Docker Swarm

Container Orchestration

- Container Orchestration
  - ✓ 개요
    - 컨테이너 오케스트레이션(Container Orchestration)을 한마디로 정의하자면 수많은 컨테이너의 배포, 관리, 확장, 네트워킹을 자동화하는 프로세스
    - 수천 명의 연주자가 모인 오케스트라에서 각 연주자(컨테이너)가 제시간에 정확한 음을 낼 수 있도록 지휘하는 지휘자와 같은 역할
  - ✓ 필요한 이유
    - 대규모 관리: 수천 개의 컨테이너를 하나하나 수동으로 실행하고 끄는 것은 비효율적
    - 장애 복구(Self-healing): 특정 컨테이너가 죽었을 때 즉시 감지하고 새 컨테이너를 띄워야 함
    - 확장성(Scaling): 사용자가 몰릴 때 컨테이너 수를 자동으로 늘리고 한산해지면 줄여야 함
    - 로드 밸런싱: 트래픽을 여러 컨테이너에 골고루 분산시켜야 함

Container Orchestration

- Container Orchestration
  - ✓ 핵심 기능
    - 프로비저닝 및 배포: 어떤 서버에 컨테이너를 띄울지 결정하고 실행
    - 리소스 할당: 컨테이너가 필요로 하는 CPU나 메모리를 적절히 배정
    - 상태 모니터링: 컨테이너가 건강하게 돌아가고 있는지 감시
    - 서비스 디스커버리: 컨테이너끼리 서로를 찾아서 통신할 수 있게 연결
    - 보안: 컨테이너 간의 접근 권한을 관리
  - ✓ 대표적인 도구
    - Kubernetes (K8s): 사실상 표준인데 기능이 매우 강력하고 생태계가 넓지만 배우기가 다소 어려움
    - Docker Swarm: 도커(Docker)에서 직접 만든 도구로 설치가 쉽고 간단한 환경에 적합
    - Public Cloud에서 제공하는 Amazon ECS: AWS 전용 컨테이너 관리 서비스로 AWS 환경을 쓰고 있다면 연동성이 매우 뛰어난데 현재는 다양한 CSP에서 제공

Docker Swarm

- Docker Swarm
  - ✓ 개요
    - 도커 스웜은 여러 도커 호스트를 클러스터로 묶어주는 컨테이너 오케스트레이션 도구의 한 종류
    - 컨테이너 오케스트레이션 도구 없이는 도커 호스트 여러 대를 사용하는 확장성 있는 애플리케이션을 만들기가 매우 어려움
    - 어느 도커 호스트에 어떤 컨테이너를 배치해야 하는지 서로 다른 호스트에 위치한 컨테이너 간의 통신은 어떻게 제어하는지 등의 조율을 오케스트레이션 도구 없이 하기는 무리
    - 오케스트레이션 도구를 도입하면 이러한 조율에 수고를 절감할 뿐만 아니라 호스트가 여러 대로 나뉘어 있다는 점을 신경 쓰지 않고 클러스터를 투명하게 다룰 수 있다는 이점도 있음

이름	역할	대응하는 명령어
컴포즈	여러 컨테이너로 구성된 도커 애플리케이션을 관리(주로 단일 호스트)	docker-compose
스웜	클러스터 구축 및 관리(주로 멀티 호스트)	docker swarm
서비스	스웜에서 클러스터 안의 서비스(컨테이너 하나 이상의 집합)를 관리	docker service
스택	스웜에서 여러 개의 서비스를 합한 전체 애플리케이션을 관리	docker stack

Docker Swarm

- Docker Swarm
  - ✓ 여러 대의 도커 호스트로 스웜 클러스터 구성
    - 윈도우용/macOS용 도커를 설치하면 호스트가 1대 뿐이므로 여러 대의 호스트를 갖추려면 클라우드 서비스 등을 사용해야 함
    - 도커 머신즈(Docker Machines)를 사용하면 여러 대의 도커 호스트를 만들 수 있지만 도커 머신즈를 사용하려면 macOS에서는 virtualbox 드라이버 윈도우에서는 hyperv 드라이버가 각각 필요
    - 버추얼 박스 같은 가상화 소프트웨어를 사용하면 물리 호스트 1대에 여러 대의 도커 호스트를 실행할 수 있는데 도커 호스트 역할을 할 도커 컨테이너를 여러 개 실행하는 방법으로도 가능
    - 도커 인 도커(Docker in Docker)라는 기능을 사용하면 도커 컨테이너 안에서 도커 호스트를 실행할 수 있음
    - 가상 머신을 사용하면 가상 머신 시작 및 정지에 시간이 걸리고 오버헤드로 소모되는 머신 리소스도 큼

Docker Swarm

- Docker Swarm
  - ✓ 주요 기능
    - 역할이 분리된 분산 설계:
      - 다중 서버를 클러스터에 합류시키면 도커 스웜 모드의 노드의 역할 종류
        - manager node
        - leader node
        - worker node
      - 클러스터에 연결된 서버의 역할은 단일 관리자 환경이면 매니저 노드(리더 노드)와 작업자 노드로 분리하여 각 역할에 따른 전문적 관리를 수행하지만 다중 매니저 노드로 구성하게 되면 그 중 하나를 리더 매니저 노드로 구성하여 나머지 매니저 노드와 작업자 노드를 관리
      - 매니저 노드는 클러스터의 관리 역할로 컨테이너 스케줄링 서비스 및 상태 유지 등을 제공하고 작업자 노드는 컨테이너를 실행하는 역할만 수행
      - 오케스트레이션 도구의 사실상 표준인 쿠버네티스 와 비교하자면 쿠버네티스의 매니저 노드인 마스터 노드는 기본적으로 작업자 노드의 전체적인 관리만 수행하고 서비스 컨테이너는 수행하지 않지만(변경 가능) 도커 스웜 모드의 매니저 노드는 작업자 노드의 역할인 서비스 컨테이너도 수행할 수 있음
      - 관리 역할을 수행하는 노드의 부하를 고려해서 각 역할은 분리해 사용하는 것을 권장하는데 매니저 노드에 서비스 컨테이너를 수행하지 않도록 역할 분리를 수행하는 방법은 도커 서비스 생성 시 --constraint node.role!=manager 옵션을 사용하여 매니저 노드는 서비스 컨테이너를 수행하지 않도록 제외시킬 수 있음

