

Pilvipalvelut

Contriboat Labra2

Ryhmä 3

Alexander Andreev

Rami Ojala

Ilari Rajala

Asko Ropponen

Laboratorioraportti

TTTW0430 - Pilvipalvelut, Jarmo Viinikanoja

18.11.2019

Tieto- ja viestintätekniikan tutkinto-ohjelma

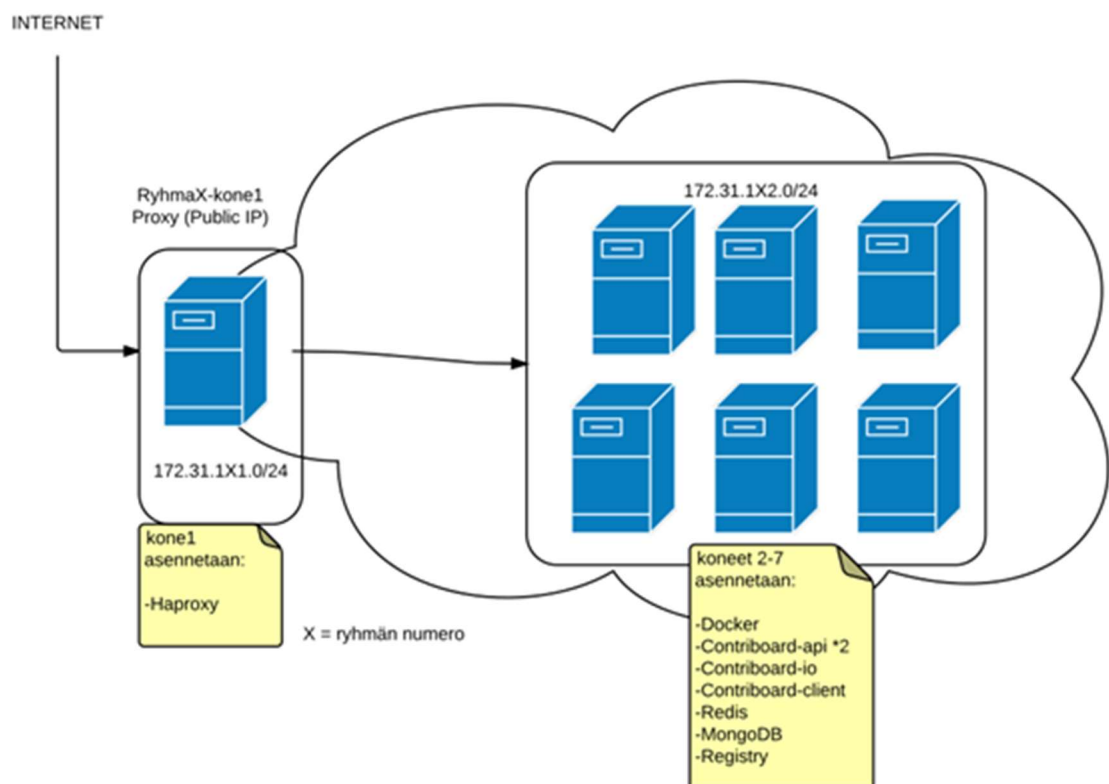
Sisältö

1	Johdanto	1
2	Tehtävänanto	1
3	Teoreettiset lähtökohdat	2
3.1	AWS	2
3.2	Docker	3
3.3	HAProxy	3
4	Työn kulku.....	4
4.1	AWS	4
4.2	Contriboat.....	4
4.2.1	Oman Docker Registryn luominen	4
4.2.2	Varmenteiden jakaminen.....	7
4.2.3	Konttien käynnistys	8
4.3	Kuormantasaas	12
4.3.1	Elastic Load Balancer	12
4.3.2	HAProxy	14
5	Pohdinta	16
	Lähteet	18

1 Johdanto

Pilvipalvelut laboratorio kaksi työnä tehtiin Contriboard palvelun pystytys Amazon Web Services pilvipalveluun siten, että sovelluksen komponentit jaettiin eri koneille. Laboratoriotyössä käytettiin laboratoriotyön ohjeistusta ja opettaja Viinikanojan asiantuntevia vinkkejä. Työssä käytettiin myös Elastic Load Balancing AWS järjestelmää.

2 Tehtävänanto



Kuvio 1 Tehtävänannon mukainen ympäristö.

- Asenna yhdelle koneelle Docker Registry ja laita Contriboardin komponentit sille niin, että kun niitä asennetaan Dockerilla imaget haetaan omasta Docker registrystä.
- Asenna Contriboard toimimaan neljällä koneella niin, että jokaisella koneella on yksi komponentti paitsi Redis ja Mongo, jotka ovat samalla koneella.
- Lisää kaksi API-konetta, toinen samalle koneelle, kuin missä on jo API ja toinen eri koneelle ja jaa liikenne näille API-koneille käyttäen ELB:tä.

- Muuta HAProxyn asetuksia niin, että kuorma tasataan API-koneiden kesken ja todenna että kuormantasaus toimii. Tasaa liikenne myös sisäverkossa API-koneille.
- Tietoturva: Muuttakaa security grouppien asetuksia niin, että vain tarvittava liikenne sallitaan vain teidän ryhmänne subnetistä + proxyille tietenkin täytyy sallia jostain kaikkialta. Sulkekaa myös ylimääräiset portit.
- Lisätkää ryhmänne avain HAProxy koneelle niin, että saatte yhteyden muihin koneisiin antamalla käskyn `ssh *koneen-ip tai esim osan nimi (api, io, mongo jne.)*`.
- Tehkää kuormantasaus myös IO-komponentille, eli asentakaa jollekin koneelle toinen IO-kontti (vaikka samalle missä toinenkin on) ja tasatkaa kuorma IO-konttien kesken. Kiinnittäkää huomiota siihen, että IO:lle jutellaan käyttäen websockettia, joka on tilallinen yhteys.
- Todentakaa että, jos resetoitte HAProxyn, sen jälkeen, kun yksi käyttäjä muodostaa yhteyden palveluun, pitäisi kaikkien pyyntöjen päätyä yhdelle IO-koneelle, vaikka backendissä näkyy useampi IO toiminnassa. Jos tulee toiselta koneelta yhteys, pitäisi sitten sen päätyä toiselle koneelle.
- Suunnitelkaa kuinka nykyisellä kokoonpanolla voisimme varautua siihen, että jos Mongo kone hajoaa, tietokanta olisi muuallakin tallella.

