Container: cosa sono e perché stanno cambiando l'informatica

DI MASSIMO VAILATI
10/04/2026

DOCKER CONTAINER
TT INFORMATICA

Contenuto: l'articolo spiega in modo semplice cosa sono i container (in particolare Docker), come si differenziano dalla virtualizzazione tradizionale e perché rappresentano una soluzione leggera, portabile ed efficiente, ideale anche per l'uso nei laboratori scolastici.

Attività pratica: una prima guida all'installazione di Docker Desktop e poi la creazione e avvio di un server web eseguito all'interno di un container.

Autori

MASSIMO VAILATI

Negli ultimi anni la parola *container* è diventata sempre più frequente nel mondo dell'informatica. Aziende, sviluppatori e piattaforme cloud li usano ogni giorno, ma spesso non è chiaro cosa siano davvero e perché siano considerati così importanti. In questo articolo scopriremo **cosa sono i container**, **come funzionano** e **perché rappresentano una piccola rivoluzione tecnologica**.

IL PROBLEMA DA RISOLVERE

Chiunque abbia scritto o usato un software conosce questa frase:

"Sul mio computer funziona, sul tuo no."

Il motivo è semplice. I programmi dipendono da:

- sistema operativo
- librerie
- versioni del software
- configurazioni di sistema

Basta una piccola differenza e un'applicazione può smettere di funzionare. Per anni questo problema ha rallentato sviluppo, distribuzione e manutenzione del software.

Anche chi insegna informatica in laboratorio conosce bene queste difficoltà:

- PC **datati** o con risorse limitate
- Software diversi richiesti da classi o progetti differenti
- Conflitti tra versioni (Java, Python, PHP, database...)
- Ripristino delle macchine dopo errori degli studenti
- Impossibilità di installare software per mancanza di permessi

Domanda chiave: come fornire ambienti di lavoro **uniformi**, **isolati e leggeri** senza reinstallare tutto ogni volta?

PRIMA DEI CONTAINER: LE MACCHINE VIRTUALI

Per risolvere questi problemi si è iniziato a usare le **macchine virtuali**.
Una macchina virtuale simula un intero computer dentro un altro computer:

- ha un proprio sistema operativo
- è isolata dal sistema principale
- garantisce che il software funzioni sempre allo stesso modo

Le macchine virtuali offrono un **isolamento completo** tra il sistema host e l'ambiente virtualizzato. Questo significa che ogni macchina virtuale funziona come un vero computer indipendente, con il proprio sistema operativo, le proprie impostazioni e le proprie applicazioni.

Grazie a questo livello di separazione, sono particolarmente **adatte a simulare reti**, **server e sistemi reali**, permettendo di ricreare scenari complessi molto vicini a quelli presenti in ambito professionale.

Il rovescio della medaglia? Le macchine virtuali sono **pesanti**, consumano molta memoria e si avviano lentamente. Ogni macchina virtuale richiede infatti una **notevole quantità di risorse**, come memoria RAM e potenza di calcolo, risorse che spesso i computer delle scuole non possono offrire in abbondanza.
L'**avvio è piuttosto lento**, perché deve essere caricato un intero sistema operativo, e la **gestione risulta complessa**, sia per l'installazione iniziale sia per la manutenzione nel tempo. Tutti questi aspetti rendono le macchine virtuali **poco adatte ai PC di laboratorio**, soprattutto se datati o utilizzati da più classi con esigenze diverse.

COSA SONO I CONTAINER

I **container** sono una soluzione più moderna e intelligente.

Un container è un ambiente isolato che contiene:

- un'applicazione
- tutto ciò che le serve per funzionare

Ma **non contiene un sistema operativo completo**.

I container **condividono il sistema operativo del computer su cui girano**, rendendoli molto più leggeri e veloci rispetto alle macchine virtuali.

La tecnologia dei container esiste da tempo, ma è diventata popolare grazie a **Docker**, uno strumento che ne ha semplificato l'uso.

Docker permette di:

- creare container facilmente
- distribuirli su qualsiasi computer
- avviarli in pochi secondi

Per questo oggi è uno standard nel mondo dello sviluppo software e del cloud.

I container hanno assunto un ruolo centrale nell'informatica moderna perché rispondono in modo efficace a esigenze concrete di velocità, efficienza e affidabilità. Uno dei loro punti di forza principali è la **rapidità di avvio**: un container può essere avviato in pochi secondi, e in molti casi quasi istantaneamente, consentendo di lavorare senza lunghe attese e rendendo più agili le fasi di sviluppo e test.

Un altro aspetto fondamentale è la **leggerezza**. I container utilizzano solo le risorse strettamente necessarie al funzionamento dell'applicazione, senza dover caricare un intero sistema operativo. Questo permette di eseguire **numerosi container sullo stesso computer**, anche su macchine non particolarmente potenti, ottimizzando l'uso di memoria e CPU.

La **portabilità** è forse uno dei vantaggi più apprezzati. Un container funziona esattamente allo stesso modo sul computer di uno sviluppatore, su un server aziendale o in un ambiente cloud. Questa caratteristica elimina molte delle incompatibilità tradizionali tra ambienti diversi e garantisce che un'applicazione si comporti sempre in modo prevedibile.

Infine, i container offrono un buon livello di **isolamento** tra le applicazioni. Ogni container opera in modo indipendente dagli altri, riducendo il rischio che un errore o un malfunzionamento possa influenzare il resto del sistema. Questo isolamento contribuisce anche a migliorare la sicurezza complessiva e la stabilità degli ambienti di lavoro.

Perché Docker è ideale per la scuola

Docker si rivela uno strumento particolarmente adatto all'ambiente scolastico perché risponde in modo concreto a molte delle difficoltà tipiche dei laboratori di informatica. Uno dei principali vantaggi è la possibilità di creare **ambienti di lavoro identici per tutti gli studenti**. Ogni studente utilizza lo stesso software, con le stesse versioni e configurazioni, eliminando il classico problema del "a me non funziona sul mio PC" e permettendo al docente di concentrarsi sulla didattica anziché sulla risoluzione di problemi tecnici.

