분산 커널 기반의 컨테이너 관리 시스템

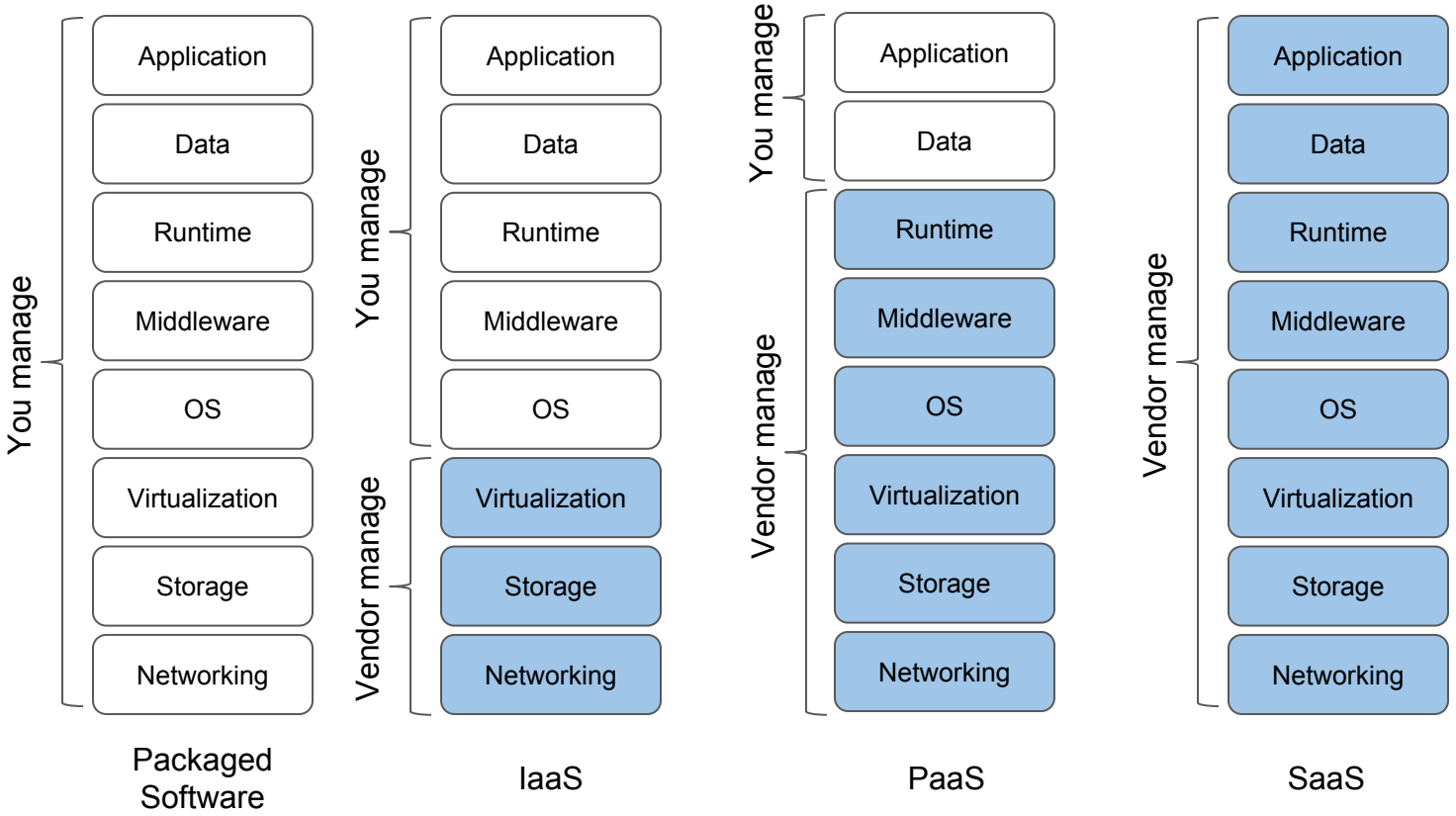
컨테이너 기반의 SaaS/PaaS 인프라 구축 개요

yundream@gmail.com
http://www.joinc.co.kr

다룰 내용

- IaaS, PaaS, SaaS 개요
- VM과 Container의 주 차이점
- 분산 커널 관점에서의 컨테이너의 주요 특징
- 분산 커널 개요
- 분산 커널 환경에서의 컨테이너 스케쥴링
- 분산 커널 환경에서의 컨테이너 네트워크 시스템
- 분산 커널 환경에서의 컨테이너 스토리지 시스템
- 분산 커널 환경에서의 컨테이너 Discovery(Service discovery) 시스템

IaaS & PaaS & SaaS 구분



컨테이너는 프로세스다.