3 Teoreettiset lähtökohdat

3.1 AWS

Contriboat palvelu pystytetään Amazon Web Services palveluun, johon opettaja Viinikanoja oli jo valmiiksi luonut tunnukset ja avaimet. Työssä käytetään Amazon EC2 (Elastic Compute Cloud), joka tarjoaa muutettavissa olevaa laskenta kapasiteettiä pilvipalvelussa. EC2 koneissa käytetään korkea taajuisia Intelin Xeon prosessoreita ja niissä on tasapainotettu laskenta muisti ja verkkoresurssit. Työssä käytettävissä t2.micro koneissa on 3,3GHz skaalautuva prosessori teho, muistia yksi gigatavu ja tallennus Amazon EBS. (Elastic Block Store, Amazon EC2 Instance IP Addressing).

Tietoturvallisuuden vuoksi vain kahdelle koneelle (ulkoverkko) on annettu mahdollisuus päästä suoraan internettiin julkisilla IP-osoitteilla. Näille koneille asennetaan HAProxy, joiden kautta muut private IP koneet (sisäverkko) keskustelevat toistensa kanssa ja ovat yhteydessä internettiin.

SSH yhteydellä ulkoverkon koneeseen saa ryhmälle valmiiksi määritetyllä PEM avaimella käyttäen PuTTY terminaali emulaattoria. Ulkokoneen kautta saadaan taas SSH yhteys muodostettua sisäverkon koneisiin käyttäen niiden private IP-osoitteita.

3.2 Docker

Contriboboard asennetaan käyttäen Docker kontteja. Docker on joukko PaaS (Platform as a Service) tuotteita ja ne käyttävät käyttöjärjestelmä tason virtualisointia ohjelmiston ja siihen tarvittavien kirjastojen ja riippuvuuksien toimittamiseen, joita sanotaan konteiksi. Niillä pystytään helposti jakamaan toiselle koneelle identtinen ohjelmisto ympäristöineen. (What is a Container).

3.3 HAProxy

Kuorman tasaamisessa laboratorio työssä käytetään HAProxya. HAProxy on TCP / http kuormantasaus- ja välityspalvelin, jonka avulla verkkopalvelin jakaa saapuvat pyynnöt useille päätepisteille. Tämä on hyödyllinen tapauksissa, joissa liian monet samanaikaiset yhteydet ylikyllästävät yhden palvelimen ominaisuuksia. Yhdelle palvelimelle yhteyden muodostamisen sijaan, joka käsittelee kaikkia pyyntöjä, client muodostaa yhteyden HAProxy-instanssiin, joka käyttää käänteistä välityspalvelinta lähettämään pyynnön yhdelle käytettävissä olevista päätepisteistä, kuormantasaus algoritmin perusteella (How to Use HAProxy for Load Balancing.)

4 Työn kulku

4.1 AWS

Toisen laboratoriotyön aikana, meillä on jo valmiina ensimmäisen laboratoriotyön aikana luotu ympäristö. Ainoa tehty muutos, hankittu Elastic IP, joten julkinen IP, pysyy samana koneiden uudelleenkäynnistyksen jälkeen.

4.2 Contriboatd

4.2.1 Oman Docker Registryn luominen

Jotta Docker registryä voitaisiin käyttää, sillä pitäisi olla tunnetun tahon myöntämä varmenne (en. Certificate). Testitapauksessa sitä voi käyttää myös ilman kyseistä varmennettä, joko pelkällä http-yhteydellä, tai itse allekirjoitetulla varmenteella (Test an insecure registry.)

Loimme siis oman varmenteen (Kuvio 2). Varmenteen luonnissa millään muulla ei tässä tapauksessa ole väliä, kuin että Common Name-kenttään (FQDN) laitetaan jokin järkevää. Tätä käytetään, jotta varmenne ei ole sidottu IP-osoitteeseen, joka voi muuttua.

```

ubuntu@ip-172-31-132-111:~$ mkdir -p certs
ubuntu@ip-172-31-132-111:~$ openssl req \
> -newkey rsa:4096 -nodes -sha256 -keyout certs/domain.key \
> -x509 -days 365 -out certs/domain.crt
Generating a 4096 bit RSA private key
.....
++++
writing new private key to 'certs/domain.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:dockerlabra2.org
Email Address []:
ubuntu@ip-172-31-132-111:~$ ls
certs  envit
ubuntu@ip-172-31-132-111:~$ ls certs/
domain.crt  domain.key

```

Kuvio 2 Varmenteen luonti.

Sitten käynnistimme rekisterin (Kuviot 3 ja 4). Registryä käynnistäessä annetaan ympäristömuuttujina äsken luodun varmenteen sijainti.

```

ubuntu@ip-172-31-132-111:~$ sudo docker run -d --restart=always --name registry -v `pwd`/certs:/certs -e REGISTRY_HTTP_ADDR=0.0.0.0:443 -e REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e REGISTRY_HTTP_TLS_KEY=/certs/domain.key -p 5000:443 registry:2
Unable to find image 'registry:2' locally
2: Pulling from library/registry
c87736221ed0: Pull complete
1cc8e0bb44df: Pull complete
54d33bcb37f5: Pull complete
e8afc091c171: Pull complete
b4541f6d3db6: Pull complete
Digest: sha256:8004747f1e8cd820a148fb7499d71a76d45ff66bac6a29129bdfbdc0154d146
Status: Downloaded newer image for registry:2
a8636160f5dd665ddb3ce10c3f757a7325724d10296c3bc23e6ef8abadede1ea

```

Kuvio 3 Docker Registryn käynnistys.

```

ubuntu@ip-172-31-132-111:~$ sudo docker ps -a

```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
a8636160f5dd	registry:2	"/entrypoint.sh /etc..."	54 seconds ago	Up 52 seconds	5000/tcp, 0.0.0.0:5000->443/tcp	registry

Kuvio 4 Docker Registry käynnissä.

