Vilniaus Gedimino technikos universitetas

Elektronikos fakultetas

Kompiuterijos ir ryšių technologijų katedra

Debesų kompiuterija

Modulis ELKRM17304

Docker Swarm klasterio įdiegimas ir tyrimas

Kursinis darbas

Atliko: TETfm-20 grupės magistrantas
Saulius Krasuckas
Tikrino: lekt. dr. Liudas Duoba

VILNIUS, 2022

Kursinis darbas

Docker Swarm klasterio įdiegimas ir tyrimas

1. Kursinio darbo užduotis

- Savarankiškai išstudijuoti *Docker Swarm* veikimo principus, juos aprašyti
- Sukonfigūruoti *Docker Swarm* klasterį, kurį sudaro ne mažiau trijų kompiuterių / virtualių mašinų (VM)
- Atvaizduoti klasterio konfigūraciją grafiškai (pateikti schemas)
- Klasteryje paleisti *Web*-servisą, veikiantį konteinerių pagrindu. Servisas turi būti pasiekiamas.
- Sukurti ne mažiau kaip 3 serviso kopijas (replikas). Parodyti, kaip servisai pasiskirsto klasteryje.
- Imituoti vieno iš klasterio elementų gedimą; aprašyti / parodyti poveikį servisams.
- Parodyti / atvaizduoti *Web*-serviso pasiekiamumą visuose etapuose.
- Pateikti naudojamas komandas

2. Pagrindinė dalis

Docker Swarm veikimo principai

1. Nuo versijos 1.12 *Swarm mode* mechanizmas buvo integruotas į Docker CLI tiesiogiai. [4]
   Anksčiau jis reikalaudavo papildomų konteinerių kūrimo ir vadinosi *Classic Swarm*.
2. Mechanizmas įgalina skirstyti konteinerius tarp daugelio Docker hostų.
3. Tam naudojamas *Overlay* tipo tinklas su *Mesh routing* technlogija.
4. Jis leidžia apjungti hostus į vieną tinklą ir įgalina jame atlikti servisų *Discovery* ir *Load*-balansavimą.
5. Konteinerių idėją papildo trys nauji konceptai:
   a. **Mazgas** (angl. *Node*) — tai atskiras Docker variklio egzempliorius (hostas), prijungtas prie *Swarm* klasterio.
      Mazgai būna arba *Worker* tipo, arba *Manager* tipo:
      - *Manager* dėlioja, kur koks konteineris veiks.
      - *Worker* tiesiog tuos konteinerius vykdo.
      - Taipogi *Manager* tipo mazgas gali būti ir *Worker* mazgu.
   b. **Servisas** (angl. *Service*) — abstraktesnis konceptas, siejantis grupę užduočių, kurias turėtų vykdyti *Workers* tam, kad pasiektų integralų rezultatą.
   c. **Apkrovos balansavimas** (angl. *Load Balancing*) — tai užklausų, ateinančių į bet kurį *Swarm* mazgą paskirstymas į konteinerius, vykdančius atitinkamą paslaugą.

6. *Overlay* tinklas.
   Tai `VxLAN` technologija grįsta tinklo architektūra, leidžianti apimti skirtinguose debesyse ir duomenų centruose veikiančius Docker hostus.
   Veikimui naudojamas *Routing Mesh*, *Virtual IP* ir *Linux IPVS* — multiprotokolinis (Layer-4) apkrovos balansavimo mechanizmas.

Docker Swarm klasterio konfigūravimas

1. Infrastruktūrai kuri pasirinkau namie turimą nešiojamąjį kompiuterį su Windows OS.
2. Jame įsidiegiau VirtualBox VMM (arba hipervizorių).
3. Kaip *Guest OS* pasirinkau 64-bit **Ubuntu 20.04.3 LTS** Linux distribuciją.
4. Pasinaudojau `OSboxes.org` projekto teikiamu įdiegtos OS atvaizdžiu. [1]
5. VM kūrimui ir valdymui pasirinkau VirtualBox CLI `VBoxManage` ir MSYS2 įrankį, kuris Windows OS suteikia **nix* tipo aplinką.
6. Čia susikūriau kelis *Bash* skriptus:
   - `build-infra.sh`
     (https://github.com/VGTU-ELF/TETfm-20/tree/main/Semestras-3/2-Debes%C5%B3-kompiuterija/kursinis-darbas/Saulius-Krasuckas#:~:text=build%2Dinfra.sh,ubuntu%2Dhostnames.sh)
     (*Golden image* ir atskirų VM formavimui)
   - `setup-osboxes-ubuntu-20.04.sh`
     (https://github.com/VGTU-ELF/TETfm-20/blob/main/Semestras-3/2-Debes%C5%B3-kompiuterija/kursinis-darbas/Saulius-Krasuckas/setup-osboxes-ubuntu-20.04.sh)
     (VM tvarkymo eiga)
   - `osboxes-ubuntu-20.04-changes.sh`
     (https://github.com/VGTU-ELF/TETfm-20/blob/main/Semestras-3/2-Debes%C5%B3-kompiuterija/kursinis-darbas/Saulius-Krasuckas/osboxes-ubuntu-20.04-changes.sh)
     (pagrindiniai Guest OS tvarkymo veiksmai)
   - `setup-ubuntu-docker.sh`
     (https://github.com/VGTU-ELF/TETfm-20/blob/main/Semestras-3/2-Debes%C5%B3-kompiuterija/kursinis-darbas/Saulius-Krasuckas/setup-ubuntu-docker.sh)
     (*Docker* įdiegimas)
   - `setup-ubuntu-hostnames.sh`
     (https://github.com/VGTU-ELF/TETfm-20/blob/main/Semestras-3/2-Debes%C5%B3-kompiuterija/kursinis-darbas/Saulius-Krasuckas/setup-ubuntu-hostnames.sh)
     (individualizuotų mazgo vardų tvarkymas)
   - Skriptų naudojimo privalumas — lengva turėti kad ir 20 identiškų virtualių mašinų.
     O padarius konfigūravimo klaidą, lengva ją pataisyti ir visą infrastruktūrą susikurti iš naujo.
7. Startavus `build-infra.sh`:
   - Parsisiunčiamas `Ubuntu 20.04.3 (64bit).vdi` atvaizdis.
   - Jo pagrindu sukuriama etaloninė VM.
   - Ji startuojama, ir atliekami pagrindiniai OS tvarkymo veiksmai (SSH raktų tvarkymas, `sudo` perkonfigūravimas, naujinimai, paketų diegimas, perkrovimas, Docker diegimas).
   - VM išjungiama, o disko atvaizdis paruošiamas jungimui prie keleto mašinų (angl. *Multi-attach*).

- Sukuriamos trys VM pagal bendrą šabloną:
  - 1 GiB RAM, 2 CPU.
  - 1 NIC išėjimui į internetą (angl. *Default route*);
  - 1 NIC Docker klasterio ryšiui (*App*);
  - 1 NIC OAM ryšiui (angl. *Operation, Administration, Maintenance*).
  - Visi NIC gauna adresus iš VBox integruoto DHCP serviso.
  - Kiekvienai VM nustatomas OAM IP adresas.
  - Prie jo prisijungiama automatiškai.
  - `/etc/hosts` faile užregistruojami suteikti IP adresai ir mazgo vardai.
  - Tuomet šie duomenys surenkami į bendrą failą ir padalinimi į visus Guest OS iš eilės.
  - Taip pat patvirtinami SSH ECDSA raktai tarp skirtingų mazgų.
- Trys VM paruoštos darbui.
- Išskyrus atvaizdžio siuntimo laiką, paruošimas trunka apie 65 min.

