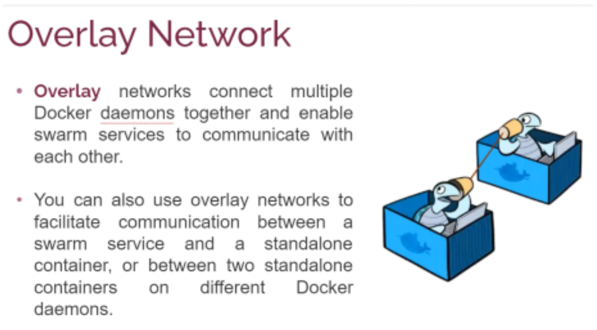
7- Docker Swarm -NEtworking

Thursday, 23 September 2021 9.37

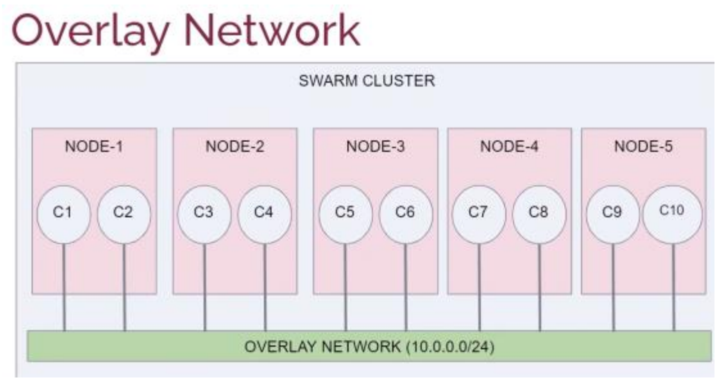


Overlay Network

- **Overlay** networks connect multiple Docker daemons together and enable swarm services to communicate with each other.
- You can also use overlay networks to facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker daemons.

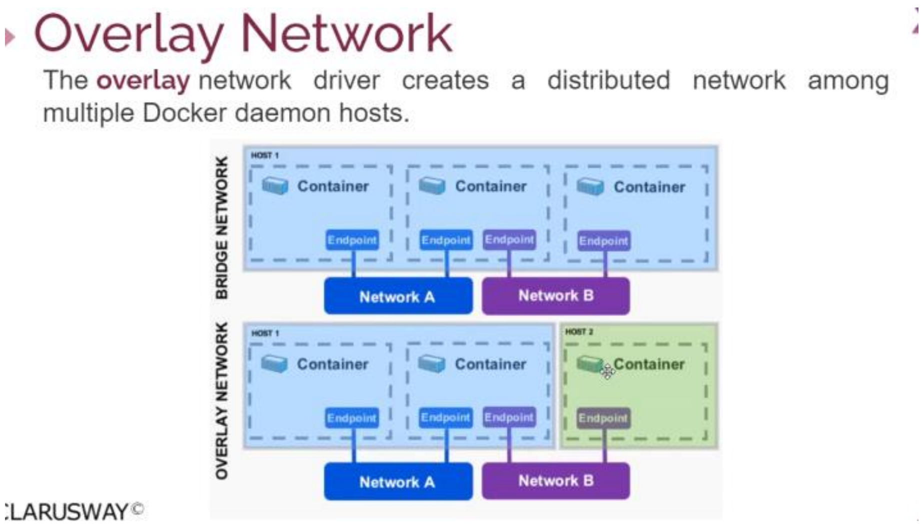
Bir cok dockers i bir birine baglar
Ayrica bagimsiz olarak kurulan natlar ar ise onlar ile de iletisim kurmaya yariyor

Overlay Network



`docker container`

Resomada de göruldugu gibi tum swarm icind farkli konteynirlar bir birleri ile silsetidisime dgecebiliyor



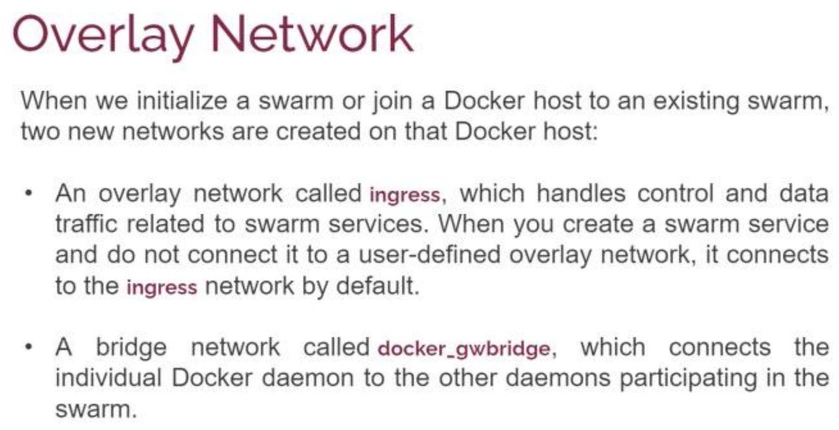
Overlay Network

The **overlay** network driver creates a distributed network among multiple Docker daemon hosts.

Ilk sekilde bir host ve icinde uc adet container var Bridge network de overlay network de ayni islemi ayapaniliyor aralarinda ki fark su diyebiliriz

Resmi su sekile anlayabiliriz ,Network A --Bridger Network
Ner´twork B ---Overlay Network

Ilk resimde Tek Host icin de farkli containerlarda her ikisi de con kari boir birkerine bagliyabiliyor
Ancak ikinci reismde farkli iki host var ve burda bulunan Cont lari bir birine baglayabiliyor



Overlay Network

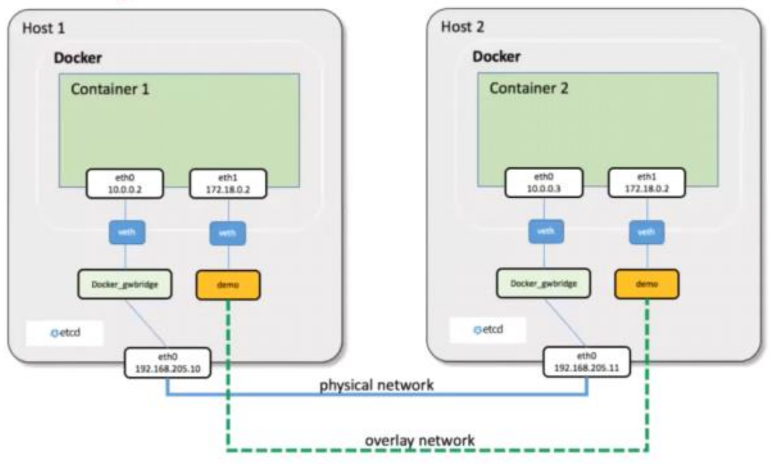
When we initialize a swarm or join a Docker host to an existing swarm, two new networks are created on that Docker host:

- An overlay network called ingress, which handles control and data traffic related to swarm services. When you create a swarm service and do not connect it to a user-defined overlay network, it connects to the ingress network by default.
- A bridge network called docker_gwbridge, which connects the individual Docker daemon to the other daemons participating in the swarm.

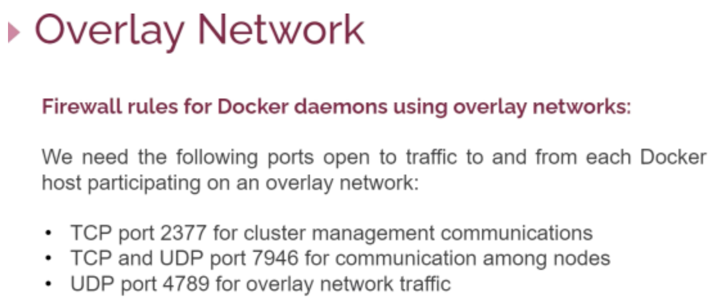
Docker Swarm a bagalandigimizda Docker bize iki yeni Network olusturuyor

- **Birii ingress ; default swarm service**
- **Docker_gwbridge; docjker host un kendisine bagli oldufu bir obje**

Overlay Network



Docker_gwbridge ; fiziksel olarak bir birlerine bagliyor diyebilirisiz ethernet kartlarini



Overlay Network

Firewall rules for Docker daemons using overlay networks:

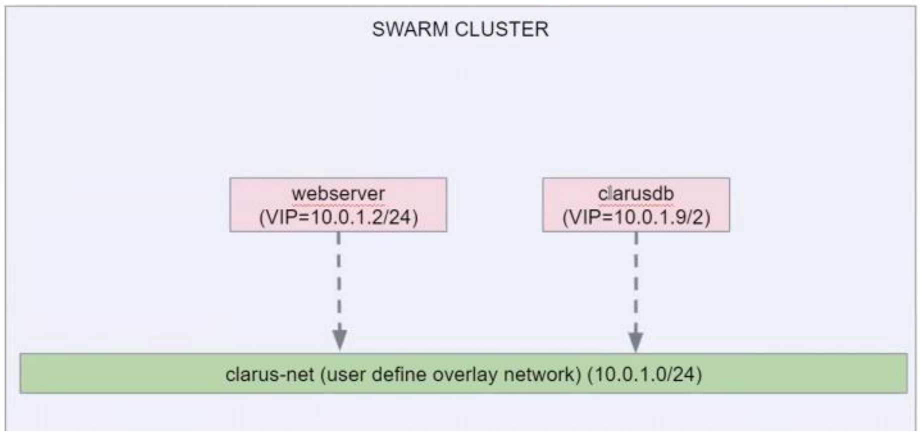
We need the following ports open to traffic to and from each Docker host participating on an overlay network:

- TCP port 2377 for cluster management communications
- TCP and UDP port 7946 for communication among nodes
- UDP port 4789 for overlay network traffic

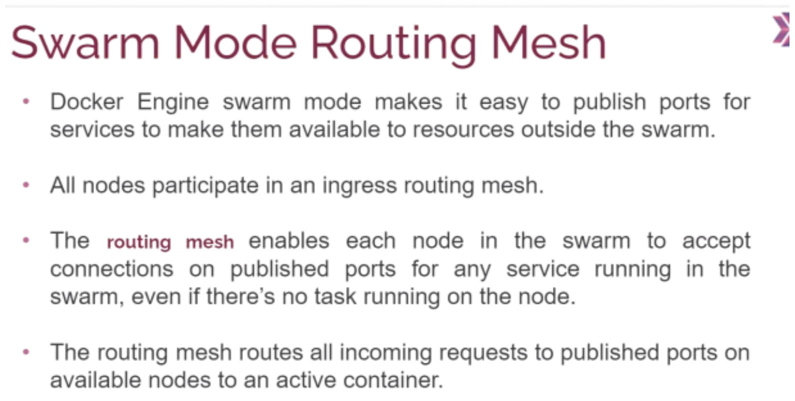
Overlay network ile birlikte bazi acilmasi gereken port lar oluyor ki bunlar

