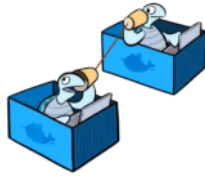


# Docker Swarm Networking and Stack



## Overlay Network

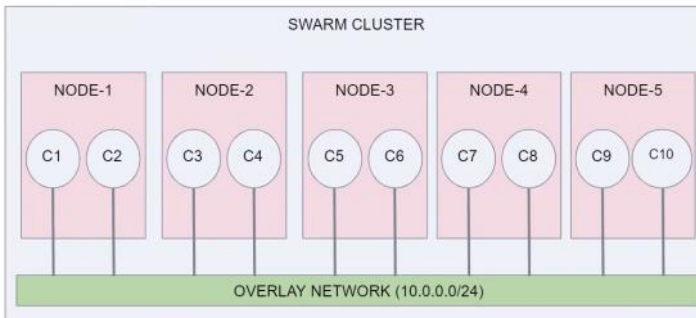
- **Overlay** networks connect multiple Docker daemons together and enable swarm services to communicate with each other.
- You can also use overlay networks to facilitate communication between a swarm service and a standalone container, or between two standalone containers on different Docker daemons.



**Bir çok dockers i bir birine baglar**

**Ayrıca bagimsiz olarak kurulan natlar ar ise onlar ile de iletisim kurmaya yariyor**

## Overlay Network

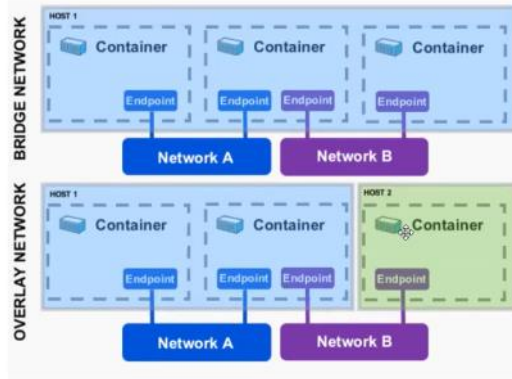


docker container

**Resomada de göruldugu gibi tum swarm icind farkli konteynirlar bir birleri ile iletisime gcebiliyor**

## ► Overlay Network

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



:LARUSWAY©

İlk şekilde bir host ve içinde uc adet container var Bridge network de overlay network de aynı işlemi yapabiliyor aralarında ki fark su diyebiliriz

Resmi su şekile anlayabiliriz ,Network A --Bridge Network  
Network B ---Overlay Network

İlk resimde Tek Host için de farklı containerlarda her ikisi de con kari boir birkerine bağliyalıyor

Ancak ikinci resimde farklı iki host var ve burada bulunan Cont lari bir birine bağliyalıyor

## Overlay Network

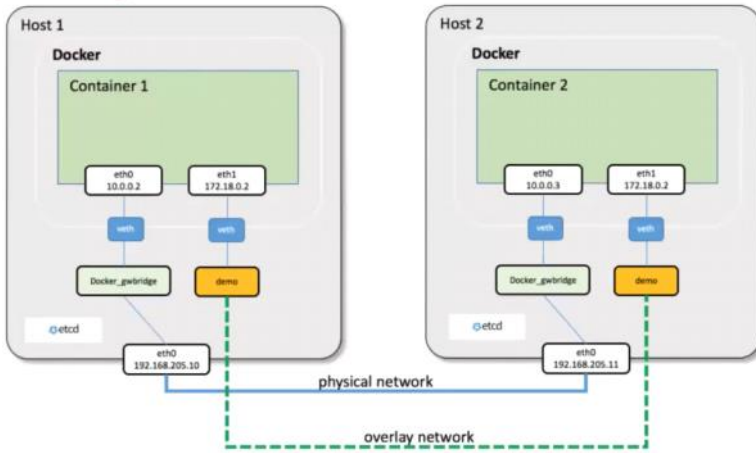
When we initialize a swarm or join a Docker host to an existing swarm, two new networks are created on that Docker host:

- 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 bağlandığımızda Docker bize iki yeni Network olusturuyor**

- Birii **ingress** ; default swarm service
- **Docker\_gwbridge**; docjker host un kendisine bağli 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:**

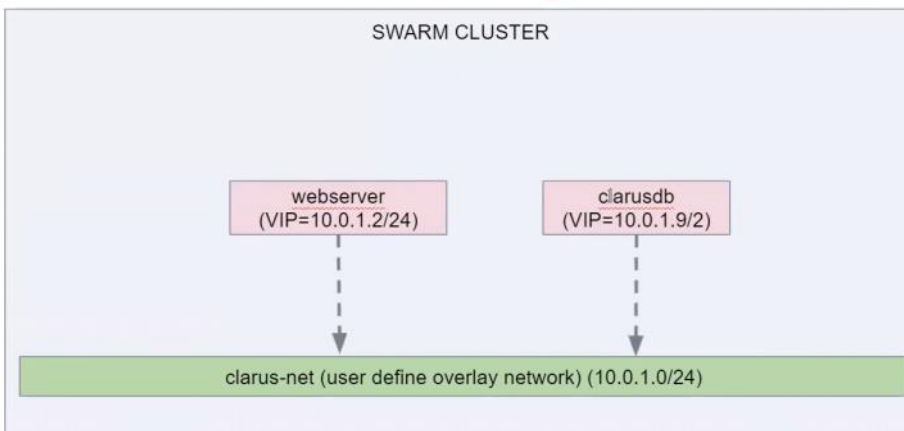
We need the following ports open to traffic to and from each Docker host participating on an overlay network:

- 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

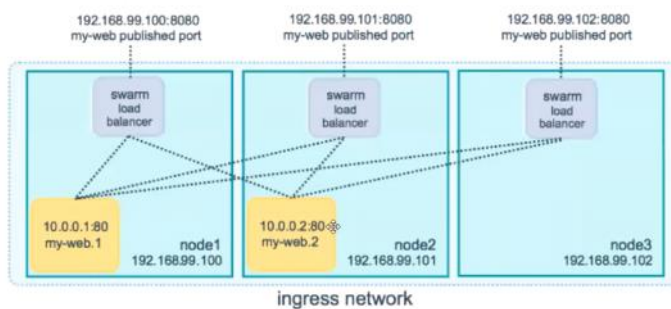


**Bu da hands sonda yapacagimiz isi resmediyor**

## 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

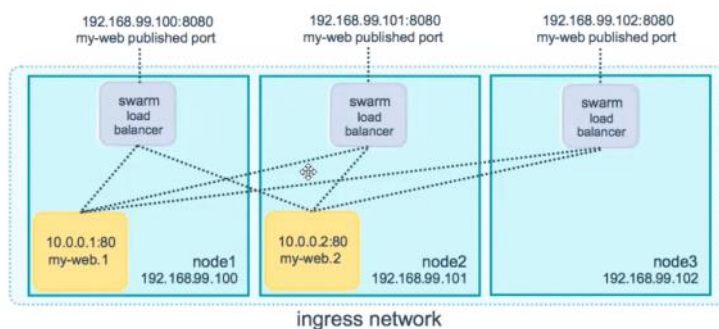


ARUSWAY ©

**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**

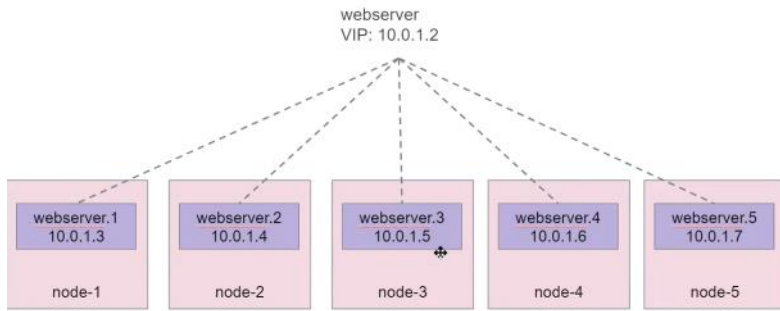
## Swarm Load balancing

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



ARUSWAY ©

## Swarm Load balancing



LABSWAY

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

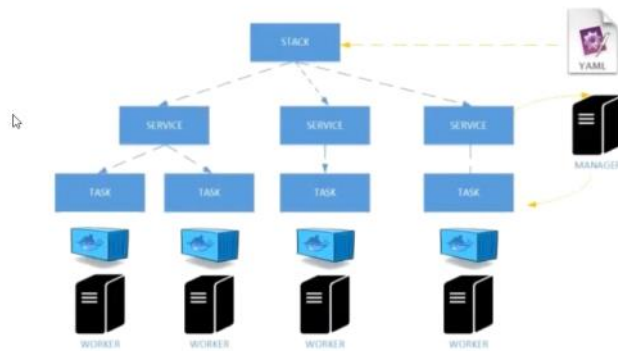
## Docker secret

- 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.



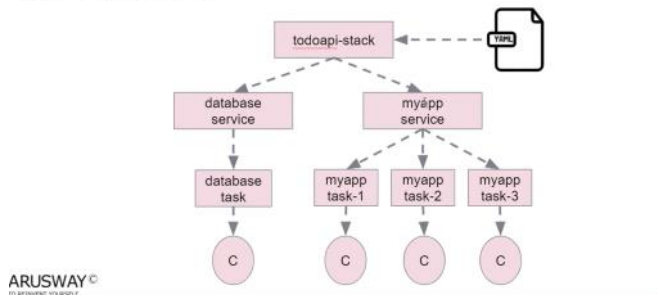
# Docker stack

**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.



**Bir yaml file hazirlayark bir cok service duzenleyebiliyoruz**

## docker stack Commands

Command	Description
<a href="#">docker stack deploy</a>	Deploy a new stack or update an existing stack
<a href="#">docker stack ls</a>	List stacks
<a href="#">docker stack ps</a>	List the tasks in the stack
<a href="#">docker stack rm</a>	Remove one or more stacks
<a href="#">docker stack services</a>	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.

```
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:

```
docker swarm init
# or
docker swarm init --advertise-addr <Private IPs>
```

- Check if the `docker swarm` is active or not.

```
docker info
```

- Get the manager token with `docker swarm join-token manager` command.

```
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.

```
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)

```
docker swarm join --token <worker_token> <manager_ip>:2377
```

- List the connected nodes in `Swarm`.

```
docker node ls
```

### ## Part 3 - Using Overlay Network in Docker Swarm

- List Docker networks and explain overlay network (ingress)

```
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
r2xjxerdvqvj7fzfxepunb0l	worker-1	Ready	Active		20.10.4
00xuhssrrrr24ustp77eo9ks3w	worker-2	Ready	Active		20.10.4

```
[ec2-user@manager-1 ~]$
```

**Docker node ls # —komutu ile daha önceki dersste hazırladığımız 3 manager ve 2 worker i gördük**

```
[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 ~]$
```

**İki tane yeni geliyor**

**Ingress Overlay(default) ver docker\_gwbridge**

**Ve şimdi bunu incelemeye başlıyoruz**

**- 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 bagliyoruz 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 which is not.

```
'''bash
```

```
docker service ps webserver
```

```
'''
```

