1 Il progetto completo

1.1 Docker compose

Verranno analizzate due varianti del docker compose, la prima relativa al progetto, la seconda relativa ad un ambiente reale.

1.1.1 Versione simulata

Questo è il docker compose usato nel progetto, in un ambiente reale una configurazione simile avrebbe poco senso poichè prevede l'esecuzione di molteplici webhost sulla stessa macchina fisica.

```
version: "3"

services:
  WebServer1:
    build: Dockerfiles/webserver/.
    image: webserver
    container_name: WS1
    networks:
      - Internal
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://localhost:514"
        tag: "WS1"
    restart: unless-stopped
  WebServer2:
    build: Dockerfiles/webserver/.
    image: webserver
    container_name: WS2
    networks:
      - Internal
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://localhost:514"
        tag: "WS2"
    restart: unless-stopped
  WebServer3:
    build: Dockerfiles/webserver/.
    image: webserver
    container_name: WS3
    networks:
      - Internal
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
```

```
      options:
        syslog-address: "tcp://localhost:514"
        tag: "WS3"
    restart: unless-stopped
  WebServer4:
    build: Dockerfiles/webserver/.
    image: webserver
    container_name: WS4
    networks:
      - Internal
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://localhost:514"
        tag: "WS4"
    restart: unless-stopped
  LoadBalancer:
    build: Dockerfiles/load-balancer/.
    image: loadbalancer
    container_name: LB
    networks:
      - Internal
      - External
    ports:
      - "80:80"
    depends_on:
      - WebServer1
      - WebServer2
      - WebServer3
      - WebServer4
      - Syslogserver
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://localhost:514"
        tag: "LB"
    restart: unless-stopped
  Syslogserver:
    build: Dockerfiles/rsyslog/.
    image: syslogserver
    container_name: Syslog
    volumes:
      - "[CARTELLA LOCALE LOG]:/var/log"
    ports:
      - 514:514
      - 514:514/udp
    cap_add:
      - SYSLOG
    restart: unless-stopped

networks:
```

```
  Internal:
    internal: true
  External:
```

Listing 1: Docker Compose Progetto

1.1.2 Analisi Docker Compose

Nel file sono presenti 6 servizi, di cui: 4 web server, 1 load balancer e 1 server log.
In questa configurazione l'unica porta da esporre verso l'esterno è la porta 80 indirizzata all'ip della macchina su cui girano tutti i servizi.
Verrà esposta anche la 514 ma NON è necessario esporla fuori dalla lan locale.
I dettagli di configurazione dei vari servizi verranno analizzati in seguito nelle relative sezioni, si noti però che tutti i servizi hanno delle caratteristiche comuni:

- *build*: Eventuale percorso al Dockerfile per costruire l'immagine
- *image*: Nome dell'immagine da usare, oppure nome dell'immagine costruita tramite build
- *container_name*: Nome univoco del container nel sistema
- *networks*: Lista di network a cui collegare il sistema
- *depends_on*: Lista di servizi da avviare prima del servizio in questione
- *logging*
  - *syslog-address*: Indirizzo del server di log, in questo caso *localhost*
  - *tag*: Nome del servizio con cui identificarsi nel log server
- *restart*: azione da intraprendere in caso di errore o arresto anomalo

1.1.3 Versione effettiva

Versione utile in un ambiente reale.
In questo caso il load balancer e i webserver vengono eseguiti su macchine diverse.

```
version: "3"

services:
  LoadBalancer:
    build: Dockerfiles/load-balancer/.
    image: loadbalancer
    container_name: LB
    ports:
      - "80:80"
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
```

```
      options:
        syslog-address: "tcp://localhost:514"
        tag: "LB"
    restart: unless-stopped
  Syslogserver:
    build: Dockerfiles/rsyslog/.
    image: syslogserver
    container_name: Syslog
    volumes:
      - "[CARTELLA LOCALE LOG]:/var/log"
    ports:
      - 514:514
      - 514:514/udp
    cap_add:
      - SYSLOG
    restart: unless-stopped
```

