

PROGRAMMAZIONE ORIENTATA AI MICROSERVIZI

Container e Kubernetes

Tommaso Nanu

tommaso.nanu@unipr.it



UNIVERSITÀ DI PARMA

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE

Corso di Laurea in Informatica

Presentazione di Fabio Iotti
fabio.iotti@alad.cloud

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

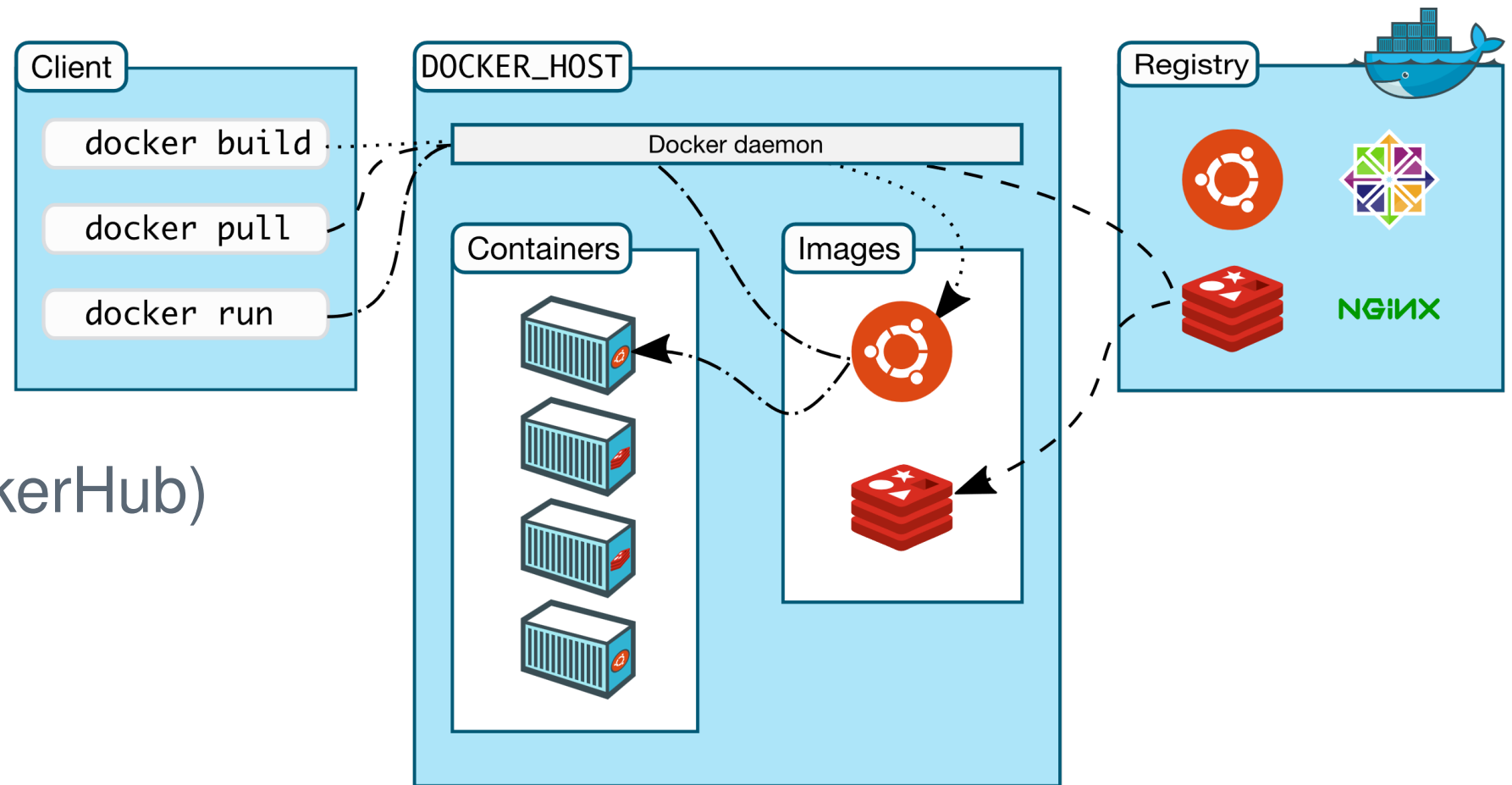
- Ambienti isolati
- Hypervisor (hardware)
- Bootloader (BIOS/UEFI)
- Kernel separato dall'host
- Dischi virtualizzati
Schede di rete virtualizzate
ecc...
- Avvio lento (boot sistema operativo)
- Overhead a runtime
- Guest ed host possono essere sistemi operativi diversi con kernel diversi

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

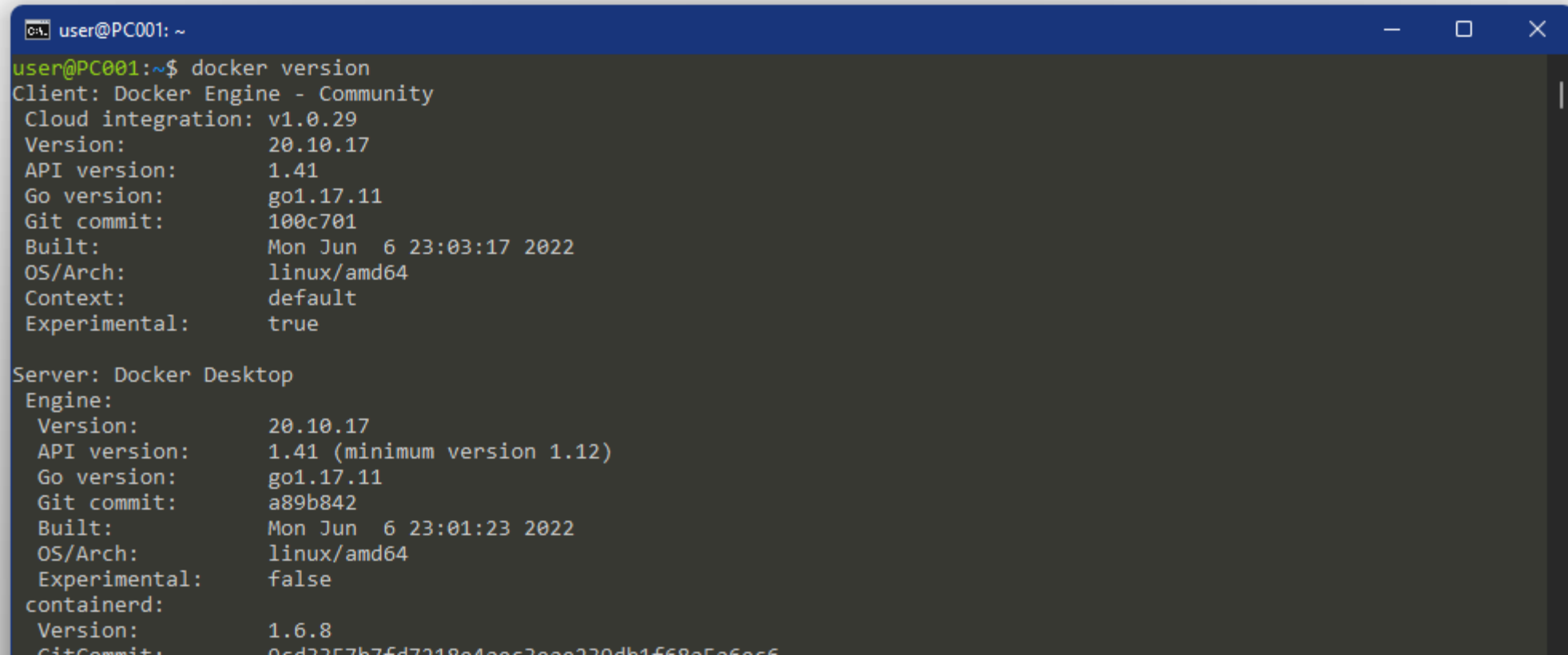


- Daemon
- Client
- Registro (DockerHub)
- Immagini
- Containers
- Volumi



Client (docker)