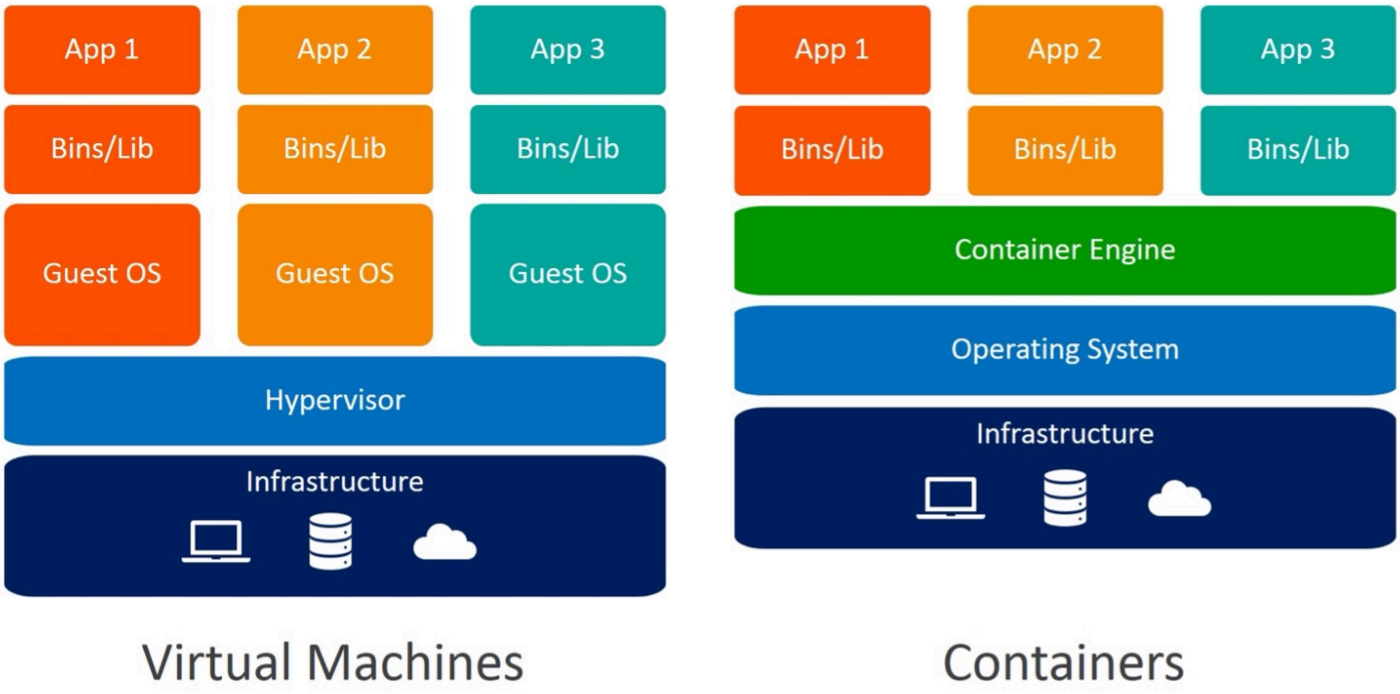
Un altro punto di forza è la **semplicità di utilizzo**. Con Docker non è necessario affrontare lunghe e complesse installazioni: spesso è sufficiente un solo comando per avviare un intero ambiente di lavoro che può includere un web server, un database o un'applicazione completa. Questo consente di risparmiare tempo prezioso durante le lezioni e di avviare rapidamente le attività di laboratorio.

Docker garantisce inoltre un **isolamento sicuro**. Gli studenti possono sperimentare liberamente, fare errori e modificare configurazioni senza il rischio di compromettere il sistema operativo del computer. Se qualcosa smette di funzionare, è sufficiente eliminare il container e ricrearlo, ripartendo da una situazione pulita in pochi secondi.

Infine, Docker permette un notevole **risparmio di risorse**, caratteristica fondamentale nei contesti scolastici. I container sono leggeri e funzionano bene anche su computer con 4–8 GB di RAM, rendendo Docker una soluzione ideale per **laboratori non recenti** o con hardware limitato, senza rinunciare a strumenti moderni e professionali.

VIRTUALIZZAZIONE VS CONTAINER: CONFRONTO DIRETTO

La figura seguente rappresenta un **confronto visivo tra l'architettura delle macchine virtuali (Virtual Machines)** e quella dei **container**, mostrando chiaramente **come sono organizzati i diversi livelli software** nei due approcci.



La tabella mette a confronto, in modo sintetico, le **macchine virtuali** e i **container (come Docker)** evidenziando le principali differenze dal punto di vista tecnico e pratico, con particolare attenzione all'uso nei laboratori scolastici.

Caratteristica	Macchine Virtuali	Container (Docker)
Sistema operativo	Completo	Condiviso
Avvio	Minuti	Secondi
Consumo risorse	Alto	Basso
Isolamento	Totale	Logico
Facilità di gestione	Media	Alta
Adatto a PC scolastici	❌	✅

ARCHITETTURA DI DOCKER

Docker è organizzato secondo un'architettura semplice ma molto efficace. Alla base c'è il **sistema operativo host**, sopra il quale gira il **Docker Engine**, il componente che gestisce tutto il funzionamento di Docker.

Le **immagini Docker** sono modelli **statici e immutabili** che contengono l'applicazione e tutte le sue dipendenze. Possono essere viste come dei "progetti pronti all'uso" e vengono salvate localmente o scaricate da un registro come Docker Hub. In pratica è un **pacchetto statico** che contiene tutto ciò che serve per eseguire un'applicazione. Al suo interno troviamo un sistema di base, spesso una versione minimale di Linux, l'applicazione vera e propria, le librerie necessarie al suo funzionamento e le configurazioni richieste.

L'immagine non è un programma in esecuzione, ma una sorta di **modello pronto all'uso**. Una volta creata, **non cambia**: è immutabile e può essere riutilizzata tutte le volte che serve per creare nuovi ambienti identici. Proprio per questo viene spesso paragonata a uno "stampino", da cui è possibile ottenere più copie uguali.

