

## Pilvipalvelut

### Contribo board asennus

#### Ryhmä 3

Alexander Andreev

Rami Ojala

Ilari Rajala

Asko Ropponen

Laboratorioraportti

TTTW0430 - Pilvipalvelut, Jarmo Viinikanoja

1.10.2019

Tieto- ja viestintätekniikan tutkinto-ohjelma

# Sisältö

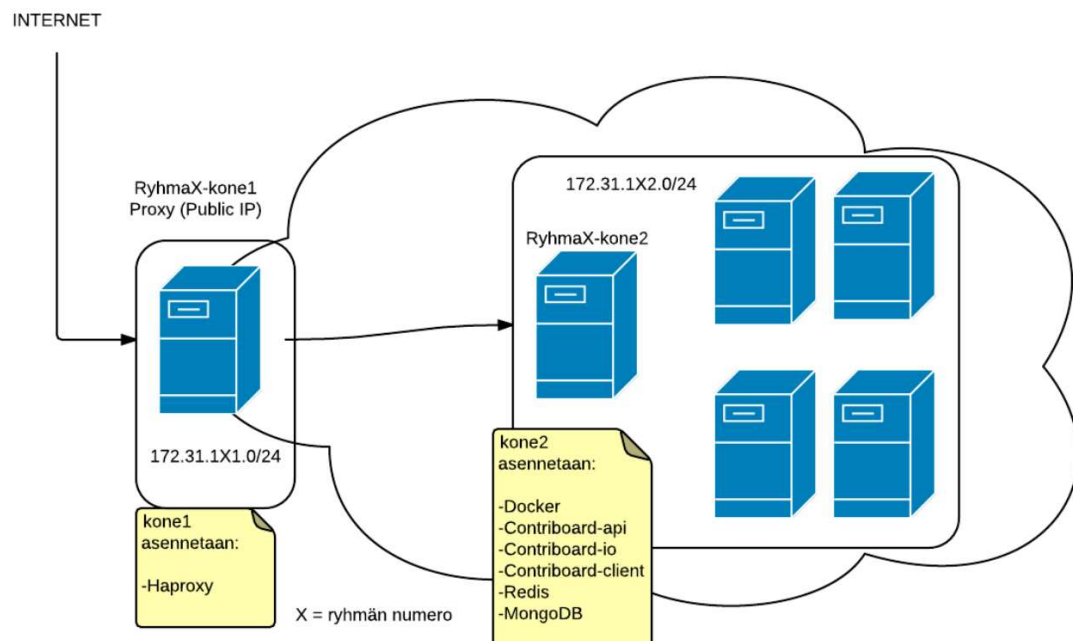
<b>1</b>	<b>Johdanto .....</b>	<b>1</b>
<b>2</b>	<b>Tehtävänanto .....</b>	<b>1</b>
<b>3</b>	<b>Teoreettiset lähtökohdat .....</b>	<b>2</b>
<b>4</b>	<b>Työn kulku.....</b>	<b>3</b>
4.1	AWS .....	3
4.2	HAProxy .....	5
4.3	Docker.....	6
4.4	Contriboat.....	7
4.4.1	Asennuksen valmistelu.....	7
4.4.2	Konttien käynnistys .....	9
<b>5</b>	<b>Pohdinta .....</b>	<b>Virhe. Kirjanmerkkiä ei ole määritetty.</b>
	<b>Lähteet .....</b>	<b>12</b>

# 1 Johdanto

Pilvipalvelut laboratorio yksi työnä tehtiin Contriboard palvelun pystytys Amazon Web Services pilvipalveluun, ja siinä käytettiin aiemmin kurssilla opetettuja menetelmiä ja laboratoriotyön ohjeistusta. Opiskelijat jaettiin neljän hengen ryhmiin, jotka ryhmänä suorittivat laboratoriotyön sekä tekivät tämän kyseisen raportin aiheesta. Raportissa kerrotaan ja kuvaillaan Contriboardin asennuksen vaiheita.