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

A terminal window with a dark blue title bar containing the text 'user@PC001: ~' and standard window control icons. The terminal output shows the command 'docker version' and its results, divided into 'Client' and 'Server' sections. The client is Docker Engine - Community, and the server is Docker Desktop. Both sections list various version and configuration details.

```
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: 0cd2257b7f5d7218e4a6c2e6e230db1f668e5e6e6
```

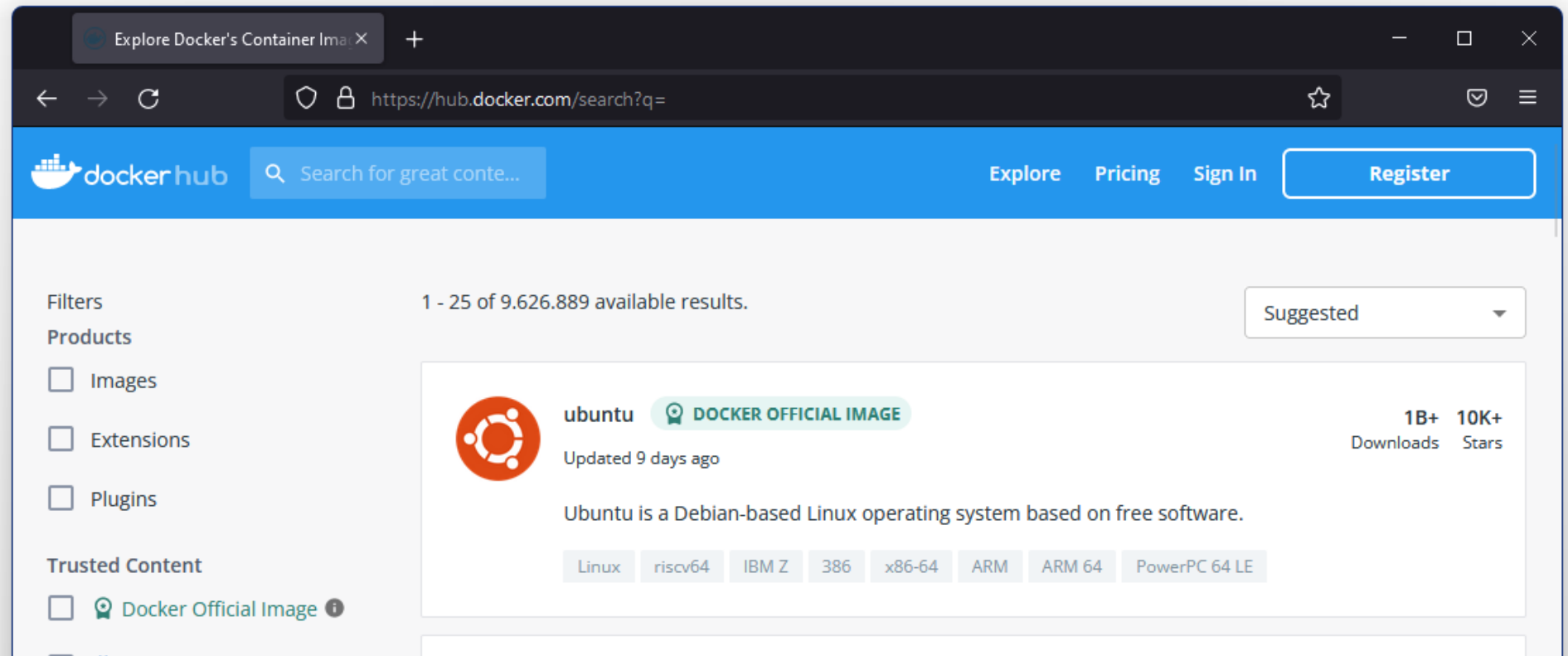
Daemon (dockerd)