Bu komut ile de proses leri gorduk.

```
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 service mizin olustugunu gorduk

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

Komnteynir actigimiz makinada gordugumuz kadari ile bir takim bilgileri gorduk sayfayi yeniledigimizde load balancerda etksi ile farkli ipler geldigini gorebiliyoruz

3.92.184.186



#### Project - Docker Container Info Demo

##### Container Info v1.0

Host:2f95fe5085a

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'gorebiliriz

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

baglanamyiyoruz

Ne yapmamiz gerekiyor bazi portlari acmamiz gerekiyor

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

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

### Edit inbound rules [Info](#)

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

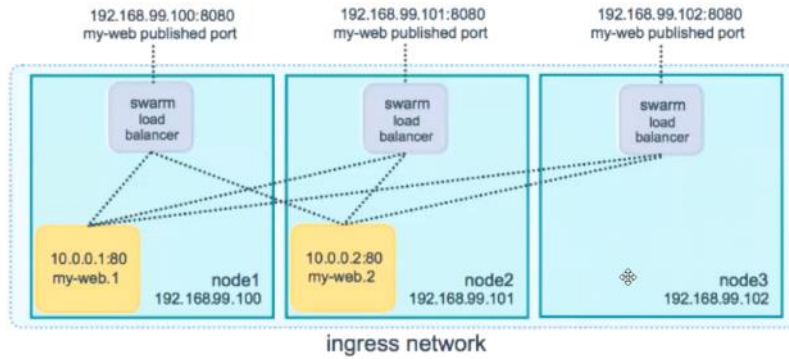
Security group rule ID	Type <a href="#">Info</a>	Protocol <a href="#">Info</a>	Port range <a href="#">Info</a>	Source <a href="#">Info</a>	Description - optional <a href="#">Info</a>	
sgf-0a1fe89ca296ce16c	Custom TCP ▼	TCP	2377	Custom ▼	Q	De let e
					0.0.0.0 /0	
sgf-0555674db59a7d0ac	HTTP ▼	TCP	80	Custom ▼	Q	De let e
					0.0.0.0 /0	
sgf-0c8898c92f8395225	SSH ▼	TCP	22	Custom ▼	Q	De let e
					0.0.0.0 /0	
sgf-08910bad6f2f933a0	Custom TCP ▼	TCP	8080	Custom ▼	Q	De let e
					0.0.0.0 /0	

[Add rule](#)

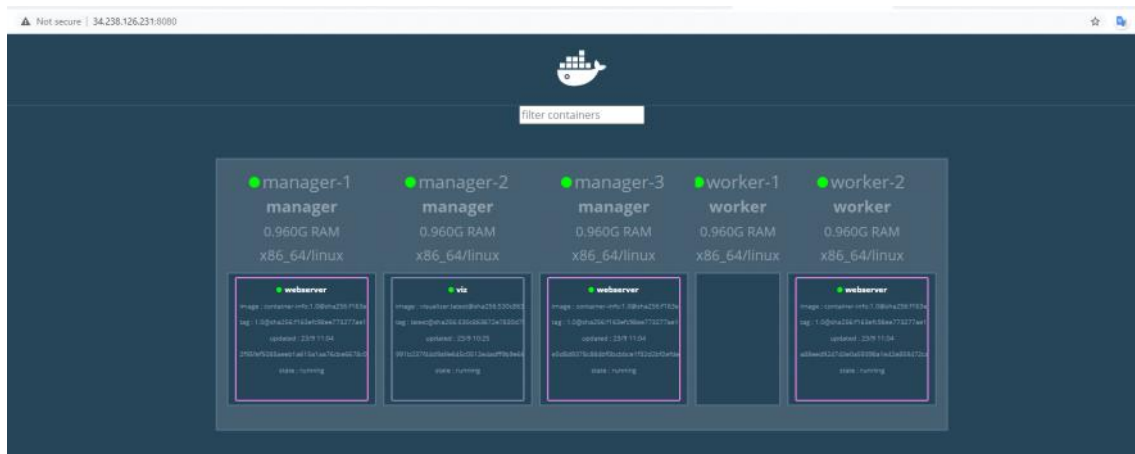
[Cancel](#) [Preview changes](#) [Save rules](#)

AWS Services ▼ Search for services, features, marketplace products, and docs [Alt+S] [Help](#) [Account](#) [Sign out](#) [My Account](#) [My Settings](#) [My Billing](#) [My Support](#) [My Resources](#) [My Recent](#) [My Favorites](#) [My Alerts](#) [My Notifications](#) [My Activity](#) [My Logs](#) [My Metrics](#) [My Dashboards](#) [My Reports](#) [My Analytics](#) [My Insights](#) [My Recommendations](#) [My Suggestions](#) [My Tips](#) [My Tricks](#) [My Hacks](#) [My Scripts](#) [My Tools](#) [My Frameworks](#) [My Libraries](#) [My Packages](#) [My Modules](#) [My Plugins](#) [My Extensions](#) [My Add-ons](#) [My Themes](#) [My Templates](#) [My Widgets](#) [My Components](#) [My Elements](#) [My UI Kits](#) [My Design Systems](#) [My Branding](#) [My Marketing](#) [My Sales](#) [My Customer Support](#) [My HR](#) [My Finance](#) [My Legal](#) [My Compliance](#) [My Security](#) [My Privacy](#) [My Accessibility](#) [My Usability](#) [My User Research](#) [My Analytics](#) [My Performance](#) [My Scalability](#) [My Reliability](#) [My Availability](#) [My Disaster Recovery](#) [My Business Continuity](#) [My Risk Management](#) [My Quality Assurance](#) [My Testing](#) [My Deployment](#) [My Monitoring](#) [My Logging](#) [My Alerting](#) [My Incident Response](#) [My Post-mortem](#) [My Retrospective](#) [My Lessons Learned](#) [My Knowledge Base](#) [My Wiki](#) [My Forum](#) [My Blog](#) [My Newsletter](#) [My Podcast](#) [My YouTube](#) [My Twitter](#) [My LinkedIn](#) [My Facebook](#) [My Instagram](#) [My TikTok](#) [My Snapchat](#) [My Messenger](#) [My WhatsApp](#) [My Telegram](#) [My Signal](#) [My WeChat](#) [My QQ](#) [My Weibo](#) [My Douyin](#) [My Bilibili](#) [My iQiyi](#) [My Tencent Video](#) [My Youku](#) [My Sohu](#) [My PPTV](#) [My LeTV](#) [My 1905](#) [My CCTV](#) [My CCTV-5](#) [My CCTV-6](#) [My CCTV-7](#) [My CCTV-8](#) [My CCTV-9](#) [My CCTV-10](#) [My CCTV-11](#) [My CCTV-12](#) [My CCTV-13](#) [My CCTV-14](#) [My CCTV-15](#) [My CCTV-16](#) [My CCTV-17](#) [My CCTV-18](#) [My CCTV-19](#) [My CCTV-20](#) [My CCTV-21](#) [My CCTV-22](#) [My CCTV-23](#) [My CCTV-24](#) [My CCTV-25](#) [My CCTV-26](#) [My CCTV-27](#) [My CCTV-28](#) [My CCTV-29](#) [My CCTV-30](#) [My CCTV-31](#) [My CCTV-32](#) [My CCTV-33](#) [My CCTV-34](#) [My CCTV-35](#) [My CCTV-36](#) [My CCTV-37](#) [My CCTV-38](#) [My CCTV-39](#) [My CCTV-40](#) [My CCTV-41](#) [My CCTV-42](#) [My CCTV-43](#) [My CCTV-44](#) [My CCTV-45](#) [My CCTV-46](#) [My CCTV-47](#) [My CCTV-48](#) [My CCTV-49](#) [My CCTV-50](#) [My CCTV-51](#) [My CCTV-52](#) [My CCTV-53](#) [My CCTV-54](#) [My CCTV-55](#) [My CCTV-56](#) [My CCTV-57](#) [My CCTV-58](#) [My CCTV-59](#) [My CCTV-60](#) [My CCTV-61](#) [My CCTV-62](#) [My CCTV-63](#) [My CCTV-64](#) [My CCTV-65](#) [My CCTV-66](#) [My CCTV-67](#) [My