## 2 Tehtävänanto

- Asenna HAProxy kone1:seen ja säädetään se kuuntelemaan porttia 80 ja ohjaamaan liikennettä eri komponenttien välillä.
- Ota yhteys koneen yksi kautta sisäverkon koneeseen.
- Asenna kone3:seen Docker ja Contriboardin kontit.
- Tee enviroment variableja käyttäviä tiedostoja, joissa määritetään konteille tarvittavat ympäristön asetukset.
- Kokeile että ryhmän julkisesta IP-osoitteesta pääsee toimivaan Contriboard palveluun.



Kuva 1 Contriboard palvelun rakennekaavio.

### 3 Teoreettiset lähtökohdat

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 1 GiB ja tallennus Amazon EBS (Elastic Block Store) ([Amazon EC2 Instance IP Addressing](#)).

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

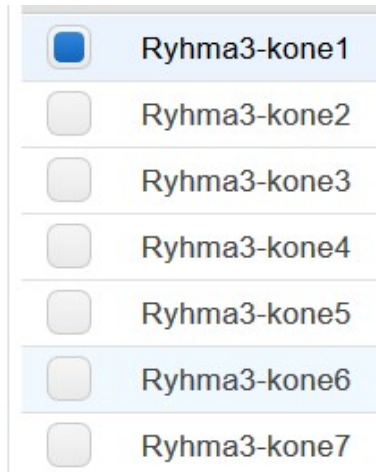
SSH yhteydellä ulkoverkon koneeseen saa ryhmälle valmiiksi määritetyllä PEM avaimella käyttäen PuTTY terminaali emulaattoria. Ulkokoneen kautta saadaan taas SSH yhteys muodostettua sisäverkon koneisiin käyttäen niiden private IP-osoitteita.

Contriboat 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).

## 4 Työn kulku

### 4.1 AWS

Ohjeiden mukaan luottiin seitsemän virtuaalikonetta, joilla oli kaksi security groupia.



Kuva 2. Lista ryhmän koneista Amazon Web Services palvelussa.



Kuva 3. Lista Ryhmän verkoista Amazon Web Services palvelussa.

Laboratorio työssä tarvittiin vain koneita Ryhmä3-kone1 (julkinen IP) ja Ryhmä3-kone3 (private IP).

Sisäverkon koneelle annettiin seuraavat Security Group säännöt (Kuva 4).

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ	Description ⓘ
All traffic	All	All	172.31.131.0/24	
SSH	TCP	22	0.0.0.0/0	

Kuva 4. Sisäverkon Security Group säännöt.

Ulkoverkon koneelle seuraavat säännöt (Kuva 5).