Seuraavaksi lisäämme rekisteriin haluamamme paketit (Kuviot 5 ja 6).

```
ubuntu@ip-172-31-132-111:~$ sudo docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
redis                latest              01a52b3b5cd1       3 weeks ago        98.2MB
registry             2                  f32a97de94e1       7 months ago       25.8MB
mongo                3.2                fb885d89ea5c       11 months ago      300MB
n4sjamk/teamboard-io latest             cfa2b17ffd55       3 years ago        327MB
n4sjamk/teamboard-api latest             3be5ab65756a       3 years ago        489MB
n4sjamk/teamboard-client latest             f4762f2de240       3 years ago        844MB
ubuntu@ip-172-31-132-111:~$ sudo docker image tag redis localhost:5000/redis
ubuntu@ip-172-31-132-111:~$ sudo docker push localhost:5000/redis
The push refers to repository [localhost:5000/redis]
6f3fa587ec88: Pushed
68ffe58b3e94: Pushed
c7fcc133516e: Pushed
a0e3cc85530d: Pushed
06550adba4f9: Pushed
2db44bce66cd: Pushed
latest: digest: sha256:4332745b2116859f4b77ff9dbc57ad9f5327dfe5e65807e064bdd2fe7c3cbfef size: 1572
```

Kuvio 5 Docker-pakettien lisäys rekisteriin.

```
ubuntu@ip-172-31-132-111:~$ sudo docker image tag mongo:3.2 localhost:5000/mongo
ubuntu@ip-172-31-132-111:~$ sudo docker push localhost:5000/mongo
The push refers to repository [localhost:5000/mongo]
7eaf69109a22: Pushed
b436f480c034: Pushed
f6a5611931ed: Pushed
2bcf250f2488: Pushed
fcd5eec06559: Pushed
1f5a9fb2648f: Pushed
88ccb1166c8a: Pushed
ed81bb40beff: Pushed
75c203162075: Pushed
9d3049f87bb2: Pushed
837a2e6463ae: Pushed
latest: digest: sha256:3176c386ff76d928893a4be19d6b33a4ed64032266335d9dfa6e3103c152b719 size: 2614
ubuntu@ip-172-31-132-111:~$ sudo docker image tag n4sjamk/teamboard-io localhost:5000/teamboard-io
ubuntu@ip-172-31-132-111:~$ sudo docker push localhost:5000/teamboard-io
The push refers to repository [localhost:5000/teamboard-io]
5f70bf18a086: Pushed
ab45967bd1d0: Pushed
f93f4be07c8f: Pushed
4c0b80774d6b: Pushed
127bd5dfbd90: Pushed
2069b8a7f871: Pushed
f007f46cf986: Pushed
fd0f1d8a7fd0: Pushed
11ddcbabd68c: Pushed
0d81735d8272: Pushed
982549bd6b32: Pushed
8698b31c92d5: Pushed
latest: digest: sha256:aabc45677caa8e39300d08316cdf228db78a1ab2aebac595f5d18002743a4a2 size: 3449
ubuntu@ip-172-31-132-111:~$ sudo docker image tag n4sjamk/teamboard-api localhost:5000/teamboard-api
ubuntu@ip-172-31-132-111:~$ sudo docker push localhost:5000/teamboard-api
The push refers to repository [localhost:5000/teamboard-api]
5f70bf18a086: Mounted from teamboard-io
5b9bbc427137: Pushed
29a9377ac298: Pushed
dc634a91fe14: Pushed
c485ddbc9ce3: Pushed
aa235c9cc64f: Pushed
506e96b4fda4: Pushed
8062c38aeb3d: Pushed
5f7f0e5d6f5f: Pushed
0d81735d8272: Mounted from teamboard-io
982549bd6b32: Mounted from teamboard-io
8698b31c92d5: Mounted from teamboard-io
latest: digest: sha256:45f42e0ad815e9533b484708670424d2a922fd6d1f10002dfb41a9ca38dcdbda size: 3449
```

Kuvio 6 Docker-pakettien lisäys rekisteriin.


```

ubuntu@ip-172-31-132-111:~$ sudo docker image tag n4sjamk/teamboard-client localhost:5000/teamboard-client
ubuntu@ip-172-31-132-111:~$ sudo docker push localhost:5000/teamboard-client
The push refers to repository [localhost:5000/teamboard-client]
5f70bf18a086: Mounted from teamboard-api
4ab7f7a8d7e7: Pushed
b473dbc01d6d: Pushed
043dc2c58741: Pushed
1c2664eb086e: Pushed
3c5a1626c222: Pushed
68fe1fde9e2b: Pushed
e3d65303de04: Pushed
8dbef2777134: Pushed
bb746342aa81: Pushed
54a344295792: Pushed
ed31f4c1b6dd: Pushed
2386afb8f445: Pushed
0d81735d8272: Mounted from teamboard-api
982549bd6b32: Mounted from teamboard-api
8698b31c92d5: Mounted from teamboard-api
latest: digest: sha256:d9e46d6e679de407522f99c276bef04f076ecd749551b9b422a0aa56472e071b size: 5111

```

Kuvio 7 Docker-pakettien lisäys rekisteriin.

4.2.2 Varmenteiden jakaminen

Jotta rekisteriä voidaan käyttää, täytyy koneiden, jotka hakevat Docker-paketteja rekisteristä luottaa rekisteriin. Tämä tarkoittaa, että sen luomamme varmenne täytyy siirtää jokaiselle koneelle, joka tahtoo käyttää rekisteriä. Ensimmäiseksi siirsimme edellisessä kappaleessa luomamme domain.crt -tiedoston koneelle, jolla on pääsy muille ryhmämme koneille, koska se kone, jolla Docker Registry sijaitsee, ei ole koneiden väliseen SSH-yhteyteen vaadittavaa avainta. Sieltä lähetimme domain.crt tiedoston jokaiselle koneelle, joka sitä tarvitsi (Kuvio 9). Eli jokainen kone, joka ajaa Contriboard-komponenttia.

```

ubuntu@ip-172-31-131-239:~$ rsync -v -e "ssh -i $HOME/ryhma3.pem" ubuntu@172.31.132.111:/home/ubuntu/certs/domain.crt /home/ubuntu/certs/
created directory /home/ubuntu/certs
domain.crt

sent 43 bytes received 2,080 bytes 4,246.00 bytes/sec
total size is 1,992 speedup is 0.94
ubuntu@ip-172-31-131-239:~$ ls
certs ryhma3.pem
ubuntu@ip-172-31-131-239:~$ ls certs/
domain.crt

```

Kuvio 8 domain.crt siirtäminen.

Koska meillä ei ollut oikeuksia sijoittaa domain.crt-varmennetiedostoa oikeaan paikkaan suoraan rsync:illä ssh:n yli, jouduimme siirtämään sen ensin kotikansioon kohdekoneella. Sitten otimme manuaalisesti SSH yhteyden koneeseen, ja siirsimme tiedoston oikeaan paikkaan (Kuvio 9). Tässä käytämme varmenteen luonnissa määriteltyä Common Name-arvoa, ja tiedoston nimi on ca.crt, eikä domain.crt (meidän tapauksessa dockerlabra2.org).