CCTV-68](#) [My CCTV-69](#) [My CCTV-70](#) [My CCTV-71](#) [My CCTV-72](#) [My CCTV-73](#) [My CCTV-74](#) [My CCTV-75](#) [My CCTV-76](#) [My CCTV-77](#) [My CCTV-78](#) [My CCTV-79](#) [My CCTV-80](#) [My CCTV-81](#) [My CCTV-82](#) [My CCTV-83](#) [My CCTV-84](#) [My CCTV-85](#) [My CCTV-86](#) [My CCTV-87](#) [My CCTV-88](#) [My CCTV-89](#) [My CCTV-90](#) [My CCTV-91](#) [My CCTV-92](#) [My CCTV-93](#) [My CCTV-94](#) [My CCTV-95](#) [My CCTV-96](#) [My CCTV-97](#) [My CCTV-98](#) [My CCTV-99](#) [My CCTV-100](#) [My CCTV-101](#) [My CCTV-102](#) [My CCTV-103](#) [My CCTV-104](#) [My CCTV-105](#) [My CCTV-106](#) [My CCTV-107](#) [My CCTV-108](#) [My CCTV-109](#) [My CCTV-110](#) [My CCTV-111](#) [My CCTV-112](#) [My CCTV-113](#) [My CCTV-114](#) [My CCTV-115](#) [My CCTV-116](#) [My CCTV-117](#) [My CCTV-118](#) [My CCTV-119](#) [My CCTV-120](#) [My CCTV-121](#) [My CCTV-122](#) [My CCTV-123](#) [My CCTV-124](#) [My CCTV-125](#) [My CCTV-126](#) [My CCTV-127](#) [My CCTV-128](#) [My CCTV-129](#) [My CCTV-130](#) [My CCTV-131](#) [My CCTV-132](#) [My CCTV-133](#) [My CCTV-134](#) [My CCTV-135](#) [My CCTV-136](#) [My CCTV-137](#) [My CCTV-138](#) [My CCTV-139](#) [My CCTV-140](#) [My CCTV-141](#) [My CCTV-142](#) [My CCTV-143](#) [My CCTV-144](#) [My CCTV-145](#) [My CCTV-146](#) [My CCTV-147](#) [My CCTV-148](#) [My CCTV-149](#) [My CCTV-150](#) [My CCTV-151](#) [My CCTV-152](#) [My CCTV-153](#) [My CCTV-154](#) [My CCTV-155](#) [My CCTV-156](#) [My CCTV-157](#) [My CCTV-158](#) [My CCTV-159](#) [My CCTV-160](#) [My CCTV-161](#) [My CCTV-162](#) [My CCTV-163](#) [My CCTV-164](#) [My CCTV-165](#) [My CCTV-166](#) [My CCTV-167](#) [My CCTV-168](#) [My CCTV-169](#) [My CCTV-170](#) [My CCTV-171](#) [My CCTV-172](#) [My CCTV-173](#) [My CCTV-174](#) [My CCTV-175](#) [My CCTV-176](#) [My CCTV-177](#) [My CCTV-178](#) [My CCTV-179](#) [My CCTV-180](#) [My CCTV-181](#) [My CCTV-182](#) [My CCTV-183](#) [My CCTV-184](#) [My CCTV-185](#) [My CCTV-186](#) [My CCTV-187](#) [My CCTV-188](#) [My CCTV-189](#) [My CCTV-190](#) [My CCTV-191](#) [My CCTV-192](#) [My CCTV-193](#) [My CCTV-194](#) [My CCTV-195](#) [My CCTV-196](#) [My CCTV-197](#) [My CCTV-198](#) [My CCTV-199](#) [My CCTV-200](#) [My CCTV-201](#) [My CCTV-202](#) [My CCTV-203](#) [My CCTV-204](#) [My CCTV-205](#) [My CCTV-206](#) [My CCTV-207](#) [My CCTV-208](#) [My CCTV-209](#) [My CCTV-210](#) [My CCTV-211](#) [My CCTV-212](#) [My CCTV-213](#) [My CCTV-214](#) [My CCTV-215](#) [My CCTV-216](#) [My CCTV-217](#) [My CCTV-218](#) [My CCTV-219](#) [My CCTV-220](#) [My CCTV-221](#) [My CCTV-222](#) [My CCTV-223](#) [My CCTV-224](#) [My CCTV-225](#) [My CCTV-226](#) [My CCTV-227](#) [My CCTV-228](#) [My CCTV-229](#) [My CCTV-230](#) [My CCTV-231](#) [My CCTV-232](#) [My CCTV-233](#) [My CCTV-234](#) [My CCTV-235](#) [My CCTV-236](#) [My CCTV-237](#) [My CCTV-238](#) [My CCTV-239](#) [My CCTV-240](#) [My CCTV-241](#) [My CCTV-242](#) [My CCTV-243](#) [My CCTV-244](#) [My CCTV-245](#) [My CCTV-246](#) [My CCTV-247](#) [My CCTV-248](#) [My CCTV-249](#) [My CCTV-250](#) [My CCTV-251](#) [My CCTV-252](#) [My CCTV-253](#) [My CCTV-254](#) [My CCTV-255](#) [My CCTV-256](#) [My CCTV-257](#) [My CCTV-258](#) [My CCTV-259](#) [My CCTV-260](#) [My CCTV-261](#) [My CCTV-262](#) [My CCTV-263](#) [My CCTV-264](#) [My CCTV-265](#) [My CCTV-266](#) [My CCTV-267](#) [My CCTV-268](#) [My CCTV-269](#) [My CCTV-270](#) [My CCTV-271](#) [My CCTV-272](#) [My CCTV-273](#) [My CCTV-274](#) [My CCTV-275](#) [My CCTV-276](#) [My CCTV-277](#) [My CCTV-278](#) [My CCTV-279](#) [My CCTV-280](#) [My CCTV-281](#) [My CCTV-282](#) [My CCTV-283](#) [My CCTV-284](#) [My CCTV-285](#) [My CCTV-286](#) [My CCTV-287](#) [My CCTV-288](#) [My CCTV-289](#) [My CCTV-290](#) [My CCTV-291](#) [My CCTV-292](#) [My CCTV-293](#) [My CCTV-294](#) [My CCTV-295](#) [My CCTV-296](#) [My CCTV-297](#) [My CCTV-298](#) [My CCTV-299](#) [My CCTV-300](#) [My CCTV-301](#) [My CCTV-302](#) [My CCTV-303](#) [My CCTV-304](#) [My CCTV-305](#) [My CCTV-306](#) [My CCTV-307](#) [My CCTV-308](#) [My CCTV-309](#) [My CCTV-310](#) [My CCTV-311](#) [My CCTV-312](#) [My CCTV-313](#) [My CCTV-314](#) [My CCTV-315](#) [My CCTV-316](#) [My CCTV-317](#) [My CCTV-318](#) [My CCTV-319](#) [My CCTV-320](#) [My CCTV-321](#) [My CCTV-322](#) [My CCTV-323](#) [My