8. Rankiniu būdu konfigūruoju *Docker Swarm mode* klasterį pagal Docker dokumentacijos pamoką: [2]

   Patikrinimas:

   ```
   $ ssh swarm-n01-oam sudo docker info | grep --color -e Swarm: -e CPUs: -e Total.Memory:
    Swarm: inactive
    CPUs: 2
    Total Memory: 971.2MiB
   ```

   Pirmas bandymas startuoti *Swarm* klasterį:

   ```
   $ ssh swarm-n01-oam sudo docker swarm init
   Error response from daemon: could not choose an IP address to advertise since this system has multiple
   addresses on different interfaces (10.0.2.15 on enp0s3 and 10.1.1.24 on enp0s8) - specify one with --
   advertise-addr
   ```

   Bandau nurodyti klasterio ryšio adresą kaip mazgo vardą:

   ```
   $ ssh swarm-n01-oam sudo docker swarm init --advertise-addr swarm-n01
   Error response from daemon: advertise address must be a non-zero IP address or network interface (with
   optional port number)
   ```

   Netiko. Pateikus interfeiso vardą tiko:

   ```
   $ ssh swarm-n01-oam sudo docker swarm init --advertise-addr enp0s8
   Swarm initialized: current node (l6wnnbsgv2th6nq05e9j02srj) is now a manager.

   To add a worker to this swarm, run the following command:

       docker swarm join --token SWMTKN-1-40jfoeoj9kgwtcqbtc9klrwaeoh8ogfebxoa8r1euzxnzfe7ha-
   ee278x6iuxb6ny7g4v34z9phw 10.1.1.24:2377

   To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
   ```

   Tikrinu būseną:

```
$ ssh swarm-n01-oam sudo docker info | grep --color -e ^ -e Swarm
  ...
 Swarm: active
  NodeID: l6wnnbsgv2th6nq05e9j02srj
  Is Manager: true
  ClusterID: tbmszwsuuyydpgzw90lsblvjd
  Managers: 1
  Nodes: 1
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
   Snapshot Interval: 10000
   Number of Old Snapshots to Retain: 0
   Heartbeat Tick: 1
   Election Tick: 10
  Dispatcher:
   Heartbeat Period: 5 seconds
  CA Configuration:
   Expiry Duration: 3 months
   Force Rotate: 0
  Autolock Managers: false
  Root Rotation In Progress: false
  Node Address: 10.1.1.24
  Manager Addresses:
   10.1.1.24:2377
  ...
```

Sutikrinu su interfeisų IP adresais:

```
$ ssh swarm-n01-oam ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7e:2a:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 85952sec preferred_lft 85952sec
    inet6 fe80::72a6:ed0b:5033:2f37/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f0:5c:76 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.24/24 brd 10.1.1.255 scope global dynamic noprefixroute enp0s8
       valid_lft 453sec preferred_lft 453sec
    inet6 fe80::1d12:9739:5544:643a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:57:72:bd brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s9
       valid_lft 453sec preferred_lft 453sec
    inet6 fe80::e076:cc40:af50:5f45/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:2f:74:cc:6f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
10: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:2c:42:48:68 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
    inet6 fe80::42:2cff:fe42:4868/64 scope link
       valid_lft forever preferred_lft forever
12: vethb5ec981@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether 06:3e:2a:e4:2c:55 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::43e:2aff:fee4:2c55/64 scope link
       valid_lft forever preferred_lft forever
```

Atitinka `enp0s8`. Tikrinu mazgų sąrašą:

```
$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
```

Prijungiu antrą mazgą:

```
$ ssh swarm-n02-oam sudo docker swarm join --token SWMTKN-1-
40jfoeoj9kgwtcqbtc9klrwaeoh8ogfebxoa8r1euzxnzfe7ha-ee278x6iuxb6ny7g4v34z9phw 10.1.1.24:2377
This node joined a swarm as a worker.
```

Tikrinu mazgų sąrašą:

```
$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
```

Prijungiu trečią narį:

```
$ ssh swarm-n03-oam sudo docker swarm join --token SWMTKN-1-
40jfoeoj9kgwtcqbtc9klrwaeoh8ogfebxoa8r1euzxnzfe7ha-ee278x6iuxb6ny7g4v34z9phw 10.1.1.24:2377
This node joined a swarm as a worker.
```

Patikrinu, jau visi trys klasteryje:

```
$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12
```

Klasterio konfigūracija

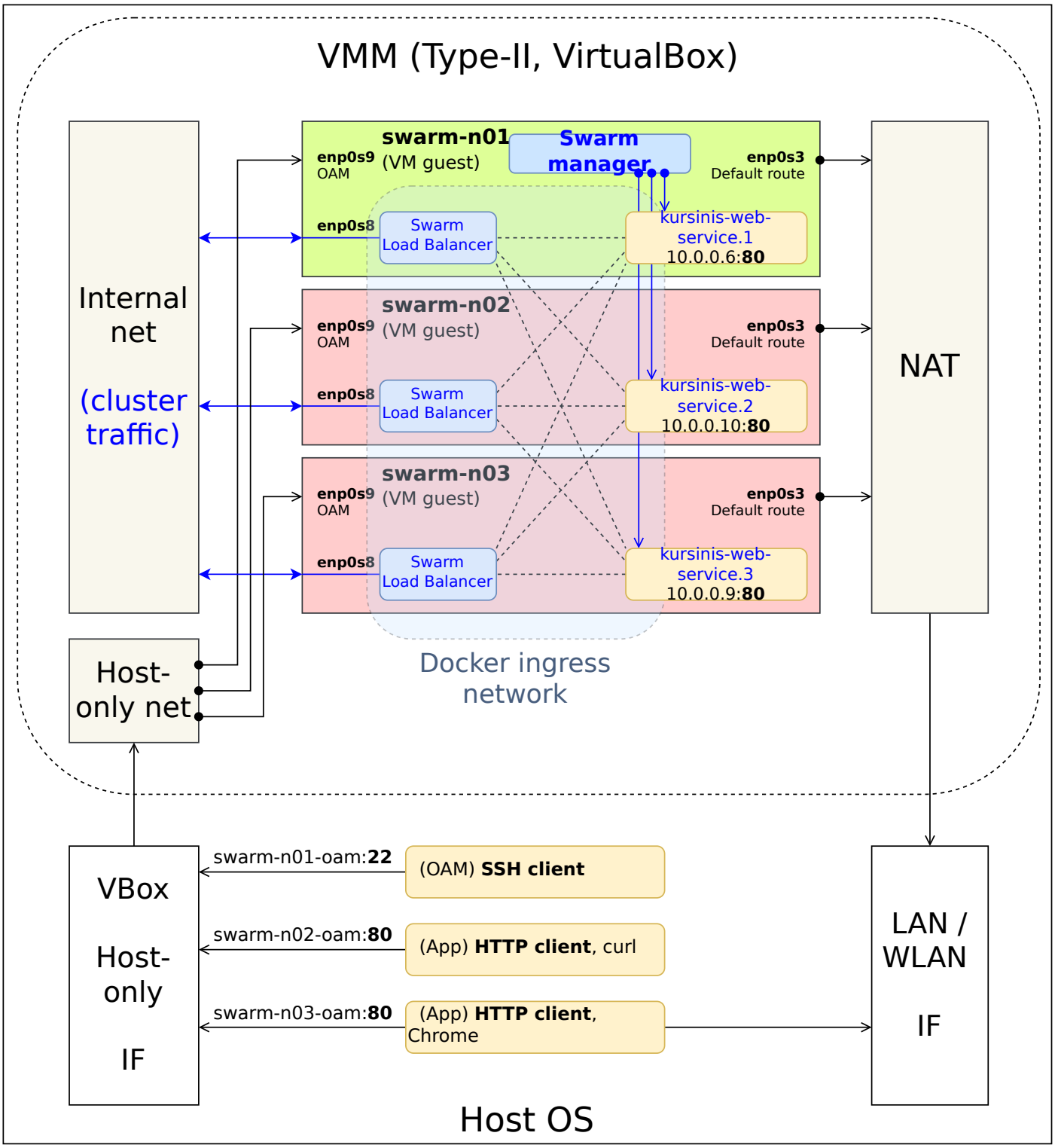


Figure 1. (1 pav.) Klasterio mazgų konfigūracija tarpusavyje ir Host OS atžvilgiu.

