

Comandi Linux

Clear, **ls**, **cd ..**, **cd <nome_cartella>**, **ifconfig**, **Ctrl + c** (per uscire), **cp <nome_file> <nome_file_nuovo>**,
mkdir (creare una cartella), **touch <nomefile>** (creare un file)
In Linux,

1. **APT (Advanced Package Tool)** è un gestore di pacchetti usato principalmente nelle distribuzioni Debian e Ubuntu
2. **APK (Android Package Kit)** è il formato di pacchetto utilizzato per le applicazioni Android (Alpine). In sostanza, APT è per il software Linux, mentre APK è per le app Android.

apk update

apk add iputils

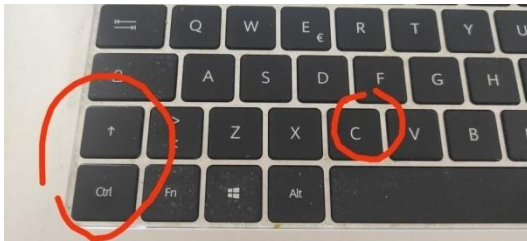
ping <nome_container> control-c per interrompere il ping

Per fare copia ed incolla su linux fare:

Ctrl + Maiusc + c

Ctrl + Maiusc + v

Per fare annulla l'ultima operazione su linux usare il tastierino numerico su



apk add nano (installa un editor di testo)

nano il_tuo_file.py (per avviarlo)

cd ..

mkdir test

touch prova.txt

Per accedere alla macchina linux **docker-desktop** tramite terminale devo digitare:

wsl -d docker-desktop

-d sta per distribuzione

```
Windows PowerShell
PS C:\Users\alber> wsl -d docker-desktop
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt/host/c/Users/alber# cd ..
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt/host/c/Users/alber# cd ..
ls: cd: not found
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt/host/c/Users/alber# cd ..
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt/host/c/Users# cd ..
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt/host# cd ..
docker-desktop:/tmp/docker-desktop-root/run/desktop/mt# cd ..
docker-desktop:/tmp/docker-desktop-root/run/desktop# cd ..
docker-desktop:/tmp/docker-desktop-root/run# cd ..
docker-desktop:/tmp# cd ..
docker-desktop/# cd ..
docker-desktop/# ls
bin          install      shlib
dev          media       sys
docker-desktop-user-distro
etc          net         tmp
home        proc        var
init        root        var
lib         run
```

Esempi di istruzioni Docker

docker build -t albertocesaretti1967/pythonweb1:versione1 .

(-t sta per tag)

docker run -d -p 5000:5000 albertocesaretti1967/pythonweb1:versione1

(-d detach, cioè in background)

docker run -d -p 5000:5000 albertocesaretti1967/pythonweb2:versione1

(-p porte)

docker create -it -p 5000:5000 --name web2 --mount source=volume1,target=/app
albertocesaretti1967/pythonweb1:versione1

```
docker network create network1
docker network ls
docker network inspect network1
docker network connect network1 web1
docker network disconnect network1 web1
```

```
docker run -d --name web1 --network network1 albertocesaretti1967/pythonweb1:versione1
docker run -d --name web2 --network network1 albertocesaretti1967/pythonweb1:versione1
```

```
docker pull mvertes/alpine-mongo (immagine scaricata da docker hub con mongodb)
docker push albertocesaretti1967/pythonweb1:versione1
docker pull albertocesaretti1967/pythonweb1:versione1
```

Esempi di istruzione pip (python install package)

pip list (per vedere le librerie installate)

pip install pymongo

pip list (elenca tutte le librerie installate e la loro versione)

database:

connessione di un client ad un database di un container chiamato mongo dove è installato mongo server

```
from pymongo import MongoClient

try:
    client = MongoClient('mongo', 27017)
    print("connessione al server Mongodb riuscita")
except Exception as e:
    print(f"Errore connessione al server Mongodb non riuscita : {e}")
```

Comandi principali per la gestione dei container:

- **docker run**: Crea e avvia un nuovo container.
- **docker ps**: Elenca i container in esecuzione.
- **docker start**: Avvia un container.
- **docker stop**: Arresta un container.
- **docker kill**: Forza l'arresto di un container.
- **docker rm**: Elimina un container.
- **docker exec**: Esegue un comando all'interno di un container.
- **docker logs**: Visualizza i log di un container.
- **docker attach**: Si attacca a un container in esecuzione.
- **docker rename**: Rinomina un container.

Comandi principali per la gestione delle immagini:

- **docker pull**: Scarica un'immagine da un repository Docker.
- **docker images**: Elenca le immagini locali.
- **docker build**: Crea una nuova immagine da un Dockerfile.
- **docker tag**: Applica un tag a un'immagine.
- **docker push**: Invia un'immagine a un repository Docker.
- **docker rmi**: Elimina un'immagine.

Comandi principali per la gestione dei volumi:

- **docker volume ls**: Elenca i volumi.
- **docker volume create**: Crea un nuovo volume.
- **docker volume rm**: Elimina un volume.

Comandi principali per la gestione delle reti:

- **docker network create** <nome_rete> : crea una rete
- **docker network ls**: Elenca tutte le reti Docker disponibili.
- **docker network inspect** <nome_rete>: Mostra informazioni dettagliate su una specifica rete.
- **docker network connect** <nome_rete> <nome_container>: Connette un container esistente a una rete.
- **docker network disconnect** <nome_rete> <nome_container>: Disconnette un container da una rete.
- **docker network rm** <nome_rete>: Rimuove una rete (solo se nessun container è connesso).

Comandi specifici per Docker Compose:

- **docker-compose up**: Crea e avvia i servizi definiti in un file docker-compose.yml.
- **docker-compose down**: Arresta e rimuove i container e i servizi definiti nel file docker-compose.yml.
- **docker-compose build**: Costruisce le immagini Docker per i servizi definiti nel file docker-compose.yml.
- **docker-compose logs**: Visualizza i log dei servizi definiti nel file docker-compose.yml.

Altre utilità:

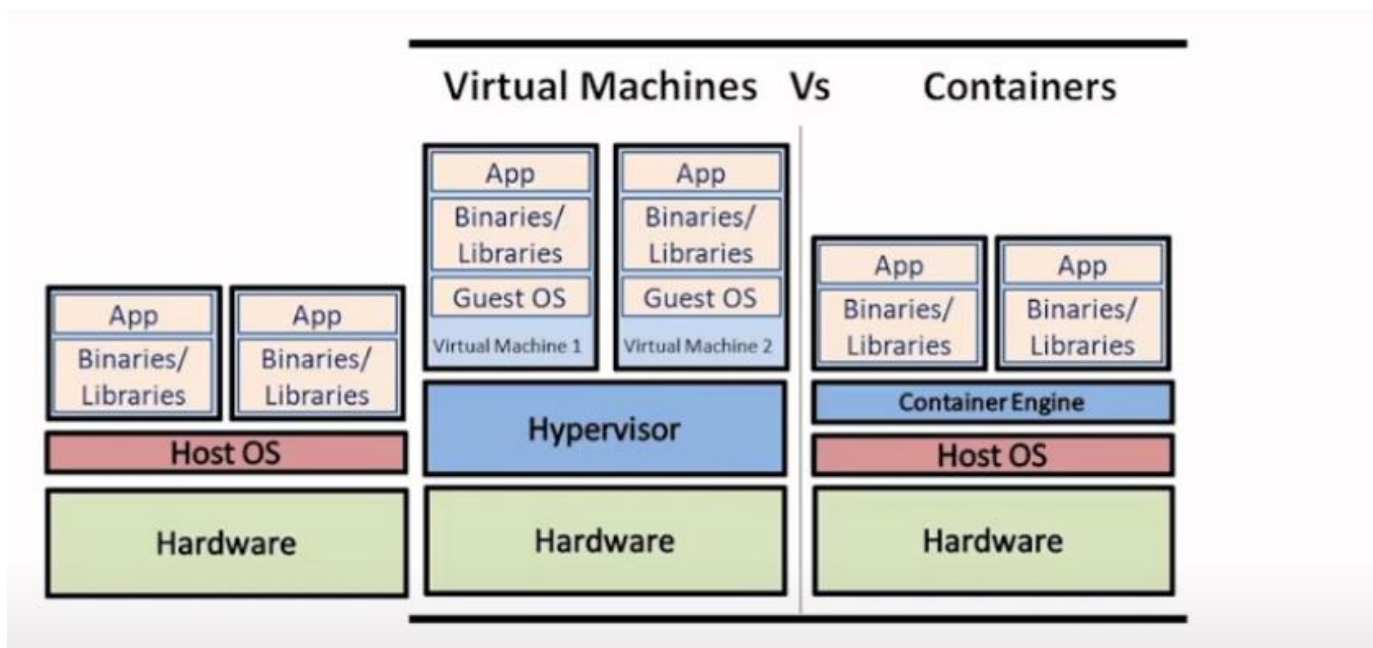
- **docker inspect**: Visualizza informazioni dettagliate su un container o un'immagine.
 - **docker context**: Gestisce i contesti Docker, permettendo di specificare un diverso ambiente per le operazioni.
 - **docker network**: Gestisce le reti Docker, permettendo ai container di comunicare tra loro.
 - **docker system**: Fornisce informazioni sul sistema Docker, come la versione, le risorse utilizzate, ecc.
 - **docker search**: Ricerca immagini su Docker Hub.
 - **docker history**: Mostra lo storico di un'immagine Docker.
 - **docker commit**: Crea una nuova immagine da un container.
- Comandi per la gestione della sicurezza:
- **docker trust**: Per la gestione della fiducia nelle immagini Docker.
 - **docker plugin install**: Per l'installazione di plugin Docker.

Docker desktop (Windows)

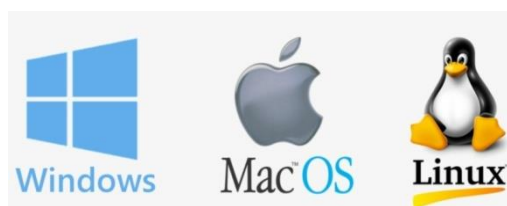
1. Cos'è Docker
2. GUI e CLI
3. Realizzare un progetto software e poterlo installare su qualsiasi PC o Server
4. Ambiente di esecuzione Containerizzato
5. WSL 2
6. Abilitare WSL 2
7. Come installare le distribuzioni linux su Windows dopo aver installato WSL
8. Come accedere al file system e distribuzioni linux installate su Windows

9. Virtualizzazione
10. Installare Docker
11. La macchina virtuale linux docker- desktop
12. Docker desktop GUI (Graphic user interface, interfaccia grafica)
13. Docker desktop CLI (Command line interface, interfaccia a riga di comando)
14. Abilitare l'integrazione Docker con WSL 2
15. Docker Hub
16. Le reti in Docker
17. Volumi in Docker

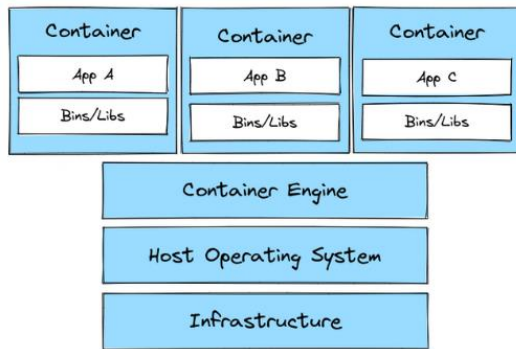
Virtualizzazione



Cos'è Docker



Docker è una piattaforma (un insieme di programmi) che permette di **impacchettare un'applicazione e tutte le sue dipendenze (librerie, strumenti di sistema, codice, runtime)** in un'unità standardizzata chiamata **container**.



Immagina di dover spedire un'applicazione. Tradizionalmente, dovresti assicurarti che l'ambiente sul server di destinazione sia esattamente lo stesso del tuo ambiente di sviluppo. Questo può portare a problemi come "funziona sulla mia macchina!".

Docker risolve questo problema creando un container isolato che contiene tutto ciò di cui l'applicazione ha bisogno per funzionare, indipendentemente dal sistema operativo sottostante.

Ecco alcuni concetti chiave di Docker:

- **DockerFile** : è un file di testo che contiene tutte le istruzioni necessarie per costruire un'immagine Docker
- **Immagine Docker (Docker Image)**: È un modello di sola lettura con le istruzioni per creare un container. Pensa a un'immagine come a un "pacchetto" che contiene il codice dell'applicazione, le librerie, le variabili d'ambiente, i file di configurazione e il runtime.
- **Container Docker (Docker Container)**: È un'istanza eseguibile di un'immagine Docker. È un ambiente isolato in cui l'applicazione viene eseguita. Puoi avere più container in esecuzione dalla stessa immagine. Pensa a un container come a una "scatola" creata a partire dal "pacchetto" dell'immagine.
- **Docker Hub**: È un registro (repository) pubblico e privato per la condivisione di immagini Docker. È come un "app store" per le immagini.
- **Docker Desktop**: È un'applicazione facile da usare per Mac e Windows che ti permette di costruire, eseguire e gestire container Docker sulla tua macchina locale.
- **Docker Engine**: È il motore principale di Docker, responsabile della creazione e gestione di immagini, container, reti e volumi.

In sintesi, Docker ti permette di:

- **Sviluppare più velocemente**: Crea ambienti coerenti per lo sviluppo, il test e la produzione.
- **Spedire software più rapidamente**: Impacchetta le applicazioni e le loro dipendenze in modo portatile.
- **Eseguire ovunque**: I container possono essere eseguiti su qualsiasi sistema che supporti Docker (Linux, Windows, macOS, cloud).
- **Scalare facilmente**: Puoi avviare e arrestare rapidamente più istanze dei tuoi container.
- **Migliorare l'utilizzo delle risorse**: I container sono leggeri e condividono il kernel del sistema operativo host, il che li rende più efficienti delle macchine virtuali tradizionali.

Docker è diventato uno strumento fondamentale per gli sviluppatori e i team DevOps per semplificare e accelerare il ciclo di vita delle applicazioni.

GUI e CLI

1. Programmi con solo GUI (graphic user interface, interfaccia grafica)
2. Programmi con solo CLI (command line interface, a riga di comando)
3. Programmi sia con GUI che CLI

1. Programmi con solo GUI

Ci sono moltissimi programmi su Windows che hanno un'interfaccia utente esclusivamente grafica (GUI) e non offrono un'interazione diretta tramite riga di comando. La maggior parte delle applicazioni che usiamo quotidianamente rientra in questa categoria.

Ecco alcuni esempi di programmi comuni su Windows che sono prevalentemente o esclusivamente GUI-based:

- **Browser Web (Chrome, Firefox, Edge, Safari):** Interagisci con internet tramite finestre, menu, pulsanti e campi di testo. Anche se alcuni browser hanno strumenti per sviluppatori accessibili tramite tasti di scelta rapida, l'interazione principale è grafica.
- **Elaboratori di Testo (Microsoft Word, LibreOffice Writer):** La creazione e la formattazione di documenti avviene tramite barre degli strumenti, menu e l'area di editing visuale.
- **Fogli di Calcolo (Microsoft Excel, LibreOffice Calc):** L'inserimento, la manipolazione e la visualizzazione dei dati sono gestite tramite celle, formule inserite in barre dedicate e grafici generati visivamente.
- **Presentazioni (Microsoft PowerPoint, LibreOffice Impress):** La creazione di slide, l'aggiunta di elementi multimediali e l'organizzazione della presentazione avvengono tramite un'interfaccia visuale.
- **Lettori Multimediali (VLC Media Player, Windows Media Player):** La riproduzione di audio e video è controllata tramite pulsanti, barre di scorrimento e menu grafici.
- **Software di Grafica e Design (Adobe Photoshop, GIMP (anche se ha alcune interazioni via script), Inkscape):** La creazione e la modifica di immagini e grafiche avviene tramite strumenti visuali, pennelli, filtri e livelli gestiti con il mouse.
- **La maggior parte dei giochi:** L'interazione con i videogiochi avviene tramite controlli grafici, menu di impostazione visuali e l'ambiente di gioco renderizzato.
- **Esplora File (Windows Explorer):** La gestione di file e cartelle avviene tramite finestre, icone, menu contestuali e operazioni di drag-and-drop. Anche se ci sono comandi che puoi usare nella barra degli indirizzi o nel Prompt dei comandi per interagire con il file system, l'interfaccia principale è grafica.
- **Pannello di Controllo e Impostazioni di Windows:** La configurazione del sistema operativo avviene tramite finestre, menu a tendina, caselle di controllo e cursori.
- **La maggior parte delle applicazioni di produttività personale:** Client di posta elettronica (Outlook, Thunderbird), calendari, app per le note (Evernote, OneNote), ecc., sono principalmente GUI-based.