- VM과는 다르다.
- 컨테이너는 “격리된 프로세스"다.
  - 인터넷은 프로세스의 연결이다.
  - 프로세스를 연결하는데 굳이 VM을 띄울 필요는 없다.
  - 유저, CPU, Memory, 디스크, 네트워크를 격리 혹은 제한(limit)한 프로세스를 제공하자.
- 프로세스의 데이터 저장소. 파일 시스템 ? 블럭 스토리지 ?
  - VM(운영체제)과 달리 프로세스는 파일 시스템을 사용한다.
  - 파일 시스템을 격리 할 수 있는가.
  - 블럭 스토리지도 대안으로 사용 할 수 있는가
- 프로세스의 네트워크. 공개된 IP ? Port ?
  - VM과 달리 프로세스는 Port를 사용 한다.
  - Port forwarding과 NAT 기반으로 네트워킹 한다.
  - IP를 대안으로 사용 할 수 있는가 ?

컨테이너 기반의 분산 커널 시스템

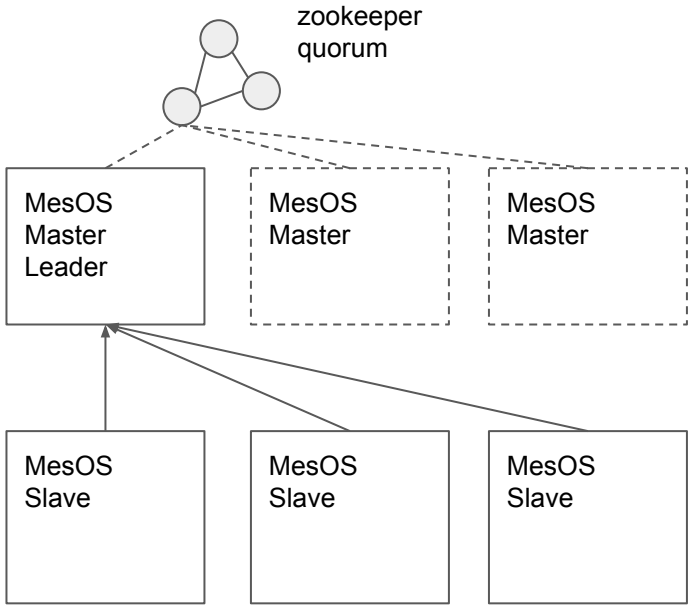
- 커널의 추상화 : 분산된 자원을 관리하는 추상화된 커널
- 리소스 풀을 만들고
  - CPU, Memory, Network, Disk를 하나의 리소스로 관리
- 리소스를 모니터링 및 관리, 스케쥴링을 위한 정보를 생성
- 유저가 요청하면 스케쥴링 정보를 활용 프로세스를 실행
- 프로세스에 작업을 요청. Process discovery
  - 일반 커널 : PID
  - 분산 커널 : DNS를 이용한 Service discovery
- 프로세스의 STOP, CONT, KILL : SIGNAL
- 네트워크 제공 : PIPE
- **MesOS**

바퀴를 새로 만들 필요가 있는가 ?