- 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
```

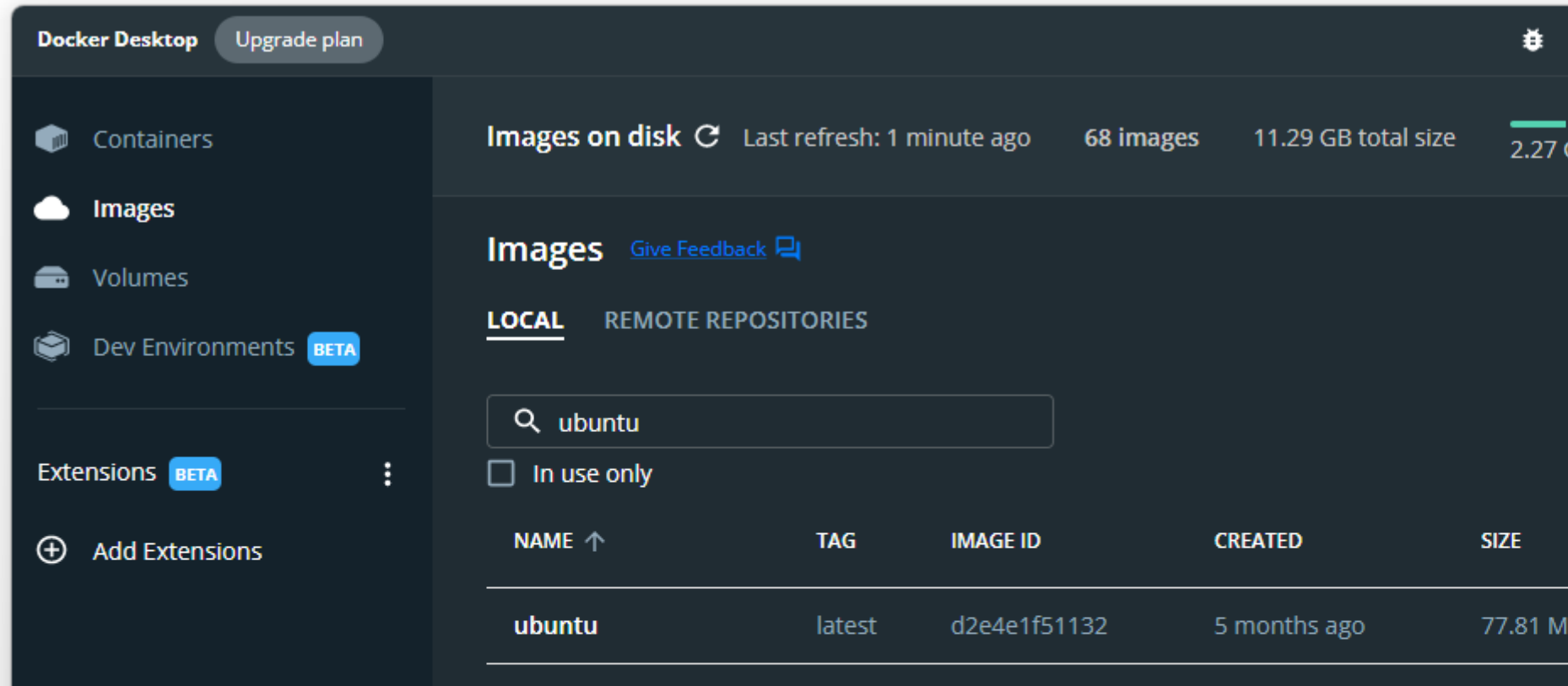
Registro (DockerHub)

- Interfaccia web
- Contiene le immagini pubbliche



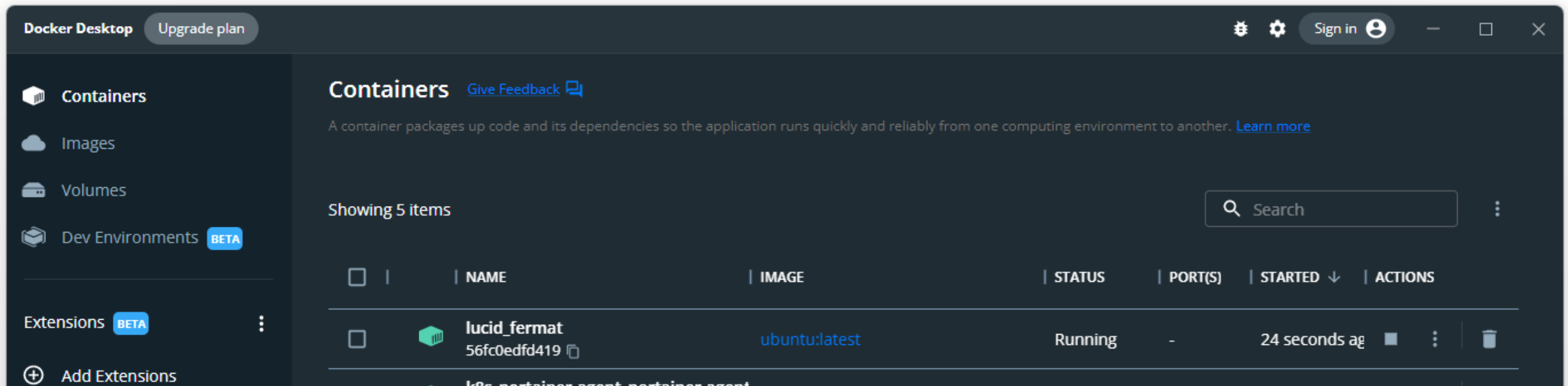
Immagini

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



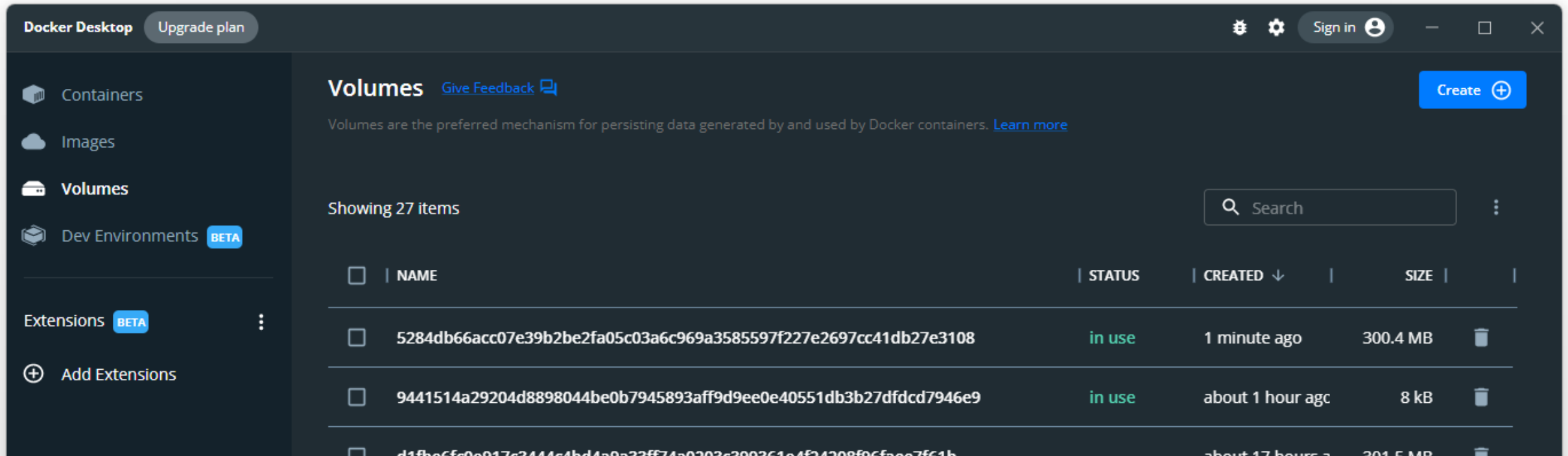
Containers

- 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

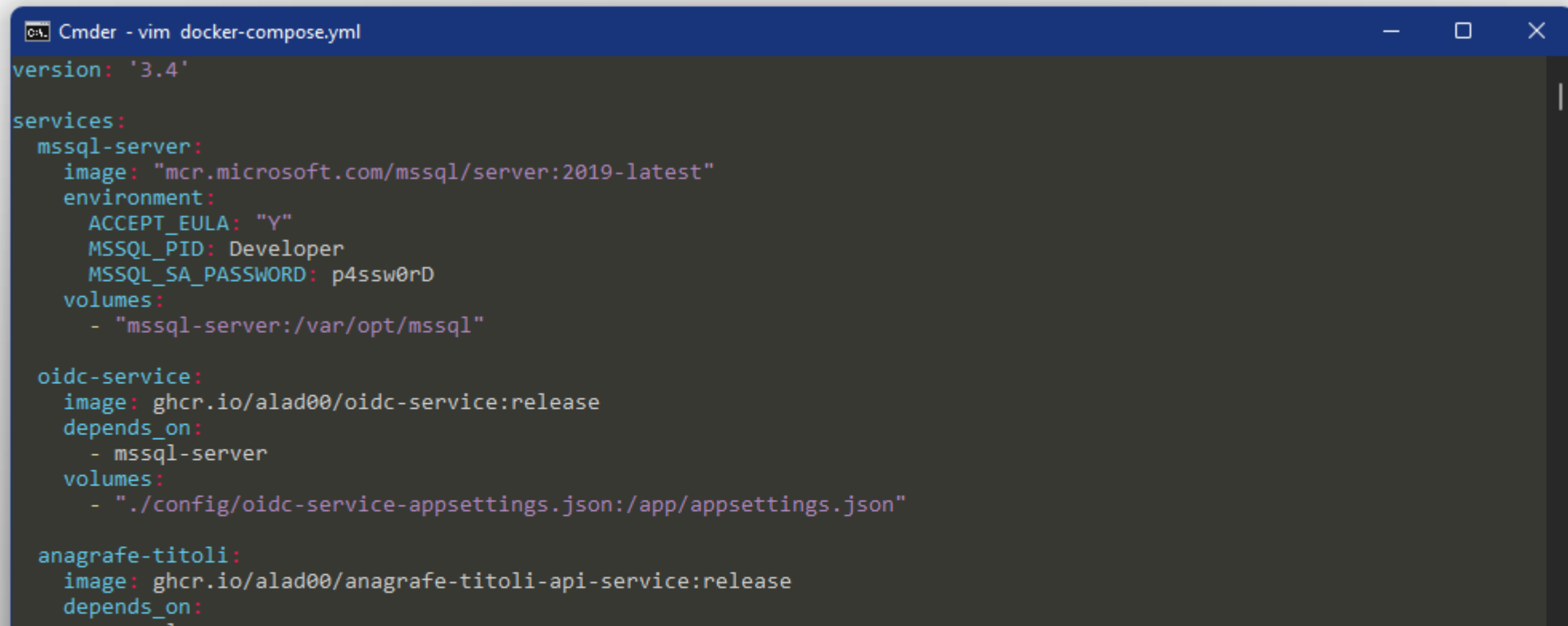
- Ogni container ha un suo localhost isolato
- È possibile esporre all'host una porta del container tramite **-p HOST:CONTAINER**, ad esempio **-p 8080:80** per esporre la porta **80** sulla porta **8080** dell'host
- È possibile creare reti virtuali condivise tra container

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 with a dark blue title bar that reads "Cmder - vim docker-compose.yml". The terminal content shows a YAML configuration for Docker Compose. It starts with "version: '3.4'", followed by a "services:" section. Under "services:", there are three entries: "mssql-server", "oidc-service", and "anagrafe-titoli". Each entry specifies an "image", "environment" variables, and "volumes". The "mssql-server" service uses the image "mcr.microsoft.com/mssql/server:2019-latest" and has environment variables for "ACCEPT_EULA", "MSSQL_PID", and "MSSQL_SA_PASSWORD". The "oidc-service" depends on "mssql-server" and uses the image "ghcr.io/alad00/oidc-service:release". The "anagrafe-titoli" service uses the image "ghcr.io/alad00/anagrafe-titoli-api-service:release".