Docker Swarm

- Docker Swarm
  - ✓ 주요 기능
    - 도커 엔진과 통합된 다중 서버 클러스터 환경: 도커 엔진에 포함된 도커 스웜 모드를 통해 별도의 오케스트레이션 도구를 설치하지 않아도 컨테이너 애플리케이션 서비스를 배포하고 관리할 수 있음
    - 서비스 확장과 원하는 상태 조정
      - 도커 스웜 모드에서 서비스 생성 시 안정적인 서비스를 위해 중복(Replicas Option)된 서비스 배포를 할 수 있고 초기 구성 후 스웜 관리자를 통해 애플리케이션 가용성에 영향을 주지 않고도 서비스 확장 및 축소를 수행할 수 있음
      - 배포된 서비스는 매니저 노드를 통해 지속적으로 모니터링
      - 사용자가 요청한 상태와 다르게 서비스 장애(노드 장애 및 서비스 실패 등)가 생길 경우 장애가 발생한 서비스를 대체할 새로운 복제본을 자동으로 생성하여 사용자 요구를 지속하게 되는데 이것이 요구 상태 관리(desire state management)
      - 대부분의 오케스트레이션 도구는 요구 상태 관리를 기본 목적으로 함

Docker Swarm

- Docker Swarm
  - ✓ 주요 기능
    - 서비스 스케줄링
      - 스케줄링 기능은 우리가 구성한 도커 스웜 모드 클러스터 내의 노드에 작업 단위의 서비스 컨테이너를 배포하는 작업
      - 도커 클래식 스웜 엔진의 노드 선택 전략은 swarm manage --strategy (option) 방식을 통해 선택
        - 모든 작업자 노드에 균등하게 할당하는 spread(분산) 전략
        - 작업자 노드의 자원 사용량을 고려하여 할당하는 binpack(자원대비) 전략
        - 임의의 노드에 할당하는 random(무작위) 전략
      - 도커 스웜 모드는 단일 옵션으로 고가용성 분산 알고리즘(HA spread algorithm)을 사용하는데 이 방식은 생성되는 서비스의 복제본을 분산 배포하기 위해 현재 복제본이 가장 적은 작업자 노드 중 에서 이미 스케줄링된 다른 서비스 컨테이너 수가 가장 적은 작업자 노드를 우선 선택

Docker Swarm

- ❖ Docker Swarm
  - ✓ 주요 기능
    - ● 로드 밸런싱
      - ◆ 도커 스웜 모드를 초기화(docker swarm init 명령) 하면 자동으로 생성되는 네트워크 드라이버 중 하나가 Ingress network
      - ◆ 인그레스 네트워크는 서비스의 노드 간 load balancing 과 외부에 서비스를 노출하기 위해 사용되는 overlay network
      - ◆ 도커 스웜 모드는 서비스 컨테이너에 PublishedPort(--publish <published port>:<container port> 옵션)를 자동으로 할당(30000~32767)하거나 수동 노출할 포트를 구성할 수 있고 서비스 컨테이너가 포트를 오픈하면 동시에 모든 노드에서 동일한 포트가 오픈되기 때문에 클러스터에 합류되어 있는 어떤 노드에 요청을 전달해도 실행 중인 서비스 컨테이너에 자동 전달되는데 모든 노드가 스웜 모드의 routing mesh에 참여하기 때문
      - ◆ 작업자 노드1에 nginx 웹 서비스 컨테이너가 실행 중이라고 가정하면 작업자 노드2에서 노출된 포트로 접속해도 작업자 노드1의 nginx 웹 서비스에 접속이 가능한데 인그레스 네트워크를 통해 자동 로드 밸런싱이 수행되는 것

Docker Swarm

- ❖ Docker Swarm
  - ✓ 주요 기능
    - ● 서비스 검색
      - ◆ 도커 스웜 모드는 서비스 검색을 위해 자체 DNS 서버를 통한 서비스 검색 기능을 제공
      - ◆ 도커 스웜 모드 매니저 노드는 클러스터에 합류된 모든 노드의 서비스에 고유한 DNS 이름을 할당하고 할당된 DNS 이름은 도커 스웜 모드에 내장된 DNS 서버를 통해 스웜 모드에서 실행 중인 모든 컨테이너를 조회할 수 있게 되는데 이것을 service discovery 라고 함

Docker Swarm

- ❖ Docker Swarm
  - ✓ 주요 기능
    - Rolling Update
      - ◆ 도커 스웜 모드 매니저 노드를 통해 현재 실행 중인 서비스 컨테이너의 업데이트를 노드 단위로 점진적으로 적용할 수 있는 Rolling update를 할 수 있음
      - ◆ 롤링 업데이트는 각 작업자 노드에서 실행 중인 서비스 컨테이너를 노드 단위의 지연적 업데이트를 수행하는 것
      - ◆ 3개의 작업자 노드를 사용하는 환경에서 nginx:1.19 버전으로 운영 중인 서비스 컨테이너가 있는 경우 nginx를 1.21 버전으로 업데이트를 수행한다면 한 작업자 노드에서 nginx:1.19 서비스 컨테이너를 중지하고 nginx:1.21 버전의 새로운 서비스 컨테이너를 생성하고 이후 업데이트 지연 시간(옵션: update-delay 초)만큼 대기한 뒤 다른 작업자 노드에서 동일한 작업을 반복 수행하여 업데이트를 마무리하는데 이를 통해 서비스 지속성을 갖게 됨
      - ◆ 롤링 업데이트는 새 버전 서비스 컨테이너를 하나씩 늘려가고 이전 버전 서비스 컨테이너를 하나씩 줄여가는 방식인 것
      - ◆ 업데이트가 실패할 경우 다양한 기능을 통해 재시도 및 업데이트 중지 등을 수행할 수도 있고 잘못된 업데이트를 취소하기 위해 rollback 기능도 제공

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 서버 구성
    - ● 서버 기본 구성
      - ◆ swarm-manager: 192.168.56.100
      - ◆ swarm-worker1: 192.168.56.101
      - ◆ swarm-worker2: 192.168.56.102

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 서버 구성
    - ● Ubuntu에서 IP 설정 방법
      - ◆ 인터페이스 확인: ip link show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000

link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000

link/ether 08:00:27:04:1b:10 brd ff:ff:ff:ff:ff:ff

altname enx080027041b10

3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default

link/ether ae:0f:90:4b:11:c6 brd ff:ff:ff:ff:ff:ff

4: br-b0d347f7be88: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default

link/ether 66:65:96:03:3d:93 brd ff:ff:ff:ff:ff:ff

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 서버 구성
    - ● Ubuntu에서 IP 설정 방법
      - ◆ netplan 설정 파일 편집: sudo nano /etc/netplan/00-installer-config.yaml

