Virtualizzazione e tecnologie cloud

Corso di formazione per neoassunti nelle attività di computing

Emidio Giorgio – INFN LNS

Virtualizzazione: cosa è?

- La virtualizzazione è un insieme di tecnologie che a partire da componenti fisiche ne astrae entità logiche per massimizzare l'utilizzo delle risorse.
- Esistono diversi tipi di virtualizzazione:
  - **Host**: permette l'emulazione di una macchina fisica;
  - **Risorse (CPU, RAM, Dischi)**: le singole risorse a disposizione in uno o più server vengono aggregate e distribuite ai singoli utenti a seconda delle necessità
  - **Rete**: adoperando la medesima rete fisica è possibile separare logicamente le risorse connesse al fine di ottimizzare la rete e migliorarne la sicurezza
- Partiamo da alcuni concetti =>

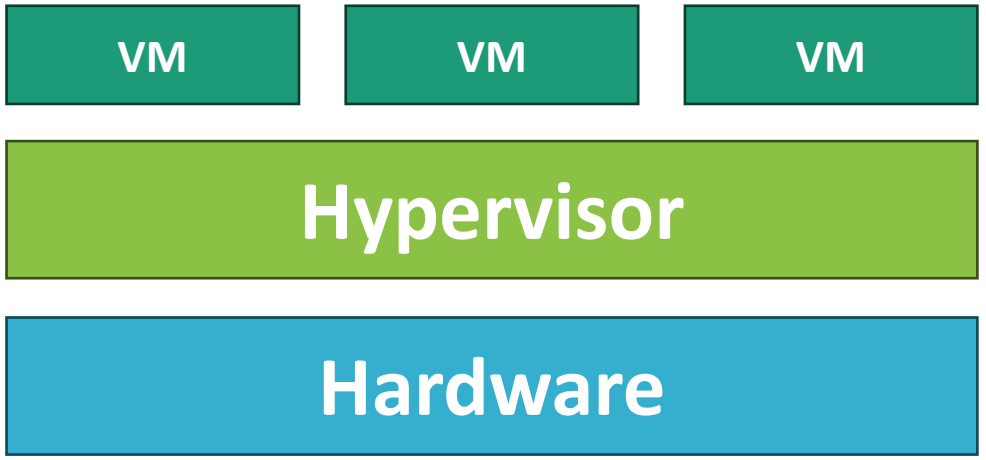
Cosa sono le VM?

- Una macchina virtuale (VM) è l’insieme delle risorse virtuali, astratte da quelle fisiche, e il software che su di esse gira.
- Su ogni VM è installato un sistema operativo (OS) detto «Guest OS», similmente alle macchine fisiche.
- Dal punto di vista del Guest OS, e quindi dell'utente finale, la VM appare identica (o quasi) ad una macchina fisica

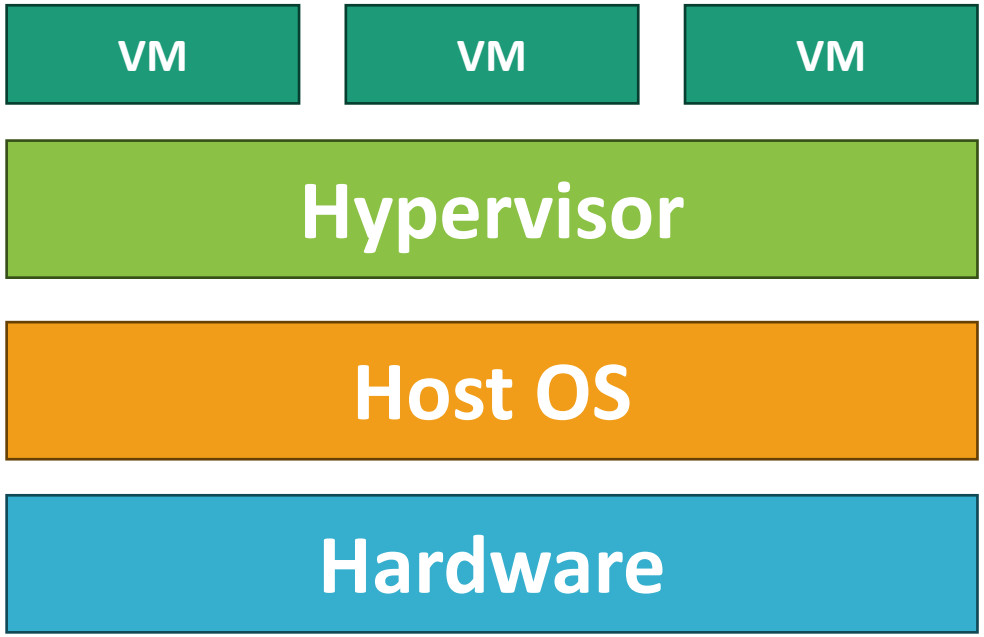
Cosa sono gli Hypervisors

- Sono elementi fondamentali della virtualizzazione:
  - gestiscono il layer di astrazione tra l'hardware e le risorse virtuali usate dalle VM
- Sono quindi piattaforme su cui far girare le VM
- Sono composti da:
  - Kernel: si comporta come il kernel del SO in quanto dialoga con l’hardware e gestisce l’utilizzo delle risorse
  - Virtual Machine Manager

Tipi di Hypervisors



Tipo I: «Bare Metal» : eseguito direttamente sull'hardware, gestione diretta delle risorse. Offre prestazioni più elevate.



Tipo II: «Hosted» : eseguito come un'applicazione, il SO ospite gestisce le risorse. Maggiore overhead, migliore scalabilità e gestione

Virtualizzazione: pro e contro

- Ottimizzare l’utilizzo delle risorse, riducendo i costi di mantenimento e gestione
- Creare ambienti di test e sviluppo in modo rapido e sicuro
- Migliorare la continuità dei servizi, grazie alla possibilità di eseguire il ripristino di server in modo rapido.

- Comporta un aumento della complessità dell'ambiente IT
- Maggiore attenzione nella gestione e nella configurazione dei server virtuali

Vantaggi: Ottimizzazione e continuità di servizio

Le risorse fisiche di diverse macchine (cluster) possono essere radunate in un unico pool dal quale le risorse virtuali vengono allocate

Poiché diverse macchine virtuali condividono le risorse dello stesso server, l'utilizzo delle risorse viene ottimizzato.

Le VM possono girare indifferentemente su uno qualunque dei nodi del cluster: **manutenzioni hardware facilitate, migliore resilienza**

Sfruttando l'over-commitment, gli **hypervisor possono assegnare più risorse di quelle realmente disponibili**, permettendo così di eseguire un numero maggiore di VM

Questa gestione dinamica, mediante algoritmi di bilanciamento delle risorse, permette l'accesso alle risorse necessarie in modo efficiente, evitando sprechi

Vantaggi: Scalabilità

- Per scalabilità, si intende la capacità dell’hypervisor di gestire un carico crescente, aggiungendo risorse in modo proporzionato

- **Vertical vs. Horizontal scaling**: la scalabilità verticale si riferisce all'aggiunta di più risorse a una singola VM (ad es. più CPU o disco), mentre la scalabilità orizzontale implica l'aggiunta di più nodi sullo stesso pool di risorse.

Vantaggi: Sicurezza

- Un servizio per ogni VM, no need to overload
- Isolamento: Ogni VM opera in modo indipendente, pertanto i problemi in una VM raramente influenzano le altre. Ad esempio, se una VM è compromessa, la minaccia rimane confinata e non si propaga facilmente alle altre VM o all'host fisico.
- Testbed: le VM offrono un ambiente di test sicuro ed isolato, totalmente indipendente dall’ambiente di produzione, che però può essere facilmente replicato

Vantaggi: Disaster Recovery

- Nell’eventualità di un «disastro» (perdita di dati, interruzione di corrente, ecc.) inaspettato, la virtualizzazione permette strategie di recupero più flessibili e veloci
- In questo modo è possibile ridurre o azzerare i tempi di assenza di servizio
- Le **VM** sono infatti **indipendenti dall’hardware:** possono essere **spostate**, **copiate** o **replicate** tra vari server fisici, indipendentemente dall'hardware sottostante
- Si possono definire strategie di backup e ripristino avanzate (automatizzazioni, backup off-site, deduplicazione, ridondanza …) così che una copia aggiornata sia sempre disponibile per il ripristino
  - INFN BC

Tecnologie di virtualizzazione

- Esistono diverse tecnologie di virtualizzazione in base all'oggetto fisico che si vuole astrarre:

  - **Virtualizzazione hardware**: un insieme di risorse (CPU, RAM, Dischi, ecc.) appartenenti a un singolo server fisico viene astratto (emulato) e diviso tra più server virtuali, ognuno dei quali può eseguire il proprio sistema operativo e applicazioni.