HTTP	TCP	80	0.0.0.0/0	
All traffic	All	All	172.31.131.0/24	
SSH	TCP	22	0.0.0.0/0	

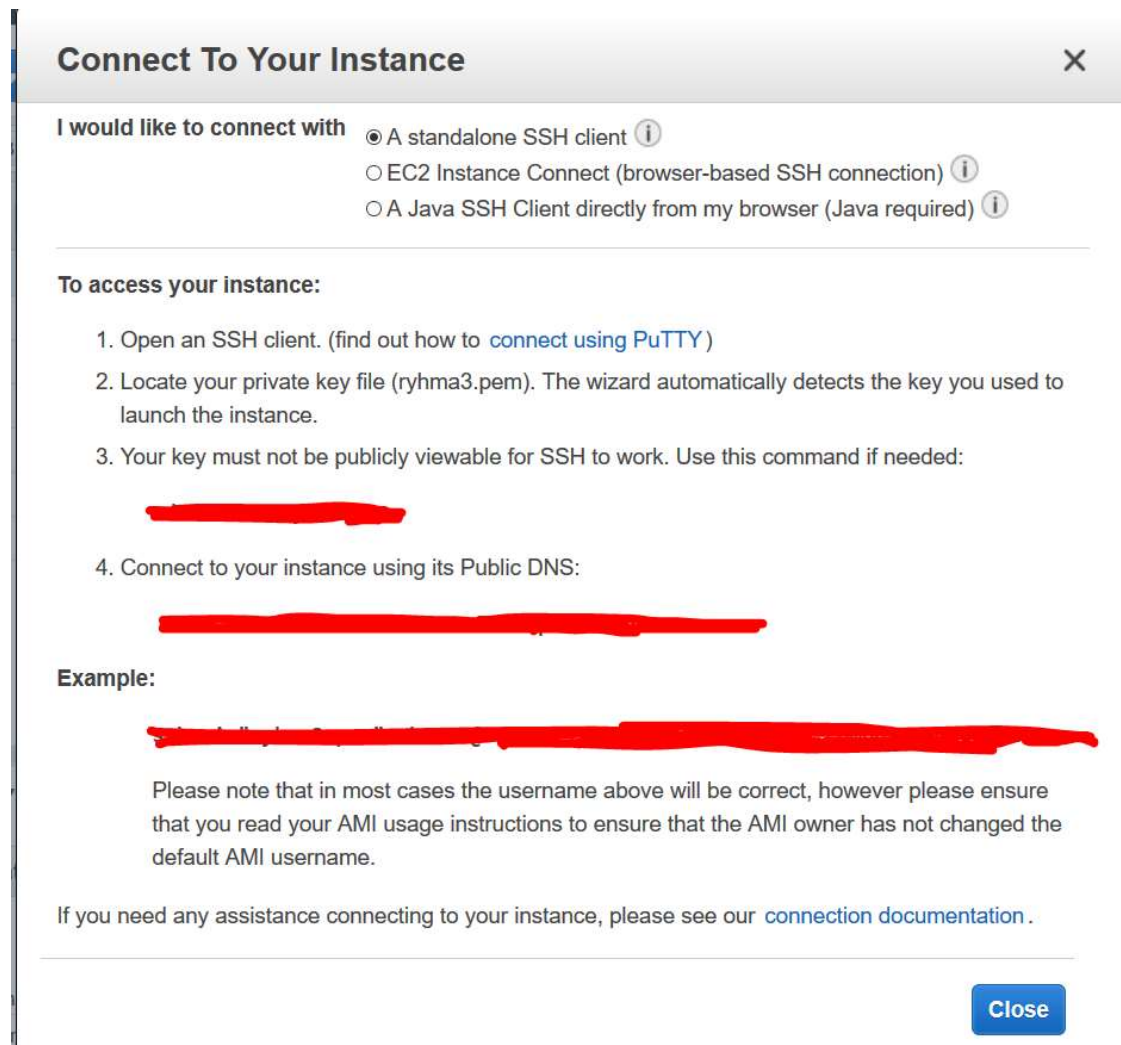
Kuva 5. Ulkoverkon Security Group säännöt.

Jokaiselle koneelle asennettiin Ubuntu 16.04 Server.

Yhteys koneeseen onnistui SSH-private avaimen avulla (Kuva 6) ja tarvittavat osoitteet löytyivät AWS – Connect valikosta.



Kuva 6 SSH-private avain.



Kuva 7. Uusi yhteys instanssiin.

Koska alustan tyyppi oli VPC, oli käytössämme staattiset sisäiset IP-osoitteet, joten hostname muokkaus ei ollut välttämätöntä (Amazon EC2 Instance IP Addressing).

## Account Attributes

### Supported Platforms

#### VPC

Your instance receives a static private IPv4 address from the address range of your default VPC.

Kuva 8. Käyttäjän asetukset.

## 4.2 HAProxy

Kone1 asetettiin ulkoverkkoon ja siihen asennettiin HAProxy, jonka tehtävänä oli järjestelmän kuormantasaaminen. Järjestelmään tuleva liikenne ohjattiin HAProxylla Contriboardin eri instansseille. Tässä tapauksessa sillä ohjattiin myös järjestelmän sisäistä liikennettä IO-komponentille.