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.56.101/24]
      gateway4: 192.168.56.1
      nameservers:
        addresses: [8.8.8.8, 1.1.1.1]
```

      - ◆ netplan 변경 내용 적용: sudo netplan apply
      - ◆ 확인: ip addr show

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 서버 구성
    - ● 호스트 명 수정
      - ◆ sudo hostnamectl set-hostname swarm-manager
      - ◆ cat /etc/hostname

        swarm-manager
      - ◆ 호스트 이름 과 IP 연결: sudo vi /etc/hosts

        127.0.0.1 localhost

        127.0.1.1 swarm-manager

        192.168.56.100 swarm-manager

        192.168.56.101 swarm-worker1

        192.168.56.102 swarm-worker2

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 도커 스웜은 도커 엔진에 포함돼 있어 별도의 설치가 필요 없음
  - ✓ 도커 엔진을 스웜 모드로 전환해 클러스터를 초기화하면 됨
  - ✓ 도커 스웜 모드 초기화: swarm init
    - 도커 스웜 모드 초기화
    - docker swarm init [옵션] 명령을 통해 도커 스웜 초기화를 수행
    - 보통은 아무 인자 없이 사용하는 명령이지만 호스트 컴퓨터가 하나 이상의 네트워크에 접속돼 있다면 스웜 통신에 사용할 IP 주소를 선택하라는 메시지와 함께 오류를 발생시킴
    - 도커 엔진 1.24 이상에서만 구성 가능하며 이 작업을 수행하는 서버가 자동으로 관리자 역할을 수행하는 매니저 노드가 됨

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 모드 초기화: swarm init
    - 현재 스웜 모드 상태를 조회: docker info | grep Swarm

      Swarm: inactive
      - ◆ inactive 상태는 현재 스웜 모드가 활성화되지 않았음을 알 수 있음
    - --advertise-addr에는 도커 스웜 모드의 노드가 매니저 노드에 접근하기 위한 IP를 입력: docker swarm init --advertise-addr 192.168.56.100

      Swarm initialized: current node (nd89a5fw43nj2cbb67ui28mrb) is now a manager.

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

      docker swarm join --token SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-ceyhpl97r6ddamirdfxkp8p9k 192.168.56.100:2377

      To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.
      - ◆ 초기화 작업의 결과에서 제공되는 docker swarm join --token 내용은 작업자 노드에 복사하고 붙여넣기 하여 작업자 노드를 매니저 노드와 하나의 클러스터로 합류시킬 수 있음

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 모드 초기화: swarm init
    - 네트워크 확인: sudo netstat -nlp | grep dockerd

```
tcp6       0      0 :::7946                 :::*                    LISTEN      957/dockerd
tcp6       0      0 :::2377                 :::*                    LISTEN      957/dockerd
udp6       0      0 :::7946                 :::*                                957/dockerd
unix  2      [ ACC ]     STREAM     LISTENING     12306    957/dockerd
/var/run/docker/metrics.sock
unix  2      [ ACC ]     STREAM     LISTENING     17157    957/dockerd
/var/run/docker/swarm/control.sock
unix  2      [ ACC ]     STREAM     LISTENING     12911    957/dockerd
/var/run/docker/libnetwork/08aafb715107.sock
```

      - 도커 스웜 모드의 매니저 노드의 기본 포트는 2377을 사용하고 작업자 노드 간의 통신은 7946/tcp, 7946/udp를 사용
      - 스웜 모드에서 사용하는 인그레스 오버레이 네트워크는 4789/tcp, 4789/udp를 사용하므로 방화벽 사용 시 추가해야 함

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 작업자 노드 연결
    - 조인 토큰을 복사하여 작업자 노드에서 사용: docker swarm join --token SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-ceyhpl97r6ddamirdfxkp8p9k 192.168.56.100:2377

      This node joined a swarm as a worker.
      - ◆ 스웜 매니저 노드에서 제공된 토큰token은 외부에 노출되지 않도록 관리
      - ◆ 언제든 확인 및 재생성하여 사용할 수 있음
    - 조인 토큰 조회(manager node에서 수행): docker swarm join-token worker

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

      docker swarm join --token SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-e7pain452umxzihcpywcosp52 192.168.56.100:2377

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 작업자 노드 연결
    - 매니저 노드를 추가하는 경우 새로운 토큰 생성 (manager node에서 수행): docker swarm join-token manager

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

      docker swarm join --token SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-e7pain452umxzihcpywcosp52 192.168.56.100:237
    - 연결 확인 (manager node에서 수행): docker node ls

      ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

      nd89a5fw43nj2cbb67ui28mrb * swarm-manager Ready Active Leader 28.2.2

      l563xejl3gyjflfgfai1wcw51 swarm-worker1 Ready Active 28.2.2

      fmdm0k4ue4kfvxoy3gi7bstk4 swarm-worker2 Ready Active 28.2.2

Docker Swarm

- ❖ Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 작업자 노드 연결
    - 도커 스웜 모드에 대한 세부 정보 확인(manager node에서 수행): docker info

```
Swarm: active
 NodeID: nd89a5fw43nj2cbb67ui28mrb
 Is Manager: true
 ClusterID: li9u68q8w2ualyitejir4gas6
 Managers: 1
 Nodes: 3
 Data Path Port: 4789
 Orchestration:
  Task History Retention Limit: 5
 Raft:
  Snapshot Interval: 10000
  Number of Old Snapshots to Retain: 0
  Heartbeat Tick: 1
  Election Tick: 10
 Dispatcher:
  Heartbeat Period: 5 seconds
 CA Configuration:
  Expiry Duration: 3 months
  Force Rotate: 0
 Autolock Managers: false
```

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 도커 스웜 작업자 노드 연결
    - 새로운 네트워크 확인(manager node에서 수행): docker network ls