Un esempio di immagine è `python:3.11`. Finché resta un'immagine, è semplicemente un file memorizzato sul sistema: **non consuma risorse e non sta facendo nulla**.

Un **container Docker** è invece l'**istanza attiva** di un'immagine. Quando un'immagine viene avviata, Docker crea un container che entra in esecuzione e diventa un ambiente isolato in cui l'applicazione può funzionare.

A differenza dell'immagine, il container **utilizza risorse reali** del computer, come CPU e memoria RAM. Può essere avviato, fermato, riavviato o eliminato in qualsiasi momento e opera in modo isolato rispetto al sistema operativo host e agli altri container.

Ad esempio, eseguendo il comando:

```
docker run python:3.11
```

Docker:

prende l'immagine python:3.11,
crea un nuovo container basato su di essa,
e lo avvia immediatamente.

In questo modo l'immagine diventa un ambiente di lavoro concreto e funzionante, pronto per eseguire applicazioni o comandi.

ATTIVITÀ PRATICA N. 1

Installazione di Docker Desktop

Docker Desktop (**https://www.docker.com/products/docker-desktop**) è un'applicazione GUI (interfaccia grafica) per Windows, Mac e Linux che semplifica la creazione, la gestione e l'esecuzione di container Docker. Questo ambiente ha diversi vantaggi:

Gestione Semplificata: Offre un'interfaccia intuitiva per avviare, fermare e gestire i container, sostituendo in parte l'uso della riga di comando.
Ambiente Unificato: Include tutto il necessario per sviluppare app containerizzate su macchine locali.
Integrazione WSL2: Su Windows, utilizza il Windows Subsystem for Linux 2 (WSL2) per prestazioni ottimali. Docker è progettato per Linux e WSL 2 fornisce un vero kernel Linux in Windows, consentendo ai container Linux di funzionare senza emulazione.
Facilità d'uso: Permette di passare facilmente da un ambiente di sviluppo all'altro e di scaricare/caricare immagini dal Docker Hub (**https://hub.docker.com**), il repository ufficiale basato su cloud di Docker, utilizzato per trovare, archiviare e condividere immagini di container.

In ambiente Windows è necessario quindi installare WSL2:

1. **Aprire PowerShell come Amministratore**
2. **Eseguire il comando di installazione:** Digitare `wsl --install` e premere Invio.

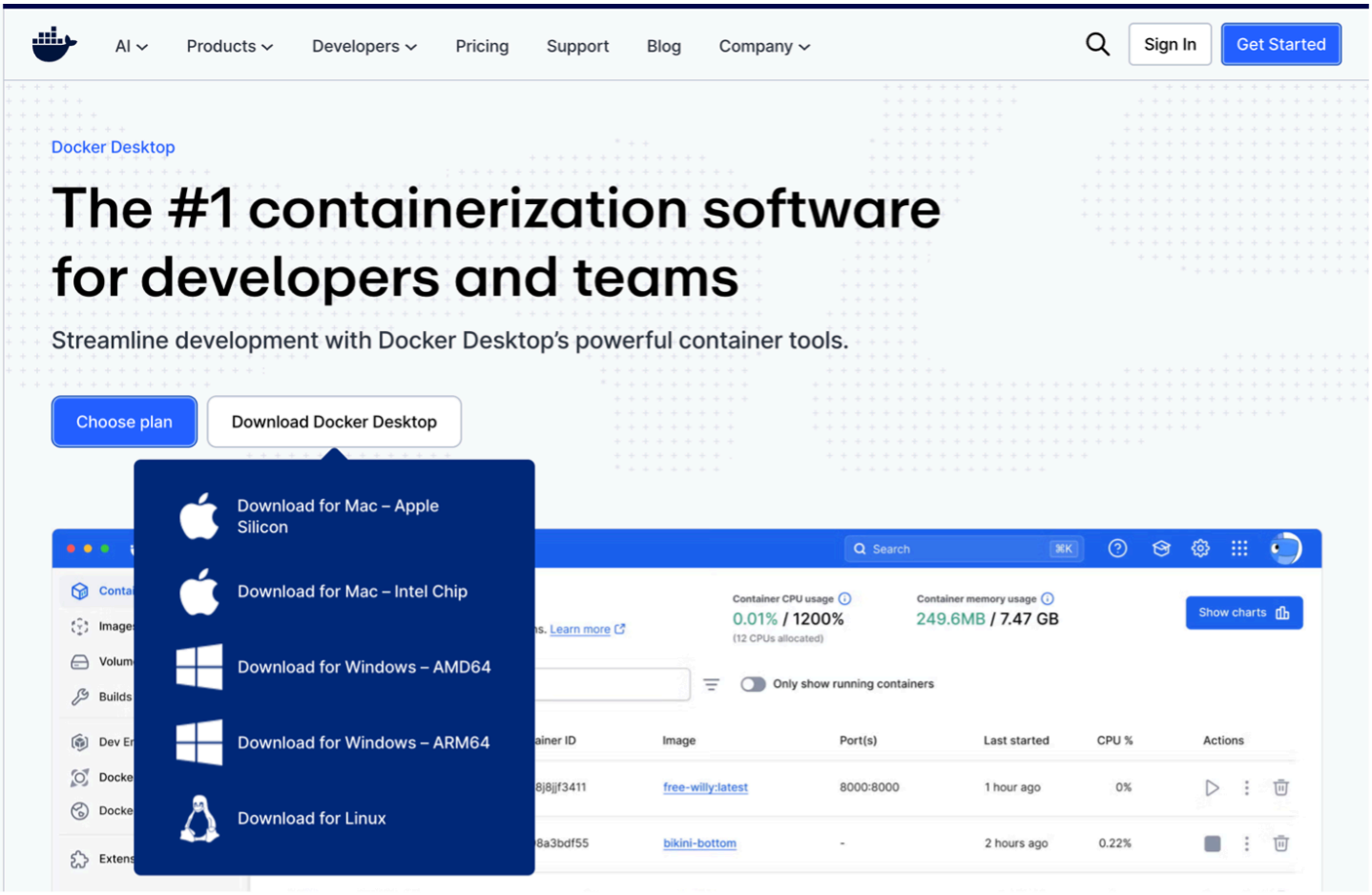
Nota: Questo comando abilita le funzionalità necessarie ("Virtual Machine Platform" e "Windows Subsystem for Linux") e scarica Ubuntu come distribuzione predefinita.