HUOM! Kuvista poiketen, polun täytyy sisältää myös portti, mikä on määritelty registryn käynnistäessä, eli meidän tapauksessa tiedoston sijainti on `"/etc/docker/certs.d/dockerlabra2.org:5000/ca.crt"`.

```
ubuntu@ip-172-31-131-239:~$ rsync -v -e "ssh -i $HOME/ryhma3.pem" /home/ubuntu/certs/domain.crt ubuntu@172.31.132.9:/home/ubuntu/certs/
created directory /home/ubuntu/certs
domain.crt

sent 2,080 bytes  received 76 bytes  4,312.00 bytes/sec
total size is 1,992  speedup is 0.92
ubuntu@ip-172-31-131-239:~$ ssh -i ryhma3.pem ubuntu@172.31.132.9
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-1087-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

46 packages can be updated.
0 updates are security updates.

New release '18.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
ubuntu@ip-172-31-132-9:~$ ls
certs
ubuntu@ip-172-31-132-9:~$ ls certs/
domain.crt
ubuntu@ip-172-31-132-9:~$ sudo cp certs/domain.crt /etc/docker/certs.d/dockerlabra2.org/ca.crt
ubuntu@ip-172-31-132-9:~$ ls /etc/docker/certs.d/dockerlabra2.org/
ca.crt
```

Kuvio 9 domain.crt -tiedoston sijoittaminen.

Sitten muokkaamme `/etc/hosts` -tiedostoa. Lisäämme sinne rivin, joka ohjaa aiemmin annetun Common Name-arvon (`dockerlabra2.org`), tarkoittamaan sen koneen IP-osoitetta, jolla Docker Registry sijaitsee (Kuvio 10).

```
GNU nano 2.5.3                                     File: /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
172.31.132.111 dockerlabra2.org
```

Kuvio 10 `/etc/hosts`.

4.2.3 Konttien käynnistys

Konttien käynnistämien sujuu nyt samalla lailla kuin normaalistikin, mutta Docker käyttää nyt myös omaa rekisteriämme. Koska Contriboard on nyt hajautettu useammalle koneelle, täytyy myös tehdä vaadittavat ympäristömuuttuja tiedostot (`client.txt`, `api.txt` ja `io.txt`), niille koneille jotka ko. komponentteja ajavat. Myös HAProxy:n asetustiedostosta (Kuvio 11), määritellään API, Client ja IO koneiden osoitteet osoittamaan niille koneille missä komponentit sijaitsevat. Komponentit käynnistetään tietyssä järjestyksessä, suositeltavaa on käynnistää Redis ja Mongo ennen IO

ja API-kontteja. Client-kontin järjestyksellä ei ole väliä, jos asetukset ovat oikein. Kuvista poiketen kannattaisi myös käyttää dockerin komentorivi argumenttia ”—restart=always”, jotta kontti käynnistyy uudestaan aina koneen käynnistyessä.

```
backend API
balance roundrobin
    timeout server 30000
    timeout connect 4000
    server API1 172.31.132.85:9001 weight 1 maxconn 2048 check

backend WS
balance leastconn
    timeout server 30000
    timeout connect 4000
    server WebSocket1 172.31.132.30:9002 weight 1 maxconn 2048 check

backend CLIENT
    server client 172.31.132.9:80 weight 1 maxconn 2048 check
```

Kuvio 11 HAProxy asetukset.

Ensimmäiseksi Client komponentti. Tässä vaiheessa tarkistetaan, että HAProxy koneen julkinen IP-osoite on muuttumaton, ja sitten lisäämme sen client.txt-tiedostoon (Kuvio 12). Tämä tiedosto siis sijaitsee sillä koneella missä Contriboardin Client-komponentti ajetaan.

```
NODE_ENV=production
IO_URL=http://52.209.227.5
IO_PORT=80
API_URL=http://52.209.227.5/api
API_PORT=80
```

Kuvio 12 client.txt.

Sitten voimme käynnistää Clientin (Kuvio 13). Käynnistyskomennossa määritetään kontin kuvan nimeksi se paketti, joka on tallennettu, omaan rekisteriimme. Näin docker hakee sen sieltä.

```

mount@ip-172-31-132-9:~$ sudo docker run -d --env-file=/envit/client.txt -p 8080:80 --expose=80 --name contribboard-client dockerlabra2.org:5000/teamboard-client
Unable to find image 'dockerlabra2.org:5000/teamboard-client:latest' locally
latest: Pulling from teamboard-client
578b327bb460: Pull complete
958d1a44bab8: Pull complete
fbd5adebec66: Pull complete
4f4fb700ef54: Pull complete
8e56c43f2a44: Pull complete
8c5c5a9d0b4: Pull complete
95fb850f853d: Pull complete
b40d99c195a8: Pull complete
b4a8fb733a8e: Pull complete
a68c7a0f6298: Pull complete
f51ad55e68fd: Pull complete
8c455a1239f5: Pull complete
45d3b5a30be0: Pull complete
e0d813e3f092: Pull complete
077a0319f8a: Pull complete
d607d7ac276d: Pull complete
Digest: sha256:d9e4d6e079de407522f99c276bef04f076ecd749551b9b422a0aa56472e071b
Status: Downloaded newer image for dockerlabra2.org:5000/teamboard-client:latest
33f47a6806bf56fc537c2d3aa637927a9af9a97c1ae2075ead00b48b8a2c8d4b

```

Kuvio 13 Client -komponentin käynnistys.

Redis ja Mongo käynnistetään samalla koneella kuin Registry. Niiden sijainti ei ole muuttunut ensimmäisestä laboratoriotyöstä.

API:n käynnistäminen (Kuvio 15) sujuu samoin, mutta api.txt tiedostoa täytyy muuttaa, koska API sijaitsee nyt omalla koneellaan. API:n täytyy siis tietää MongoDB:n ja Redis-komponentin koneen osoite (Kuvio 14).

```

NODE_ENV=production
MONGODB_URL=mongodb://172.31.132.111:27017/CB
REDIS_HOST=172.31.132.111
REDIS_PORT=6379
PORT=9001
TOKEN_SECRET=keksijotain
NEW_RELIC_LICENSE_KEY=XXXXXXXXXXXXXXXXXXXX
NEW_RELIC_LOG=/home/teamboard/logs/new_relic_api.log
NEW_RELIC_ENABLED=false

```

Kuvio 14 api.txt.

```

mount@ip-172-31-132-85:~$ mount@ip-172-31-132-85:~$ sudo docker run -d --env-file=/envit/api.txt -p 9001:9001 --expose=9001 -v /tmp/home/teamboard/logs --name contribboard-api dockerlabra2.org:5000/teamboard-api
latest: Pulling from teamboard-api
578b327bb460: Pull complete
958d1a44bab8: Pull complete
fbd5adebec66: Pull complete
4f4fb700ef54: Pull complete
c223975180e8: Pull complete
8d6e22effaed: Pull complete
4bf362044ea8: Pull complete
1c2c6d431848: Pull complete
80993af6a380: Pull complete
09ed394d1e5: Pull complete
aa66533b3762: Pull complete
a72125dc0fa0: Pull complete
Digest: sha256:45f420a0811e533b4847086704242a922fd6d1f10002dffb1a9ca38dcdbda
Status: Downloaded newer image for dockerlabra2.org:5000/teamboard-api:latest
85ddc618b00dcaca30740ded5a378c7c46f1a9a750aff8f0561a8100880cde
mount@ip-172-31-132-85:~$ sudo docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
85ddc618b00d        dockerlabra2.org:5000/teamboard-api   "/bin/sh -c '/usr/bl..."   11 seconds ago      Up 10 seconds      0.0.0.0:9001->9001/tcp   contribboard-api

```

Kuvio 15 API -kontin käynnistys.