```
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:
```

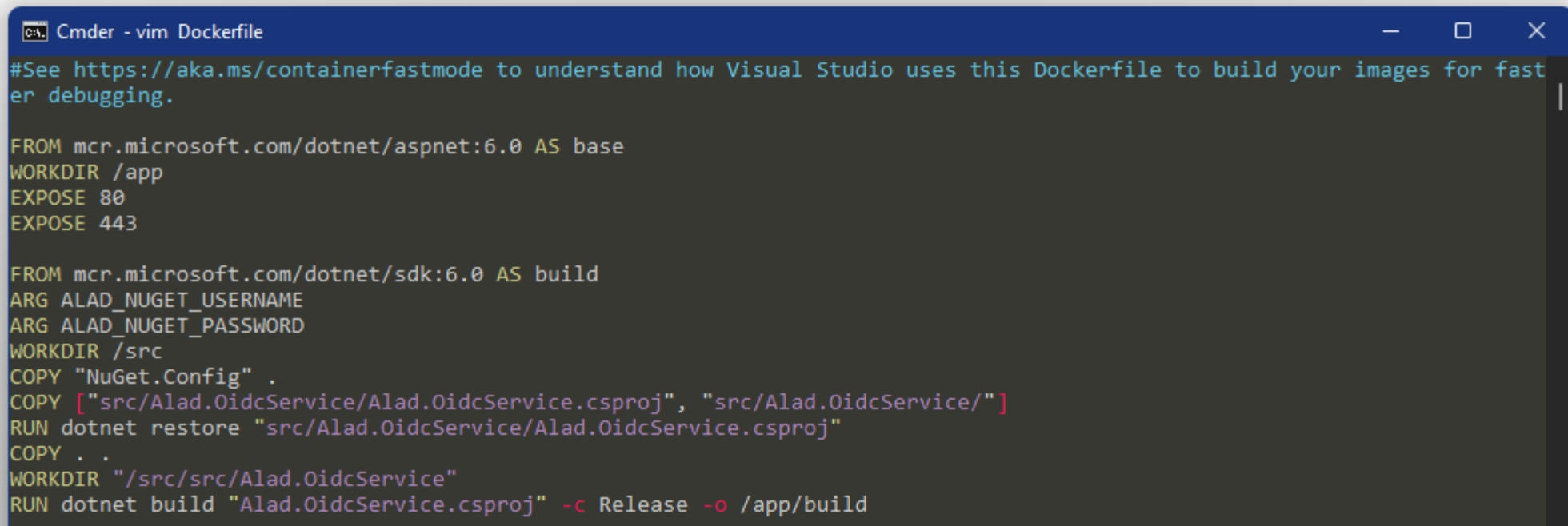
- Configura i volumi

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 text editor window titled "Cmdr - vim Dockerfile". The window 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, copying the project files from src/Alad.OidcService, restoring the NuGet packages, and building the application in Release mode to /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:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443
```

```
FROM mcr.microsoft.com/dotnet/sdk:7.0 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
```

