

Introduzione ai Containers

Su Docker

Argomenti

- Confronto tra macchine virtuali e container
- Come è fatto Docker
- Creazione di container da terminale
- Configurazione container (Docker Compose)
- Creazione di immagini (Dockerfile)
- Integrazione di Docker con Visual Studio

Machine virtuali e containers

Macchine virtuali

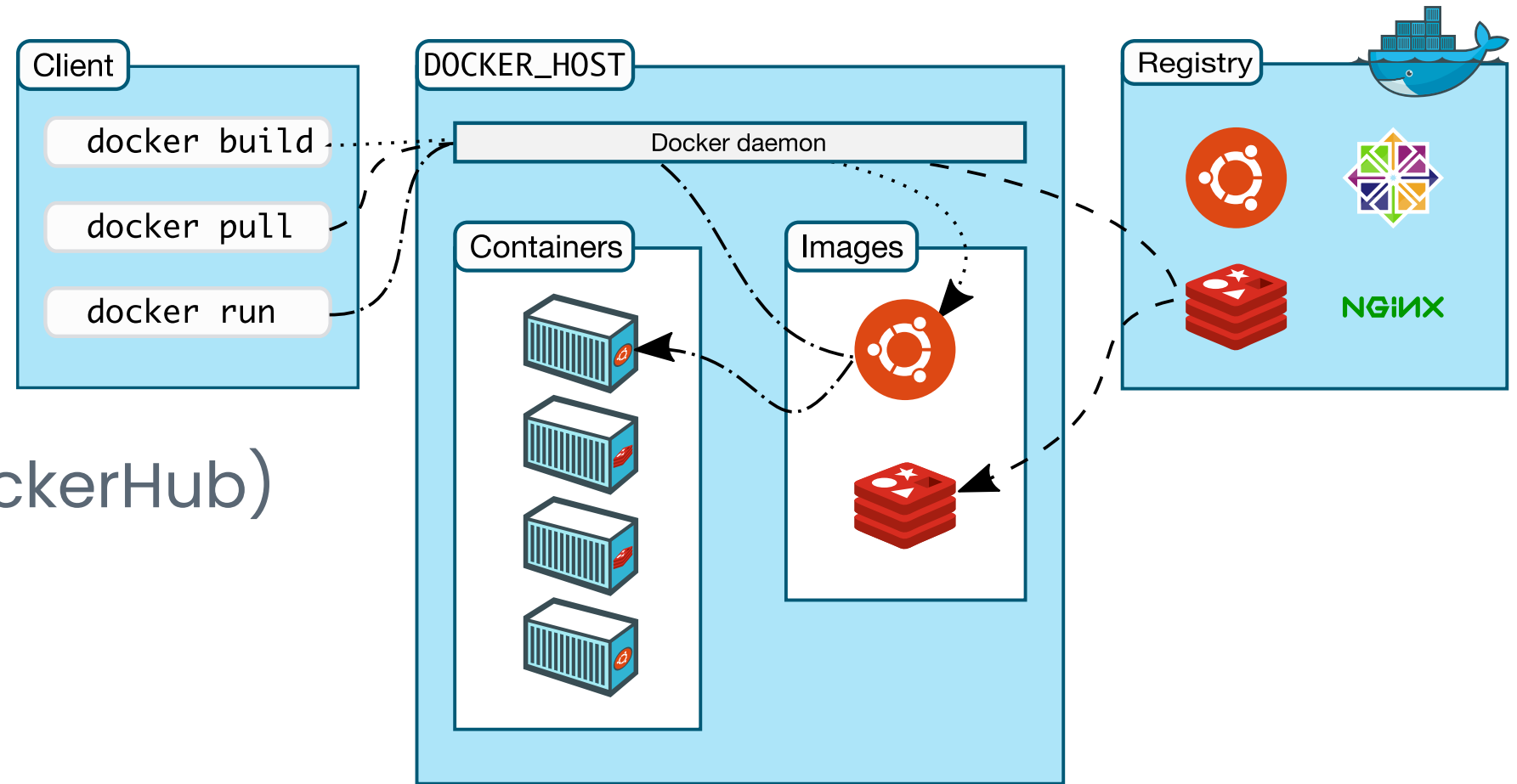
- Ambienti isolati
- Hypervisor (hardware)
- Bootloader (BIOS/UEFI)
- Kernel separato dall'host
- Dischi virtualizzati
Schede di rete virtualizzate
ecc...
- Avvio lento (boot sistema operativo)
- Overhead a runtime
- Guest ed host possono essere sistemi operativi diversi con kernel diversi

Containers

- Ambienti isolati
- Kernel (software)
- Nessun bootloader
- Kernel condiviso con l'host
- Filesystem isolato
Rete isolata
ecc...
- Avvio immediato (è solo un eseguibile)
- Overhead insignificante
- Guest ed host coincidono, sono lo stesso sistema operativo con lo stesso kernel



- Daemon
- Client
- Registro (DockerHub)
- Immagini
- Containers
- Volumi



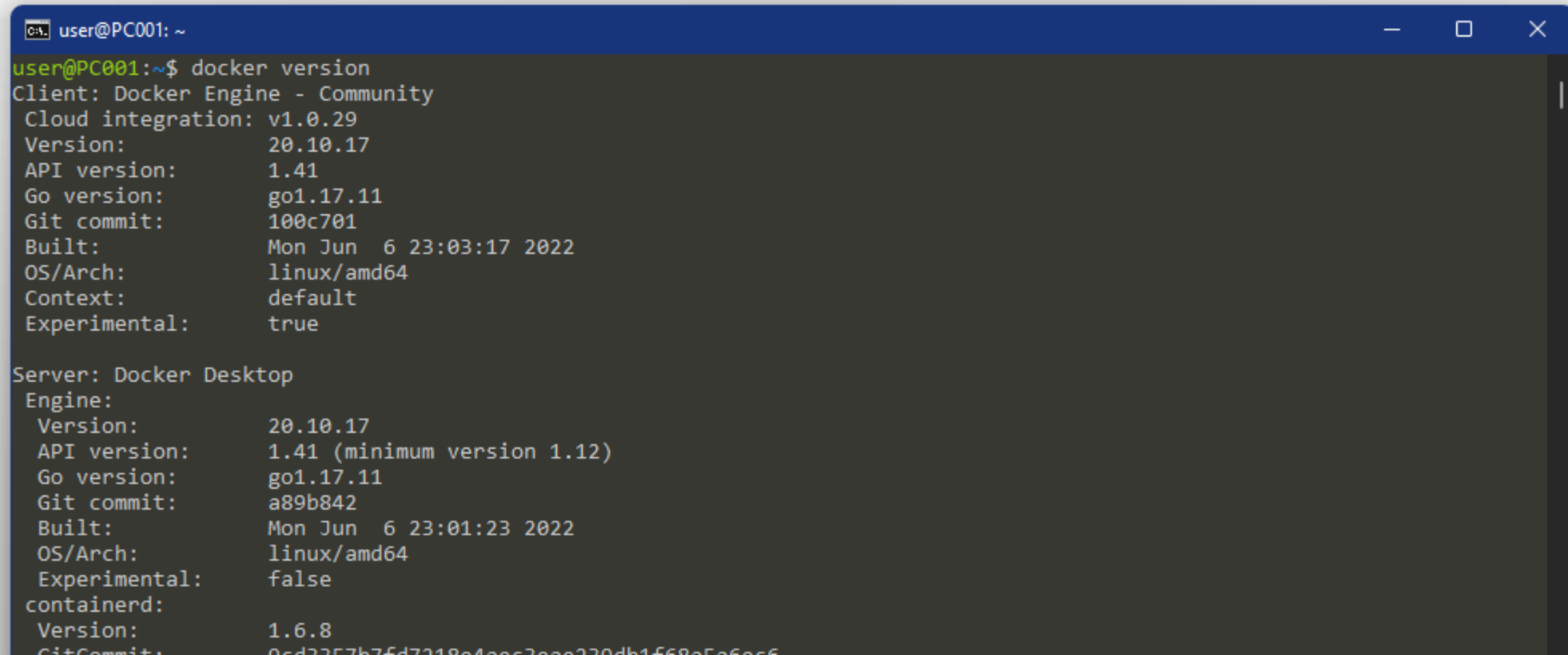
Daemon (dockerd)