3. **Riavviare il computer**: Al termine del processo, è necessario riavviare il PC per applicare le modifiche.
4. **Configurazione iniziale**: Dopo il riavvio, si aprirà automaticamente una finestra terminale (o aprendo "Ubuntu" dal menu Start) per completare l'installazione, dove sarà richiesto di impostare un nome utente e una password Linux.

Per ulteriori indicazioni vedere **https://docs.docker.com/desktop/setup/install/windows-install/**

In ambiente Linux non è necessario alcun prerequisito, in ambiente macOS Docker crea una macchina virtuale. In tutti gli ambienti deve essere quindi abilitata la virtualizzazione.

A questo punto si può installare Docker Desktop scegliendo opportunamente la propria piattaforma.



ATTIVITÀ PRATICA N. 2

Creazione di un container server web

Realizziamo un server **Web funzionante in locale** utilizzando Docker e il web server **Nginx**, comprendendo i concetti di:

- immagine Docker
- container
- mappatura delle porte
- bind mount (condivisione cartelle host–container)

Al termine dell'attività il server sarà accessibile tramite browser e potrà pubblicare pagine Web reali.

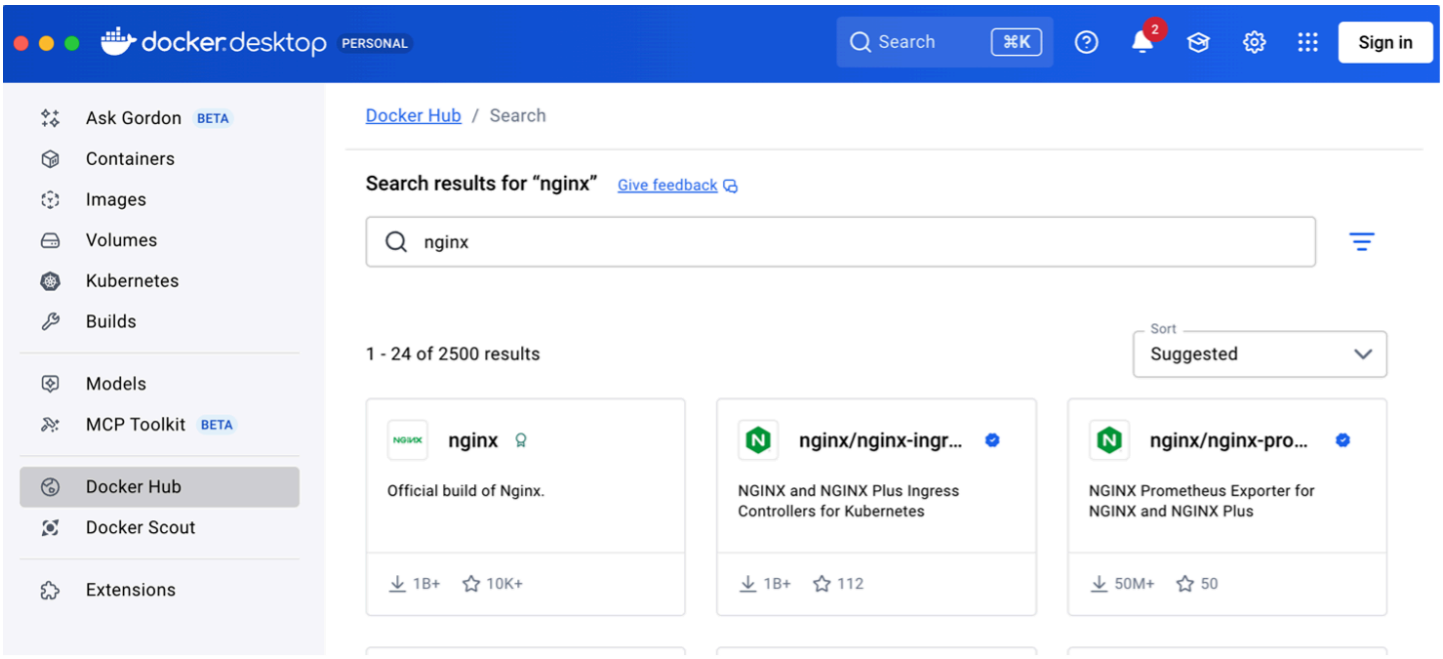
1. Download dell'immagine Nginx

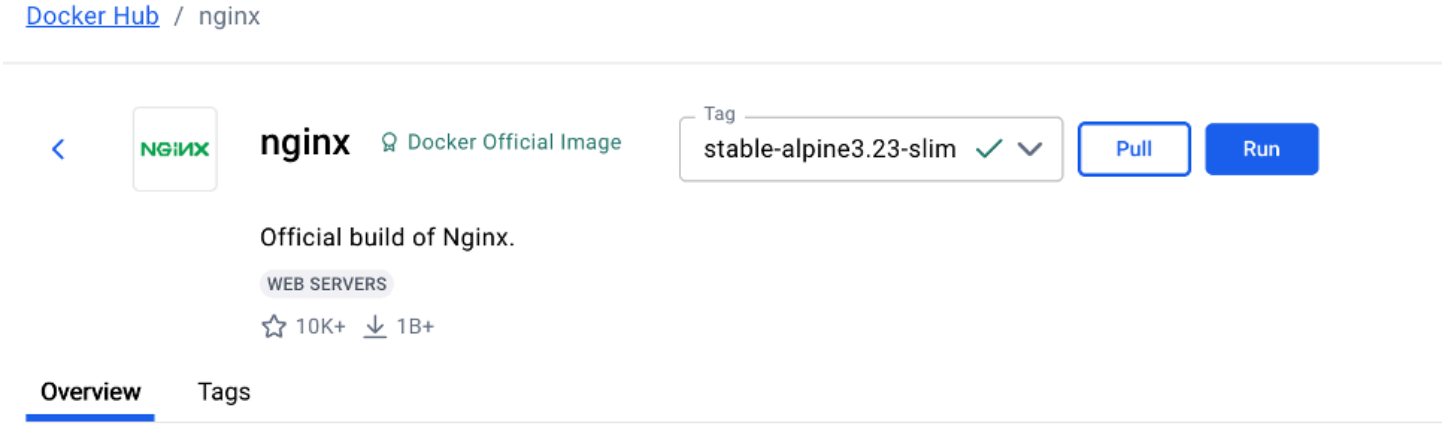
Nginx è un server **HTTP leggero, performante e ampiamente utilizzato** in ambito professionale, soprattutto in architetture a microservizi.