Perché solo GUI?

- **Facilità d'uso per l'utente finale:** Le GUI sono spesso più intuitive e facili da apprendere per gli utenti che non hanno familiarità con i comandi testuali.
- **Visualizzazione immediata:** Le GUI permettono di vedere direttamente il risultato delle azioni (ad esempio, la formattazione del testo, la modifica di un'immagine).
- **Interazione complessa:** Alcune operazioni complesse (come il disegno grafico o l'editing video) si prestano meglio a un'interfaccia visuale con manipolazione diretta.

In sintesi, la maggior parte delle applicazioni destinate all'utente finale su Windows sono progettate con un'interfaccia grafica per massimizzare l'usabilità e l'accessibilità. Anche se la riga di comando rimane uno strumento potente per sviluppatori e amministratori di sistema, l'esperienza utente quotidiana è dominata dalle GUI.

2. Programmi con solo CLI

Su Windows, ci sono molti programmi e strumenti che vengono installati e utilizzati principalmente, se non esclusivamente, tramite la Command Line Interface (CLI) e non offrono un'interfaccia grafica utente (GUI) tradizionale. Gli esempi che hai citato sono perfetti:

- **pip (Package Installer for Python):** Quando installi Python, pip viene spesso incluso. È uno strumento da riga di comando che ti permette di installare, aggiornare e gestire pacchetti software scritti in Python. Non ha una GUI integrata per le operazioni di base.
- **npm (Node Package Manager):** Installato con Node.js, npm è un gestore di pacchetti per JavaScript. Viene utilizzato tramite comandi nel terminale per installare librerie, eseguire script e gestire dipendenze per progetti JavaScript. Anche npm è prevalentemente una CLI.
- **Interprete Python (python.exe):** L'interprete Python stesso è un'applicazione da riga di comando. Lo avvii nel terminale per eseguire script Python interattivamente o per lanciare file .py. Sebbene esistano IDE (Integrated Development Environments) con GUI che facilitano lo sviluppo in Python, l'interprete base è CLI-first.

Altri esempi di programmi spesso installati e utilizzati senza GUI su Windows includono:

- **Git:** Un sistema di controllo versione distribuito. Interagisci con Git principalmente tramite comandi nel terminale per gestire repository, commit, branch e altro. Esistono GUI per Git, ma lo strumento fondamentale è la CLI.
- **PowerShell:** La shell di scripting e riga di comando avanzata di Microsoft. È un'interfaccia basata su testo per gestire il sistema operativo e le applicazioni.
- **Diversi linguaggi di programmazione e i loro compilatori/interpreti:** Oltre a Python, linguaggi come Node.js (node), Go (go), Rust (rustc, cargo), e molti altri, forniscono strumenti da riga di comando per compilare, eseguire e gestire progetti.
- **Utilità di sistema:** Molti strumenti di sistema integrati in Windows o forniti da terze parti sono accessibili tramite la riga di comando per compiti di amministrazione, configurazione di rete, gestione dei processi, ecc. Esempi includono ipconfig, netstat, tasklist, e molti altri.

- **Server web e database (in modalità server):** Software come server web (Nginx, Apache in alcune configurazioni) e sistemi di gestione di database (come MySQL, PostgreSQL in configurazioni server) spesso vengono gestiti tramite file di configurazione e comandi da riga di comando, specialmente quando operano come servizi in background.
- **Strumenti di automazione:** Molti strumenti per l'automazione di task, come Ansible (anche se ha interfacce web opzionali), vengono utilizzati principalmente tramite CLI per definire ed eseguire playbook.
- **Librerie e framework di sviluppo:** Molte librerie e framework che utilizzi per sviluppare software (anche con GUI) vengono installate e gestite tramite CLI (come pip per Python o npm per JavaScript).

Perché alcuni programmi non hanno una GUI?

- **Efficienza e automazione:** Le CLI sono spesso più efficienti per automatizzare compiti tramite script.
- **Accesso remoto:** Sono ideali per l'accesso e la gestione di sistemi remoti tramite SSH o altri protocolli basati su testo.
- **Leggerezza:** I programmi senza GUI tendono ad essere più leggeri in termini di risorse di sistema.
- **Focus sulla funzionalità principale:** Alcuni strumenti sono progettati specificamente per una serie di operazioni ben definite che si prestano bene all'interazione tramite comandi.
- **Sviluppo per server:** Molti software destinati all'esecuzione su server (dove un'interfaccia grafica sarebbe superflua) sono sviluppati senza GUI.

In sintesi, è molto comune trovare programmi e strumenti installati su Windows che non hanno un'interfaccia grafica e si utilizzano esclusivamente tramite la riga di comando.

3. Programmi con GUI e CLI

Molti programmi popolari su Windows, inclusi Docker Desktop, WinRAR e Git Desktop, offrono sia un'interfaccia utente grafica (GUI) che un'interfaccia a riga di comando (CLI).

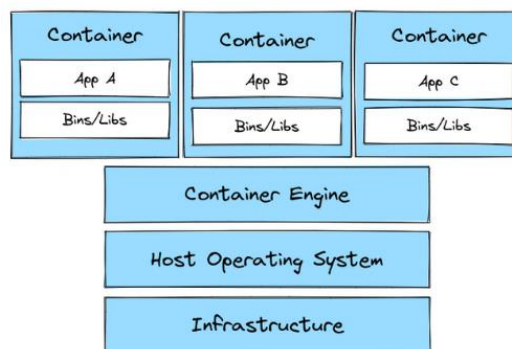
- **Docker Desktop:** Fornisce un'interfaccia grafica intuitiva per gestire container, immagini, volumi e molto altro. Parallelamente, Docker Engine, che è alla base di Docker Desktop, espone una potente CLI (il comando docker) per eseguire tutte le operazioni tramite riga di comando.
- **WinRAR:** L'applicazione che conosciamo ha una GUI completa per creare e gestire archivi compressi. Tuttavia, WinRAR include anche una potente CLI accessibile tramite il comando rar (o unrar per estrarre). Questo permette di automatizzare operazioni di compressione/decompressione tramite script.
- **Git Desktop:** Questa applicazione offre un'interfaccia grafica user-friendly per interagire con i repository Git. Dietro le quinte, Git Desktop utilizza la CLI di Git (il comando git) per eseguire le operazioni di controllo versione. Gli utenti più esperti possono sempre aprire una shell o un terminale e utilizzare direttamente i comandi Git.
- **Microsoft SQL Server:** Il sistema di gestione di database di Microsoft offre SQL Server Management Studio (SSMS), una GUI completa per amministrare server, database e scrivere query. Tuttavia, include anche utilità a riga di comando come sqlcmd e PowerShell module SqlServer per automatizzare attività di gestione e distribuzione.
- **MySQL:** Similmente a SQL Server, MySQL fornisce MySQL Workbench come GUI per l'amministrazione e lo sviluppo, ma offre anche un potente client a riga di comando (mysql) per interagire direttamente con il server.
- **Alcuni software di compressione/decompressione:** Programmi come 7-Zip offrono un'interfaccia grafica facile da usare per comprimere ed estrarre file, ma includono anche un eseguibile a riga di comando (7z.exe) per automatizzare queste operazioni tramite script.
- **Strumenti di rete (es. ping, tracert, ipconfig):** Questi strumenti fondamentali di Windows sono puramente CLI, ma esistono anche utility di rete con interfaccia grafica che forniscono funzionalità simili in modo più visuale.

Questa doppia natura (GUI e CLI) è molto comune in software per sviluppatori e amministratori di sistema perché offre flessibilità:

- **GUI:** Rende il software più accessibile e facile da usare per i principianti o per operazioni rapide e visive.
- **CLI:** Offre maggiore potenza, automazione tramite script, e controllo preciso per utenti esperti e in contesti di automazione.

Quindi, la tua osservazione è assolutamente corretta! Ci sono molti altri programmi che seguono questa filosofia di design.

Ambiente di esecuzione Containerizzato



L' **Ambiente di Esecuzione Containerizzato** comprende tre elementi base:

- **Il kernel del sistema operativo** (esempio kernel Linux)
- **Container engine**, che gestisce l'interazione con il sistema operativo e con i container (esempio Docker engine)
- **I container**, questo è il concetto astratto di ambiente isolato (esempio Docker container)

Un **container** è un ambiente di esecuzione **isolato** e **leggero** che contiene:

1. **l'applicazione**
2. **le dipendenze** (librerie, file di configurazione, runtime, ecc.)
3. **la userland** (parte del kernel linux, cioè l'insieme dei file, delle librerie e degli eseguibili che le applicazioni utilizzano per interagire con il kernel del sistema operativo).

in questo modo l'applicazione può essere eseguita in modo consistente su **qualsiasi infrastruttura che supporti i container**, quindi su qualsiasi computer e su qualsiasi sistema operativo

Pensa a un container come a una **scatola standardizzata** che contiene tutto il necessario per far funzionare un'applicazione. Questa scatola può essere spostata facilmente da un ambiente all'altro (dal tuo computer di sviluppo a un server di test, a un server di produzione) e l'applicazione al suo interno funzionerà sempre allo stesso modo, perché ha con sé tutto ciò di cui ha bisogno.

Ecco le caratteristiche chiave di un container:

- **Isolamento:** Ogni container è isolato dagli altri container e dal sistema operativo host. Questo significa che i processi in esecuzione all'interno di un container non possono interferire con i processi in altri container o con il sistema operativo sottostante.
- **Leggerezza:** I container condividono il kernel del sistema operativo host, il che li rende molto più leggeri e veloci da avviare rispetto alle macchine virtuali (VM). Le VM emulano un intero sistema operativo, consumando molte più risorse.
- **Portabilità:** Un'immagine container (il "modello" da cui viene creato un container) può essere eseguita su qualsiasi sistema che supporti la tecnologia container (come Docker o Kubernetes), garantendo coerenza tra gli ambienti.
- **Consistenza:** Poiché un container include tutte le dipendenze necessarie per l'applicazione, si elimina il problema del "funziona sulla mia macchina". L'ambiente di esecuzione è sempre lo stesso.
- **Efficienza delle risorse:** Più container possono essere eseguiti sulla stessa macchina fisica rispetto alle macchine virtuali, sfruttando meglio le risorse hardware.
- **Scalabilità:** I container possono essere facilmente replicati e distribuiti su più server per gestire un aumento del carico di lavoro.

In analogia:

Immagina di dover spedire diversi tipi di merci. Invece di imballare ogni oggetto singolarmente, li metti in dei container standardizzati. Ogni container ha un'etichetta che descrive il contenuto (l'applicazione e le sue dipendenze). La nave (il sistema operativo host) non si preoccupa di cosa c'è dentro ogni container, sa solo come trasportarli. Una volta arrivati a destinazione, i container possono essere scaricati e aperti, e il contenuto funzionerà come previsto perché era tutto incluso.

Tecnologie principali per i container:

- **Docker:** La piattaforma più popolare per la creazione, la gestione e l'esecuzione di container. Fornisce strumenti per creare immagini container (i "modelli" dei container) e per eseguire istanze di questi container.
- **Kubernetes (K8s):** Un sistema di orchestrazione di container open-source che automatizza il deployment, la scalabilità e la gestione delle applicazioni containerizzate su larga scala.

In sintesi, un container è un modo efficiente e portatile per impacchettare ed eseguire applicazioni con tutte le loro dipendenze, garantendo isolamento e consistenza tra diversi ambienti. È una tecnologia fondamentale nel mondo dello sviluppo e del deployment del software moderno.

La userland

Un contenitore Docker puoi sia includere una parte del sistema operativo (solitamente le librerie e gli eseguibili necessari) sia fare a meno di includere un intero nucleo del sistema operativo. Cerchiamo di capire meglio le due situazioni:

1. Includere una parte del sistema operativo (Userland):

Questo è lo scenario più comune. Quando crei un'immagine Docker, parti spesso da un'immagine di base che fornisce una versione minimale di un sistema operativo Linux (ad esempio Ubuntu, Alpine, CentOS). Questa immagine di base contiene:

- **Userland:** L'insieme dei file, delle librerie e degli eseguibili che le applicazioni utilizzano per interagire con il kernel del sistema operativo. Questo include cose come la shell (bash, sh), le librerie standard (libc), i gestori di pacchetti (apt, yum), e altri strumenti di base.
- **Kernel:** L'immagine di base *non* include un kernel completo. I contenitori Docker condividono il kernel del sistema operativo host (il sistema operativo su cui sta girando Docker).

Quindi, in questo caso, il tuo contenitore ha al suo interno una parte del sistema operativo (l'userland) necessaria per eseguire le tue applicazioni, ma non un kernel separato. Questo rende i contenitori leggeri ed efficienti, in quanto non devono caricare un intero sistema operativo.

2. Non includere un "nucleo" del sistema operativo (Kernel):

È vero che un contenitore Docker **non include un kernel del sistema operativo separato**. Come accennato, i contenitori Docker sfruttano le funzionalità di isolamento del kernel del sistema operativo host (principalmente attraverso i *namespaces* e i *cgroups* su Linux).

Quindi, in un certo senso, potresti dire che il contenitore "non inserisce" un nucleo del sistema operativo proprio perché lo condivide con l'host.

In sintesi:

- Un'immagine Docker contiene tipicamente una parte del sistema operativo chiamata **userland**, che fornisce l'ambiente necessario per eseguire le applicazioni.

Un contenitore Docker **non contiene un kernel completo** del sistema operativo. Utilizza il kernel del sistema operativo host.

WSL 2 (Windows Subsystem for Linux)

WSL 2 è l'acronimo di **Windows Subsystem for Linux 2**. Si tratta di una funzionalità di Windows che consente di eseguire una distribuzione Linux direttamente su Windows, senza la necessità di una macchina virtuale separata o di un dual boot.

WSL 2 fa parte del sistema operativo Windows.

È una funzionalità opzionale che deve essere abilitata dall'utente. Una volta abilitata, permette di eseguire un ambiente Linux direttamente su Windows, senza la necessità di una macchina virtuale tradizionale.

In sostanza, WSL 2 è un sottosistema di Windows che fornisce un kernel Linux completo in esecuzione all'interno di una macchina virtuale leggera, integrata strettamente con il sistema operativo host.

WSL 2 è una componente integrante di Windows, anche se richiede un'attivazione manuale.

Ecco alcuni punti chiave su WSL 2:

- **Architettura:** A differenza della prima versione di WSL, che utilizzava un layer di compatibilità per tradurre le chiamate di sistema Linux in chiamate di sistema Windows, **WSL 2 esegue un vero kernel Linux all'interno** di una macchina virtuale leggera (basata su un sottoinsieme di funzionalità di Hyper-V). Questo porta a una **compatibilità delle chiamate di sistema completa** e a **prestazioni significativamente migliorate**, specialmente per operazioni che richiedono un elevato I/O del file system.

- **Kernel Linux:** WSL 2 include il proprio kernel Linux, il che significa che si ha una compatibilità molto più elevata con le applicazioni Linux, inclusi Docker e altre applicazioni che richiedono funzionalità specifiche del kernel. Gli aggiornamenti al kernel Linux sono disponibili immediatamente.
- **Prestazioni:** WSL 2 offre prestazioni notevolmente superiori rispetto a WSL 1, in particolare per attività come la decompressione di archivi, l'utilizzo di Git, l'installazione di pacchetti tramite gestori come npm o pip, e la compilazione di progetti. In molti scenari, le prestazioni si avvicinano a quelle di un'installazione Linux nativa.
- **Integrazione con Windows:** WSL 2 mantiene una buona integrazione con Windows. È possibile accedere ai file di Windows dall'ambiente Linux e viceversa. Si possono anche eseguire applicazioni GUI Linux direttamente sul desktop di Windows (con WSLg).
- **Facilità d'uso:** L'installazione e la gestione delle distribuzioni Linux in WSL 2 sono relativamente semplici, con la possibilità di scaricare distribuzioni dal Microsoft Store o importare distribuzioni personalizzate.
- **Supporto GPU:** WSL 2 supporta l'utilizzo della GPU per accelerare carichi di lavoro di machine learning eseguiti in Linux.
- **Supporto IPv6:** A partire da Windows 11, WSL 2 supporta le connessioni IPv6.

In sintesi, WSL 2 è un potente strumento per gli sviluppatori che lavorano in ambiente Windows ma hanno bisogno di utilizzare strumenti, librerie e ambienti specifici di Linux. Offre un'esperienza più fluida, performante e compatibile rispetto alla precedente generazione di WSL.