- ❖ **2377 ; manager in yaptigi isleri**
- ❖ **7946; notlari bir birine connect liyor**
- ❖ **Ve 4789 portu notlar arasi trafici sagliyor**

User-defined Overlay Network


SWARM CLUSTER
webserver (VIP=10.0.1.2/24)
clarusdb (VIP=10.0.1.9/2)
clarus-net (user define overlay network) (10.0.1.0/24)

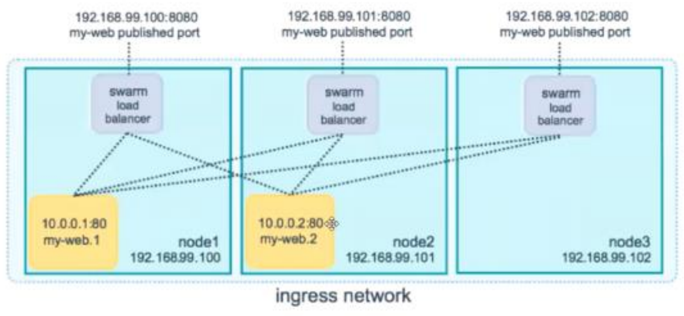
Bu da hands sonda yapacagimiz isi resmediyor



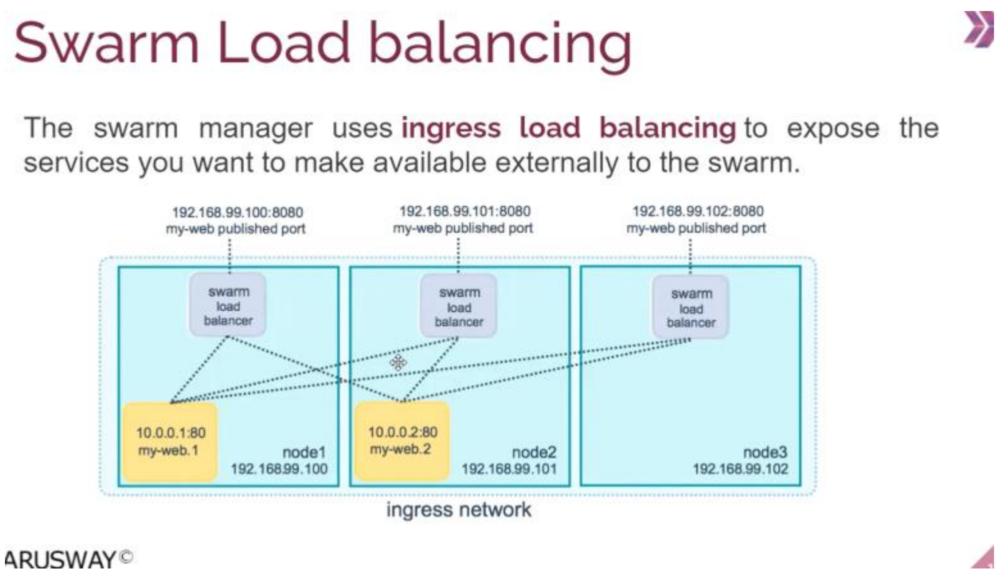
Swarm Mode Routing Mesh

- Docker Engine swarm mode makes it easy to publish ports for services to make them available to resources outside the swarm.
- All nodes participate in an ingress routing mesh.
- The **routing mesh** enables each node in the swarm to accept connections on published ports for any service running in the swarm, even if there's no task running on the node.
- The routing mesh routes all incoming requests to published ports on available nodes to an active container.

Swarm Mode Routing Mesh

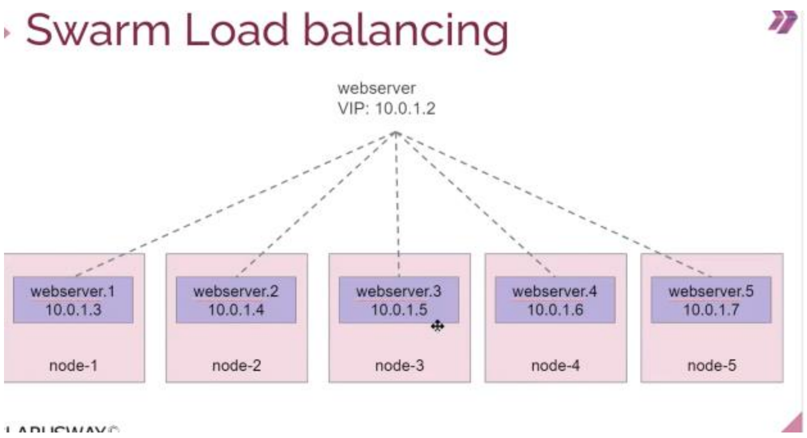


Burda toplamda 3 taane node var iki node da web application muzu var calisiyor Rouring Mesh bizim her uc port dan da bu applications a ulasmamizi sgliuyor



Swarm Load balancing

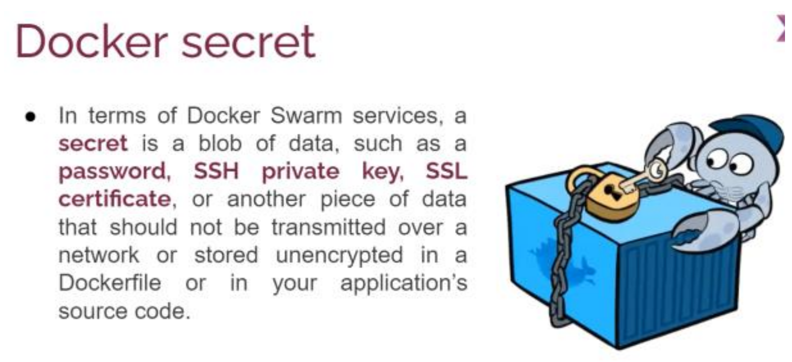
The swarm manager uses **ingress load balancing** to expose the services you want to make available externally to the swarm.



Swarm Load balancing

Gelen requestleri da her bir noda dagitiyor

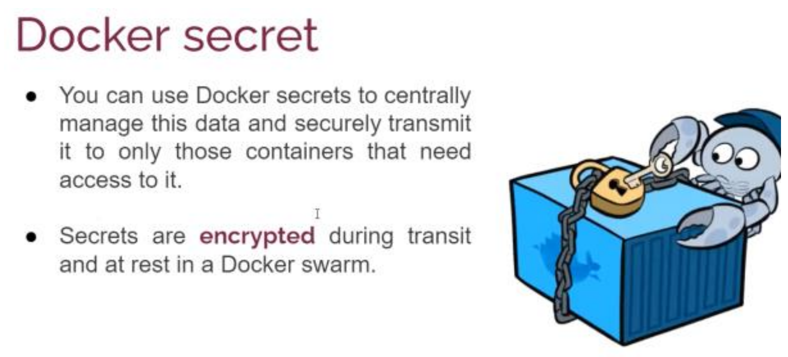
VIP ile container lara ulasabiliyor



Docker secret

- In terms of Docker Swarm services, a **secret** is a blob of data, such as a **password, SSH private key, SSL certificate**, or another piece of data that should not be transmitted over a network or stored unencrypted in a Dockerfile or in your application's source code.

Hassas bilgilerimizi sifreleyerek gönderebiliyoruz

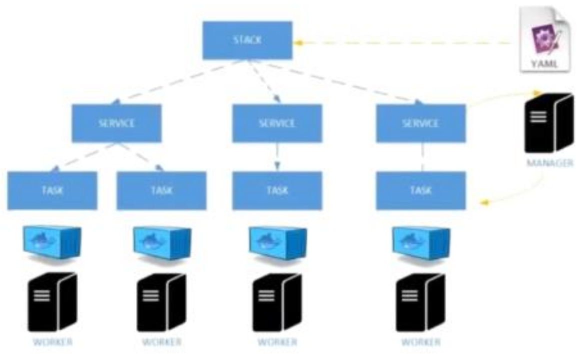


Docker secret

- You can use Docker secrets to centrally manage this data and securely transmit it to only those containers that need access to it.
- Secrets are **encrypted** during transit and at rest in a Docker swarm.

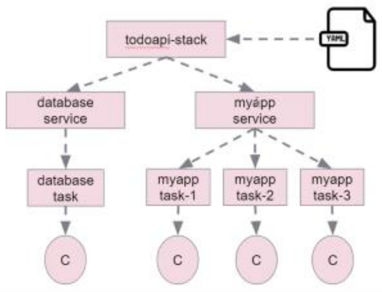
Docker stack

Docker stack is a collection of services that make up an application in a specific environment.



Docker stack

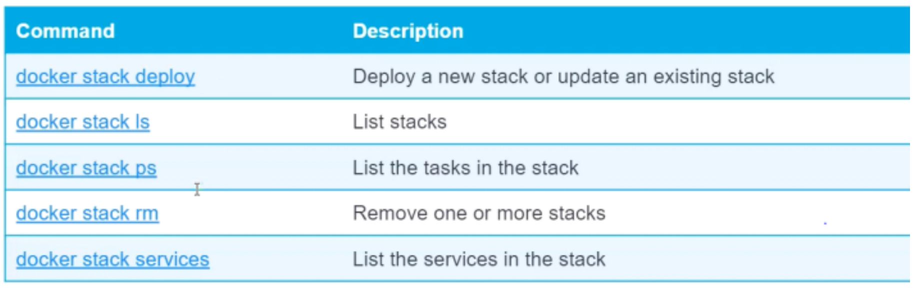
Docker stack is a collection of services that make up an application in a specific environment.



ARUSWAY©
TO REINVENT YOURSELF

Bir yaml file hazirlayark bir cok service duzenleyebiliyoruz

docker stack Commands



Command	Description
docker stack deploy	Deploy a new stack or update an existing stack
docker stack ls	List stacks
docker stack ps	List the tasks in the stack
docker stack rm	Remove one or more stacks
docker stack services	List the services in the stack

# Hands-on Docker-09 : Docker Swarm Networking, Managing Services, Secrets and Stacks
Purpose of the this hands-
on training is to give students the understanding to the Docker Swarm basic operations.

## Learning Outcomes
At the end of the this hands-on training, students will be able to;
- Explain what Docker Swarm cluster is.
- Set up a Docker Swarm cluster.
- Deploy an application as service on Docker Swarm.
- Use `overlay` network in Docker Swarm.
- Update and revert a service in Docker Swarm.

- Create and manage sensitive data with Docker Secrets.
- Create and manage Docker Stacks.