1. Aprire **Docker Desktop**.
2. Accedere alla sezione **Docker Hub**.
3. Cercare **nginx**.
4. Selezionare l'immagine ufficiale (generalmente la prima tra i risultati).

   È basata su **Linux Alpine**, una distribuzione molto leggera.

5. Scegliere una versione (Tag).
6. Fare clic su **Pull** per scaricare l'immagine nel proprio sistema.





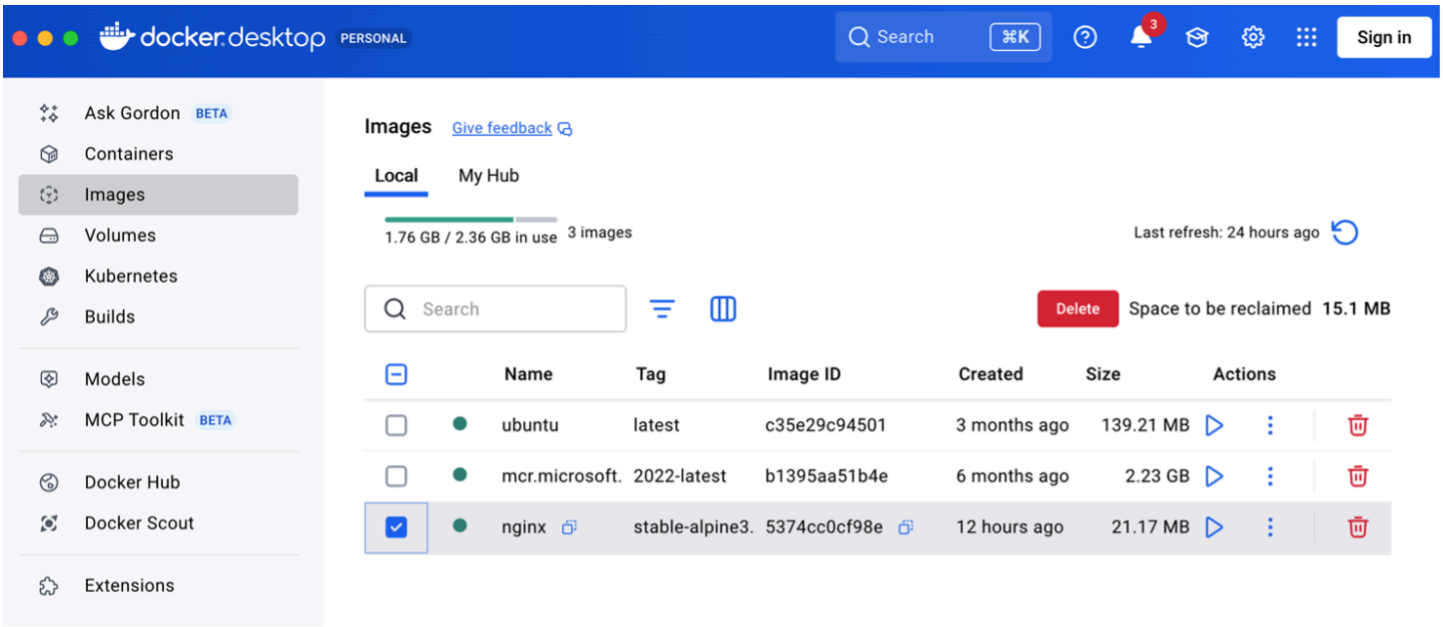
Nella sezione **Overview** sono riportate le indicazioni per utilizzare l'immagine tramite comandi Docker da riga di comando (modalità spesso necessaria in contesti professionali). In questa attività, tuttavia, seguiremo una procedura diversa, utilizzando l'interfaccia grafica di Docker Desktop, così da semplificare le operazioni.

2. Verifica delle immagini scaricate

Accedere alla scheda **Images** di Docker Desktop.

L'immagine **nginx** deve comparire nell'elenco delle immagini disponibili localmente.

Nota: Da una singola immagine è possibile creare più container indipendenti.



3. Creazione del container

Per utilizzare l'immagine è necessario creare un container, ovvero un'istanza eseguibile dell'immagine.
1. Nella sezione **Images**, individuare nginx.
2. Fare clic su **Run** (icona di avvio nella colonna **Actions**).
3. Prima di avviare il container, aprire **Optional Settings** per configurare i parametri.
Nota: se clicchiamo su Run verrebbe creato un container con parametri predefiniti e con un nome casuale.

4. Configurazioni fondamentali

a) Nome del container
Assegnare un nome identificativo, ad esempio: *mioweb*. Questo permette di riconoscerlo facilmente tra i container attivi.

b) Bind Mount (condivisione cartelle)
Creare sul computer una cartella, ad esempio: *miosito*. Questa conterrà i file del sito (HTML, CSS, JS) e potrà essere modificata con un editor come **Visual Studio Code**.