IO-kontin io.txt asetustiedostosta täytyy muuttaa, että REDIS_HOST-asetus osoittaa siihen koneeseen missä Redis-kontti pyörii. Sitten ajetaan taas samalla lailla kuin muutkin (Kuvio 16).

```

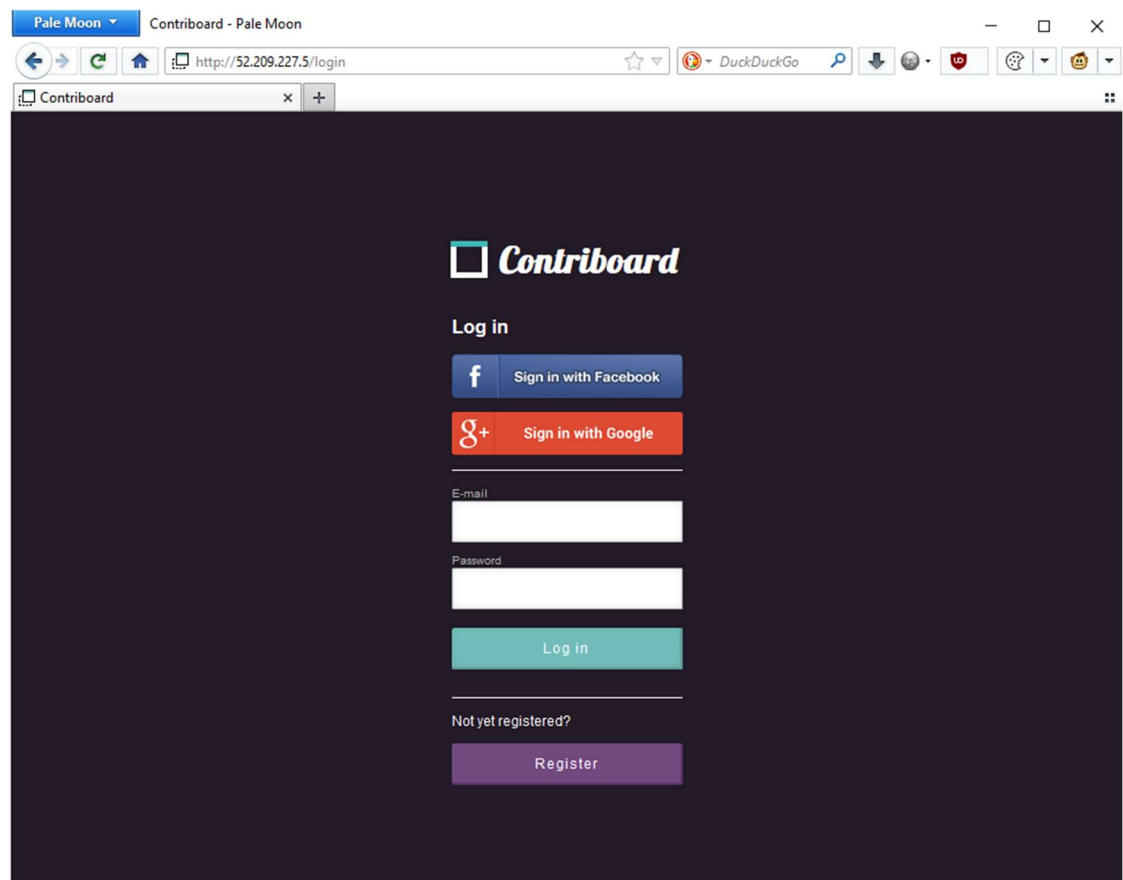
mount@ip-172-31-132-38:~$ sudo docker run -d --env-file=/env/it.io.txt -p 9002:9002 --expose=9002 -v /tmp/home/teamboard/logs --name contriboard-io dockerlabra2.org:5000/teamboard-io
Unable to find image 'dockerlabra2.org:5000/teamboard-io:latest' locally
latest: Pulling from teamboard-io
578b327bb460: Pull complete
958d1a44bab8: Pull complete
fbd5adebec66: Pull complete
4f4fb70bef54: Pull complete
76d36192313d: Pull complete
240a3b9bc60a: Pull complete
92671ac7f6e1: Pull complete
1b1f71c5c25b: Pull complete
57086d22b3c2: Pull complete
7201f144dea8: Pull complete
3863e3143806: Pull complete
8e9a5b33def4: Pull complete
Digest: sha256:aabc45677caa8e39300d08316cdf228db78a1ab2aebac595f5d18002743a4a2
Status: Downloaded newer image for dockerlabra2.org:5000/teamboard-io:latest
75f2f74f36bba277fbfad187b8f6694d2978071b163b4285b883d593617737ab
mount@ip-172-31-132-38:~$

```

Kuvio 16 IO-kontin ajaminen.

Sitten voidaankin avata se selaimessa, menemällä HAProxy koneen IP-osoitteeseen.

Jos kaikki on oikein, pitäisi näkyä kirjautumissivu (Kuvio 17).



Kuvio 17 Contriboard selaimessa.

4.3 Kuormantasaus

Jotta palvelu kestäisi monta käyttäjää, täytyy kuormaa hajauttaa usean koneen kesken. Liikenne näihin sitten tasataan joko Amazonin Elastic Load Balancerilla tai HAP-
roxyllä.

