2 per rendere effettive le modifiche:

```
sudo a2ensite reverse-proxy.conf
sudo systemctl restart apache2
```

4.2.2. Configurazione SSL/TLS per HTTPS

Per abilitare l'accesso sicuro all'interfaccia web del reverse proxy tramite il protocollo HTTPS (Hypertext Transfer Protocol Secure), è indispensabile configurare SSL/TLS (Secure Sockets Layer/Transport Layer Security) per il Virtual Host dedicato al reverse proxy. SSL/TLS è un protocollo crittografico standard che fornisce autenticazione, crittografia e integrità dei dati per le comunicazioni di rete, garantendo la privacy e la sicurezza delle informazioni trasmesse tra il client (es. browser web) e il server web (in questo caso, il reverse proxy Apache2).

La configurazione SSL/TLS in Apache2 prevede principalmente due passaggi operativi: la creazione o l'ottenimento di un certificato SSL/TLS e la configurazione del Virtual Host Apache2 per utilizzare tale certificato per le connessioni HTTPS. È possibile optare per l'utilizzo di certificati SSL/TLS autogenerati per ambienti di test, sviluppo o uso interno, oppure ottenere certificati digitali firmati da una Certificate Authority (CA) pubblica o privata per ambienti di produzione e siti web accessibili pubblicamente via Internet.

- **Creazione di Certificati SSL/TLS Autogenerati per Test e Sviluppo:** Per la creazione rapida e semplificata di certificati SSL/TLS autogenerati, adatti per ambienti di test, sviluppo o uso interno in cui la validazione da parte di una CA pubblica non è un requisito, è possibile utilizzare lo strumento a riga di comando `openssl`, utility versatile e potente per la gestione di operazioni crittografiche. Un

comando di esempio per la generazione di un certificato SSL/TLS autogenerato con `openssl` è il seguente:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt -subj '/CN=reverse-proxy.example.com'
```

L'analisi dettagliata dei parametri e delle opzioni del comando `openssl` evidenzia le seguenti funzionalità:

- `sudo openssl req -x509`: Invoca il comando `openssl req` per la gestione delle richieste di certificato X.509. L'opzione `-x509` specifica che si desidera generare direttamente un certificato autofirmato, anziché una richiesta di certificato (Certificate Signing Request - CSR) da inviare ad una CA.
- `-nodes`: Specifica che la chiave privata generata non deve essere crittografata con una passphrase. L'omissione di una passphrase semplifica la configurazione del server web, ma riduce leggermente la sicurezza della chiave privata, in quanto non è protetta da password. Per ambienti di produzione, è raccomandabile proteggere la chiave privata con una passphrase robusta, o implementare meccanismi di protezione della chiave privata basati su hardware (es. HSM - Hardware Security Module).
- `-days 365`: Specifica la validità del certificato autofirmato in giorni. `365` indica che il certificato sarà valido per 365 giorni (un anno). È possibile modificare questo valore per adattare la validità del certificato alle esigenze specifiche.
- `-newkey rsa:2048`: Specifica che deve essere generata una nuova chiave privata RSA (Rivest-Shamir-Adleman) con una lunghezza di 2048 bit. RSA è un algoritmo di crittografia a chiave pubblica ampiamente utilizzato e considerato sicuro. 2048 bit rappresenta una lunghezza di chiave minima raccomandata per garantire una sicurezza adeguata. È possibile utilizzare lunghezze di chiave superiori (es. 4096 bit) per una maggiore sicurezza, a scapito di un leggero incremento dell'overhead computazionale.
- `-keyout /etc/ssl/private/apache-selfsigned.key`: Specifica il percorso del file in cui verrà memorizzata la chiave privata RSA generata. `/etc/ssl/private/apache-selfsigned.key` rappresenta un percorso di esempio, è importante assicurarsi che la directory `/etc/ssl/private/` abbia permessi restrittivi (es. `chmod 700 /etc/ssl/private/`) per proteggere la chiave privata da accessi non autorizzati.
- `-out /etc/ssl/certs/apache-selfsigned.crt`: Specifica il percorso del file in cui verrà memorizzato il certificato autofirmato generato. `/etc/ssl/certs/apache-selfsigned.crt` rappresenta un percorso di esempio, la directory `/etc/ssl/certs/` è tipicamente utilizzata per memorizzare i certificati SSL/TLS del sistema.
- `-subj '/CN=reverse-proxy.example.com'`: Specifica il Subject Distinguished Name (DN) del certificato, ovvero le informazioni identificative del soggetto (il server web) a cui il certificato è rilasciato. `/CN=reverse-proxy.example.com` specifica il Common Name (CN) del certificato, che corrisponde al nome di dominio o hostname del reverse proxy (`reverse-proxy.example.com` in questo esempio). È fondamentale sostituire `reverse-proxy.example.com` con il nome di dominio effettivo del reverse proxy, in quanto il Common Name del certificato deve corrispondere al nome di dominio utilizzato dai client per accedere al reverse proxy, per evitare errori di validazione del certificato da parte dei browser web. Il Subject DN può includere anche altre informazioni identificative, quali l'organizzazione (O), l'unità organizzativa (OU), la località (L), lo stato (ST) e il paese (C).

- **Configurazione del Virtual Host Apache2 per SSL/TLS:** Per configurare Apache2 per l'utilizzo di SSL/TLS e abilitare l'accesso sicuro tramite HTTPS per il Virtual Host del reverse proxy, è necessario modificare il file di configurazione del Virtual Host (`reverse-proxy.conf`) e aggiungere le direttive di configurazione SSL/TLS all'interno di un blocco `<VirtualHost *:443>`. È prassi comune duplicare il Virtual Host HTTP (configurato per la porta 80) e modificarlo per il supporto HTTPS (porta 443), mantenendo separate le configurazioni per HTTP e HTTPS. Un esempio di

configurazione di Virtual Host per HTTPS, basato sul Virtual Host HTTP precedentemente configurato, è il seguente:

```
<VirtualHost *:443>
    ServerName reverse-proxy.example.com
     ServerAdmin webmaster@example.com
     DocumentRoot /var/www/html

     ErrorLog ${APACHE_LOG_DIR}/error.log
     CustomLog ${APACHE_LOG_DIR}/access.log combined

     SSLEngine on
     SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt
     SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