- È un normale eseguibile
- Tiene traccia dei container in esecuzione
- Accessibile tramite socket TCP

```
Cmder - wsl -d docker-desktop
PC001:~# ps aux | grep dockerd | less -FS~
1280 root      0:00 /usr/bin/logwrite -n dockerd /usr/local/bin/dockerd --containerd /var/run/desktop-containerd/contai
1285 root      1:02 /usr/local/bin/dockerd --containerd /var/run/desktop-containerd/containerd.sock --pidfile /run/desk
2020 root      0:00 /usr/bin/logwrite -n cri-dockerd /usr/bin/cri-dockerd --docker-endpoint unix:///run/guest-services/
2025 root      1:02 /usr/bin/cri-dockerd --docker-endpoint unix:///run/guest-services/docker.sock --pod-infra-container
2096 root      0:00 /usr/bin/logwrite -n kubelet kubelet --kubeconfig=/etc/kubernetes/kubelet.conf --config /etc/kubead
2102 root      1:19 kubelet --kubeconfig=/etc/kubernetes/kubelet.conf --config /etc/kubeadm/kubelet.yaml --bootstrap-ku
16010 root      0:00 grep dockerd
```

Client (docker)

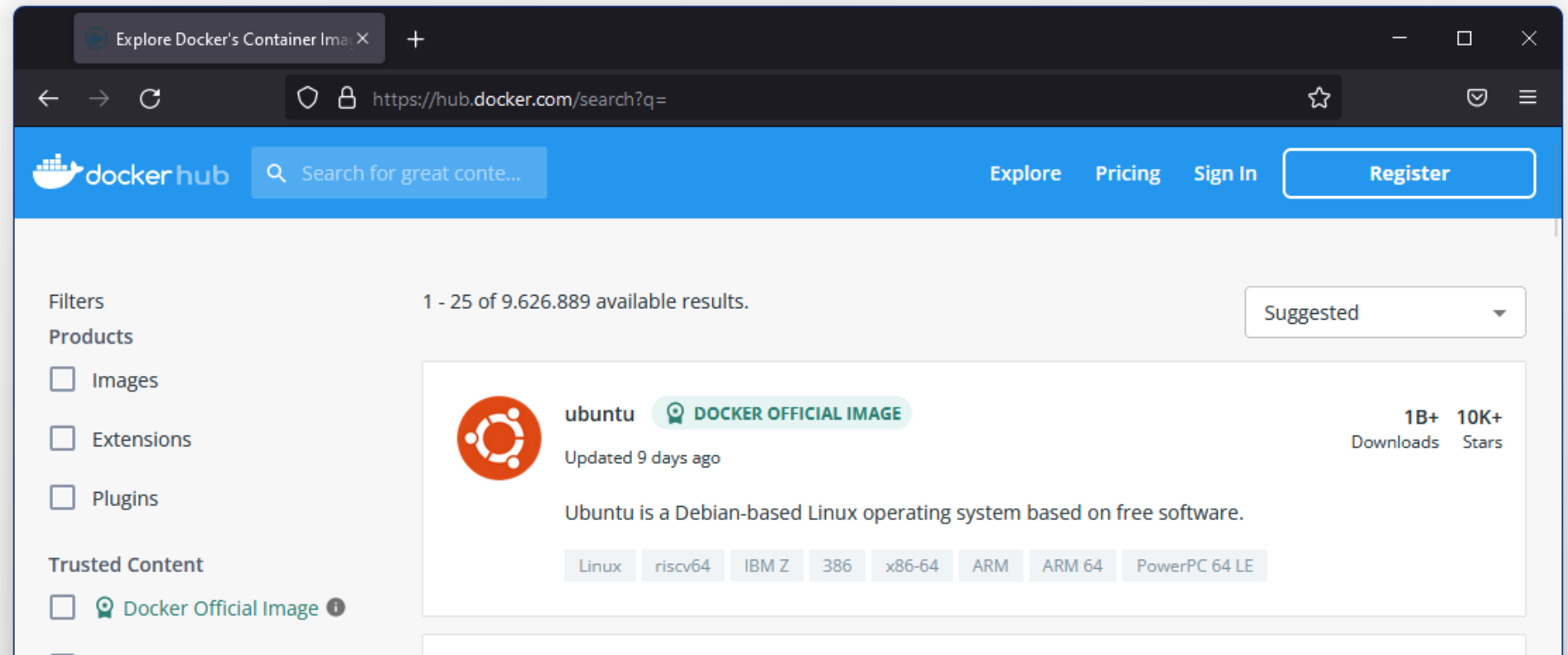
- Interfaccia da linea di comando
- Permette di comunicare con il daemon

A terminal window with a dark blue title bar and a dark gray background. The title bar shows 'user@PC001: ~' and standard window controls. The terminal displays the output of the 'docker version' command. The output is divided into two sections: 'Client: Docker Engine - Community' and 'Server: Docker Desktop'. The client section lists various version details, including 'Cloud integration: v1.0.29', 'Version: 20.10.17', 'API version: 1.41', 'Go version: go1.17.11', 'Git commit: 100c701', 'Built: Mon Jun 6 23:03:17 2022', 'OS/Arch: linux/amd64', 'Context: default', and 'Experimental: true'. The server section lists details for 'Engine' and 'containerd'. The engine details include 'Version: 20.10.17', 'API version: 1.41 (minimum version 1.12)', 'Go version: go1.17.11', 'Git commit: a89b842', 'Built: Mon Jun 6 23:01:23 2022', 'OS/Arch: linux/amd64', and 'Experimental: false'. The containerd details include 'Version: 1.6.8' and a partially visible 'GitCommit' line.

```
user@PC001: ~  
user@PC001:~$ docker version  
Client: Docker Engine - Community  
Cloud integration: v1.0.29  
Version: 20.10.17  
API version: 1.41  
Go version: go1.17.11  
Git commit: 100c701  
Built: Mon Jun 6 23:03:17 2022  
OS/Arch: linux/amd64  
Context: default  
Experimental: true  
  
Server: Docker Desktop  
Engine:  
Version: 20.10.17  
API version: 1.41 (minimum version 1.12)  
Go version: go1.17.11  
Git commit: a89b842  
Built: Mon Jun 6 23:01:23 2022  
OS/Arch: linux/amd64  
Experimental: false  
containerd:  
Version: 1.6.8  
GitCommit: 0cd2257b7f5d7218e4a0c2e0e230db1f668e5a6066
```