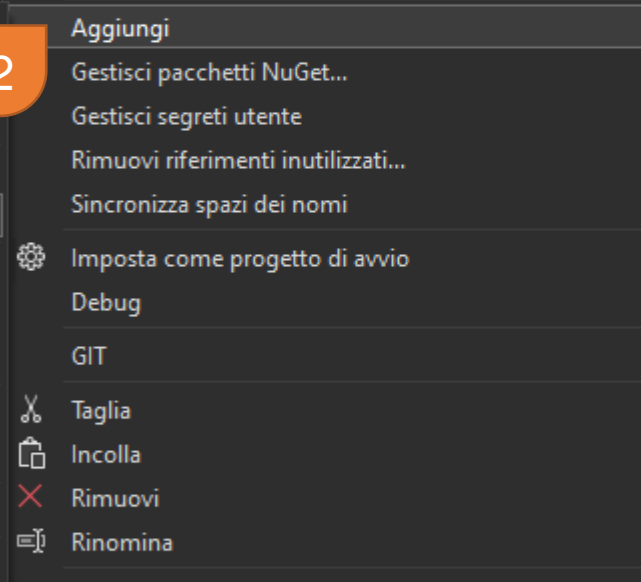
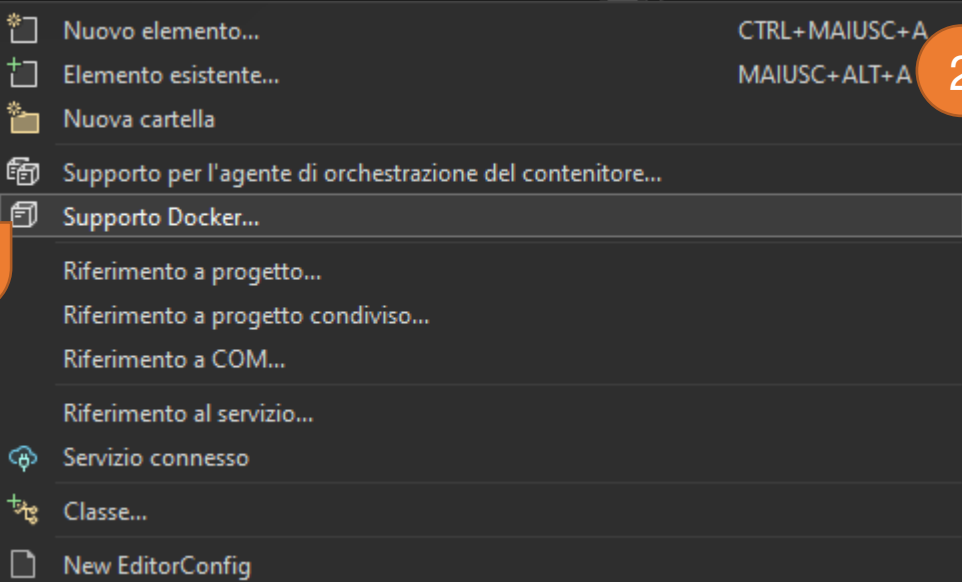
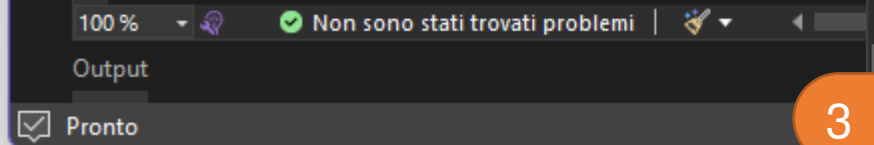
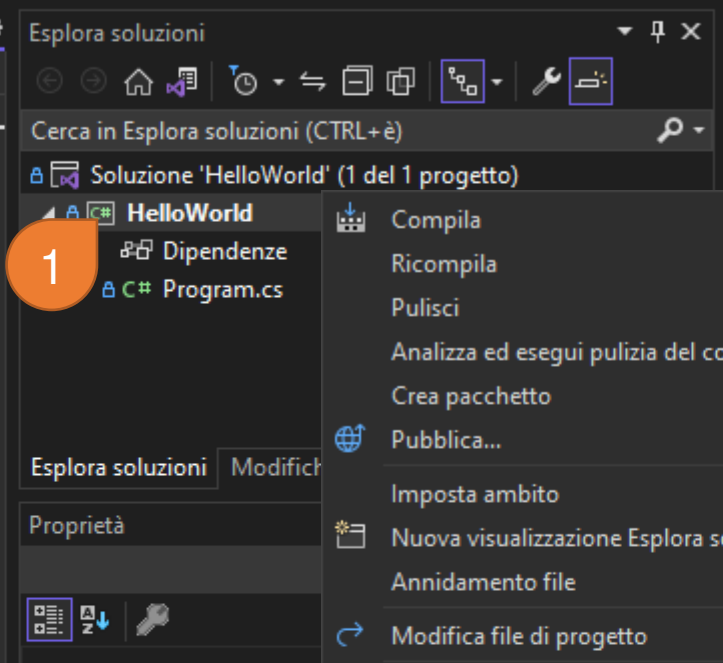
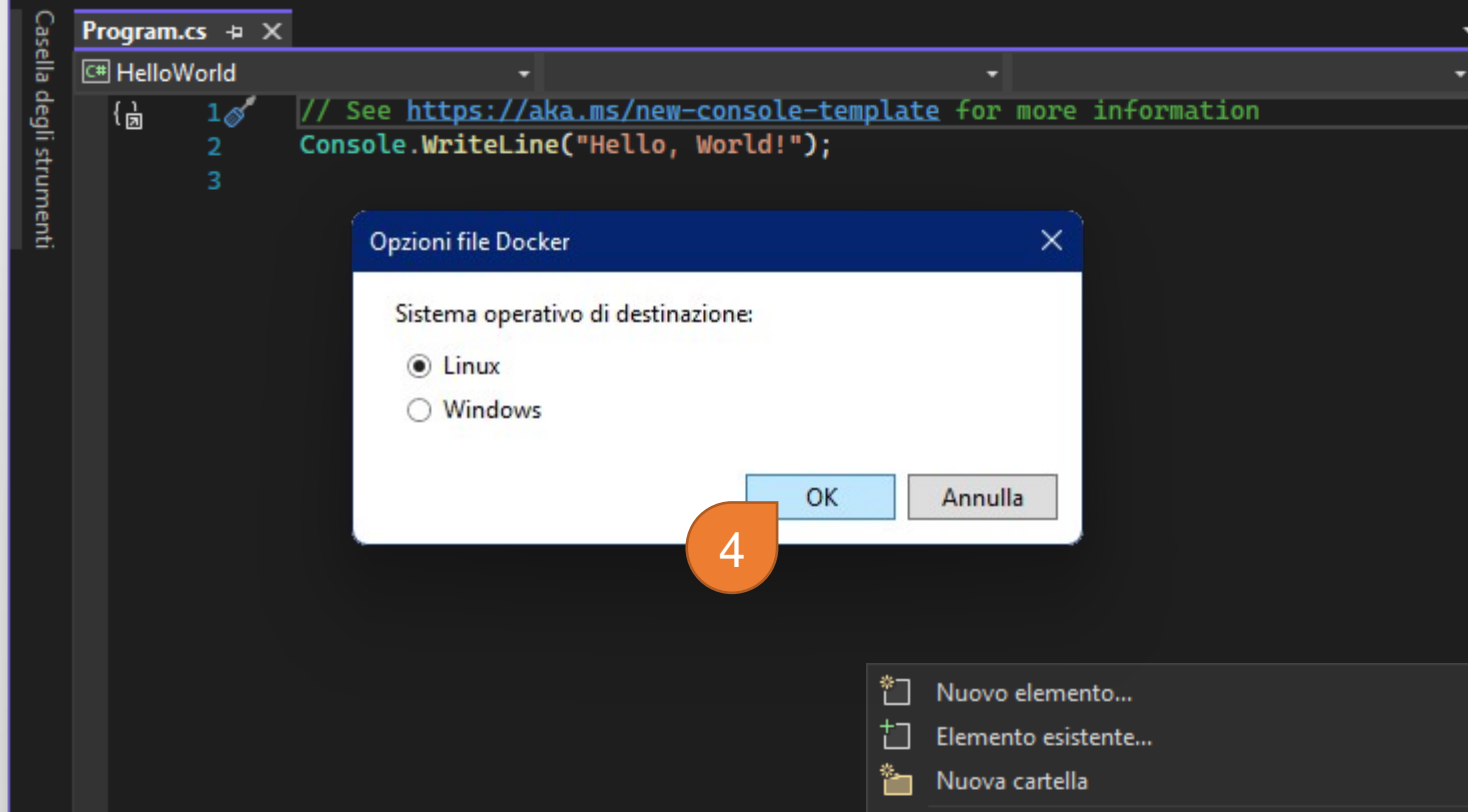
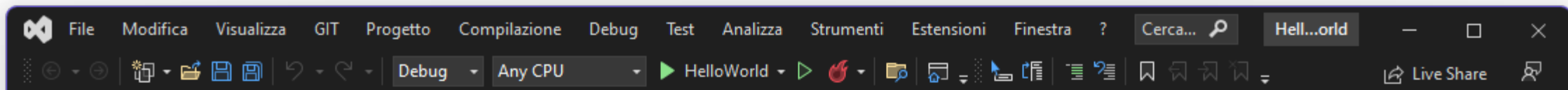
```
FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "Example.dll"]
```