           ProxyPass /iopp http://192.168.0.151:8080/
            ProxyPassReverse /iopp http://192.168.0.151:8080/
</VirtualHost>
```

Le direttive SSL/TLS aggiuntive, specifiche per la configurazione HTTPS, introdotte nel Virtual Host sono:

- `<VirtualHost *:443>`: Questa direttiva, analoga a `<VirtualHost *:80>` descritta in precedenza, definisce l'inizio di un blocco di configurazione Virtual Host, ma in questo caso specifica che il Virtual Host sarà in ascolto su tutte le interfacce di rete (`*`) sulla porta HTTPS standard `443`. Le richieste in ingresso sulla porta 443 verranno gestite da questo Virtual Host, abilitato per il protocollo HTTPS.
- `SSLEngine on`: La direttiva `SSLEngine on` abilita esplicitamente il motore SSL/TLS per il Virtual Host corrente. Questa direttiva è indispensabile per attivare il supporto HTTPS per il Virtual Host e per consentire ad Apache2 di gestire connessioni HTTPS crittografate.
- `SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt`: La direttiva `SSLCertificateFile` specifica il percorso del file contenente il certificato SSL/TLS del server web, ovvero il certificato digitale che il server web presenterà ai client durante la fase di handshake SSL/TLS per autenticare la propria identità e stabilire una connessione crittografata. `/etc/ssl/certs/apache-selfsigned.crt` rappresenta il percorso del file del certificato autofirmato generato con il comando `openssl` descritto in precedenza. È necessario sostituire `/etc/ssl/certs/apache-selfsigned.crt` con il percorso effettivo del file del certificato SSL/TLS valido (autogenerato o firmato da una CA) che si desidera utilizzare per il Virtual Host. In caso di utilizzo di certificati firmati da una CA, è possibile che sia necessario specificare anche la direttiva `SSLCertificateChainFile` per indicare il percorso del file contenente la catena di certificati intermedi della CA.
- `SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key`: La direttiva `SSLCertificateKeyFile` specifica il percorso del file contenente la chiave privata SSL/TLS del server web, ovvero la chiave crittografica privata corrispondente al certificato SSL/TLS specificato con la direttiva `SSLCertificateFile`. `/etc/ssl/private/apache-selfsigned.key` rappresenta il percorso del file della chiave privata generata con il comando `openssl` descritto in precedenza. È necessario sostituire `/etc/ssl/private/apache-selfsigned.key` con il percorso effettivo del file della chiave privata SSL/TLS corrispondente al certificato. È fondamentale proteggere adeguatamente il file della chiave privata, in quanto la compromissione della chiave privata comprometterebbe la sicurezza dell'intero sistema HTTPS.

Dopo aver modificato il file di configurazione del Virtual Host (`reverse-proxy.conf`) e aver creato o ottenuto i certificati SSL/TLS (file del certificato e file della chiave privata), è necessario riavviare il servizio Apache2 per applicare le modifiche e abilitare il supporto HTTPS per il Virtual Host:

```
sudo systemctl restart apache2
```

4.3. Sfide e possibili criticità

Nonostante i molteplici vantaggi offerti dall'adozione di Apache2 come reverse proxy, è essenziale considerare alcune sfide e potenziali criticità che possono emergere in fase di implementazione e gestione.

In primo luogo, è importante valutare l'overhead computazionale introdotto dal reverse proxy stesso: apache2, pur essendo un server web performante, può rappresentare un collo di bottiglia in scenari caratterizzati da volumi di traffico estremamente elevati, specialmente se sottoposto a un carico di lavoro intenso in termini di gestione delle connessioni, analisi delle richieste e proxying verso i backend.

In secondo luogo, la gestione dei certificati SSL/TLS richiede una configurazione accurata e una manutenzione costante. Errori nella configurazione SSL/TLS, o una gestione negligente del ciclo di vita dei certificati (es. mancato rinnovo, utilizzo di certificati obsoleti o non validi), possono generare problemi di latenza nelle connessioni HTTPS, compromettere la user experience e, soprattutto, introdurre vulnerabilità di sicurezza sfruttabili da potenziali attaccanti.

Infine, il bilanciamento del carico, sebbene rappresenti un potente strumento per la scalabilità e la resilienza, richiede una configurazione adeguata e una scelta oculata dell'algoritmo di bilanciamento. Una configurazione inadeguata del bilanciamento del carico può vanificare i benefici attesi, portando ad una distribuzione non equa del traffico tra i container backend, con conseguenti squilibri di carico, inefficienze prestazionali e potenziali punti di failure.

5. Conclusioni

Il presente rapporto tecnico ha dettagliatamente documentato il processo di implementazione e configurazione di un'infrastruttura di virtualizzazione avanzata. Questa architettura è stata realizzata attraverso l'integrazione sinergica di tecnologie chiave, in particolare una piattaforma di *container management* moderna per l'orchestrazione di container applicativi, e un server web configurato come *reverse proxy* per ottimizzare la distribuzione del traffico e centralizzare le politiche di sicurezza.

L'implementazione descritta ha portato alla creazione di un ambiente IT intrinsecamente performante, resiliente, scalabile e sicuro, pienamente rispondente alle esigenze di un contesto applicativo avanzato che richiede flessibilità, sicurezza e scalabilità. L'infrastruttura realizzata costituisce un solido fondamento tecnologico per qualsiasi organizzazione moderna, fornendo un ambiente virtualizzato efficiente, all'avanguardia e predisposto per future espansioni e miglioramenti.

5.1. Punti di Forza dell'Implementazione: Efficienza, Sicurezza e Flessibilità Operativa

L'architettura virtualizzata implementata si distingue per un insieme di punti di forza che ne comprovano l'efficacia e l'adeguatezza rispetto agli obiettivi tipici di un'infrastruttura IT moderna:

- **Efficienza e Scalabilità Operativa Ottimizzate tramite Containerizzazione:** la piattaforma di *container management* adottata si è dimostrata una soluzione di livello enterprise, capace di assicurare un elevato grado di isolamento applicativo tra i container, massimizzando contestualmente l'efficienza nell'utilizzo delle risorse hardware. La tecnologia di containerizzazione, intrinsecamente efficiente e performante, ha permesso di isolare in modo rigoroso applicazioni e servizi in unità virtualizzate autonome, semplificando in misura significativa le procedure di *deployment*, aggiornamento e gestione del ciclo di vita applicativo. La piattaforma, grazie alle sue funzionalità avanzate di gestione della rete e dello storage, ha abilitato la creazione di un ambiente containerizzato intrinsecamente scalabile e flessibile, capace di adattarsi dinamicamente alle esigenze evolutive dell'organizzazione.
- **Sicurezza Centralizzata e Robustezza Architetturale con Reverse Proxy:** l'adozione di un server web in configurazione *reverse proxy* ha introdotto un punto di accesso unificato e intrinsecamente protetto per le applicazioni containerizzate. Posizionato strategicamente come *front-end* dell'infrastruttura web, il *reverse proxy* ha reso possibile l'implementazione di meccanismi di sicurezza centralizzati e robusti, tra cui la terminazione SSL/TLS per le comunicazioni HTTPS, il filtraggio del traffico malevolo a livello applicativo e la protezione dei server *backend* dall'esposizione diretta alla rete esterna. Il *reverse proxy* ha ottimizzato la distribuzione del traffico web in ingresso, garantendo performance elevate e resilienza dell'infrastruttura web, e contribuendo in modo sostanziale al rafforzamento della postura di sicurezza complessiva dell'ambiente IT.
- **Flessibilità Architetturale e Integrazione di Rete Trasparente:** l'integrazione nativa della piattaforma di containerizzazione con un *bridge* di rete, configurato in modo puntuale mediante un sistema di configurazione di rete moderno, ha garantito una connettività di rete flessibile, efficiente e trasparente per i container. I container, dinamicamente associati al *bridge* di rete, sono stati messi in condizione di comunicare in modo efficace e diretto con la rete fisica sottostante, beneficiando di performance di rete elevate e di una gestione semplificata degli indirizzi IP. Tale integrazione di rete avanzata ha consentito di superare le limitazioni tipiche di modelli di virtualizzazione più convenzionali, offrendo ai container un ambiente di rete assimilabile a quello di macchine fisiche, con piena flessibilità nella configurazione e nella gestione delle risorse di rete.
- **Gestione Semplificata e Intuitiva tramite Interfaccia Web di Gestione:** l'implementazione di un'interfaccia web di gestione, unitamente alla configurazione di certificati SSL/TLS personalizzati per l'accesso sicuro tramite HTTPS, ha reso disponibile un'interfaccia di gestione moderna, intuitiva ed efficiente per l'ambiente virtualizzato. L'interfaccia web ha semplificato in modo significativo le operazioni di gestione dei container, quali la creazione, l'avvio, l'arresto, il monitoraggio e la configurazione, proponendo un'alternativa *user-friendly* alla gestione via riga di comando. L'adozione

di *best practices* di sicurezza, quali la protezione delle chiavi private impiegate per i certificati SSL/TLS e la pianificazione del rinnovo sistematico dei certificati medesimi, ha ulteriormente consolidato la sicurezza dell'interfaccia di gestione web e dell'intera infrastruttura.

5.2. Sviluppi Futuri e Aree di Miglioramento: Verso un'Infrastruttura Dinamica e Proattiva

Nonostante i molteplici vantaggi e la solidità dell'infrastruttura basata sulla containerizzazione e sul *reverse proxy*, è essenziale riconoscere le potenziali limitazioni e criticità intrinseche a tali tecnologie, e delineare strategie di mitigazione efficaci per garantire un ambiente performante, sicuro e scalabile nel lungo periodo. Le principali aree di sviluppo e attenzione prioritaria concernono la scalabilità e la gestione delle risorse, il miglioramento continuo della sicurezza, l'automazione delle operazioni di gestione e l'implementazione di un sistema centralizzato di *log management*.

5.2.1. Scalabilità e Gestione Dinamica delle Risorse

Una potenziale limitazione intrinseca delle infrastrutture containerizzate risiede nella gestione efficiente delle risorse in contesti di elevata densità e carico di lavoro variabile. Al fine di mitigare il rischio di spreco di risorse, potenziali colli di bottiglia prestazionali e problematiche di scalabilità, si rende imperativo adottare strategie di monitoraggio proattivo e ottimizzazione dinamica delle risorse.

L'implementazione di una soluzione di monitoraggio centralizzata si configura come elemento imprescindibile per acquisire *real-time visibility* sull'utilizzo delle risorse hardware da parte della piattaforma di containerizzazione, dei container e del *reverse proxy*. Il monitoraggio continuativo delle metriche prestazionali consente di identificare tempestivamente eventuali anomalie, *trend* di crescita del carico, o situazioni di sottoutilizzo o sovrautilizzo delle risorse. Sulla base dei dati di monitoraggio acquisiti, sarà possibile implementare strategie di ottimizzazione dinamica delle risorse, quali: l'*autoscaling* orizzontale dei container in funzione del carico, la riallocazione dinamica delle risorse hardware tra i container e l'ottimizzazione della configurazione della piattaforma di containerizzazione e del *reverse proxy* per massimizzare l'efficienza e minimizzare l'*overhead* operativo.