  - **Reti**: Questa forma di virtualizzazione permette di creare versioni virtuali delle risorse di rete, come switch, router, firewall, e così via. Permette di gestire e configurare la rete a livello di software, indipendentemente dall'hardware sottostante.

Tecnologie di virtualizzazione

- Esistono diverse tecnologie di virtualizzazione in base all'oggetto fisico che si vuole astrarre:

  - **Storage**: diverse risorse di storage fisiche vengono riunite in uno o più pool che possono essere gestiti e allocati in modo flessibile. Le tecnologie di storage virtualizzato includono i sistemi di storage area network (SAN), il network attached storage (NAS) e il software defined storage (SDS).

  - **Applicazioni**: permettono di eseguire applicazioni in ambienti virtuali isolati dal sistema operativo sottostante. Sono sempre più diffuse poiché migliorano la portabilità delle applicazioni e semplificano la gestione delle dipendenze del software.

Virtualizzazione hardware

- Negli ultimi decenni sono state sviluppate diverse tecnologie di virtualizzazione hardware dai principali produttori di componenti
- CPU con Estensioni di Virtualizzazione: sia Intel (VT-x) che AMD (AMD-V) hanno introdotto nelle loro CPU un set di estensioni che permettono di ridurre l'overhead nelle VM eseguendo direttamente le istruzioni
- Virtualizzazione chipset ed I/O (IOMMU): permette alle VM di accedere direttamente alle periferiche connesse alla macchina fisica, migliorando le prestazioni e la sicurezza (AMD-Vi, Intel VT-d)
- Single Root I/O Virtualization (SR-IOV): permette di condividere un singolo bus PCIe tra più VM. La periferica fisica connessa viene astratta in più periferiche nell'hypervisor, le VM hanno così l' «illusione» di avere un dispositivo dedicato.

Virtualizzazione reti

- Grazie alla virtualizzazione è possibile collegare, separare o replicare le diverse risorse di rete da un punto di vista logico, lasciando inalterata l'infrastruttura fisica. Questo rende la rete più flessibile, scalabile ed efficiente, a parità di costi.

- **Bridging**
- **Overlay**

Bridging e Virtualizzazione

In genere un *bridge* permette di collegare due reti (o segmenti di rete), facendo sembrare queste come un'unica rete (o unico segmento).

Nel contesto della virtualizzazione si parla di Virtual Network Bridging:

- Quando si crea una VM solitamente essa è isolata dalla rete esterna o dalle altre VM.
- Grazie a un vBridge, creato sull'host e collegato all'interfaccia di rete fisica, è possibile connettere le VM tra di loro e/o con la rete esterna;
- *Una VM può quindi inviare pacchetti sulla rete fisica per mezzo dell'interfaccia dell'host ma operando come se fosse un'entità separata all'interno dello stesso dominio di broadcast*

Overlay Networks e Virtualizzazione

Le overlay networks sono reti che giacciono «sopra» altre reti, utilizzandone la medesima infrastruttura. La connessione tra nodi avviene per mezzo di tunnel virtuali, permettendo così ai protocolli di rete di operare come se i nodi fossero connessi direttamente.

Un esempio sono le reti VoIP (SIP) che sfruttano la connessione Internet (TCP/IP) preesistente per trasmettere dati restandone però separati.

Le overlay networks sono spesso usate per aggiungere funzionalità a reti esistenti, creare reti virtuali, gestire connessioni complesse in ambienti cloud.

Overlay Networks e Virtualizzazione

Vantaggi:

- permettono la creazione di reti virtuali senza dover modificare o riconfigurare la rete fisica esistente.
- la topologia della rete virtuale è indipendente e può quindi essere completamente differente dalla rete fisica sottostante.
- possono estendersi oltre le limitazioni delle reti fisiche tradizionali (connettere data center distribuiti)
- ogni rete overlay opera come una rete separata e isolata, garantendo sicurezza e segmentazione del traffico (ideale per separare il traffico in ambienti cloud)

Overlay Networks e Virtualizzazione

Alcuni esempi di tecnologie:

- VLAN/VXLAN: tecniche utilizzate per segmentare una rete fisica in diverse reti logiche, migliorando la gestione del traffico e la sicurezza.
- GRE (Generic Routing Encapsulation): protocollo proprietario (Cisco) di tunneling, usato per la creazione agevole di VPN
- OTV (Overlay Transport Virtualization): tecnologia proprietaria (Cisco), usata principalmente per collegare tra loro le LAN di diversi data center geograficamente distribuiti. *Le diverse reti appaiono come una singola rete di livello 2.*

Virtualizzazione storage

- Consiste nell'astrazione delle risorse di storage fisico in risorse virtuale, con i quali utenti ed applicazioni interagiscono, al fine di semplificare la gestione.
- I benefici fondamentali sono infatti: maggiore flessibilità, gestibilità, sicurezza, maggiori prestazioni ed ottimizzazione delle risorse fisiche.
- Flessibilità e scalabilità: è possibile modificare in tempo reale le risorse assegnate ad un utente in base alle reali necessità, senza interruzioni di servizio
- Ottimizzazione: grazie alla possibilità di allocare risorse in modo dinamico, si occupano le risorse realmente necessarie

Virtualizzazione storage

- Miglioramento delle Prestazioni: Bilanciamento del carico e caching intelligente
- Facilita' di gestione: tutte le risorse a disposizione sono gestite e monitorate da un’unica interfaccia
- Sicurezza: backup e ripristino avanzati, migrazione dei dati senza interruzioni

Virtualizzazione storage: NAS vs SAN

- Un network attached storage (NAS) è un dispositivo che permette, per mezzo di diversi protocolli, la condivisione di uno spazio disco all'interno della rete.
- I NAS operano a livello di filesystem, esportando porzioni di spazio disco che viene condiviso tra i diversi client
- Storage area network (SAN) è una rete che permette a più client di accedere a blocchi di storage gestiti in modo centralizzato, solitamente esposti ai client come dischi locali.
- A differenza del NAS, che opera a livello di file, i SAN operano a livello di blocco, rendendoli più veloci e adatti a applicazioni come database e virtualizzazione.

Virtualizzazione storage: NAS vs SAN

- Un NAS è generalmente votato alla condivisione di file, ed è molto più economico rispetto una SAN
- Un NAS permette una migliore gestione di accessi e permessi
- Le SAN hanno una migliore scalabilità, arrivando a supportare elevate quantità di dati
- Spesso includono sistemi per migliorare la sicurezza e per garantire alta disponibilità, come il failover automatico e la ridondanza multipath

