Node.js: configurazione di nginx

nginx può essere configurato per gestire applicazioni in Node.js.

Proxy HTTP

Per impostazione predefinita Node.js non può operare sulla canonica porta 80 HTTP. Dobbiamo quindi utilizzare un proxy HTTP.

Se ad esempio abbiamo creato il dominio locale `http://app.test`, un proxy HTTP che gestisce questo dominio reindirizzerà tutte le richieste HTTP all'indirizzo interno su cui opera Node.js (ad esempio `http://127.0.0.1:3000`).

Con nginx possiamo creare un proxy HTTP come segue usando il seguente file di configurazione per il nostro dominio locale:

```
upstream app {
    server 127.0.0.1:3000;
}

server {
    listen 80;

    server_name app.test;

    root /home/app/www;
    index index.html;

    client_max_body_size 8m;

    location / {
```

```
    try_files $uri @app;
}

location @app {
    proxy_pass http://app;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
}
}
```

La regola `upstream` definisce il blocco relativo a Node.js che verrà poi salvato nella variabile `app`. In pratica ora tutte le richieste HTTP che arrivano al nostro dominio di test verranno reindirizzate internamente all'applicazione in Node.js.

Risorse statiche

Possiamo usare `nginx` per servire le risorse statiche di un'app in Node.js.

Supponiamo di avere questa struttura di directory:

```
– home
  – utente
    – app
      – assets
    – www
```

Se abbiamo usato `www` come document root per nginx, possiamo creare un link simbolico in questa directory che punti ad `assets`:

```
ln -s /home/utente/app/assets /home/utente/www/assets
```

Quindi ora avremo:

```
- home
  - utente
    - app
      - assets
    - www
      - assets(*)
```

In questo modo la gestione delle risorse statiche verrà delegata ad nginx.