CCTV-324](#) [My CCTV-325](#) [My CCTV-326](#) [My CCTV-327](#) [My CCTV-328](#) [My CCTV-329](#) [My CCTV-330](#) [My CCTV-331](#) [My CCTV-332](#) [My CCTV-333](#) [My CCTV-334](#) [My CCTV-335](#) [My CCTV-336](#) [My CCTV-337](#) [My CCTV-338](#) [My CCTV-339](#) [My CCTV-340](#) [My CCTV-341](#) [My CCTV-342](#) [My CCTV-343](#) [My CCTV-344](#) [My CCTV-345](#) [My CCTV-346](#) [My CCTV-347](#) [My CCTV-348](#) [My CCTV-349](#) [My CCTV-350](#) [My CCTV-351](#) [My CCTV-352](#) [My CCTV-353](#) [My CCTV-354](#) [My CCTV-355](#) [My CCTV-356](#) [My CCTV-357](#) [My CCTV-358](#) [My CCTV-359](#) [My CCTV-360](#) [My CCTV-361](#) [My CCTV-362](#) [My CCTV-363](#) [My CCTV-364](#) [My CCTV-365](#) [My CCTV-366](#) [My CCTV-367](#) [My CCTV-368](#) [My CCTV-369](#) [My CCTV-370](#) [My CCTV-371](#) [My CCTV-372](#) [My CCTV-373](#) [My CCTV-374](#) [My CCTV-375](#) [My CCTV-376](#) [My CCTV-377](#) [My CCTV-378](#) [My CCTV-379](#) [My CCTV-380](#) [My CCTV-381](#) [My CCTV-382](#) [My CCTV-383](#) [My CCTV-384](#) [My CCTV-385](#) [My CCTV-386](#) [My CCTV-387](#) [My CCTV-388](#) [My CCTV-389](#) [My CCTV-390](#) [My CCTV-391](#) [My CCTV-392](#) [My CCTV-393](#) [My CCTV-394](#) [My CCTV-395](#) [My CCTV-396](#) [My CCTV-397](#) [My CCTV-398](#) [My CCTV-399](#) [My CCTV-400](#) [My CCTV-401](#) [My CCTV-402](#) [My CCTV-403](#) [My CCTV-404](#) [My CCTV-405](#) [My CCTV-406](#) [My CCTV-407](#) [My CCTV-408](#) [My CCTV-409](#) [My CCTV-410](#) [My CCTV-411](#) [My CCTV-412](#) [My CCTV-413](#) [My CCTV-414](#) [My CCTV-415](#) [My CCTV-416](#) [My CCTV-417](#) [My CCTV-418](#) [My CCTV-419](#) [My CCTV-420](#) [My CCTV-421](#) [My CCTV-422](#) [My CCTV-423](#) [My CCTV-424](#) [My CCTV-425](#) [My CCTV-426](#) [My CCTV-427](#) [My CCTV-428](#) [My CCTV-429](#) [My CCTV-430](#) [My CCTV-431](#) [My CCTV-432](#) [My CCTV-433](#) [My CCTV-434](#) [My CCTV-435](#) [My CCTV-436](#) [My CCTV-437](#) [My CCTV-438](#) [My CCTV-439](#) [My CCTV-440](#) [My CCTV-441](#) [My CCTV-442](#) [My CCTV-443](#) [My CCTV-444](#) [My CCTV-445](#) [My CCTV-446](#) [My CCTV-447](#) [My CCTV-448](#) [My CCTV-449](#) [My CCTV-450](#) [My CCTV-451](#) [My CCTV-452](#) [My CCTV-453](#) [My CCTV-454](#) [My CCTV-455](#) [My CCTV-456](#) [My CCTV-457](#) [My CCTV-458](#) [My CCTV-459](#) [My CCTV-460](#) [My CCTV-461](#) [My CCTV-462](#) [My CCTV-463](#) [My CCTV-464](#) [My CCTV-465](#) [My CCTV-466](#) [My CCTV-467](#) [My CCTV-468](#) [My CCTV-469](#) [My CCTV-470](#) [My CCTV-471](#) [My CCTV-472](#) [My CCTV-473](#) [My CCTV-474](#) [My CCTV-475](#) [My CCTV-476](#) [My CCTV-477](#) [My CCTV-478](#) [My CCTV-479](#) [My CCTV-480](#) [My CCTV-481](#) [My CCTV-482](#) [My CCTV-483](#) [My CCTV-484](#) [My CCTV-485](#) [My CCTV-486](#) [My CCTV-487](#) [My CCTV-488](#) [My CCTV-489](#) [My CCTV-490](#) [My CCTV-491](#) [My CCTV-492](#) [My CCTV-493](#) [My CCTV-494](#) [My CCTV-495](#) [My CCTV-496](#) [My CCTV-497](#) [My CCTV-498](#) [My CCTV-499](#) [My CCTV-500](#) [My CCTV-501](#) [My CCTV-502](#) [My CCTV-503](#) [My CCTV-504](#) [My CCTV-505](#) [My CCTV-506](#) [My CCTV-507](#) [My CCTV-508](#) [My CCTV-509](#) [My CCTV-510](#) [My CCTV-511](#) [My CCTV-512](#) [My CCTV-513](#) [My CCTV-514](#) [My CCTV-515](#) [My CCTV-516](#) [My CCTV-517](#) [My CCTV-518](#) [My CCTV-519](#) [My CCTV-520](#) [My CCTV-521](#) [My CCTV-522](#) [My CCTV-523](#) [My CCTV-524](#) [My CCTV-525](#) [My CCTV-526](#) [My CCTV-527](#) [My CCTV-528](#) [My CCTV-529](#) [My CCTV-530](#) [My CCTV-531](#) [My CCTV-532](#) [My CCTV-533](#) [My CCTV-534](#) [My CCTV-535](#) [My CCTV-536](#) [My CCTV-537](#) [My CCTV-538](#) [My CCTV-539](#) [My CCTV-540](#) [My CCTV-541](#) [My CCTV-542](#) [My CCTV-543](#) [My CCTV-544](#) [My CCTV-545](#) [My CCTV-546](#) [My CCTV-547](#) [My CCTV-548](#) [My CCTV-549](#) [My CCTV-550](#) [My CCTV-551](#) [My CCTV-552](#) [My CCTV-553](#) [My CCTV-554](#) [My CCTV-555](#) [My CCTV-556](#) [My CCTV-557](#) [My CCTV-558](#) [My CCTV-559](#) [My CCTV-560](#) [My CCTV-561](#) [My CCTV-562](#) [My CCTV-563](#) [My CCTV-564](#) [My CCTV-565](#) [My CCTV-566](#) [My CCTV-567](#) [My CCTV-568](#) [My CCTV-569](#) [My CCTV-570](#) [My CCTV-571](#) [My CCTV-572](#) [My CCTV-573](#) [My CCTV-574](#) [My CCTV-575](#) [My CCTV-576](#) [My CCTV-577](#) [My CCTV-578](#) [My