- MesOS는 Berkeley 대학에서 만든 오픈소스다.
- 대학 연구소 환경에 최적화된 소프트웨어다.
  - Private 환경.
  - 정적 환경.
  - 관리 가능한 소프트웨어(Hadoop, spark)의 대량 배포.
  - LifeCycle를 관리. Batch
- 범용성이 떨어진다.
  - Public 환경
  - 동적인 인터넷 환경
  - 임의의 소프트웨어 배포
- 개발이 쉬워졌다. (정말 ?)
  - 분산 코디네이터
  - 클라우드
  - Docker API
  - 모니터링 툴

컨테이너 기반의 분산 커널 시스템

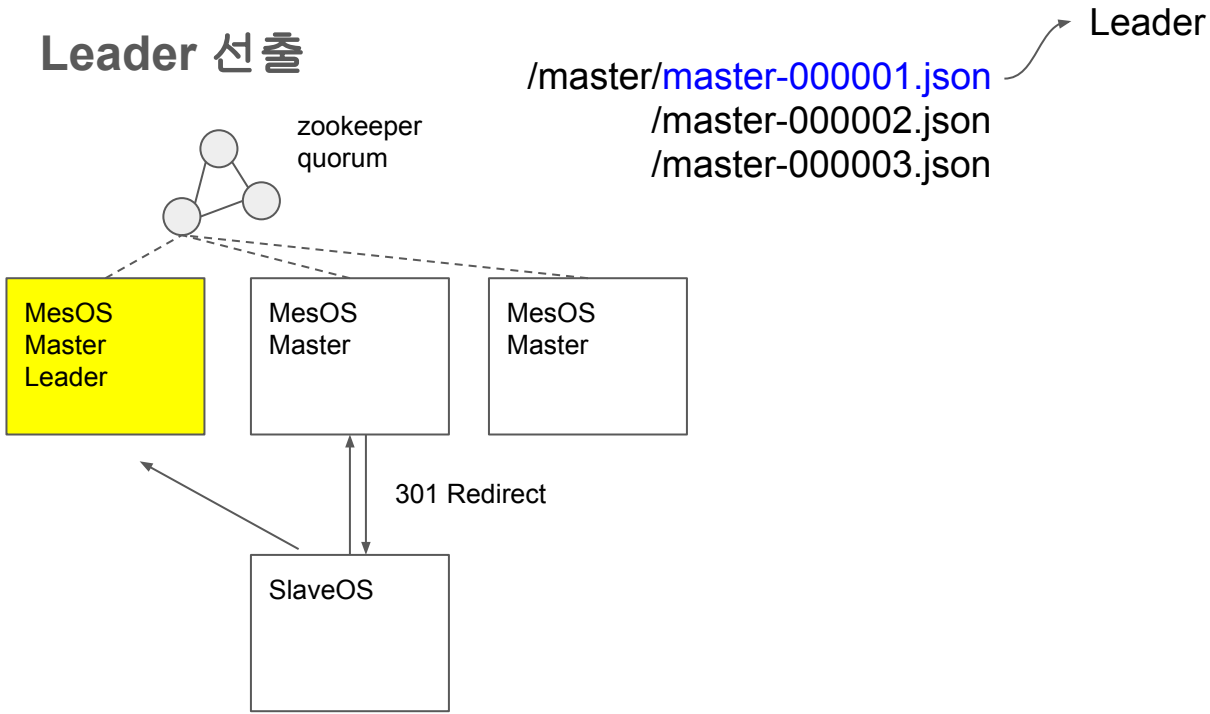
- **MesOS 모델**



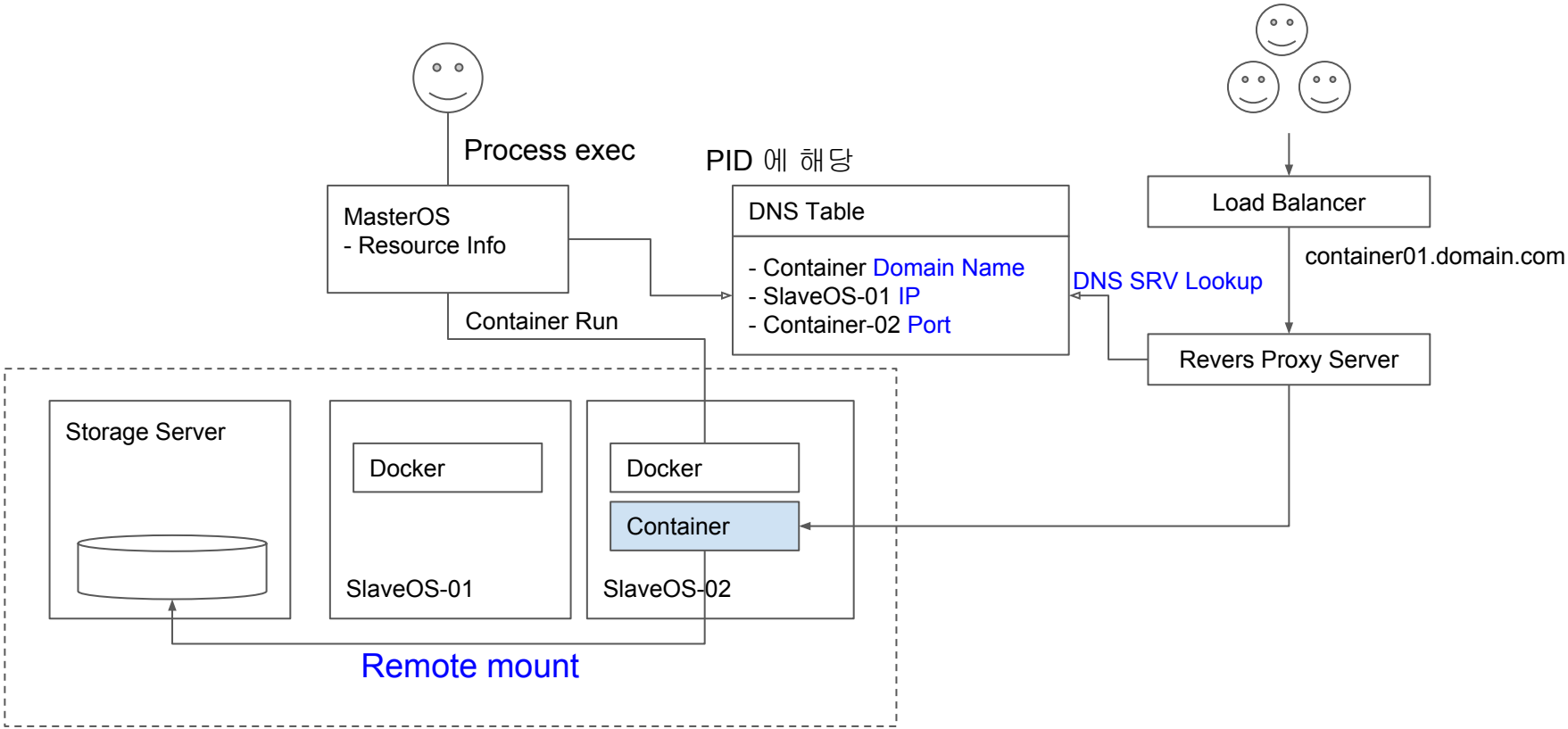
- Slave의 등록 및 자원 리포팅
  - CPU, Memory, Disk
  - Running Container
- 리포팅된 자원을 이용 스케쥴링