L'analisi predittiva, basata su tecniche di *machine learning*, rappresenta un'evoluzione avanzata del monitoraggio, permettendo di anticipare i picchi di carico e allocare preventivamente le risorse necessarie, ottimizzando ulteriormente l'impiego dell'infrastruttura e garantendo *performance* elevate anche in condizioni di carico variabile.

5.2.2. Miglioramento Proattivo della Sicurezza e Vulnerability Assessment Continuativo

La sicurezza costituisce una priorità critica per qualsiasi infrastruttura IT: benché l'adozione di un *reverse proxy* e la configurazione SSL/TLS forniscano una solida base di sicurezza, si rivela imprescindibile implementare strategie di miglioramento continuo della sicurezza e adottare strumenti di protezione proattiva per mitigare le minacce emergenti e le potenziali vulnerabilità.

L'integrazione di un Web Application Firewall (WAF) avanzato, posizionato a protezione del *reverse proxy*, rappresenta un passo fondamentale per rafforzare la sicurezza dell'infrastruttura web: un WAF moderno, dotato di funzionalità di analisi comportamentale, *machine learning* e protezione da attacchi *zero-day*, è in grado di rilevare e bloccare in tempo reale un ampio spettro di minacce web, quali *injection* SQL, *cross-site scripting* (XSS), attacchi DDoS e vulnerabilità applicative.

L'implementazione di sistemi di *intrusion detection and prevention* (IDS/IPS) a livello di rete e di *host* consente di monitorare attivamente il traffico di rete e l'attività dei sistemi, rilevando comportamenti anomali o sospetti e prevenendo intrusioni e attacchi.

L'esecuzione regolare di *vulnerability assessment* e *penetration testing*, sia a livello infrastrutturale che applicativo, permette di identificare proattivamente eventuali vulnerabilità *software* o errori di configurazione, consentendo di applicare tempestivamente *patch* di sicurezza e correttivi, e di mantenere un elevato livello di protezione nel tempo.

5.2.3. Automazione Avanzata della Gestione e Orchestrazione Infrastrutturale

La gestione manuale di un'infrastruttura complessa basata sulla containerizzazione e su un *reverse proxy* può risultare onerosa, complessa e intrinsecamente soggetta a errori umani. L'automazione della gestione si configura come strategia di mitigazione fondamentale per semplificare le operazioni, ridurre i costi operativi, incrementare l'efficienza e garantire la coerenza della configurazione nel tempo.

L'adozione di strumenti di orchestrazione infrastrutturale consente di automatizzare il *deployment*, la configurazione e la gestione dell'intera infrastruttura in modo dichiarativo e riproducibile, mentre l'automazione può essere estesa a molteplici aspetti operativi, tra cui la creazione e configurazione automatica di container, la gestione delle reti virtualizzate, la configurazione del *reverse proxy*, il *deployment* e l'aggiornamento delle applicazioni containerizzate e la gestione centralizzata dei certificati SSL/TLS.

L'implementazione di una piattaforma di configurazione centralizzata rende possibile gestire in modo unitario la configurazione di tutti i componenti infrastrutturali, applicare *policy* di sicurezza uniformi, e semplificare le operazioni di aggiornamento e manutenzione.

L'automazione del monitoraggio, attraverso l'integrazione della piattaforma di monitoraggio con sistemi di *alerting* e notifica automatica, consente di essere avvisati tempestivamente in caso di problematiche o anomalie, e di automatizzare altresì le procedure di *remediation* e ripristino in caso di *failure*, migliorando la resilienza e la continuità di servizio dell'infrastruttura.

5.2.4. Monitoraggio Centralizzato e Gestione Avanzata dei Log

Un'ulteriore area di miglioramento di rilievo per l'infrastruttura concerne l'implementazione di un sistema di monitoraggio centralizzato e di gestione avanzata dei *log*: benché la configurazione corrente preveda la generazione di *log* da parte del *reverse proxy*, della piattaforma di containerizzazione e dei container, si rileva la mancanza di una strategia centralizzata per la raccolta, l'analisi e l'archiviazione di tali dati.

Un approccio proattivo alla gestione dei *log* si rivela fondamentale per:

- **Identificare tempestivamente problematiche di sicurezza:** rilevare attività sospette, tentativi di accesso non autorizzati o anomalie comportamentali attraverso l'analisi dei *log* di tutti i componenti infrastrutturali.
- **Diagnosticare e risolvere problematiche prestazionali:** individuare colli di bottiglia, errori di configurazione o problematiche applicative mediante l'analisi dei *log* del *reverse proxy*, della piattaforma di containerizzazione e dei container.
- **Monitorare l'utilizzo delle risorse infrastrutturali:** tracciare l'utilizzo di CPU, memoria, storage e rete da parte dei container e del *reverse proxy* al fine di ottimizzare l'allocazione delle risorse e prevenire fenomeni di sovrautilizzo.
- **Garantire la conformità normativa:** assicurare la conservazione dei *log* per un periodo temporale adeguato e implementare meccanismi di tracciamento degli accessi e delle attività degli utenti, in ottemperanza ai requisiti normativi in materia di *privacy* e sicurezza dei dati.

Per conseguire tali obiettivi, si raccomanda di implementare le seguenti azioni:

- **Centralizzare la raccolta dei *log*:** configurare la piattaforma di containerizzazione, il *reverse proxy* e i container per l'invio sistematico dei *log* a un server centralizzato, adottando strumenti quali `rsyslog`, `syslog-ng` o agenti di *log* specifici per ciascuna applicazione.

- **Implementare una piattaforma di *log management*:** integrare l'infrastruttura con una piattaforma di *log management* evoluta, come l'ELK stack (Elasticsearch, Logstash, Kibana) o Graylog. Tali piattaforme mettono a disposizione funzionalità avanzate di indicizzazione, ricerca, analisi e visualizzazione dei *log*, semplificando in misura significativa l'identificazione e la risoluzione di problematiche operative e di sicurezza.
- **Definire *policy* di conservazione dei *log*:** stabilire *policy* di conservazione dei *log* coerenti con i requisiti di sicurezza, conformità normativa e analisi forense, oltre a configurare meccanismi di rotazione dei *log* per prevenire il consumo eccessivo di spazio disco.
- **Implementare monitoraggio proattivo e analisi predittiva:** adottare strumenti di analisi comportamentale e *machine learning* per rilevare anomalie nei *log* e anticipare potenziali problematiche operative o di sicurezza, oltre a configurare sistemi di *alerting* automatico in caso di eventi critici o sospetti.

L'implementazione di un sistema di monitoraggio centralizzato e di gestione avanzata dei *log* rappresenta un investimento strategico per migliorare sensibilmente la sicurezza, l'affidabilità e la gestibilità dell'infrastruttura nel lungo periodo.

5.3. Conclusioni Finali: Un'Infrastruttura Solida e Proiettata al Futuro

In conclusione, l'implementazione di una piattaforma di containerizzazione e di un *reverse proxy* ha delineato una solida e performante base tecnologica per qualsiasi infrastruttura IT moderna. L'architettura virtualizzata realizzata si distingue per efficienza, scalabilità, sicurezza e flessibilità operativa, rispondendo in modo efficace alle esigenze attuali e predisponendosi intrinsecamente per future evoluzioni e ampliamenti.

I futuri sviluppi si focalizzeranno sull'automazione avanzata, il monitoraggio centralizzato e proattivo, il miglioramento continuo della sicurezza, l'ottimizzazione delle *performance* e l'integrazione con sistemi *enterprise*, con l'obiettivo primario di garantire un ambiente IT progressivamente più efficiente, sicuro, scalabile e all'avanguardia, in grado di supportare in modo ottimale le attività operative e di sviluppo nel lungo termine.

Appendici

Appendice A: Dettagli sull'Installazione e Configurazione Iniziale di LXD

Questa appendice raccoglie:

- I comandi per l'aggiornamento del sistema e l'installazione di snapd.
- La procedura completa per installare LXD tramite Snap (`sudo snap install lxd`).
- Istruzioni dettagliate per la verifica dell'installazione (`lxd --version`).
- La procedura di inizializzazione con `sudo lxd init`, con spiegazione dei parametri (scelta del backend di storage, configurazione della rete, abilitazione dell'accesso remoto).

A.1. Prerequisiti

In fase preliminare all'installazione di LXD, si è provveduto ad assicurare l'aggiornamento del sistema operativo e l'installazione dei pacchetti software necessari per il corretto funzionamento di LXD. A tal fine, sono stati eseguiti i seguenti comandi tramite l'interfaccia a riga di comando del sistema operativo:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y snapd
```