L'indirizzo di rete per le macchine virtuali

```

Selezione Prompt dei comandi

Scheda Ethernet vEthernet (WSL):
  Suffisso DNS specifico per connessione:
  Indirizzo IPv6 locale rispetto al collegamento . : fe80::e8a6:c82:4c84:2c4eX35
  Indirizzo IPv4 . . . . . : 172.22.96.1
  Subnet mask . . . . . : 255.255.240.0
  Gateway predefinito . . . . . :

Scheda LAN wireless Connessione alla rete locale (LAN)* 1:
  Stato supporto. . . . . : Supporto disconnesso
  Suffisso DNS specifico per connessione:

Scheda LAN wireless Connessione alla rete locale (LAN)* 2:
  Stato supporto. . . . . : Supporto disconnesso
  Suffisso DNS specifico per connessione:

Scheda LAN wireless Wi-Fi:
  Suffisso DNS specifico per connessione: wind3.hub
  Indirizzo IPv6 locale rispetto al collegamento . : fe80::37b7:4f2b:d899:544c36
  Indirizzo IPv4 . . . . . : 192.168.1.10
  Subnet mask . . . . . : 255.255.255.0
  Gateway predefinito . . . . . : 192.168.1.1

Scheda Ethernet Connessione di rete Bluetooth:
  Stato supporto. . . . . : Supporto disconnesso
  Suffisso DNS specifico per connessione:
  
```

Per accedere alla macchina linux **docker-desktop** tramite terminale devo digitare:

wsl -d docker-desktop
-d sta per distribuzione

```

Windows PowerShell

PS C:\> wsl -d docker-desktop
docker-desktop: /mnt/desktop/mnt/host/c/Users/alber# ls
bin
media
srv
docker-desktop-user-distro
apt
tmp
home
proc
asr
lib
run
docker-desktop:/#
  
```

Abilitare WSL 2

Ecco i passaggi per abilitare WSL 2 sul tuo sistema Windows:

Prerequisiti:

Prima di iniziare, assicurati che il tuo sistema soddisfi i requisiti per eseguire WSL 2:

- **Versione di Windows:**
 - Per sistemi **x64**: Versione 1903 o successiva con Build 18362 o successiva.
 - Per sistemi **ARM64**: Versione 2004 o successiva con Build 19041 o successiva.
- **Virtualizzazione abilitata:** Potrebbe essere necessario abilitare le funzionalità di virtualizzazione nel BIOS/UEFI del tuo computer.

Passaggi per abilitare WSL 2:

Esistono diversi modi per abilitare WSL 2. Ecco il principale

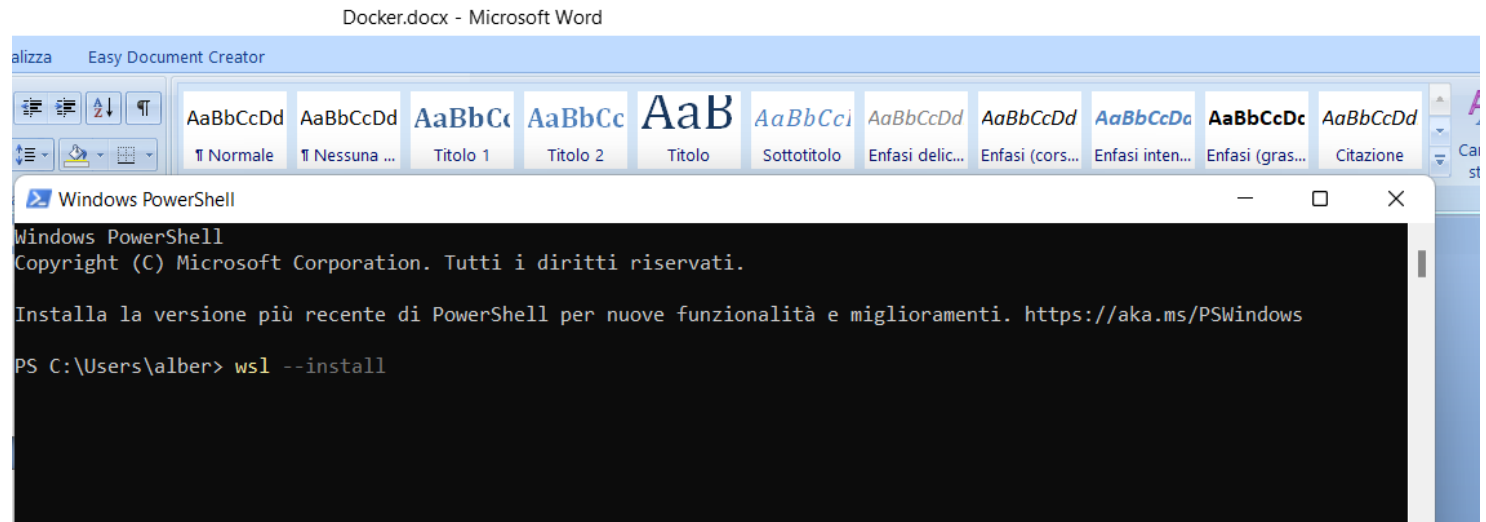
Utilizzando un singolo comando (per Windows 10 versione 2004 e successive, e Windows 11)

Questo è il modo più semplice e veloce:

1. Apri **PowerShell** come amministratore (clicca con il tasto destro sul pulsante Start e seleziona "Windows PowerShell (amministratore)" o "Terminale (amministratore)").

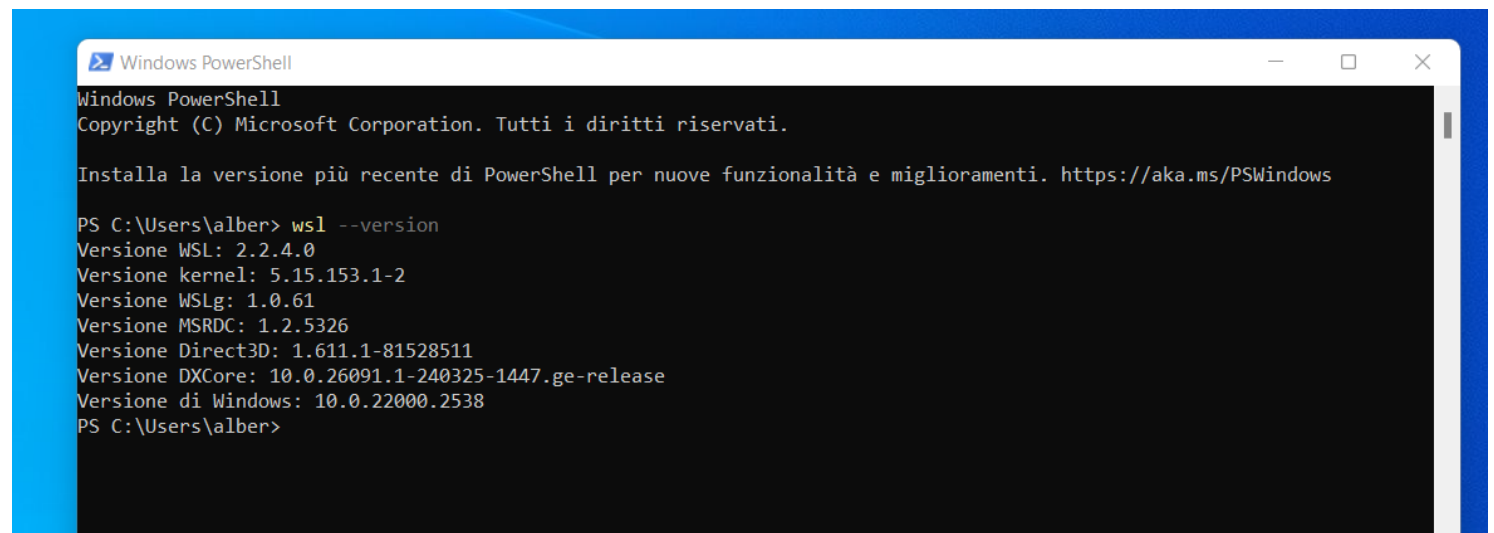
Digita il seguente comando e premi **Invio**:

```
wsl --install
```



Digitare il seguente comando e poi invio per vedere la versione installata

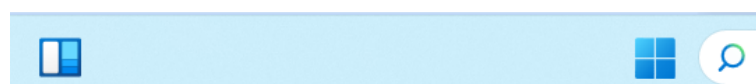
```
wsl --version
```

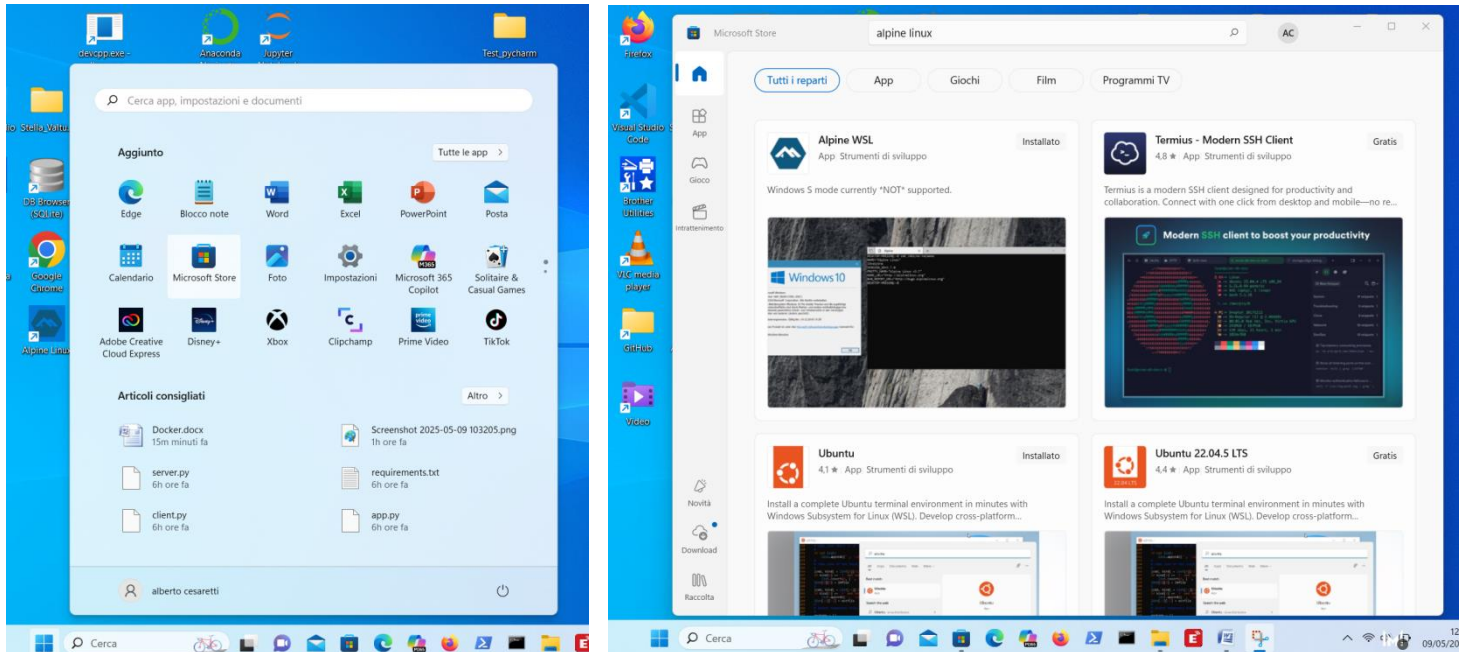


Per controllare la versione

Come installare le distribuzioni Linux su Windows

1. Pulsante di windows
2. Poi **Microsoft Store**
3. Poi si cerca la distribuzione linux da installare





Quando crei un container con Docker Desktop utilizzando WSL e una distribuzione Alpine Linux, e lo lanci con `docker run`, il container viene eseguito direttamente sull'istanza Linux del Windows Subsystem for Linux (WSL) che hai configurato.

Ecco alcuni punti chiave per capire meglio questo processo:

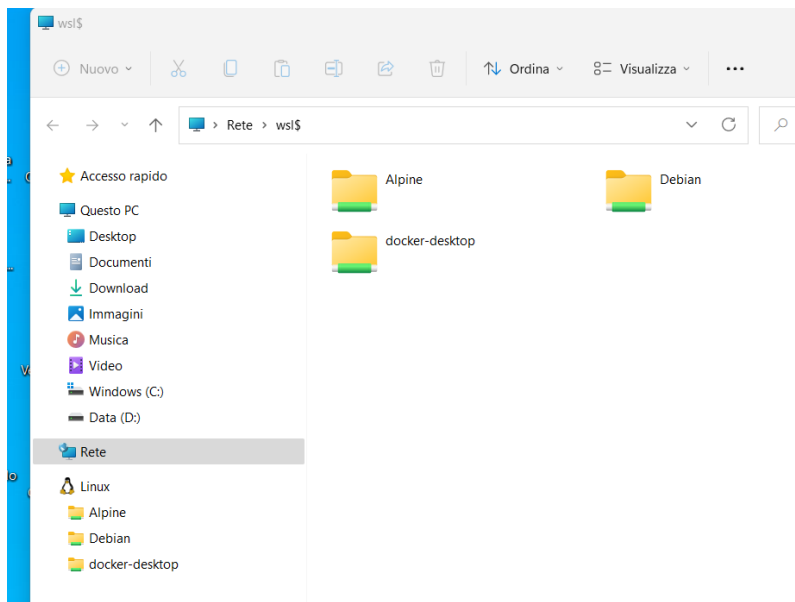
- **Docker Desktop e WSL Integration:** Docker Desktop si integra strettamente con WSL. Quando installi Docker Desktop, ti offre la possibilità di utilizzare il backend WSL. Questo significa che invece di far girare una macchina virtuale Hyper-V completa per gestire i container, Docker sfrutta l'ambiente Linux leggero fornito da WSL.
- **Esecuzione Diretta:** Quando esegui un container, Docker lo avvia all'interno dell'ambiente Linux della tua distribuzione Alpine (o qualsiasi altra distribuzione WSL tu stia usando). Non c'è un ulteriore livello di virtualizzazione pesante coinvolto.
- **Vantaggi di WSL:** L'utilizzo di WSL offre diversi vantaggi in termini di performance e utilizzo di risorse rispetto all'utilizzo di una macchina virtuale tradizionale. I container tendono ad avviarsi più velocemente e consumano meno risorse di sistema perché condividono il kernel del sistema operativo host (Windows) in modo più efficiente.
- **File System:** Il file system all'interno del container interagisce con il file system della tua istanza WSL. Questo facilita la condivisione di file tra il tuo ambiente Windows e i container in esecuzione.

Quindi, per ricapitolare, il container Docker creato con Docker Desktop e lanciato in esecuzione con WSL e Alpine Linux gira **localmente** all'interno della tua istanza Linux di Windows Subsystem for Linux.

File system e distribuzioni linux installate su Windows

Avviare Docker e digitare su cerca di windows `\\wsl\\` poi invio come da figura





Ci sono diversi modi per accedere al file system Linux di WSL 2 da Windows:

1. Tramite Esplora File (il metodo più semplice):

- Nella casella cerca di windows digita **\\wsl\$** e premi Invio.
- Vedrai un elenco delle tue distribuzioni Linux installate. Fai doppio clic sulla distribuzione a cui vuoi accedere (ad esempio, Ubuntu).
- Sarai reindirizzato al file system radice (/) di quella distribuzione. Da qui, puoi navigare attraverso i file e le cartelle come faresti con qualsiasi altra cartella in Windows.

Percorsi utili all'interno del file system Linux:

- La tua cartella home di solito si trova in `/home/<nomeutente>`.
- I file di sistema si trovano in varie cartelle come `/etc`, `/var`, `/usr`, ecc.

2. Utilizzando il comando `explorer.exe` . all'interno di WSL:

- Apri il terminale della tua distribuzione Linux (ad esempio, Ubuntu).
- Naviga fino alla directory Linux a cui vuoi accedere in Esplora File.
- Digita il comando `explorer.exe` . (assicurati di includere il punto finale) e premi Invio.
- Si aprirà una finestra di Esplora File che mostrerà il contenuto della directory Linux corrente.

3. Tramite il percorso di rete `\\wsl.localhost<distro_name>` (Windows 11):

- Se stai utilizzando Windows 11, puoi anche accedere al file system Linux utilizzando il percorso `\\wsl.localhost<nome_della_distribuzione>`. Ad esempio, per Ubuntu, sarebbe `\\wsl.localhost\Ubuntu`. Questo metodo è simile a `\\wsl$` ma utilizza un percorso localhost.