Web-serviso startavimas ir tikrinimas

1. Iš visos galybės pasirenku *Web*-servisą, matytą *KataCoda* puslapio treniruotėse: `katacoda/docker-http-server`. [3]
   Ir startuoju pavienį konteinerį su juo:

```
$ ssh swarm-n01-oam sudo docker run -d --name http-band -p 80:80 katacoda/docker-http-server
Unable to find image 'katacoda/docker-http-server:latest' locally
latest: Pulling from katacoda/docker-http-server
f139eb4721ae: Pulling fs layer
f139eb4721ae: Verifying Checksum
f139eb4721ae: Download complete
f139eb4721ae: Pull complete
Digest: sha256:76dc8a47fd019f80f2a3163aba789faf55b41b2fb06397653610c754cb12d3ee
Status: Downloaded newer image for katacoda/docker-http-server:latest
84f32317148cac3ea8dfffb6587258f905d8563064302b7fc457d35156dd4240
```

2. Tikrinu, konteineris veikia:

```
$ ssh swarm-n01-oam sudo docker ps
CONTAINER ID   IMAGE                          COMMAND   CREATED          STATUS          PORTS
NAMES
84f32317148c   katacoda/docker-http-server   "/app"    20 seconds ago   Up 19 seconds   0.0.0.0:80-
>80/tcp, :::80->80/tcp   http-band
```

3. Tikrinu su lokaliu http-klientu, veikia:

```
$ ssh swarm-n01-oam curl -s localhost
<h1>This request was processed by host: 84f32317148c</h1>
```

4. Ištrinu bandomąjį konteinerį:

```
$ ssh swarm-n01-oam sudo docker stop http-band
http-band

$ ssh swarm-n01-oam sudo docker rm http-band
http-band
```

5. Pagal jo atvaizdį kuriu jau ne pavienį, o klasterinį *Web*-servisą `kursinis-web-service`:

```
$ ssh swarm-n01-oam sudo docker service create --name kursinis-web-service -p 80:80 katacoda/docker-http-
server
p7imsxwi9midpu5b378srq54w
overall progress: 0 out of 1 tasks
1/1:
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 0 out of 1 tasks
overall progress: 1 out of 1 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Service converged
```

6. Klasterinių servisų sąrašas:

```
$ ssh swarm-n01-oam sudo docker service ls
ID             NAME                   MODE         REPLICAS   IMAGE                                PORTS
p7imsxwi9mid   kursinis-web-service   replicated   1/1        katacoda/docker-http-server:latest   *:80-
>80/tcp
```

   Veikia, naudoja tik 1 repliką (pagal nutylėjimą).

7. Tikrinu servisą lokaliai:

```
$ ssh swarm-n01-oam curl -s localhost
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

8. Patikrinu lokalių konteinerių būseną pirmame mazge, veikia lygiai vienas:

```
$ ssh swarm-n01-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED          STATUS          PORTS
NAMES
9c2d26cbff9e   katacoda/docker-http-server:latest   "/app"    34 seconds ago   Up 32 seconds   80/tcp
kursinis-web-service.1.vekfptu7x3egid5me161x4gbj
```

9. Serviso pasiekiamumas mazguose:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service
ID             NAME                     IMAGE                                NODE        DESIRED STATE
CURRENT STATE                ERROR     PORTS
vekfptu7x3eg   kursinis-web-service.1   katacoda/docker-http-server:latest   swarm-n01   Running
Running about a minute ago
```

Kol kas veikia tik viename mazge.

10. Detalus serviso inspektavimas:

```
$ ssh swarm-n01-oam sudo docker service inspect kursinis-web-service
[
    {
        "ID": "p7imsxwi9midpu5b378srq54w",
        "Version": {
            "Index": 23
        },
        "CreatedAt": "2022-02-09T09:48:19.22071502Z",
        "UpdatedAt": "2022-02-09T09:48:19.226648699Z",
        "Spec": {
            "Name": "kursinis-web-service",
            "Labels": {},
            "TaskTemplate": {
                "ContainerSpec": {
                    "Image": "katacoda/docker-http-
server:latest@sha256:76dc8a47fd019f80f2a3163aba789faf55b41b2fb06397653610c754cb12d3ee",
                    "Init": false,
                    "StopGracePeriod": 10000000000,
                    "DNSConfig": {},
                    "Isolation": "default"
                },
                "Resources": {
                    "Limits": {},
                    "Reservations": {}
                },
                "RestartPolicy": {
                    "Condition": "any",
                    "Delay": 5000000000,
                    "MaxAttempts": 0
                },
                "Placement": {
                    "Platforms": [
                        {
                            "Architecture": "amd64",
                            "OS": "linux"
                        }
                    ]
                },
                "ForceUpdate": 0,
                "Runtime": "container"
            },
            "Mode": {
                "Replicated": {
                    "Replicas": 1
                }
            },
            "UpdateConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "RollbackConfig": {
                "Parallelism": 1,
                "FailureAction": "pause",
                "Monitor": 5000000000,
                "MaxFailureRatio": 0,
                "Order": "stop-first"
            },
            "EndpointSpec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
```

```
                        "PublishMode": "ingress"
                    }
                ]
            }
        },
        "Endpoint": {
            "Spec": {
                "Mode": "vip",
                "Ports": [
                    {
                        "Protocol": "tcp",
                        "TargetPort": 80,
                        "PublishedPort": 80,
                        "PublishMode": "ingress"
                    }
                ]
            },
            "Ports": [
                {
                    "Protocol": "tcp",
                    "TargetPort": 80,
                    "PublishedPort": 80,
                    "PublishMode": "ingress"
                }
            ],
            "VirtualIPs": [
                {
                    "NetworkID": "vpyp1hp7w63i40yltmydhwl8o",
                    "Addr": "10.0.0.5/24"
                }
            ]
        }
    }
]
```


11. Malonesnis skaitymui serviso būsenos pavidalas:

```
$ ssh swarm-n01-oam sudo docker service inspect --pretty kursinis-web-service

ID:             p7imsxwi9midpu5b378srq54w
Name:           kursinis-web-service
Service Mode:   Replicated
 Replicas:      1
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         katacoda/docker-http-
server:latest@sha256:76dc8a47fd019f80f2a3163aba789faf55b41b2fb06397653610c754cb12d3ee
 Init:          false
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress
```

12. Servisas lyg veikia. Tačiau tarp įprastų *Listening* TCP soketų `80/TCP` nesimato:

```
$ ssh swarm-n01-oam ss -4nl
Netid  State   Recv-Q  Send-Q   Local Address:Port    Peer Address:Port Process
udp    UNCONN  0       0        127.0.0.53%lo:53           0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:631          0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:4789         0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:5353         0.0.0.0:*
udp    UNCONN  0       0              0.0.0.0:59148        0.0.0.0:*
tcp    LISTEN  0       4096     127.0.0.53%lo:53           0.0.0.0:*
tcp    LISTEN  0       128            0.0.0.0:22           0.0.0.0:*
tcp    LISTEN  0       5            127.0.0.1:631          0.0.0.0:*
```

13. Nuimu *tik* IPv4 rodymą ir tikrinu iš naujo:

```
$ ssh swarm-n02-oam sudo ss -nlp | grep :80
tcp    LISTEN  0        4096                                                *:80
*:*                      users:(("dockerd",pid=693,fd=25))
```

    ⇒ Matyti, jog mano serviso TCP soketą aptarnauja procesas `docker`.