```
NETWORK ID      NAME              DRIVER    SCOPE
9cd06a089cfe    bridge            bridge    local
58301605bdff    docker_gwbridge   bridge    local
0b6151cbcc04    host              host      local
nwu47j0ps3dp    ingress           overlay   swarm
b0d347f7be88    kafka_default     bridge    local
1e4eb74a9bd8    none              null      local
```

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 기타 작업
    - 운영 중 노드의 확장을 위해 토큰이 필요한 경우(manager node에서 수행): docker swarm join-token --rotate worker

      Successfully rotated worker join token.

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

      docker swarm join --token SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-26q0y7s38wvepqtl099sbfrek 192.168.56.100:2377
    - join 토큰 만 필요한 경우(manager node에서 수행): docker swarm join-token -q worker

      SWMTKN-1-1fg7u28amdxbhrcyikttyxb3zt3vkmohfie0bkf2no366tqk39-26q0y7s38wvepqtl099sbfrek

Docker Swarm

- Docker Swarm 클러스터 구성
  - ✓ 기타 작업
    - 작업자 노드 제거(manager node에서 수행): docker swarm leave swarm-worker2
    - 작업자 노드 제거(worker node에서 수행): docker swarm leave

Docker Swarm

- ❖ 서비스 생성 및 제거
  - ✓ 서비스 생성
    - docker service create --name swarm-start alpine:3 /bin/ash -c "while true; do echo 'Docker Swarm Mode Start'; sleep 3; done"
      - ◆ 이 서비스 컨테이너는 알파인셸을 이용해 3초 간격으로 특정 문장을 계속 실행
    - 서비스 목록 조회: docker service ls
    - 서비스의 정보 조회: docker service ps swarm-start
    - 로그 확인: docker service logs -f {ID}
    - 서비스 삭제: docker service rm swarm-start
    - 서비스 목록 조회: docker service ls

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 도커 스웜 모드 클러스터는 서비스 컨테이너를 복제 모드로 배포할 경우 구성된 모든 클러스터 노드에서 접속이 가능
  - ✓ 서비스 컨테이너 생성 시 노출할 포트를 --publish 옵션으로 구성할 수 있고 서비스 컨테이너가 포트를 오픈하면 동시에 모든 노드에서 동일한 포트가 오픈되기 때문에 클러스터에 합류된 어떤 노드에 요청을 전달해도 실행 중인 서비스 컨테이너에 자동 전달되는데 이것은 모든 노드가 스웜 모드의 라우팅 메시에 참여하기 때문
  - ✓ 인그레스 네트워크를 통해 자동 로드 밸런싱을 수행
  - ✓ nginx 서비스 생성
    - docker service create --name web-alb --constraint node.role==worker --replicas 2 --publish 8001:80 nginx
  - ✓ 서비스 목록 조회: docker service ls

```
ID             NAME      MODE         REPLICAS   IMAGE          PORTS
4khn1b85lfhf   web-alb   replicated   2/2        nginx:latest   *:8001->80/tcp
```

  - ✓ 각각의 서비스 조회: docker service ps web-alb