HAProxyn asennus Ubuntu ympäristössä oli hyvin yksinkertainen ja se tapahtui yhdellä komennolla:

```
sudo apt install haproxy
```

Tämä asentaa HAProxyn ja luo vaadittavat servicet ja muut sellaiset. HAProxy palvelua voi ohjata seuraavilla komennoilla:

1. `sudo service haproxy stop`
2. `sudo service haproxy start`
3. `sudo service haproxy restart`

HAProxyn konfiguraatio tiedostot sijaitsevat polussa `/etc/haproxy/haproxy.cfg`. Tästä tiedostosta oli hyvä ottaa kopio komennolla:

```
sudo apt cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg.bak
```

Laboratorion materiaaleissa annetun HAProxyn konfiguraation pystyi kopioimaan täysin, mutta muutama IP-osoite piti korjata, jotta Contriboard toimi oikein.

backend API:

```
server API1 172.31.132.111:9001 weight 1 maxconn 2048 check
```

backend WS:

```
server WebSocket1 172.31.132.111:9002 weight 1 maxconn 2048 check
```

backend CLIENT

```
server client 172.31.132.111:80 weight 1 maxconn 2048 check
```

Nämä kolme riviä muutettiin siten, että IP osoite vastasi sisäverkon (Kone3) IP-osoitetta. Tämän jälkeen HAProxy oli palvelun uudelleenkäynnistämisen jälkeen valmis.

## 4.3 Docker

Dockerin asennus sujui Dockerin tarjoamilla ohjeilla. Ohjeet oli tarkoitettu Ubuntu versioille 16.04 LTS, 18.04 LTS, 18.10 sekä 19.04. Koneessamme oli Ubuntu Server 16.04. Asennus sujui seuraavalla tavalla.

1. Päivitetään Ubuntu pakettilähtö:

```
$ sudo apt-get update
```

2. Asennetaan paketit, jotta APT voi käyttää repositorioita http:n yli:

```
sudo apt-get install \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common
```

3. Lisää Dockerin GPG-avain, jotta voidaan asentaa paketteja Dockerin repoista ja tarkistetaan että avain on ok:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
$ sudo apt-key fingerprint 0EBFCD88
```

```
pub   rsa4096 2017-02-22 [SCEA]
      9DC8 5822 9FC7 DD38 854A E2D8 8D81 803C 0EBF CD88
uid           [ unknown] Docker Release (CE deb) <docker@docker.com>
sub   rsa4096 2017-02-22 [S]
```



#### 4. Lisätään Dockerin repositoriot Ubuntuun:

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

Tämä siis x86\_64 prosessoriarkkitehtuurille, jos eri, niin vaihtaa arch=amd64 johonkin toiseen, esim. arm64, jos 64-bittinen arm-prosessori.

Meidän serverimme oli x86\_64.

#### 5. Sitten päivitetään paketticache:

```
sudo apt-get update
```

#### 6. Ja asennetaan Dockerin paketit:

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

## 4.4 Contriboat

### 4.4.1 Asennuksen valmistelu

Contriboat koostuu viidestä eri Docker-kontista:

1. Client
2. API
3. IO
4. REDIS
5. MongoDB

Kontit tarvitsevat erinäisiä ympäristömuuttujia, jotka määritellään tiedostoissa, jotka luetaan kontin ajon yhteydessä. Tehdään kansio, jossa tiedostot ovat, meidän tapauksessamme se oli /envit/ (Kuva 9).

```
ubuntu@ip-172-31-132-111:~$ ls /envit/
api.txt client.txt io.txt
```

Kuva 9 envit kansio.

Eli api.txt sisältää ympäristömuuttujat API-kontille, client.txt Client-kontille jne. Niistä muutettiin lähinnä IP-osoitteet: api.txt tiedostosta MongoDB-kontin sekä

REDIS-kontin osoitteet, client.txt tiedostosta API- ja IO-konttien osoitteet ja io.txt tiedostosta REDIS- ja API-konttien osoitteet.

API-kontin täytyi tietää MongoDB-kontin sijainti, ja tässä tapauksessa se oli samalla koneella eli käytimme koneen omaa IP-osoitetta, myös REDIS-kontti sijaitsi samalla koneella (Kuva 10).