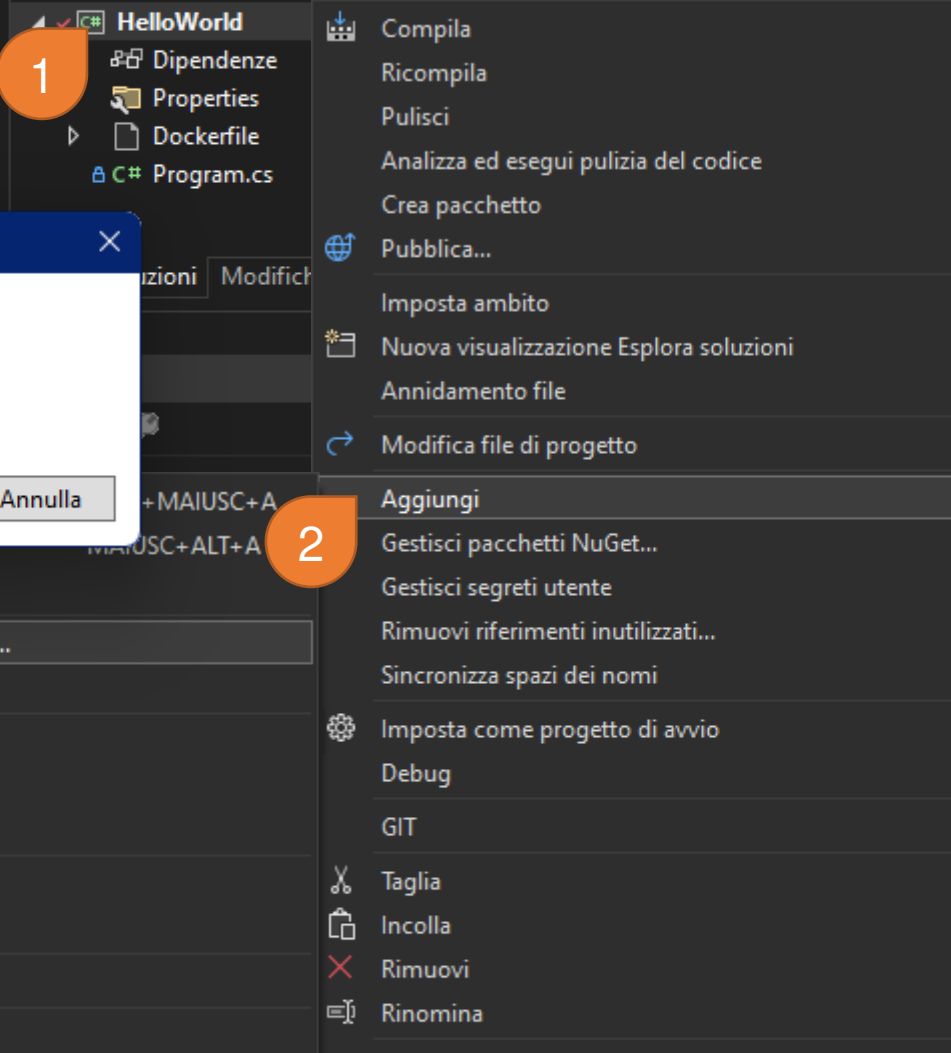
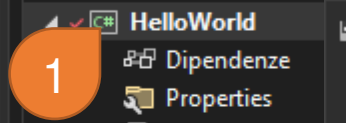
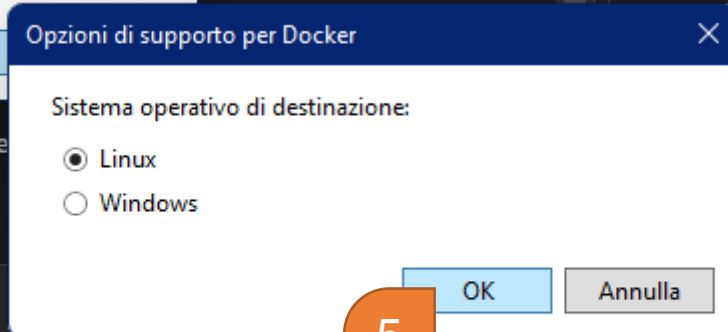
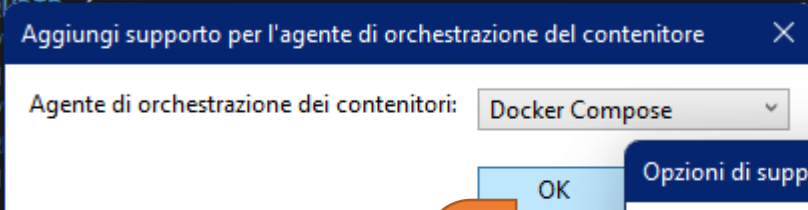
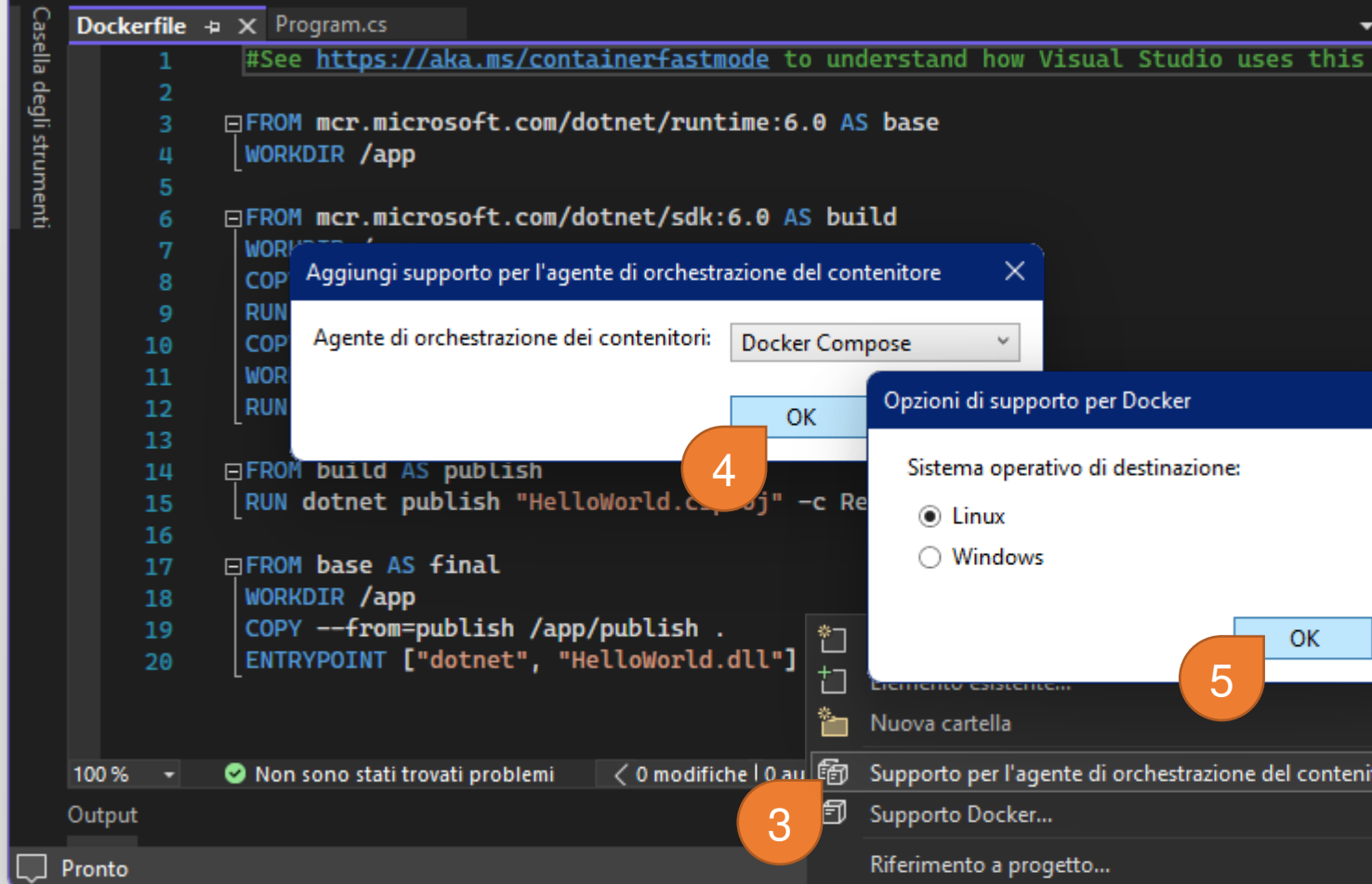
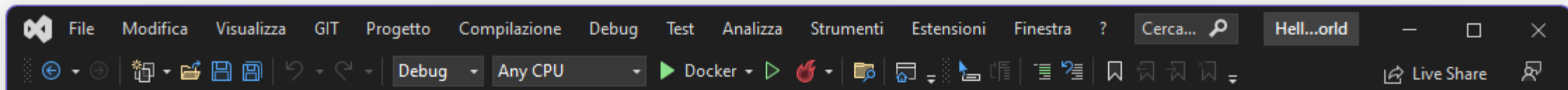
.dockerignore