## Outline
- Part 1 - Launch Docker Machine Instances and Connect with SSH
- Part 2 - Set up a Swarm Cluster with Manager and Worker Nodes
- Part 3 - Using Overlay Network in Docker Swarm
- Part 4 - Managing Sensitive Data with Docker Secrets
- Part 5 - Managing Docker Stack

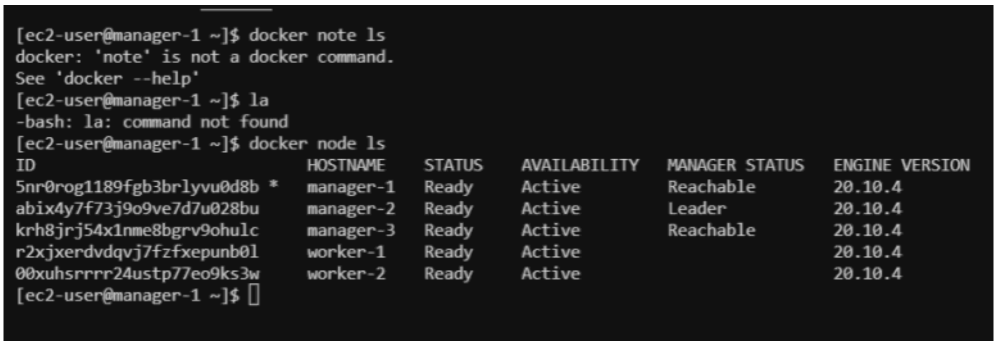
- Part 6 - Running WordPress as a Docker Stack

## Part 1 - Launch Docker Machine Instances and Connect with SSH
- Launch `five` Compose enabled Docker machines on Amazon Linux 2 with security group allowing SSH connections using the of [Clarusway Docker Swarm Cloudformation Template](./clarusway-docker-swarm-cfn-template.yml).
- Connect to your instances with SSH.
```bash
ssh -i .ssh/call-training.pem ec2-user@ec2-3-133-106-98.us-east-2.compute.amazonaws.com
```

## Part 2 - Set up a Swarm Cluster with Manager and Worker Nodes
- Prerequisites (Those prerequisites are satisfied within cloudformation template in Part 1)
  - Five EC2 instances on Amazon Linux 2 with `Docker` and `Docker Compose` installed.
  - Set these ingress rules on your EC2 security groups:
    - HTTP port 80 from 0.0.0.0\0
    - TCP port 2377 from 0.0.0.0\0
    - TCP port 8080 from 0.0.0.0\0
    - SSH port 22 from 0.0.0.0\0 (for increased security replace this with your own IP)
- Initialize `docker swarm` with Private IP and assign your first docker machine as manager:
```bash
docker swarm init
# or
docker swarm init --advertise-addr <Private IPs>
```
- Check if the `docker swarm` is active or not.
```bash
docker info
```
- Get the manager token with `docker swarm join-token manager` command.
```bash
docker swarm join-token manager
```
- Add second and third Docker Machine instances as manager nodes, by connecting with SSH and running the given command above.
```bash
docker swarm join --token <manager_token> <manager_ip>:2377
```
- Add fourth and fifth Docker Machine instances as worker nodes. (Run `docker swarm join-token worker` command to get join-token for worker, if needed)
```bash
docker swarm join --token <worker_token> <manager_ip>:2377
```
- List the connected nodes in `Swarm`.
```bash
docker node ls
`
```

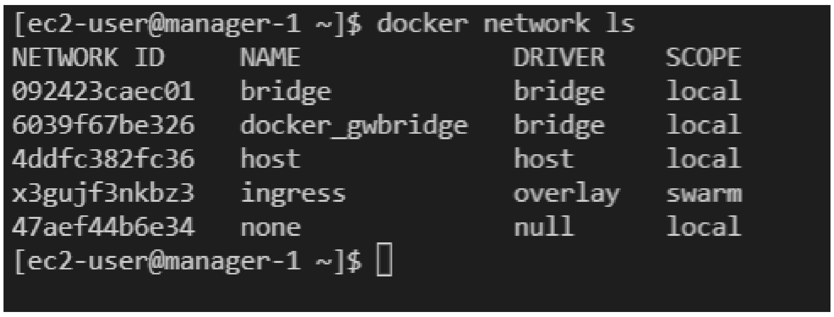
## Part 3 - Using Overlay Network in Docker Swarm
- List Docker networks and explain overlay network (ingress)

```bash
docker network ls
docker network inspect ingress
```



```
[ec2-user@manager-1 ~]$ docker note ls
docker: 'note' is not a docker command.
See 'docker --help'
[ec2-user@manager-1 ~]$ la
-bash: la: command not found
[ec2-user@manager-1 ~]$ docker node ls
ID                            HOSTNAME    STATUS    AVAILABILITY    MANAGER STATUS    ENGINE VERSION
5nr0rog1189fgb3brlyvu0d8b *   manager-1   Ready     Active          Reachable         20.10.4
abix4y7f73j9o9ve7d7u028bu     manager-2   Ready     Active          Leader            20.10.4
krh8jrj54x1nme8bgrv9ohulc     manager-3   Ready     Active          Reachable         20.10.4
r2xjxerdvdqvj7fzfxepunb0l     worker-1    Ready     Active                            20.10.4
00xuhsrrrr24ustp77eo9ks3w     worker-2    Ready     Active                            20.10.4
[ec2-user@manager-1 ~]$
```

Docker node ls #-----komutu ile daha önceki dersste hazirladigimiz 3 manager ve 2 worker i görduk



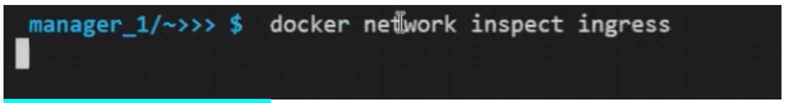
```
[ec2-user@manager-1 ~]$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
092423caec01   bridge            bridge    local
6039f67be326   docker_gwbridge   bridge    local
4ddfc382fc36   host              host      local
x3gujf3nkbz3   ingress           overlay   swarm
47aef44b6e34   none              null      local
[ec2-user@manager-1 ~]$
```

Iki tane yeni geliyor

Ingress Overlay(default) ver docker_gwbridge

Ve simdi bunu incelemeye basliyoruz

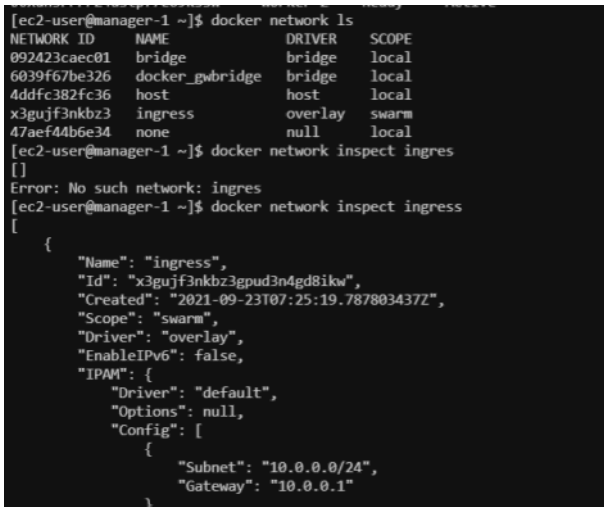
- Create a user defined overlay network.



```
manager_1/~>>> $  docker network inspect ingress
```

Ingres olani inceliyoruz

~~~
``bash
docker network create -d overlay clarus-net
```
~~~



```
[ec2-user@manager-1 ~]$ docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
092423caec01   bridge            bridge    local
6039f67be326   docker_gwbridge   bridge    local
4ddfc382fc36   host              host      local
x3gujf3nkbz3   ingress           overlay   swarm
47aef44b6e34   none              null      local
[ec2-user@manager-1 ~]$ docker network inspect ingres
[]
Error: No such network: ingres
[ec2-user@manager-1 ~]$ docker network inspect ingress
[
    {
        "Name": "ingress",
        "Id": "x3gujf3nkbz3gpud3n4gd8ikw",
        "Created": "2021-09-23T07:25:19.787803437Z",
        "Scope": "swarm",
        "Driver": "overlay",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "10.0.0.0/24",
                    "Gateway": "10.0.0.1"
```

Hadi inceleyelim bunu inspect comutu ile

- Explain user-defined overlay network (clarus-net)

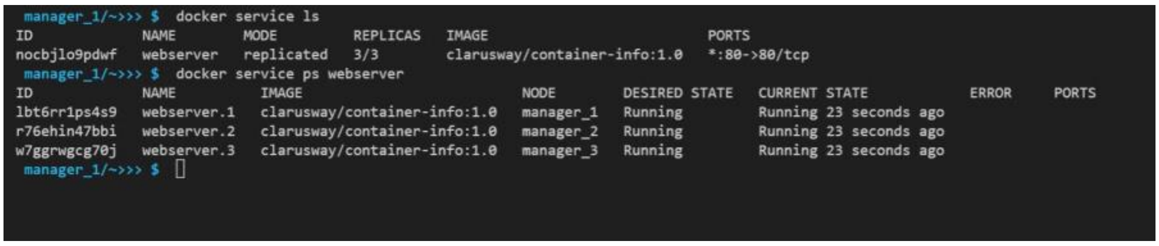
~~~bash
docker network inspect clarus-net
``

******`service olusturup network u buna baglayacagiz ****

- Create a new service with 3 replicas.

```bash

~~~

Name ile ismini yaziyoruz --network ile baglyoruz   kime bagalayacaksak onu yaziyoruz - p portumuzu yazdik replicalarimizi belirledik imajimizi da clarusway container-info ile hazirladik

- List the tasks of `webserver` service, detect the nodes which is running the task and whic
h is not.

```bash
docker service ps webserver
```

Bu komut ile de proces leri gördük.