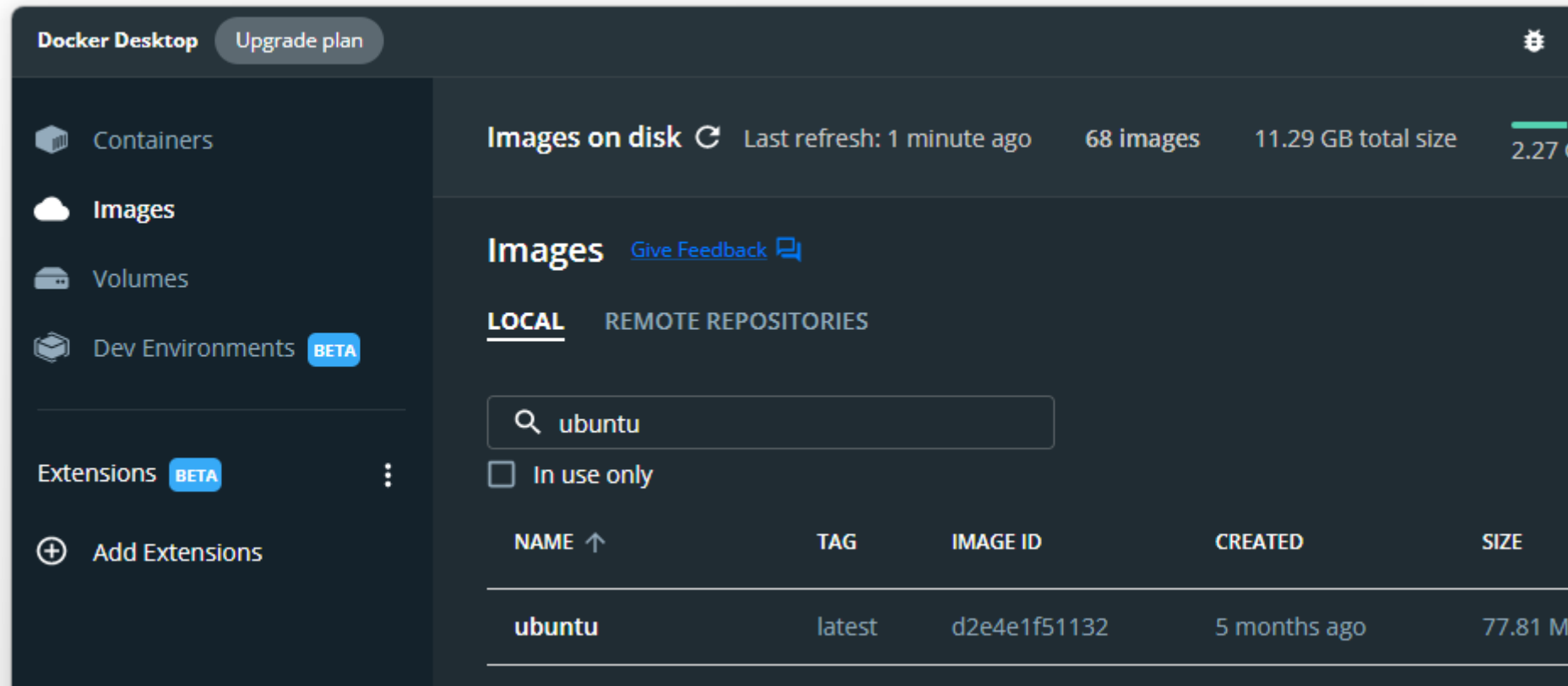
Registro (DockerHub)

- Interfaccia web
- Contiene le immagini pubbliche



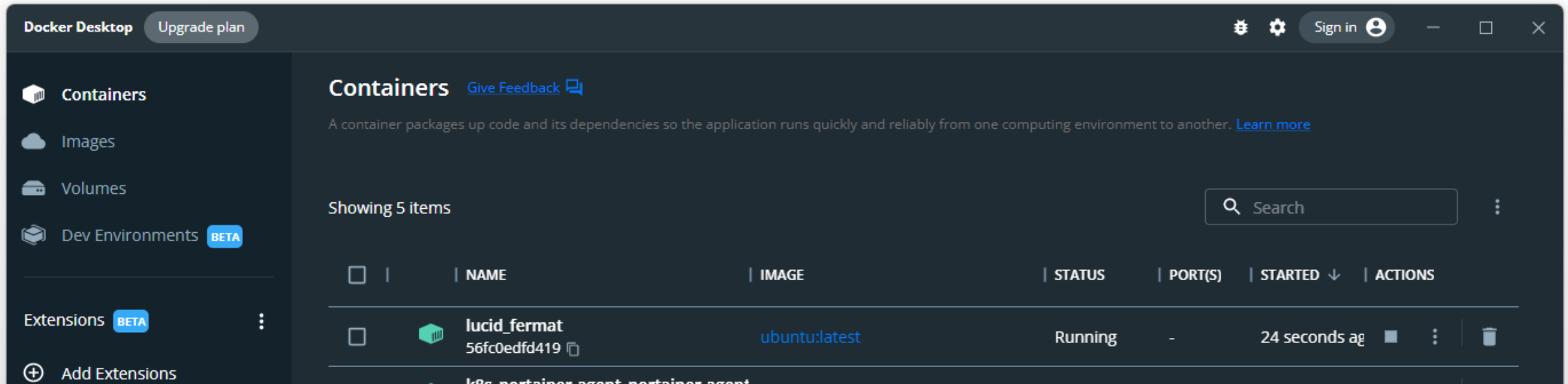
Immagini

- Possono essere scaricate dal repository
- Contengono una copia del filesystem virtuale
- Immutabili
- Più layers