Il comando `sudo apt update && sudo apt upgrade -y` garantisce l'aggiornamento dell'indice dei pacchetti software disponibili dai repository APT e l'aggiornamento all'ultima versione di tutti i pacchetti installati nel sistema, assicurando la presenza delle versioni più recenti delle librerie e dei componenti software. Successivamente, il comando `sudo apt install -y snapd` provvede all'installazione di `snapd`, il demone di gestione dei pacchetti Snap. L'utilizzo di Snap per la distribuzione di LXD offre vantaggi significativi in termini di gestione degli aggiornamenti e isolamento applicativo, come descritto nella sezione successiva.

A.2. Installazione di LXD tramite Snap

LXD è distribuito e installabile come pacchetto Snap, un sistema di packaging e distribuzione di applicazioni containerizzate. L'adozione del formato Snap per LXD presenta diversi vantaggi, tra cui:

- **Aggiornamenti automatici e regolari:** Snap garantisce l'erogazione di aggiornamenti software in modo automatico e frequente, assicurando che LXD sia sempre aggiornato all'ultima versione stabile, beneficiando delle nuove funzionalità e delle correzioni di sicurezza più recenti.
- **Isolamento applicativo:** I pacchetti Snap sono eseguiti in ambienti isolati (sandbox), riducendo il rischio di conflitti con altre librerie di sistema e aumentando la stabilità del sistema.
- **Rollback semplificato:** In caso di problemi a seguito di un aggiornamento, Snap permette di effettuare il rollback a versioni precedenti del pacchetto in modo semplice e rapido.

L'installazione di LXD tramite Snap è stata eseguita mediante il seguente comando:

```
sudo snap install lxd
```

Il comando `sudo snap install lxd` scarica e installa l'ultima versione stabile di LXD dal repository Snap Store, configurando automaticamente il sistema per l'esecuzione del demone LXD.

A.3. Verifica dell'installazione

Al fine di verificare la corretta installazione di LXD, è stato eseguito il comando:

```
lxd –version
```

Il comando `lxd --version` interroga il demone LXD e restituisce la versione installata. Sul sistema prototipale oggetto del presente rapporto tecnico, il comando ha restituito il seguente output:

```
5.21.2 LTS
```

Confermando l'installazione della versione 5.21.2 LTS (Long Term Support) di LXD, versione stabile e supportata a lungo termine, adatta ad ambienti di produzione e prototipali.

Appendice B: Configurazione della Rete con Netplan

Questa appendice presenta la configurazione di rete originale del prototipo HPC, realizzata utilizzando Netplan – un sistema dichiarativo basato su file YAML. La configurazione definisce:

- **Interfacce Ethernet:**
  - *eno8303:* Configurata con l'indirizzo IP statico 192.168.0.12/24, nameserver (8.8.8.8) e una rotta di default verso 192.168.0.1.
  - *eno8403:* Configurata con l'indirizzo IP statico 140.164.14.126/24, nameserver (8.8.8.8) e una rotta di default verso 140.164.14.1.
- **Interfaccia Aggregata (Bond):**
  - *bond0:* Un'interfaccia aggregata (LACP, MTU 9000) che unisce le interfacce fisiche enp129s0f0np0 ed enp129s0f1np1, con l'indirizzo IP statico 10.10.10.12/24.

La configurazione YAML originale è la seguente:

```
network:
  ethernets:
    eno8303:
      addresses:
        - 192.168.0.12/24
      nameservers:
        addresses:
          - 8.8.8.8
      routes:
        - to: default
          via: 192.168.0.1
    eno8403:
      addresses:
        - 140.164.14.126/24
      nameservers:
        addresses:
          - 8.8.8.8
      routes:
        - to: default
          via: 140.164.14.1
  bonds:
    bond0:
      addresses: [10.10.10.12/24]
      interfaces:
        - enp129s0f0np0
        - enp129s0f1np1
      mtu: 9000
      parameters:
        lacp-rate: fast
        mode: 802.3ad
        mii-monitor-interval: 100
version: 2
```

Questa configurazione ibrida consente di connettere il sistema a due reti IP distinte e di sfruttare l'aggregazione dei link per migliorare banda e ridondanza.

Appendice C: Configurazione del Bridge di Rete per LXD

Per predisporre l'ambiente di containerizzazione LXD, è stata introdotta la configurazione di un bridge di rete (br0) tramite Netplan. Tale bridge consente ai container di comunicare tra loro e con la rete fisica. Le modifiche principali comprendono:

- **Riorganizzazione delle interfacce Ethernet:**
  - *eno8303* è disabilitata per DHCP e contrassegnata come opzionale.
  - *eno8403* mantiene la configurazione statica originale.
  - Le interfacce enp129s0f0np0 ed enp129s0f1np1 sono ora gestite dall'interfaccia aggregata bond0.
- **Introduzione del Bridge br0:**
  - Il bridge include l'interfaccia eno8303 e viene configurato con due indirizzi IP statici:
    - 192.168.0.10/24 (per la rete locale)
    - 192.168.71.1/24 (dedicato alla sottorete dei container LXD)
  - Vengono specificati anche i nameserver (8.8.8.8 e 8.8.4.4).

Il file YAML aggiornato è il seguente:

```
network:
    ethernets:
        eno8303:
            dhcp4: no
            optional: true
        eno8403:
            addresses:
            - 140.164.14.126/24
            nameservers:
                addresses:
                - 8.8.8.8
                search: []
            routes:
            - to: default
              via: 140.164.14.1
        enp129s0f0np0: {}
        enp129s0f1np1: {}
    bridges:
        br0:
            interfaces: [eno8303]
            addresses:
            - 192.168.0.10/24  # Rete locale
            - 192.168.71.1/24  # Sottorete container
            dhcp4: no
            nameservers:
                addresses:
                - 8.8.8.8
                - 8.8.4.4
                search: []
    bonds:
        bond0:
          addresses: [10.10.10.10/24]
          interfaces:
```

```
            - enp129s0f0np0
            - enp129s0f1np1
          mtu: 9000
          parameters:
                lacp-rate: fast
                mode: 802.3ad
                mii-monitor-interval: 100
    version: 2
```

Successivamente, la configurazione viene applicata con il comando:

```
sudo netplan apply
```

La verifica dell'operatività del bridge avviene con il comando:

```
ip a show br0
```

che conferma l'assegnazione degli indirizzi IP e lo stato operativo "UP" del bridge, garantendo la corretta interconnessione tra le reti.

Appendice D: Installazione e Configurazione di LXD-UI

Questa appendice descrive in dettaglio il processo di installazione, configurazione e messa in sicurezza di LXD-UI, l'interfaccia web per la gestione degli ambienti containerizzati basati su LXD. Verranno illustrati i passaggi per l'installazione tramite Snap, la configurazione della porta di ascolto, l'abilitazione della comunicazione sicura tramite certificati SSL/TLS e l'importazione dei certificati nel browser. L'adozione di LXD-UI consente agli amministratori di monitorare e gestire i container in modo intuitivo e sicuro, grazie all'impiego di comunicazioni crittografate tramite certificati SSL/TLS.