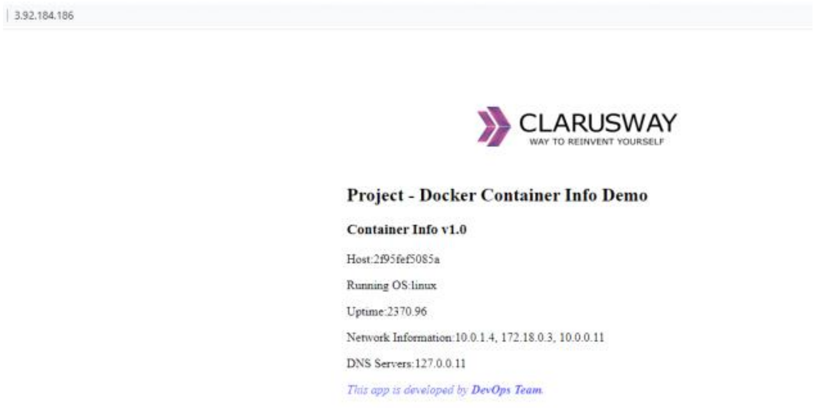


```
 manager_1/~>>> $  docker service ls
ID             NAME        MODE         REPLICAS   IMAGE                              PORTS
nocbjlo9pdwf   webserver   replicated   3/3        clarusway/container-info:1.0       *:80->80/tcp
 manager_1/~>>> $  docker service ps webserver
ID             NAME          IMAGE                          NODE        DESIRED STATE   CURRENT STATE            ERROR     PORTS
lbt6rr1ps4s9   webserver.1   clarusway/container-info:1.0   manager_1   Running         Running 23 seconds ago
r76ehin47bbi   webserver.2   clarusway/container-info:1.0   manager_2   Running         Running 23 seconds ago
w7ggrwgcg70j   webserver.3   clarusway/container-info:1.0   manager_3   Running         Running 23 seconds ago
 manager_1/~>>> $ 
```

docker service ls # --- ile servuce mizin olustugunu gördük

Ve konsolumuza gittik baglanti kurdugumuz bir ec2 nun ip ile ve 80 portu ile actik

Komnteynir actigimiz makinada gördugumuz kadari ile bir takim bilgileeri gördük sayfayi yeniledigimzde load balancerda etksi ile farkli ipler geldigini görebiliyoruz



3.92.184.186

Project - Docker Container Info Demo

Container Info v1.0

Host:2f95fef5085a

Running OS:linux

Uptime:2370.96

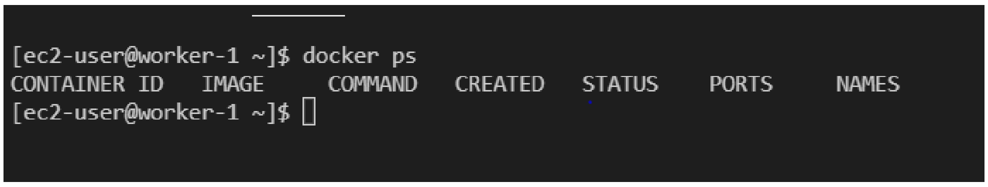
Network Information:10.0.1.4, 172.18.0.3, 10.0.0.11

DNS Servers:127.0.0.11

This app is developed by DevOps Team

Su ana kadar bir overlay network olusturdi+ukn
Bir de sservice olustiurduk

Bu service network u 3 replica yani 3 tesy ile 80 portunda actik



```
[ec2-user@worker-1 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@worker-1 ~]$ 
```

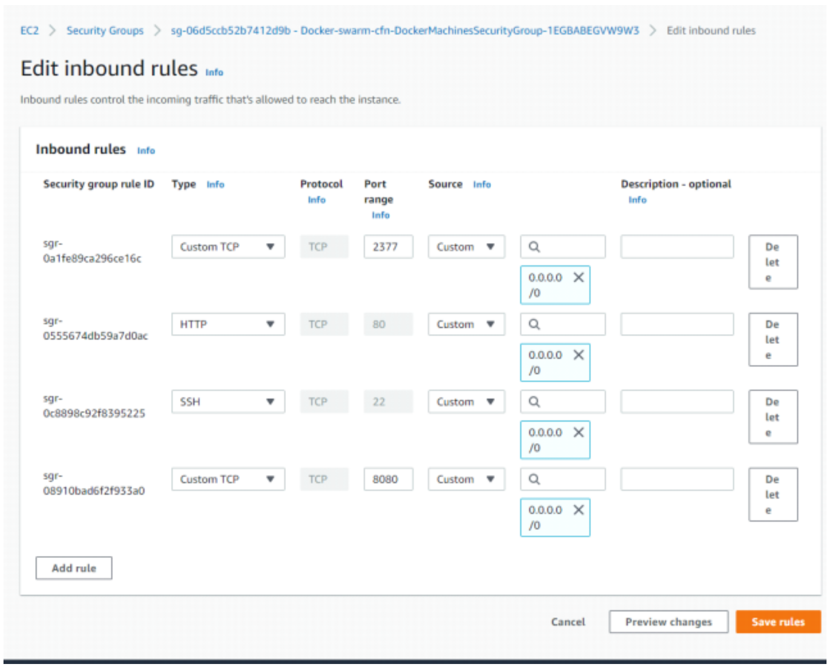
Docker 4 instance miz da container olmadigini ve calismadigini h´görebiliriz

Su sekilde bu instance in public ip ile 80 portuna baglanmaya calistugumuzda

baglanamyiyoruz

Ne yapmmaiz gerekiyor bazi portlari acmamiz gerekiyor

Instancemiza sec grb na edit inbount rule ile yeni sec grb tanimlayacagiz



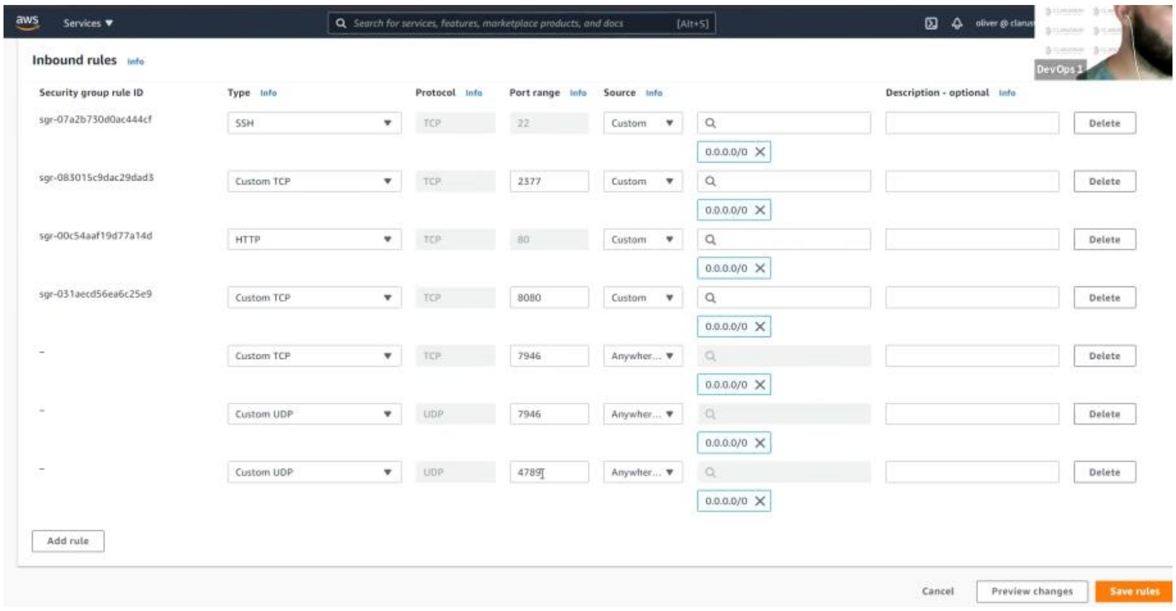
EC2 > Security Groups > sg-06d5ccb52b7412d9b - Docker-swarm-cfn-DockerMachinesSecurityGroup-1EGBABEGVW9W3 > Edit inbound rules

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-0a1fe89ca296ce16c	Custom TCP	TCP	2377	Custom	0.0.0.0/0	
sgr-0555674db59a7d0ac	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-0c8898c92f8395225	SSH	TCP	22	Custom	0.0.0.0/0	
sgr-08910bad6f2f933a0	Custom TCP	TCP	8080	Custom	0.0.0.0/0	

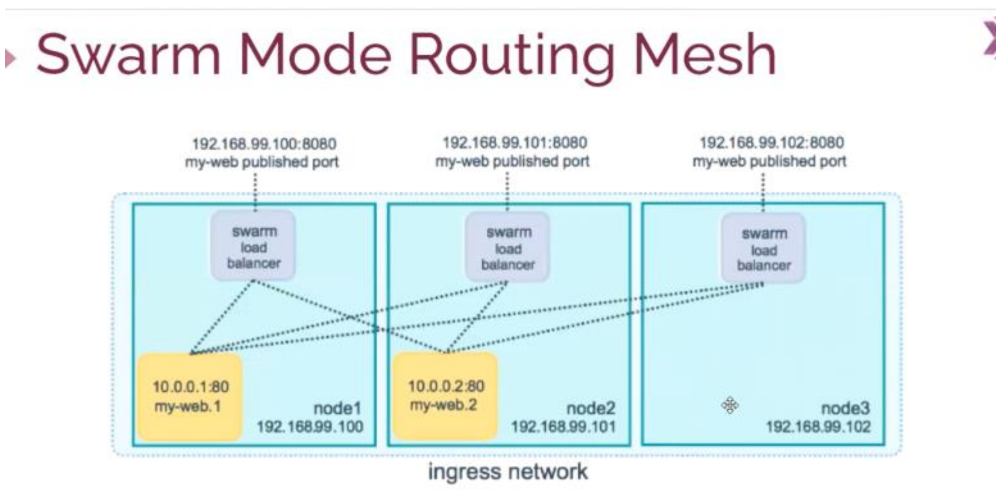


Inbound rules

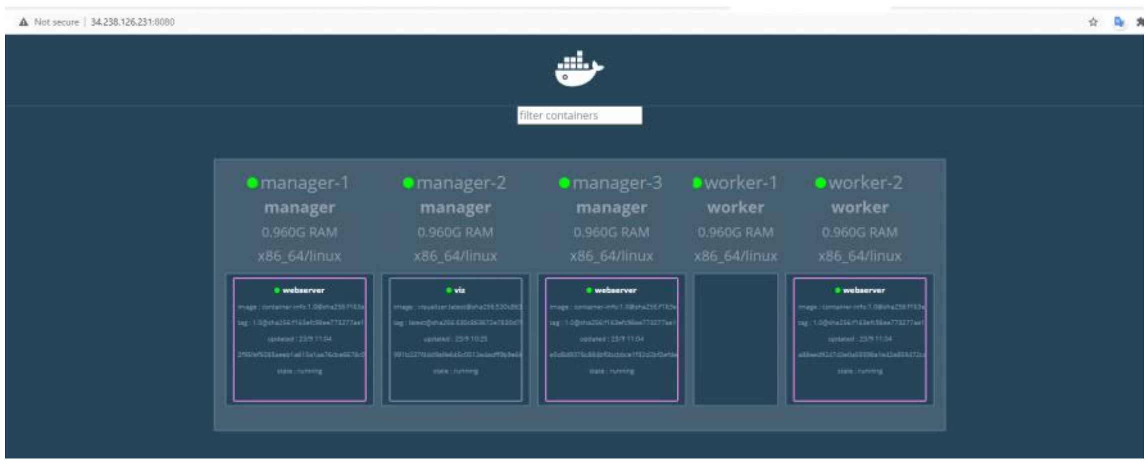
Security group rule ID	Type	Protocol	Port range	Source		Description - optional
sgr-07a2b730d0ac444cf	SSH	TCP	22	Custom	0.0.0.0/0	
sgr-083015c9dac29dad3	Custom TCP	TCP	2377	Custom	0.0.0.0/0	
sgr-00c54aaf19d77a14d	HTTP	TCP	80	Custom	0.0.0.0/0	
sgr-031aecd56ea6c25e9	Custom TCP	TCP	8080	Custom	0.0.0.0/0	
-	Custom TCP	TCP	7946	Anywher...	0.0.0.0/0	
-	Custom UDP	UDP	7946	Anywher...	0.0.0.0/0	
-	Custom UDP	UDP	4789	Anywher...	0.0.0.0/0	