Occorre mappare questa cartella con la directory predefinita di Nginx nel container: `/usr/share/nginx/html`

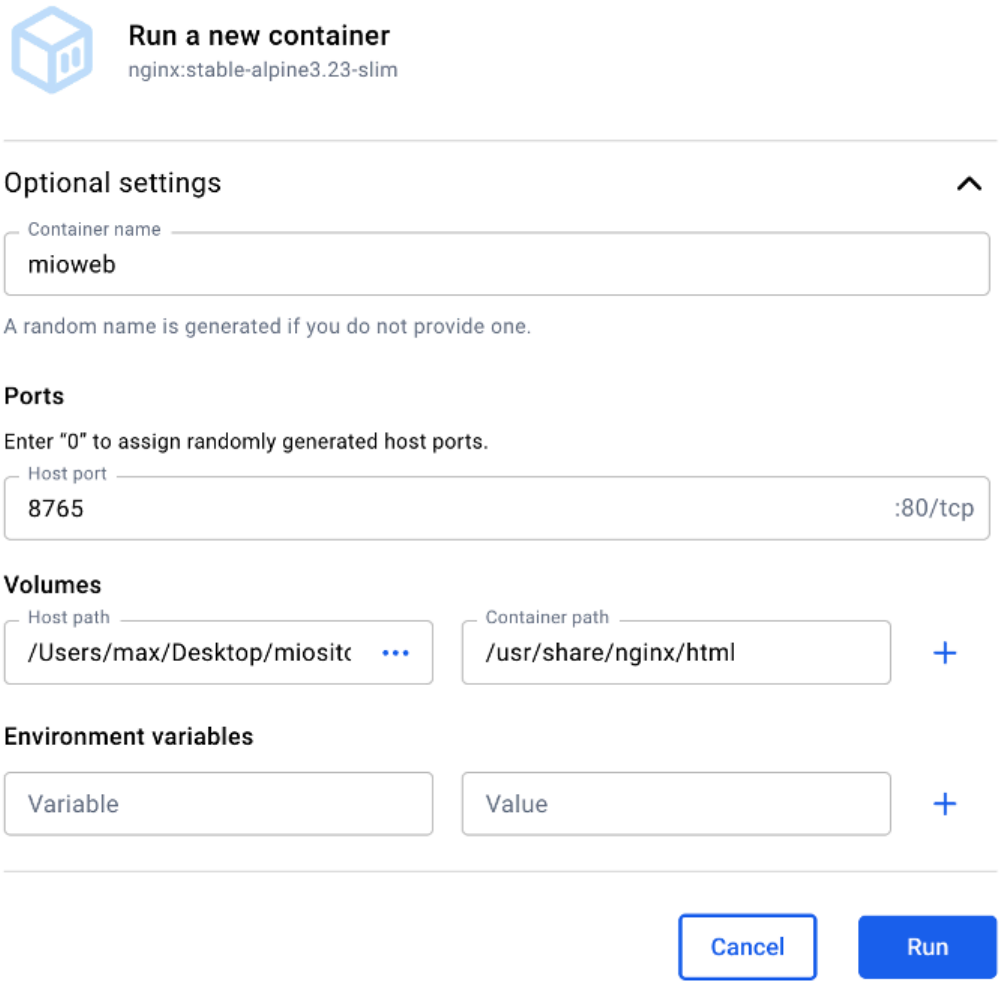
Docker utilizzerà i file presenti sul computer al posto di quelli interni al container.

c) Mappatura delle porte

Configurare il port mapping (ad esempio la porta 8765):

Porta computer: 8765 → Porta container: 80

In questo modo, quando un browser accederà a: `http://localhost:8765` la richiesta verrà inoltrata al server Nginx in esecuzione nel container.

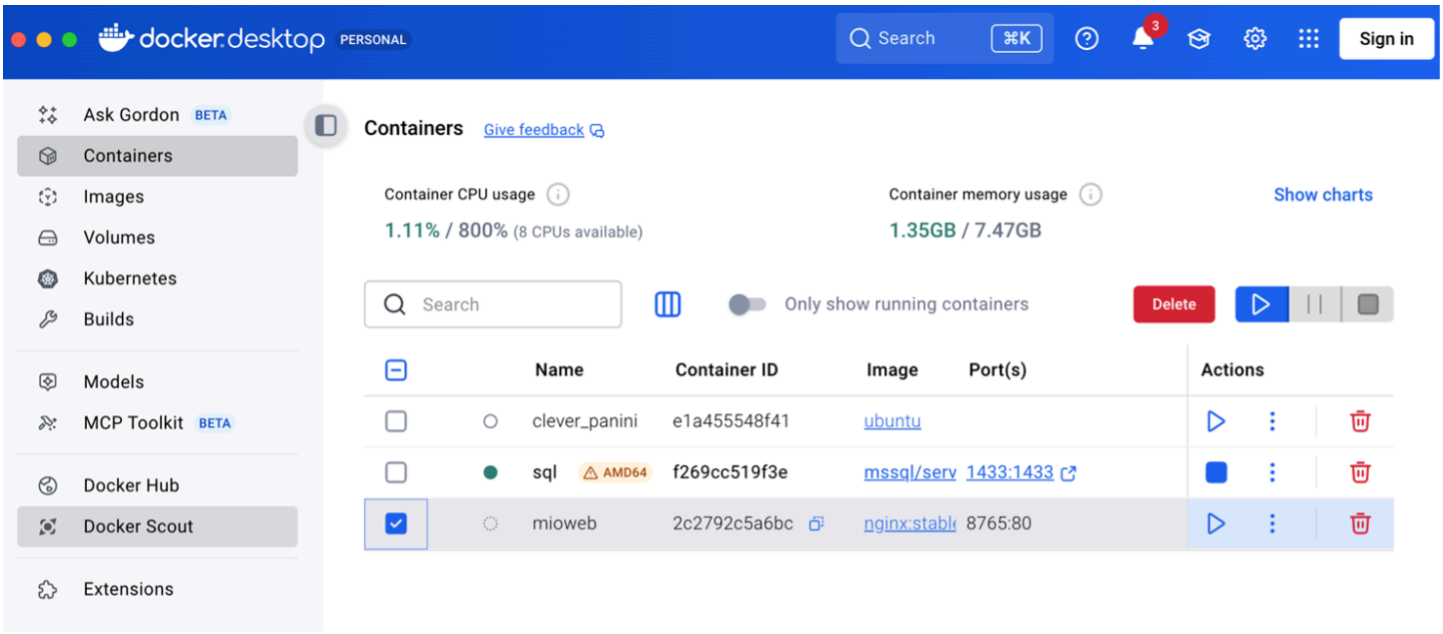


5. Avvio del container

Fare clic su **Run** per creare ed eseguire il container.

