Java 설치 = 제타위키

Java home 등록

https://bamdule.tistory.com/57

mysqladmin -u root -p password '새로운비밀번호'

mysql>show databases; //데이터베이스 목록 확인

mysql>use mysql; //mysql 데이터베이스 사용

mysql>update user set password=password('새로운비밀번호') where user='root'; //root의 비밀번호를 설정

mysql>flush privileges; //변경된 설정 적용

/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.262.b10-0.el7_8.x86_64/jre/lib/ext/

/usr/share/tomcat/lib/

Scout

Raw disk

Docker build files

Docker swarm에 대해서 알아오기
****아키텍쳐 고객한테 알아야 그럴싸한 말도 할수 있다
설치 ->
다음주에 설명하자
온라인으로

Compose

항공권

Apache
Npm

서버 오케스트레이션이라는 용어는 모호한 의미를 가지고 있습니다. 간단하게 정의하면 **여러 대의 서버와 여러 개의 서비스를 편리하게 관리해주는 작업**이라고 할 수 있고 실제로는 스케줄링scheduling, 클러스터링clustering, 서비스 디스커버리service discovery, 로깅logging, 모니터링monitoring 같은 일을 합니다.

스케줄링: 컨테이너를 적당한 서버에 배포해 주는 작업입니다. 툴에 따라서 지원하는 전략이 조금씩 다른데 여러 대의 서버 중 가장 할일 없는 서버에 배포하거나 그냥 차례대로 배포 또는 아예 랜덤하게 배포할 수도 있습니다. 컨테이너 개수를 여러 개로 늘리면 적당히 나눠서 배포하고 서버가 죽으면 실행 중이던 컨테이너를 다른 서버에 띄워주기도 합니다.

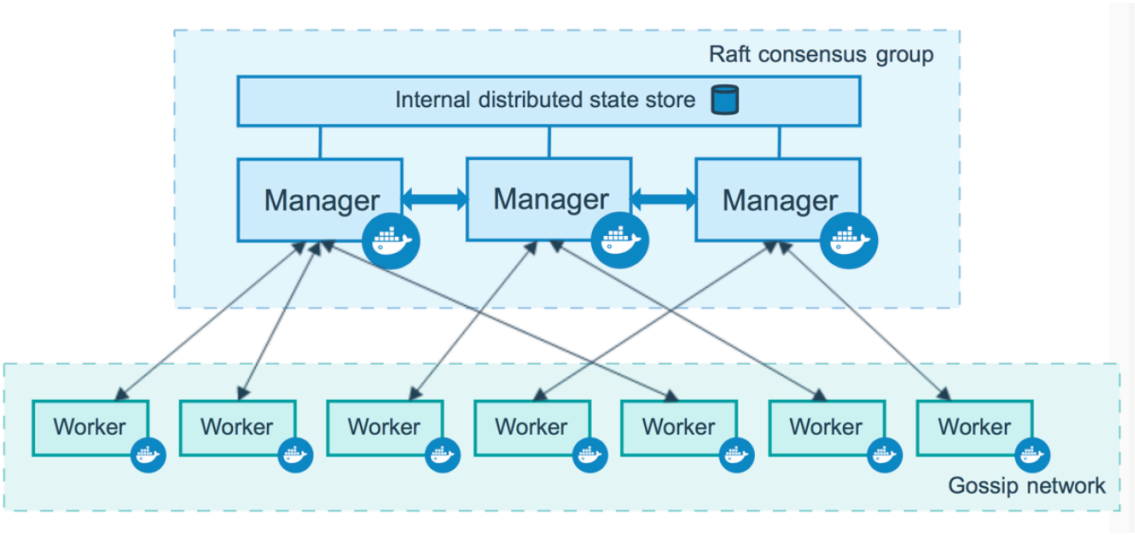
Docker Swarm 이란?
수많은 컨테이너 오케스트레이션 도구 중의 하나로, 여러 대의 Docker 호스트들을 마치 하나인 것처럼 만들어주는 Orchestration 도구입니다.

일반적인 성능의 머신을 여러 대 준비한 다음, 하나의 컴퓨팅 자원처럼 사용하는 것입니다. 이것이 가져다 주는 장점은 명확합니다. 예를 들어 8GB 메모리를 가지고 있는 3대의 머신으로 Docker를 사용한다면 마치 24GB 메모리를 가진 하나의 컴퓨팅 자원처럼 사용할 수 있습니다. 또한 24GB 메모리를 다 사용하여 더 늘릴 필요가 있을 때에는 기존의 3대에 1대를 추가로 Scale Out(노드 수를 늘림으로써 컴퓨팅 자원을 확장시킴) 하여 유동적으로 자원을 사용할 수 있습니다.

