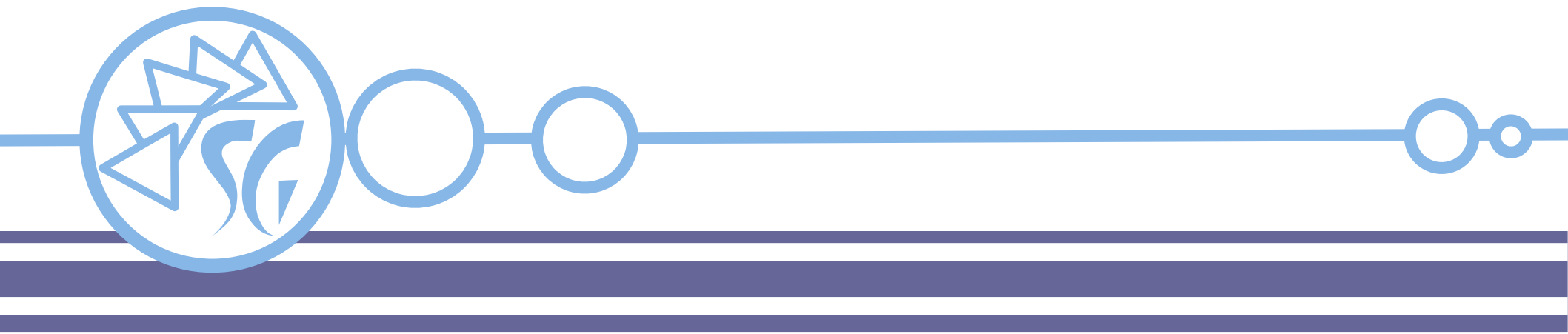

DOCKER Compose

Ing. Simone Giustetti

www.giustetti.net

Docker Compose

Sempre più spesso è richiesto agli sviluppatori di scrivere applicazioni che si appoggino su una moltitudine di servizi distinti.

Una applicazione web, ad esempio, si appoggia ad un web server, ad un RDBMS, a linguaggi di scripting ed altro ancora.

Docker Compose è uno strumento progettato per gestire applicazioni distribuite su più contenitori.

Vantaggi Offerti da Docker Compose

È possibile far girare più servizi all'interno del medesimo contenitore.

Distribuire i servizi su contenitori distinti garantisce:

- Maggior flessibilità di amministrazione;
- Facilità nell'eseguire aggiornamenti parziali;
- Maggior scalabilità;
- Miglior distribuzione del carico;
- Facilità di migrazione dei contenitori.

Docker / Docker Compose

Docker è usato per amministrare un singolo contenitore o più contenitori separatamente.

Docker Compose consente di amministrare molti contenitori all'unisono, come fossero una entità unica.

Funziona come Docker, ma permette di sviluppare ed amministrare applicazioni distribuite complesse.

Esempi d’Uso di Docker Compose

È stata scritta una applicazione web che fornisce servizi ai clienti dell’organizzazione. Una seconda applicazione fornisce supporto su richiesta. Entrambe le applicazioni si appoggiano al medesimo database.

Il server web non è più in grado di gestire il carico di lavoro per le due applicazioni.

Non avendo usato Docker Compose bisogna migrare e riconfigurare i servizi uno per volta.

Esempi d’Uso di Docker Compose

Usando Docker Compose, invece, il tempo di intervento è limitato alla migrazione del contenitore su un nuovo server.

È richiesto solo uno sforzo minimo o nullo per aggiornare la configurazione e verificare che i servizi migrati siano in grado di comunicare correttamente.

Esempi d’Uso di Docker Compose

Configurare **ambienti di sviluppo** che possono essere distribuiti facilmente a tutti gli sviluppatori.

Configurare ambienti per i **test automatizzati**. Usando un file per compose è possibile creare o distruggere tali ambienti con pochi comandi.

Configurare ambienti di produzione che possono essere migrati facilmente nel cloud.

File YAML

I servizi che costituiscono una applicazione sono configurati con un file in formato **YAML**.

Docker Compose consente di creare, avviare ed amministrare tutti i servizi con un solo comando.

Docker Compose supporta comandi per:

- Avviare, fermare e reinstallare i contenitori;
- Monitorare lo stato dei contenitori attivi;
- Redirigere lo output dei contenitori;
- Lanciare comandi ad un singolo contenitore.