```
ID             NAME        IMAGE          NODE            DESIRED STATE   CURRENT STATE            ERROR     PORTS
zm4itm5o88q3   web-alb.1   nginx:latest   swarm-worker1   Running         Running about a minute ago
6g3x2rn7pv3k   web-alb.2   nginx:latest   swarm-worker2   Running         Running about a minute ago
```

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 8001 포트를 클러스터에 합류되어 있는 모든 노드가 오픈했는지 확인
    - adam@swarm-manager:~/docker$ sudo netstat -nlp | grep 8001

    tcp6 0 0 :::8001 :::* LISTEN 819/dockerd
    - adam@swarm-worker1:~$ sudo netstat -nlp | grep 8001

    tcp6 0 0 :::8001 :::* LISTEN 2431/dockerd
    - adam@swarm-worker2:~$ sudo netstat -nlp | grep 8001

    tcp6 0 0 :::8001 :::* LISTEN 2431/dockerd

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 로드밸런싱 확인을 위해 worker에서 웰컴 파일 작성
    - adam@swarm-worker1:~$ vi index.html

      <h1>Hi Docker Swarm 1</h1>
    - adam@swarm-worker1:~$ docker ps

      CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

      2f57180a4595 nginx:latest "/docker-entrypoint...." 12 minutes ago Up 12 minutes 80/tcp web-alb.1.zm4itm5o88q3rr656jrat09k4
    - adam@swarm-worker1:~$ docker cp index.html web-alb.1.zm4itm5o88q3rr656jrat09k4:/usr/share/nginx/html/index.html

      Successfully copied 2.05kB to web-alb.1.zm4itm5o88q3rr656jrat09k4:/usr/share/nginx/html/index.html

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 로드밸런싱 확인을 위해 worker에서 웰컴 파일 작성
    - adam@swarm-worker2:~$ vi index.html

      <h1>Hi Docker Swarm 2</h1>
    - adam@swarm-worker2:~$ docker ps

      CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

      ce7cd418e84b nginx:latest "/docker-entrypoint...." 14 minutes ago Up 14 minutes 80/tcp web-alb.2.6g3x2rn7pv3koi80o9hoi0pg2
    - adam@swarm-worker2:~$ docker cp index.html web-alb.2.6g3x2rn7pv3koi80o9hoi0pg2:/usr/share/nginx/html/index.html

      Successfully copied 2.05kB to web-alb.2.6g3x2rn7pv3koi80o9hoi0pg2:/usr/share/nginx/html/index.html

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 로드밸런싱 확인
    - adam@swarm-manager:~/docker$ curl 192.168.56.100:8001

<h1>Hi Docker Swarm 1</h1>

    - adam@swarm-manager:~/docker$ curl 192.168.56.100:8001

<h1>Hi Docker Swarm 2</h1>

    - adam@swarm-worker1:~$ curl 192.168.56.101:8001

<h1>Hi Docker Swarm 1</h1>

    - adam@swarm-worker1:~$ curl 192.168.56.101:8001

<h1>Hi Docker Swarm 2</h1>

    - adam@swarm-worker2:~$ curl 192.168.56.102:8001

<h1>Hi Docker Swarm 1</h1>

    - adam@swarm-worker2:~$ curl 192.168.56.102:8001

<h1>Hi Docker Swarm 2</h1>

Docker Swarm

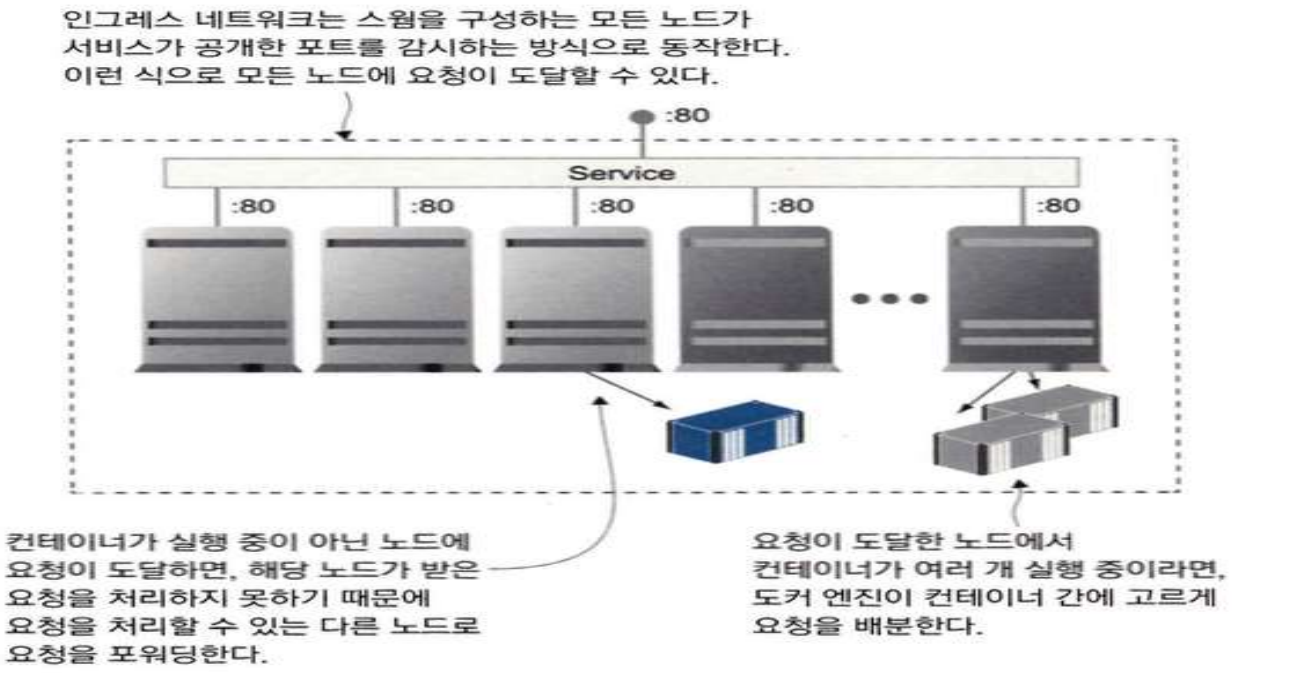
- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 도커 스웜 클러스터에서 네트워크 트래픽 관리
    - 컨테이너에서 실행되는 애플리케이션의 입장에서 스웜 모드의 네트워크는 표준 TCP/IP 방식
    - 컴포넌트는 도메인 네임으로 서로를 식별하며 도커 DNS 서버가 도메인 네임을 조회해 IP 주소를 알려 주면 이 IP 주소로 트래픽을 전달
    - 스웜 모드에서는 서로 다른 노드에서 실행 중인 컨테이너 간에 요청과 응답을 주고받을 수 있지만 컨테이너와 컨테이너에서 실행 중인 애플리케이션은 이를 신경 쓸 필요가 없음
    - 스웜 모드에서는 오버레이 네트워크(overlay network)라는 새로운 형태의 도커 네트워크를 사용하는데 오버레이 네트워크는 클러스터에 속한 모든 노드를 연결하는 가상 네트워크
    - 오버레이 네트워크에 연결된 서비스는 서비스 이름을 도메인 네임 삼아 다른 서비스와 통신할 수 있음
    - 각 애플리케이션은 여러 노드에 걸쳐 실행되는 여러 개의 서비스로 구성되는데 같은 애플리케이션에 속하는 서비스는 오버레이 네트워크를 통해 통신이 가능하지만 오버레이 네트워크 자체는 서로 독립적이기 때문에 서로 다른 네트워크에 속한 서비스는 통신이 불가능

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 도커 스웜 클러스터에서 네트워크 트래픽 관리
    - 서비스가 여러 개의 컨테이너를 실행함에서 각 서비스마다 IP 주소는 하나가 할당되고 이 IP 주소는 서비스에 속하는 모든 레플리카가 공유하는 IP 주소
    - 서비스의 스케일링 규모와 상관없이 DNS 조회 결과에는 IP 주소가 하나만 조회되는데 클라이언트가 이 IP 주소로 요청을 보내면 운영체제의 네트워크 계층에서 이 IP 주소의 종착점이 여러 곳임을 파악하고 그중 하나를 결정함
    - 도커 스웜은 서비스 접근에 대한 신뢰성을 높이고 부하를 잘 분산시키기 위해 VIP 네트워크를 사용
    - 스웜 서비스 형태로 실행되는 애플리케이션도 DNS 네임을 그대로 사용하기 때문에 오버레이 네트워크는 애플리케이션 관점에서는 전혀 겉으로 드러나지 않음
    - 스웜 모드 역시 클러스터로 들어오는 트래픽을 처리하는 복잡한 과정을 이 같은 방식으로 드러내지 않고 숨기는데 이러한 네트워크의 동작은 클러스터를 스케일링하거나 애플리케이션을 스케일링하게 되면 훨씬 더 복잡해짐

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 도커 스웜 클러스터에서 네트워크 트래픽 관리
    - 스웜은 인그레스 네트워킹(ingress networking)을 이용
    - 모든 노드가 같은 포트를 감시하며 외부 트래픽을 받고 이렇게 노드에 도달한 트래픽에 대한 내부적인 로드 밸런싱을 도커가 맡는 방식



Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 도커 스웜 클러스터에서 네트워크 트래픽 관리
    - 스웜은 인그레스 네트워킹(ingress networking)을 이용
    - 모든 노드가 같은 포트를 감시하며 외부 트래픽을 받고 이렇게 노드에 도달한 트래픽에 대한 내부적인 로드 밸런싱을 도커가 맡는 방식
    - 서비스의 포트를 공개하면 인그레스 네트워크가 기본적으로 적용
    - 서비스를 생성할 때 공개할 포트를 지정하기만 하면 인그레스 네트워크를 사용할 수 있음
    - 도커 컴포즈로는 여러 컨테이너가 같은 포트를 감시하도록 할 수 없지만 도커 스웜에서는 인그레스 네트워크를 사용해 서비스 하나가 포트 하나를 사용하는 것이 되므로 요청이 클러스터로 인입되면 서비스 레플리카가 요청을 처음 받은 노드에 실행 중이 아니었더라도 인그레스 네트워크를 통해 서비스 레플리카로 요청이 전달됨

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 동적 확장 과 축소
    - 동적 확장: docker service scale web-alb=3

```
web-alb scaled to 3
overall progress: 3 out of 3 tasks
1/3: running
[==================================================>]
2/3: running
[==================================================>]
3/3: running
[==================================================>]
verify: Service web-alb converged
```

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 동적 확장 과 축소
    - 확인: docker service ls