Considerazioni importanti:

- **Permessi:** Quando modifichi i file Linux da Windows, le autorizzazioni potrebbero non essere sempre gestite nello stesso modo. Potresti riscontrare problemi di autorizzazione a seconda di come modifichi i file. È generalmente consigliabile eseguire operazioni che richiedono permessi specifici (come la modifica di file di configurazione di sistema) direttamente all'interno dell'ambiente WSL.
- **Prestazioni:** Per prestazioni ottimali, soprattutto quando lavori con progetti che coinvolgono molti file, è consigliabile archiviare i file all'interno del file system Linux di WSL piuttosto che accedervi tramite il mount del file system Windows (`/mnt/c`).
- **File di sistema:** Fai attenzione quando modifichi i file di sistema Linux da Windows, in quanto potresti inavvertitamente causare problemi al tuo ambiente WSL.

In sintesi, il modo più semplice e comune per accedere al file system Linux di WSL 2 da Windows è tramite il percorso `\\wsl$` in Esplora File. Questo ti offre un'interfaccia grafica familiare per navigare e gestire i tuoi file Linux.

Solitamente, le distribuzioni Linux installate tramite il Windows Subsystem for Linux 2 (WSL 2) vengono archiviate all'interno di un disco rigido virtuale (VHDX). La posizione predefinita di questo file VHDX è all'interno della cartella AppData del tuo profilo utente.

Il percorso tipico è:

```
C:\Users\<Il tuo nome utente>\AppData\Local\Packages\<Nome del pacchetto della distribuzione>\LocalState\
```


All'interno di questa cartella LocalState, troverai un file chiamato ext4.vhdx. Questo file è essenzialmente l'intero filesystem della tua distribuzione Linux.

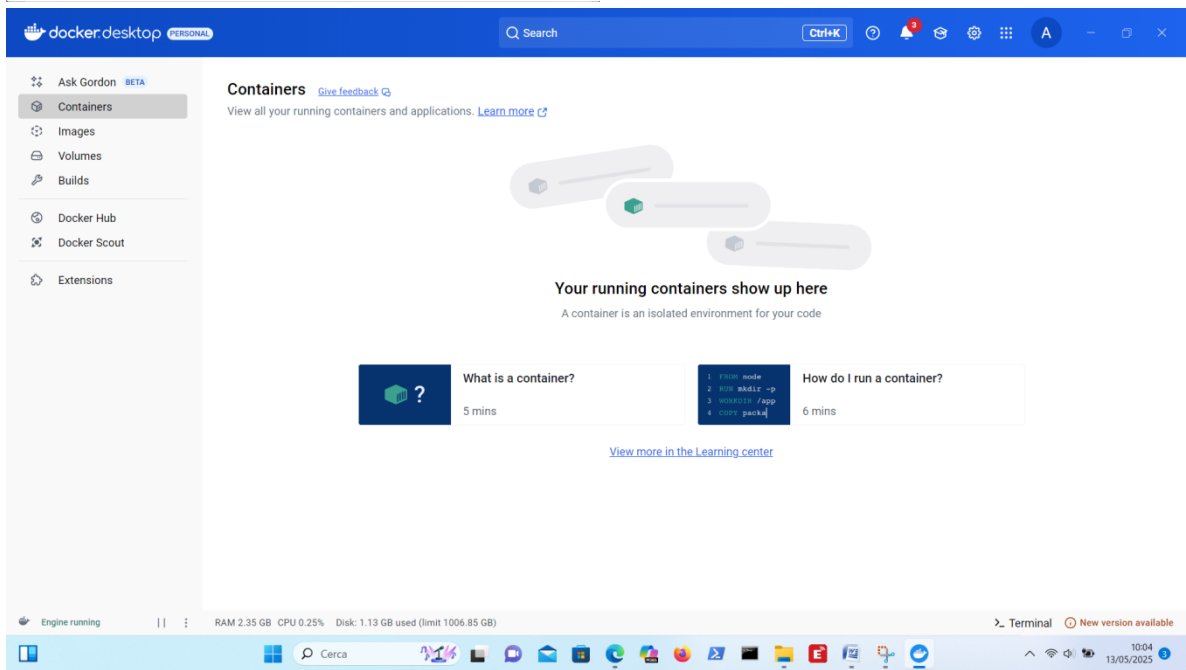
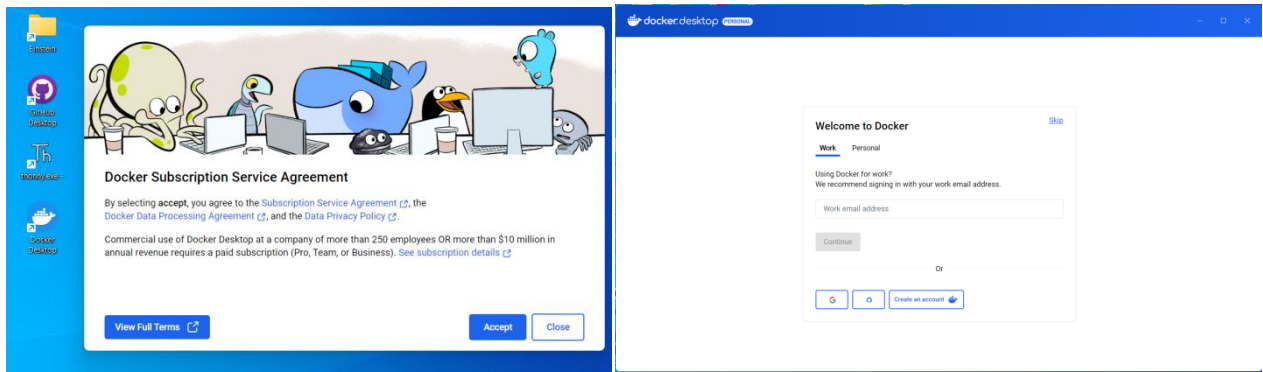
Ad esempio, per Ubuntu, il percorso potrebbe essere simile a:

```
C:\Users\<Il tuo nome  
utente>\AppData\Local\Packages\CanonicalGroupLimited.UbuntuonWindows_79rhkplfndgsc\  
LocalState\
```

Importante:

- La cartella AppData è nascosta per impostazione predefinita. Per visualizzarla in Esplora file, dovrai abilitare l'opzione "Mostra elementi nascosti" nella scheda "Visualizza".
- Non è consigliabile modificare direttamente i file all'interno del file ext4.vhdx tramite strumenti di Windows, in quanto ciò potrebbe causare problemi al tuo ambiente WSL. Interagisci con i file della tua distribuzione Linux tramite il terminale WSL.

Installare Docker



Containers

[Give feedback](#)

View all your running containers and applications.

[Learn more](#)

Container CPU usage

No containers are running.

Container memory usage

No containers are running.

Show charts

Search

Only show running containers

	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	welcome-to-doc	050316adac84	docker/welc	8088:80	N/A	0 seconds ago	<div></div> <div></div> <div></div>

Showing 1 item

Walkthroughs

Multi-container applications

8 mins

Containerize your application

3 mins

[View more in the Learning center](#)

RAM 2.48 GB CPU 0.37% Disk: 1.13 GB used (limit 1006.85 GB)

Terminal

New version available

What is a container?

Estimated time: 5 mins

1 Containers on Docker Desktop

The best way to learn about containers is to first see it in action. We have created a welcome container for you.

You can check it out in the Containers tab (welcome-to-docker).

Back

Next

2 View the frontend

3 Explore your container

4 Stop your container

5 What's next

Linux

Nuovo

Ordina

Visualizza

← → ↕

Linux

Cerca in Linux

Accesso rapido

Questo PC

Desktop

Documenti

Download

Immagini

Musica

Video

Windows (C:)

Data (D:)

Rete

Linux

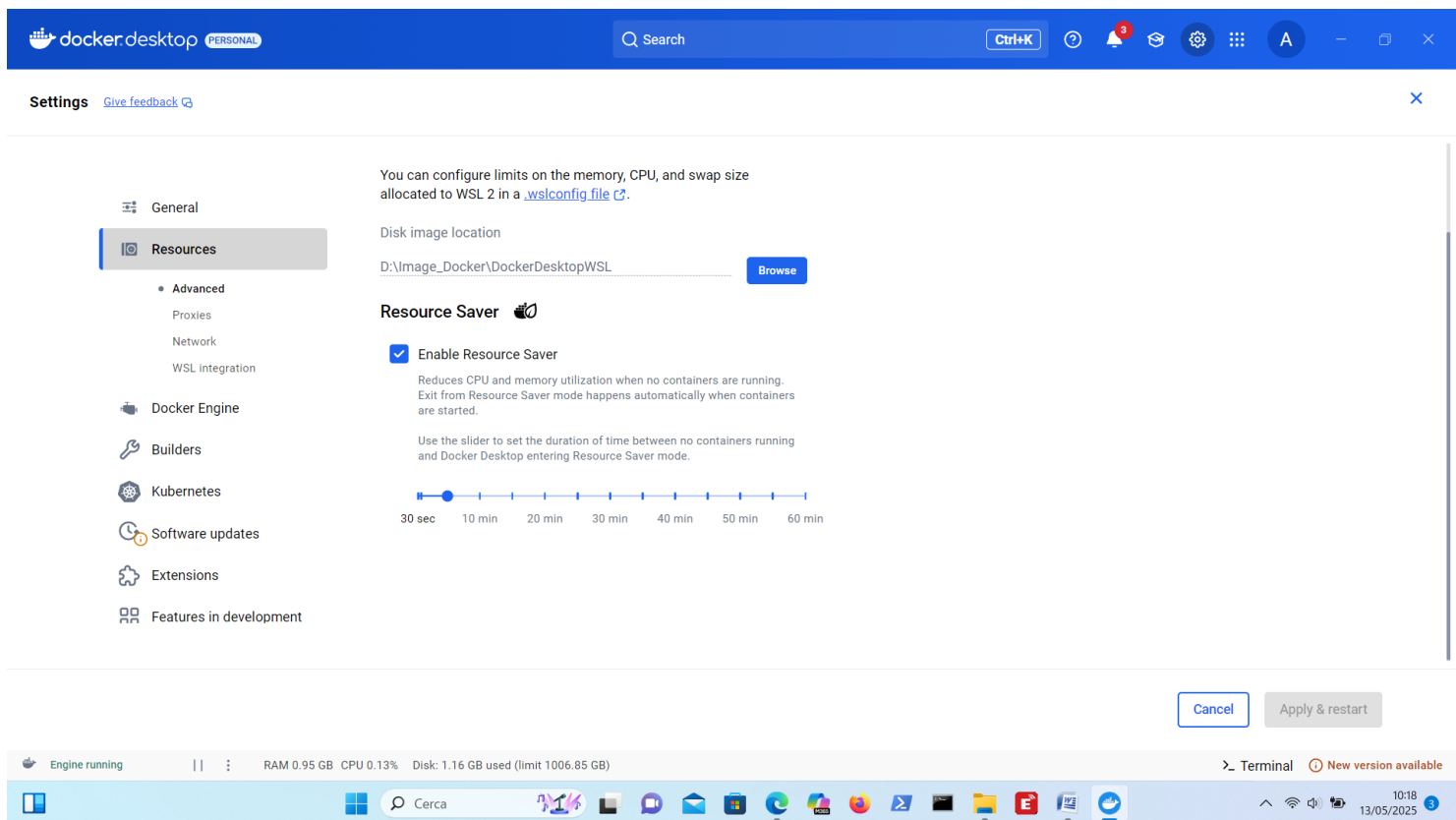
Alpine

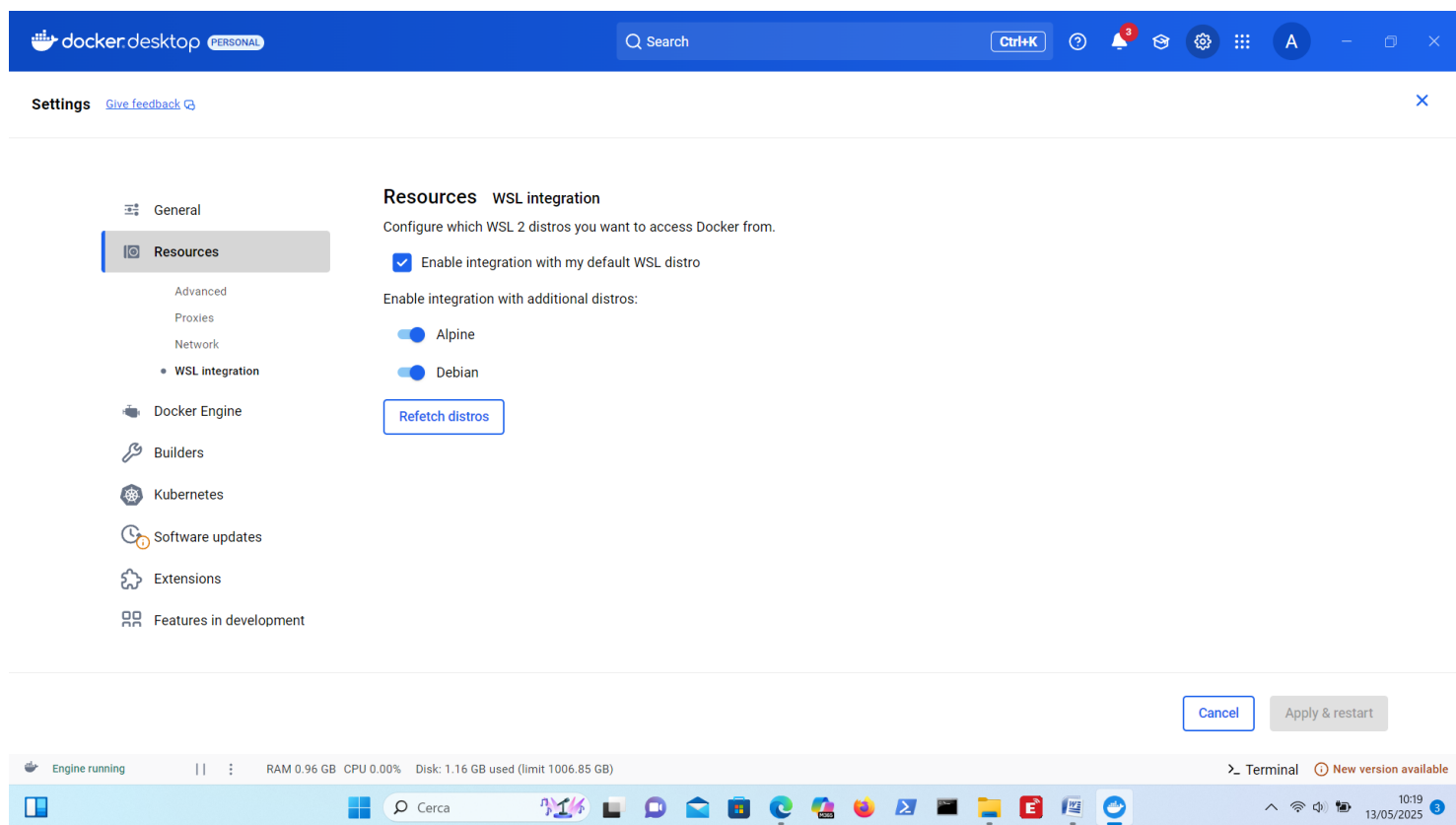
Debian

Alpine

Debian

docker-desktop





Come installare Docker : in un sistema Windows. Tieni presente che il processo comporta alcuni passaggi che devono essere eseguiti con attenzione.

Prerequisiti:

- **Windows 10 o 11 64-bit:** Docker Desktop richiede una versione a 64 bit di Windows 10 (Pro, Enterprise o Education build 1903 o successiva) o Windows 11.
- **Virtualizzazione abilitata nel BIOS:** Assicurati che la virtualizzazione hardware (Intel VT-x o AMD-V) sia abilitata nelle impostazioni del BIOS del tuo computer. Solitamente è abilitata di default, ma potrebbe essere necessario verificarla.
- **Pacchetto di funzionalità Hyper-V e Strumenti di gestione di Hyper-V:** Questi componenti di Windows devono essere abilitati.

Passaggi per l'installazione:

1. **Scarica Docker Desktop:**
 - Apri il tuo browser web e vai al sito ufficiale di Docker: <https://www.docker.com/products/docker-desktop/>
 - Fai clic sul pulsante "Download for Windows".
2. **Esegui l'Installer:**
 - Una volta scaricato il file `Docker Desktop Installer.exe`, fai doppio clic su di esso per avviare l'installazione.
3. **Configurazione dell'Installazione:**
 - Durante l'installazione, ti verrà chiesto di configurare alcuni aspetti. **È fondamentale deselezionare l'opzione predefinita che installa Docker nel percorso di sistema (unità C:).**
 - **Cerca un'opzione di installazione personalizzata o avanzata.** L'interfaccia dell'installer potrebbe variare leggermente a seconda della versione, ma cerca un modo per specificare il percorso di installazione.
 - **Se trovi l'opzione per cambiare il percorso di installazione, seleziona l'unità D:\ (o qualsiasi altra cartella desideri all'interno di D:) come destinazione.**

- Assicurati che le caselle relative a "Installazione di WSL 2" e "Aggiungi Docker alla variabile PATH" siano selezionate (se presenti e se non hai già WSL 2 installato). WSL 2 è il backend raccomandato per Docker Desktop su Windows.