Condizioni per Uso Ottimale

Per sfruttare al meglio Docker Compose l'applicazione deve:

- Essere composta da ambienti isolati;
- Preservare i dati quando il contenitore è creato;
- Reinstallare un contenitore solo se è stata aggiornata l'immagine;
- Utilizzare variabili di ambiente nel file di configurazione. Il valore delle variabili è impostato all'avvio di ogni contenitore.

Installare Docker Compose su Linux

Prima di Docker Compose bisogna installare alcuni prerequisiti:

- Docker.
- Docker CLI.

Alternativamente: installare **Docker Desktop** che contiene sia Docker Compose che i prerequisiti.

Per funzionare in Linux, **Docker Desktop** richiede che sia abilitata la virtualizzazione tramite **KVM**.

KVM

Abilitare le funzioni di virtualizzazione hardware via BiOS.

Caricare il modulo del kernel per la propria CPU:

```
modprobe kvm_amd    # AMD processors
modprobe kvm_intel  # Intel processors
```

Verificare che il modulo sia attivo:

```
lsmod | grep --color kvm
kvm_amd                167936  0
ccp                    126976  1 kvm_amd
kvm                   1089536  1 kvm_amd
irqbypass               16384  1 kvm
```

Installare Docker Desktop su Linux

Per Debian Gnu/Linux e derivate:

```
apt-get install ./docker-desktop-<version>-<arch>.deb
```

Per Red Hat Linux e derivate:

```
dnf install gnome-terminal
dnf install ./docker-desktop-<version>-<arch>.rpm
```

Avviare Docker Desktop:

```
systemctl --user start docker-desktop
```

Verificare l'installazione:

```
docker compose version
```

Installare Docker Compose su MacOsX

- Scaricare il file *Docker.dmg* per la piattaforma di destinazione: Apple Silicon oppure Intel.
- Nel caso di piattaforma **Apple Silicon**, installare **Rosetta 2** da terminale:

```
softwareupdate --install-rosetta
```

- Avviare la procedura di installazione mediante doppio click sul file *Docker.dmg*.
- Avviare l'applicazione *Docker.app* ed accettare la licenza d'uso.

Installare Docker Compose su Windows

- Abilitare il supporto alla virtualizzazione hardware nel BiOS.
- Scaricare il file *Docker Desktop Installer.exe*.
- Avviare l'installazione mediante doppio click sul file *Docker Desktop Installer.exe*.
- Preferire **WSL 2** a **Hyper-V** per piattaforme che supportino entrambi.
- Aggiungere le utenze degli sviluppatori al gruppo **docker-users**.
- Avviare l'applicazione **Docker-Desktop** ed accettare la licenza d'uso.

Configurare una Applicazione Client / Server

Prerequisiti:

1) Creare una cartella dedicata per l’applicazione.
2) Creare il file *compose.yaml* che conterrà la configurazione dei contenitori.
3) Creare la cartella server.
4) Creare la cartella client.

Configurare il Server

1) Spostarsi nella cartella *server*.
2) Creare il file *Dockerfile* contenente la configurazione del contenitore.
3) Copiare i file necessari a far girare il contenitore e l'applicazione ivi contenuta.
4) Avviare il contenitore.

Configurare il Client

1) Spostarsi nella cartella *client*.
2) Creare il file *Dockerfile* contenente la configurazione del contenitore.
3) Copiare i file necessari a far girare il contenitore e l'applicazione ivi contenuta.
4) Avviare il contenitore.
5) Interrogare l'applicazione per verificare che comunichi correttamente con il server.

Compilare ed Avviare una Applicazione

1) “Compilare” l’applicazione:
   `docker compose` `build`

2) Avviare l’applicazione:
   `docker compose` `up`

3) Interrogare l’applicazione server per verificare che giri.
4) Interrogare l’applicazione client per verificare che giri e comunichi correttamente con il server.

Docker Compose – Comandi di Avvio

Avviare tutti i contenitori:

```
docker compose up
docker compose up -d  # Avvio in background
```

Avviare solo un contenitore ed i prerequisiti:

```
docker compose run <contenitore>
```

Avviare un contenitore fermo:

```
docker compose start <contenitore>
```

Riavviare forzatamente tutti i contenitori:

```
docker compose restart
```

Docker Compose – Comandi di Fermo

Fermare tutti i contenitori:

```
docker compose down
```

Fermare un contenitore (Ha un timeout predefinito di 10 secondi):

```
docker compose stop <contenitore>
```

Terminare forzatamente i contenitori:

```
docker compose kill
```

Mettere in pausa, far ripartire i contenitori:

```
docker compose pause
docker compose unpause
```