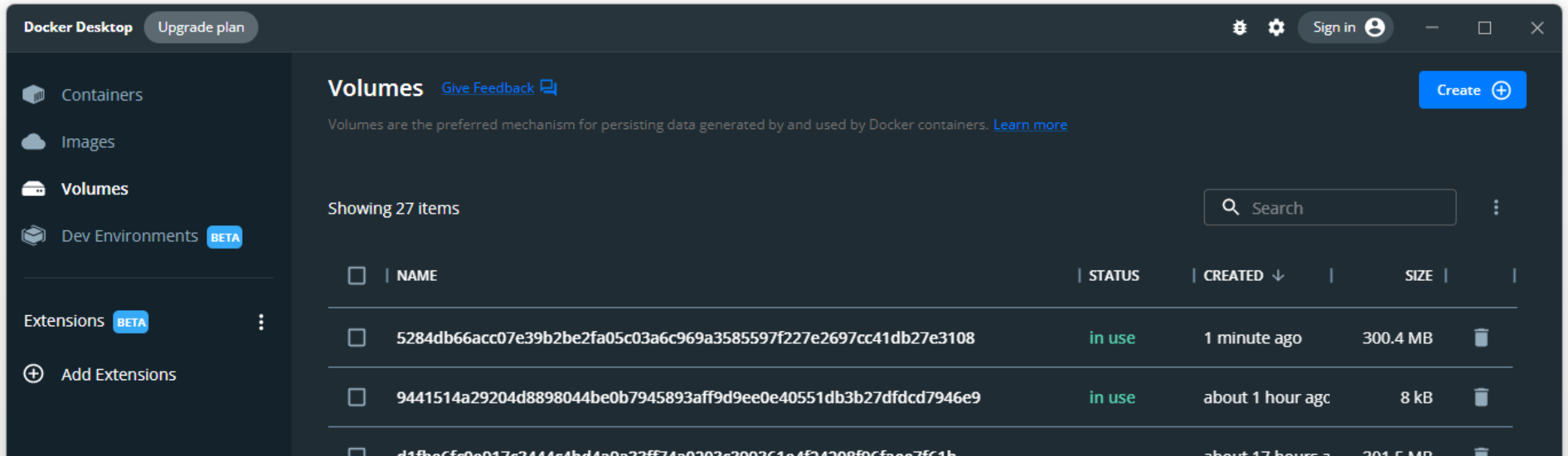
Containers

- Sono istanze delle immagini
- Contengono una copia del filesystem virtuale
- Mutabili
- Contengono uno o più processi



Volumi

- Non vengono eliminati insieme ai container
- “Agganciabili” ai container
- Possono essere normali cartelle sull’host



Docker su Linux

- Docker su Linux non è nulla di speciale, è solo una serie di strumenti che usano le funzionalità del Kernel per simulare ambienti separati
- I processi in esecuzione all'interno di un container non vedono nulla di ciò che si trova all'esterno
- I layers delle immagini sono normali cartelle
- I containers sono normali cartelle
- I volumi sono normali cartelle

Docker su Windows

- Docker nativo per Windows
- Docker su Windows tramite WSL 2

Docker nativo per Windows

- Usa API del Kernel scritte ad-hoc per Docker
- Una versione di Docker per ogni versione di Windows
- È in grado di eseguire solo immagini create per la stessa versione di Docker in esecuzione

Docker su Windows tramite WSL 2

- È una macchina virtuale che esegue Linux
- Il daemon è installato su Linux nella macchina virtuale
- Il client è installato su Windows
- Utilizzo trasparente

Docker Desktop

- Tool ufficiale
- Contiene tutti gli strumenti di Docker
- Disponibile per Windows, Linux, Mac
- Non è open source

E ora...

Spostiamoci su Docker

Comandi utili

Scarica immagine di ubuntu

```
docker pull ubuntu
```

Lancia ubuntu in un nuovo container temporaneo con nome "my_container"

```
docker run -it --rm --name my_container ubuntu
```

Mostra container in esecuzione

```
docker ps
```

Mostra tutti i container

```
docker ps --all
```

Apri un collegamento al processo principale di un container esistente

```
docker attach my_container
```

Apri una nuova shell in un container esistente

```
docker exec -it my_container bash
```

Lancia ubuntu mappando la cartella "/hello" a "C:\Hello"

```
docker run -it --rm -v C:\Hello:/hello ubuntu
```

Filesystem

- Ogni container ha un filesystem isolato
- È possibile esporre al container una cartella dell'host tramite `-v HOST:CONTAINER`, ad esempio `-v C:\Hello:/hello` per esporre `C:\Hello` su `/hello`
- È possibile creare cartelle condivise tra più container

Rete

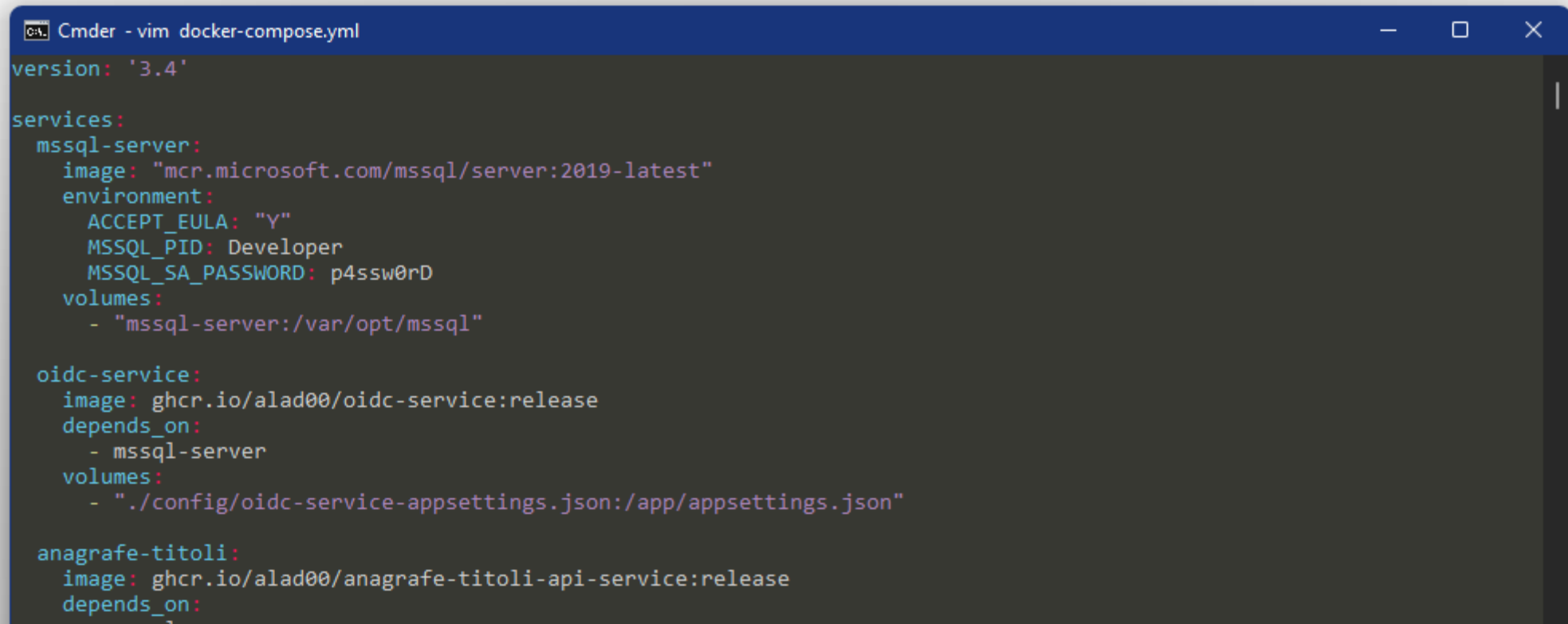
- Ogni container ha un suo localhost isolato
- È possibile esporre all'host una porta del container tramite `-p HOST:CONTAINER`, ad esempio `-p 8080:80` per esporre la porta 80 sulla porta 8080 dell'host
- È possibile creare reti virtuali condivise tra container

Docker Compose

E tutti i container vissero in armonia e serenità...

