Load balance a Swarm Service behind HA Proxy with automatic reconfiguration

29 November 2016

In this blog post, I'll be using Victor Farcic (Cloudbees senior employee, Docker captain, written books about Devops) Docker flow proxy (https://github.com/vfarcic/docker-flow-proxy/blob/master/articles/swarm-mode-listener.md) and docker flow swarm listener (https://github.com/vfarcic/docker-flow-swarm-listener) to reconfigure an HAproxy based on a Swarm service!

If you want to see how to do the same with Traefik, check this post (/traefik-as-a-dynamic-reverse-proxy-for-docker-swarm/)!

These two services work hand in hand. Docker Flow Listener will listen to the Swarm events over Docker's remote API. It will inform Docker flow proxy so it reconfigure the HAPRoxy frontend and backend.

Enough talk. Let's go.

Admin/Viz - if you need to look into what is happening

Let's start by installing Portainer and Docker Swarm Visualizer to be able to follow along on what is happening on our cluster. To do that:

```
docker service create \
  --publish 9008:9000 \
  --limit-cpu 0.5 \
  --name portainer-swarm \
  --constraint=node.role==manager \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  portainer/portainer --swarm

docker service create \
  --publish 9009:9000 \
  --limit-cpu 0.5 \
  --name portainer \
  --mode global \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  portainer/portainer

docker service create \
  --publish=9090:8080 \
  --limit-cpu 0.5 \
  --name=viz \
  --env PORT=9090 \
  --constraint=node.role==manager \
  --mount=type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
  manomarks/visualizer
```

Next Open - http://stratus-clay:9090 (http://stratus-clay:9090) and http://stratus-clay:9008 (http://stratus-clay:9008), and http://stratus-clay:9009 (http://stratus-clay:9009) (and on each of the swarm nodes if needed)

Utility function to wait for a service... (used below for demo purposes)

```
function wait_for_service()
 if [ $# -ne 1 ]
  then
    echo usage $FUNCNAME "service";
    echo e.g: $FUNCNAME docker-proxy
  else
serviceName=$1

while true; do
    REPLICAS=$(docker service ls | grep -E "(^| )$serviceName( |$)" | awk '{print $3}'
    if [[ $REPLICAS == "1/1" ]]; then
        break
    else
        echo "Waiting for the $serviceName service... ($REPLICAS)"
        sleep 5
    fi
done
fi
```

Alternatively you can run watch in another shell and visually check that the service is ready

```
watch -d -n 2 docker service ls
```

Create Networks

In order to isolate our application from the outside, we will configure two networks. One that will include the services which are accessible from the outside (read HAProxy), and another which will have our microservices Stack deployed on it. The networks will be

overlay networks, meaning they will span the whole cluster.

```
docker network create --driver overlay proxy
docker network create --driver overlay go-demo
```

Create Docker Listener

In order to create the listener service, add it to the network proxy, enable it to monitor the swarm (mount the docker.sock and constraint it to managers nodes), and link it to the docker flow proxy (which we will be creating next) we need to issue the below command:

```
docker service create --name swarm-listener --network proxy --mount "type=bind,source=/

wait_for_service swarm-listener
```

Create HAProxy with automatic reconfiguration

Let's create the Docker Flow proxy now. It will be listening on the port 80 and 443 (secure). It also need to link to the previous swarm-listener service, be active in swarm mode

```
docker service create --name docker-proxy -p 80:80 -p 443:443 -p 8080:8080 --network p

wait_for_service docker-proxy
```

Alternatively, if you need to customize the configuration or the image, you can rebuild it and use your own image instead (like the below)

```
docker service create --name docker-proxy -p 80:80 -p 443:443 -p 8080:8080 --network p
```

Prepare demo services

Here we can deploy our "complicated" stack offering. Could be a Mongo DB with a REST service based on it or the hello from "hostname" we've built in previous posts... but we don't really care... you can go with one or the other, or both...

```
docker service create --name go-demo-db --network go-demo mongo
wait_for_service go-demo-db

docker service create --name go-demo -e DB=go-demo-db --network go-demo --network proxy
wait_for_service go-demo

docker service create --name hello-svc --network go-demo --network proxy --label com.df
wait_for_service hello-svc
```

The labels com.df.notify=true, com.df.distribute=true, com.df.servicePath=/jmkhael and com.df.port=5000 are needed to inform Docker Flow Proxy how to reconfigure the service we are exposing. They will get into the HAProxy config as per the below:

```
acl url_hello-svc path_beg /jmkhael
use_backend hello-svc-be if url_hello-svc

backend hello-svc-be
    mode http
    server hello-svc hello-svc:5000
```

As you can see, the HAProxy will use the Swarm load balancing on the service name we used. Given that the Docker Flow Proxy is on both networks (the internal one too), it will be able to communicate with our service!