```
ubuntu@ip-172-31-132-111:~$ cat /envit/api.txt
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
```

Kuva 10 api.txt sisältö.

Client-kontin asetuksissa (Kuva 11) täytyi huomioida, että client latautuu selaimeen ja se kommunikoi API:n ja IO:n kanssa ulkoapäin. Eli kyseiset osoitteet ovat HAProxy-koneen julkiset IP-osoitteet. Tällöin Client ottaa yhteyttä API:iin ja IO:hon HAProxy:n kautta, joka ohjaa liikenteen oikealle koneelle.

```
ubuntu@ip-172-31-132-111:~$ cat /envit/client.txt
NODE_ENV=production
IO_URL=http://34.244.248.43
IO_PORT=80
API_URL=http://34.244.248.43/api
API_PORT=80
```

Kuva 11 client.txt.

IO-kontissa määritetään REDIS- ja API-konttien osoite. IO:n ja API:n osoitteen täytyi osoittaa HAProxy-palvelimeen, jotta reaaliaikainen päivitys onnistui (Kuva 12).

```
ubuntu@ip-172-31-132-111:~$ cat /envit/io.txt
NODE_ENV=production
REDIS_HOST=172.31.132.111
REDIS_PORT=6379
API_URL=http://172.31.131.239:80/api
PORT=9002
NEW_RELIC_LOG=/home/teamboard/logs/new_relic_io.log
NEW_RELIC_ENABLED=false
```

Kuva 12 io.txt.

## 4.4.2 Konttien käynnistys

Kun konttien ympäristömuutujat oli määritelty, voitiin käynnistää kontit. Jos konttia ei ole lokaalisti, Docker lataa sen. Kuvissa 13-17 näytetään millä käskyillä kontit käynnistetään.

```
ubuntu@ip-172-31-132-111:~$ sudo docker run -d -p 27017:27017 -v
/var/lib/mongodb:/data/db --name mongodb mongo:3.2
Unable to find image 'mongo:3.2' locally
3.2: Pulling from library/mongo
a92a4af0fb9c: Pull complete
74a2c7f3849e: Pull complete
927b52ab29bb: Pull complete
e941def14025: Pull complete
be6fce289e32: Pull complete
f6d82baac946: Pull complete
7cla640b9ded: Pull complete
e8b2fc34c941: Pull complete
1fd822faa46a: Pull complete
61ba5f01559c: Pull complete
db344da27f9a: Pull complete
Digest: sha256:0463a91d8eff189747348c154507afc7aba045baa40e8d58d8
a4c798e71001f3
Status: Downloaded newer image for mongo:3.2
f0e551bfca8b9beec18e7211719a2239cfb6920d379f4ae9e71e40e3225b9c7e
ubuntu@ip-172-31-132-111:~$
```

Kuva 13 MongoDB-kontin käynnistys.

```
db344da27f9a: Pull complete
Digest: sha256:0463a91d8eff189747348c154507afc7aba045baa40e8d58d8
a4c798e71001f3
Status: Downloaded newer image for mongo:3.2
f0e551bfca8b9beec18e7211719a2239cfb6920d379f4ae9e71e40e3225b9c7e
ubuntu@ip-172-31-132-111:~$ sudo docker run -p 6379:6379 --name r
edis -d redis
Unable to find image 'redis:latest' locally
latest: Pulling from library/redis
b8f262c62ec6: Pull complete
93789b5343a5: Pull complete
49cddb315637: Pull complete
ea0579387266: Pull complete
6b8ecda334de: Pull complete
ac5a9c26d32a: Pull complete
Digest: sha256:cb379e1a076fcd3d3f09e10d7b47ca631fb98fb33149ab559f
a02c1b11436345
Status: Downloaded newer image for redis:latest
d0df8338b1e20f448d7a98e2ede52a3b7c2378bd01de53bf174d60a0610e865f
ubuntu@ip-172-31-132-111:~$
```

Kuva 14 REDIS-kontin käynnistys.