Ne yaptik ;

- **7946 hem tcp hemde udp actik**
- **4789 saece udp acik**



Routing Mesh olayini yaptik hic container olmayan bir ip ile portlari acmak suretiyke baglanabildik

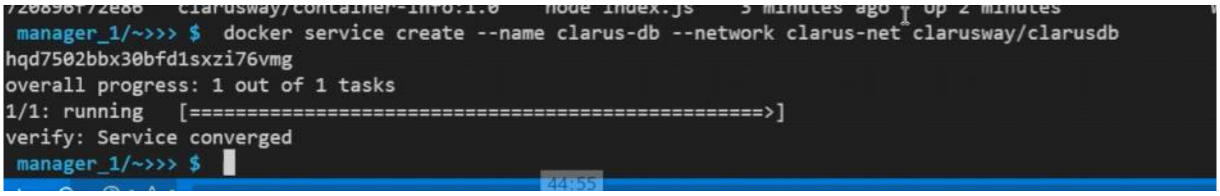


- Check the URLs of nodes that is running the task with `http://<ec2-public-hostname-of-node>` in the browser and show that the app is accessible, and explain `Container Info` on the app page. (`Host` is the name of container hosting the app, `Network Information` is giving IP addresses attached to `container` by different networks, for example; `10.0.1.3 from clarus-net`, `172.18.0.3 from docker_gwbridge`, `10.0.0.8 from ingress network`)
- Check the URLs of nodes that is not running the task with `http://<ec2-public-hostname-of-node>` in the browser and show that the app is not accessible.
- Add following rules to security group of the nodes to enable the ingress network in the swarm and explain `swarm routing mesh`. *All nodes participate in an `ingress routing mesh`. The `routing mesh` enables each node in the `swarm` to accept connections on published ports for any service running in the swarm, **even if there's no task running on the node**. The routing mesh routes all incoming requests to published ports on available nodes to an active container.* [Using swarm mode routing mesh](https://docs.docker.com/engine/swarm/ingress/#bypass-the-routing-mesh)
  - For container network discovery -> Protocol: TCP, Port: 7946, Source: security group itself
  - For container network discovery -> Protocol: UDP, Port: 7946, Source: security group itself
  - For the container ingress network -> Protocol: UDP, Port: 4789, Source: security group itself
- Check the URLs of nodes that is not running the task with `http://<ec2-public-hostname-of-node>` in the browser and show that the app is **now** accessible.

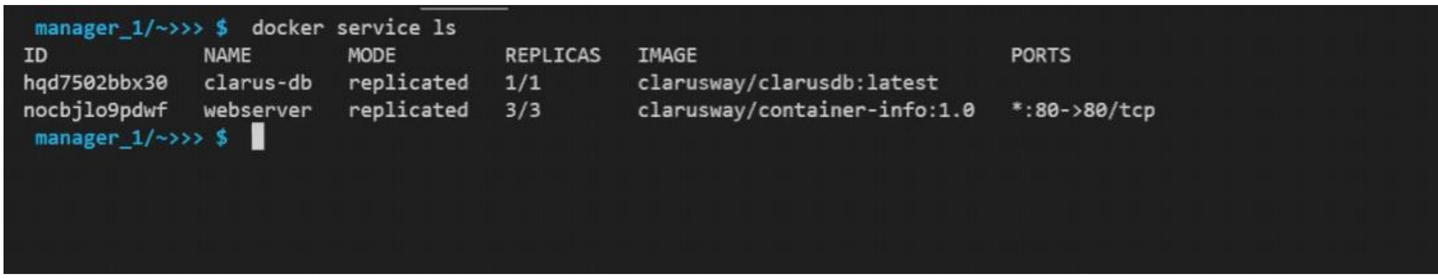
Bir service daha olusturacagiz kendi olusturdugumuz network e bunu da baglayacagzi

- **Create a service for `clarusway/clarusdb` and connect it clarus-net.**
**```bash**
docker service create --name clarus-db --network clarus-net clarusway/clarusdb
```

**Clrus-db**

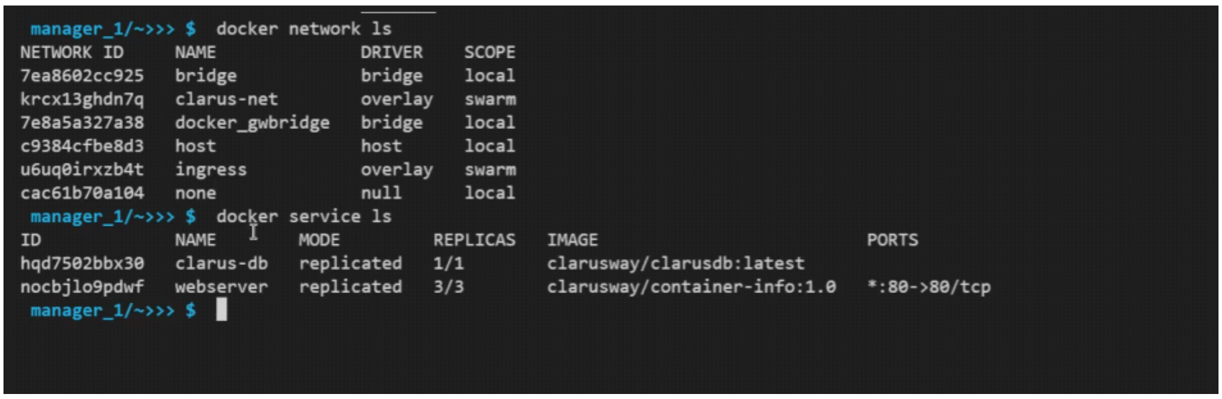


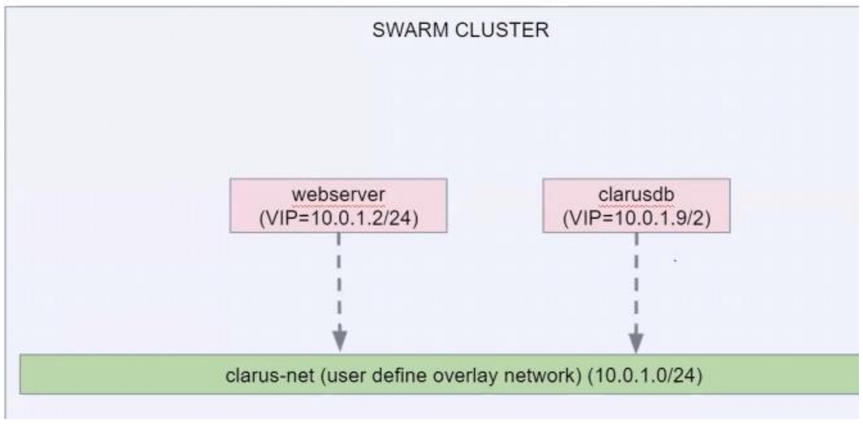
```
720896172e86  clarusway/container-info:1.0   node index.js   3 minutes ago   Up 2 minutes
 manager_1/~>>> $  docker service create --name clarus-db --network clarus-net clarusway/clarusdb
hqd7502bbx30bfd1sxzi76vmg
overall progress: 1 out of 1 tasks
1/1: running   [==================================================>]
verify: Service converged
 manager_1/~>>> $
```



```
  manager_1/~>>> $  docker service ls
 ID             NAME        MODE         REPLICAS   IMAGE                          PORTS
 hqd7502bbx30   clarus-db   replicated   1/1        clarusway/clarusdb:latest
 nocbjlo9pdwf   webserver   replicated   3/3        clarusway/container-info:1.0   *:80->80/tcp
  manager_1/~>>> $
```

Iki service miz var bunlarin nerelere bagli olduklarini görebiliyoruz

- **List services**
**```bash**
docker service ls



```
  manager_1/~>>> $  docker network ls
 NETWORK ID     NAME              DRIVER    SCOPE
 7ea8602cc925   bridge            bridge    local
 krcx13ghdn7q   clarus-net        overlay   swarm
 7e8a5a327a38   docker_gwbridge   bridge    local
 c9384cfbe8d3   host              host      local
 u6uq0irxzb4t   ingress           overlay   swarm
 cac61b70a104   none              null      local
  manager_1/~>>> $  docker service ls
 ID             NAME        MODE         REPLICAS   IMAGE                          PORTS
 hqd7502bbx30   clarus-db   replicated   1/1        clarusway/clarusdb:latest
 nocbjlo9pdwf   webserver   replicated   3/3        clarusway/container-info:1.0   *:80->80/tcp
  manager_1/~>>> $
```

```

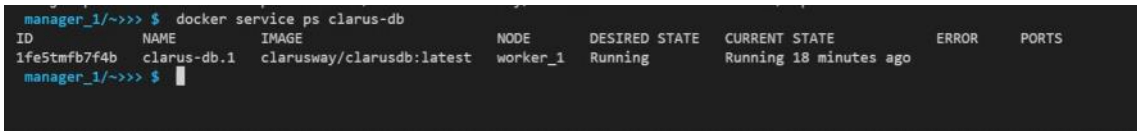

SWARM CLUSTER
webserver (VIP=10.0.1.2/24)
clarusdb (VIP=10.0.1.9/2)
clarus-net (user define overlay network) (10.0.1.0/24)

**simdi yapmak istedigimiz su Iiki service ousturduk ve bunlarin bir birleri ile irtibat kuup kuramayacaklarini ögrenecegiz**

**Bunlarin iletisim kurup kuramayacagini görecegiz**

**- List the tasks and go to terminal of ec2-instance which is running `clarus-db` task.**
**```bash**
**docker service ps clarus-db**
**```**