Principali implementazioni VM manager

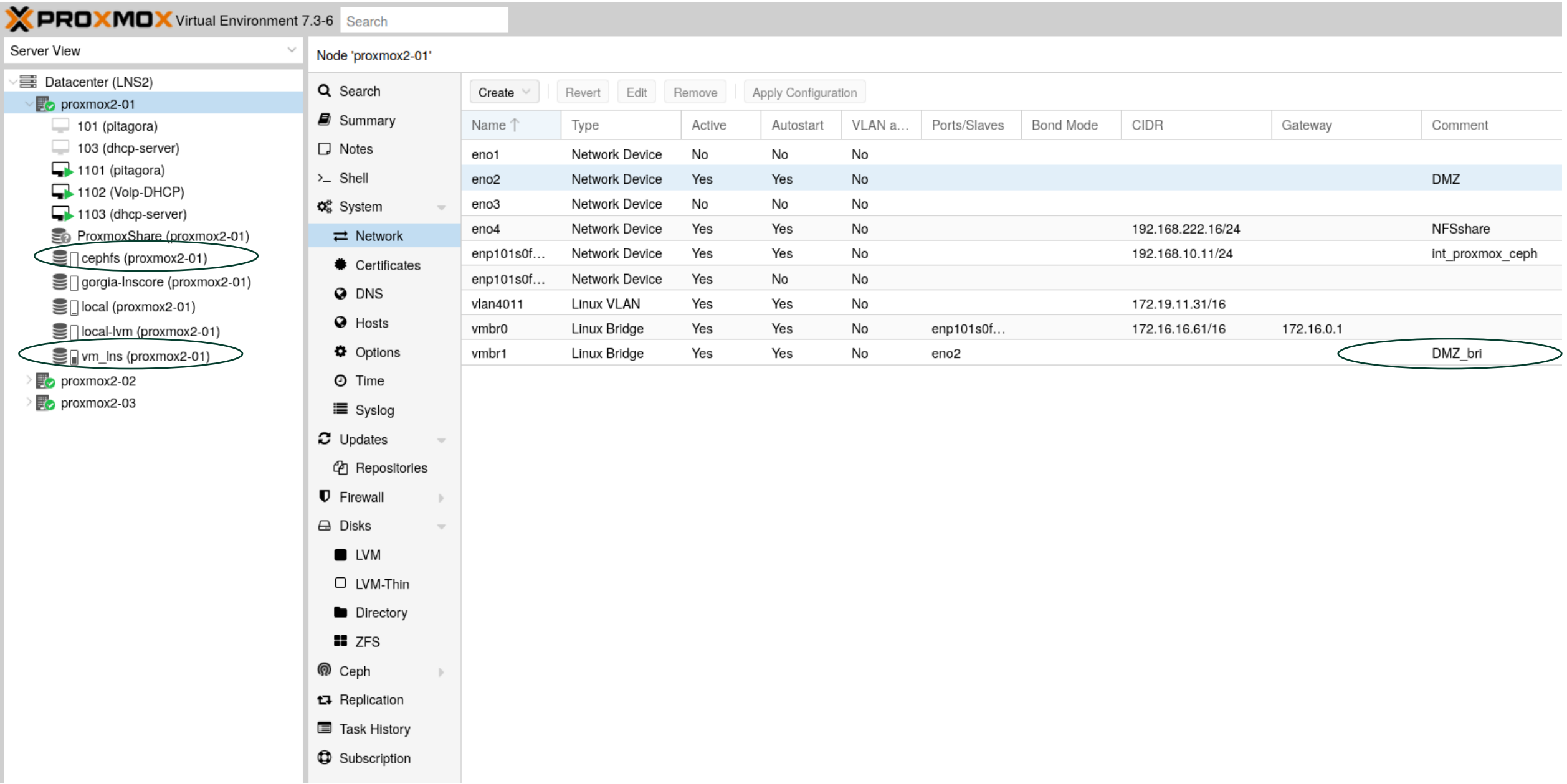
- **KVM** è una soluzione di virtualizzazione già integrata nel kernel Linux e di fatto lo rende un hypervisor, in quanto si possono eseguire VM direttamente su Linux, come se fossero processi del kernel
- KVM si occupa della virtualizzazione del processore, supportato dalle tecnologie come Intel VT-x e AMD-v, ed espone un'interfaccia (*/dev/kvm*) che viene usata da altri software per gestire la comunicazione tra VM e Host
- Di default la parte userspace è delegata a QEMU che si occupa di virtualizzare il resto dell'hardware (dispositivi I/O, interfacce di rete, ecc.) e configurare e gestire le VM.

Principali implementazioni

- **Proxmox VE:** piattaforma di virtualizzazione **open-source** basata su Debian
- Combina in un'unica soluzione un hypervisor bare-metal (KVM), Linux Containers (LXC) →
- Offre, in un'unica interfaccia web, numerose funzioni come:
  - gestioni reti
  - software-defined storage (CEPH)
  - high availability: permette di unire in cluster le risorse e, nel caso di problemi a un host, spostare «a caldo» le istanze tra i nodi del cluster
  - disaster recovery: implementa soluzioni di backup e ripristino, anche off-site

Principali implementazioni

- **VMware vSphere**: suite di virtualizzazione, permette la gestione di server, storage e rete.
  - **ESXi**: hypervisor bare-metal, è responsabile per la gestione delle risorse e l'esecuzione delle macchine virtuali (VM).
  - **vCenter Server**: servizio di gestione centralizzata, permette di lavorare con molteplici host ESXi e relative VM, offrendo diverse funzionalità:
    - vMotion & Storage vMotion: Migrazione in tempo reale di VM e storage.
    - HA & DRS: Alta disponibilità e bilanciamento dinamico delle risorse.
  - **vSphere Client**: interfaccia utente che consente di operare o a livello di vCenter Server o direttamente su un host ESXi


PROXMOX Virtual Environment 7.3-6
Search
Server View
Node 'proxmox2-01'
Datacenter (LNS2)
proxmox2-01
101 (pitagora)
103 (dhcp-server)
1101 (pitagora)
1102 (Voip-DHCP)
1103 (dhcp-server)
ProxmoxShare (proxmox2-01)
cephfs (proxmox2-01)
gorgia-lnscore (proxmox2-01)
local (proxmox2-01)
local-lvm (proxmox2-01)
vm_lns (proxmox2-01)
proxmox2-02
proxmox2-03
Search
Summary
Notes
Shell
System
Network
Certificates
DNS
Hosts
Options
Time
Syslog
Updates
Repositories
Firewall
Disks
LVM
LVM-Thin
Directory
ZFS
Ceph
Replication
Task History
Subscription
Create
Revert
Edit
Remove
Apply Configuration
Name
Type
Active
Autostart
VLAN a…
Ports/Slaves
Bond Mode
CIDR
Gateway
Comment
eno1 Network Device No No No
eno2 Network Device Yes Yes No DMZ
eno3 Network Device No No No
eno4 Network Device Yes Yes No 192.168.222.16/24 NFSshare
enp101s0f… Network Device Yes Yes No 192.168.10.11/24 int_proxmox_ceph
enp101s0f… Network Device Yes No No
vlan4011 Linux VLAN Yes Yes No 172.19.11.31/16
vmbr0 Linux Bridge Yes Yes No enp101s0f… 172.16.16.61/16 172.16.0.1
vmbr1 Linux Bridge Yes Yes No eno2 DMZ_bri

Container

- Partendo dal concetto di virtualizzazione delle applicazioni, i container implementano una virtualizzazione *ibrida*
- Il kernel del SO è in grado di gestire la coesistenza di diverse istanze dello spazio utente
- I container sono quindi **unità monolitiche** di software, contenenti tutti i file necessari all'esecuzione dello stesso (codice, file utente, librerie, moduli...)
- I container sono **immutabili**: le modifiche sono volatili, e perse ad una nuova esecuzione
  - Per rendere persistenti le modifiche si aggiornano le **immagini** da cui è generato il container

Container

- I container sono **portabili**: contenendo tutto ciò di cui hanno bisogno, i container possono essere eseguiti in qualsiasi ambiente che li supporti
- L’unione di immutabilità e portabilità comporta che i container avranno lo **stesso comportamento su qualunque macchina** in cui vengono eseguiti
- I container sono **leggeri**: poiché usano lo stesso kernel dell'host, richiedono molte meno risorse rispetto ad una VM
- Sono la soluzione ideale per eseguire applicazioni in scenari in cui la scalabilità e la portabilità sono essenziali (esecuzione di **microservizi**)

```
alexoliva@vnode-0:~$ sudo docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
alexoliva@vnode-0:~$
```

```
alexoliva@vnode-0:~$ sudo docker images ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
alexoliva@vnode-0:~$
```

In questo momento nella nostra macchina non ci sono né **container**…

…né **immagini**

A questo punto eseguiamo il container con il comando

docker run hello-world

Il container viene eseguito e stampa un messaggio sul terminale

Se controlliamo adesso la lista dei container vedremo che…

```
alexoliva@vnode-0:~$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/

alexoliva@vnode-0:~$ sudo docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
3f82f014a059   hello-world   "/hello"   20 seconds ago   Exited (0) 9 seconds ago              magical_bohr
alexoliva@vnode-0:~$
```

```
alexoliva@vnode-0:~$ sudo docker container ls -a
CONTAINER ID   IMAGE         COMMAND    CREATED          STATUS                      PORTS     NAMES
3f82f014a059   hello-world   "/hello"   20 seconds ago   Exited (0) 9 seconds ago              magical_bohr
alexoliva@vnode-0:~$
alexoliva@vnode-0:~$ sudo docker rm 3f82f014a059
3f82f014a059
alexoliva@vnode-0:~$ sudo docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
alexoliva@vnode-0:~$
```