Listing 2: Docker Compose Reale Load Balancer

Nel primo file troviamo il load balancer e il server log.
La porta 80 del load balancer deve essere esposta al di fuori della lan locale, la 514 del server log deve essere esposta SOLO SE le macchine su cui gireranno i webserver saranno al di fuori della rete locale.
È inoltre importante ricordarsi di aggiornare gli ip nel config del load balancer per puntare agli ip delle macchine effettive su cui girano i webhost (**??**).

```
version: "3"

services:
  WebServer:
    build: Dockerfiles/webserver/.
    image: webserver
    container_name: WS
    depends_on:
      - Syslogserver
    logging:
      driver: syslog
      options:
        syslog-address: "tcp://[IP Syslogserver]:514"
        tag: "WS-N"
    restart: unless-stopped
```

Listing 3: Docker Compose Reale Load Balancer

Nel secondo file troviamo la configurazione di un webserver.
Questo dockerfile verrà distribuito a tutte le macchine che dovranno ospitare un webserver.
È necessario aggiornare l'indirizzo del server di log in *syslog-server* con l'indirizzo della macchina su cui gira il servizio Syslogserver, è inoltre necessario inserire un id univoco nella sezione *tag* del logging al fine di consentire al server log di distinguere tra le varie istanze del webserver.
Dato che non abbiamo più un network docker, è necessario esporre le porte 80 di tutti gli host alla rete interna, al fine di consentire al load balancer reindirizzare le richieste.

1.2 Come avviare un docker compose

Per avviare un docker compose è sufficiente, una volta avviato il daemon docker, posizionarsi nella cartella contenente il file docker compose ed eseguire il comando *docker compose up*, questo comando creerà e avvierà tutti i servizi descritti nel file.

È inoltre possibile avviare solo una parte dei servizi specificando dopo up il nome del servizio come definito nel docker compose.

Per terminare tutti i servizi si usa il comando *docker compose down*.

Per comandi più avanzati visitare la documentazione ufficiale docker compose cli.

1.3 Screenshot funzionamento

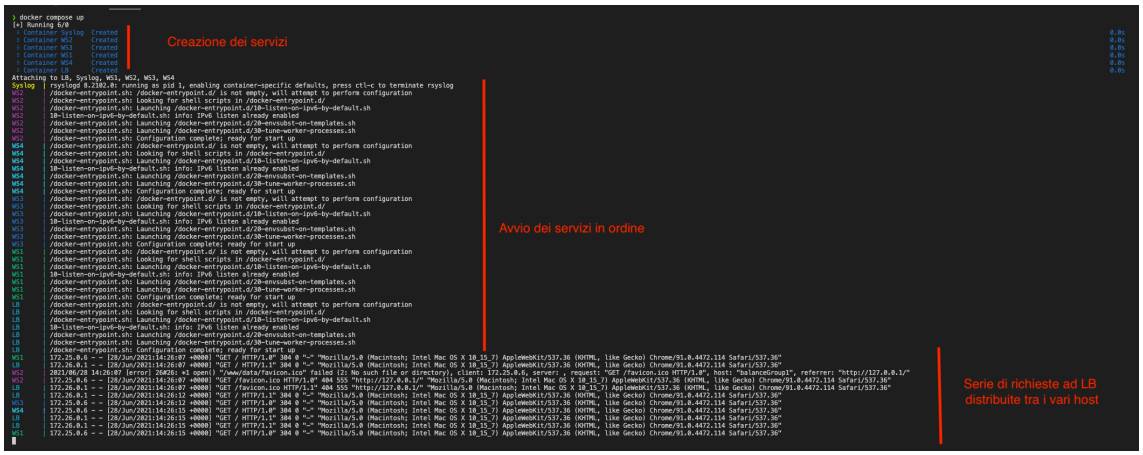

Figura 1: Log di avvio servizi e richieste distribuite

Come si vede dalla Figura 1, il primo servizio avviato è il server log, seguito dai 4 webserver in parallelo e, per ultimo, dal load balancer.
Una volta avviato, il load balancer risponde alle richieste dei client distribuendole tra i server definiti nel config, come visibile nelle ultime righe della Figura 1.