4. Completa l'Installazione:

- Segui le istruzioni a schermo per completare l'installazione. Potrebbe essere necessario concedere autorizzazioni di amministratore durante il processo.
- Una volta completata l'installazione, fai clic su "Close and Restart" (o un'opzione simile) per riavviare il computer.

5. Verifica dell'Installazione:

- Dopo il riavvio, cerca "Docker Desktop" nel menu Start e avvialo.
- Dovrebbe comparire l'icona di Docker nella barra delle applicazioni.
- Apri il Prompt dei comandi o PowerShell e digita il seguente comando:

```
Bash
docker --version
```

- Se Docker è installato correttamente, dovresti visualizzare la versione di Docker.

Considerazioni Importanti:

- **Percorso di Installazione:** Anche se installi l'applicazione Docker Desktop sull'unità D:, alcune configurazioni e file virtuali potrebbero comunque essere creati sull'unità C: (ad esempio, immagini Docker, volumi). Questo è il comportamento predefinito di Docker Desktop.
- **WSL 2 (Windows Subsystem for Linux 2):** Docker Desktop per Windows utilizza WSL 2 come backend predefinito, che offre prestazioni migliori rispetto al precedente Hyper-V. Se non hai WSL 2 installato, l'installer di Docker Desktop dovrebbe offrirti di installarlo. Potrebbe essere necessario abilitare manualmente la funzionalità WSL in Windows prima dell'installazione. Puoi farlo cercando "Attiva o disattiva funzionalità di Windows" nel menu Start e selezionando "Sottosistema Windows per Linux".
- **Spostare i Dati Docker (Avanzato):** Se desideri spostare la posizione predefinita in cui Docker memorizza immagini, volumi e altri dati, dovrai configurare le impostazioni di Docker Desktop dopo l'installazione. **Puoi trovare queste impostazioni nel menu dell'applicazione Docker Desktop (icona nella barra delle applicazioni -> Settings -> Resources -> Advanced).** Fai attenzione quando modifichi queste impostazioni, in quanto configurazioni errate possono causare problemi.

Su Docker Desktop, la posizione predefinita in cui vengono salvati i container creati dipende dal sistema operativo che stai utilizzando:

Su Windows:

I container e le immagini Docker sono memorizzati in diverse posizioni, a seconda della configurazione e se stai utilizzando container Linux o Windows.

- **Container Linux (con WSL 2):** Se stai utilizzando il backend WSL 2 (Windows Subsystem for Linux 2), i file dei container si trovano all'interno del file system Linux della tua distribuzione WSL. Docker Desktop memorizza un disco rigido virtuale contenente questi dati nella seguente posizione (potrebbe variare leggermente):
`\\wsl$\\docker-desktop-data\\version-pack-data\\community\\docker\\`
 All'interno di questa posizione, i file system dei container si trovano tipicamente in directory come overlay2. I volumi Docker si trovano in una sottocartella volumes.
- **Container Windows (Hyper-V):** Se stai utilizzando Hyper-V come backend, le immagini e i livelli dei container Windows sono generalmente memorizzati in:
`C:\\ProgramData\\Docker`
 Questi sono suddivisi tra le directory image e windowsfilter.

Anche in questo caso, **non è consigliabile modificare direttamente questi file**. Utilizza i comandi Docker per gestire i tuoi container e i loro dati.

In sintesi:

- **Linux (WSL 2):** All'interno del file system WSL 2 (\wsl\$\docker-desktop-data\version-pack-data\community\docker\).
- **Windows (Hyper-V):** In C:\ProgramData\Docker.

Ricorda che la gestione dei dati dei container dovrebbe avvenire principalmente tramite i volumi Docker e i bind mount, piuttosto che accedendo direttamente ai file sottostanti.

La macchina virtuale linux docker-desktop

Quando installi Docker Desktop su Windows con WSL 2 abilitato, Docker crea e gestisce una o più distribuzioni Linux leggere all'interno di WSL. Una di queste distribuzioni è specificamente dedicata all'esecuzione del Docker Engine e delle sue funzionalità di backend. Solitamente questa distribuzione si chiama docker-desktop e un'altra per i dati è docker-desktop-data.

Come interagire con questa istanza Linux?

Non interagisci direttamente con l'istanza docker-desktop come faresti con una normale macchina virtuale tramite una shell interattiva (come SSH). Questa istanza è gestita internamente da Docker Desktop per far funzionare i container.

L'interazione principale avviene tramite la riga di comando (CLI) di Docker, eseguita direttamente dal tuo terminale di Windows (PowerShell o CMD) o da una shell WSL.

Docker Desktop configura l'ambiente in modo che il client Docker (il comando docker) in Windows possa comunicare con il Docker Engine in esecuzione nell'istanza Linux di WSL.

Esempi di interazione tramite CLI:

Ecco alcuni esempi di comandi Docker che eseguirai dal tuo terminale di Windows o da una shell WSL per interagire con Docker, il quale a sua volta utilizzerà l'istanza Linux in background:

1. Verificare la versione di Docker:

```
docker version
```

Questo comando ti mostrerà le informazioni sul client Docker installato su Windows e sul Docker Engine in esecuzione nell'istanza Linux di WSL.

2. Scaricare un'immagine Docker:

```
docker pull ubuntu:latest
```

Questo comando chiede al Docker Engine (in esecuzione su WSL) di scaricare l'ultima versione dell'immagine Ubuntu da Docker Hub.

3. Elencare le immagini Docker scaricate:

```
docker images
```

Questo comando mostra le immagini Docker presenti sul tuo sistema (gestite dall'istanza Linux di WSL).

4. Eseguire un container:

```
docker run -it --name mio-ubuntu ubuntu:latest /bin/bash
```

Questo comando crea ed esegue un nuovo container basato sull'immagine Ubuntu.

- **-it: rende l'esecuzione interattiva e ti connette alla shell del container.**
- **--name mio-ubuntu:** assegna un nome al container.
- **/bin/bash:** specifica il comando da eseguire all'interno del container (in questo caso, una shell Bash).

Quando esegui questo comando, il Docker Engine nell'istanza Linux di WSL crea e avvia il container.

5. Elencare i container in esecuzione:

```
docker ps
```

Questo comando mostra i container attualmente in esecuzione (gestiti dall'istanza Linux di WSL).

6. Fermare un container:

```
docker stop mio-ubuntu
```

Questo comando ferma il container con il nome mio-ubuntu.

7. Rimuovere un container:

```
docker rm mio-ubuntu
```

Questo comando rimuove il container con il nome mio-ubuntu (deve essere fermo per poter essere rimosso).

8. Accedere alla shell di un container in esecuzione:

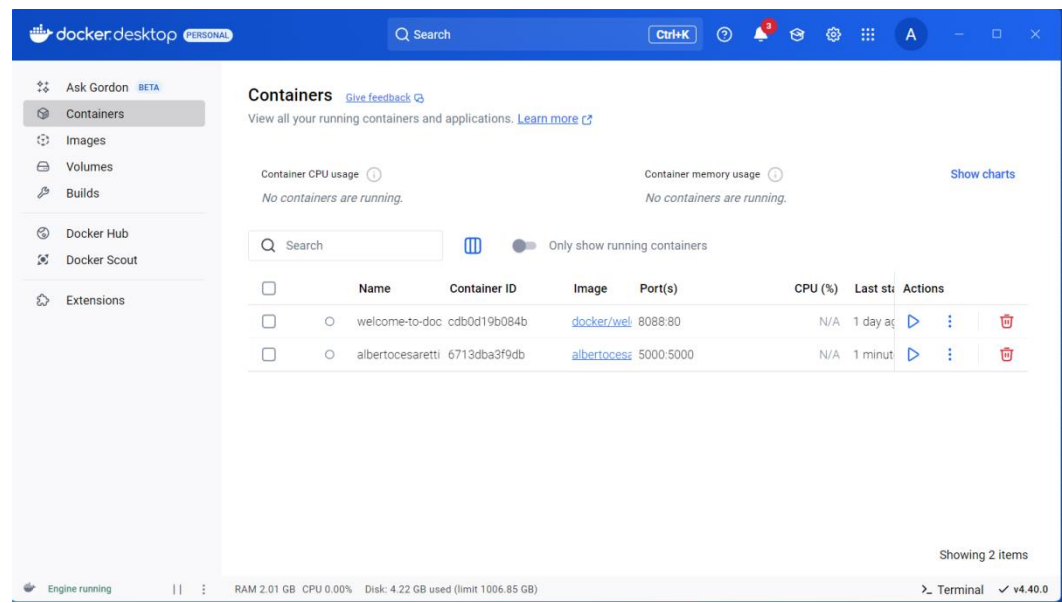
```
docker exec -it <nome_o_ID_container> /bin/bash
```

Sostituisci <nome_o_ID_container> con il nome o l'ID del container a cui vuoi accedere. Questo ti permette di eseguire comandi all'interno del container.

In sintesi, l'interazione con l'ambiente Docker gestito da Docker Desktop su WSL avviene principalmente tramite i comandi docker eseguiti nel tuo terminale Windows o in una shell WSL. Docker Desktop si occupa di far comunicare questi comandi con il Docker Engine in esecuzione nell'istanza Linux dedicata. Non hai bisogno di "entrare" direttamente nell'istanza docker-desktop per gestire i tuoi container e le tue immagini. Tutto il necessario è accessibile tramite la CLI di Docker.

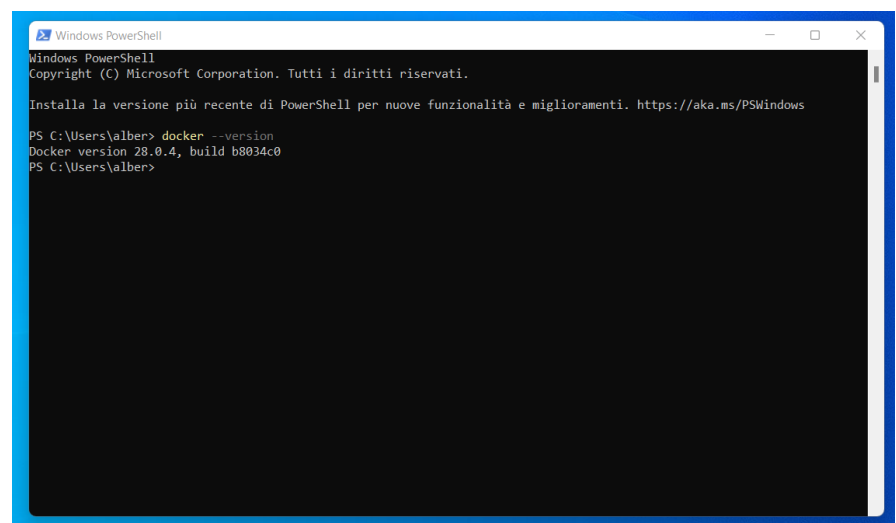
Docker desktop GUI (interfaccia grafica)

Rappresenta l'interfaccia grafica



Docker desktop CLI (interfaccia a riga di comando)

La CLI (Command Line Interface) di Docker Desktop è fondamentale per interagire con i container e le immagini Docker.



Ecco come puoi fare e alcuni esempi pratici:

Come usare la CLI di Docker Desktop:

Docker Desktop, una volta installato, ti fornisce l'eseguibile docker che puoi utilizzare direttamente dal tuo terminale (PowerShell o Prompt su Windows, Terminale su macOS e Linux). Non devi fare nulla di speciale per "usare la CLI di Docker Desktop" se non aprire il tuo terminale preferito.

Esempi pratici:

Ecco alcuni comandi Docker di base con esempi per iniziare:

1. Verificare l'installazione e la versione di Docker:

```
docker --version
```

Questo comando ti mostrerà la versione di Docker installata sul tuo sistema.

2. Eseguire un container da un'immagine:

```
docker run -d -p 8080:80 nginx
```

- o `docker run`: Questo comando avvia un nuovo container.
- o `-d`: Esegue il container in background (detached mode).
- o `-p 8080:80`: Mappa la porta 80 del container (dove il server Nginx è in ascolto) alla porta 8080 della tua macchina host. Potrai accedere a Nginx tramite il tuo browser all'indirizzo `http://localhost:8080`.
- o `nginx`: L'immagine Docker da cui creare il container (in questo caso, l'immagine ufficiale di Nginx).

3. Elencare i container in esecuzione:

```
docker ps
```

Questo comando mostra tutti i container attualmente in esecuzione. Vedrai informazioni come l'ID del container, l'immagine utilizzata, i comandi eseguiti, lo stato e le porte mappate.

4. Elencare tutti i container (anche quelli fermi):

```
docker ps -a
```

Aggiungendo l'opzione `-a`, vedrai anche i container che sono stati creati ma non sono attualmente in esecuzione.

5. Scaricare un'immagine Docker:

```
docker pull ubuntu:latest
```

- o `docker pull`: Questo comando scarica un'immagine Docker da Docker Hub (il registro predefinito).
- o `ubuntu`: Il nome dell'immagine che vuoi scaricare.
- o `:latest`: Un tag che specifica la versione dell'immagine (in questo caso, l'ultima versione stabile di Ubuntu). Se non specifichi un tag, Docker assumerà `:latest`.

6. Elencare le immagini Docker locali:

```
docker images
```

Questo comando mostra tutte le immagini Docker che sono state scaricate e memorizzate localmente sulla tua macchina.

7. Fermare un container in esecuzione: Per fermare un container, devi conoscerne l'ID (lo trovi con `docker ps`).

```
docker stop <ID_DEL_CONTAINER>
```

Sostituisci `<ID_DEL_CONTAINER>` con l'ID effettivo del container che vuoi fermare. Puoi usare anche i primi pochi caratteri dell'ID (solitamente sono sufficienti per identificarlo univocamente).

8. Rimuovere un container fermo: Dopo aver fermato un container, puoi rimuoverlo (eliminarlo completamente) con il comando `docker rm`, sempre usando l'ID del container:

```
docker rm <ID_DEL_CONTAINER>
```

Non puoi rimuovere un container se è ancora in esecuzione.

9. Rimuovere un'immagine Docker: Per rimuovere un'immagine Docker dalla tua macchina, usa il comando `docker rmi` seguito dall'ID o dal nome dell'immagine (eventualmente con il tag):

```
docker rmi <ID_O_NOME_IMMAGINE>
```

Consigli utili:

- **Leggi la documentazione:** La documentazione ufficiale di Docker è una risorsa preziosa: <https://docs.docker.com/>
- **Usa l'autocompletamento:** Molti terminali supportano l'autocompletamento per i comandi Docker (premendo il tasto Tab). Questo può farti risparmiare molto tempo e ridurre gli errori di battitura.

Esplora le opzioni dei comandi: La maggior parte dei comandi Docker ha diverse opzioni. Usa `--help` dopo un comando per vedere la sua sintassi e le opzioni disponibili (ad esempio, `docker run --help`).