Supponiamo che adesso il nostro container non ci serva più, possiamo rimuoverlo con

docker rm *CONTAINER ID*

Elencando i container nella macchina la lista è adesso vuota

Se però stampiamo la lista delle immagini locali troveremo ancora «hello-world» che abbiamo usato per eseguire il container

Se vogliamo rimuovere anche l'immagine dobbiamo usare il comando

docker rmi *IMAGE ID*

```
alexoliva@vnode-0:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
hello-world   latest    9c7a54a9a43c   5 months ago   13.3kB
alexoliva@vnode-0:~$ sudo docker rmi 9c7a54a9a43c
Untagged: hello-world:latest
Untagged: hello-world@sha256:4f53e2564790c8e7856ec08e384732aa38dc43c52f02952483e3f003afbf23db
Deleted: sha256:9c7a54a9a43cca047013b82af109fe963fde787f63f9e016fdc3384500c2823d
Deleted: sha256:01bb4fce3eb1b56b05adf99504dafd31907a5aadac736e36b27595c8b92f07f1
alexoliva@vnode-0:~$
```

```
alexoliva@vnode-0:~$ sudo docker images ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
alexoliva@vnode-0:~$
```

Solo adesso la lista delle immagini sarà vuota

Containers VS virtualizzazione

Container	VM
Agisce al livello del SO: usa lo stesso kernel dell'host per creare molteplici spazi utente	**Agisce a livello hardware**: usa un hypervisor per emulare l'hardware dell'host ed eseguire molteplici SO
Consuma meno risorse: usa lo stesso kernel, SO e risorse dell'host, separa solo l'esecuzione dei processi	**Consuma più risorse**: ogni VM ha il proprio SO, il proprio kernel e un set di risorse I/O virtualizzate
Isolamento solo a livello di processo: processi in un userspace diversi, ma kernel e risorse condivise → potenziali rischi per la sicurezza in certi scenari	**Isolamento completo tra VM**: ogni VM funziona come una macchina fisica separata e non condivide risorse, l'interazione con l'hardware è mediata dall'hypervisor
Migliori prestazioni e gestione semplificata	**Maggiore overhead e gestione più complessa**

Container: svantaggi

- Condivisione del kernel: la compromissione del kernel host causerebbe la compromissione di tutti i container, la compromissione di un container potrebbe diffondersi anche ad altri container
- Isolamento incompleto: i container non offrono lo stesso grado di isolamento delle VM, alcune soluzioni (Docker) hanno meccanismi di sicurezza avanzati ma l'isolamento non è impenetrabile.
- Vulnerabilità e compromissione dell’immagine: più semplice creare e pubblicare immagini che potrebbero contenere vulnerabilità software o codice malevolo: mitigabile usando repository ufficiali ed aggiornando costantemente le immagini

Container: svantaggi

- Persistenza dei dati: di norma quando un container viene fermato o distrutto i file generati durante l'esecuzione vengono persi. La persistenza dei dati richiede soluzioni aggiuntive come volumi o storage esterni da montare nel container.
- Utilizzo avanzato complesso: sebbene adoperare container sia semplice, utilizzi avanzati richiedono l'adozione di stack più complessi come tool multi-container (Docker Compose), orchestratori (Kubernetes), sistemi di logging e monitoraggio, ecc.
- Gestione reti: configurare e gestire la rete negli orchestratori richiede un livello di astrazione notevole

Tecnologie e Implementazioni dei Container

- **Docker**: è la piattaforma più popolare, mette a disposizione diversi tool per la creazione, gestione ed esecuzione dei container
  - Docker Engine: «motore» dello stack, responsabile della creazione e gestione dei container e le interazioni con il SO
  - Docker Image: snapshot in sola lettura del file system di un container, usate per avviare i container, permettono il versioning
  - Dockerfile: script che contiene istruzioni per costruire una Docker Image.
  - Docker Hub/Registry: repository dove le immagini Docker vengono archiviate e condivise
  - Docker Compose: permette di definire e gestire applicazioni multi-container tramite file YAML → avvio simultaneo di microservizi correlati tra loro
  - Docker Volumes, Docker Networking, Docker Machine, …

Tecnologie e Implementazioni dei Container

- **Singularity**: progetto open-source, creato nel Lawrence Berkeley National Laboratory per eseguire in modo portatile e riproducibile le applicazioni di calcolo scientifico in ambienti HPC
- Singularity è stato creato con particolare attenzione a
  - **Sicurezza**: a differenza di Docker, che necessita di privilegi, permette di eseguire container nello userspace riducendo i rischi di sicurezza
  - **Portabilità**: come in altre piattaforme, è possibile creare un file immagine che contiene tutti i file necessari ad eseguire il container
  - **Performance**: Singularity può accedere direttamente alle risorse hardware, come GPU e reti ad alte prestazioni
- Singularity è inoltre compatibile con le immagini Docker, rendendo possibile eseguire il vasto panorama di applicazioni già disponibili

Tecnologie e Implementazioni dei Container

- **Linux Containers (LXC)**: soluzione nativa del kernel Linux, fornisce un’interfaccia per la creazione e gestione di container e la loro esecuzione come processi del kernel
- In sostanza utilizza e coordina diverse funzionalità del kernel:
  - Control Groups (cgroups): gestiscono l'uso delle risorse (CPU, memoria, disco, rete, ecc.) dei processi eseguiti sul kernel.
  - Namespace: forniscono l’isolamento delle risorse di sistema visibili al processo (namespace del processo (PID), del network, dello storage, ecc.)
  - Sicurezza (AppArmor, SELinux, ecc.) : garantire l'isolamento e la sicurezza tra i container.
- Sebbene meno popolare di Docker, LXC rimane una scelta solida per chi cerca un approccio più vicino al kernel Linux.

Container: esempio pratico

- Immaginiamo di voler utilizzare Docker e Docker Compose per creare e gestire un'istanza di WordPress e un database associato, garantendo che questi due servizi siano correttamente connessi.