**Service nin nerede  calistigini görmeye calisacagiz**

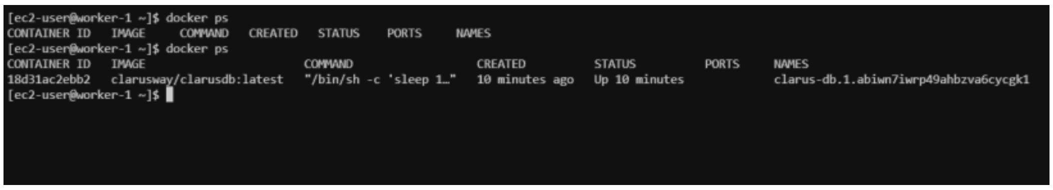


```
manager_1/~>>> $ docker service ps clarus-db
ID   NAME   IMAGE   NODE   DESIRED STATE   CURRENT STATE   ERROR   PORTS
1fe5tmfb7f4b   clarus-db.1   clarusway/clarusdb:latest worker_1   Running   Running 18 minutes ago
manager_1/~>>> $
```

Worker 1 de calistugunu göruyoruz

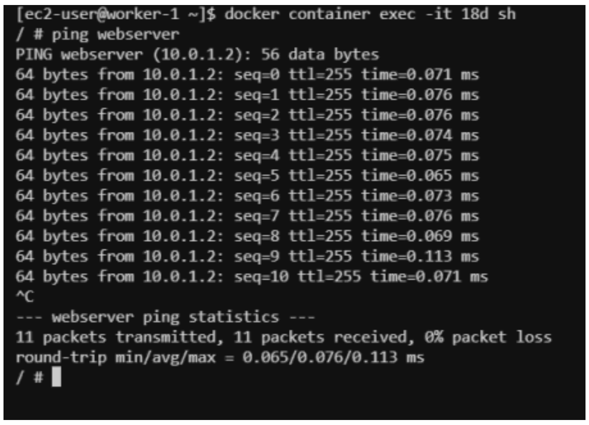
- List the containers in ec2-instance which is running `clarus-db` task.
**```bash**
docker container ls
**```**

Calisan noda geldik



```
[ec2-user@worker-1 ~]$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
[ec2-user@worker-1 ~]$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED          STATUS          PORTS     NAMES
18d31ac2ebb2   clarusway/clarusdb:latest   "/bin/sh -c 'sleep 1…"   10 minutes ago   Up 10 minutes             clarus-db.1.abiwn7iwrp49ahbzva6cycgk1
[ec2-user@worker-1 ~]$
```

- Connect the `clarus-db` container.
**```bash**
docker container exec -it <container_id> sh
**```**



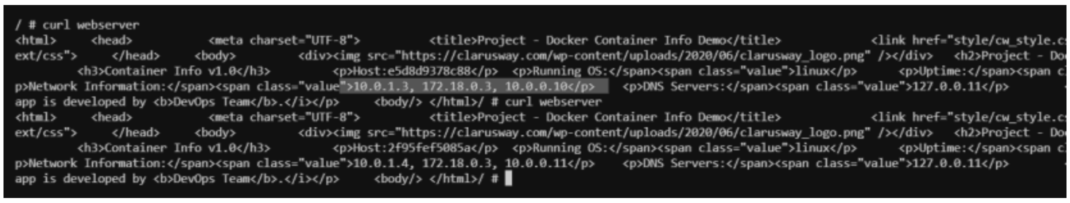
```
[ec2-user@worker-1 ~]$ docker container exec -it 18d sh
/ # ping webserver
PING webserver (10.0.1.2): 56 data bytes
64 bytes from 10.0.1.2: seq=0 ttl=255 time=0.071 ms
64 bytes from 10.0.1.2: seq=1 ttl=255 time=0.076 ms
64 bytes from 10.0.1.2: seq=2 ttl=255 time=0.076 ms
64 bytes from 10.0.1.2: seq=3 ttl=255 time=0.074 ms
64 bytes from 10.0.1.2: seq=4 ttl=255 time=0.075 ms
64 bytes from 10.0.1.2: seq=5 ttl=255 time=0.065 ms
64 bytes from 10.0.1.2: seq=6 ttl=255 time=0.073 ms
64 bytes from 10.0.1.2: seq=7 ttl=255 time=0.076 ms
64 bytes from 10.0.1.2: seq=8 ttl=255 time=0.069 ms
64 bytes from 10.0.1.2: seq=9 ttl=255 time=0.113 ms
64 bytes from 10.0.1.2: seq=10 ttl=255 time=0.071 ms
^C
--- webserver ping statistics ---
11 packets transmitted, 11 packets received, 0% packet loss
round-trip min/avg/max = 0.065/0.076/0.113 ms
/ #
```

Container a baglandik ve icerisinden webserver PING ATTIK CALISTIGINI GÖRDUK

- Ping the webserver service and explain DNS resolution. (When we ping the `Service Name` , it returns Virtual IP of `webserver`).
**```bash**
ping webserver
**```**

- Explain the `load balancing` with the curl command. (Pay attention to the host when input `curl http://webserver`)

```bash
curl http://webserver
```



```
/ # curl webserver
<html>     <head>          <meta charset="UTF-8">          <title>Project - Docker Container Info Demo</title>          <link href="style/cw_style.c
ext/css">     </head>     <body>          <div><img src="https://clarusway.com/wp-content/uploads/2020/06/clarusway_logo.png" /></div>    <h2>Project - Do
         <h3>Container Info v1.0</h3>          <p>Host:e5d8d9378c88</p>  <p>Running OS:</span><span class="value">linux</p>      <p>Uptime:</span><span c
p>Network Information:</span><span class="value">10.0.1.3, 172.18.0.3, 10.0.0.10</p>    <p>DNS Servers:</span><span class="value">127.0.0.11</p>
app is developed by <b>DevOps Team</b>.</i></p>     <body/> </html>/ # curl webserver
<html>     <head>          <meta charset="UTF-8">          <title>Project - Docker Container Info Demo</title>          <link href="style/cw_style.c
ext/css">     </head>     <body>          <div><img src="https://clarusway.com/wp-content/uploads/2020/06/clarusway_logo.png" /></div>    <h2>Project - Do
         <h3>Container Info v1.0</h3>          <p>Host:2f95fef5085a</p>  <p>Running OS:</span><span class="value">linux</p>      <p>Uptime:</span><span c
p>Network Information:</span><span class="value">10.0.1.4, 172.18.0.3, 10.0.0.11</p>    <p>DNS Servers:</span><span class="value">127.0.0.11</p>
app is developed by <b>DevOps Team</b>.</i></p>     <body/> </html>/ #
```

Her seferinde curl webserver ile farkli ipler verdigini görduk buda aslinda her request de farkli ip vermesi anLAMINA HGELIR KI BUNU LOAD BALANCER SAGLIYOR

- Remove the services.

```bash
docker service rm webserver clarus-db
```

Öncelikle 2 ayri txt dosyasi olusturacagiz ve bunlarin icerisine bir takim seyler yaacagiz

## Part 4 - Managing Sensitive Data with Docker Secrets

Hassas datalarimizi sifreleyerek göndermeye secret ile yapiyoruz demistik

- Explain [how to manage sensitive data with Docker secrets](https://docs.docker.com/engine/swarm/secrets/).

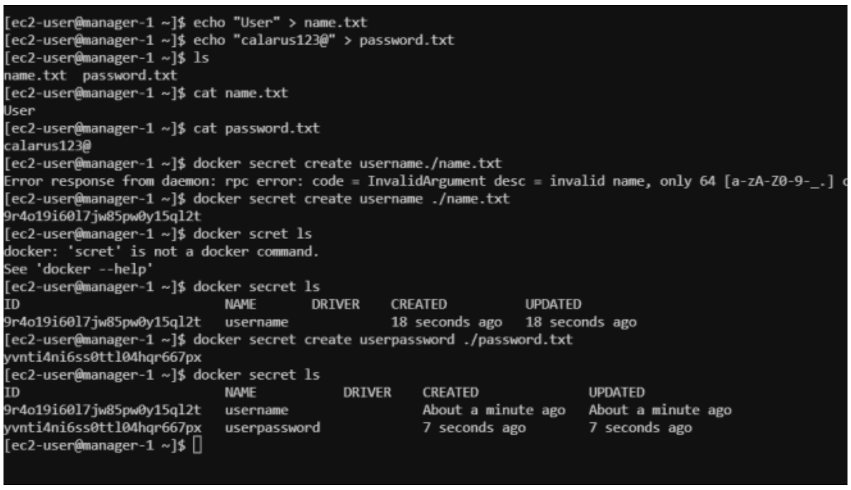
- Create two files named `name.txt` and `password.txt`.

```bash
echo "User" > name.txt
echo "clarus123@" > password.txt
```

- Create docker secrets for both.

```bash
docker secret create username ./name.txt     # secret a username ismini verdik ve bunu bir önceki komutta olsturdugumuz fuile dan al diyoruz

do cker secret create userpassword ./password.txt
```

`



```
[ec2-user@manager-1 ~]$ echo "User" > name.txt
[ec2-user@manager-1 ~]$ echo "calarus123@" > password.txt
[ec2-user@manager-1 ~]$ ls
name.txt  password.txt
[ec2-user@manager-1 ~]$ cat name.txt
User
[ec2-user@manager-1 ~]$ cat password.txt
calarus123@
[ec2-user@manager-1 ~]$ docker secret create username./name.txt
Error response from daemon: rpc error: code = InvalidArgument desc = invalid name, only 64 [a-zA-Z0-9-_.] c
[ec2-user@manager-1 ~]$ docker secret create username ./name.txt
9r4o19i60l7jw85pw0y15ql2t
[ec2-user@manager-1 ~]$ docker scret ls
docker: 'scret' is not a docker command.
See 'docker --help'
[ec2-user@manager-1 ~]$ docker secret ls
ID                          NAME       DRIVER    CREATED          UPDATED
9r4o19i60l7jw85pw0y15ql2t   username             18 seconds ago   18 seconds ago
[ec2-user@manager-1 ~]$ docker secret create userpassword ./password.txt
yvnti4ni6ss0ttl04hqr667px
[ec2-user@manager-1 ~]$ docker secret ls
ID                          NAME           DRIVER    CREATED              UPDATED
9r4o19i60l7jw85pw0y15ql2t   username                 About a minute ago   About a minute ago
yvnti4ni6ss0ttl04hqr667px   userpassword             7 seconds ago        7 seconds ago
[ec2-user@manager-1 ~]$ 
```

docker secret create yazdiktan sonra neyi aktarmak istiyoruz ./ hangi klasörde aktarmak

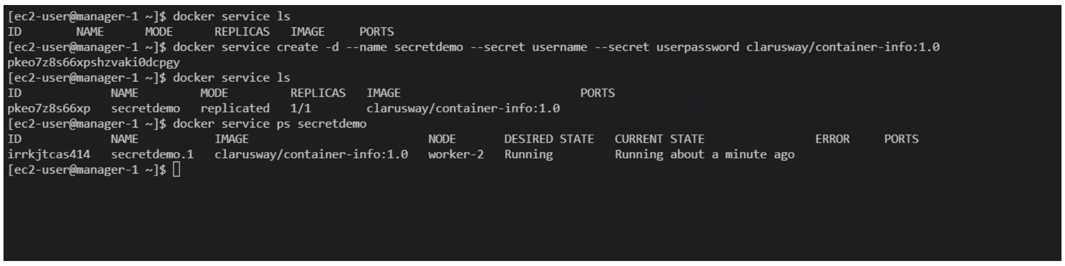
istiyrouz

- List docker secrets.
**```bash**
docker secret ls
**```**

- Create a new service with secrets.