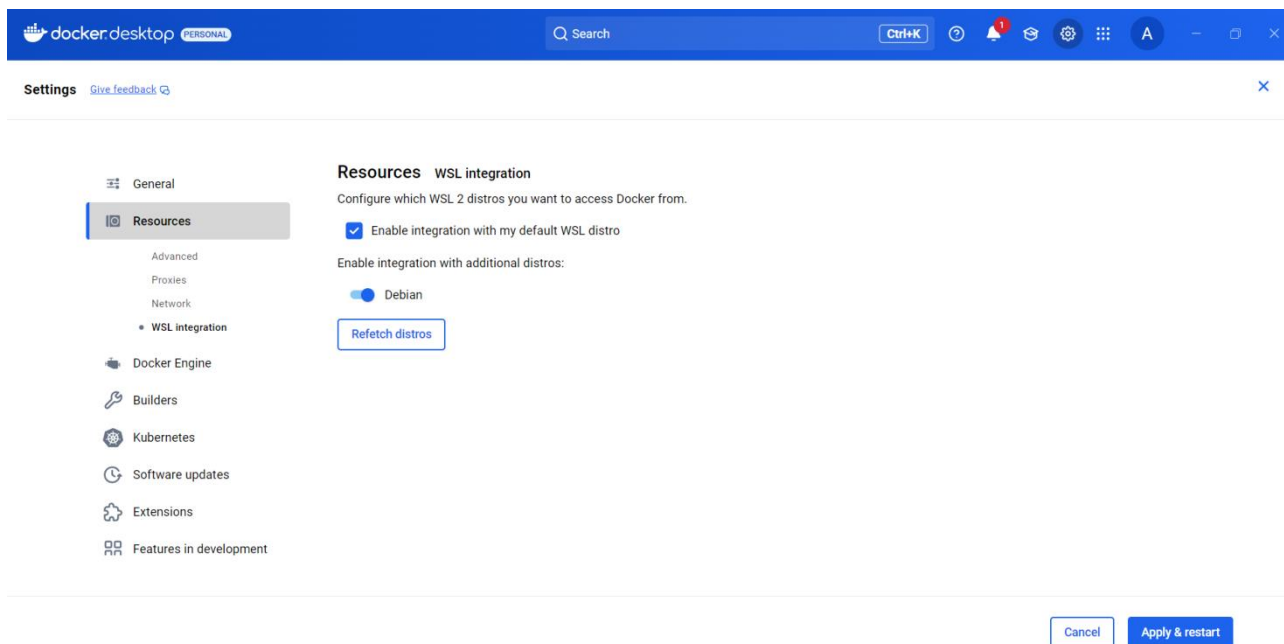
Abilitare l' integrazione Docker con WSL 2

Come abilitare l'integrazione WSL 2 con Docker Desktop:

Durante l'installazione di Docker Desktop, ti verrà chiesto se desideri abilitare l'integrazione con WSL 2. Assicurati che questa opzione sia selezionata.

Se l'hai saltata o vuoi verificarla in seguito:

1. **Apri Docker Desktop.**
2. **Vai su Settings (l'icona a forma di ingranaggio).**
3. **Seleziona la scheda "Resources".**
4. **Seleziona la sottoscheda "WSL Integration".**
5. **Vedrai un elenco delle tue distribuzioni WSL 2 installate.** Assicurati che la casella di controllo accanto alla distribuzione che vuoi utilizzare con Docker sia **selezionata**.
6. **Clicca su "Apply & Restart".**



Dopo aver abilitato l'integrazione, potrai aprire il terminale della tua distribuzione Linux WSL 2 e utilizzare i comandi docker, docker-compose, ecc., come se Docker fosse installato localmente. In realtà, questi comandi comunicheranno con il Docker Engine in esecuzione su WSL tramite l'integrazione fornita da Docker Desktop.

Il flusso di comunicazione quando Docker Desktop è in esecuzione su Windows e utilizza il backend WSL 2. Il percorso che hai descritto è esattamente quello:

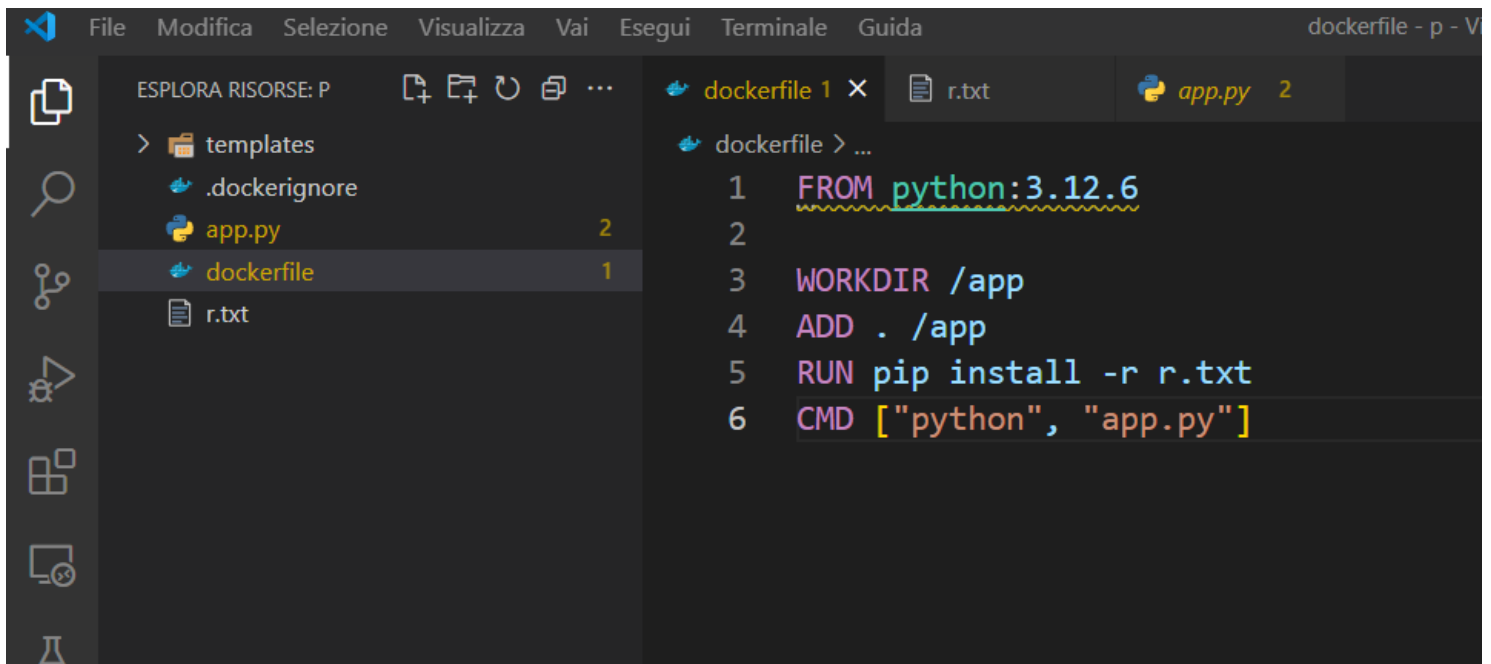
1. **Docker Desktop:** L'applicazione che vedi e con cui interagisci. Fornisce un'interfaccia utente (GUI) e strumenti a riga di comando (CLI) per gestire i tuoi container, immagini, volumi, ecc.
2. **Docker Engine (dockerd):** Il demone Docker vero e proprio. È il processo in background responsabile della creazione ed esecuzione dei container, della gestione delle immagini, delle reti e dello storage. Quando dai un comando Docker tramite Docker Desktop o la CLI, Docker Desktop comunica con il Docker Engine.
3. **WSL 2 (Windows Subsystem for Linux versione 2):** Un ambiente Linux completo in esecuzione su Windows. Docker Desktop si integra con WSL 2 per eseguire il Docker Engine all'interno di una distribuzione Linux leggera e ottimizzata.
4. **Istanze Linux (Distribuzioni Linux in WSL 2):** Le distribuzioni Linux (come Ubuntu, Debian, ecc.) che hai installato all'interno di WSL 2. Il Docker Engine viene eseguito all'interno di una di queste istanze Linux (solitamente una distribuzione Docker dedicata). I container che crei vengono eseguiti all'interno di questo ambiente Linux.

Quindi, in sintesi, il flusso è:

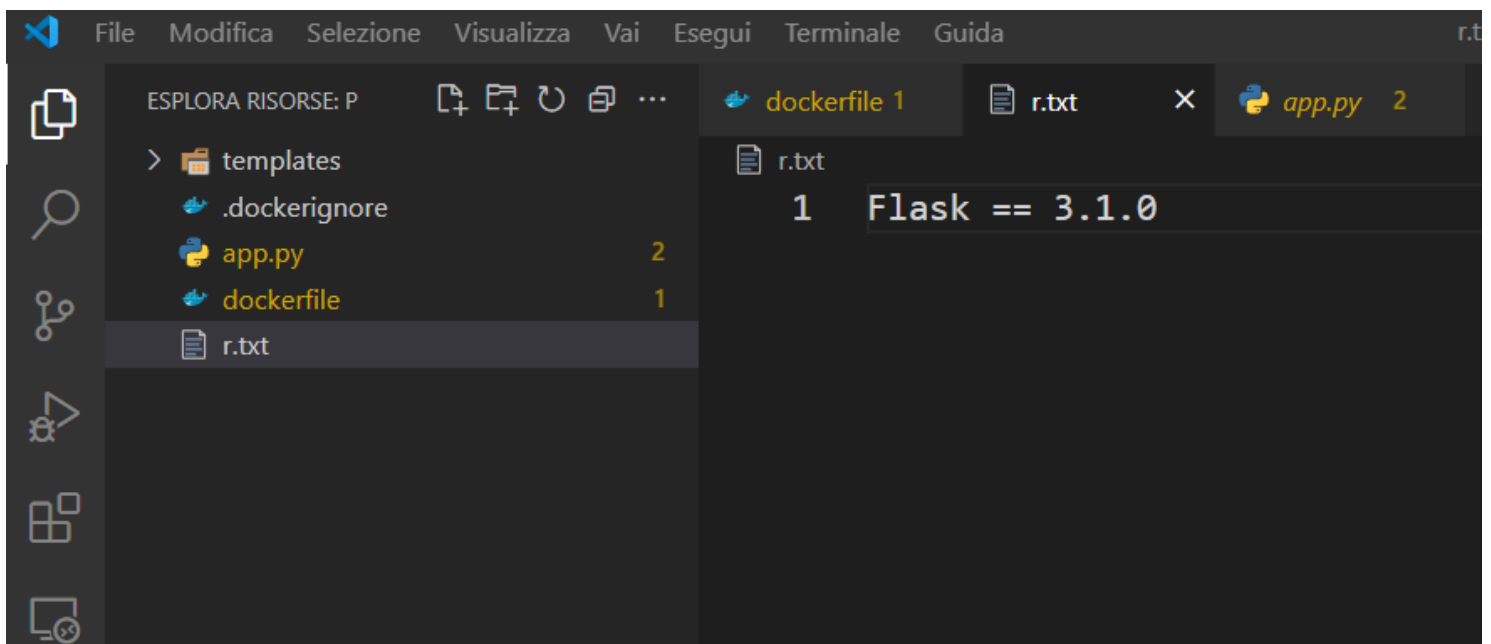
Tu (tramite Docker Desktop o CLI) → Docker Desktop → Docker Engine (in esecuzione in WSL 2) → Istanze Linux (dove i container vengono eseguiti)

Questa architettura offre diversi vantaggi, tra cui migliori prestazioni rispetto alle precedenti versioni di Docker su Windows che utilizzavano una macchina virtuale Hyper-V più pesante. WSL 2 permette al Docker Engine di sfruttare il kernel Linux nativo, rendendo le operazioni di containerizzazione più veloci ed efficienti.

```
app.py
1 from flask import Flask
2 app=Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return 'Hello World'
7
8 if __name__=='__main__':
9     app.run(host='0.0.0.0')
```



```
dockerfile
1 FROM python:3.12.6
2
3 WORKDIR /app
4 ADD . /app
5 RUN pip install -r r.txt
6 CMD ["python", "app.py"]
```



```
r.txt
1 Flask == 3.1.0
```

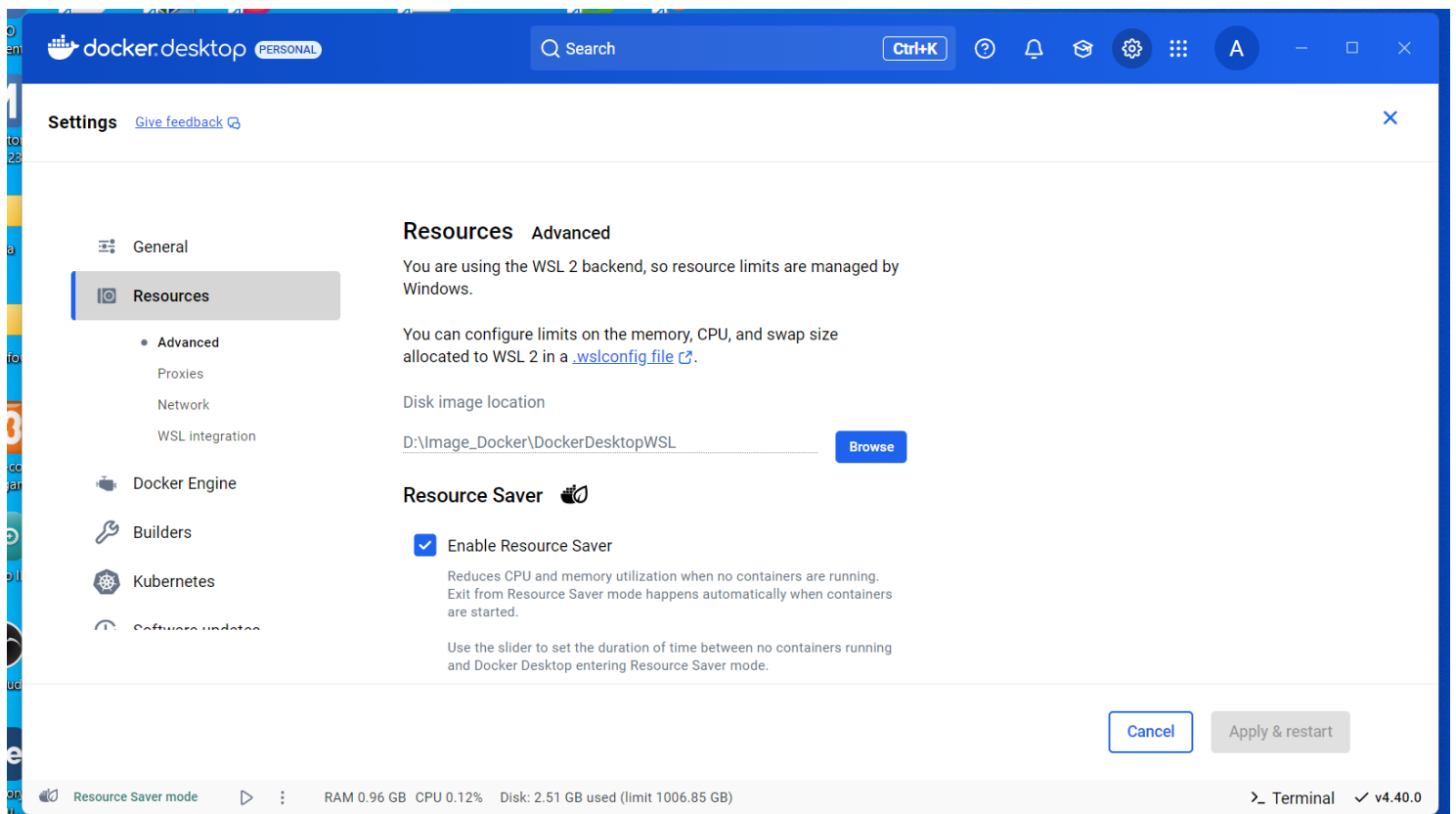
Impostare la cartella dei container

Per impostare la cartella in cui Docker memorizza i container, le immagini e altri dati persistenti, devi configurare il **data root** di Docker. Ecco come puoi fare, tenendo presente che il metodo principale varia a seconda del sistema operativo:

Su Windows:

Su Windows, la cartella predefinita per i dati di Docker Desktop è solitamente in `C:\ProgramData\DockerDesktop`. Puoi modificare questa posizione tramite l'interfaccia utente di Docker Desktop:

1. **Apri Docker Desktop.**
2. **Fai clic sull'icona dell'ingranaggio (Settings) nella barra di sistema.**
3. **Vai alla sezione "Resources" e poi "Advanced".**
4. **Nel campo "Disk image location", puoi specificare il nuovo percorso desiderato.**
5. **Fai clic su "Apply & Restart".** Docker Desktop sposterà i dati nella nuova posizione e si riavvierà.



- **Python:** Assicurati di avere Python installato sul tuo sistema. Puoi verificarlo aprendo il Prompt dei comandi (digita `cmd` nella barra di ricerca di Windows e premi Invio) e digitando `python --version` o `python3 --version`. Se Python è installato, vedrai la versione. Se non lo è, puoi scaricarlo dal sito ufficiale di Python (<https://www.python.org/downloads/>). Durante l'installazione, assicurati di selezionare l'opzione "Aggiungi Python al PATH" per poterlo usare facilmente dal Prompt dei comandi.
- **pip:** Pip è il gestore di pacchetti per Python e viene generalmente installato automaticamente con le versioni recenti di Python. Puoi verificarlo aprendo il Prompt dei comandi e digitando `pip --version` o `pip3 --version`.

Passaggi per installare Flask:

1. **Apri il Prompt dei comandi:** Come accennato prima, digita `cmd` nella barra di ricerca di Windows e premi Invio.
2. **Utilizza pip per installare Flask:** Nella finestra del Prompt dei comandi, digita il seguente comando e premi Invio:
3. `pip install Flask`

oppure, se hai più versioni di Python installate, potresti dover usare:

```
pip3 install Flask
```

Pip si conatterà al Python Package Index (PyPI) e scaricherà e installerà Flask e le sue dipendenze. Vedrai un output che indica il processo di download e installazione.