Durante l'implementazione, assicurarsi di sostituire tutti i valori di esempio (ad esempio, indirizzi IP, nomi di dominio, password) con quelli effettivi specifici del proprio ambiente. La mancata sostituzione di questi valori potrebbe causare errori operativi o compromettere la sicurezza dell'infrastruttura.

D.1 Installazione di LXD-UI

LXD-UI viene distribuito come pacchetto Snap, garantendo isolamento applicativo e aggiornamenti automatici. L'installazione viene eseguita tramite un semplice comando, che scarica e configura l'ultima versione stabile dell'interfaccia web.

Per installare LXD-UI, utilizzare il seguente comando:

```
sudo snap install lxd-ui
```

Dopo l'installazione, è necessario configurare la porta di ascolto per accettare connessioni HTTPS e definire una politica Cross-Origin per la gestione delle richieste web.

- **La porta predefinita per l'interfaccia web è 8443.** Per configurare LXD-UI per accettare connessioni HTTPS sulla porta 8443, utilizzare il comando:

  ```
  sudo snap set lxd-ui ports.ui=8443
  ```

- **La politica di Cross-Origin (CORS) può essere configurata per consentire richieste da origini specifiche o, in ambienti di test, da qualsiasi origine.** Per consentire richieste da qualsiasi origine (da utilizzare solo in ambienti di test), utilizzare il comando:

```
sudo snap set lxd-ui allow-origin="*"
```

In ambienti di produzione, è fondamentale limitare questa impostazione ai soli domini autorizzati per motivi di sicurezza. Assicurarsi di sostituire "*" con il dominio o l'indirizzo IP effettivo del client autorizzato.

D.2. Generazione e Configurazione dei Certificati SSL/TLS

Per garantire connessioni sicure tra il browser e LXD-UI, è necessario generare e configurare certificati **SSL/TLS**. Il processo prevede:

1. **Creazione di un'Autorità di Certificazione (CA) autofirmata**, che fungerà da ente certificatore.
2. **Generazione del certificato server**, firmato dalla CA, per proteggere la comunicazione con LXD-UI.
3. **Conversione del certificato in formato PFX**, per consentirne l'importazione nel browser.

D.3.1 Creazione del Certificato Root

La prima fase consiste nella creazione di un certificato root autofirmato, utilizzato per firmare il certificato server. Questo certificato verrà poi importato nel browser per riconoscere come attendibile il certificato LXD-UI.

I dettagli operativi, inclusi i comandi per la generazione della chiave privata della CA e la creazione del certificato root, sono i seguenti:

1. **Generazione della chiave privata della CA radice:**

   ```
   openssl genrsa -out rootCA.key 4096
   ```

   Il comando `openssl genrsa` genera una chiave privata RSA (Rivest-Shamir-Adleman) con una lunghezza di 4096 bit e la salva nel file `rootCA.key`. La chiave privata è un componente crittografico fondamentale per la firma digitale dei certificati e deve essere conservata in modo sicuro e confidenziale. La lunghezza di 4096 bit rappresenta una robustezza crittografica elevata, adeguata per la protezione della CA radice.

2. **Creazione del certificato root autofirmato:**

   ```
   openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 3650 -
   out rootCA.crt -subj \\
   "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=LXD Root CA"
   ```

   Il comando `openssl req -x509` genera un certificato X.509 autofirmato. I parametri utilizzati sono i seguenti:

   - `-x509`: Specifica la generazione di un certificato X.509 autofirmato.
   - `-new`: Indica la creazione di una nuova richiesta di certificato (Certificate Signing Request - CSR) e di un certificato.
   - `-nodes`: Specifica di non crittografare la chiave privata (opzione non raccomandata in ambienti di produzione, ma accettabile in contesti di test).
   - `-key rootCA.key`: Specifica il file contenente la chiave privata della CA radice (`rootCA.key`) generata nel passaggio precedente.
   - `-sha256`: Specifica l'utilizzo dell'algoritmo di hash SHA-256 (Secure Hash Algorithm 256-bit) per la firma digitale del certificato. SHA-256 è un algoritmo di hash crittograficamente robusto, ampiamente utilizzato per la firma di certificati digitali.

- `-days 3650`: Specifica la validità del certificato in giorni (3650 giorni, equivalenti a 10 anni). La durata di validità di un certificato autofirmato può essere estesa, ma è buona prassi rinnovare i certificati periodicamente per ragioni di sicurezza e gestione.
- `-out rootCA.crt`: Specifica il file di output per il certificato root autofirmato (`rootCA.crt`). Il file `rootCA.crt` conterrà il certificato root in formato PEM (Privacy Enhanced Mail), un formato standard per la codifica di certificati digitali.
- `-subj "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=LXD Root CA"`: Specifica il Subject Distinguished Name (DN) del certificato root. Il Subject DN contiene informazioni identificative sull'entità (in questo caso, la CA radice LXD Root CA) a cui il certificato è rilasciato. I campi utilizzati nel Subject DN sono:
  - `C=IT`: Country Code (IT per Italia).
  - `ST=State`: State or Province (Stato/Regione).
  - `L=City`: Locality Name (Città).
  - `O=Organization`: Organization Name (Nome dell'organizzazione).
  - `OU=Unit`: Organizational Unit Name (Unità organizzativa).
  - `CN=LXD Root CA`: Common Name (Nome Comune), in questo caso "LXD Root CA", che identifica in modo univoco la CA radice.

D.3.2 Generazione del Certificato Server

Una volta creato il certificato root, viene generato il certificato server specifico per LXD-UI, utilizzando la CA autofirmata. Questo certificato garantirà una connessione sicura tra client e server.

Il certificato server deve includere:

- Il nome di dominio o l'indirizzo IP del server LXD-UI.
- Un'estensione Subject Alternative Name (SAN) per evitare errori di validazione nei browser moderni.

La procedura di generazione del certificato server si articola nei seguenti passaggi:

1. **Creazione della chiave privata del server:**

   ```
   openssl genrsa -out server.key 2048
   ```

   Il comando `openssl genrsa` genera una chiave privata RSA con una lunghezza di 2048 bit e la salva nel file `server.key`. La chiave privata del server è utilizzata per la cifratura delle comunicazioni HTTPS e per l'autenticazione del server LXD-UI. La lunghezza di 2048 bit rappresenta un buon compromesso tra sicurezza e prestazioni per le chiavi server.

2. **Generazione della richiesta di firma del certificato (CSR - Certificate Signing Request):**

   ```
   openssl req -new -key server.key -out server.csr \
   -subj "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=lxd.local"
   ```

   Il comando `openssl req -new` genera una richiesta di firma del certificato (CSR). La CSR contiene informazioni sul server (Subject DN) e la chiave pubblica del server, e viene inviata alla CA (in questo caso, la CA radice autofirmata) per la firma digitale. I parametri utilizzati sono:

   - `-new`: Indica la creazione di una nuova CSR.
   - `-key server.key`: Specifica il file contenente la chiave privata del server (`server.key`) generata nel passaggio precedente.
   - `-out server.csr`: Specifica il file di output per la CSR (`server.csr`). Il file `server.csr` conterrà la CSR in formato PEM.