CCTV-579](#) [My CCTV-580](#) [My CCTV-581](#) [My CCTV-582](#) [My CCTV-583](#) [My CCTV-584](#) [My CCTV-585](#) [My CCTV-586](#) [My CCTV-587](#) [My CCTV-588](#) [My CCTV-589](#) [My CCTV-590](#) [My CCTV-591](#) [My CCTV-592](#) [My CCTV-593](#) [My CCTV-594](#) [My CCTV-595](#) [My CCTV-596](#) [My CCTV-597](#) [My CCTV-598](#) [My CCTV-599](#) [My CCTV-600](#) [My CCTV-601](#) [My CCTV-602](#) [My CCTV-603](#) [My CCTV-604](#) [My CCTV-605](#) [My CCTV-606](#) [My CCTV-607](#) [My CCTV-608](#) [My CCTV-609](#) [My CCTV-610](#) [My CCTV-611](#) [My CCTV-612](#) [My CCTV-613](#) [My CCTV-614](#) [My CCTV-615](#) [My CCTV-616](#) [My CCTV-617](#) [My CCTV-618](#) [My CCTV-619](#) [My CCTV-620](#) [My CCTV-621](#) [My CCTV-622](#) [My CCTV-623](#) [My CCTV-624](#) [My CCTV-625](#) [My CCTV-626](#) [My CCTV-627](#) [My CCTV-628](#) [My CCTV-629](#) [My CCTV-630](#) [My CCTV-631](#) [My CCTV-632](#) [My CCTV-633](#) [My CCTV-634](#) [My CCTV-635](#) [My CCTV-636](#) [My CCTV-637](#) [My CCTV-638](#) [My CCTV-639](#) [My CCTV-640](#) [My CCTV-641](#) [My CCTV-642](#) [My CCTV-643](#) [My CCTV-644](#) [My CCTV-645](#) [My CCTV-646](#) [My CCTV-647](#) [My CCTV-648](#) [My CCTV-649](#) [My CCTV-650](#) [My CCTV-651](#) [My CCTV-652](#) [My CCTV-653](#) [My CCTV-654](#) [My CCTV-655](#) [My CCTV-656](#) [My CCTV-657](#) [My CCTV-658](#) [My CCTV-659](#) [My CCTV-660](#) [My CCTV-661](#) [My CCTV-662](#) [My CCTV-663](#) [My CCTV-664](#) [My CCTV-665](#) [My CCTV-666](#) [My CCTV-667](#) [My CCTV-668](#) [My CCTV-669](#) [My CCTV-670](#) [My CCTV-671](#) [My CCTV-672](#) [My CCTV-673](#) [My CCTV-674](#) [My CCTV-675](#) [My CCTV-676](#) [My CCTV-677](#) [My CCTV-678](#) [My CCTV-679](#) [My CCTV-680](#) [My CCTV-681](#) [My CCTV-682](#) [My CCTV-683](#) [My CCTV-684](#) [My CCTV-685](#) [My CCTV-686](#) [My CCTV-687](#) [My CCTV-688](#) [My CCTV-689](#) [My CCTV-690](#) [My CCTV-691](#) [My CCTV-692](#) [My CCTV-693](#) [My CCTV-694](#) [My CCTV-695](#) [My CCTV-696](#) [My CCTV-697](#) [My CCTV-698](#) [My CCTV-699](#) [My CCTV-700](#) [My CCTV-701](#) [My CCTV-702](#) [My CCTV-703](#) [My CCTV-704](#) [My CCTV-705](#) [My CCTV-706](#) [My CCTV-707](#) [My CCTV-708](#) [My CCTV-709](#) [My CCTV-710](#) [My CCTV-711](#) [My CCTV-712](#) [My CCTV-713](#) [My CCTV-714](#) [My CCTV-715](#) [My CCTV-716](#) [My CCTV-717](#) [My CCTV-718](#) [My CCTV-719](#) [My CCTV-720](#) [My CCTV-721](#) [My CCTV-722](#) [My CCTV-723](#) [My CCTV-724](#) [My CCTV-725](#) [My CCTV-726](#) [My CCTV-727](#) [My CCTV-728](#) [My CCTV-729](#) [My CCTV-730](#) [My CCTV-731](#) [My CCTV-732](#) [My CCTV-733](#) [My CCTV-734](#) [My CCTV-735](#) [My CCTV-736](#) [My CCTV-737](#) [My CCTV-738](#) [My CCTV-739](#) [My CCTV-740](#) [My CCTV-741](#) [My CCTV-742](#) [My CCTV-743](#) [My CCTV-744](#) [My CCTV-745](#) [My CCTV-746](#) [My CCTV-747](#) [My CCTV-748](#) [My CCTV-749](#) [My CCTV-750](#) [My CCTV-751](#) [My CCTV-752](#) [My CCTV-753](#) [My CCTV-754](#) [My CCTV-755](#) [My CCTV-756](#) [My CCTV-757](#) [My CCTV-758](#) [My CCTV-759](#) [My CCTV-760](#) [My CCTV-761](#) [My CCTV-762](#) [My CCTV-763](#) [My CCTV-764](#) [My CCTV-765](#) [My CCTV-766](#) [My CCTV-767](#) [My CCTV-768](#) [My CCTV-769](#) [My CCTV-770](#) [My CCTV-771](#) [My CCTV-772](#) [My CCTV-773](#) [My CCTV-774](#) [My CCTV-775](#) [My CCTV-776](#) [My CCTV-777](#) [My CCTV-778](#) [My CCTV-779](#) [My CCTV-780](#) [My CCTV-781](#) [My CCTV-782](#) [My CCTV-783](#) [My CCTV-784](#) [My CCTV-785](#) [My CCTV-786](#) [My CCTV-787](#) [My CCTV-788](#) [My CCTV-789](#) [My CCTV-790](#) [My CCTV-791](#) [My CCTV-792](#) [My CCTV-793](#) [My CCTV-794](#) [My CCTV-795](#) [My CCTV-796](#) [My CCTV-797](#) [My CCTV-798](#) [My CCTV-799](#) [My CCTV-800](#) [My CCTV-801](#) [My CCTV-802](#) [My CCTV-803](#) [My CCTV-804](#) [My CCTV-805](#) [My CCTV-806](#) [My CCTV-807](#) [My CCTV-808](#) [My CCTV-809](#) [My CCTV-810](#) [My CCTV-811](#) [My CCTV-812](#) [My CCTV-813](#) [My CCTV-814](#) [My CCTV-815](#) [My CCTV-816](#) [My CCTV-817](#) [My CCTV-818](#) [My CCTV-819](#) [My CCTV-820](#) [My CCTV-821](#) [My CCTV-822](#) [My CCTV-823](#)

## Swarm Mode Routing Mesh



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 baglayacagiz

- 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