4. **Verifica l'installazione (opzionale ma consigliato):** Dopo che l'installazione è completa, puoi verificare che Flask sia stato installato correttamente aprendo l'interprete Python. Digita `python` o `python3` nel Prompt dei comandi e premi Invio. Poi, prova a importare Flask:

Python

```
import flask
print(flask.__version__)
```

Se non ricevi errori e vedi stampata la versione di Flask, l'installazione è andata a buon fine. Per uscire dall'interprete Python, digita `exit()` e premi Invio.

Ah, vedo che hai un Dockerfile! Ottimo modo per assicurarti che il tuo ambiente sia consistente. Sembra che tu stia cercando di installare le dipendenze del tuo progetto Flask all'interno di un container Docker basato su Python 3.12.6.

Analizzando il tuo Dockerfile:

- `FROM python:3.12.6`: Questa istruzione definisce l'immagine base che verrà utilizzata per costruire il tuo container. In questo caso, stai partendo dall'immagine ufficiale di Python nella versione 3.12.6.
- `WORKDIR /app`: Questa imposta la directory di lavoro all'interno del container a `/app`. Tutte le istruzioni successive verranno eseguite a partire da questa directory.
- `ADD . /app`: Questa copia tutti i file e le cartelle dal tuo host (dove si trova il Dockerfile) alla directory `/app` all'interno del container. Questo include il tuo codice sorgente (`app.py` presumibilmente) e il file `requirements.txt`.
- `RUN pip -r install requirements.txt`: Questa è l'istruzione chiave per installare le tue dipendenze. `pip` è il gestore di pacchetti per Python, `-r` indica di leggere le dipendenze da un file (in questo caso `requirements.txt`), e `install requirements.txt` esegue l'installazione di tutte le librerie elencate in quel file. **Se Flask è elencato nel tuo `requirements.txt`, allora verrà installato correttamente all'interno del container.**
- `CMD ["python", "app.py"]`: Questa istruzione specifica il comando da eseguire quando il container viene avviato. In questo caso, eseguirà il tuo script Python principale, `app.py`.

Quindi, per rispondere alla tua domanda implicita su come installare Flask in questo contesto:

Assicurati che `Flask` sia presente nel tuo file `requirements.txt`.

Il file `requirements.txt` è un semplice file di testo in cui ogni riga specifica un pacchetto Python da installare (solitamente con una specifica di versione). Ad esempio, il tuo `requirements.txt` potrebbe contenere una riga come questa per installare l'ultima versione di Flask:

Flask

Oppure, per specificare una versione precisa:

```
Flask==3.0.0
```

Flusso di lavoro tipico:

1. **Crea il tuo file `requirements.txt`:** Elenca tutte le dipendenze del tuo progetto, inclusa Flask, in questo file.
2. **Scrivi il tuo Dockerfile** (come quello che hai mostrato).
3. **Costruisci l'immagine Docker:** Apri il Prompt dei comandi (o il terminale) nella directory dove si trova il tuo Dockerfile ed esegui il comando:

Bash

```
docker build -t nome_della_tua_immagine .
```

Sostituisci `nome_della_tua_immagine` con un nome significativo per la tua immagine Docker (ad esempio, `mio-progetto-flask`). Il `.` alla fine indica che il contesto di build è la directory corrente. Docker leggerà il Dockerfile, costruirà l'immagine e durante il processo eseguirà l'istruzione `RUN pip -r install requirements.txt`, installando Flask (se presente in `requirements.txt`) all'interno dell'immagine.

4. **Esegui il container Docker:** Una volta che l'immagine è stata costruita con successo, puoi eseguire un container basato su quell'immagine con il comando:

Bash

```
docker run -p 5000:5000 nome_della_tua_immagine
```

L'opzione `-p 5000:5000` mappa la porta 5000 del container (la porta predefinita su cui Flask esegue l'applicazione in modalità di sviluppo) alla porta 5000 della tua macchina host.

In sintesi, il tuo Dockerfile è già configurato per installare le dipendenze definite in `requirements.txt`. Devi solo assicurarti che `Flask` sia elencato in quel file.

Docker File che installa su Alpine, python, mongodb

Certo, ecco un Dockerfile che utilizza Alpine Linux, installa Python 3 e MongoDB:

```
FROM alpine:latest
```

```
# Installazione di Python 3 e pip
```

```
RUN apk add --no-cache python3 py3-pip
```

```
# Installazione di MongoDB
```

```
RUN apk add --no-cache mongodb
```

```
# Creazione della directory per i dati di MongoDB
```

```
RUN mkdir -p /data/db
```

```
# Impostazione dei permessi per la directory dei dati (potrebbe essere necessario in alcuni contesti)
```

```
RUN chown -R mongodb:mongodb /data/db
```

```
# Definizione della porta esposta per MongoDB
EXPOSE 27017

# Definizione del volume per la persistenza dei dati
VOLUME /data/db

# Comando per avviare il server MongoDB
CMD ["mongod", "--bind_ip_all", "--dbpath", "/data/db"]
```

Spiegazione del Dockerfile:

1. **FROM alpine:latest:** Indica che l'immagine base da utilizzare è l'ultima versione di Alpine Linux, una distribuzione Linux leggera e orientata alla sicurezza.
2. **RUN apk add --no-cache python3 py3-pip:** Utilizza il gestore di pacchetti apk di Alpine per installare Python 3 e il suo gestore di pacchetti pip. L'opzione --no-cache assicura che la lista degli indici dei pacchetti non venga salvata nella cache all'interno dell'immagine, mantenendola più leggera.
3. **RUN apk add --no-cache mongodb:** Installa il pacchetto mongodb dal repository di Alpine.
4. **RUN mkdir -p /data/db:** Crea la directory /data/db all'interno del container. Questa è la directory predefinita dove MongoDB memorizza i suoi file di dati. L'opzione -p crea anche le directory padre se non esistono.
5. **RUN chown -R mongodb:mongodb /data/db:** Cambia il proprietario e il gruppo della directory /data/db all'utente e al gruppo mongodb. Questo è importante per i permessi di accesso ai file di dati di MongoDB all'interno del container.
6. **EXPOSE 27017:** Dichiarare che la porta 27017 (la porta predefinita di MongoDB) sarà utilizzata dal container. Questa è una dichiarazione di intenti e non pubblica automaticamente la porta all'esterno del container. Dovrai farlo esplicitamente durante l'esecuzione con l'opzione -p di docker run.
7. **VOLUME /data/db:** Definisce un volume Docker nel percorso /data/db. Questo assicura che i dati di MongoDB siano persistenti anche se il container viene eliminato. Quando esegui il container, puoi mappare questo volume a una directory sul tuo host o a un volume Docker named.
8. **CMD ["mongod", "--bind_ip_all", "--dbpath", "/data/db"]:** Specifica il comando che verrà eseguito quando il container viene avviato.
 - o mongod: è il processo del server MongoDB.
 - o --bind_ip_all: fa sì che MongoDB ascolti su tutte le interfacce di rete all'interno del container.
 - o --dbpath, "/data/db": specifica il percorso della directory dei dati.

Come utilizzare questo Dockerfile:

1. Salva il contenuto qui sopra in un file chiamato Dockerfile (senza estensione).
2. Apri il tuo terminale o prompt dei comandi nella stessa directory in cui hai salvato il file.
3. Costruisci l'immagine Docker utilizzando il comando:

```
docker build -t mio-mongodb-python .
```

 (dove mio-mongodb-python è il nome che vuoi dare alla tua immagine).
4. Esegui il container Docker:

```
docker run -d -p 27017:27017 --name mio-mongodb-container -v  
mongodb_data:/data/db mio-mongodb-python
```

 - o -d: esegue il container in background (detached mode).
 - o -p 27017:27017: mappa la porta 27017 del container alla porta 27017 del tuo host.
 - o --name mio-mongodb-container: assegna un nome al container.
 - o -v mongodb_data:/data/db: crea un volume Docker named chiamato mongodb_data e lo monta nella directory /data/db del container per la persistenza dei dati.

Ora avrai un container Docker in esecuzione con Alpine Linux, Python 3 e un server MongoDB pronto all'uso sulla porta 27017 del tuo host.

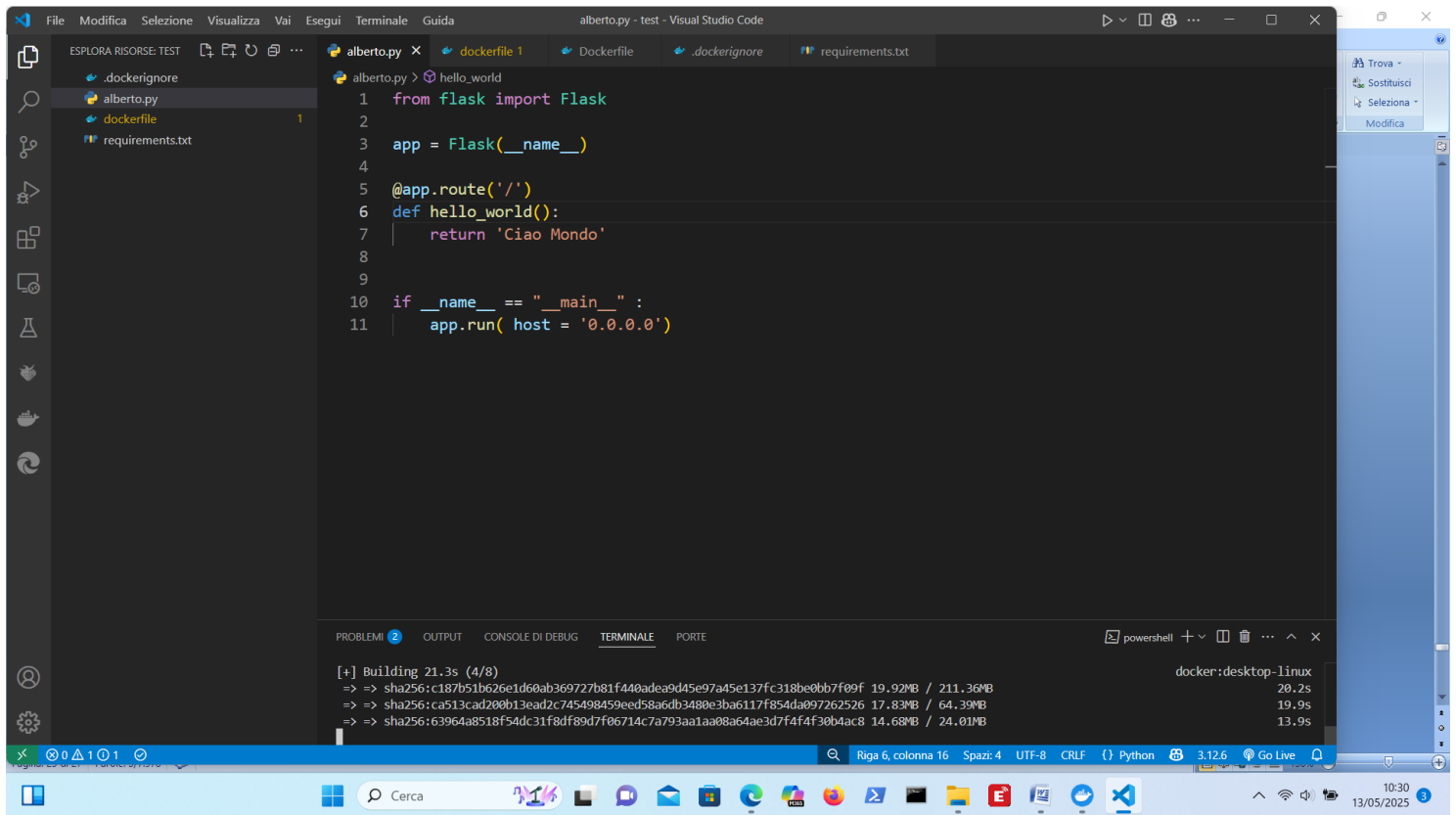
Comandi

```
C:\Users\alber>wsl --distribution Debian --user root
```


PS D:\Einstein\0_Anno_2025_2026\Modulo_12_Docker\app>

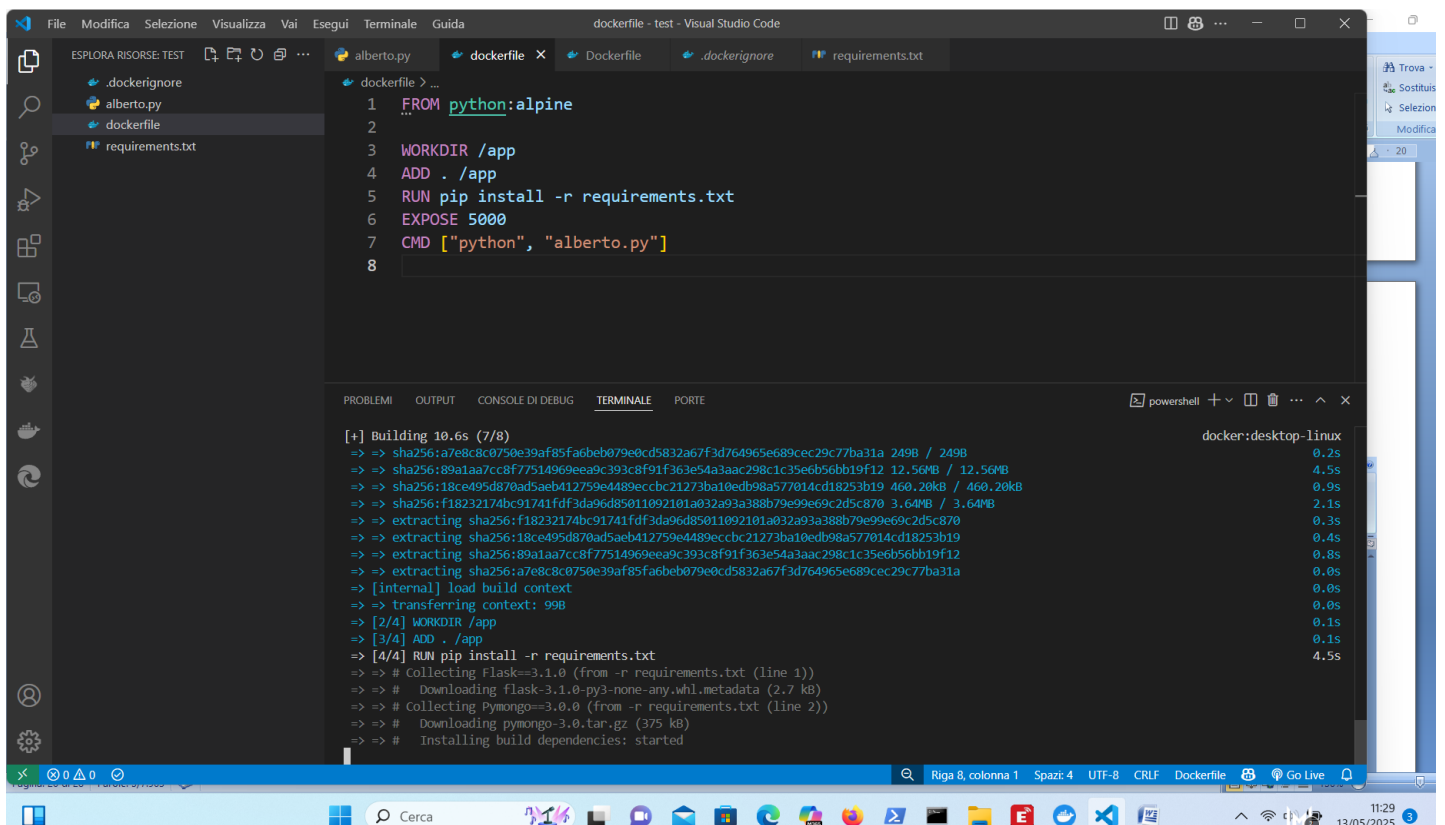
Creare l'immagine

docker build -t albertocesaretti1967/pythonweb1:versione1 .

