Scale services with load balancing

https://github.com/knehez/isi - folder example_2

HAProxy is an open-source software that provides High Availability services, load balancing, and proxying for **TCP** and **HTTP**-based applications. It is used to distribute incoming network traffic across multiple servers to improve performance, scalability, and reliability of applications. **HAProxy** acts as a reverse proxy, meaning that it receives requests from clients and forwards them to the appropriate server based on various criteria such as load balancing algorithms, server health checks, and session persistence.

docker-compose.yml

```
version: "3.3"
services:
  web:
    build: .
    ports:
      - "5000"
  redis:
    image: "redis:alpine"
  haproxy:
    image: "haproxytech/haproxy-alpine:2.4"
    volumes:
        - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg:ro
    depends_on:
        - web
    ports:
      - "80:80"
```

HAPROXY config:

```
global
  stats socket /var/run/api.sock user haproxy group haproxy mode 660 level
admin expose-fd listeners
  log stdout format raw local0 info

defaults
  mode http
  timeout client 10s
  timeout connect 5s
  timeout server 10s
  timeout http-request 10s
  log global

frontend stats
  bind *:8404
  stats enable
  stats uri /
  stats refresh 10s
```

```
frontend myfrontend
  bind 0.0.0.0:80
  mode http
  default_backend webservers

backend webservers
  server s1 web:5000 check
  server s2 web:5000 check
  server s3 web:5000 check
  server s4 web:5000 check
  server s5 web:5000 check
```

This is a sample HAProxy configuration that defines various global settings, default settings, and frontend/backend configurations. Here's an explanation of each section:

1. The global section defines global settings for the entire HAProxy configuration. In this configuration, it specifies the location of the HAProxy stats socket at /var/run/api.sock and sets the user and group ownership to haproxy. It also specifies the log format and location, where logs are sent to stdout and use the raw format with a logging level of info.
2. The defaults section defines default settings for all frontends and backends. In this configuration, it sets the mode to http, which means that it expects HTTP requests. It also sets various timeout values, including client, connect, server, and HTTP request timeouts. Additionally, it enables global logging.
3. The frontend stats section defines a frontend for the HAProxy stats interface. It binds to port 8404 and enables the stats interface. It sets the stats URI to / and refreshes the stats page every 10 seconds.
4. The frontend myfrontend section defines a frontend for incoming HTTP traffic. It binds to port 80 and specifies the mode as http. It also specifies the default backend webservers, which will receive traffic that does not match any other defined frontend.
5. The backend webservers section defines a backend that consists of five servers, all with the name web and listening on port 5000. The check option specifies that HAProxy should check the health of each server before forwarding traffic to it.

In summary, this configuration sets up HAProxy to listen on port 80 for incoming HTTP traffic and distribute it across five backend servers that are checked for health before traffic is forwarded. It also sets up a stats interface on port 8404 to monitor the HAProxy instance.