```
7/20/2017 7:28:00 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/~>>> $
```

44:55

```
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/~>>> $
```

**İki service miz var bunların nerelere bağlı olduklarını görebiliyoruz**

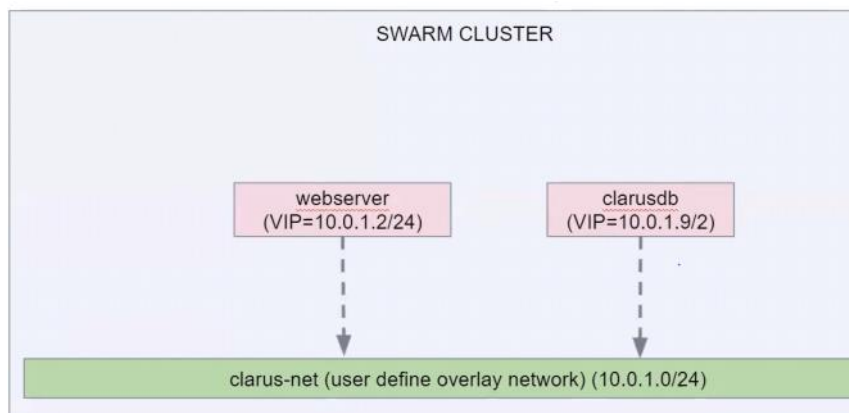
- List services