```
ID             NAME      MODE         REPLICAS  IMAGE          PORTS
4khn1b85lfhf   web-alb   replicated   3/3       nginx:latest   *:8001->80/tcp
```

    - 현재 노드 수보다 많은 복제본으로 확장할 경우에는 특정 노드에 추가 배치를 수행
    - 확인: docker service ps web-alb

```
ID             NAME        IMAGE          NODE            DESIRED STATE  CURRENT STATE            ERROR  PORTS
zm4itm5o88q3   web-alb.1   nginx:latest   swarm-worker1   Running        Running 3 hours ago
6g3x2rn7pv3k   web-alb.2   nginx:latest   swarm-worker2   Running        Running 3 hours ago
vu3qlzgo030b   web-alb.3   nginx:latest   swarm-worker1   Running        Running about a minute ago
```

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 전역 모드: 모든 노드에 전부 노드를 배치
    - 확인: docker service create --name global_nginx --mode global nginx

```
akr1z7w4h1297tmg23zewtjph
overall progress: 3 out of 3 tasks
v4m1upho90ox: running
[==================================================>]
tcj15cbd9xj7: running
[==================================================>]
i7a8tjreuylg: running
[==================================================>]
verify: Service akr1z7w4h1297tmg23zewtjph converged
```

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 전역 모드: 모든 노드에 전부 노드를 배치
    - 확인: docker service ls

```
ID             NAME          MODE         REPLICAS  IMAGE          PORTS
akr1z7w4h129   global_nginx  global       3/3       nginx:latest
4khn1b85lfhf   web-alb       replicated   3/3       nginx:latest   *:8001->80/tcp
```

    - 확인: docker service ps global_nginx

```
ID             NAME                                     IMAGE          NODE           DESIRED STATE  CURRENT STATE            ERROR  PORTS
yioip5jak89r   global_nginx.i7a8tjreuylgtty82ucjlzqbf    nginx:latest   swarm-worker2  Running        Running about a minute ago
rqzr49p8h2tu   global_nginx.tcj15cbd9xj7w2mfbvoy38tq4   nginx:latest   swarm-worker1  Running        Running about a minute ago
bsusp8m8xsrv   global_nginx.v4m1upho90oxycqnsxehfukkg   nginx:latest   swarm-manager  Running        Running about a minute ago
```

Docker Swarm

- nginx를 이용한 서비스 컨테이너 배포 와 관리
  - ✓ 전역 모드: 모든 노드에 전부 노드를 배치
    - 모든 서비스 삭제: docker service rm web-alb global_nginx

web-alb

global_nginx

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 기본적으로 대부분의 오케스트레이션 도구는 장애 복구 기능을 내장하고 있음
  - ✓ 도커 스웜 모드 클러스터에서는 장애 대비를 할 수 있는 복제 모드(--replicas) 기능을 제공하고 배포된 서비스 장애 및 노드의 장애가 발생하면 자동으로 복제 수만큼의 서비스를 맞추어 장애에 대한 자동 복구를 수행
  - ✓ 패키지 버전 변경 수행 시 서비스를 중단하지 않고 롤링 업데이트 기능을 사용하면 새 버전 서비스 컨테이너는 하나씩 늘려가고 이전 버전 서비스 컨테이너는 하나씩 줄여가는 빙식으로 버전 업데이트를 수행할 수 있음
  - ✓ 업데이트가 실패할 경우 다양한 기능을 통해 재시도 및 업데이트 중지 등을 선택 수행할 수도 있고 잘못된 업데이트를 취소하기 위해 롤백 기능도 제공

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 생성: docker service create --name web-alb --replicas 3 --publish 8001:80 nginx

```
4ccl84k6yk4r5efzzvmkrrlnb
overall progress: 3 out of 3 tasks
1/3: running
[==================================================>]
2/3: running
[==================================================>]
3/3: running
[==================================================>]
verify: Service 4ccl84k6yk4r5efzzvmkrrlnb converged
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 노드 중 하나에서 켄테이너 확인: docker ps

```
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS
PORTS     NAMES
bbc330599dbc   nginx:latest   "/docker-entrypoint.…"   About a minute ago   Up 59 seconds
80/tcp    web-alb.3.5vb2133avjwyscakrltxk8656
```

  - ✓ 노드 중 하나에서 컨테이너 강제 삭제: docker rm -f web-alb.3.5vb2133avjwyscakrltxk8656

```
web-alb.3.5vb2133avjwyscakrltxk8656
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 확인: docker service ps web-alb

```
ID             NAME          IMAGE          NODE           DESIRED STATE  CURRENT STATE
ERROR                        PORTS
r04ait89osff   web-alb.1     nginx:latest   swarm-worker1  Running        Running 3
minutes ago
n0cdxm8ihig3   web-alb.2     nginx:latest   swarm-worker2  Running        Running 3
minutes ago
979vax9wfvx9   web-alb.3     nginx:latest   swarm-manager  Running        Running less
than a second ago
5vb2133avjwy    ₩_ web-alb.3  nginx:latest   swarm-manager  Shutdown       Failed 5
seconds ago                  "task: non-zero exit (137)"
```

  - ✓ 매니저 노드에서 장애를 유발한 서비스를 조회해 보면 삭제된 서비스 컨테이너는 shutdown 상태가 되었고 새로 서비스 컨테이너를 생성한 것을 확인할 수 있음

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 생성: docker service create --name my-database --replicas 3 redis:6.0-alpine

```
yk5arj0zcquxfl1ly76znikuz
overall progress: 3 out of 3 tasks
1/3: running
[==================================================>]
2/3: running
[==================================================>]
3/3: running
[==================================================>]
verify: Service yk5arj0zcquxfl1ly76znikuz converged
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 이미지 업데이트: docker service update --image redis:6.2.5-alpine my-database

```
my-database
overall progress: 3 out of 3 tasks
1/3: running
[==================================================>]
2/3: running
[==================================================>]
3/3: running
[==================================================>]
verify: Service my-database converged
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 확인: docker service ps my-database