Docker compose

- Permette di configurare i container tramite un file YAML
- I container condividono la rete e possono comunicare

A screenshot of a terminal window titled 'Cmder - vim docker-compose.yml'. The window displays a YAML configuration for Docker Compose. The configuration includes a version field set to '3.4' and a services section with three services: mssql-server, oidc-service, and anagrafe-titoli. The mssql-server service uses the image 'mcr.microsoft.com/mssql/server:2019-latest' and sets environment variables for ACCEPT_EULA, MSSQL_PID, and MSSQL_SA_PASSWORD. It also mounts a volume for the database. The oidc-service and anagrafe-titoli services use images from ghcr.io and depend on the mssql-server service. The oidc-service also mounts a volume for its configuration file.

```
version: '3.4'

services:
  mssql-server:
    image: "mcr.microsoft.com/mssql/server:2019-latest"
    environment:
      ACCEPT_EULA: "Y"
      MSSQL_PID: Developer
      MSSQL_SA_PASSWORD: p4ssw0rD
    volumes:
      - "mssql-server:/var/opt/mssql"

  oidc-service:
    image: ghcr.io/alad00/oidc-service:release
    depends_on:
      - mssql-server
    volumes:
      - "./config/oidc-service-appsettings.json:/app/appsettings.json"

  anagrafe-titoli:
    image: ghcr.io/alad00/anagrafe-titoli-api-service:release
    depends_on:
```

Struttura

```
version: "3.8"
```

```
services:
```

```
  db:
```

```
    image: mysql
```

```
    command: --default-authentication-plugin=mysql_native_password
```

```
    restart: always
```

```
    environment:
```

```
      MYSQL_ROOT_PASSWORD: password
```

```
    volumes:
```

```
      - "mysql-data:/var/lib/mysql"
```

```
  web-server:
```

```
    image: httpd:2.4
```

```
    ports:
```

```
      - 8080:80
```

```
    volumes:
```

```
      - "./htdocs:/usr/local/apache2/htdocs"
```

```
volumes:
```

```
  mysql-data:
```

version

version: "3.8"

- Indica il numero di versione del file Docker Compose

services

- Configura i container
- Si indica un'immagine ed opzionalmente una versione
- Variabili d'ambiente passate al container
- Porte esposte all'host
- Volumi

```
services:
  db:
    image: mysql
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: password
    volumes:
      - "mysql-data:/var/lib/mysql"
  web-server:
    image: httpd:2.4
    ports:
      - 8080:80
    volumes:
      - "./htdocs:/usr/local/apache2/htdocs"
```

<https://docs.docker.com/compose/compose-file/#services-top-level-element>

volumes

```
volumes:  
  mysql-data:
```

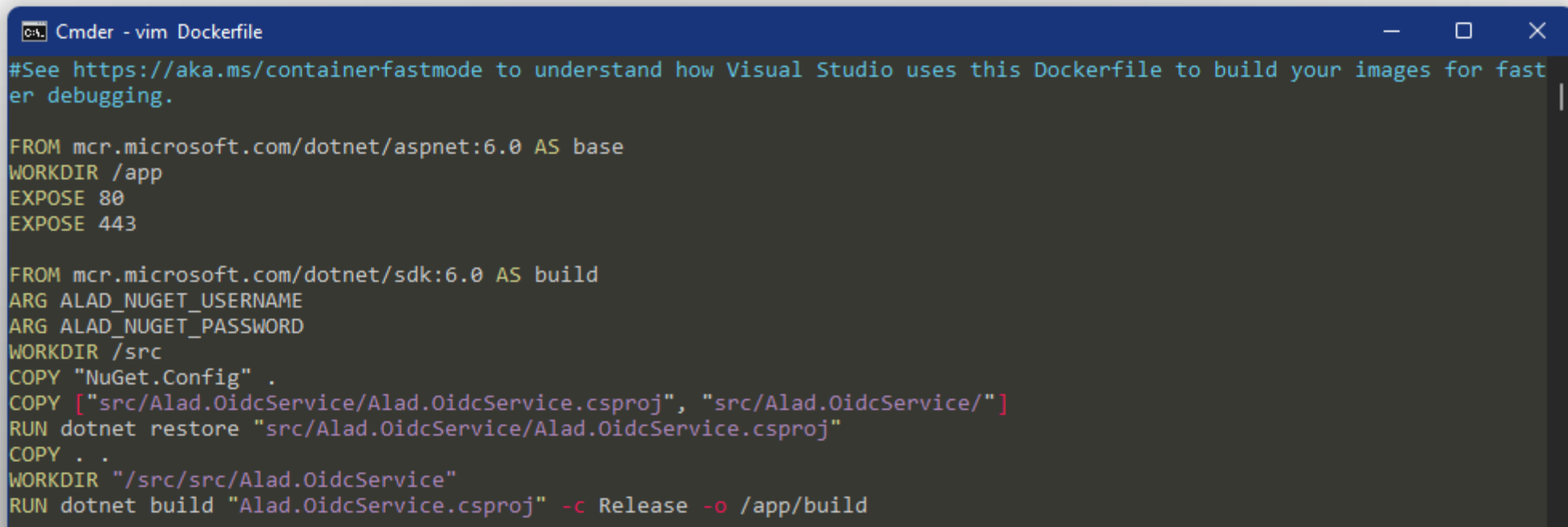
- Configura i volumi

Dockerfile

Per creare nuove immagini

Dockerfile

- È la ricetta per creare un'immagine per Docker
- Sequenza di istruzioni
- Ogni istruzione genera un layer

A screenshot of a terminal window titled "Cmdr - vim Dockerfile". The terminal displays a Dockerfile with instructions for building a .NET application. The instructions include setting the base image to mcr.microsoft.com/dotnet/aspnet:6.0, setting the working directory to /app, exposing ports 80 and 443, and then building the application using mcr.microsoft.com/dotnet/sdk:6.0. The build process involves copying the NuGet.Config file, the Alad.OidcService.csproj file, and the source code, followed by running dotnet restore and dotnet build. The final output is the built application located at /app/build.

```
Cmdr - vim Dockerfile
#See https://aka.ms/containerfastmode to understand how Visual Studio uses this Dockerfile to build your images for faster debugging.

FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
ARG ALAD_NUGET_USERNAME
ARG ALAD_NUGET_PASSWORD
WORKDIR /src
COPY "NuGet.Config" .
COPY ["src/Alad.OidcService/Alad.OidcService.csproj", "src/Alad.OidcService/"]
RUN dotnet restore "src/Alad.OidcService/Alad.OidcService.csproj"
COPY . .
WORKDIR "/src/src/Alad.OidcService"
RUN dotnet build "Alad.OidcService.csproj" -c Release -o /app/build
```