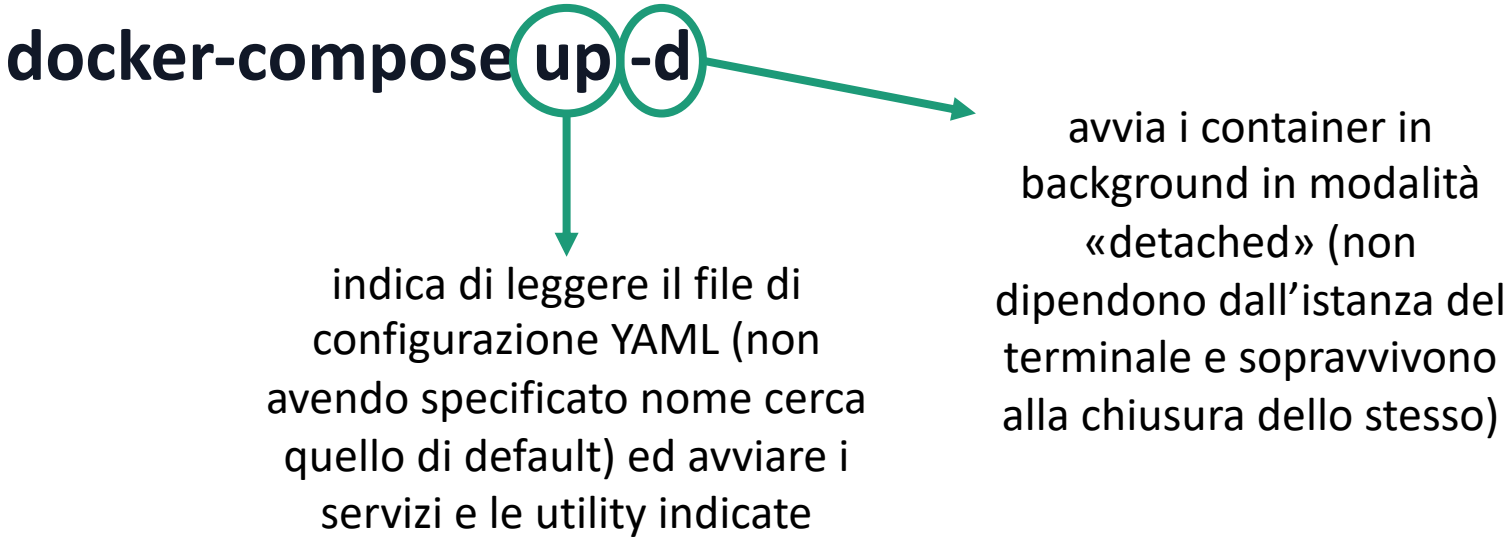
Container: esempio pratico

- Creiamo all'interno della directory del progetto il file **`docker-compose.yml`**
- All'interno specifichiamo i **servizi**, ossia le immagini docker di cui fare il deployment (in questo caso ci serve un database come mariadb e wordpress stesso) e le relative configurazioni (mappatura delle porte, variabili di ambiente, storage, ecc.)
- Poiché siamo interessati a creare dei dati che siano persistenti, specifichiamo due volumi, uno per servizio

```yaml
version: '3'

services:

  db:
    container_name: db
    image: mariadb:latest
    volumes:
      - db_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: rootpass
      MYSQL_DATABASE: wordpress
      MYSQL_USER: sql_username
      MYSQL_PASSWORD: sql_password

  wordpress:
    container_name: wordpress
    depends_on:
      - db
    image: wordpress:latest
    ports:
      - "8000:80"
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: username
      WORDPRESS_DB_PASSWORD: password
    volumes:
      - wordpress_data:/var/www/html

volumes:
  db_data: {}
  wordpress_data: {}
```

Container: esempio pratico

- A questo punto, aprendo un terminale nella directory di lavoro si esegue il comando



- I container vengono quindi istanziati e configurati come richiesto. Aprendo **http://localhost:8000** (la porta 8000 era mappata con la 80 del servizio) tramite browser (o client http) sarà possibile accedere all'istanza di wordpress appena avviata

Container: esempio pratico

- In ogni momento è possibile controllare la lista dei container e il loro stato

docker-compose ps

- Inoltre è possibile consultare i log di tutti i container
- **docker-compose logs**
- o solo di uno specifico, indicando il nome del container così come definito nel docker-compose.yml

docker-compose logs *wordpress*

docker-compose logs *db*

Orchestratori e Container

- Gli orchestratori sono strumenti avanzati nati per la configurazione, gestione, esecuzione e coordinamento di container
- L’orchestrazione è essenziale per applicazione dei container su larga scala ed in ambienti distribuiti: gestire manualmente ciascun container e come questo si interfaccia con le risorse a disposizione e con gli altri container, sarebbe impraticabile.

Orchestratori e Container

- Funzioni fondamentali:
  - Gestione del Deployment: distribuzione uniforme delle applicazioni.
  - Scalabilità: adattano le risorse dinamicamente al carico di lavoro.
  - Self-Healing: rilevamento e correzione automatica delle anomalie.
  - Load Balancing: distribuzione ottimizzata del traffico e delle risorse.
  - Gestione dello Storage: connessione e gestione dello storage persistente.

Orchestratori: Kubernetes

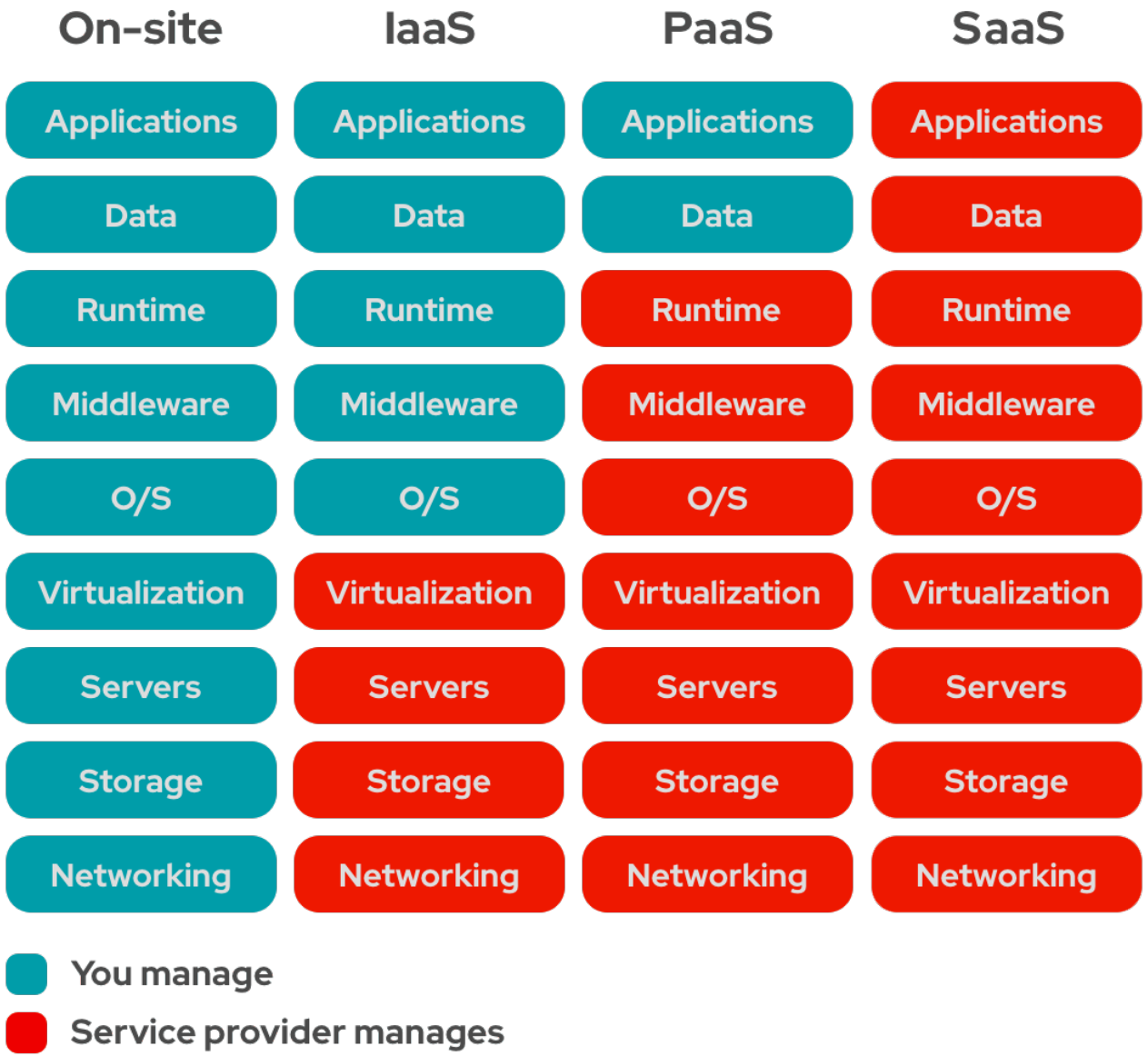
- Kubernetes (K8s) è un orchestratore open-source (originariamente sviluppato da Google), è rapidamente diventato lo standard
- Organizza i container in **pods**, unità atomiche in un cluster K8s
- Ogni pod può contenere uno o più container che condividono le risorse a disposizione
- I **cluster** K8s sono divisi in:
  - Master node(s): controlla e gestisce il cluster (contiene il server API, il controller manager, lo scheduler, ecc.)
  - Worker node(s): eseguono le applicazioni, ciascuno contiene il runtime necessario ai container e le utility per dialogare con il master

Orchestratori: altre soluzioni

- Docker Swarm: apprezzato per la semplicità di utilizzo e l'integrazione diretta con Docker.
- Amazon ECS & EKS: orchestrazione su AWS.
- Apache Mesos & Marathon: adatto per grandi data center.
- OpenShift: piattaforma di RedHat basata su Kubernetes che funzionalità aggiuntive focalizzate a contesti enterprise.