Hajautetaan API ja IO konttien kuormaa. Ajetaan niistä uudet instanssit. API-konttien määrää lisätään kahdella, ja IO-kontteja yhdellä. Kaksi API-konttia pyörii samalla palvelimella (Kuvio 18), ja yksi ajetaan eri palvelimella (Kuvio 19). IO-kontteja ajetaan kaksi samalla palvelimella (Kuvio 20).

```
ubuntu@ip-172-31-132-85:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
d265130ca959	dockerlabra2.org:5000/teamboard-api	"/bin/sh -c '/usr/bi..."	2 weeks ago	Up 37 minutes	0.0.0
.0:9002->9001/tcp	contriboard-api2				
05ddc618b00d	dockerlabra2.org:5000/teamboard-api	"/bin/sh -c '/usr/bi..."	3 weeks ago	Up 37 minutes	0.0.0
.0:9001->9001/tcp	contriboard-api				

```
ubuntu@ip-172-31-132-85:~$
```

Kuvio 18 Kaksi API-konttia samalla koneella

```
ubuntu@ip-172-31-132-119:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
d906e4d524a3	dockerlabra2.org:5000/teamboard-api	"/bin/sh -c '/usr/bi..."	2 weeks ago	Up 37 minutes	0.0.0
.0:9001->9001/tcp	contriboard-api				

```
ubuntu@ip-172-31-132-119:~$
```

Kuvio 19 Kolmas API-kontti eri koneella

```
ubuntu@ip-172-31-132-30:~$ sudo docker run -d --env-file=/envit/io.txt -p 9003:9002 --expose=9002 -v /tmp:/home/teamboard/logs -
```

```
name contriboard-io2 dockerlabra2.org:5000/teamboard-io
```

```
93d7ff8612f3c72321282e165137d90768b6a1981eb9b0d99c413e51ff8de7fb
```

```
ubuntu@ip-172-31-132-30:~$ sudo docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
93d7ff8612f3	dockerlabra2.org:5000/teamboard-io	"/bin/sh -c '/usr/bi..."	12 seconds ago	Up 11 seconds	0.0.0.
.0:9003->9002/tcp	contriboard-io2				
75f2f74f36bb	dockerlabra2.org:5000/teamboard-io	"/bin/sh -c '/usr/bi..."	2 weeks ago	Up 2 minutes	0.0.0.
.0:9002->9002/tcp	contriboard-io				

Kuvio 20 IO-konttien kahdennus

4.3.1 Elastic Load Balancer

Elastic Load Balancer eli ELB on Amazon Web Servicen tarjoama kuormantasaaja, joka jakaa tulevan liikenteen tasaisesti valituille EC2-instansseille. Palvelu otetaan käyttöön AWS käyttöliittymän kautta. Kokeilimme kuormantasausta tätä kautta vain API:lle.

Ensin annetaan nimi uudelle kuormantasaajalle (Kuvio 21).

Step 1: Define Load Balancer

Basic Configuration

This wizard will walk you through setting up a new load balancer. Begin by giving your new load balancer a unique name so that you can identify it from other load balancers you might create. You will also need to configure ports and protocols for your load balancer. Traffic from your clients can be routed from any load balancer port to any port on your EC2 instances. By default, we've configured your load balancer with a standard web server on port 80.

Load Balancer name:

Create LB inside:

Create an internal load balancer: ☐

Enable advanced VPC configuration: ☐

Listener Configuration:

Load Balancer Protocol	Load Balancer Port	Instance Protocol	Instance Port
HTTP	80	HTTP	8001

[Add](#)

[Cancel](#) [Next: Assign Security Groups](#)

Kuvio 21 ELB Askel 1.

Määritellään Security Groups (Kuvio 22).

Step 2: Assign Security Groups

Filter: VPC security groups

Security Group ID	Name	Description	Actions
sg-f6fe9a9c	default	default VPC security group	Copy to new
sg-01305994	GATEWAY	GATEWAY	Copy to new
sg-0d1d8975	launch-wizard-1	launch-wizard-1 created 2017-06-23T12:51:30.452+03:00	Copy to new
sg-0ed108d339e7734ca	launch-wizard-2	launch-wizard-2 created 2019-09-30T16:12:47.302+03:00	Copy to new
sg-0d1501adeb9072c2d	Ryhma13-ulkoverikko	launch-wizard-2 created 2019-09-30T16:07:50.355+03:00	Copy to new
sg-0985b4d5b71947fc1	Ryhma1-sisaverikko	launch-wizard-2 created 2019-09-30T15:52:07.482+03:00	Copy to new
sg-066503d012c1abeb	Ryhma1-ulkoverikko	launch-wizard-2 created 2019-09-30T15:35:53.896+03:00	Copy to new
sg-01e4f355e5d93cb65	Ryhma10-sisaverikko	launch-wizard-3 created 2019-10-01T11:55:04.028+03:00	Copy to new
sg-0146d8972313aff6e	Ryhma10-ulkoverikko	launch-wizard-3 created 2019-10-01T11:48:19.826+03:00	Copy to new
sg-06d5d551f41c1b0b	Ryhma11-sisaverikko	launch-wizard-3 created 2019-10-01T12:03:10.302+03:00	Copy to new
sg-0c3d7395e862a1f3	Ryhma11-ulkoverikko	launch-wizard-3 created 2019-10-01T12:00:17.353+03:00	Copy to new
sg-0442d6e16861010	Ryhma12-sisaverikko	launch-wizard-2 created 2019-09-30T15:52:00.525+03:00	Copy to new
sg-0c327dce40ff7c86	Ryhma12-ulkoverikko	launch-wizard-2 created 2019-09-30T15:35:44.780+03:00	Copy to new
sg-00328567c2b9cc1	Ryhma13-sisaverikko	launch-wizard-2 created 2019-09-30T16:08:50.355+03:00	Copy to new
sg-0020202cb0954d77ef	Ryhma14-sisaverikko	launch-wizard-2 created 2019-09-30T15:52:00.425+03:00	Copy to new
sg-0f38e0acc059eb6	Ryhma14-ulkoverikko	launch-wizard-2 created 2019-09-30T15:35:43.867+03:00	Copy to new
sg-0091597429137024	Ryhma15-sisaverikko	launch-wizard-3 created 2019-10-01T12:08:03.238+03:00	Copy to new
sg-06267b0b9822f2ad2	Ryhma15-ulkoverikko	launch-wizard-3 created 2019-10-01T12:05:02.484+03:00	Copy to new

[Cancel](#) [Previous](#) [Next: Configure Security Settings](#)

Kuvio 22 ELB Askel 2.

Märitellään Health Check (Kuvio 23).

Step 4: Configure Health Check

Your load balancer will automatically perform health checks on your EC2 instances and only route traffic to instances that pass the health check. If an instance fails the health check, it is automatically removed from the load balancer. Customize the health check to meet your specific needs.

Ping Protocol:

Ping Port:

Ping Path:

Advanced Details

Response Timeout: seconds

Interval: seconds

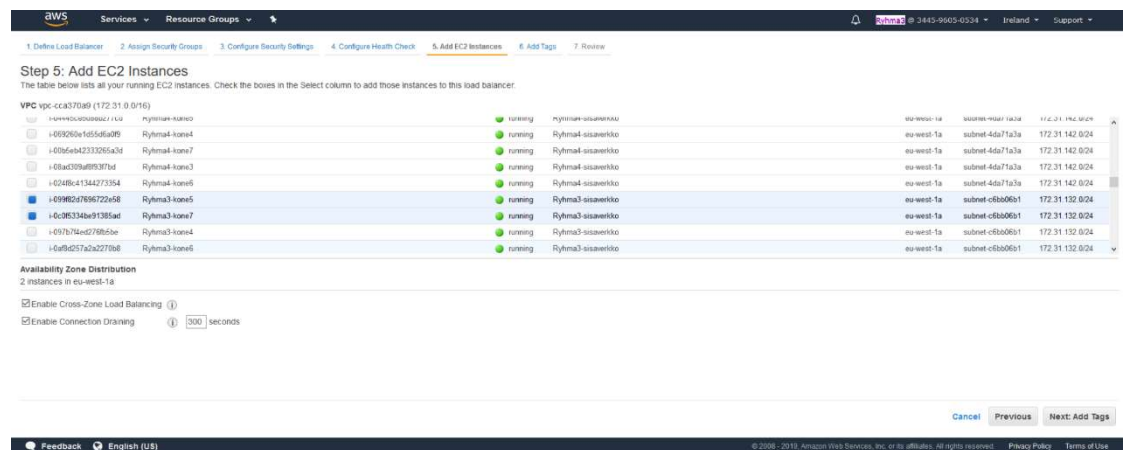
Unhealthy threshold:

Healthy threshold:

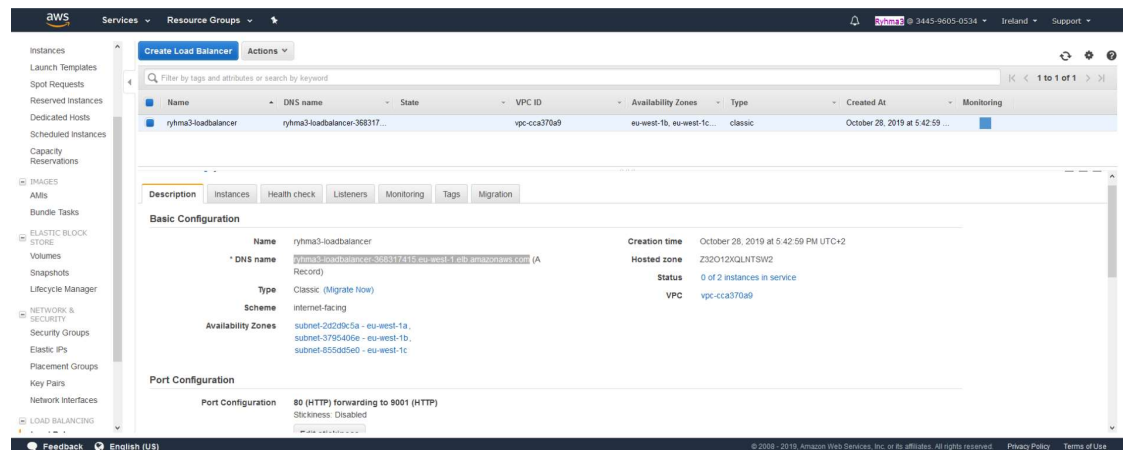
[Cancel](#) [Previous](#) [Next: Add EC2 Instances](#)

Kuvio 23 ELB Askel 4.

Lisätään instanssit kuormatasajaan (Kuvio 24).



Kuvio 24 ELB Askel 5.



Kuvio 25 ELB DNS name.

4.3.2 HAProxy

Ennen HAProxya käyttöönottoa otettiin ELB pois käytöstä. HAProxyn asetuksiin lisättiin API:n kuormantasaus, joka onnistui lisäämällä muutama rivi HAProxyn konfiguraatio tiedostoihin (Kuvio 26).