- 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

Search Bar: Cerca... Hell...orld

Debugger: Debug, Any CPU, docker-compose

Code Editor: Dockerfile, Program.cs, docker-compose.dproj

```
<?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 (CTRL+è):

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

Proprietà:

docker-compose Project Properties

| Property | Value |
|----------------|-------------------------------|
| Project File | docker-compose.dproj |
| Project Folder | C:\Users\Fabiolotti\Desktop\c |

Project File: The name of the project file.

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

Output: Pronto

Bottom Bar: 0/0 8 master docker

Risorse esterne

Repository esempi <https://github.com/fiotti/unipr-docker-2023>

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

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

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



Kubernetes

Orchestrazione di container

Argomenti

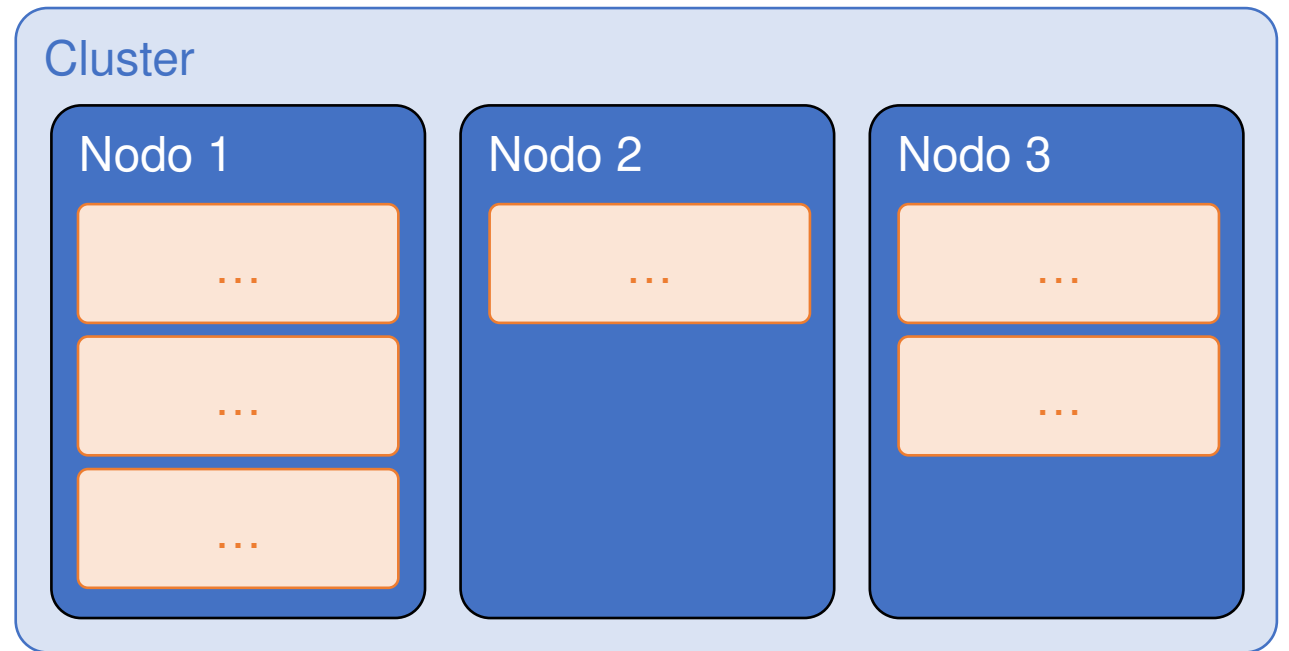
- Cos'è Kubernetes
- Le principali risorse di Kubernetes
 - Cluster e nodi
 - Pod
 - Deployment
 - Service
 - Ingress
- kubectl
- Configurazione tramite file YAML
- Kubernetes in cloud ed in locale

Kubernetes

- Orchestratore
- Gestisce carichi di lavoro containerizzati
- Configurazione dichiarativa
- Gestione automatizzata
- Facilita lo scaling orizzontale

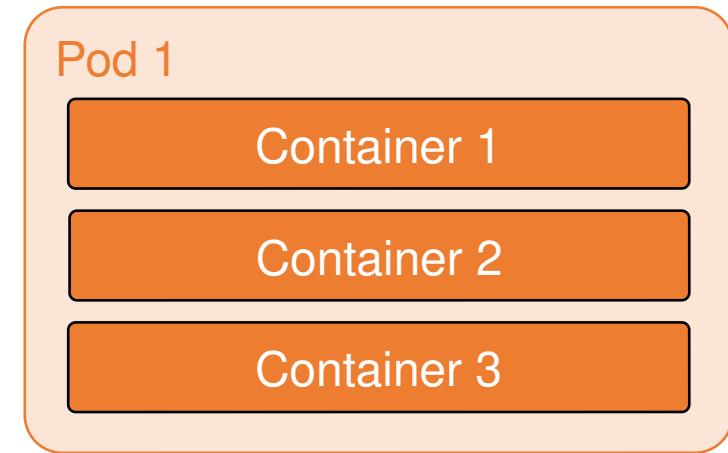
Cluster e nodi

- Un cluster è un gruppo di nodi interconnessi
- Un nodo è un singolo server che fa parte di un cluster



Pod

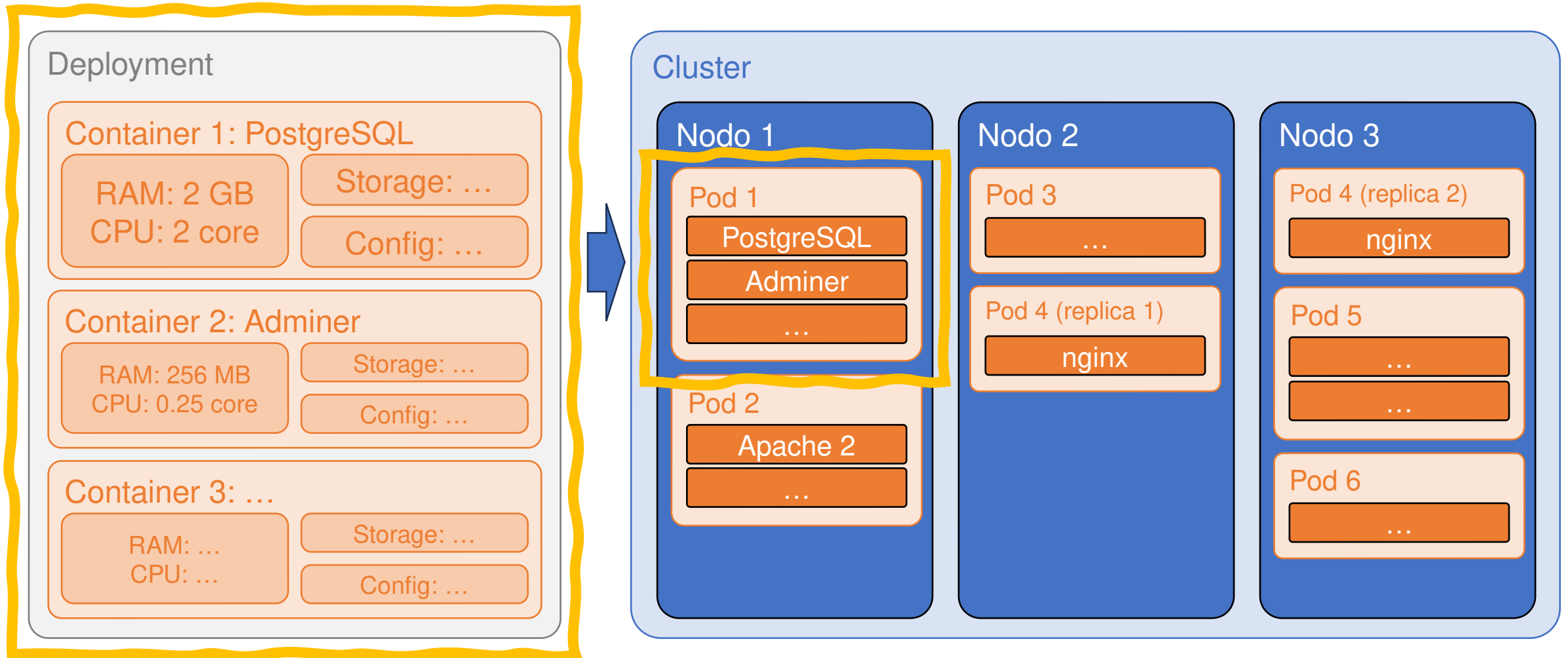
- Un pod è un gruppo di container
- I container in un pod condividono storage e rete



Deployment

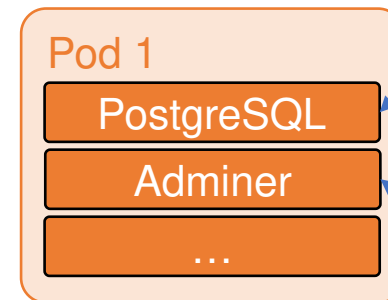
- I deployment descrivono lo stato desiderato del cluster
- Nei deployment si indica la configurazione di un pod
- Durante la definizione di un deployment è possibile indicare le risorse da associare ai vari container contenuti nel pod
 - Variabili d'ambiente
 - Volumi persistenti
 - File di configurazione o secret
 - Limiti di memoria e CPU
- Quando si applica un deployment, Kubernetes cerca un nodo che abbia le risorse necessarie per ospitarlo, e vi crea un nuovo pod

Deployment



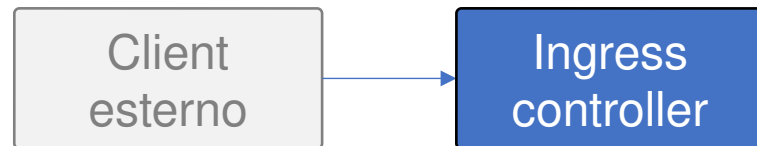
Service

- I service permettono di esporre porte dai container che rispondono ad un determinato selettore
- I pod/container possono fare riferimento ai service per comunicare con gli altri container, anche se si trovano sugli altri pod o nodi del cluster
- Kubernetes crea una rete virtuale che collega tutti i service che fanno parte di un namespace
- Un service può riferirsi a più di un container

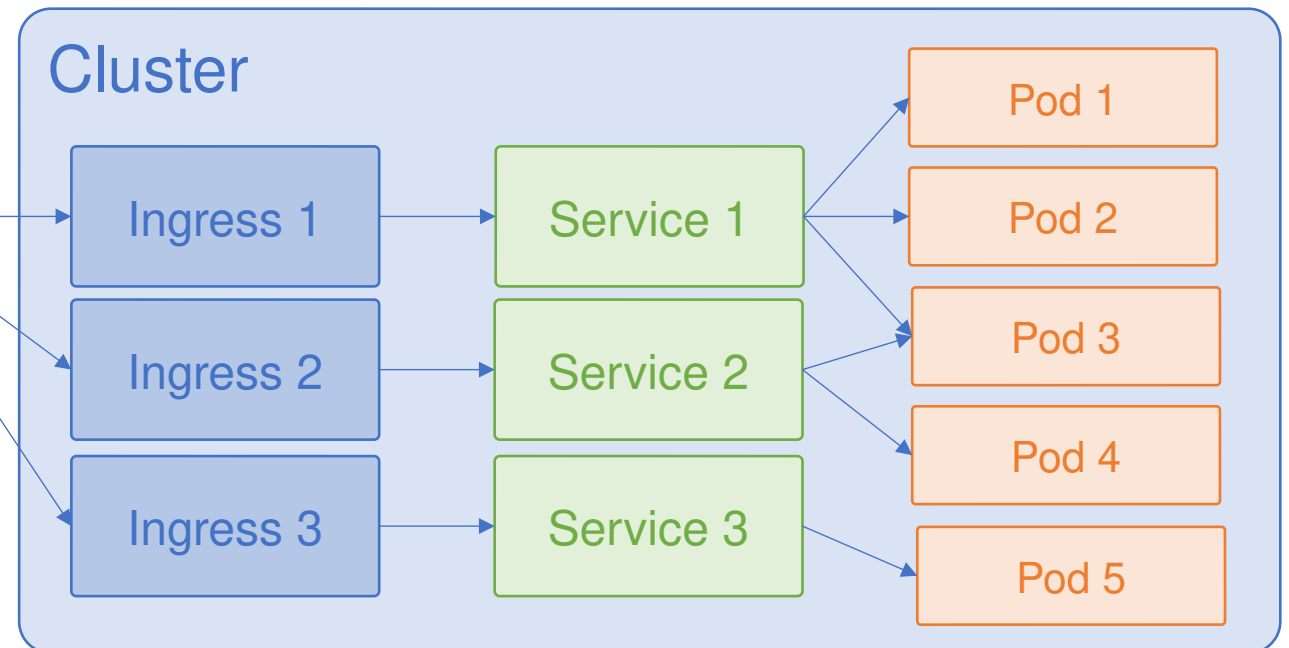


Ingress

- Alcuni servizi devono essere accessibili anche dall'esterno del cluster, per esempio nginx
- Gli ingress permettono di configurare l'accesso ai service del cluster dall'esterno
- Vengono gestiti da un ingress controller



- L'ingress controller è generalmente fornito dal cloud-provider e non fa parte del cluster Kubernetes





kubectl

L'interfaccia a linea di comando per Kubernetes

Comandi utili

Cerca nel cluster tutte le risorse che si trovano in qualsiasi namespace
kubectl get all --all-namespaces

Cerca nel cluster tutti i pod che si trovano nel namespace di default
kubectl get pods

Cerca nel cluster tutti i service che si trovano nel namespace indicato
kubectl get services -n **my-namespace**

Esporta su file il secret indicato
kubectl get secret **my-secret** -o yaml > **hello.yml**

Verifica quali modifiche verrebbero effettuate nel cluster applicando un file YAML
kubectl diff -f **hello.yml**

Applica un file YAML al cluster creando o aggiornando tutte le risorse indicate nel file
kubectl apply -f **hello.yml**

Elimina dal cluster tutte le risorse indicate in un file YAML
kubectl delete -f **hello.yml**

Descrive nel dettaglio lo stato un pod
kubectl describe pod **my-pod** -n **my-namespace**

Legge i log di un pod
kubectl logs **my-pod** -n **my-namespace**

Avvia un container effimero in un pod, utilizzabile per fare debugging
kubectl debug -it pods/**my-pod** -n **my-namespace** --image=**ubuntu**

Spegne temporaneamente tutti i pod che fanno parte di un deployment (0 per spegnere, 1 per avviare)
kubectl scale deploy **my-deployment** -n **my-namespace** --replicas=0
kubectl scale deploy **my-deployment** -n **my-namespace** --replicas=1

Spegne tutti i pod che fanno parte di un deployment e li ricrea nuovi
kubectl rollout restart deployment **my-deployment** -n **my-namespace**

È possibile rimpiazzare le parole sottolineate con una qualsiasi risorsa.

Le **parole in grassetto** rappresentano le parti variabili del comando.

File YAML di Kubernetes