Accedere poi alla scheda **Containers** per visualizzarlo.

- Il triangolo blu indica **avvio**
- Il quadrato blu indica **container in esecuzione**
- I pulsanti **Pause**, **Stop** e **Delete** permettono di gestire il ciclo di vita del container

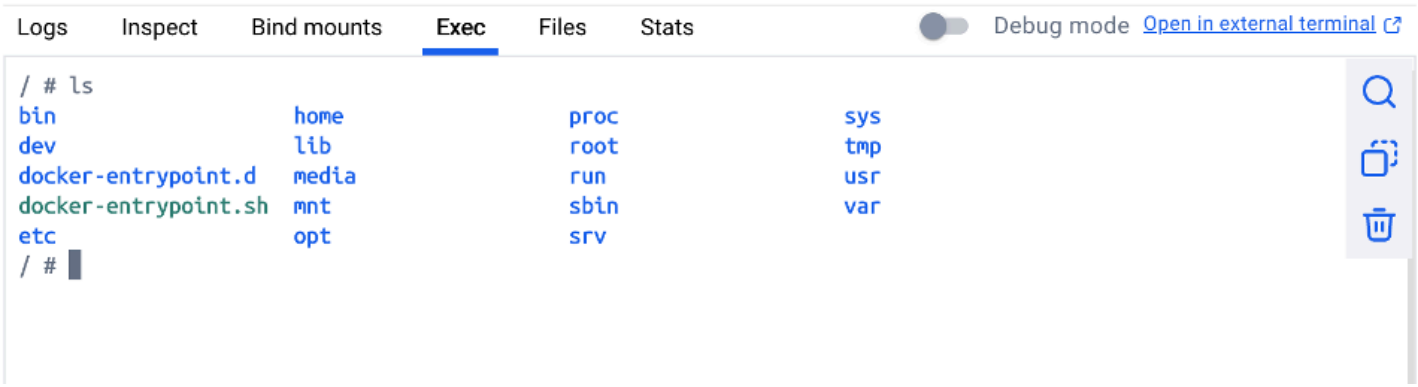


6. Strumenti di gestione del container

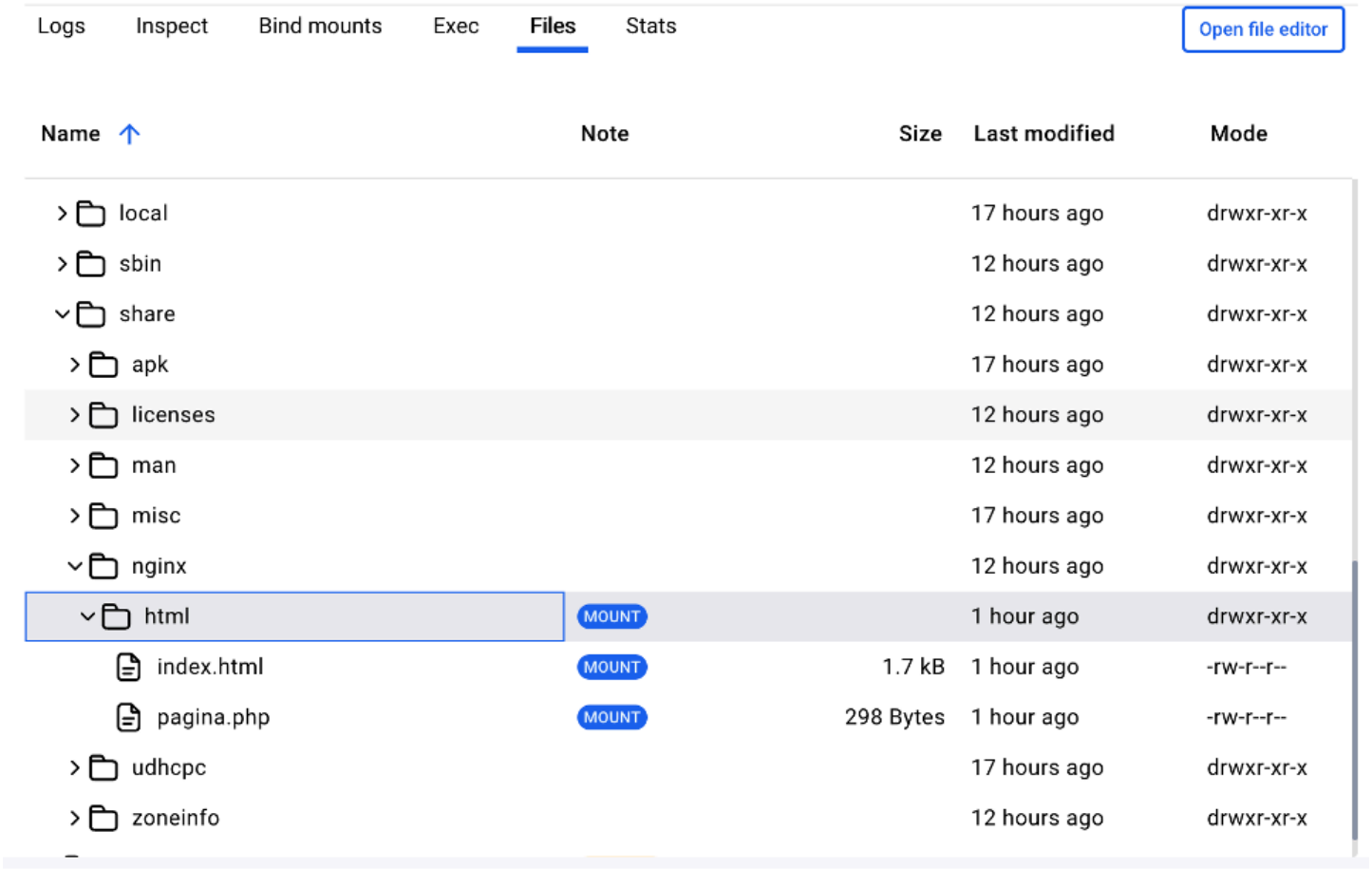
Cliccando sul nome del container (*mioweb*) si accede al pannello di gestione. Possiamo esplorare le varie sezioni. Noi ci soffermiamo sulle seguenti, utili per l'attività.