Nella Figura 2 si può notare il template del server di log in azione, i file vengono suddivisi secondo il template e, come previsto, ogni ora viene generato un nuovo file di log.

La Figura 3 mostra il contenuto di uno dei file di log (**??**).
Ogni riga contiene:

1. Un timestamp di ricezione del log
2. Indirizzo ip del mittente
3. Tag del servizio
4. Pid del servizio
5. Messaggio

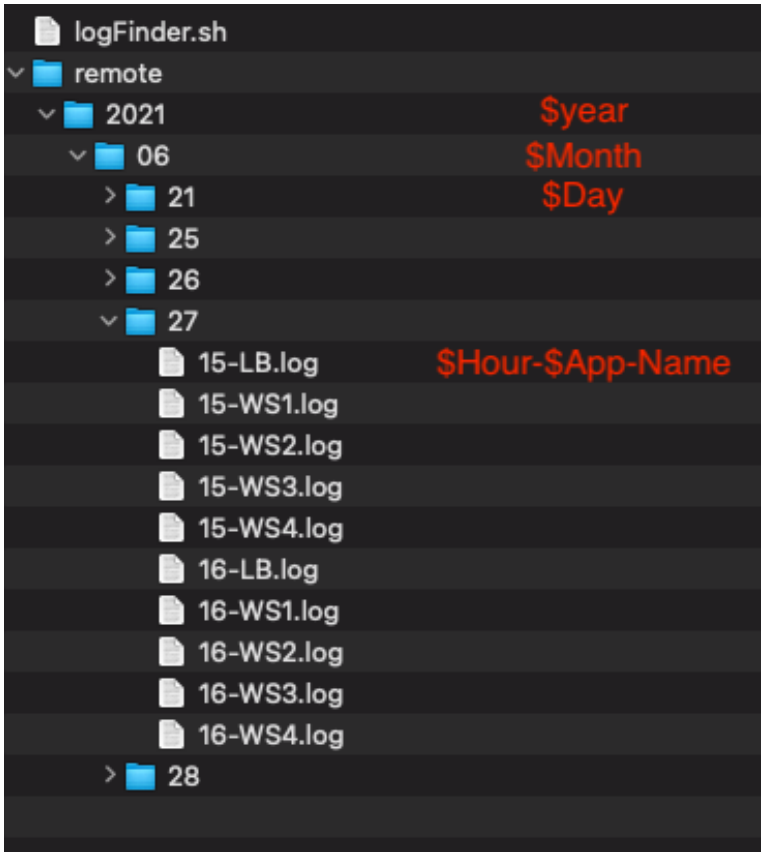

Figura 2: Cartella di salvataggio server di log (**??**)

Come previsto, i messaggi mostrati in Figura 3 corrispondono con i log del servizio LB visti in Figura 1.

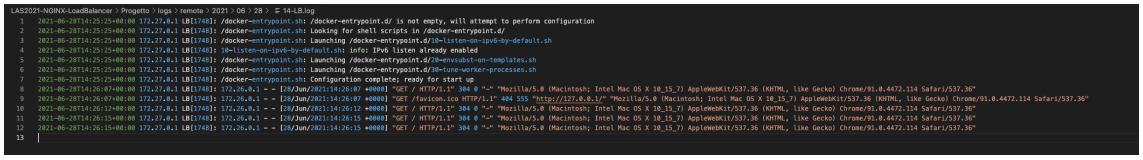

Figura 3: Contenuto del file di log *2021/06/28/14-LB.log*