```
'''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/~>>> $
```

```
'''
```



**şimdi yapmak istediğimiz şu iki service oluşturduk ve bunların bir birleri ile irtibat kurup kuramayacaklarını öğreneceğiz**

## Bunların iletişim kurup kuramayacağını göreceğiz

- 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 çalıştığını görmeye çalışacağız

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

Worker 1 de çalıştığını görüyoruz

- List the containers in ec2-instance which is running `clarus-db` task.

```
'''bash
```

```
docker container ls
```

```
'''
```

Çalışan 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 bağlandık ve içerisinden webserver PING ATTIK ÇALIŞTIRINI GÖRDÜK

- 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></div> <h2>Project - Do
<h3>Container Info v1.0</h3> <p>Host: e5d8d9378c88</p> <p>Running OS: <span><span class="value">linux</span> </span></p> <p>Uptime: <span><span class="value">10.0.1.3, 172.18.0.3, 10.0.0.10</span> </span></p> <p>Network Information: <span><span class="value">10.0.1.3, 172.18.0.3, 10.0.0.10</span> </span></p> <p>DNS Servers: <span><span class="value">127.0.0.11</span> </span></p>
app is developed by <b>DevOps Team</b></p></div> </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></div> <h2>Project - Do
<h3>Container Info v1.0</h3> <p>Host: 2f95fef5085a</p> <p>Running OS: <span><span class="value">linux</span> </span></p> <p>Uptime: <span><span class="value">10.0.1.4, 172.18.0.3, 10.0.0.11</span> </span></p> <p>DNS Servers: <span><span class="value">127.0.0.11</span> </span></p>
app is developed by <b>DevOps Team</b></p></div> </html> / #
```

Her seferinde curl webserver ile farklı ip'ler verdiğini gördük buda aslında her request de farklı ip vermesi anlamına gelir ki bunu load balancer sağlıyor

- Remove the services.

```
'''bash
```

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

```
'''
```

Öncelikle 2 ayrı txt dosyası oluşturacağız ve bunların içerisine bir takım şeyler yazacağız

## ## Part 4 - Managing Sensitive Data with Docker Secrets

Hassas verilerimizi şifreleyerek göndermeye secret ile yapıyoruz 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 oluşturduğumuz fiile dan al diyoruz
```

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

```
'''
```

```
,
```

```
[ec2-user@manager-1 ~]$ echo "User" > name.txt
[ec2-user@manager-1 ~]$ echo "clarus123@" > 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
clarus123@
[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-_.] characters allowed
[ec2-user@manager-1 ~]$ docker secret create username ./name.txt
9r4o19i60l7jw85pw0y15ql2t
[ec2-user@manager-1 ~]$ docker secret ls
docker: 'secret' 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 yazdıktan 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
pkeo7z8s66xpszvaki0dcpgy
[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 öğreniyor ve gidiyor containerimize

```
'''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.b7s0vo24ffjv
[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.irrkjtcas414
[ec2-user@worker-2 ~]$ docker container exec -it 268 sh
# cd /run/secrets
# ls
username  userpassword
# cat userpassword
clarus123@
# 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 olarak 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 öğrendik**

**Cd komutunu 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 secret olusturacagiz. Baska bir secret olusturma yöntemine geciyoruz echo ile kurduk ve update komutu ile öncekini sildik ve yenisini olusturduk**

```
``bash
```

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

```
`` #
```

```
echo "qwerty@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.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
manager_1/~>>> $ echo "qwerty@123" | docker secret create newpassword -
cuqdbpnxpg3voyh1946xe9nd
manager_1/~>>> $ docker secret ls
ID NAME DRIVER CREATED UPDATED
cuqdbpnxpg3voyh1946xe9nd 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
```

**Containerimizin nerde oldugunu öğrendik**

- **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
<a href="#">docker stack deploy</a>	Deploy a new stack or update an existing stack
<a href="#">docker stack ls</a>	List stacks
<a href="#">docker stack ps</a>	List the tasks in the stack
<a href="#">docker stack rm</a>	Remove one or more stacks
<a href="#">docker stack services</a>	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
```

```
'''
```

Kısaca ne yaptığımıza bakalım;

**Docker stack oluşturmak için bir yaml dosyası oluşturduk**

```
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
```

**İçerisinde iki adet service oluşturduk**

**biri myapp isimli diğeri database isimli**

**My app de deploy diye bir arguman oluşturduk**

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

- Deploy a new stack.

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

**Docker stack deploy** Komutu ile bu stack çalıştırdık . / dan sonra nereden alacağını 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
psb5e3pjp1nf     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>>> $
```

**Komutlarımız ile içerisinde bulunan service leri ve stack leri listeledik**

- List the tasks in the stack

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

Docker ps komutu ile bu callsan replicaların hangi conteynirlarda çalıştığını görebiliriz

## ## 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 oluşturduk
  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 tanımlıyoruz
    secrets:
      - wp_password
    networks:
      - clarusnet
  wpserver: # diğer servismizde 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 komutları ile de bağlantıyı sağlıyoruz
      - wp_password
    networks:
      - clarusnet
networks:  # bu komutlar ile networkleermiz oluşacak
  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 networkler sonra secret dosyamız en son olarakda servicelerimiz olustu**