Bind Mounts: mostra le cartelle dell'host montate nel file system Linux del container.

Exec: apre un terminale del sistema operativo del container (bash), dal quale è possibile eseguire comandi Linux.

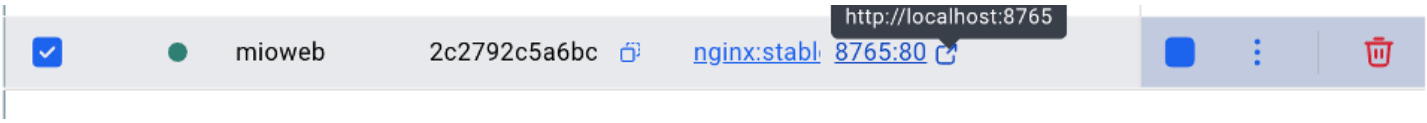


Files: permette di esplorare il file system del container. In figura la cartella `/usr/share/nginx/html` mostra i file presenti nella cartella *miosito* del computer.



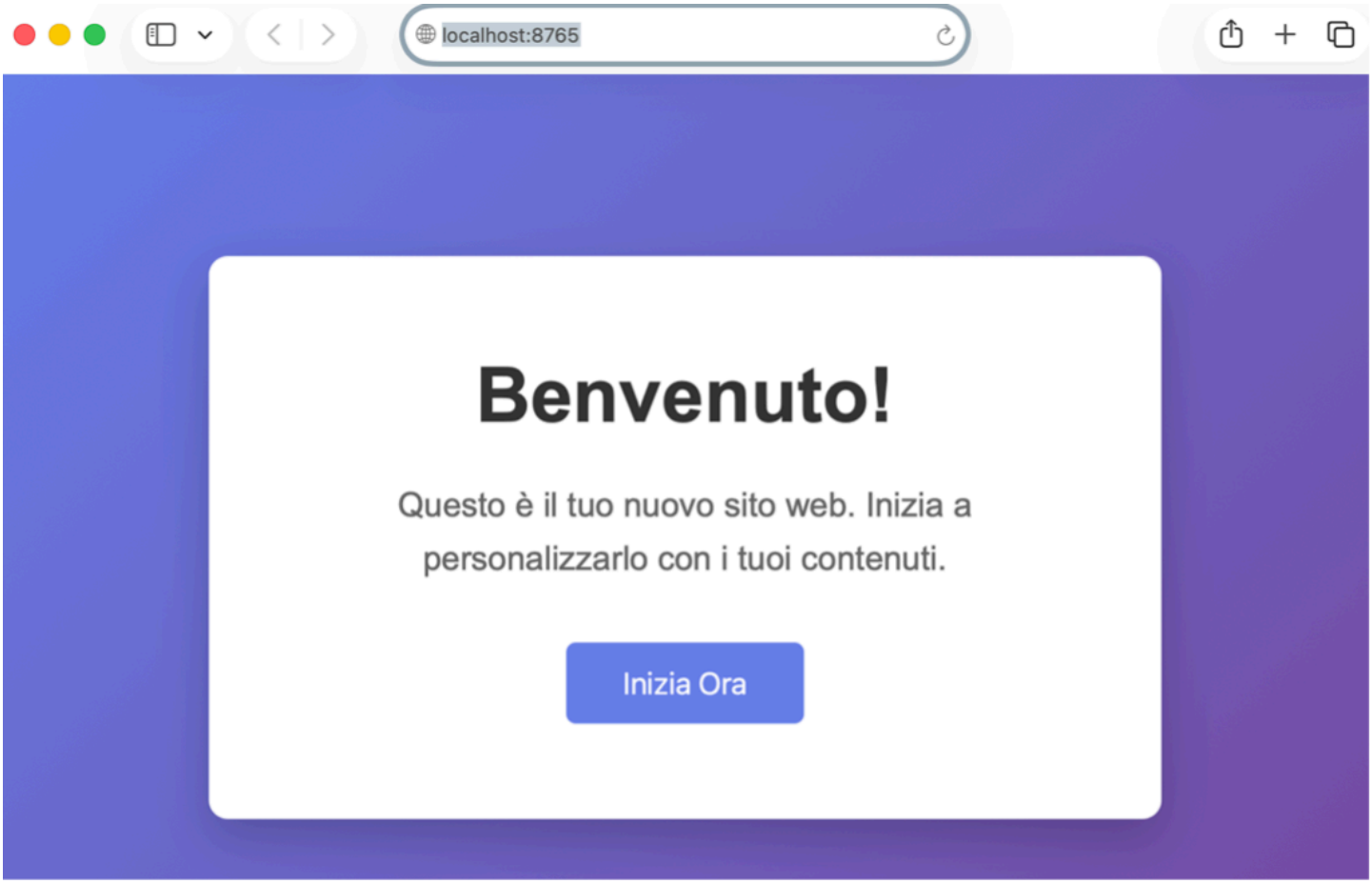
7. Verifica del funzionamento

Per testare il server Web:
1. Aprire il link relativo alle porte esposte nella scheda del container



oppure

2. Digitare nel browser:
`http://localhost:8765`
Verrà caricata la pagina **index.html** presente nella cartella *miosito*.



8. Accesso dalla rete locale

Poiché il computer espone la porta **8765**, il sito può essere visualizzato anche da altri dispositivi della rete locale utilizzando: http://INDIRIZZO_IP_DEL_COMPUTER:8765

Conclusione

Con questa attività è stato realizzato un **server Web containerizzato**, utile per:

- sviluppo e test di siti Web
- esercitazioni di laboratorio
- pubblicazione di servizi nella rete locale

L'utilizzo dei container consente di avere ambienti **isolati, riproducibili e facilmente distribuibili**, competenze fondamentali nell'informatica moderna.