Cloud Computing: Definizione

- Il cloud computing è la fornitura **on-demand** di risorse informatiche (CPU, RAM, storage, ecc.) attraverso servizi Internet
- La gestione dell'infrastruttura hardware sottostante viene quindi totalmente disaccoppiata dall'utilizzo della stessa da parte degli utenti
- All'utente vengono forniti dei servizi, il suo coinvolgimento varia in base al paradigma scelto. In particolare si parla di:
  - **IaaS (Infrastructure as a Service)**: è il paradigma di livello più basso, viene fornita una infrastruttura virtuale «grezza» su cui l'utente può istanziare il software necessario, (Amazon EC2, Google Compute Engine, ecc.)
  - **PaaS (Platform as a Service)**: piattaforme già configurate (compilatori, ambienti di sviluppo, librerie, ecc.) su cui gli sviluppatori possono costruire, personalizzare e implementare le proprie applicazioni (Google App Engine, Microsoft Azure, ecc.)
  - **SaaS (Software as a Service)**: software già configurati e pronti all'uso, ospitati, insieme ai relativi dati, sul cloud. Agli utenti resta solo di utilizzare i software (Microsoft Office 365, Google Workspace, ecc.)



https://www.redhat.com/en/topics/cloud-computing/iaas-vs-paas-vs-saas

Cloud Computing: vantaggi

- Costo: **l'utente non deve acquistare e mantenere l'hardware**, nel caso di SaaS l'utente non si deve nemmeno occupare della **manutenzione** del software;
- Scalabilità: le risorse sono disponibili in tempo reale e in base alla domanda (e alla disponibilita' finanziaria...)
- Performance: le cloud sono solitamente ospitate in data center all'avanguardia ed offrono più performance e meno downtime rispetto soluzioni locali;
- Sicurezza: con le cloud sono spesso offerte soluzioni e tecnologie di protezione dei dati (ridondanza multi sito, cifratura end-to-end, ecc.) difficilmente applicabili in altri contesti

Cloud Computing: svantaggi

- Dipendenza dal provider: inevitabilmente il funzionamento della cloud scelta dipende dal fornitore, **non si ha alcun controllo** su eventuali downtime e problemi
- Sicurezza e Privacy: sebbene vi sono numerose misure di sicurezza, l'archiviazione dei dati e delle applicazioni off-site potrebbe sempre comportare potenziali rischi
- Latenza: passando attraverso Internet, la latenza delle applicazioni può essere sensibilmente maggiore rispetto soluzioni locali

Cloud Computing: elasticità

- Allude alla capacità di gestire agevolmente la variazione della richiesta di risorse, aggiungendone quando necessario e rilasciandole quando il carico diminuisce.
- Questa scalabilità dinamica in base alle reali esigenze permette una maggiore efficienza (si utilizza e paga il necessario)

Cloud Computing: multi-tenancy

- La multi-tenancy è una soluzione architetturale in cui una singola istanza di un'applicazione viene utilizzata da più «tenant» (clienti) mantenendo per ciascuno uno spazio separato ed isolato per i dati, configurazioni, utenti, funzionalità, ecc.
- Un esempio sono i software gestionali (CRM) online: ogni azienda ha una versione separata e privata dei propri dati, ma tutte eseguono la stessa identica versione del software
- Il vantaggio maggiore è l’ottimizzazione delle risorse e la semplificazione della gestione, dovendo curare solo un’istanza del software comune a tutti i tenant

Cloud Computing: federazione

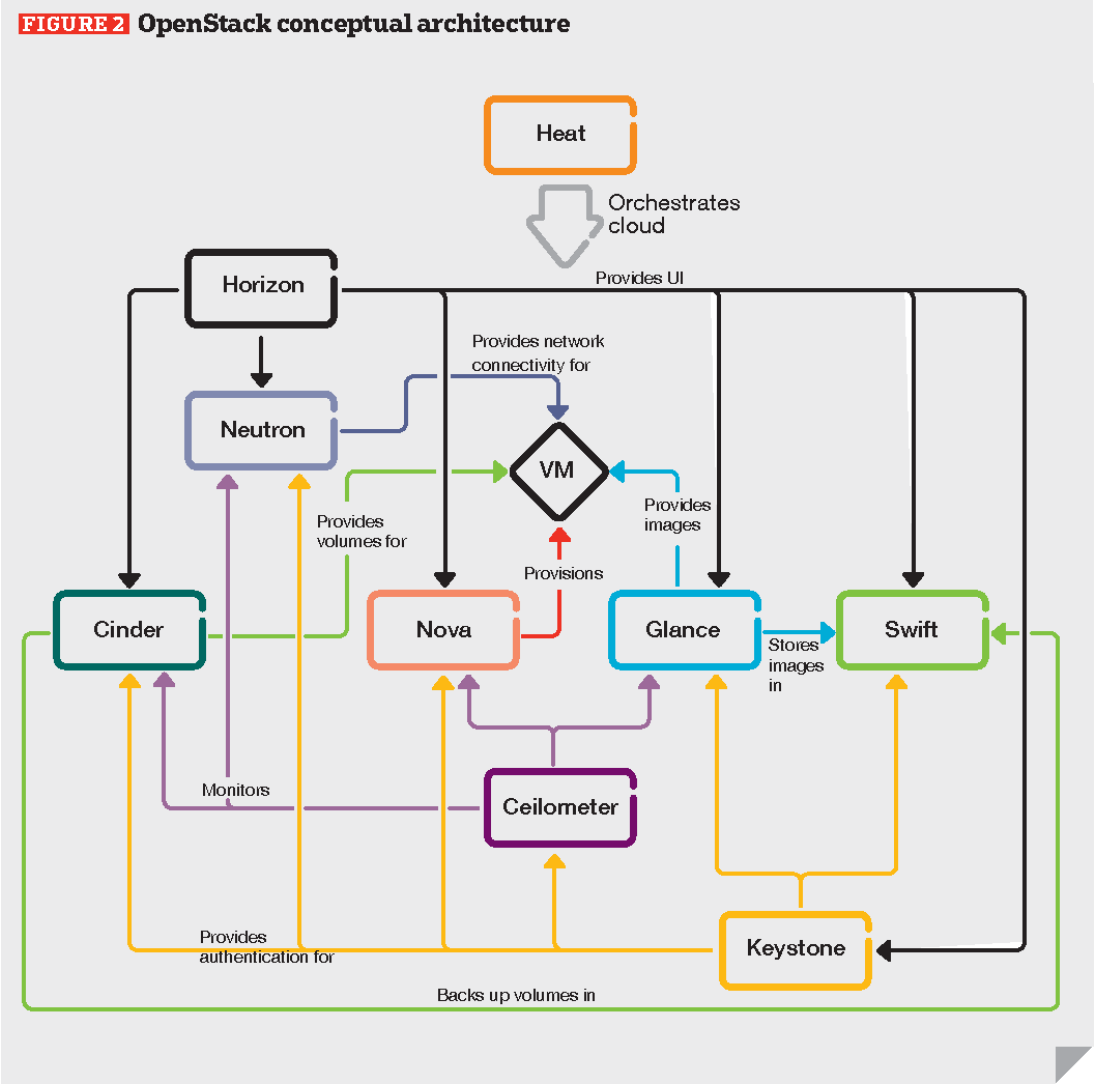
- Necessità di calcolo sempre più avanzato, ridondanza dei servizi e minimizzare i disservizi richiedono l'utilizzo di risorse distribuite
- La federazione nasce con lo scopo di radunare insieme risorse già esistenti ma indipendenti.
- Consiste nell'unione di due o più ambienti di cloud computing per formare un insieme coerente, pur mantenendo la loro autonomia individuale.
- Ciascun membro della federazione mette a disposizione le proprie risorse (tutte o una parte) in modo che possano essere e utilizzate dagli utenti a seconda delle esigenze.