- `-subj "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=lxd.local"`: Sostituire lxd.local con il nome di dominio o l'indirizzo IP effettivo del server LXD-UI. Il valore del campo CN deve corrispondere al nome o all'indirizzo utilizzato per accedere all'interfaccia web.

3. **Creazione del file di configurazione per le estensioni SSL:**

```
cat > server.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,\
dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = lxd.local
DNS.2 = localhost
IP.1 = 192.168.0.10 #Sostituire con l'IP effettivo del server LXD-UI
EOF
```

Assicurarsi di sostituire `192.168.0.10` con l'indirizzo IP reale del server LXD-UI. L'inclusione di SANs errati o incompleti può causare errori di validazione del certificato nei browser moderni.

Questo blocco di codice crea un file di configurazione denominato `server.ext` che definisce le estensioni X.509 per il certificato server. Le estensioni X.509 permettono di aggiungere informazioni supplementari al certificato, specificando vincoli d'uso e nomi alternativi del soggetto (Subject Alternative Names - SANs). Il file `server.ext` contiene le seguenti sezioni:

- `authorityKeyIdentifier=keyid,issuer`: Include l'identificativo della chiave pubblica e l'identità dell'emittente (la CA radice) nel certificato server.
- `basicConstraints=CA:FALSE`: Specifica che il certificato server non è un certificato di CA e non può essere utilizzato per firmare altri certificati.
- `keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment`: Definisce gli usi consentiti per la chiave pubblica del certificato server. In questo caso, la chiave può essere utilizzata per la firma digitale, il non ripudio, la cifratura della chiave e la cifratura dei dati.
- `subjectAltName = @alt_names`: Abilita l'estensione Subject Alternative Name (SAN), che permette di specificare nomi di dominio alternativi e indirizzi IP validi per il certificato server.
- `[alt_names]`: Definisce la sezione contenente i nomi alternativi del soggetto.
  - `DNS.1 = lxd.local`: Specifica `lxd.local` come nome di dominio alternativo valido per il certificato server.
  - `DNS.2 = localhost`: Specifica `localhost` come nome di dominio alternativo valido.
  - `IP.1 = 192.168.0.10`: Specifica l'indirizzo IP `192.168.0.10` come indirizzo IP alternativo valido. È necessario sostituire `192.168.0.10` con l'indirizzo IP effettivo del server LXD-UI. L'inclusione di SANs è fondamentale per evitare errori di validazione del certificato da parte del browser quando si accede all'interfaccia LXD-UI utilizzando nomi di dominio diversi dal CN o indirizzi IP.

4. **Firma digitale del certificato server:**

```
openssl x509 -req -in server.csr -CA rootCA.crt -CAkey rootCA.key \
-CAcreateserial -out server.crt -days 365 -sha256 -extfile server.ext
```

Il comando `openssl x509 -req` firma digitalmente la CSR del server (`server.csr`) utilizzando il certificato e la chiave privata della CA radice (`rootCA.crt` e `rootCA.key`). I parametri utilizzati sono:

- `-req`: Indica l'operazione di firma di una CSR.
- `-in server.csr`: Specifica il file contenente la CSR del server (`server.csr`).
- `-CA rootCA.crt`: Specifica il file contenente il certificato della CA radice (`rootCA.crt`).
- `-CAkey rootCA.key`: Specifica il file contenente la chiave privata della CA radice (`rootCA.key`).
- `-CAcreateserial`: Indica di creare un file seriale per la CA radice (necessario per la gestione dei numeri seriali dei certificati).
- `-out server.crt`: Specifica il file di output per il certificato server firmato (`server.crt`). Il file `server.crt` conterrà il certificato server firmato in formato PEM.
- `-days 365`: Specifica la validità del certificato server in giorni (365 giorni, equivalenti a 1 anno). La durata di validità del certificato server è inferiore rispetto al certificato root, in quanto i certificati server vengono tipicamente rinnovati più frequentemente per ragioni di sicurezza.
- `-sha256`: Specifica l'utilizzo dell'algoritmo di hash SHA-256 per la firma digitale del certificato server.
- `-extfile server.ext`: Specifica il file di configurazione delle estensioni X.509 (`server.ext`) creato nel passaggio precedente.

D.4 Configurazione del Browser Web

Dopo aver generato il certificato, è necessario importarlo nel browser per evitare avvisi di sicurezza durante l'accesso a LXD-UI.

D.4.1 Creazione del File PFX

Il formato PFX (Personal eXchange Format) permette di raggruppare il certificato server, la chiave privata e il certificato root in un unico file. Questo file sarà importato nel browser per garantire il riconoscimento della connessione HTTPS senza errori.

La conversione in formato PFX è stata eseguita tramite il comando `openssl pkcs12`:

```
openssl pkcs12 -export -out certificate.pfx -inkey server.key \
-in server.crt -certfile rootCA.crt -passout pass:YourSecurePassword
```

Sostituire YourSecurePassword con una password robusta e conservarla in modo sicuro. La password protegge il file PFX da accessi non autorizzati.

Il comando `openssl pkcs12 -export` converte i certificati in formato PFX. I parametri utilizzati sono:

- `-export`: Specifica l'operazione di esportazione in formato PFX.
- `-out certificate.pfx`: Specifica il file di output per il file PFX (`certificate.pfx`). Il file `certificate.pfx` conterrà il certificato server, la chiave privata e il certificato root in formato PFX.
- `-inkey server.key`: Specifica il file contenente la chiave privata del server (`server.key`).
- `-in server.crt`: Specifica il file contenente il certificato server firmato (`server.crt`).
- `-certfile rootCA.crt`: Specifica il file contenente il certificato root della CA radice (`rootCA.crt`). L'inclusione del certificato root nel file PFX permette al browser di validare l'intera catena di certificati.

- `-passout pass:YourSecurePassword`: Specifica la password di protezione per il file PFX. È fondamentale sostituire `YourSecurePassword` con una password robusta e conservarla in modo sicuro. La password protegge il file PFX da accessi non autorizzati e ne impedisce l'utilizzo improprio.