14. Peržiūriu tinklo interfeisų sąrašą:

```
$ ssh swarm-n01-oam ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:7e:2a:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 82807sec preferred_lft 82807sec
    inet6 fe80::72a6:ed0b:5033:2f37/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f0:5c:76 brd ff:ff:ff:ff:ff:ff
    inet 10.1.1.24/24 brd 10.1.1.255 scope global dynamic noprefixroute enp0s8
       valid_lft 307sec preferred_lft 307sec
    inet6 fe80::1d12:9739:5544:643a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
4: enp0s9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:57:72:bd brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.101/24 brd 192.168.56.255 scope global dynamic noprefixroute enp0s9
       valid_lft 307sec preferred_lft 307sec
    inet6 fe80::e076:cc40:af50:5f45/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
5: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:2f:74:cc:6f brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
    inet6 fe80::42:2fff:fe74:cc6f/64 scope link
       valid_lft forever preferred_lft forever
10: docker_gwbridge: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:2c:42:48:68 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.1/16 brd 172.18.255.255 scope global docker_gwbridge
       valid_lft forever preferred_lft forever
    inet6 fe80::42:2cff:fe42:4868/64 scope link
       valid_lft forever preferred_lft forever
12: vethb5ec981@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether 06:3e:2a:e4:2c:55 brd ff:ff:ff:ff:ff:ff link-netnsid 1
    inet6 fe80::43e:2aff:fee4:2c55/64 scope link
       valid_lft forever preferred_lft forever
24: veth3505fba@if23: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master docker_gwbridge
state UP group default
    link/ether 86:1b:0d:15:e5:7b brd ff:ff:ff:ff:ff:ff link-netnsid 2
    inet6 fe80::841b:dff:fe15:e57b/64 scope link
       valid_lft forever preferred_lft forever
```

Dabar jis pasipildė dar vienu: `veth*@if23`

15. Patikrinu serviso pasiekiamumą lokaliai kreipiantis ne į Docker skirtą sisteminį tinklo interfeisą `enp0s8`, bet į OAM dedikuotą interfeisą `enp0s9` su visai kitu IP adresu:

```
$ ssh swarm-n01-oam ping 192.168.56.101
PING 192.168.56.101 (192.168.56.101) 56(84) bytes of data.
64 bytes from 192.168.56.101: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from 192.168.56.101: icmp_seq=2 ttl=64 time=0.083 ms

$ ssh swarm-n01-oam curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

Atsakymą iš serviso vis tiek gaunu. Kiek netikėta ir malonu.

16. Taip pat tikrinu serviso pasiekiamumą OAM interfeisu ir išorėje, ne tik lokaliai:

```
$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

⇒ Servisas atsiliepia ir Host OSe veikiančiam http-klientui. Puiku.

Serviso didinimas (plėtimas)

1. Padidinu (išplečiu) servisą iki trijų replikų:

```
$ ssh swarm-n01-oam sudo docker service scale kursinis-web-service=3
kursinis-web-service scaled to 3
overall progress: 0 out of 3 tasks
1/3:
2/3:
3/3:
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 1 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 3 out of 3 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
```

```
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Service converged
```

2. Ir tikrinu serviso pasiekiamumą iš naujo:

```
$ curl -s 192.168.56.101
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s 192.168.56.101
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

⇒ Atsakymuose matyti trys skirtingi Host-id.

3. Tikrinu serviso būsenos detales:

```
$ ssh swarm-n01-oam sudo docker service inspect --pretty kursinis-web-service

ID:             p7imsxwi9midpu5b378srq54w
Name:           kursinis-web-service
Service Mode:   Replicated
 Replicas:      3
Placement:
UpdateConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Update order:      stop-first
RollbackConfig:
 Parallelism:   1
 On failure:    pause
 Monitoring Period: 5s
 Max failure ratio: 0
 Rollback order:    stop-first
ContainerSpec:
 Image:         katacoda/docker-http-
server:latest@sha256:76dc8a47fd019f80f2a3163aba789faf55b41b2fb06397653610c754cb12d3ee
 Init:          false
Resources:
Endpoint Mode:  vip
Ports:
 PublishedPort = 80
  Protocol = tcp
  TargetPort = 80
  PublishMode = ingress

$ ssh swarm-n01-oam sudo docker service ls
ID             NAME                   MODE         REPLICAS   IMAGE                                PORTS
p7imsxwi9mid   kursinis-web-service   replicated   3/3        katacoda/docker-http-server:latest   *:80-
>80/tcp
```

Rodo tris replikas, kaip ir nurodžiau plėsdamas.

4. Tikrinu pavienius konteinerius:

```
$ ssh swarm-n01-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED         STATUS         PORTS
NAMES
9c2d26cbff9e   katacoda/docker-http-server:latest   "/app"    7 minutes ago   Up 7 minutes   80/tcp
kursinis-web-service.1.vekfptu7x3egid5me161x4gbj

$ ssh swarm-n02-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED              STATUS
PORTS     NAMES
04bce300bcc1   katacoda/docker-http-server:latest   "/app"    About a minute ago   Up About a minute
80/tcp    kursinis-web-service.2.y8zkkc2pgpj6q0rijb372wooo

$ ssh swarm-n03-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED              STATUS
PORTS     NAMES
3d80ed3126b6   katacoda/docker-http-server:latest   "/app"    About a minute ago   Up About a minute
80/tcp    kursinis-web-service.3.f7aekzcbukx3c8bjojsl2v4i1
```

⇒ Konteinerių ID atitinka http-atsakymuose matomus Hostų id.

5. Dabartinis paslaugos pasiekiamumas klasteryje pagal *Manager*:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service
ID             NAME                     IMAGE                                  NODE        DESIRED STATE
CURRENT STATE            ERROR     PORTS
vekfptu7x3eg   kursinis-web-service.1   katacoda/docker-http-server:latest   swarm-n01   Running
Running 21 minutes ago
y8zkkc2pgpj6   kursinis-web-service.2   katacoda/docker-http-server:latest   swarm-n02   Running
Running 14 minutes ago
f7aekzcbukx3   kursinis-web-service.3   katacoda/docker-http-server:latest   swarm-n03   Running
Running 14 minutes ago
```

⇒ Kiekvienam klasterio mazge veikia po vieną serviso egzempliorių (kopiją, repliką).

6. Išorės užklausų siuntimas į pirmą mazgą:

```
$ curl -s swarm-n01-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n01-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n01-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s swarm-n01-oam
<h1>This request was processed by host: 04bce300bcc1</h1>
```

Atsako trys skirtingi konteineriai.

7. Išorės užklausų siuntimas į antrą mazgą:

```
$ curl -s swarm-n02-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n02-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n02-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s swarm-n02-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n02-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>
```

Atsako trys tie patys skirtingi konteineriai.

8. Išorės užklausų siuntimas į trečią mazgą:

```
$ curl -s swarm-n03-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 04bce300bcc1</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 3d80ed3126b6</h1>

$ curl -s swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

Atsako vėl tie patys trys konteineriai.

⇒ Paslauga veikia trijuose mazguose, visame klasteryje.

Klasterio elemento gedimas ir įtaka

1. Tikrinu paslaugos pasiskirstymą:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                         IMAGE                               NODE        DESIRED STATE
CURRENT STATE                    ERROR                         PORTS
j7pgc2beau6m   kursinis-web-service.1       katacoda/docker-http-server:latest   swarm-n01   Running
Running about a minute ago
thiskoiwddq2   kursinis-web-service.2       katacoda/docker-http-server:latest   swarm-n02   Running
Running less than a second ago
dkz7wktnswg0   kursinis-web-service.4       katacoda/docker-http-server:latest   swarm-n03   Running
Running 41 minutes ago
```

2. Pasirenku pagrindinio mazgo (kuriame veikia _Manager) klasterinę tinklo „koją“ ir ją atjungiu:

```
$ VBoxManage list vms
"swarm-n01" {ae06ba44-a60d-44a3-91ac-abae7edfa962}
"swarm-n02" {ab715077-c6b7-4f6a-bb9a-aeed78bd658e}
"swarm-n03" {9c308870-6fa6-4288-bfb3-5446d37652a1}

$ VBoxManage controlvm "swarm-n01" setlinkstate2 off
```

3. Iškart tikrinu tiesiogines užklausas per antrą mazgą:

```
$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl --connect-timeout 1 swarm-n02-oam
curl: (28) Connection timeout after 1001 ms

$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>

$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>

$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl --connect-timeout 1 swarm-n02-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>
```

   ⇒ Netrukus po pirmo mazgo klasterinės „kojos“ atjungimo antrame mazge įvyko trūktelėjimas. ⇒ Antro mazgo atsakymuose teliko tik du skirtingi konteinerių / virtualių hostų ID.