Implementazioni del Cloud Computing

- Piattaforme commerciali, proprietarie: AWS, Google computing...
- Piattaforme open-source sviluppate per creare e gestire in prima persona delle cloud, sia private che pubbliche
  - OpenStack: è la più popolare ed utilizzata piattaforma cloud IaaS, nata nel 2010 come un progetto congiunto tra Rackspace Hosting e NASA, oggi è supportata da un consorzio di aziende e numerosi sviluppatori indipendenti
  - OpenNebula: progettata per gestire risorse eterogenee di data center, cloud pubblici e infrastrutture di edge computing. I principali utilizzi di OpenNebula riguardano la virtualizzazione di data center e l'implementazioni cloud basate su KVM, LXD/LXC e AWS Firecracker. *Disponibile in Community Edition (open-source) e Enterprise Edition (licenza commerciale).*

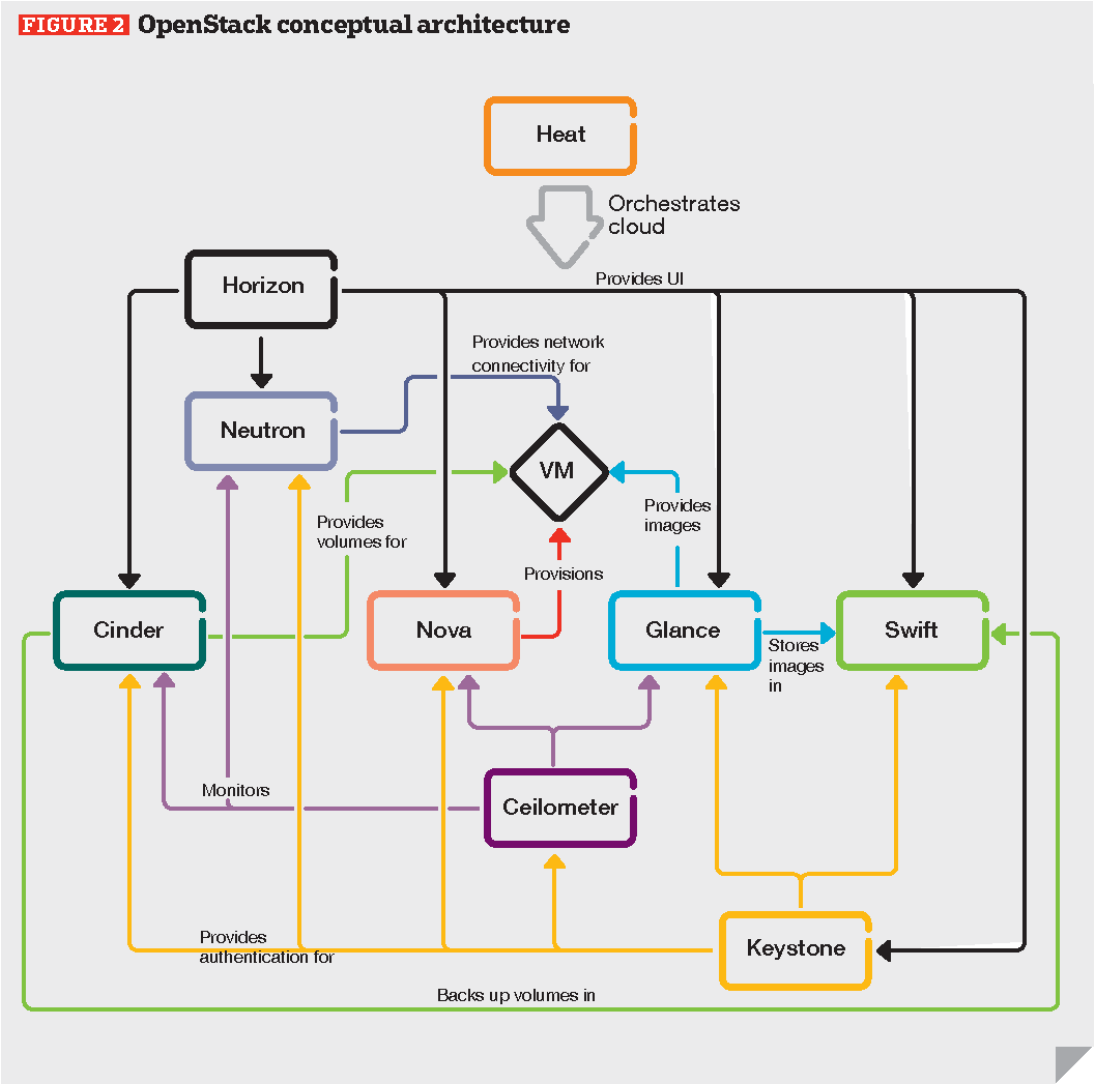
OpenStack

- OpenStack è costituito da una serie di componenti, ognuno con una specifica funzione. I principali sono:
  - **Nova**: componente di calcolo, gestisce le VM.
  - **Swift**: fornisce lo storage ad oggetti, progettato per grandi quantità di dati distribuiti.
  - **Cinder**: storage a blocchi per le istanze virtuali.
  - **Neutron**: gestisce le connessioni tra le interfacce di rete di altri componenti.



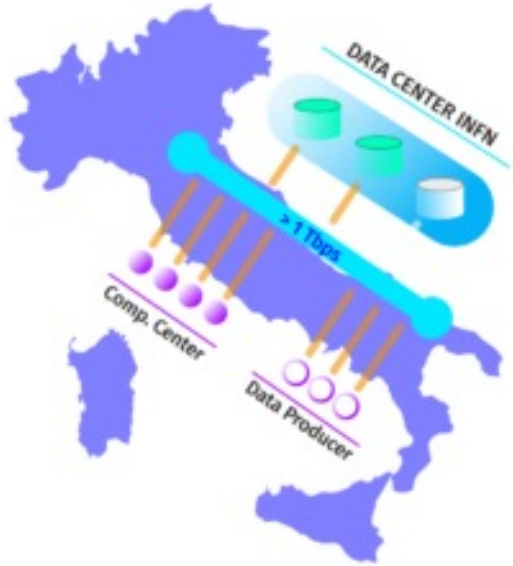
OpenStack

- **Keystone**: Si occupa dell'autenticazione e dell'autorizzazione
- **Horizon**: dashboard attraverso cui gli utenti interagiscono con OpenStack
- **Heat**: orchestrazione e automazione delle risorse cloud
- **Ceilometer**: funzionalità di telemetria e monitoraggio
- **Glance**: si occupa della gestione delle immagini virtuali



INFN Cloud: Architettura

- INFN Cloud sfrutta una backbone che collega insieme due centri di calcolo INFN (CNAF-Bologna e ReCaS-Bari)
- Alla backbone sono poi connessi diverse infrastrutture cloud federate dell'INFN e diverse altre sono in corso di integrazione
- Supporta la federazione di cloud private e pubbliche basate su diverse piattaforme (OpenStack, OpenNebula, AWS, Google Cloud, Microsoft Azure, ecc.)