Struttura

```
FROM ubuntu  
WORKDIR /app
```

```
COPY . .
```

```
ENTRYPOINT ["echo", "ok"]
```

Struttura complessa

```
FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:6.0 AS build
WORKDIR /src
COPY "Example.csproj" .
RUN dotnet restore "Example.csproj"
COPY . .
RUN dotnet build "Example.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "Example.csproj" -c Release -o
/app/publish

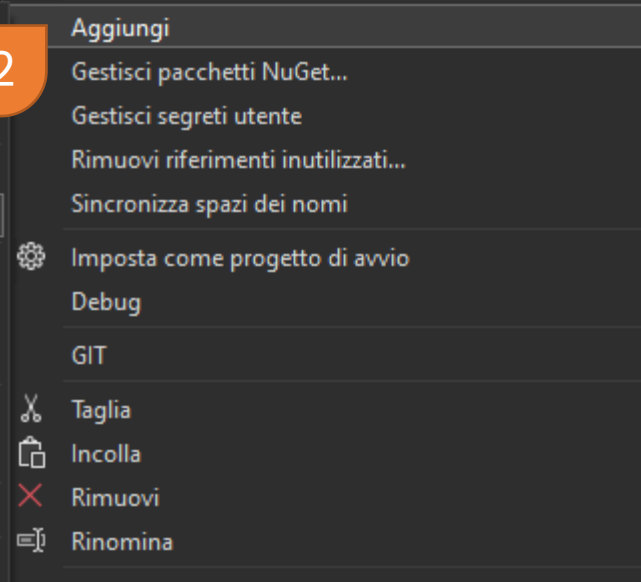
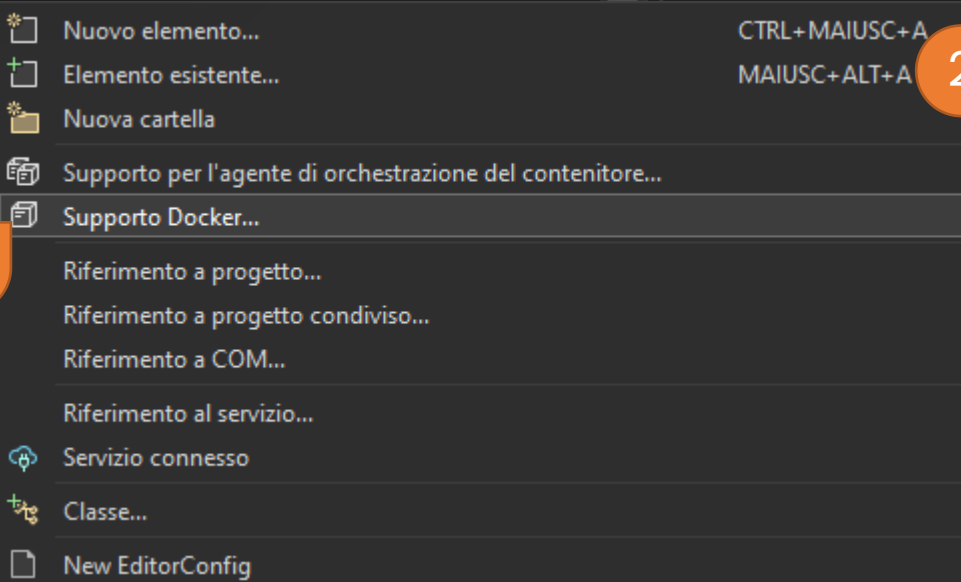
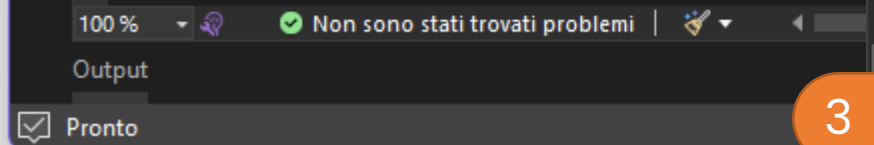
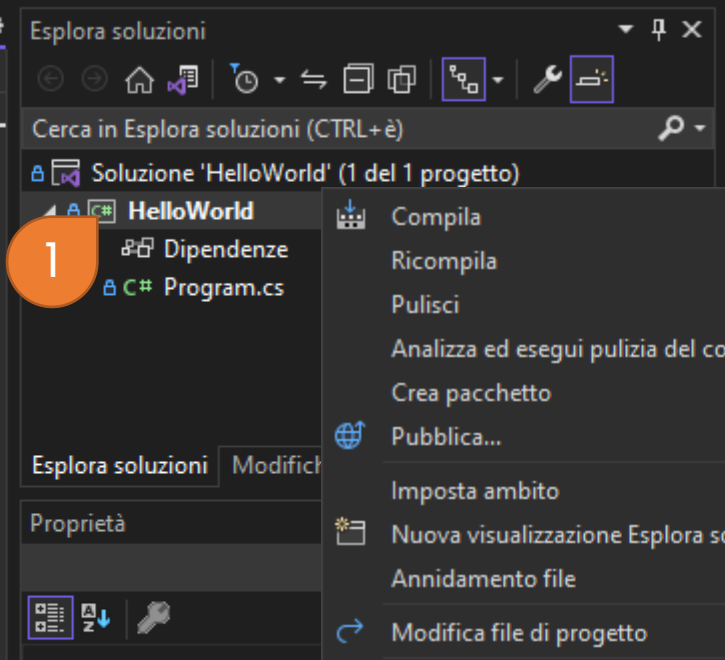
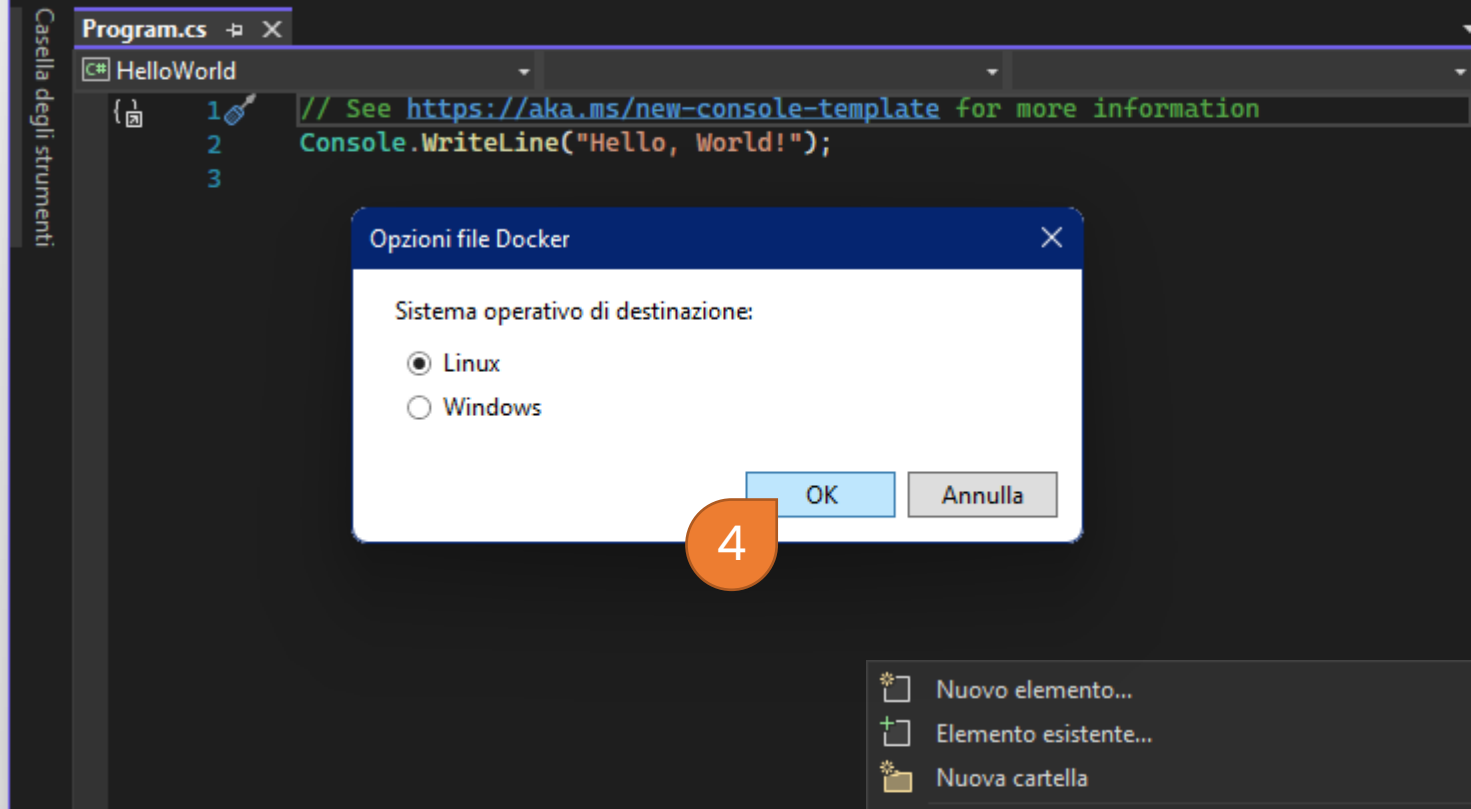
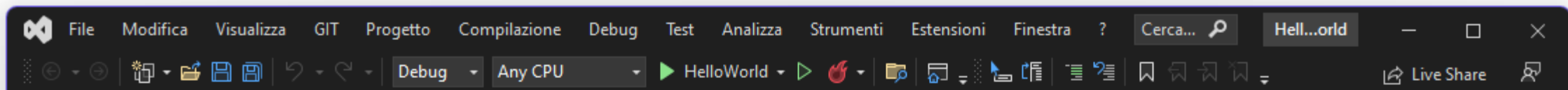
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Example.dll"]
```