```
ubuntu@ip-172-31-132-111:~$ sudo docker run -d --env-file=/envit/
api.txt -p 9001:9001 --expose=9001 -v /tmp:/home/teamboard/logs -
--name contriboard-api n4sjamk/teamboard-api
Unable to find image 'n4sjamk/teamboard-api:latest' locally
latest: Pulling from n4sjamk/teamboard-api
[DEPRECATION NOTICE] registry v2 schemal support will be removed
in an upcoming release. Please contact admins of the docker.io re
gistry NOW to avoid future disruption.
8387d9ff0016: Pull complete
3b52deaaf0ed: Pull complete
4bd501fad6de: Pull complete
a3ed95caeb02: Pull complete
2632858d59eb: Pull complete
25b4711b9829: Pull complete
9c841d203e4e: Pull complete
874648b7bc41: Pull complete
93f06be0f0e2: Pull complete
34ca340504dc: Pull complete
bb9fc5169158: Pull complete
47662b5c54fd: Pull complete
```

Kuva 15 API-kontin käynnistys.

```

ubuntu@ip-172-31-132-111:~$ sudo docker run -d --env-file=/envit/
io.txt -p 9002:9002 --expose=9002 -v /tmp:/home/teamboard/logs --
name contriboard-io n4sjamk/teamboard-io
Unable to find image 'n4sjamk/teamboard-io:latest' locally

latest: Pulling from n4sjamk/teamboard-io
[DEPRECATION NOTICE] registry v2 schema1 support will be removed
in an upcoming release. Please contact admins of the docker.io re
gistry NOW to avoid future disruption.
8387d9ff0016: Already exists
3b52deaaf0ed: Already exists
4bd501fad6de: Already exists
a3ed95caeb02: Pull complete
3353c382df3f: Pull complete
06307108d5fa: Pull complete
464c57e0685d: Pull complete
b97967ae673b: Pull complete
ec722a433a1b: Pull complete
67d6dacaceb9: Pull complete
778a5e17433c: Pull complete

```

Kuva 16 IO-kontin käynnistys.

```

ubuntu@ip-172-31-132-111:~$ sudo docker run -d --env-file=/envit/
client.txt -p 8080:80 --expose=80 --name contriboard-client n4sja
mk/teamboard-client
Unable to find image 'n4sjamk/teamboard-client:latest' locally
latest: Pulling from n4sjamk/teamboard-client
[DEPRECATION NOTICE] registry v2 schema1 support will be removed
in an upcoming release. Please contact admins of the docker.io re
gistry NOW to avoid future disruption.
8387d9ff0016: Already exists
3b52deaaf0ed: Already exists
4bd501fad6de: Already exists
a3ed95caeb02: Pull complete
5e2910959c1c: Pull complete
7bd9ba9463a1: Pull complete
cd91a7bb6586: Pull complete
5f269a3150a8: Pull complete
771694e3c292: Pull complete
3f6dab737d4f: Pull complete
06fcb876394f: Pull complete
f1965c83959c: Pull complete

```

Kuva 17 Client-kontin käynnistys.

Komennolla "sudo docker ps -a" (Kuva 18) voidaan tarkistaa, että kontit ovat käyn-  
nissä.

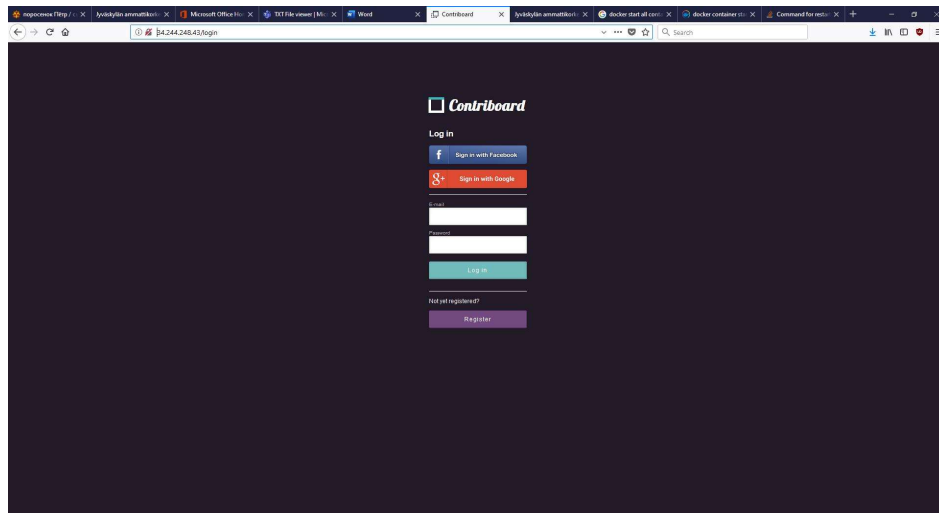
```