```
# Esempio deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
              name: http-web-svc
```

- Contiene la descrizione di una risorsa
- Configurazione dichiarativa
- Può essere applicato al cluster
- Può essere aggiornato in un secondo tempo, e Kubernetes applicherà al cluster solo le modifiche

File YAML di Kubernetes

```
# Esempio service
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - name: name-of-service-port
    protocol: TCP
    port: 80
    targetPort: http-web-svc
```

```
# Esempio ingress
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: nginx-service
            port:
              number: 80
```

<https://kubernetes.io/docs/concepts/services-networking/service/>

<https://kubernetes.io/docs/concepts/services-networking/ingress/>

File YAML di Kubernetes

- I tipi di risorsa in Kubernetes più importanti sono quelli descritti nelle slide precedenti
- Sono presenti innumerevoli altri tipi di risorsa
- È possibile crearne di nuove non standard CRD (Custom Resource Definition)
- Fare riferimento alla documentazione di Kubernetes
- Fare riferimento alla documentazione delle estensioni

Kubernetes in cloud

- Kubernetes non è semplicissimo da installare
- Solitamente i cloud provider offrono Kubernetes
- Spesso vengono forniti volumi e load balancer integrati con i servizi e il DNS del cloud provider
- Impegno minimo, è quasi tutto preconfigurato
- Raramente Kubernetes è fornito come servizio gratuito

Kubernetes in locale

- Kubernetes è integrato in Docker Desktop
- In alternativa è possibile installarlo seguendo la documentazione ufficiale
- Ideale per lo sviluppo
- Installazione in produzione relativamente complessa

Risorse esterne

Repository esempi <https://github.com/fiotti/unipr-kubernetes-2023>

Kubernetes <https://kubernetes.io/>



Grazie!