```
backend API
balance roundrobin
    timeout server 30000
    timeout connect 4000
    server API1 172.31.132.85:9001 weight 1 maxconn 2048 check
    server API2 172.31.132.85:9002 weight 1 maxconn 2048 check
    server API3 172.31.132.119:9001 weight 1 maxconn 2048 check
```

Kuvio 26 HAProxy config API kuormantasaus.

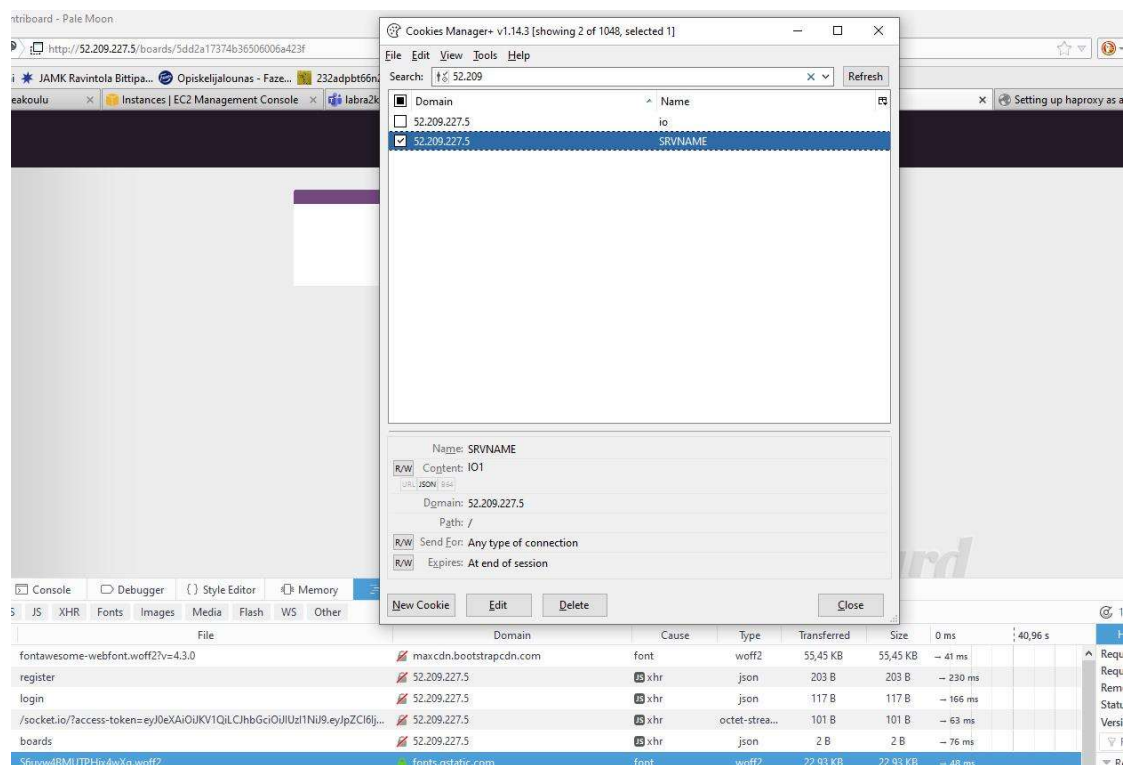
Nämä rivit lisättyä boottasimme HAProxyn ja kuormantasaus oli valmis API-konttien osalta. IO-konttien osalta kuormantasauksen täytyy aina ohjata saman clientin yhteys samalle koneelle (Sticky Load Balancing). HAProxyn asetustiedostossa asetetaan toinen palvelin ja niille asetus ”cookie <nimi>”. Myös lisätään asetus ”cookie SRVNAME insert”. Tämä antaa selaimelle keksin missä arvona sen serverin ”cookie <nimi>”. Sit-
ten kun selain ottaa HAProxy:n yhteyden, HAProxy osaa tämän keksin perusteella ohjata yhteyden oikeaan IO-palvelimeen (Kuviot 28 ja 29).