4. Tas pats ir su užklausomis į trečią mazgą:

```
$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 84fe7b5b47b1</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>
```

5. Tačiau pirmas mazgas gražina jau **tris** skirtingus, bet jau truputį kitokius ID:

```
$ curl --connect-timeout 1 swarm-n01-oam
<h1>This request was processed by host: 070aa3e17d4a</h1>

$ curl --connect-timeout 1 swarm-n01-oam
<h1>This request was processed by host: fce23fdeab52</h1>

$ curl --connect-timeout 1 swarm-n01-oam
<h1>This request was processed by host: fda749b30050</h1>

$ curl --connect-timeout 1 swarm-n01-oam
<h1>This request was processed by host: 070aa3e17d4a</h1>
```

6. Tikrinu serviso replikas pagal menedžerį:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                        IMAGE                                NODE        DESIRED STATE
CURRENT STATE                     ERROR                         PORTS
j7pgc2beau6m   kursinis-web-service.1      katacoda/docker-http-server:latest   swarm-n01   Running
Running 5 minutes ago
ypinuj5i68bo   kursinis-web-service.2      katacoda/docker-http-server:latest   swarm-n01   Running
Running 2 minutes ago
1ygslm1t3iwn   kursinis-web-service.4      katacoda/docker-http-server:latest   swarm-n01   Running
Running 2 minutes ago
```

⇒ Panašu, kad *Manager* trūko dviejų serviso replikų, ir jis jas susikūrė savo mazge.

Tik liūdna, kad jei nodas teturėtų vienintelį tinklo interfeisą, jis nebebūtų niekaip pasiekiamas.
Ir produkcijoje jis tiesiog neveiktų, nors jam „atrodytų“, kad jis veikia.

7. Tikrinu atskirus konteinerius pirmame mazge:

```
$ ssh swarm-n01-oam sudo docker ps
CONTAINER ID   IMAGE                               COMMAND   CREATED                  STATUS
PORTS     NAMES
fda749b30050   katacoda/docker-http-server:latest   "/app"    Less than a second ago   Up 3 minutes
80/tcp    kursinis-web-service.1.j7pgc2beau6m7m59wc49parbu
070aa3e17d4a   katacoda/docker-http-server:latest   "/app"    30 seconds ago           Up 23 seconds
80/tcp    kursinis-web-service.2.ypinuj5i68bo9u9tvyrm08l5l
fce23fdeab52   katacoda/docker-http-server:latest   "/app"    30 seconds ago           Up 26 seconds
80/tcp    kursinis-web-service.4.1ygslm1t3iwnmykv0rl8au9ww
```

8. Tikrinu atskirus konteinerius antrame mazge:

```
$ ssh swarm-n02-oam sudo docker ps
CONTAINER ID   IMAGE                               COMMAND   CREATED          STATUS          PORTS
NAMES
84fe7b5b47b1   katacoda/docker-http-server:latest   "/app"    42 minutes ago   Up 42 minutes   80/tcp
kursinis-web-service.2.thiskoiwddq2w5h561v789pzd
```

9. Tikrinu atskirus konteinerius trečiame mazge:

```
$ ssh swarm-n03-oam sudo docker ps
CONTAINER ID   IMAGE                               COMMAND   CREATED             STATUS             PORTS
NAMES
1bc7dedde9b6   katacoda/docker-http-server:latest   "/app"    About an hour ago   Up About an hour
80/tcp    kursinis-web-service.4.dkz7wktnswg0ajzc5v58xwytc
```

⇒ Panašu, kad klasteris pateko į *Split-brain* būseną:

- Antrame ir trečiame mazguose veikia po vieną konteinerį (kaip ir buvo iki splito).
  Kadangi juose neveikia menedžeris, jie stengiasi išlaikyti būseną.
  Jie tiesiog aptarnauja užklausas jas tarpusavyje balansuodami.
- Gi pirmame mazge susikūrė po dvi kopijas trūkstamų serviso replikų
  (likusių antrame ir trečiame mazguose, ir dabar pirmam nebematomų).
  Ir jis **irgi** atsako į užklausas, bet nebepriklausomai nuo veiklos antrame ir trečiame mazguose.
- T. y. iš esmės gavome du klasterius:
  1. `swarm-n01` su trimis replikomis viename mazge.
  2. `swarm-n02` + `swarm-n03` su dviem replikomis (po vieną kiekviename mazge).

10. Grąžinu virtualų tinklo kabelį į vietą:

```
$ VBoxManage controlvm "swarm-n01" setlinkstate2 on
```

11. Tikrinu konteinerius trečiame mazge:

```
$ ssh swarm-n03-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED             STATUS             PORTS
NAMES
1bc7dedde9b6   katacoda/docker-http-server:latest   "/app"    About an hour ago   Up About an hour
80/tcp    kursinis-web-service.4.dkz7wktnswg0ajzc5v58xwytc
```

    ⇒ Vis dar veikia.

12. Tikrinu

```
osboxes@swarm-n01:~$ dmesg -T | tail
[Wed Feb  9 12:16:15 2022] br0: port 4(veth2) entered forwarding state
[Wed Feb  9 12:16:15 2022] eth1: renamed from veth5b1e10c
[Wed Feb  9 12:16:15 2022] IPv6: ADDRCONF(NETDEV_CHANGE): vethffa8387: link becomes ready
[Wed Feb  9 12:16:15 2022] docker_gwbridge: port 4(vethffa8387) entered blocking state
[Wed Feb  9 12:16:15 2022] docker_gwbridge: port 4(vethffa8387) entered forwarding state
[Wed Feb  9 12:16:15 2022] eth1: renamed from vetha1c4f83
[Wed Feb  9 12:16:15 2022] IPv6: ADDRCONF(NETDEV_CHANGE): veth36d686a: link becomes ready
[Wed Feb  9 12:16:15 2022] docker_gwbridge: port 3(veth36d686a) entered blocking state
[Wed Feb  9 12:16:15 2022] docker_gwbridge: port 3(veth36d686a) entered forwarding state
[Wed Feb  9 12:17:40 2022] e1000: enp0s8 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX

osboxes@swarm-n01:~$ ethtool enp0s8
Settings for enp0s8:
        Supported ports: [ TP ]
        Supported link modes:   10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Supported pause frame use: No
        Supports auto-negotiation: Yes
        Supported FEC modes: Not reported
        Advertised link modes:  10baseT/Half 10baseT/Full
                                100baseT/Half 100baseT/Full
                                1000baseT/Full
        Advertised pause frame use: No
        Advertised auto-negotiation: Yes
        Advertised FEC modes: Not reported
        Speed: 1000Mb/s
        Duplex: Full
        Port: Twisted Pair
        PHYAD: 0
        Transceiver: internal
        Auto-negotiation: on
        MDI-X: off (auto)
Cannot get wake-on-lan settings: Operation not permitted
        Current message level: 0x00000007 (7)
                               drv probe link
        Link detected: yes
```

⇒ Virtualus tinklo kabelis tikrai vėl prijungtas.

13. Tikrinu mazgų būsenas ir serviso replikas:

```
$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12

$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                          IMAGE                                NODE        DESIRED STATE
CURRENT STATE                    ERROR                         PORTS
j7pgc2beau6m   kursinis-web-service.1       katacoda/docker-http-server:latest   swarm-n01   Running
Running 5 minutes ago
ypinuj5i68bo   kursinis-web-service.2       katacoda/docker-http-server:latest   swarm-n01   Running
Running 2 minutes ago
1ygslm1t3iwn   kursinis-web-service.4       katacoda/docker-http-server:latest   swarm-n01   Running
Running 2 minutes ago
```

⇒ Klasteryje vėl trys mazgai. Visos trys replikos veikia tik pirmame mazge.

14. Tikrinu konteinerius kituose individualiai. Pirmame mazge:

```
$ ssh swarm-n01-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED                  STATUS
PORTS     NAMES
fda749b30050   katacoda/docker-http-server:latest   "/app"    Less than a second ago   Up 3 minutes
80/tcp    kursinis-web-service.1.j7pgc2beau6m7m59wc49parbu
070aa3e17d4a   katacoda/docker-http-server:latest   "/app"    30 seconds ago           Up 23 seconds
80/tcp    kursinis-web-service.2.ypinuj5i68bo9u9tvyrm08l5l
fce23fdeab52   katacoda/docker-http-server:latest   "/app"    30 seconds ago           Up 26 seconds
80/tcp    kursinis-web-service.4.1ygslm1t3iwnmykv0rl8au9ww
```

15. Antrame mazge:

```
$ ssh swarm-n02-oam sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

16. Trečiame mazge:

```
$ ssh swarm-n03-oam sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

    ⇒ Kai tik pirmas mazgas „pamatė“ antrąjį ir trečiąjį, iškart išjungė perteklines replikas juose. *Split-brain* būsena išnyko.

17. Dėl visa ko tikrinu ID, gražinamus užklausose į trečią mazgą:

```
$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 070aa3e17d4a</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: fce23fdeab52</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: fda749b30050</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 070aa3e17d4a</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: fce23fdeab52</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: fda749b30050</h1>

$ curl --connect-timeout 1 swarm-n03-oam
<h1>This request was processed by host: 070aa3e17d4a</h1>
```

    ⇒ ID matyti trys skirtingi, ir jie atitinka `swarm-n01` konteinerius.

18. Taip keičiasi mazgų būsenos atjungus interfeisą:

```
$ VBoxManage controlvm "swarm-n01" setlinkstate2 off

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Down      Active                          20.10.12
```

19. O taip keičiasi interfeisą vėl prijungus:

```
$ VBoxManage controlvm "swarm-n01" setlinkstate2 on

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Down      Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Down      Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Down      Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Down      Active                          20.10.12

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12
```

Tolygus replikų paskirstymo atstatymas klasteryje

1. Atstatau tinklo ryšį:

```
$ VBoxManage controlvm "swarm-n01" setlinkstate2 on
```

2. … ir sumažinu servisą iki dviejų replikų:

```
$ ssh swarm-n01-oam sudo docker service scale kursinis-web-service=2
kursinis-web-service scaled to 2
overall progress: 0 out of 2 tasks
1/2:
2/2:
overall progress: 3 out of 2 tasks
overall progress: 2 out of 2 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 4 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 4 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 4 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 4 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 4 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 3 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 3 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 3 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 3 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 2 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 2 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 2 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 2 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 2 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 1 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 1 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 1 seconds to verify that tasks are stable...
overall progress: 2 out of 2 tasks
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Service converged
```

3. Patikrinu konteinerius atskiruose mazguose:

```
$ for NODE in swarm-n0{1..3}-oam; do echo On $NODE:; ssh $NODE sudo docker ps; echo; done
On swarm-n01-oam:
CONTAINER ID   IMAGE                                COMMAND   CREATED          STATUS          PORTS
NAMES
9c2d26cbff9e   katacoda/docker-http-server:latest   "/app"    42 minutes ago   Up 42 minutes   80/tcp
kursinis-web-service.1.vekfptu7x3egid5me161x4gbj

On swarm-n02-oam:
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

On swarm-n03-oam:
CONTAINER ID   IMAGE                                COMMAND   CREATED              STATUS
PORTS     NAMES
742f620f02d8   katacoda/docker-http-server:latest   "/app"    About a minute ago   Up About a minute
80/tcp    kursinis-web-service.6.h6ofhnw567zm36xdrgk0wbae6
```

4. Padidinu servisą iki trijų replikų:

```
$ ssh swarm-n01-oam sudo docker service scale kursinis-web-service=3
kursinis-web-service scaled to 3
overall progress: 0 out of 3 tasks
1/3:
2/3:
3/3:
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 2 out of 3 tasks
overall progress: 3 out of 3 tasks
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 5 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 4 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 3 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 2 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Waiting 1 seconds to verify that tasks are stable...
verify: Service converged
```

5. Tikrinu, servisui skirtos trys replikos:

```
$ ssh swarm-n01-oam sudo docker service ls
ID             NAME                   MODE         REPLICAS   IMAGE                                PORTS
p7imsxwi9mid   kursinis-web-service   replicated   3/3        katacoda/docker-http-server:latest   *:80-
>80/tcp
```

6. Vėl veikia po vieną konteinerį (repliką) kiekviename mazge:

```
$ for NODE in swarm-n0{1..3}-oam; do echo On $NODE:; ssh $NODE sudo docker ps; echo; done
On swarm-n01-oam:
CONTAINER ID   IMAGE                                COMMAND   CREATED          STATUS          PORTS
NAMES
9c2d26cbff9e   katacoda/docker-http-server:latest   "/app"    42 minutes ago   Up 42 minutes   80/tcp
kursinis-web-service.1.vekfptu7x3egid5me161x4gbj

On swarm-n02-oam:
CONTAINER ID   IMAGE                                COMMAND   CREATED          STATUS         PORTS
NAMES
75e5c7ff94cc   katacoda/docker-http-server:latest   "/app"    10 seconds ago   Up 8 seconds   80/tcp
kursinis-web-service.2.gmq2jit794eofojcuwlfwda3z

On swarm-n03-oam:
CONTAINER ID   IMAGE                                COMMAND   CREATED         STATUS              PORTS
NAMES
742f620f02d8   katacoda/docker-http-server:latest   "/app"    2 minutes ago   Up About a minute   80/tcp
kursinis-web-service.6.h6ofhnw567zm36xdrgk0wbae6
```

Kito klasterio elemento gedimas ir jo įtaka

1. Dėl visa ko pasitikrinu Docker pasistemės vidinių tinklų konfigūraciją:

```
$ ssh swarm-n01-oam sudo docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
55432975850d   bridge            bridge    local
d5c8e496a396   docker_gwbridge   bridge    local
0cde9da9f77d   host              host      local
vpyp1hp7w63i   ingress           overlay   swarm
7613a3238dce   none              null      local
```

2. Vėl turime servisą, tolygiai pasiskirsčiusį klasteryje:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                      IMAGE                                NODE        DESIRED STATE
CURRENT STATE                ERROR                       PORTS
vekfptu7x3eg   kursinis-web-service.1   katacoda/docker-http-server:latest   swarm-n01   Running
Running 4 hours ago
xrbcof1qu7n8   kursinis-web-service.2   katacoda/docker-http-server:latest   swarm-n02   Running
Running less than a second ago
dkz7wktnswg0   kursinis-web-service.4   katacoda/docker-http-server:latest   swarm-n03   Running
Running 10 minutes ago
```