그러나 이를 실제로 구축하는 것은 쉬운 일이 아닙니다. 여러 대의 머신을 하나의 컴퓨팅 자원처럼 사용한다고는 해도 실제로는 물리적인 머신이 각각 구분되어져 있는 것이기 때문에 이들을 관리하는 것은 매우 어려운 일입니다. 어느 머신에 컨테이너를 할당하는 것이 가장 효율적인지, 새로운 머신의 발견(Discovery라고 보통 말합니다) 은 어떻게 할 것인지, 컨테이너가 어느 노드에 어떻게 할당되어 있는지 등에 대한 관리가 필요합니다. 이와 같은 컨테이너 관리 기술들을 통틀어 '오케스트레이션' 이라고 합니다.

Clustering
Vmare vmware
Docker open stack
2중화 3중화로 묶어버리자

Worker 노드
자원에 비례해서
Auto scaling
Ochestraling
Ha 고가용성



위와 같이 기본적으로 Docker Swarm 은 Master 노드와 Worker 노드로 시스템을 구성합니다.
Master 노드에서는 클러스터 관리 작업을 하고 클러스터 상태 유지, 스케줄링 서비스, Swarm HTTP API Endpoint 를 제공합니다.
Worker 노드는 컨테이너를 실행하는 역할만 합니다.
yum -y install docker
도커 설치

1. swarm cluster 구성하기

swarm-manager에서 아래의 명령어를 입력합니다. swarm init은 swarm 클러스터를 생성하기 위한 명령어로, 최초 1회에 한해 입력하는 명령어입니다. --advertise-addr 은 다른 노드들이 manager 에게 접근하기 위한 IP 주소입니다. 포트는 기본적으로 2377을 사용하게 되어 있으므로 포트를 개방하는 것을 잊지 마시기 바랍니다.

```
[root@swarm-manager ~]# docker swarm init --advertise-addr 192.168.100.101
```

manager 노드에 2개 이상의 NIC(네트워크 인터페이스 카드) 가 있을 경우 어느 IP로 manager에 접근해야 할 지 다른 머신들에게 알려줘야 할 필요가 있습니다. 예를 들어 ifconfig 를 입력했을 때 출력되는 IP 가 172.17.0.5 / 192.168.100.101 : 두 개라면, swarm 클러스터 내에서 사용할 IP를 정해야만 합니다. 두 IP 중 전자는 NAT, 후자가 Public IP라면 후자를 --advertise-addr에 적어줌으로써 다른 머신들이 해당 머신에 접근할 수 있도록 합니다. 이것이 --advertise-addr 을 설정해야 하는 이유입니다.

위의 커맨드를 입력하면 아래와 같은 출력을 얻을 수 있습니다.

```
[root@swarm-manager ~]# docker swarm init --advertise-addr 192.168.100.101
Swarm initialized: current node (0f43e381p9uchiutn29l91hew) is now a manager.

To add a worker to this swarm, run the following command:
    docker swarm join \
    --token SWMTKN-1-5cjwwzjp2m1bbur2lvsg8wthty4rzacer25z3d1xby79nt00cm-bg8t6ju0k6lv5iy941etk9cvg \
    192.168.100.101:2377

To add a manager to this swarm, run the following command:
    docker swarm join \
    --token SWMTKN-1-5cjwwzjp2m1bbur2lvsg8wthty4rzacer25z3d1xby79nt00cm-e1n8bkrasbqx6ktv2qzj8gn6c \
    192.168.100.101:2377

Colored by Color Scripter
```

```
[root@instance-20200815-1306 opc]# docker node ls
ID                            HOSTNAME                STATUS              AVAILABILITY        MANAGER STATUS      ENGINE VERSION
7xeedebxq523c9nq7bqef7ixz     docker-desktop          Ready               Active                                  19.03.8
vm3qvb7gji6vcsvwypiadwy6r *   instance-20200815-1306  Ready               Active              Leader              19.03.11-ol
lwmbq189g23c9pcwaflcihgzx     instance-20200822-2024  Ready               Active                                  19.03.12
[root@instance-20200815-1306 opc]#
```

```
[root@swarm-manager ~]# docker swarm join-token  --rotate worker
```

```
yslog
 Swarm: active
  NodeID: vm3qvb7gji6vcsvwypiadwy6r
  Is Manager: true
  ClusterID: wzal9444hjgy0lmldqlgjyx26
  Managers: 1
  Nodes: 3
  Default Address Pool: 10.0.0.0/8
  SubnetSize: 24
  Data Path Port: 4789
  Orchestration:
   Task History Retention Limit: 5
  Raft:
   Snapshot Interval: 10000
   Number of Old Snapshots to Retain: 0
   Heartbeat Tick: 1
   Election Tick: 10
  Dispatcher:
   Heartbeat Period: 5 seconds
  CA Configuration:
   Expiry Duration: 3 months
   Force Rotate: 0
  Autolock Managers: false
```