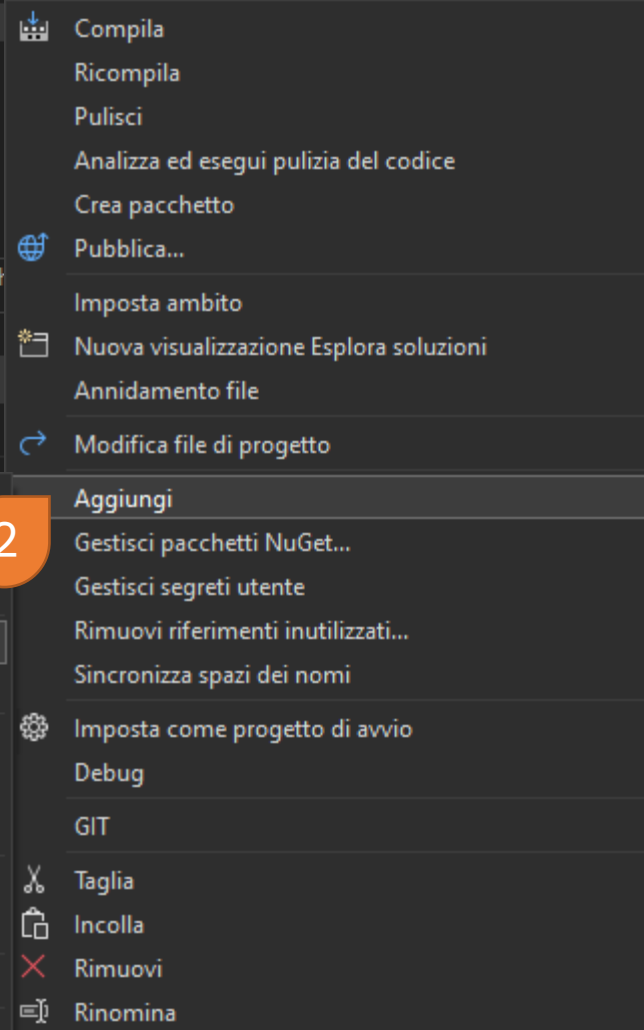
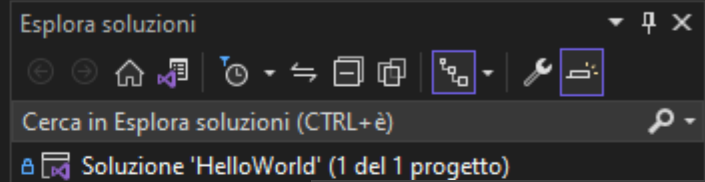
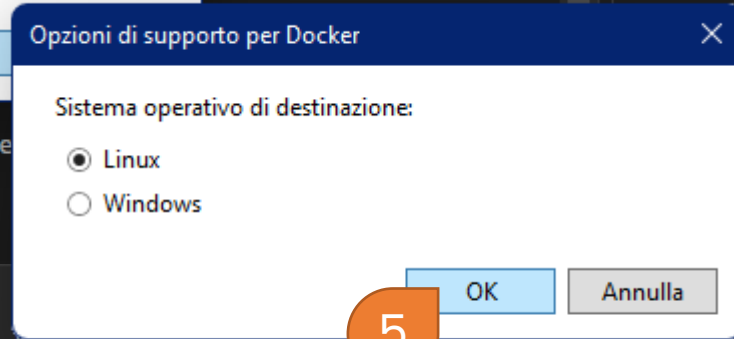
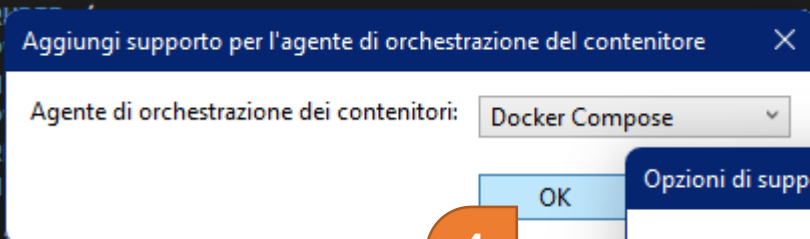
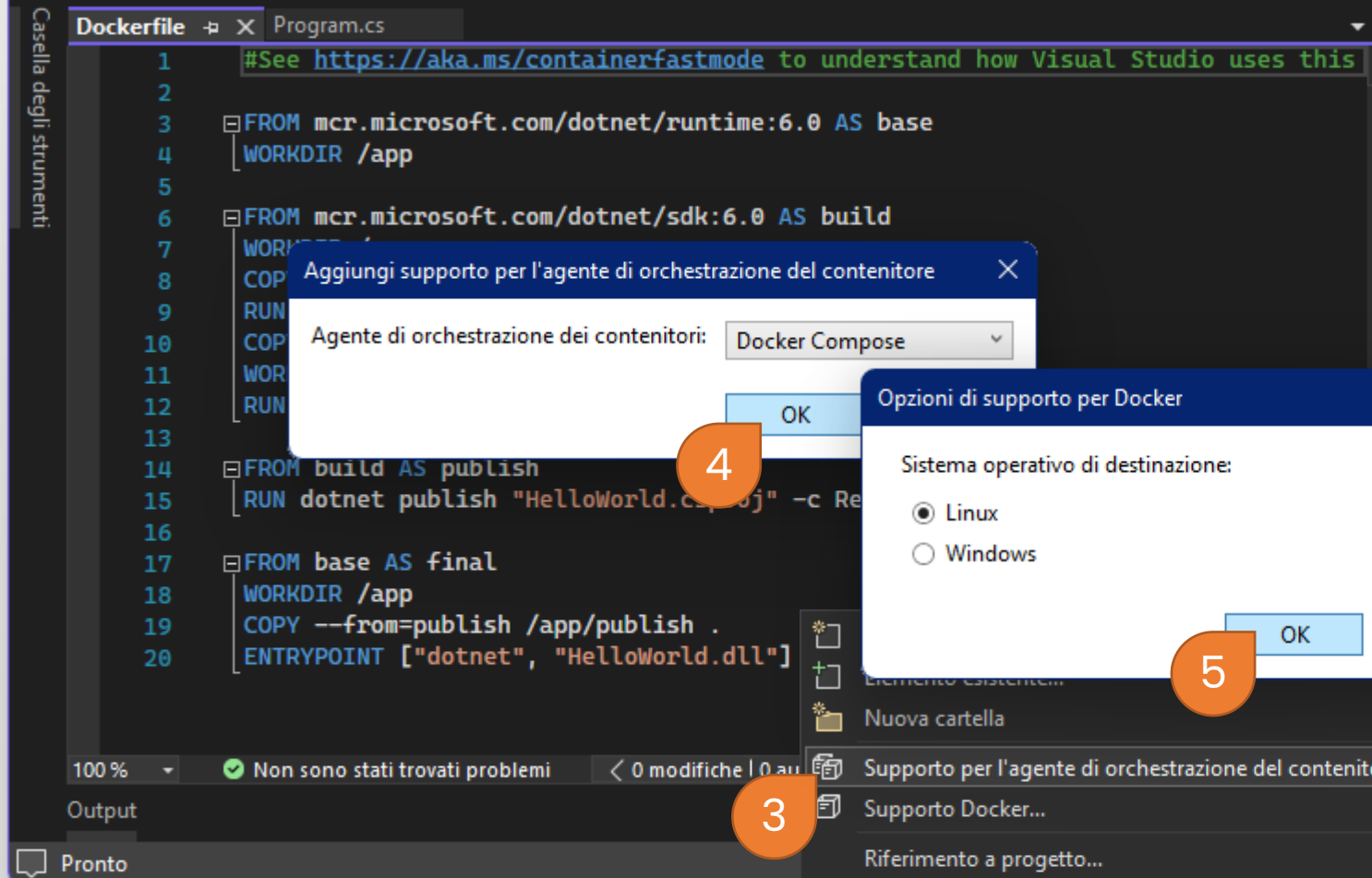
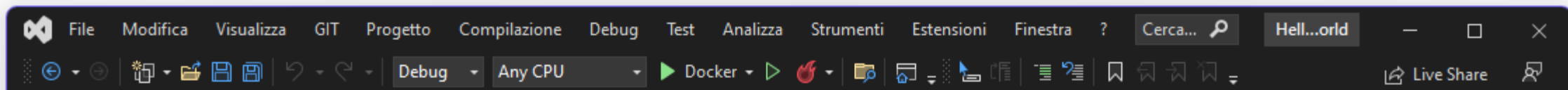
.dockerignore