ubuntu@ip-172-31-132-111:~$ sudo docker ps -a
CONTAINER ID        IMAGE               STATUS             COMMAND
a337e3e3f955       n4sjamk/teamboard-client   Up 9 seconds      /bin/sh -c 'od /h
m...'
33 minutes ago    Up 9 seconds      0.0.0.0:8080->8080/t
cp
a3ed95caeb02       n4sjamk/teamboard-io       Up 8 seconds      /bin/sh -c '/usr/
/bin/redis-server
--port 9002'
33 minutes ago    Up 8 seconds      0.0.0.0:9002->9002/
tcp
a3ed95caeb02       n4sjamk/teamboard-api       Up 7 seconds      /bin/sh -c '/usr/
/bin/redis-server
--port 9001'
34 minutes ago    Up 7 seconds      0.0.0.0:9001->9001/
tcp
a3ed95caeb02       redis                    "docker-entrypoint
"
41 minutes ago    Up 6 seconds      0.0.0.0:6379->6379/
tcp
a3ed95caeb02       mongo:3.2                "docker-entrypoint
"
45 minutes ago    Up 5 seconds      0.0.0.0:27017->270
17/tcp
a3ed95caeb02       mongo:3.2                "docker-entrypoint
"
45 minutes ago    Up 5 seconds      0.0.0.0:27017->270
17/tcp
CONTAINER ID        IMAGE               STATUS             COMMAND
a337e3e3f955       n4sjamk/teamboard-client   Up About a minute /bin/sh -c 'od /h
m...'
33 minutes ago    Up About a minute 0.0.0.0:8080->8080/t
cp
a3ed95caeb02       n4sjamk/teamboard-io       Up About a minute /bin/sh -c '/usr/
/bin/redis-server
--port 9002'
34 minutes ago    Up About a minute 0.0.0.0:9002->9002/
tcp
a3ed95caeb02       n4sjamk/teamboard-api       Up About a minute /bin/sh -c '/usr/
/bin/redis-server
--port 9001'
35 minutes ago    Up About a minute 0.0.0.0:9001->9001/
tcp
a3ed95caeb02       redis                    "docker-entrypoint
"
43 minutes ago    Up About a minute 0.0.0.0:6379->6379/
tcp
a3ed95caeb02       mongo:3.2                "docker-entrypoint
"
46 minutes ago    Up About a minute 0.0.0.0:27017->270
17/tcp
a3ed95caeb02       mongo:3.2                "docker-entrypoint
"
47 minutes ago    Up About a minute 0.0.0.0:27017->270
17/tcp
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS             PORTS             NAMES
a337e3e3f955       n4sjamk/teamboard-client   "/bin/sh -c 'od /h..."   33 minutes ago      Up About a minute   0.0.0.0:8080->8080/tcp   contriboard-client
a3ed95caeb02       n4sjamk/teamboard-io       "/bin/sh -c '/usr/bi..."   35 minutes ago      Up About a minute   0.0.0.0:9002->9002/tcp   contriboard-io
a3ed95caeb02       n4sjamk/teamboard-api       "/bin/sh -c '/usr/bi..."   35 minutes ago      Up About a minute   0.0.0.0:9001->9001/tcp   contriboard-api
a3ed95caeb02       redis                       "docker-entrypoint.sh..."   43 minutes ago      Up About a minute   0.0.0.0:6379->6379/tcp   redis
a3ed95caeb02       mongo:3.2                  "docker-entrypoint.sh..."   47 minutes ago      Up About a minute   0.0.0.0:27017->27017/tcp   mongoDB