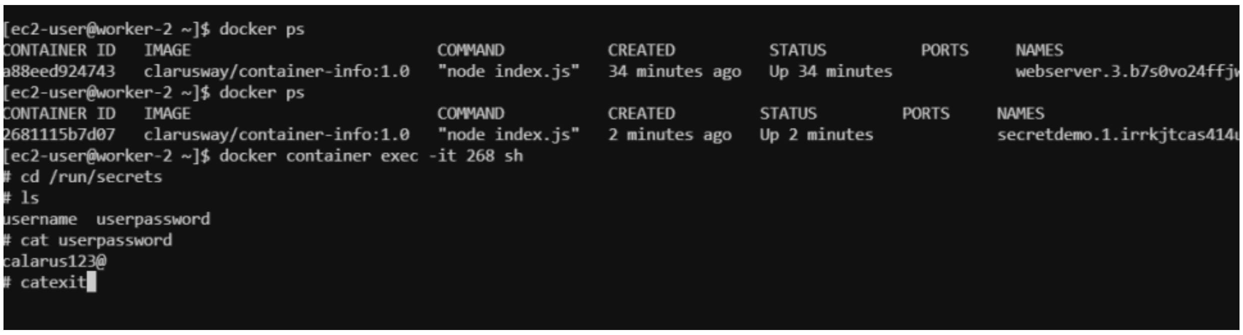
**```bash**
docker service create -d --name secretdemo --secret username --secret userpassword clarusway/container-info:1.0 # yeni bir servis olusturduk secretdemo isimli
**```**

ne demek istedik bu komutta bir servis cretae etmek istedik secretdemo isimli secret userr name ve secret user passwor leri kullanmasini istedik bu iki secreti kullanmasini istiyoruz son olarak ilmaj simimizi yaziyoruz
ayrica clarusway imajinda container info isimli 1.0

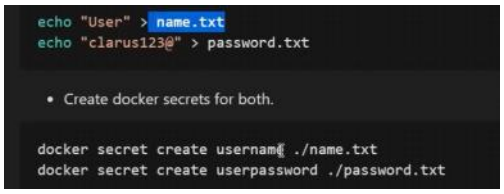

```
[ec2-user@manager-1 ~]$ docker service ls
ID        NAME      MODE      REPLICAS   IMAGE     PORTS
[ec2-user@manager-1 ~]$ docker service create -d --name secretdemo --secret username --secret userpassword clarusway/container-info:1.0
pkeo7z8s66xpshzvaki0dcpgy
[ec2-user@manager-1 ~]$ docker service ls
ID             NAME         MODE         REPLICAS   IMAGE                          PORTS
pkeo7z8s66xp   secretdemo   replicated   1/1        clarusway/container-info:1.0
[ec2-user@manager-1 ~]$ docker service ps secretdemo
ID             NAME           IMAGE                          NODE       DESIRED STATE   CURRENT STATE             ERROR     PORTS
irrkjtcas414   secretdemo.1   clarusway/container-info:1.0   worker-2   Running         Running about a minute ago
[ec2-user@manager-1 ~]$ 
```

- List the tasks and go to terminal of ec2-instance which is running `secretdemo` task.
Nerede kurdugumuz ögreniyor ve gidiyor containerimiza
**```bash**
docker service ps secretdemo # conteynir in nerede oldugunu gördük
**```**
- Connect the `secretdemo` container and show the secrets.

**```bash**
docker container exec -it <container_id> sh
cd /run/secrets
ls
cat username
cat userpassword   # exec ile icine girdik


```
[ec2-user@worker-2 ~]$ docker ps
CONTAINER ID   IMAGE                          COMMAND            CREATED          STATUS          PORTS     NAMES
a88eed924743   clarusway/container-info:1.0   "node index.js"    34 minutes ago   Up 34 minutes             webserver.3.b7s0vo24ffjw
[ec2-user@worker-2 ~]$ docker ps
CONTAINER ID   IMAGE                          COMMAND            CREATED         STATUS         PORTS     NAMES
2681115b7d07   clarusway/container-info:1.0   "node index.js"    2 minutes ago   Up 2 minutes             secretdemo.1.irrkjtcas414u
[ec2-user@worker-2 ~]$ docker container exec -it 268 sh
# cd /run/secrets
# ls
username  userpassword
# cat userpassword
calarus123@
# catexit
```

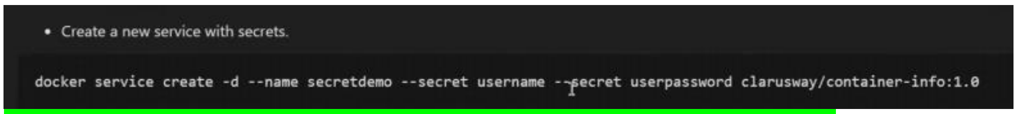
Exec komutu ile cotainerlarimizin icerisine girdik


```
echo "User" > name.txt
echo "clarus123@" > password.txt
```

- Create docker secrets for both.

```
docker secret create username ./name.txt
docker secret create userpassword ./password.txt
```

Ilk olarab docker secret create ile iki adet scret olusturduk ve bunlari hangi file lardan alacagini da belirttik



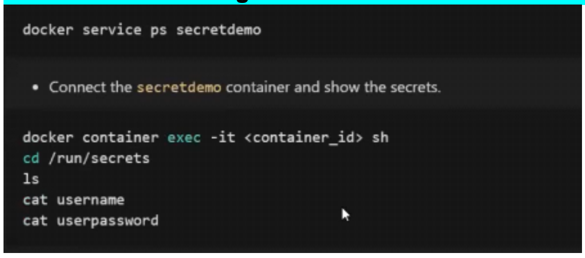
- Create a new service with secrets.

```
docker service create -d --name secretdemo --secret username --secret userpassword clarusway/container-info:1.0
```

Daha sonra service olusturduk ve bunlarad secretleri kullanmasini istedik

Daha sonra docker ps ile nerde oldugunu ögrendik

Cd komurtu ile icine girdik ve cat komutu ile kontrol ettik



```
docker service ps secretdemo
```

- Connect the secretdemo container and show the secrets.

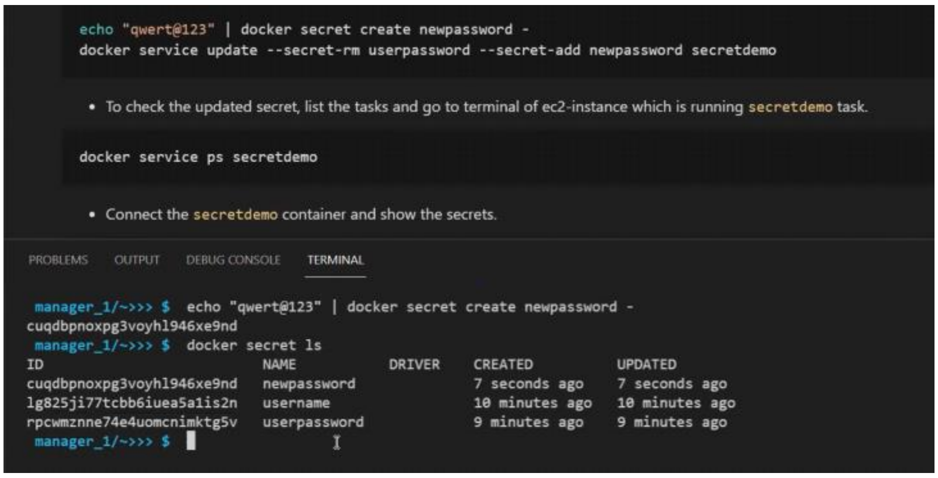
```
docker container exec -it <container_id> sh
cd /run/secrets
ls
cat username
cat userpassword
```

```
```

- To update the secrets; create another secret using `standard input` and remove the old one.(We can't update the secrets.)

Simdi standart pipe isareti ile bir scret olustruacagiz. Baska bir secret olusturma yöntemine geciyoruz echo ile kurduk ve update komutu ile öncekini sildik ve yenisini olusturduk

``bash
echo "qwert@123" | docker secret create newpassword -
docker service update --secret-rm userpassword --secret-add newpassword secretdemo
**```   #**



```
echo "qwert@123" | docker secret create newpassword -
docker service update --secret-rm userpassword --secret-add newpassword secretdemo
```

- To check the updated secret, list the tasks and go to terminal of ec2-instance which is running secretdemo task.

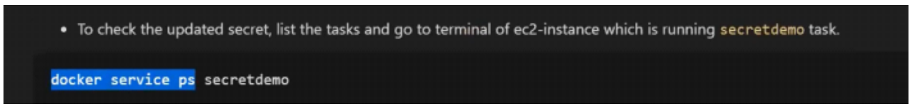
```
docker service ps secretdemo
```

- Connect the secretdemo container and show the secrets.

```
 manager_1/~>>> $  echo "qwert@123" | docker secret create newpassword -
cuqdbpnoxpg3voyhl946xe9nd
 manager_1/~>>> $  docker secret ls
ID                          NAME            DRIVER    CREATED          UPDATED
cuqdbpnoxpg3voyhl946xe9nd   newpassword               7 seconds ago    7 seconds ago
lg825ji77tcbb6iuea5a1is2n   username                  10 minutes ago   10 minutes ago
rpcwmznne74e4uomcnimktg5v   userpassword              9 minutes ago    9 minutes ago
 manager_1/~>>> $
```

- To check the updated secret, list the tasks and go to terminal of ec2-instance which is running `secretdemo` task.

**```bash**
docker service ps secretdemo
**```**



- To check the updated secret, list the tasks and go to terminal of ec2-instance which is running secretdemo task.