```
ID             NAME                IMAGE              NODE            DESIRED STATE  CURRENT STATE            ERROR     PORTS
fj08jjc8a2xd   my-database.1       redis:6.2.5-alpine  swarm-worker2   Running        Running 18 seconds ago
465o9jmawej1    ₩_ my-database.1   redis:6.0-alpine    swarm-worker2   Shutdown       Shutdown 22 seconds ago
9j2rv2nnzd1d   my-database.2       redis:6.2.5-alpine  swarm-manager   Running        Running 7 seconds ago
u08zyn62591a    ₩_ my-database.2   redis:6.0-alpine    swarm-manager   Shutdown       Shutdown 11 seconds ago
xujjstcp990g   my-database.3       redis:6.2.5-alpine  swarm-worker1   Running        Running 12 seconds ago
mavv5cchydcy    ₩_ my-database.3   redis:6.0-alpine    swarm-worker1   Shutdown       Shutdown 17 seconds ago
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 롤링 업데이트 수행 과정을 보면 첫 번째 작업 대상을 Shutdown한 뒤 새로운 작업을 준비 ready, running으로 반복하는 것을 확인할 수 있음
  - ✓ 작업 교체 방식으로 업데이트를 수행 – 롤링 업데이트
  - ✓ 롤링 업데이트에는 다음과 같은 추가 기능이 있음
    - --update-parallelism: 동시에 업데이트할 컨테이너 개수를 지정 (기본값: 1)
    - --update-delay: 한 그룹의 업데이트가 끝난 뒤 다음 그룹을 업데이트하기까지의 대기 시간(예: 10s, 1m)
    - --update-order: 업데이트 순서를 결정
      - start-first: 새 컨테이너를 먼저 띄운 뒤 기존 것을 삭제 (중단 시간 최소화)
      - stop-first: 기존 컨테이너를 먼저 죽이고 새 것을 실행 (리소스 부족 시 유리)
      - 2. 업데이트 실패 시 대응 (Rollback)
  - ✓ 업데이트 중에 오류가 발생하면 어떻게 할지 결정하는 옵션
    - --update-failure-action: 업데이트 실패 시 행동 정의
      - pause: 업데이트를 일시 중지 (관리자가 수동 확인)
      - rollback: 자동으로 이전 버전으로 되돌림 (가장 권장됨)
      - continue: 오류를 무시하고 진행
    - --update-max-failure-ratio: 업데이트 중 허용할 수 있는 실패율(예: 0.2는 20% 이상 실패 시 실패로 간주)

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 리소스 및 설정 실시간 변경: 이미지 버전뿐만 아니라 CPU 제한이나 환경 변수도 서비스 중단 없이 업데이트 가능
    - 리소스 변경: --limit-cpu 0.5, --limit-memory 512mb
    - 환경 변수 추가/삭제: --env-add MY_VAR=value, --env-rm OLD_VAR
    - 포트 추가: --publish-add 8080:80
    - 복제본 수 변경 (Scale): --replicas 5

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 상태 조회: docker service inspect --pretty my-database

```
ID:                     yk5arj0zcquxfl1ly76znikuz
Name:                   my-database
Service Mode:           Replicated
 Replicas:  3
UpdateStatus:
 State:                 completed
 Started:   14 hours ago
 Completed:             14 hours ago
 Message:  update completed
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 서비스 생성: docker service create --name my-database2 --replicas 4 --update-delay 10s --update-parallelism 2 redis:6.0-alpine

```
t9brrv15voy9euw52k0x3s8cs
overall progress: 4 out of 4 tasks
1/4: running
[==================================================>]
2/4: running
[==================================================>]
3/4: running
[==================================================>]
4/4: running
[==================================================>]
verify: Service t9brrv15voy9euw52k0x3s8cs converged
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 업데이트: docker service update --image redis:6.2.5-alpine my-database2

```
overall progress: 2 out of 4 tasks
1/4: running
[==================================================>]
2/4: running
[==================================================>]
3/4:
4/4:
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 롤백: docker service rollback my-database2

```
my-database2
rollback: manually requested rollback
overall progress: rolling back update: 4 out of 4 tasks
1/4: running
[==================================================>]
2/4: running
[==================================================>]
3/4: running
[==================================================>]
4/4: running
[==================================================>]
verify: Service my-database2 converged
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 확인: docker service ps my-database2

```
ID             NAME                IMAGE              NODE           DESIRED STATE
CURRENT STATE           ERROR     PORTS
v9mudsd868eg   my-database2.1      redis:6.0-alpine   swarm-manager  Running
Running 47 seconds ago
j36q8fednzpl    ₩_ my-database2.1  redis:6.2.5-alpine swarm-worker1  Shutdown
Shutdown 47 seconds ago
js7taaub81vv    ₩_ my-database2.1  redis:6.0-alpine   swarm-worker2  Shutdown
Shutdown 2 minutes ago
k9kdufjk6t2q   my-database2.2      redis:6.0-alpine   swarm-manager  Running
Running 50 seconds ago
2i4d9qrwfd0u    ₩_ my-database2.2  redis:6.2.5-alpine swarm-manager  Shutdown
Shutdown 50 seconds ago
mec7wpseiytn    ₩_ my-database2.2  redis:6.0-alpine   swarm-manager  Shutdown
Shutdown 2 minutes ago
l2d9s00z8i5o   my-database2.3      redis:6.0-alpine   swarm-worker1  Running
Running 45 seconds ago
v9y31rveh13j    ₩_ my-database2.3  redis:6.2.5-alpine swarm-worker1  Shutdown
Shutdown 46 seconds ago
jrhnr2796o5e    ₩_ my-database2.3  redis:6.0-alpine   swarm-worker1  Shutdown
Shutdown 2 minutes ago
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 롤백 가능 개수 확인: docker info

    Swarm: active

    NodeID: v4m1upho90oxycqnsxehfukkg

    Is Manager: true

    ClusterID: 6391st3053ogfb6nwgwq9syzr

    Managers: 1

    Nodes: 3

    Data Path Port: 4789

    Orchestration:

    Task History Retention Limit: 5
  - ✓ 롤백 가능 개수 수정: docker swarm update --task-history-limit 10

    Swarm updated.

Docker Swarm

- ❖ 서비스 유지 관리 기능
  - ✓ 특정 노드에 업데이트에서 제외: docker node update --availability drain swarm-worker2

    swarm-worker2
  - ✓ 확인: docker node ls

    ID HOSTNAME STATUS AVAILABILITY MANAGER STATUS ENGINE VERSION

    v4m1upho90oxycqnsxehfukkg * swarm-manager Ready Active Leader 29.2.0

    tcj15cbd9xj7w2mfbvoy38tq4 swarm-worker1 Ready Active 29.2.1

    i7a8tjreuylgtty82ucjlzqbf swarm-worker2 Ready Drain 29.2.1

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 확인: docker service ps web-alb