docker swarm leave로 worker의 swarm 클러스터에서 제거 하고 manager일 경우에는 —force를 이용하여 제거

manager 노드가 1개만 있을 때 manager를 삭제하게 되면 그 클러스터는 사용 불가능 상태가 됩니다(.....). 하지 마세요.

https://hidekuma.github.io/docker/swarm/docker-swarm-fail-join-node-as-worker/

```
$ docker swarm join --token SWMTKN-xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx <private:
ip>:2377

Error response from daemon: Timeout was reached before node joined. The attempt
to join the swarm will continue in the background. Use the "docker info" comman
d to see the current swarm status of your node.
Use the "docker info" command to see the current swarm status of your node.
```

결과적으로

보통 1번 문제에 대해 인식을 하지 못했을 경우, 2번 문제(컨테이너 배포가 이루어지지 않을 경우)도 고스란히 발생한다. 해당 문제도 동일하게 방화벽 문제이다.

도커 공식 문서에 의하면 다음과 같이 방화벽을 열어줘야한다.

- **TCP:2377**
  클러스터 매니지먼트(스웜 조인과 같은)에서 사용
- **TCP/UDP:7946**
  노드간 통신
- **UDP:4789**
  오버레이 네트워크 간 트래픽 통신

이렇게 해당 서버의 포트만 열어주면, 문제없이 스웜이 전개된다.

스케줄링: 컨테이너를 적당한 서버에 배포해 주는 작업입니다. 툴에 따라서 지원하는 전략이 조금씩 다른데 여러 대의 서버 중 가장 할일 없는 서버에 배포하거나 그냥 차례대로 배포 또는 아예 랜덤하게 배포할 수도 있습니다. 컨테이너 개수를 여러 개로 늘리면 적당히 나눠서 배포하고 서버가 죽으면 실행 중이던 컨테이너를 다른 서버에 띄워주기도 합니다.

클러스터링: 여러 개의 서버를 하나의 서버처럼 사용할 수 있습니다. 클러스터에 새로운 서버를 추가할 수도 있고 제거할 수도 있습니다. 작게는 몇 개 안 되는 서버부터 많게는 수천 대의 서버를 하나의 클러스터로 만들 수 있습니다. 여기저기 흩어져 있는 컨테이너도 가상 네트워크를 이용하여 마치 같은 서버에 있는 것처럼 쉽게 통신할 수 있습니다.

서비스 디스커버리: 말 그대로 서비스를 찾아주는 기능입니다. 클러스터 환경에서 컨테이너는 어느 서버에 생성될지 알 수 없고 다른 서버로 이동할 수도 있습니다. 따라서 컨테이너와 통신을 하기 위해서 어느 서버에서 실행중인지 알아야 하고 컨테이너가 생성되고 중지될 때 어딘가에 IP와 Port같은 정보를 업데이트해줘야 합니다. 키-벨류 스토리지에 정보를 저장할 수도 있고 내부 DNS 서버를 이용할 수도 있습니다.

로깅, 모니터링: 여러 대의 서버를 관리하는 경우 로그와 서버 상태를 한곳에서 관리하는게 편합니다. 툴에서 직접 지원하는 경우도 있고 따로 프로그램을 설치해야 하는 경우도 있습니다. ELK와 prometheus등 다양한 툴이 있습니다.