3. Šį sykį pilnai išjungiu antrąjį mazgą:

```
$ VBoxManage controlvm "swarm-n02" poweroff
0%...10%...20%...30%...40%...50%...60%...70%...80%...90%...100%
```

4. Po sekundės klasteris gedimo dar nėra aptikęs:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                         IMAGE                                  NODE        DESIRED STATE
CURRENT STATE                    ERROR                         PORTS
vekfptu7x3eg   kursinis-web-service.1       katacoda/docker-http-server:latest   swarm-n01   Running
Running 4 hours ago
xrbcof1qu7n8   kursinis-web-service.2       katacoda/docker-http-server:latest   swarm-n02   Running
Running less than a second ago
dkz7wktnswg0   kursinis-web-service.4       katacoda/docker-http-server:latest   swarm-n03   Running
Running 14 minutes ago
```

5. Tačiau po maždaug dviejų sekundžių jau aptinka, kad mazgas `swarm-n02` nebeatsiliepia:

```
$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Down      Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12
```

6. ... ir perkuria trūkstamą repliką mazge `swarm-n01`:

```
$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service | grep -v Shut
ID             NAME                         IMAGE                                  NODE        DESIRED STATE
CURRENT STATE                    ERROR                         PORTS
vekfptu7x3eg   kursinis-web-service.1       katacoda/docker-http-server:latest   swarm-n01   Running
Running 4 hours ago
oweiigoueoww   kursinis-web-service.2       katacoda/docker-http-server:latest   swarm-n01   Running
Running 3 seconds ago
dkz7wktnswg0   kursinis-web-service.4       katacoda/docker-http-server:latest   swarm-n03   Running
Running 14 minutes ago

$ ssh swarm-n01-oam sudo docker service ps kursinis-web-service
ID             NAME                         IMAGE                                  NODE        DESIRED STATE
CURRENT STATE                    ERROR                         PORTS
vekfptu7x3eg   kursinis-web-service.1       katacoda/docker-http-server:latest   swarm-n01   Running
Running 4 hours ago
oweiigoueoww   kursinis-web-service.2       katacoda/docker-http-server:latest   swarm-n01   Running
Running about a minute ago
xrbcof1qu7n8    \_ kursinis-web-service.2   katacoda/docker-http-server:latest   swarm-n02   Shutdown
Running less than a second ago
gmq2jit794eo    \_ kursinis-web-service.2   katacoda/docker-http-server:latest   swarm-n02   Shutdown
Shutdown about an hour ago
4ekln6k23p0b   kursinis-web-service.3       katacoda/docker-http-server:latest   swarm-n03   Shutdown
Shutdown 25 minutes ago
dkz7wktnswg0   kursinis-web-service.4       katacoda/docker-http-server:latest   swarm-n03   Running
Running 16 minutes ago
q6xeb0896glj    \_ kursinis-web-service.4   katacoda/docker-http-server:latest   swarm-n02   Shutdown
Failed less than a second ago    "task: non-zero exit (255)"
```

7. Tuo tarpu trečias mazgas nukreipia užklausas į visas 3 replikas:

```
$ curl swarm-n03-oam
<h1>This request was processed by host: f88eb8f1a09d</h1>

$ curl swarm-n03-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl swarm-n03-oam
<h1>This request was processed by host: f88eb8f1a09d</h1>

$ curl swarm-n03-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl swarm-n03-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

8. Taip pat nukreipia ir pirmas mazgas:

```
$ curl swarm-n01-oam
<h1>This request was processed by host: f88eb8f1a09d</h1>

$ curl swarm-n01-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl swarm-n01-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>

$ curl swarm-n01-oam
<h1>This request was processed by host: f88eb8f1a09d</h1>

$ curl swarm-n01-oam
<h1>This request was processed by host: 1bc7dedde9b6</h1>

$ curl swarm-n01-oam
<h1>This request was processed by host: 9c2d26cbff9e</h1>
```

9. Antras mazgas neatsiliepia, žinoma (nes išjungtas):

```
$ curl swarm-n02-oam
curl: (28) Failed to connect to swarm-n02-oam port 80 after 21011 ms: Connection timed out
```

10. Įjungiu antrą mazgą ir sulaukiu, kol jis grįš į klasterį:

```
$ VBoxManage startvm "swarm-n02"
Waiting for VM "swarm-n02" to power on...
VM "swarm-n02" has been successfully started.

$ ssh swarm-n01-oam sudo docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY   MANAGER STATUS   ENGINE VERSION
l6wnnbsgv2th6nq05e9j02srj *   swarm-n01   Ready     Active         Leader           20.10.12
a50ddvlva40nzzvtxu1hpsus7     swarm-n02   Ready     Active                          20.10.12
6qeivl6aatpwxb50vi8hpr4bh     swarm-n03   Ready     Active                          20.10.12
```

11. Deja, nepatikrinau, kaip pasiskirstė serviso replikos.

    Bet pagal šiuos du konteinerių sąrašus panašu, kad dvi veikė `swarm-n01`, o viena liko `swarm-03`:

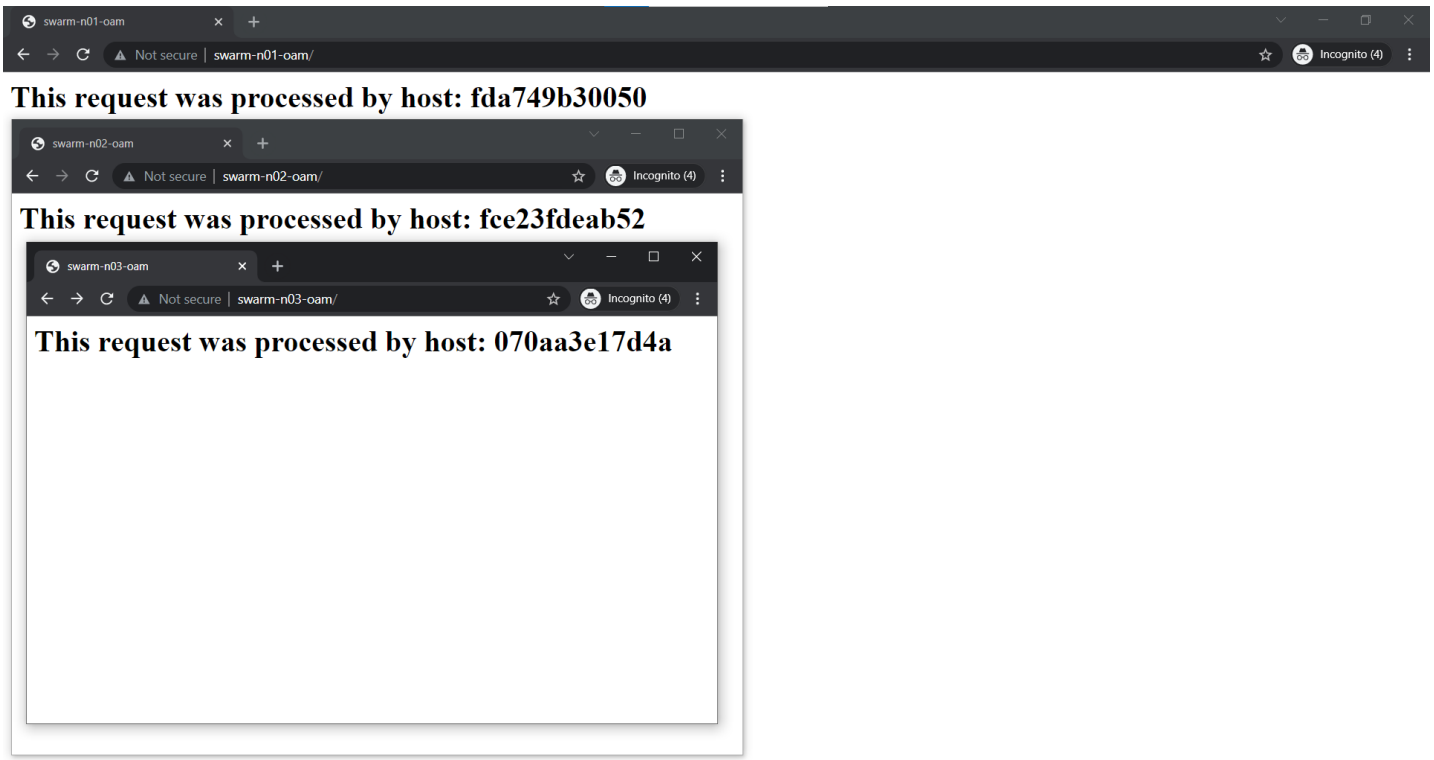
```
$ ssh swarm-n02-oam sudo docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES

$ ssh swarm-n03-oam sudo docker ps
CONTAINER ID   IMAGE                                COMMAND   CREATED          STATUS          PORTS
NAMES
1bc7dedde9b6   katacoda/docker-http-server:latest   "/app"    38 minutes ago   Up 38 minutes   80/tcp
kursinis-web-service.4.dkz7wktnswg0ajzc5v58xwytc
```