```
ID             NAME          IMAGE          NODE            DESIRED STATE  CURRENT STATE           ERROR                       PORTS
r04ait89osff   web-alb.1     nginx:latest   swarm-worker1   Running        Running 15 hours ago
s4gric4ztc08   web-alb.2     nginx:latest   swarm-worker1   Running        Running 2 minutes ago
n0cdxm8ihig3    ₩_ web-alb.2  nginx:latest   swarm-worker2   Shutdown       Shutdown 2 minutes ago
979vax9wfvx9   web-alb.3     nginx:latest   swarm-manager   Running        Running 15 hours ago
5vb2133avjwy    ₩_ web-alb.3  nginx:latest   swarm-manager   Shutdown       Failed 15 hours ago     "task: non-zero exit (137)"
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 활성 모드로 변경: docker node update --availability active swarm-worker2
  - ✓ 확인: docker service ps web-alb

```
ID             NAME          IMAGE         NODE           DESIRED STATE  CURRENT STATE           ERROR                       PORTS
r04ait89osff   web-alb.1     nginx:latest  swarm-worker1  Running        Running 15 hours ago
s4gric4ztc08   web-alb.2     nginx:latest  swarm-worker1  Running        Running 2 minutes ago
n0cdxm8ihig3    ₩_ web-alb.2  nginx:latest  swarm-worker2  Shutdown       Shutdown 2 minutes ago
979vax9wfvx9   web-alb.3     nginx:latest  swarm-manager  Running        Running 15 hours ago
5vb2133avjwy    ₩_ web-alb.3  nginx:latest  swarm-manager  Shutdown       Failed 15 hours ago     "task: non-zero exit (137)"
```

Docker Swarm

- 서비스 유지 관리 기능
  - ✓ 스케일링: docker service scale web-alb=4
  - ✓ 확인: docker service ps web-alb

```
ID             NAME          IMAGE          NODE            DESIRED STATE  CURRENT
STATE          ERROR                        PORTS
r04ait89osff   web-alb.1     nginx:latest   swarm-worker1   Running        Running 15
hours ago
s4gric4ztc08   web-alb.2     nginx:latest   swarm-worker1   Running        Running 5
minutes ago
n0cdxm8ihig3    ₩_ web-alb.2  nginx:latest   swarm-worker2   Shutdown
Shutdown 5 minutes ago
979vax9wfvx9   web-alb.3     nginx:latest   swarm-manager   Running        Running
15 hours ago
5vb2133avjwy    ₩_ web-alb.3  nginx:latest   swarm-manager   Shutdown       Failed
15 hours ago       "task: non-zero exit (137)"
zih5u0y25kmd   web-alb.4     nginx:latest   swarm-worker2   Running        Running
6 seconds ago
```

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ 네트워크 생성: docker network create --driver overlay dailylog-net
  - ✓ 확인: docker network ls

```
NETWORK ID     NAME              DRIVER    SCOPE
c7a3bcabbdd4   bridge            bridge    local
ohd2bkwn1u0c   dailylog-net      overlay   swarm
89427ca5235d   docker_gwbridge   bridge    local
c88439e4b52a   host              host      local
mixq81p9sap2   ingress           overlay   swarm
b34edce41fb1   none              null      local
```

  - ✓ 디렉토리 생성 및 이동: mkdir dailylog && cd $_

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ docker-compose 파일 작성: nano daily-log.yaml

```
version: '3.9'
services:
  mongodb:
    image: dbgurum/dailylog:db_1.0
    ports:
      - "27017:27017"
    networks:
      - dailylog-net
    deploy:
      placement:
        constraints: [node.role != manager]
      restart_policy:
        condition: on-failure
        max_attempts: 3
        delay: 10s
        window: 120s
```

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ docker-compose 파일 작성: nano daily-log.yaml

```
backend:
  image: dbgurum/dailylog:back_1.0
  ports:
    - "8000:8000"
  networks:
    - dailylog-net
  environment:
    - MONGO_URL=mongodb://mongodb:27017
  depends_on:
    - mongodb
  deploy:
    replicas: 2
    placement:
      constraints: [node.role != manager]
    restart_policy:
      condition: on-failure
      max_attempts: 3
      delay: 10s
      window: 120s
```

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ docker-compose 파일 작성: nano daily-log.yaml

```
frontend:
  image: dbgurum/dailylog:front_1.0
  ports:
    - "3000:8000"
  networks:
    - dailylog-net
  environment:
    - PORT=8000
    - DAILYLOG_API_ADDR=backend:8000
  depends_on:
    - backend
  deploy:
    replicas: 2
    placement:
      constraints: [node.role != manager]
    restart_policy:
      condition: on-failure
      max_attempts: 3
      delay: 10s
      window: 120s
```

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ docker-compose 파일 작성: nano daily-log.yaml

```
networks:
  dailylog-net:
    external: true
```

Docker Swarm

❖ 도커 스웜 스택을 이용한 Compose 파일 배포

✓ 스택 배포: docker stack deploy --compose-file daily-log.yaml dailylog

```
Since --detach=false was not specified, tasks will be created in the background.
In a future release, --detach=false will become the default.
Creating service dailylog_mongodb
Creating service dailylog_backend
Creating service dailylog_frontend
```

✓ 스택 확인: docker stack ls

```
NAME        SERVICES
dailylog    3
```

✓ 서비스 확인: docker service ls

```
ID             NAME               MODE         REPLICAS   IMAGE                      PORTS
lbqbnztgb6qz   dailylog_backend   replicated   0/2        dbgurum/dailylog:back_1.0   *:8000->8000/tcp
gl93bexaogze   dailylog_frontend  replicated   0/2        dbgurum/dailylog:front_1.0  *:3000->8000/tcp
s9896kza5mrv   dailylog_mongodb   replicated   0/1        dbgurum/dailylog:db_1.0     *:27017->27017/tcp
```

Docker Swarm

- 도커 스웜 스택을 이용한 Compose 파일 배포
  - ✓ 서비스 확인: docker stack ps dailylog

```
ID             NAME                  IMAGE                        NODE      DESIRED STATE
CURRENT STATE            ERROR                         PORTS

y45t0wckiiys   dailylog_backend.1     dbgurum/dailylog:back_1.0              Running
Pending about an hour ago   "no suitable node (unsupported…"

fgcvm5xk318r   dailylog_backend.2     dbgurum/dailylog:back_1.0              Running
Pending about an hour ago   "no suitable node (unsupported…"

vhe58cyl7ovt   dailylog_frontend.1    dbgurum/dailylog:front_1.0             Running
Pending about an hour ago   "no suitable node (unsupported…"

fnkow7lt2qtg   dailylog_frontend.2    dbgurum/dailylog:front_1.0             Running
Pending about an hour ago   "no suitable node (unsupported…"

82xgbohq1cut   dailylog_mongodb.1     dbgurum/dailylog:db_1.0                Running
Pending about an hour ago   "no suitable node (unsupported…"
```