INFN Cloud: Servizi

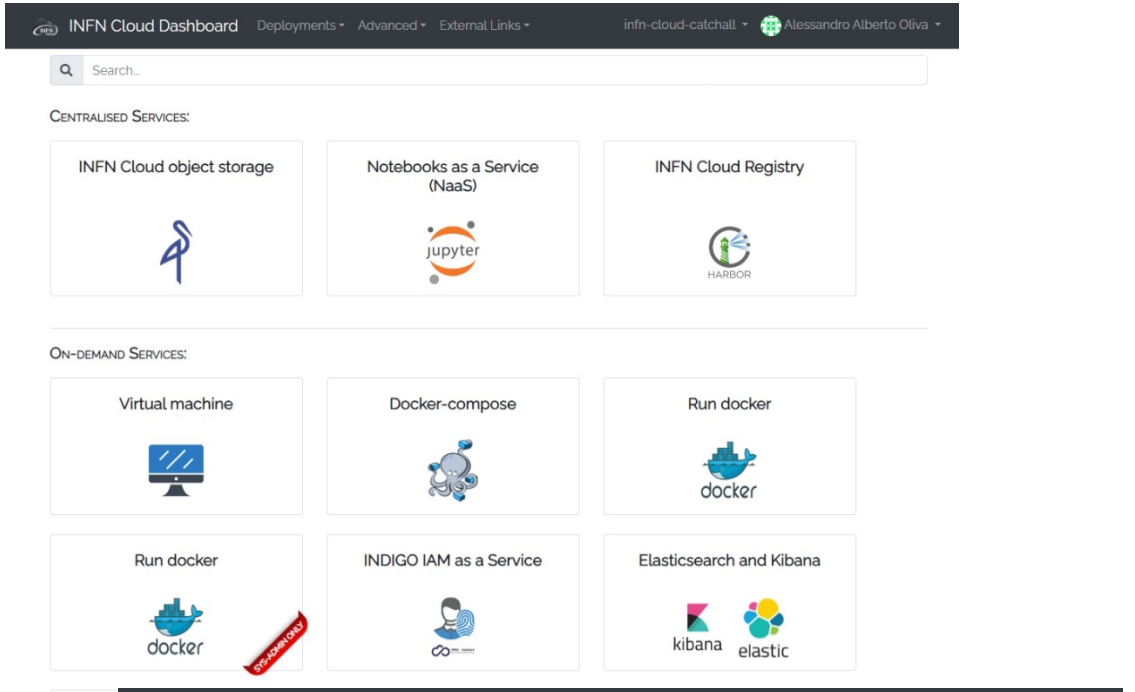
- SaaS:
  - Jupyter notebook
  - Object Storage
  - Cloud Registry
- PaaS (utilizzabili da utenti/amministratori di sistema e personalizzabili per lo specifico use-case):
  - VM on-demand
  - Docker e Docker-compose
  - Cluster K8s
  - ElasticSearch e Kibana
  - IAM as a Service
  - JupyterHub (admin) con kernel personalizzati (Python, MATLAB) e persistenza dei dati

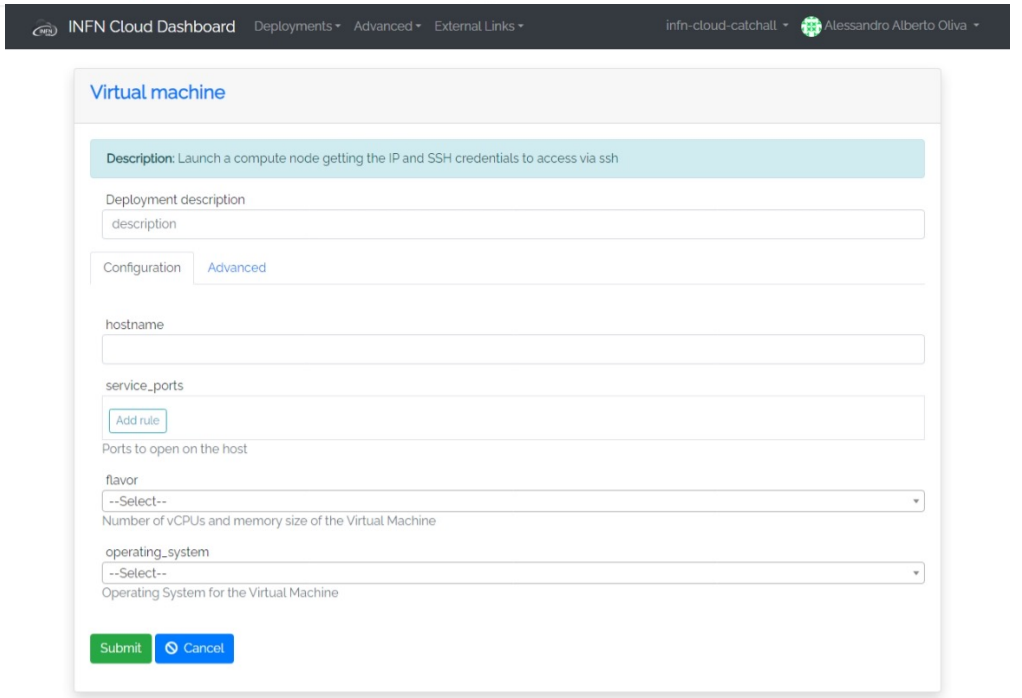
INFN Cloud: Autenticazione

- L’ autenticazione ai servizi INFN Cloud è gestita tramite la piattaforma Identity and Access Management (IAM) https://iam.cloud.infn.it/
- Tipicamente per ottenere un account IAM bisogna essere in possesso di credenziali INFN-AAI e ricoprire un ruolo attivo presso l’INFN (dipendente, associato, visitatore o ospite)
- I servizi a disposizione e i privilegi degli utenti vengono gestiti mediante gruppi e sottogruppi IAM
- E’ possibile implementare una propria instanza di IAM per gestire autonomamente authN/authZ sulle proprie risorse INFN Cloud

INFN Cloud: Dashboard

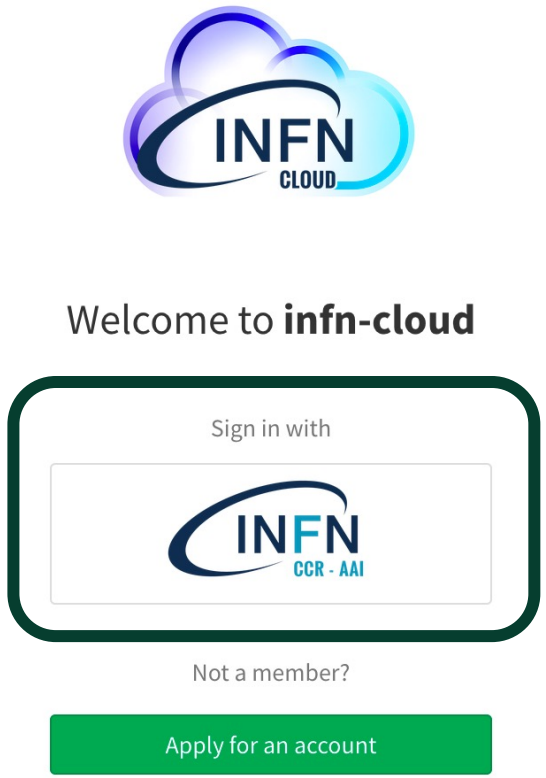
- Tutti i servizi sono gestibili facilmente tramite dashboard web
- L' utente non deve preoccuparsi dell'implementazione dei servizi, ma solo della loro personalizzazione al proprio caso d'uso





INFN Cloud : Come richiedere un account

- Leggere nel dettaglio la guida «Getting Started» disponibile al link: https://guides.cloud.infn.it/docs/users-guides/en/latest/ e assicurarsi di avere i requisiti di accesso
- Applicare per un account IAM al link https://iam.cloud.infn.it/ : al primo accesso, tramite credenziali INFN-AAI, sarà possibile compilare un form in cui indicare nelle note l'utilizzo che si intende fare delle risorse
- Un ticket verrà aperto in automatico tramite cui il team di supporto INFN Cloud aggiornerà lo stato della richiesta



INFN Cloud

- L’INFN mette a disposizione dei suoi utenti una soluzione di Cloud Computing sviluppata ad hoc utilizzando componenti open-source, modulari e scalabili
- Molteplici servizi a disposizione dell’utente, sia di tipo SaaS che PaaS
- Tutti i servizi sono descritti mediante Template TOSCA, un linguaggio open-source nato per descrivere la topologia dei servizi ed applicazioni in cloud, e tool come Ansible ed Helm

Referenze

- https://www.ibm.com/topics/virtualization
- https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/
- https://www.ibm.com/topics/containers
- https://linuxcontainers.org/lxc/introduction/
- https://docs.docker.com/get-started/overview/
- https://kubernetes.io/docs/concepts/overview/
- https://docs.podman.io/en/latest/
- https://www.openstack.org/software/
- https://www.cloud.infn.it/

Bibliografia

- Portnoy M., **Virtualization Essentials *2nd Edition***, John Wiley & Sons (2018)
- Zhang Y., **Network Function Virtualization**, John Wiley & Sons (2018)
- Kocher P. S., **Microservices and containers**, Addison-Wesley Professional (2018)
- Nickoloff J. & Kuenzli S., **Docker in Action *Second Edition***, Manning (2019)
- Hurwitz J. S. & Kirsch D., **Cloud Computing For Dummies**, John Wiley & Sons (2020)
- Silverman B. & Solberg M., **OpenStack for Architects: Design production-ready private cloud infrastructure *2nd Edition,*** Packt Publishing Ltd. (2018).