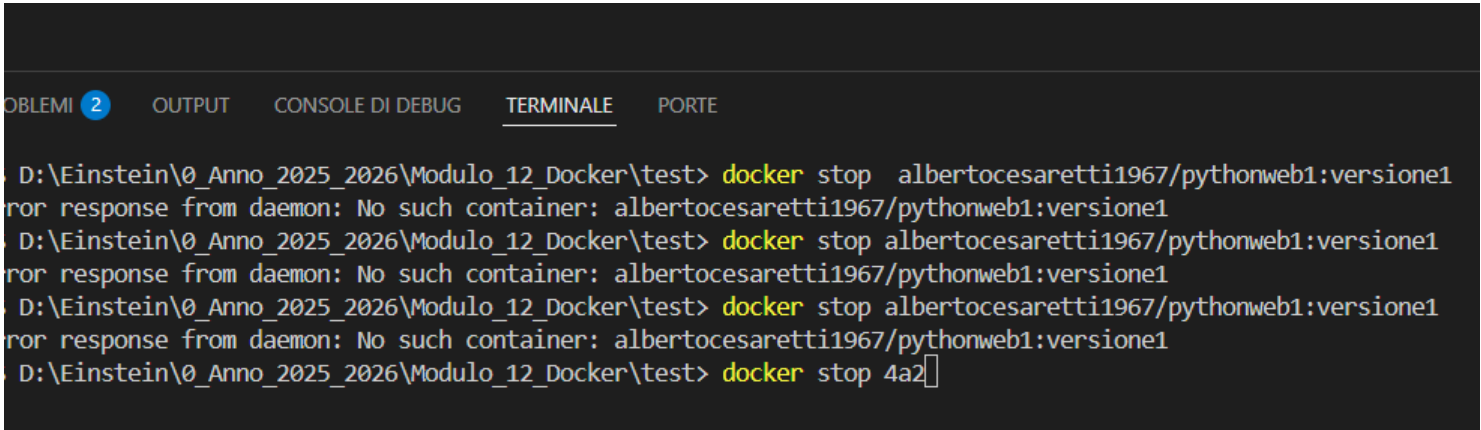
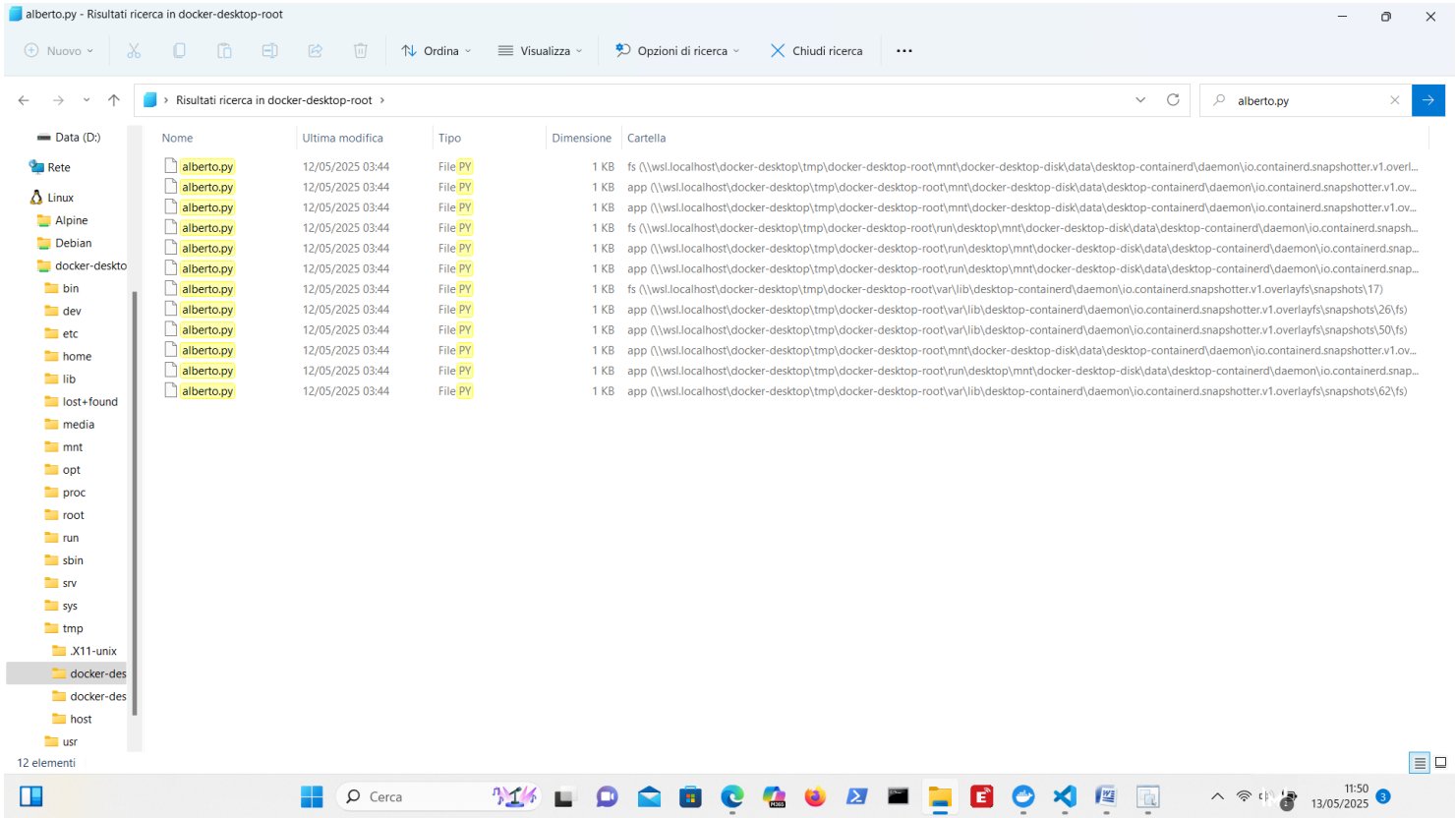


Compilare l'immagine esponendo la porta 8080 del mio PC

docker run -d -p 5000:5000 albertocesaretti1967/pythonweb1:versione1



Immagine



The screenshot shows the Docker Desktop interface. On the left is a sidebar with navigation options: Ask Gordon (BETA), Containers (selected), Images, Volumes, Builds, Docker Hub, Docker Scout, and Extensions. The main area displays the 'web1' container. At the top, it shows the container ID '0b1388c679bd' and the image 'albertocesaretti1967/pythonweb1:versione1'. Below this are tabs for Logs, Inspect, Bind mounts, Exec (active), Files, and Stats. A purple banner for 'Docker Debug' is visible. The terminal output shows the following commands and results:

```
/usr # ls
bin  lib  local sbin share src
/usr # cd ..
/ # ls
app  bin  dev  etc  home lib  media mnt  opt  proc root run  sbin srv  sys  tmp  usr  var
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/community/x86_64/APKINDEX.tar.gz
v3.21.3-480-gb04256b6954 [https://dl-cdn.alpinelinux.org/alpine/v3.21/main]
v3.21.3-478-g842893ceca8 [https://dl-cdn.alpinelinux.org/alpine/v3.21/community]
OK: 25398 distinct packages available
/ # apk add iputils
(1/6) Installing libcap2 (2.71-r0)
(2/6) Installing iputils-arping (20240905-r0)
(3/6) Installing iputils-clockdiff (20240905-r0)
(4/6) Installing iputils-ping (20240905-r0)
(5/6) Installing iputils-tracepath (20240905-r0)
(6/6) Installing iputils (20240905-r0)
Executing busybox-1.37.0-r12.trigger
OK: 11 MiB in 34 packages
```

Docker Hub

Per eliminare un repository da [Docker Hub](https://hub.docker.com/), devi accedere al tuo account, andare alla sezione "My Hub" e poi "Repositories". Seleziona il repository che desideri eliminare, vai all'opzione "Settings" e quindi "Delete repository". Conferma l'eliminazione inserendo il nome del repository e clicca su "Delete Repository Forever"

Come verificare le versioni software delle applicazioni nei container

```
docker exec <nome_o_id_container> nginx -v
```

```
docker exec -it <nome_o_id_container> /bin/bash
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tutti i diritti riservati.

Installa la versione più recente di PowerShell per nuove funzionalità e miglioramenti. https://aka.ms/PSWindows

PS C:\Users\alber> docker exec albertocesaretti1967/pythonweb1 python -V
Error response from daemon: No such container: albertocesaretti1967/pythonweb1
PS C:\Users\alber> docker exec albertocesaretti1967/pythonweb1
docker: 'docker exec' requires at least 2 arguments

Usage: docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

See 'docker exec --help' for more information
PS C:\Users\alber> docker exec albertocesaretti1967/pythonweb1 python3 --version
Error response from daemon: No such container: albertocesaretti1967/pythonweb1
PS C:\Users\alber> docker exec welcome nginx --version
nginx: invalid option: "-"
PS C:\Users\alber> docker exec welcome nginx -v
nginx version: nginx/1.25.3
PS C:\Users\alber>
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLens AZURE
PS C:\temp\repos\docker-tutorial> docker pull nginx
Using default tag: latest
```

/usr/share/nginx/html

[\\wsl.localhost\docker-desktop\tmp\docker-desktop-root\var\lib\desktop-containerd\daemon](#)

\\wsl.localhost\docker-desktop\tmp\docker-desktop-root\var\lib\desktop-containerd\daemon\io.containerd.runtime.v2.task\moby

05d152a38c7e8d4ddc5c42c6b16b20fdcfdc8688e10874a5d42bc0b1d2025e6d

C:\Users\alber>docker tag docker/welcome-to-docker
albertocesaretti1967/albertocesaretti:welcome

docker push albertocesaretti1967/albertocesaretti:tagname

docker pull albertocesaretti1967/albertocesaretti:welcome

Installare un editor di testo CLI

apk add nano

GUI

apk add xfce4 xfce4-terminal thunar

apk add dbus # Necessario per avviare l'ambiente desktop

apk add mousepad

Le reti in Docker

Per prima cosa installare le applicazioni per monitorare le reti, ping e ipconfig

Da terminale:

docker apk add iputils

docker apk add net-tools

Creare una rete in Docker è un'operazione fondamentale per far comunicare i tuoi container in modo isolato e sicuro. Ci sono diversi modi per farlo, a seconda delle tue esigenze.

Vediamo i più comuni:

1. La rete predefinita `bridge`:

Questa è la rete che Docker crea automaticamente quando viene installato. I container che non vengono specificatamente assegnati a un'altra rete vengono collegati a questa rete `bridge`.

- **Come funziona:** Docker assegna a ogni container connesso a questa rete un indirizzo IP all'interno di un sottorete privata. I container sulla stessa rete `bridge` possono comunicare tra loro utilizzando gli indirizzi IP. Per esporre un servizio all'esterno dell'host, devi mappare le porte del container alle porte dell'host utilizzando l'opzione `-p` durante l'esecuzione di `docker run`.
- **Esempio di esecuzione di un container sulla rete `bridge` (implicito):**

Bash

```
docker run -d --name mio-nginx nginx:latest -p 80:80
```

In questo caso, il container `mio-nginx` viene eseguito e collegato alla rete `bridge` predefinita. La porta 80 del container è mappata alla porta 80 dell'host.

2. Creare una rete `bridge` personalizzata:

Creare reti `bridge` personalizzate offre maggiore flessibilità e isolamento rispetto alla rete `bridge` predefinita.

- **Come fare:** Utilizza il comando `docker network create`.
- **Esempio:** Creiamo una rete chiamata `mia-rete`:

Bash

```
docker network create mia-rete
```

- **Come connettere un container a questa rete:** Utilizza l'opzione `--network` durante l'esecuzione di `docker run`.

Bash

```
docker run -d --name mio-web --network mia-rete mio-immagine-web:latest
docker run -d --name mio-db --network mia-rete mio-immagine-db:latest
```

Ora i container `mio-web` e `mio-db` sono sulla stessa rete `mia-rete` e possono comunicare tra loro utilizzando i nomi dei container come hostname (grazie al DNS interno di Docker).

3. Rete `host`:

Questa rete bypassa completamente l'isolamento di rete di Docker per un container. Il container condivide direttamente lo spazio dei nomi di rete dell'host.

- **Quando usarla:** In rari casi in cui le prestazioni di rete sono critiche e l'isolamento non è una preoccupazione primaria.
- **Come usarla:** Utilizza l'opzione `--network host` durante l'esecuzione di `docker run`.

Bash

```
docker run -d --name mio-app --network host mio-immagine-app:latest
```

In questo caso, se l'applicazione nel container ascolta sulla porta 8080, sarà accessibile direttamente sulla porta 8080 dell'host.

- **Importante:** Questa modalità compromette l'isolamento e può portare a conflitti di porte se più container host cercano di utilizzare la stessa porta.

4. Rete none:

Questa rete crea un container che non ha alcuna interfaccia di rete configurata, ad eccezione del loopback (10).

- **Quando usarla:** Per container che non necessitano di accesso alla rete o per eseguire operazioni batch isolate.
- **Come usarla:** Utilizza l'opzione `--network none` durante l'esecuzione di `docker run`.

Bash

```
docker run -d --name mio-task --network none mio-immagine-task:latest
```

Comandi utili per la gestione delle reti Docker:

- `docker network ls`: Elenca tutte le reti Docker disponibili.
- `docker network inspect <nome_rete>`: Mostra informazioni dettagliate su una specifica rete.
- `docker network connect <nome_rete> <nome_container>`: Connette un container esistente a una rete.
- `docker network disconnect <nome_rete> <nome_container>`: Disconnette un container da una rete.
- `docker network rm <nome_rete>`: Rimuove una rete (solo se nessun container è connesso).

In sintesi, per creare una rete in Docker:

1. **Identifica le tue esigenze:** Hai bisogno di isolamento? I container devono comunicare tra loro? Devi esporre porte all'esterno?
2. **Scegli il tipo di rete appropriato:**
 - Usa la rete `bridge` predefinita per casi semplici.
 - Crea reti `bridge` personalizzate per un migliore isolamento e comunicazione interna tra gruppi di container.
 - Considera `host` solo in scenari specifici dove le prestazioni sono cruciali e l'isolamento non è prioritario.
 - Usa `none` per container isolati.
3. **Utilizza `docker network create` per creare reti personalizzate.**
4. **Utilizza l'opzione `--network` con `docker run` per connettere i container alle reti desiderate.**

I Volumi in Docker

I volumi di Docker! Sono un concetto fondamentale per la gestione dei dati nelle tue applicazioni containerizzate. In sostanza, i volumi sono meccanismi per **persistenza dei dati generati e utilizzati dai container Docker**. Immagina un container come un ambiente isolato ed effimero. Quando un container viene creato, esegue i suoi processi e può generare o modificare file. Tuttavia, quando il container viene **eliminato**, tutte le modifiche apportate al suo filesystem interno **vengono perse**.

È qui che entrano in gioco i volumi. Essi forniscono un modo per **separare il ciclo di vita dei dati dal ciclo di vita del container**. In altre parole, i dati memorizzati in un volume persistono anche se il container viene fermato, riavviato o eliminato.

Ci sono principalmente tre tipi di volumi in Docker:

- **Named Volumes (Volumi Nominati):** Questi sono volumi gestiti direttamente da Docker. Hanno un nome specifico (che puoi scegliere tu o Docker) e Docker si occupa di creare la directory sul sistema host dove i

dati vengono effettivamente memorizzati. Sono la tipologia di volume più comunemente utilizzata perché sono facili da gestire e Docker si occupa dei dettagli di implementazione.

- **Bind Mounts (Montaggi Host):** Con i bind mount, specifichi un file o una directory presente sul sistema host e la monti all'interno del container. In questo modo, i file e le directory nel container sono direttamente collegati a quelli sul tuo host. Questo può essere utile per la condivisione di configurazioni o codice sorgente durante lo sviluppo, ma può presentare problemi di portabilità se il percorso specificato non esiste su un altro host.
- **tmpfs Mounts (Montaggi tmpfs):** Questi volumi risiedono nella memoria del sistema host e non vengono mai scritti sul filesystem host. Sono utili per dati temporanei e sensibili che non necessitano di persistenza, offrendo anche un buon livello di performance. Quando il container viene fermato, i dati nel tmpfs mount vengono persi.

Perché sono importanti i volumi?

- **Persistenza dei dati:** Come accennato, garantiscono che i dati importanti (come database, file di configurazione, log, ecc.) non vengano persi quando i container vengono gestiti.
- **Condivisione dei dati:** Permettono a più container di accedere e condividere gli stessi dati.
- **Backup e ripristino:** Rendono più semplice il backup e il ripristino dei dati delle tue applicazioni, dato che sono isolati dai container stessi.
- **Isolamento:** Separano i dati dalla logica applicativa contenuta nel container, migliorando l'organizzazione e la manutenibilità.

In sintesi, i volumi in Docker sono essenziali per gestire i dati in modo efficiente e persistente nelle tue applicazioni containerizzate, offrendo flessibilità e affidabilità nella gestione del ciclo di vita dei tuoi dati.