```

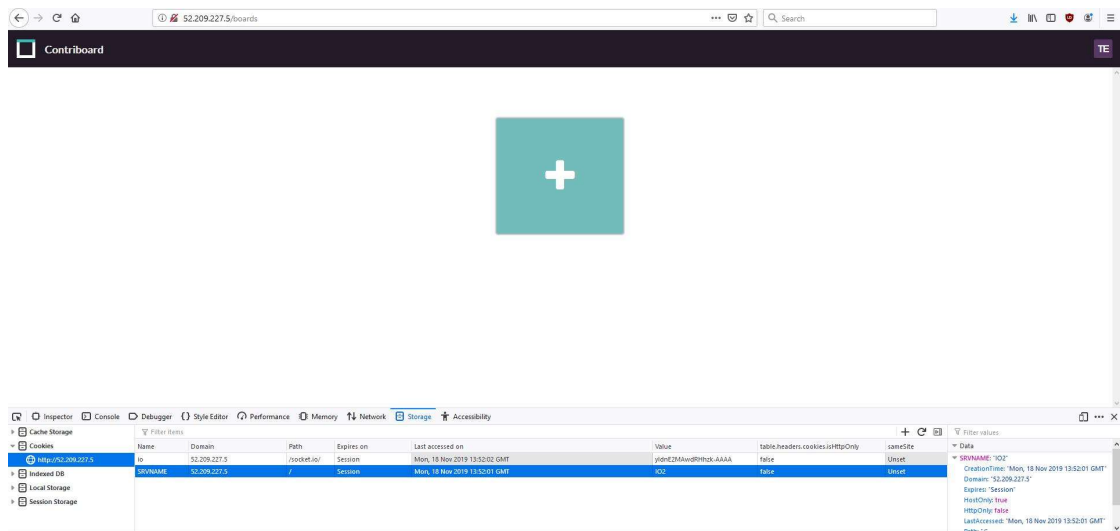
backend WS
balance leastconn
    timeout server 30000
    timeout connect 4000
    cookie SRVNAME insert
    server WebSocket1 172.31.132.30:9002 weight 1 maxconn 2048 cookie IO1 check
    server WebSocket2 172.31.132.30:9003 weight 1 maxconn 2048 cookie IO2 check

```

Kuvio 27 IO -konttien load balance.



Kuvio 28 Sticky Load Balance slaimessa.



Kuvio 29 Eri koneella menee eri IO-serverille.

5 Pohdinta

Työn teko meni sukkelaan, sillä kuivaharjoittelu muistiinpanot sekä labra yksi raportti auttoi palauttamaan asioita mieleen. Päätimme jo alussa, että Docker rekisteri tulee toimimaan Kone kolmosen päällä, sillä jokainen Docker container oli jo sillä koneella. Tällä tavoin oman rekisterin pystyttäminen tapahtui todella helposti ja jokainen image oli vain Tagia ja pushia vaille valmis.

Otimme rekisterin sertifikaatit kone kolmesta ja siirsimme ne rsync SSH:n avulla kone ykköselle, josta siirsimme ne eteenpäin sisäverkon muihin koneisiin, jonka avulla Docker pull komento saatiin toimimaan myös omaan rekisteriin. Ensimmäistä Docker run komentoa ajettaessa huomasimme, että emme saaneet yhteyttä Docker rekisteri koneeseen, jolloin ylistetty opettajamme Viinikanoja lopulta onnistui auttamaan meitä kertomalla sen, että sisäverkko tarvitsi uuden säännön, joka salli liikenteen Docker rekisteriin. Lisäsimme Amazon Security Group:iin säännön, joka salli sisäisen liikenteen kyseisessä verkossa.

Docker rekisterin toimimisen jälkeen, aloimme pystyttämään komponentteja eri Amazonin koneille. Tämä sujui suhteellisen kivuttomasti taas vanhoja muistiinpanoja välillä tarkastellen. Saimme koko järjestelmän lopulta suhteellisen nopeasti toimimaan, mutta ContriboBoardin reaaliaikaisuus oli hajalla, joka johtui siitä, että emme osien asetusten muutosten jäljiltä olleet koskaan käynnistäneet kontteja uudestaan.

Reaaliaikaisuus saatiin lopulta toimimaan oikein käynnistämällä IO ja API kontit uudesta.

HAProxy kuormantasausta API konttien kesken oli pystyssä jo ennen kuin Elastic Load Balancea edes katsottiin. Otimme kuitenkin HAProxy kuormantasauksen pois päältä, kun teimme ELB kuormantasausta. ELB käyttöön otossa huomasimme, että se on täysin turha ja joutava kapistus sekä, että HAProxy on äärettömästi parempi.

Lähteet

Amazon EC2 Instance IP Addressing. Artikkelin docs.aws.amazon.com sivustolla. Viitattu 28.10.2019. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#concepts-private-addresses>.

How to Use HAProxy for Load Balancing. HAProxy:n kuorman tasauksen käyttöohje linode.com sivustolla. Viitattu 11.11.2019. <https://www.linode.com/docs/up-time/loadbalancing/how-to-use-haproxy-for-load-balancing/>.

Test an insecure registry. Docker dokumentaatio. Viitattu 11.11.2019. <https://docs.docker.com/registry/insecure/#use-self-signed-certificates>.

What is a Container. Artikkelin docker.com sivustolla. Viitattu 28.10.2019. <https://www.docker.com/resources/what-container>.