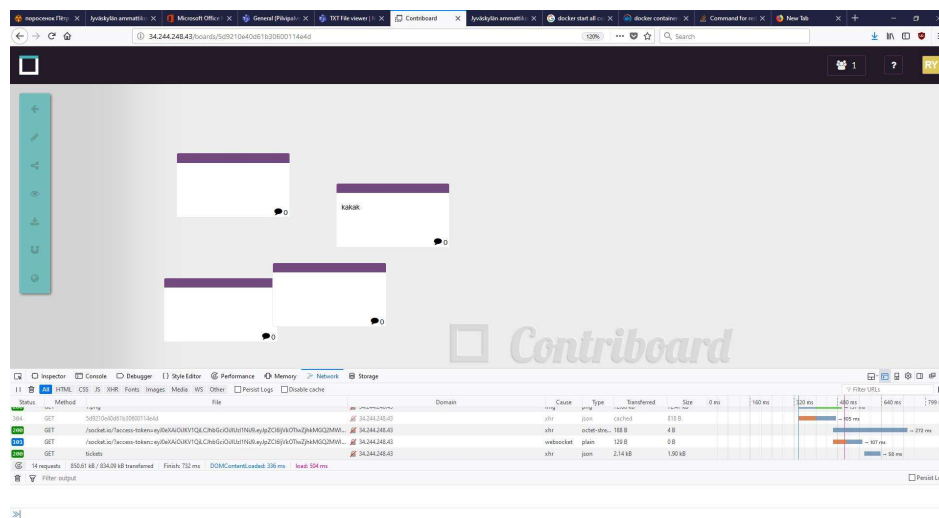
```

Kuva 18 Docker-kontit käynnissä.

Tämän jälkeen voitiin mennä selaimella IP-osoitteeseen, joka oli HAProxy-koneen jul-  
kinen IP. Tämä ohjasi ContriBoard-Client:in näkymään selaimen (Kuva 19 ja Kuva  
20).



Kuva 19 Contriboard toiminnassa.



Kuva 20 Contriboard toiminnassa.

## 5 Pohdinta

Työn aloitus ja sen tekeminen kokonaisuudessaan oli hyvin sulava prosessi. Työssä käytettiin kuivaharjoittelussa kirjoitettuja muistiinpanoja, jotka nopeuttivat työn valmistumista huomattavasti. Contriboardin toiminta oli alussa hieman kankeaa, koska eräs asetus oli jäänyt väärään IP-osoitteeseen. Tästä johtuen, Contriboard ei päivittänyt reaaliajassa verkon ylitse. Tämä saatiin kuitenkin korjattua HAProxyn asetuksista helposti.

## Lähteet

Amazon EC2 Instance IP Addressing, Artikkel [docs.aws.amazon.com](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#concepts-private-addresses) sivustolla. Viitattu 1.10.2019, <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-instance-addressing.html#concepts-private-addresses>.

What is a Container, Artikkel [docker.com](https://www.docker.com/resources/what-container) sivustolla. Viitattu 27.10.2019. <https://www.docker.com/resources/what-container>.