결론

- 적당한 규모(수십 대 이내)에서는 `swarm`
- 큰 규모의 클러스터에서는 `kubernetes`
- AWS에서 단순하게 사용한다면 `ECS`
- HashiCorp팬이라면 `Nomad`

https://league-cat.tistory.com/375

```
[[root@instance-20200815-1306 ~]# docker service ls
ID                  NAME                MODE                REPLICAS
IMAGE               PORTS
0jwj3e3e3lco        nginx_service       replicated          3/3
nginx:latest
[[root@instance-20200815-1306 ~]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
     STATUS              PORTS               NAMES
b9ef5abb0cca        nginx:latest        "/docker-entrypoint.…"   43 seconds ago
     Up 41 seconds       80/tcp              nginx_service.3.89ba7fqd0yf20xaix52
xly4v3
[root@instance-20200815-1306 ~]#
```

```
[→  ~ git:(master) ✗ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
      STATUS              PORTS               NAMES
468ce9da44c6        nginx:latest        "/docker-entrypoint.…"   About a minute
```

```
[[root@instance-20200822-2024 opc]# docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED
      STATUS              PORTS               NAMES
28567e69e0d2        nginx:latest        "/docker-entrypoint.…"   2 minutes ago
      Up 2 minutes        80/tcp              nginx_service.2.n14coa9il5nxzlx7vax
68cc7s
[root@instance-20200822-2024 opc]#
```

```
[root@instance-20200815-1306 ~]# docker service scale nginx_service=5
nginx_service scaled to 5
overall progress: 5 out of 5 tasks
1/5: running
2/5: running
3/5: running
4/5: running
5/5: running
verify: Service converged
[root@instance-20200815-1306 ~]# docker service ps nginx_service
ID                  NAME                IMAGE               NODE                DESIRED STATE       CURRENT STATE            ERR
R              PORTS
i485ijl5cvqm        nginx_service.1     nginx:latest        docker-desktop      Running             Running 8 minutes ago

n14coa9il5nx        nginx_service.2     nginx:latest        instance-20200822-2024   Running             Running 8 minutes ago

89ba7fqd0yf2        nginx_service.3     nginx:latest        instance-20200815-1306   Running             Running 8 minutes ago

bmig79fgrnjl        nginx_service.4     nginx:latest        docker-desktop      Running             Running 17 seconds ago

5pbur95uzubc        nginx_service.5     nginx:latest        instance-20200822-2024   Running             Running 16 seconds ago

[root@instance-20200815-1306 ~]#
```

```
[[root@instance-20200815-1306 ~]# docker service rm nginx_service
nginx_service
```

Vagrant 설치
sudo yum install https://releases.hashicorp.com/vagrant/2.2.6/vagrant_2.2.6_x86_64.rpm

yum install kernel-uek-devel-4.14.35-1902.304.6.el7uek.x86_64
(The last command may fail if your system is not fully updated.)
yum install kernel-uek-devel
vboxdrv.sh: failed: Look at /var/log/vbox-install.log to find out what went wrong.
This system is not currently set up to build kernel modules (system extensions).
Running the following commands should set the system up correctly:

yum install kernel-uek-devel-4.14.35-1902.304.6.el7uek.x86_64
(The last command may fail if your system is not fully updated.)
yum install kernel-uek-devel

There were problems setting up VirtualBox. To re-start the set-up process, run

/sbin/vboxconfig
as root.

Virtual box 설치 오류시 재설치 방법

```
[core@core-01 ~ $ curl core-01:4568
 b8caa77f79b9 > 1
[c4e1743754f7 > 1core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 1
 b8caa77f79b9 > 1
[c4e1743754f7 > 1core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 1
 b8caa77f79b9 > 2
[c4e1743754f7 > 1core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 1
 b8caa77f79b9 > 2
[c4e1743754f7 > 2core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 2
 b8caa77f79b9 > 2
[c4e1743754f7 > 2core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 2
 b8caa77f79b9 > 3
[c4e1743754f7 > 2core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 2
 b8caa77f79b9 > 3
[c4e1743754f7 > 3core@core-01 ~ $ curl core-01:4568
 ece7414d5129 > 3
```

Api

Application programing interface

연계해서 쓰는것들
Rest api
Soft ware
Web
System 을 위한
아키텍쳐 하나의 형식
State transfer
Representation
Socket l4
Load balancing
L1 r
L2 switch
L3 router
L4 load balancing
Osi 7계층 http
Rest api
각각의 노드 네트워크 이중화

Master slave
Active backup

Worker node 일만 하는 거
Task
클라이언트 매니저한테
뭐만들라고 web
서비스정보 서비스에 대한것을 task로 복제 테스크
워커 노드
Routing mesh

이미지를 만들어서 컨테이너 올리기
도커 이미지 생성 구축 빌드 컴포즈
Docker compose
Web was db 최적화를 해서 이미지 만들기
limit에 맞춰 web was db 띄우기
1cpu node1개
이미지
Tcpdump -nni 180.100.105.254