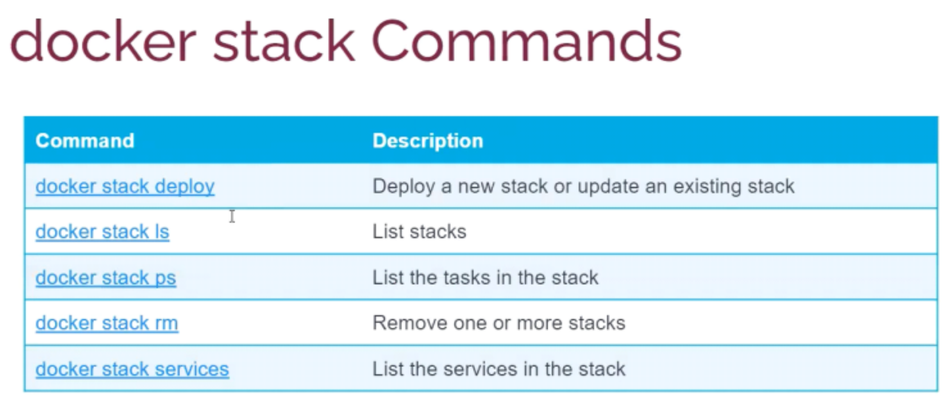
```
docker service ps secretdemo
```

Contaierimizin nerde oldugunu ögrendik

- Connect the `secretdemo` container and show the secrets.
**```bash**
docker container exec -it <container_id> sh
cd /run/secrets
ls
cat newpassword
**```**

## Part 5 - Managing Docker Stack

- Explain `Docker Stack`.
- Create a folder for the project and change into your project directory



docker stack Commands

Command	Description
docker stack deploy	Deploy a new stack or update an existing stack
docker stack ls	List stacks
docker stack ps	List the tasks in the stack
docker stack rm	Remove one or more stacks
docker stack services	List the services in the stack

```bash
mkdir todoapi
cd todoapi
```

- Create a file called `docker-compose.yml` in your project folder with following setup and explain it.

```yaml
version: "3.8"
services:
  database:
    image: mysql:5.7
    environment:
      MYSQL_ROOT_PASSWORD: R1234r
      MYSQL_DATABASE: todo_db
      MYSQL_USER: clarusway
      MYSQL_PASSWORD: Clarusway_1
    networks:
      - clarusnet
  myapp:
    image: clarusway/to-do-api:latest
    deploy:
      replicas: 5
    depends_on:
      - database
    ports:
      - "80:80"
    networks:
      - clarusnet
networks:
  clarusnet:
    driver: overlay
```

- Deploy a new stack.

```bash
docker stack deploy -c ./docker-compose.yml clarus-todoapi
```

- List stacks.

```bash
docker stack ls
```

- List the services in the stack.

```bash
docker stack services clarus-todoapi
```

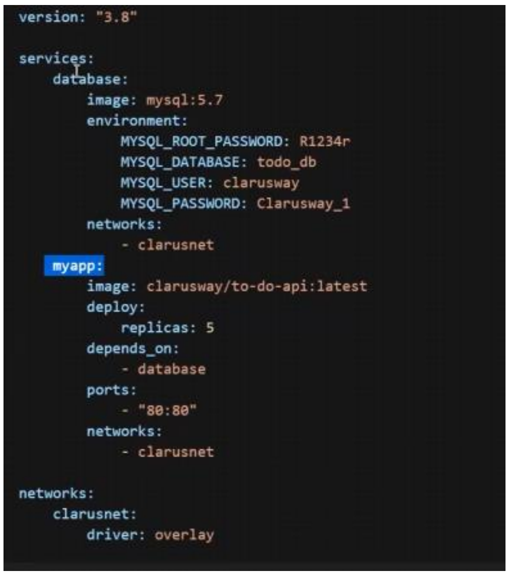
- List the tasks in the stack

**```bash**
docker stack ps clarus-todoapi
**```**
- Check if the `clarus-todoapi` is running by entering `http://<ec2-host-name>` in a browser.
- Remove stacks.

**```bash**
docker stack rm clarus-todoapi
**```**

Kisaca ne yaptigimiza bakalim;

Docker stack olusturmak icin bir yaml dosyasi olusturduk



```
version: "3.8"

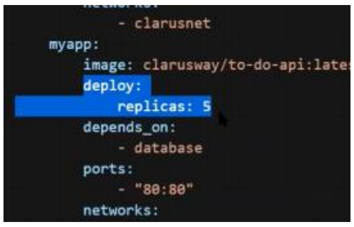
services:
    database:
        image: mysql:5.7
        environment:
            MYSQL_ROOT_PASSWORD: R1234r
            MYSQL_DATABASE: todo_db
            MYSQL_USER: clarusway
            MYSQL_PASSWORD: Clarusway_1
        networks:
            - clarusnet
    myapp:
        image: clarusway/to-do-api:latest
        deploy:
            replicas: 5
        depends_on:
            - database
        ports:
            - "80:80"
        networks:
            - clarusnet

networks:
    clarusnet:
        driver: overlay
```

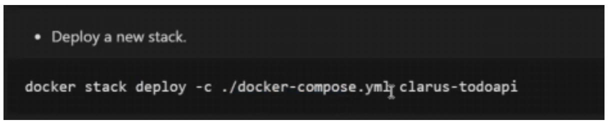
Icerisinde iki adet service olusturduk

biri myapp isimli digeri database isimli

My app de deploy diye bir arguman olusturduk



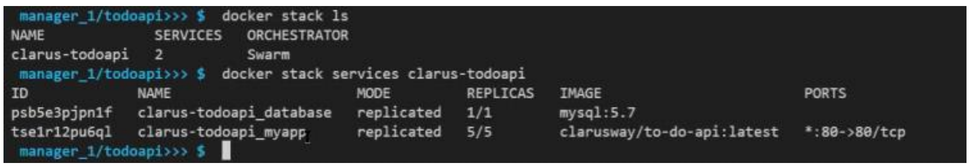
```
            - clarusnet
    myapp:
        image: clarusway/to-do-api:late
        deploy:
            replicas: 5
        depends_on:
            - database
        ports:
            - "80:80"
        networks:
```



- Deploy a new stack.

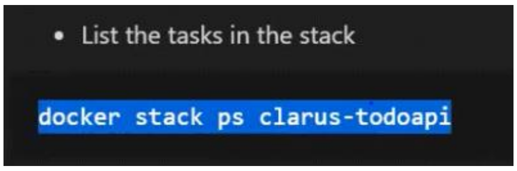
```
docker stack deploy -c ./docker-compose.yml clarus-todoapi
```

Docker stack deploy Komutu ile bu stack calisturdik . / dan sonra nerden alacagini ve son olarak clarus-todoapi isimli bir stack olsun dedik



```
manager_1/todoapi>>> $  docker stack ls
NAME             SERVICES   ORCHESTRATOR
clarus-todoapi   2          Swarm
manager_1/todoapi>>> $  docker stack services clarus-todoapi
ID             NAME                      MODE         REPLICAS   IMAGE                        PORTS
psb5e3pjpn1f   clarus-todoapi_database   replicated   1/1        mysql:5.7
tse1r12pu6ql   clarus-todoapi_myapp      replicated   5/5        clarusway/to-do-api:latest   *:80->80/tcp
manager_1/todoapi>>> $
```

Komutlarimiz ile icerisinde bulunan service leri ve stack leri listeledik



- List the tasks in the stack

```
docker stack ps clarus-todoapi
```

Docker ps komutu ile bu calisan replicalarin hangi conteynirlarda calistigini görebilirz

## Part 6 - Running WordPress as a Docker Stack
- Create a folder for the project and change into your project directory

```bash
mkdir wordpress
cd wordpress


```
- Create a file called `wp_password.txt` containing a password in your project folder.

```bash
echo "Kk12345" > wp_password.txt
```

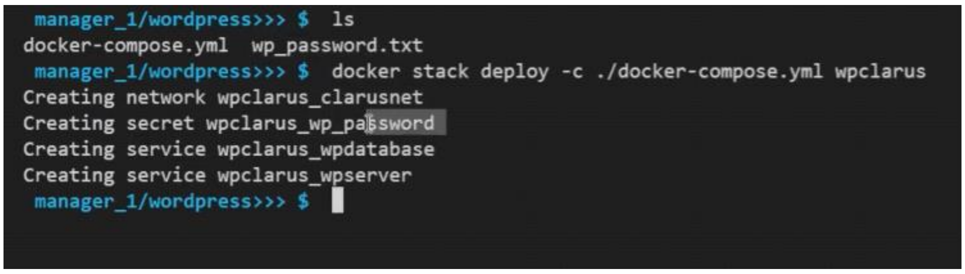
- Create a file called `docker-
compose.yml` in your project folder with following setup and explain it.

```yaml
version: "3.8"
services:    # iki adet service olustruduk
  wpdatabase:      # bir service in ismi bu
    image: mysql:latest
    environment:
      MYSQL_ROOT_PASSWORD: R1234r
      MYSQL_DATABASE: claruswaywp
      MYSQL_USER: clarusway
      MYSQL_PASSWORD_FILE: /run/secrets/wp_password   # sifreyi hangi dosyadan
alacagini tanimliyoruz
    secrets:
      - wp_password
    networks:
      - clarusnet
  wpserver:   # diger servicmizde bu
    image: wordpress:latest
    depends_on:
      - wpdatabase
    deploy:
      replicas: 3
      update_config:
        parallelism: 2
        delay: 5s
        order: start-first
    environment:
      WORDPRESS_DB_USER: clarusway
      WORDPRESS_DB_PASSWORD_FILE: /run/secrets/wp_password
      WORDPRESS_DB_HOST: wpdatabase:3306
      WORDPRESS_DB_NAME: claruswaywp
    ports:
      - "80:80"
    secrets:     # bu komutlariiz ile de baglantiyi sagliyoruz
      - wp_password
    networks:
      - clarusnet
networks:    # bu komutlar ile networkleermiz olusacak
  clarusnet:
    driver: overlay
secrets:
  wp_password:
    file: wp_password.txt
```

- Deploy a new stack.

```bash
docker stack deploy -c ./docker-compose.yml wpclarus
```
- List stacks.
```bash
docker stack ls
```

- List the services in the stack.
```bash
docker stack services wpclarus
```
- List the tasks in the stack
```bash
docker stack ps wpclarus
```
- Check if the `wordpress` is running by entering `http://<ec2-host-name>` in a browser.
- Remove stacks.
```bash
docker stack rm wpclarus
```



```
 manager_1/wordpress>>> $  ls
docker-compose.yml  wp_password.txt
 manager_1/wordpress>>> $  docker stack deploy -c ./docker-compose.yml wpclarus
Creating network wpclarus_clarusnet
Creating secret wpclarus_wp_password
Creating service wpclarus_wpdatabase
Creating service wpclarus_wpserver
 manager_1/wordpress>>> $
```

Önce networlkler sonra secret dosyamiz en son olarakda servicelerimiz olustu