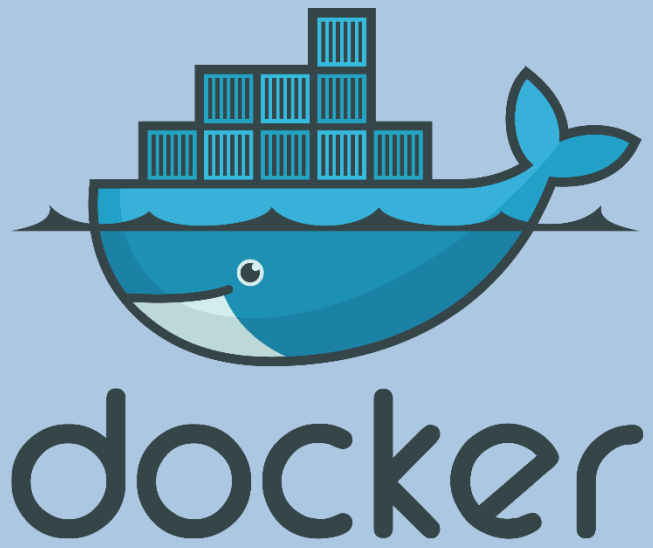

Docker Tutorials

Load Balancing Containers

Introduction❶

- In this session, we'll explore how you can use the NGINX web server to *load balance* requests between two containers running on the host.

- With Docker, there are two main ways for containers to communicate with each other. The first is via *links* which configure the container with environment variables and host entry allowing them to communicate. The second is using the *Service Discovery pattern* where uses information provided by third parties, in this scenario, it will be Docker's API.

Introduction❷

- The *Service Discovery pattern* is where the application uses a third party system to identify the location of the target service.

- For example, if our application wanted to talk to a database, it would first ask an API what the IP address of the database is. This pattern allows you to quickly reconfigure and scale your architectures with *improved fault tolerance than fixed locations.*

Step 1 - NGINX Proxy❶

- 我们希望运行一个NGINX服务，在加载新容器时可以动态发现和更新负载均衡配置
  - nginx-proxy

- *Nginx-proxy*接受HTTP请求，并根据请求头的Hostname代理到适当的容器
  - 对用户来说是透明的，不会有额外的性能开销

Step 1 - NGINX Proxy❷

Properties

a. Binding the container to port 80 on the host using -p 80:80.
   - This ensures all HTTP requests are handled by the proxy.

b. Mount the docker.sock file. This is a connection to the Docker daemon running on the host and allows containers to access it's metadata via the API.
   - Nginx-proxy uses this to listen for events and then updates the NGINX configuration based on the container IP address.
   - *-v /var/run/docker.sock:/tmp/docker.sock:ro*

Step 1 - NGINX Proxy❸

Properties

c. Set an optional *-e DEFAULTHOST=<domain>*. If a request comes in and doesn't make any specified hosts, then this is the container where the request will be handled. This enables you to run multiple websites with different domains on a single machine with a fall-back to a known website.

docker run -d -p 80:80 -e DEFAULT_HOST=proxy.example -v /var/run/docker.sock:/tmp/docker.sock:ro --name nginx jwilder/nginx-proxy

Step 2 - Single Host❶

Nginx-proxy is now listening to events which Docker raises on start / stop. A sample website called *katacoda/docker-http-server* has been created which returns the machine name it's running on. This allows us to test that our proxy is working as expected. Internally it's a PHP and Apache2 application listening on port 80.

Step 2 - Single Host❷

Starting Container

- For Nginx-proxy to start sending requests to a container you need to specify the VIRTUALHOST environment variable. This variable defines the domain where requests will come from and should be handled by the container.
  - *docker run -d -p 80 -e VIRTUAL_HOST=proxy.example katacoda/docker-http-server*
  - *curl http://docker*
  - ```
    $ curl http://docker
    <h1>This request was processed by host: a9492b45dc6e</h1>
    ```
    ```

Step 3 - Cluster

We now have successfully created a container to handle our HTTP requests. If we launch a second container with the same VIRTUAL_HOST then nginx-proxy will configure the system in a *round-robin load balanced* scenario. This means that the first request will go to one container, the second request to a second container and then repeat in a circle. There is no limit to the number of nodes you can have running.

- *docker run -d -p 80 -e VIRTUAL_HOST=proxy.example katacoda/docker-http-server*
- *curl http://docker*
- ```
  $ curl http://docker
  <h1>This request was processed by host: a9492b45dc6e</h1>
  ```
  ```

Step 4 - Generated NGINX Configuration

Whil
NGI
confi
confi

- C
- C

```
$ docker logs nginx
WARNING: /etc/nginx/dhparam/dhparam.pem was not found. A pre-generated dhparam.pem will be used fo
r now while a new one
is being generated in the background.  Once the new dhparam.pem is in place, nginx will be reloade
d.
forego     | starting dockergen.1 on port 5000
forego     | starting nginx.1 on port 5100
dockergen.1 | 2017/11/20 07:50:30 Generated '/etc/nginx/conf.d/default.conf' from 1 containers
dockergen.1 | 2017/11/20 07:50:30 Running 'nginx -s reload'
dockergen.1 | 2017/11/20 07:50:30 Watching docker events
dockergen.1 | 2017/11/20 07:50:30 Contents of /etc/nginx/conf.d/default.conf did not change. Skipp
ing notification 'nginx -s reload'
2017/11/20 07:52:01 [notice] 39#39: signal process started
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
dhparam generation complete, reloading nginx
dockergen.1 | 2017/11/20 07:54:54 Received event start for container a9492b45dc6e
dockergen.1 | 2017/11/20 07:54:54 Generated '/etc/nginx/conf.d/default.conf' from 2 containers
dockergen.1 | 2017/11/20 07:54:54 Running 'nginx -s reload'
nginx.1    | docker 172.17.0.88 - - [20/Nov/2017:07:55:00 +0000] "GET / HTTP/1.1" 200 58 "-" "curl
/7.35.0"
nginx.1    | docker 172.17.0.88 - - [20/Nov/2017:07:55:24 +0000] "GET / HTTP/1.1" 200 58 "-" "curl
/7.35.0"
nginx.1    | docker 172.17.0.88 - - [20/Nov/2017:07:57:34 +0000] "GET / HTTP/1.1" 200 58 "-" "curl
/7.35.0"
nginx.1    | docker 172.17.0.88 - - [20/Nov/2017:07:57:36 +0000] "GET / HTTP/1.1" 200 58 "-" "curl
```

engheng

Reference

- https://www.katacoda.com/courses/docker/10