Docker Compose – Controllo dei Processi

Ottenere un elenco di applicazioni attive:

```
docker compose ls
```

Ottenere un elenco dei contenitori attivi:

```
docker compose ps
```

Ottenere un elenco delle immagini usate dai contenitori:

```
docker compose images
```

Docker Compose – Comandi di Compilazione

“Compilare” tutti i contenitori:

```
docker compose build
```

Creare tutti i contenitori di una applicazione:

```
docker compose create
```

Docker Compose – Comandi di Diagnostica

Leggere i file di log di un contenitore:

```
docker compose logs <contenitore>
docker compose logs -f  # Aggiorna i log
```

Monitorare le prestazioni di tutti i processi di una applicazione:

```
docker compose top
```

Docker Compose – Comandi di Comandi

Eseguire un comando in un contenitore:

```
docker compose exec <contenitore> <comando>
```

Il File compose.yaml

Il file ha una sintassi simile a JSon:

```
services:
web:
    build: .
    ports:
        - "8000:5000"
    volumes:
        - .:/code
        - logvolume01:/var/log
    depends_on:
        - redis
redis:
    image: redis
volumes:
logvolume01: {}
```

Il Nome del File di Configurazione

Il nome predefinito del file di configurazione è: *compose.yaml*, ma il programma riconosce anche *compose.yml*, *docker-compose.yaml* o *docker-compose.yml* per compatibilità con versioni obsolete.

È possibile suddividere i parametri di configurazione in più file ed importarli nel file principale mediante l'istruzione **include**.

Definizione di un Sevizio

I servizi sono definiti attraverso l'istruzione **services**.

Ogni sezione services può includere una sottosezione **build** in cui sono specificati i parametri per creare l'immagine.

I servizi che devono essere avviati prima di quello principale sono individuati attraverso l'opzione **depends_on**.

Impostare un Contenitore / Servizio

```
services:
   app:
      image: node:18-alpine
      command: sh -c "yarn install && yarn run dev"
```

Impostare i Parametri di Rete

```
services:
   app:
      image: node:18-alpine
      command: sh -c "yarn install && yarn run dev"
      ports:
         - 127.0.0.1:3000:3000
```

Impostare le Cartelle ed i Volumi

```
services:
   app:
      image: node:18-alpine
      command: sh -c "yarn install && yarn run dev"
      ports:
         - 127.0.0.1:3000:3000
      working_dir: /app
      volumes:
         - ./:/app
```

Impostare le Variabili di Ambiente

```
services:
   app:
      image: node:18-alpine
      command: sh -c "yarn install && yarn run dev"
      ports:
         - 127.0.0.1:3000:3000
      working_dir: /app
      volumes:
         - ./:/app
      environment:
         MYSQL_HOST: mysql
         MYSQL_USER: root
         MYSQL_PASSWORD: secret
         MYSQL_DB: todos
```

Aggiungere un Secondo Servizio

```
services:
   app:
      ...

   mysql:
      image: mysql:8.0
      volumes:
         - todo-mysql-data:/var/lib/mysql
      environment:
         MYSQL_ROOT_PASSWORD: secret
         MYSQL_DATABASE: todos
```

Aggiungere la Definizione dei Volumi

```
services:
   app:
      ...

   mysql:
      image: mysql:8.0
      volumes:
         - todo-mysql-data:/var/lib/mysql
      environment:
         MYSQL_ROOT_PASSWORD: secret
         MYSQL_DATABASE: todos

volumes:
   todo-mysql-data:
```

Aggiungere le Dipendenze

```
services:
   app:
      ...
      depends_on:
         - mysql
         - redis

   mysql:
      image: mysql:8.0
      ...

   redis:
      image: redis
      ...
```

Parametri del DNS

```
services:
   app:
      ...
      hostname: “darkarm01”
      domainname: “studiosg.net”
      dns:
         - 8.8.8.8
         - 9.9.9.9
      dns_search:
         - dc1.example.com
         - dc2.example.com
      ...
```

Reti / Volumi

Docker Compose consente di definire intere reti virtuali ed aggiungere volumi (Dischi) dedicati ai contenitori.

Le reti sono configurate attraverso l'istruzione **networks** ed i suoi parametri di configurazione.

I volumi sono configurati attraverso l'istruzione **volumes** ed i suoi parametri di configurazione.

I dispositivi devono essere assegnati ai servizi.

Informazioni & Licenze