Test

Let's test that that work.

Check demo-service respond hello, world!

```
curl http://stratus-clay/demo/hello
time for i in {1..50} ; do curl http://stratus-clay/demo/hello; done
```

Check the hello-svc responds with "Hello from hostname"

```
for i in {1..5} ; do curl http://stratus-clay/jmkhael (http://stratus-clay/jmkhael) ; (
```

Check HAProxy config

```
curl http://`hostname`:8080/v1/docker-flow-proxy/config (http://`hostname`:8080/v1/docl
```

Benchmark it

```
ab -r -c 10000 -n 100000 http://stratus-clay/jmkhael (http://stratus-clay/jmkhael) | g
```

Reconfigure HAProxy

```
curl "http://`hostname`:8080/v1/docker-flow-proxy/reconfigure?serviceName=go-demo&servi
```

Cleanup Services and networks

```
docker service rm swarm-listener go-demo-db docker-proxy go-demo portainer portainer-sv
docker network rm proxy go-demo
```

HA Proxy breaking under load

> The below is mainly due to my ignorance about this subject. I am still learning. If you have any insights, feel free to leave a comment.

When trying to benchmark my service with Apache Benchmark to see how it behaves under load, I was faced with the dreaded `apr_socket_recv: Connection reset by peer (104)`

```
ab -c 50 -n 100000 http://stratus-clay/demo/hello (http://stratus-clay/demo/hello)
```

I though maybe the problem is HAProxy dropping connections, and I found that there is no default logging active. I opened this issue (https://github.com/vfarcic/docker-flow-proxy/issues/66)

Then this post (http://blog.scene.ro/posts/apache-benchmark-apr_socket_recv/) lead me to investigate a `Possible SYN flooding on port 8080. Sending cookies. Check SNMP counters.`

Debugging and checking logs

Check docker Deamon logs

```
sudo journalctl -fu docker.service
```

Check dmesg

```
dmesg | grep SYN
```

I've also tried to fiddle with Kernel configuration as per the slides from HA proxy tech: http://www.slideshare.net/haproxytech/haproxy-best-practice (http://www.slideshare.net/haproxytech/haproxy-best-practice) (slide 13)

```
sysctl -w net.ipv4.tcp_max_syn_backlog=100000
sysctl -w net.core.netdev_max_backlog=100000
sysctl -w net.core.somaxconn=65534
sysctl -w net.ipv4.tcp_syncookies=0
```

Check them using:

```
sysctl -a | grep "net.ipv4.tcp_max_syn_backlog"
sysctl -a | grep "net.core.netdev_max_backlog"
sysctl -a | grep "net.core.somaxconn"
```

but this didn't solve my issue. I've abandoned going further this route for now. if you have any insights to make things clearer for me, please drop me a comment :)

Tags: docker (/tag/docker/), docker-flow-proxy (/tag/docker-flow-proxy/), haproxy (/tag/haproxy/), dynamic-reconfiguration (/tag/dynamic-reconfiguration/)

Share this post:

(https://twitter.com/share?text=Load%20balance%20a%20Swarm%20Service%20behind%20HA%20balance-a-swarm-service-behind-ha-proxy-with-automatic-reconfiguration/) (https://www.facebook.com/sharer/sharer.php?u=https://jmkhael.io/load-balance-a-swarm-service-behind-ha-proxy-with-automatic-reconfiguration/) (https://plus.google.com/share?url=https://jmkhael.io/load-balance-a-swarm-service-behind-ha-proxy-with-automatic-reconfiguration/)

« PREVIOUS (/OVER-ENGINEERED-HELLO-WORLD-SERVICE-2/)
Scalable, and highly available (and Over engineered) "Hello, world" stack (/over-engineered-hello-world-service-2/)

NEXT » (/COLLECT-ALL-LOGS-FROM-A-DOCKER-SWARM-CLUSTER/)
Collect all Logs From a Docker Swarm Cluster (/collect-all-logs-from-a-docker-swarm-cluster/)

Author

JOHNNY MKHAEL (/AUTHOR/JOHNNY-MKHAEL/)

http://jmkhael.io/about-me (http://jmkhael.io/about-me)

Hello, world ;) Check the detailed 'About me' from the drop down menu. Or follow http://jmkhael.io/about-me (http://jmkhael.io/about-me).

Comments