D.4.2 Importazione del Certificato nel Browser

L'importazione del certificato nel browser segue questi passaggi generali:

1. Aprire le impostazioni del browser e accedere alla gestione dei certificati. Tipicamente, le impostazioni del browser sono accessibili tramite il menu principale del browser (es. "Impostazioni" o "Preferenze").
2. Navigare alla sezione "Privacy e Sicurezza" > "Certificati" (o sezioni equivalenti). La sezione relativa alla gestione dei certificati è solitamente collocata all'interno delle impostazioni di privacy e sicurezza del browser.
3. Selezionare il pulsante "Importa certificato" (o dicitura equivalente). Il pulsante "Importa certificato" avvia la procedura guidata di importazione dei certificati.
4. Scegliere il file `certificate.pfx` generato nel passaggio precedente. Nella finestra di dialogo di importazione, selezionare il file `certificate.pfx` precedentemente creato.
5. Inserire la password definita durante la creazione del file PFX. Il browser richiederà l'inserimento della password di protezione specificata durante la creazione del file PFX. Inserire la password corretta per procedere con l'importazione.
6. Confermare l'importazione e le eventuali autorizzazioni richieste. Il browser potrebbe richiedere la conferma dell'importazione del certificato e l'assegnazione di eventuali autorizzazioni (es. autorizzazione ad utilizzare il certificato per l'autenticazione client). Confermare le richieste per completare l'importazione del certificato.

D.5 Verifica dell'Accesso Sicuro

Per verificare che la configurazione sia stata eseguita correttamente:

1. Aprire il browser e navigare all'indirizzo **https://[IP_SERVER]:8443**.
2. Verificare la presenza del lucchetto nella barra degli indirizzi, segno che la connessione è crittografata.
3. Accedere all'interfaccia di login e controllare che non vengano mostrati avvisi di certificato non valido.

D.6 Comandi Dettagliati per l'Implementazione

Di seguito vengono riportati tutti i comandi necessari per l'installazione e configurazione di LXD-UI, inclusa la gestione dei certificati SSL/TLS.

D.6.1 Installazione e Configurazione Iniziale di LXD-UI

- **Installazione di LXD-UI tramite Snap:**

```
sudo snap install lxd-ui
```

- **Configurazione della porta di ascolto e della politica CORS:**

```
sudo snap set lxd-ui ports.ui=8443
sudo snap set lxd-ui allow-origin="*" # Da limitare in produzione!
```

D.6.2 Generazione e Configurazione dei Certificati SSL/TLS

- **Creazione del certificato root della CA:**

```
openssl genrsa -out rootCA.key 4096
openssl req -x509 -new -nodes -key rootCA.key -sha256 -days 3650 -
out rootCA.crt \
-subj "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=LXD Root CA"
```

- **Generazione del certificato server e aggiunta delle estensioni X.509:**

```
openssl genrsa -out server.key 2048
openssl req -new -key server.key -out server.csr \
  -subj "/C=IT/ST=State/L=City/O=Organization/OU=Unit/CN=lxd.local"
cat > server.ext << EOF
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment,
dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = lxd.local
DNS.2 = localhost
IP.1 = 192.168.0.10 # Sostituire con IP effettivo del server LXD-UI
EOF
```

- **Firma del certificato server con la CA radice:**

```
openssl x509 -req -in server.csr -CA rootCA.crt -CAkey rootCA.key \
-CAcreateserial -out server.crt -days 365 -sha256 -extfile server.ext
```

D.6.3 Conversione in Formato PFX e Importazione nel Browser

- **Creazione del file PFX contenente certificato, chiave privata e CA root:**

```
openssl pkcs12 -export -out certificate.pfx -inkey server.key \
-in server.crt -certfile rootCA.crt -passout pass:YourSecurePassword
\ # Sostituire YourSecurePassword con una password sicura
```

- **Procedura di importazione nei principali browser web:** (Varia a seconda del browser, seguire i passaggi generali descritti in D.4.2).

D.7 Best Practices per la Sicurezza di LXD-UI

Per garantire la massima sicurezza nell'utilizzo di LXD-UI, si raccomanda di adottare le seguenti *best practices*:

- **Proteggere le chiavi private e i certificati**, limitandone l'accesso agli amministratori autorizzati e conservandoli in posizioni sicure sul sistema. La compromissione delle chiavi private comprometterebbe irrimediabilmente la sicurezza dell'infrastruttura PKI.
- **Rinnovare periodicamente i certificati** (sia il certificato server che il certificato root) prima della loro scadenza per evitare interruzioni di servizio e problemi di sicurezza. Implementare procedure automatizzate per il rinnovo periodico e monitorare attivamente la data di scadenza dei certificati.
- **Configurare regole di firewall** per limitare l'accesso all'interfaccia web LXD-UI (porta TCP 8443) esclusivamente agli indirizzi IP o alle reti autorizzate. Questa misura riduce significativamente la superficie di attacco e il rischio di accessi non autorizzati.
- **Mantenere LXD-UI sempre aggiornato**, applicando tempestivamente le ultime patch di sicurezza rilasciate dagli sviluppatori per mitigare le vulnerabilità note.

- **Adottare password complesse e robuste** per proteggere il file PFX (`certificate.pfx`). Questo aggiunge un ulteriore livello di difesa in caso di furto o accesso non autorizzato al file PFX, rendendo più difficile l'estrazione delle chiavi private da parte di un attaccante.

Tabella degli Acronimi

Acronimo	Descrizione
APT	Advanced Package Tool, strumento di gestione dei pacchetti su sistemi Debian e derivate
CA	Certificate Authority (Autorità di Certificazione)
CN	Common Name (Nome Comune)
CORS	Cross-Origin Resource Sharing
CSR	Certificate Signing Request (Richiesta di Firma del Certificato)
DDoS	Distributed Denial of Service
DN	Distinguished Name
DNS	Domain Name System
DoS	Denial of Service
ELK	Elasticsearch, Logstash, Kibana
FOSSR	Fostering Open Science in Social Science Research
HPC	High Performance Computing (Calcolo ad Alte Prestazioni)
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
ICAR	Istituto di Calcolo e Reti ad Alte Prestazioni
IP	Internet Protocol
IT	Information Technology
LACP	Link Aggregation Control Protocol
LVM	Logical Volume Manager
LTS	Long Term Support
LXD	Linux Container Daemon
LXD-UI	Interfaccia di Gestione Web Sicura per LXD
NAT	Network Address Translation
OU	Organizational Unit (Unità Organizzativa)
PFX	Personal eXchange Format
PKI	Public Key Infrastructure
RSA	Rivest-Shamir-Adleman
SAN	Subject Alternative Name
SHA-256	Secure Hash Algorithm 256-bit
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	State or Province (Stato/Regione)
TLS	Transport Layer Security
URL	Uniform Resource Locator
VLAN	Virtual LAN
VM	Virtual Machine
WAF	Web Application Firewall
YAML	YAML Ain't Markup Language
ZFS	Zettabyte File System