1 I webserver NGINX

Pe l'implementazione del webserver vero e proprio useremo NGINX.
Partiremo da un immagine docker con NGINX pre installato e la integreremo nel nostro insieme di servizi docker.

Nella sezione **??** si è detto che vi saranno molteplici webserver tra cui dividere le richieste, nella realtà questi webserver si troverebbero su macchine differenti al fine di dividere il carico ma, dato che questo progetto è puramente dimostrativo, qui verranno implementate nello stesso host come istanze della stessa immagine docker (verranno comunque evidenziati i cambiamenti necessari per implementare la soluzione multi host).

1.1 Dockerfile

Di seguito il dockerfile per la creazione di un singolo webhost NGINX.

```
FROM nginx:1.20.0
RUN echo -e "\t\tCopying Config"
COPY Contents/nginx.conf /etc/nginx/nginx.conf
RUN echo -e "\t\tCopying WebServer"
COPY Contents/website /www/data
RUN echo -e "\t\Setting Permissions"
RUN chmod -R 555 /www/data/.
RUN chmod 555 /etc/nginx/nginx.conf
```

Listing 1: Dockerfile Webserver NGINX

1.1.1 Analisi Dockerfile

Scegliamo di utilizzare l'immagine *nginx* alla versione *1.21.0* presente nella repository di immagini ufficiale docker.
Si può facilmente notare che questa è una versione ufficiale poichè non è preceduta dal nome dell'utente che la gestisce (altrimenti sarebbe *utente/nginx*).
La prima modifica che facciamo all'immagine è caricare il nostro file *nginx.conf* nella cartella */etc/nginx/*. Andremo ad analizzare il file di config nella sottosezione 1.3.
Successivamente ci assicuriamo che il config e i file del webserver abbiano i giusti permessi, assegnando *rx* ad utente, gruppo e altri.

1.2 Servizio docker compose

Aggiungiamo molteplici istanze del webserver web a docker compose

```
WebServer1:
  build: Dockerfiles/webserver/.
  image: webserver
  container_name: WS1
  networks:
    - Internal
```

Listing 2: Webserver Docker Compose

La prima riga indica il nome univoco del servizio, ogni istanza del webserver deve avere il proprio nome univoco.
Riga 2 è opzionale e indica il percorso in cui effettuare la build dell'immagine se questa non è presente.
Riga 3 indica il nome dell'immagine. Se non è presente in locale verrà o presa dalla repo remota o buildata (se è presente l'istruzione build).
Riga 4 indica un nickname per il servizio.
Riga 6 impone alla macchina di collegarsi al network *Internal*, il quale non ha accesso alla rete esterna.
È possibile mappare la cartella */www/data* dell'immagine ad una cartella fisica, al fine di poter aggiornare i file del webserver senza dover ricostruire l'immagine completa. Per fare ciò basta aggiungere la voce

```
1     volumes:
2             - "[CARTELLA LOCALE WEBSERVER]:/www/data"
3
```

Per implementare un sistema multi host le righe 5 e 6 vanno sostituite con la porta su cui esporre il servizio nel formato *Porta fisica:Porta Virtuale*, dove:

- *Porta Fisica* è la porta dell'host su cui esporre il servizio
- *Porta Virtuale* è la porta del container a cui collegare la porta fisica

```
1 WebServer:
2   build: Dockerfiles/webserver/.
3   image: webserver
4   container_name: WS
5   ports:
6     - "80:80"
```

Listing 3: Webserver Docker Compose in multi host

Da notare che, dato che in questo caso i webserver vengono eseguiti su macchine diverse, non è più necessario distinguere i nomi dei servizi.

1.3 Configurazione

```
1  events {}
2  http {
3    server {
4      listen      80;
5      location /images/ {
6        root /www/static/images;
7      }
8      location / {
9        root /www/data;
10     }
11   }
```

```
12 }
```

Listing 4: File di configurazione webserver NGINX

Nella configurazione del webserver viene definita la porta su cui ascoltare e le posizioni in cui trovare i file, in questo caso *"/www/static/images"* per le pagine all'indirizzo *"/images"*, *"/www/data"* per tutte le altre.
Nuovi path per il webserver verrano aggiunti qui, rispettando le linee guida della documentazione NGINX.