Web-serviso pasiekiamumas įvairiuose etapuose

Serviso pasiekiamumą įvairiuose etapuose jau atvaizdavau CLI būdu. Kai kada, kai *Manager* neveikia, tai padaryti nėra elegantiška.

Dėl visa ko įtraukiu trijų užklausų rezultatus naršyklėje:



3. Rezultatų apibendrinimas

Susikonstravau VM infrastruktūrą VirtuaBox hipervizoriaus (Type II) pagrindu. Kiekvienai VM skyriau po tris tinklo interfeisus:

1. prisijungimui prie interneto (atnaujinimų siuntimams ir kt.)
2. aplikacijai / klasterio mazgų ryšiui;
3. OS valdymui (OAM).

Sukūriau tris VM, jose pasinaudojau *Docker Swarm Mode* technologija ir startavau trijų mazgų klasterį:

1. *Manager + Worker*;

2. *Worker*;
3. *Worker*.

Klasteryje *Docker* konteinerių pagrindu paleidau savo pasirinktą *Web*-servisą `katacoda/docker-http-server`. Patikrinau jį iš savo kompiuterio: pasiekiamas.

Sukūriau tris serviso replikas. Patikrinau ir užfiksavau jų pasiskirstymą klasteryje.

Imitavau klasterio elemento gedimą: atjungiau pirmojo mazgo `swarm-n01` klasterinį tinklo interfeisą.

Manager nustojo matyti likusius du mazgus ir perkūrė du jų konteinerius pas save. Bėda, kad jis pats būtų tapęs nepasiekiamu produkciniam tinklui (NLB ar maršrutizatoriui). Tačiau per OAM interfeisą visi trys konteineriai buvo pasiekiami.

Tuo tarpu mazgai `swarm-n02` ir `swarm-n03` iškart nustojo atsiliepti į užklausas `80/TCP` portu iš viso, nors jų klasteriniai interfeisai ir tebeveikė.

Po <20 s. jų atsakymai į užklausas atsistatė — jie jas pradėjo balansuoti tarpusavyje ir grąžindavo jau du skirtingus *Host-id*.

Iš esmės, situacija mano vertinimu atitinka klasterinį *Split-brain* scenarijų, kai abi klasterio dalys nusprendžia, kad kita pusė nebeveikia, ir bando veikti abi nepriklausomai.

- ⇒ Darau išvadą, kad klasteriui paskyrus tiek nedaug mazgų, vertėtų padidinti ne tik *Worker* skaičių, bet ir *Manager* skaičių.

  Priešingu atveju įmanomas pavojus duomenų vientisumui, kai dvi grupės vienu metu keis tuos pačius duomenis, bet kiekviena laikys, kad keičia tik ji pati, tik viena grupė.

Toliau atstačiau tinklo veikimą, ir stebėjau konteinerių būsenas tiek *Worker* mazguose, tiek *Manager* mazge. Netrukus jie pradėjo atsakymuose grąžinti naujus *Host-id*.

Patikrinus pasirodė, kad visi šie *Host-id* priklauso `swarm-n01` mazge veikiantiems dviems naujiems konteineriams, sukurtiems splito metu. Ir dabar šiaip paslaugai visos trys replikos veikė būtent šiame mazge. Konteineriai *Worker* mazguose išsijungė netrukus po *Manager* tinklo atstatymo.

Po šito paskirsčiau replikas vėl po lygiai — po vieną kiekvienam mazgui: `… scale kursinis-web-service=1` ir `… scale kursinis-web-service=3`.

Ir kai tuo tarpu pilnai išjungiau antrą mazgą, `swarm-n02`, jo replika buvo pakeista nauja replika pirmajame mazge, `swarm-n01`.

Į užklausas abu tebeveikiantys mazgai atsakydavo sėkmingai (`swarm-n01` ir `swarm-n03`).

Mazgą `swarm-n02` vėl įjungus, jis pats sugrįžo į klasterį, tačiau veikiančios replikos pasiliko savo dabartiniuose mazguose (dvi `swarm-n01` ir viena `swarm-n03`).

O štai užklausos į servisą pradėjo veikti jau ir per antrąjį mazgą — sugrįžęs į klasterį jis įsitraukė į *Routing mesh* ir *Load-balancing* mechanizmą.

- ⇒ Jei gedimas įvyksta *Worker* mazge, o ne *Manager*, įtaka paslaugai beveik nejuntama.

  Paslaugos replikų skaičius atstatomas (sukuriamos trūkstamosios) ilgiausiai po ~ 5 s.

4. Naudota literatūra

[1] `osboxes.org`, OSboxes > VirtualBox Images > Ubuntu > Ubuntu 20.04.3 Focal Fossa (https://www.osboxes.org/ubuntu/#ubuntu-20-04-3-info)

[2] `docker.com`, Docker docs > Run your app in production > Getting started with swarm mode (tutorial) (https://docs.docker.com/engine/swarm/swarm-tutorial/)

[3] `katacoda.com`, O'Reilly | Katacoda | Learn Docker Orchestration / Swarm Mode using Interactive Browser-Based Scenarios (https://www.katacoda.com/courses/docker-orchestration)

[4] `katacoda.com`, O'Reilly | Katacoda | Getting Started With Swarm Mode (https://www.katacoda.com/courses/docker-orchestration/getting-started-with-swarm-mode)

[5] `katacoda.com`, O'Reilly | Katacoda | Create Overlay Network (https://www.katacoda.com/courses/docker-orchestration/create-overlay-networks)

[6] `katacoda.com`, O'Reilly | Katacoda | Load Balance and Service Discover in Swarm Mode (https://www.katacoda.com/courses/docker-orchestration/load-balance-service-discovery-swarm-mode)

[7] `katacoda.com`, O'Reilly | Katacoda | Apply Rolling Updates Across Swarm Cluster (https://www.katacoda.com/courses/docker-orchestration/rolling-updates-services-swarm-cluster)

[8] `katacoda.com`, O'Reilly | Katacoda | Add Healthcheck for Containers (https://www.katacoda.com/courses/docker-orchestration/healthcheck)

[9] `katacoda.com`, O'Reilly | Katacoda | Deploy Swarm Services with Compose v3 (https://www.katacoda.com/courses/docker-orchestration/deploy-swarm-services-with-compose)

[10] `katacoda.com`, O'Reilly | Katacoda | Enable Maintenance Mode for a Swarm Node (https://www.katacoda.com/courses/docker-orchestration/maintenance-mode-for-swarm)

[11] `jayway.com`, Simple Clustering with Docker Swarm and Nginx (https://blog.jayway.com/2015/11/25/simple-clustering-with-docker-swarm-and-nginx/)