- Permette di escludere file dall'istruzione COPY
- Sintassi simile al .gitignore

Docker su Visual Studio

Come containerizzare un progetto .NET





Visual Studio interface showing a Docker Compose project setup.

Menu Bar: File, Modifica, Visualizza, GIT, Progetto, Compilazione, Debug, Test, Analizza, Strumenti, Estensioni, Finestra, ?

Toolbar: Docker Compose, Live Share

Debugger: Debug, Any CPU, docker-compose

Code Editor (Dockerfile):

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="15.0" Sdk="Microsoft.Docker.Sdk">
  <PropertyGroup Label="Globals">
    <ProjectVersion>2.1</ProjectVersion>
    <DockerTargetOS>Linux</DockerTargetOS>
    <ProjectGuid>3ed60c5d-4873-4c7e-812b-818cf9964aee</ProjectGuid>
  </PropertyGroup>
  <ItemGroup>
    <None Include="docker-compose.override.yml">
      <DependentUpon>docker-compose.yml</DependentUpon>
    </None>
    <None Include="docker-compose.yml" />
    <None Include=".dockerignore" />
  </ItemGroup>
</Project>
```

Esplora soluzioni (Solution Explorer):

- Soluzione 'HelloWorld' (2 di 2 progetti)
- docker-compose
 - .dockerignore
 - docker-compose.yml
- HelloWorld
 - Dipendenze
 - Properties
 - Dockerfile

Proprietà (Properties):

docker-compose Project Properties

Project File	docker-compose.dcproj
Project Folder	C:\Users\Fabiolotti\Desktop\c

Project File
The name of the project file.

Output: Pronto

Status Bar: 100 % | Non sono stati trovati problemi | 0 modifiche | 0 autori, 0 modifiche | Ri: 1 | Car: 1 | SPAZIO | CRLF

Risorse esterne

Docker <https://www.docker.com/>

DockerHub <https://hub.docker.com/>

Docker Desktop <https://www.docker.com/products/docker-desktop/>

Grazie!