- DNS를 이용 프로세스 식별

컨테이너 기반의 분산 커널 시스템

- **Leader 선출**



컨테이너 실행과 접근 프로세스

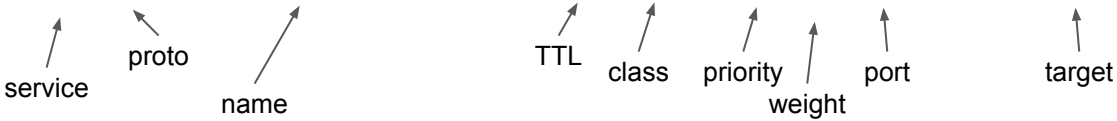


Service Discovery

- 전체 클러스터 노드에서 컨테이너를 식별할 수 있어야 한다.
- 인터넷 서비스에 사용 할 수 있는 DNS를 기반으로 한다.
- 네트워크 모델에 따라서 DNS Lookup 방식이 달라진다.
  - Port forwarding : DNS-SRV 레코드 LookUp
  - Direct IP : DNS A 레코드 LookUp
- 컨테이너 네트워크는 동적으로 바뀐다.
  - 동적인 DNS 시스템 구축

Service Discovery 모델 - Port forwarding

_http._tcp.my-container.domain.com. 86400 IN SRV 0 100 8080 my-container.domain.com.

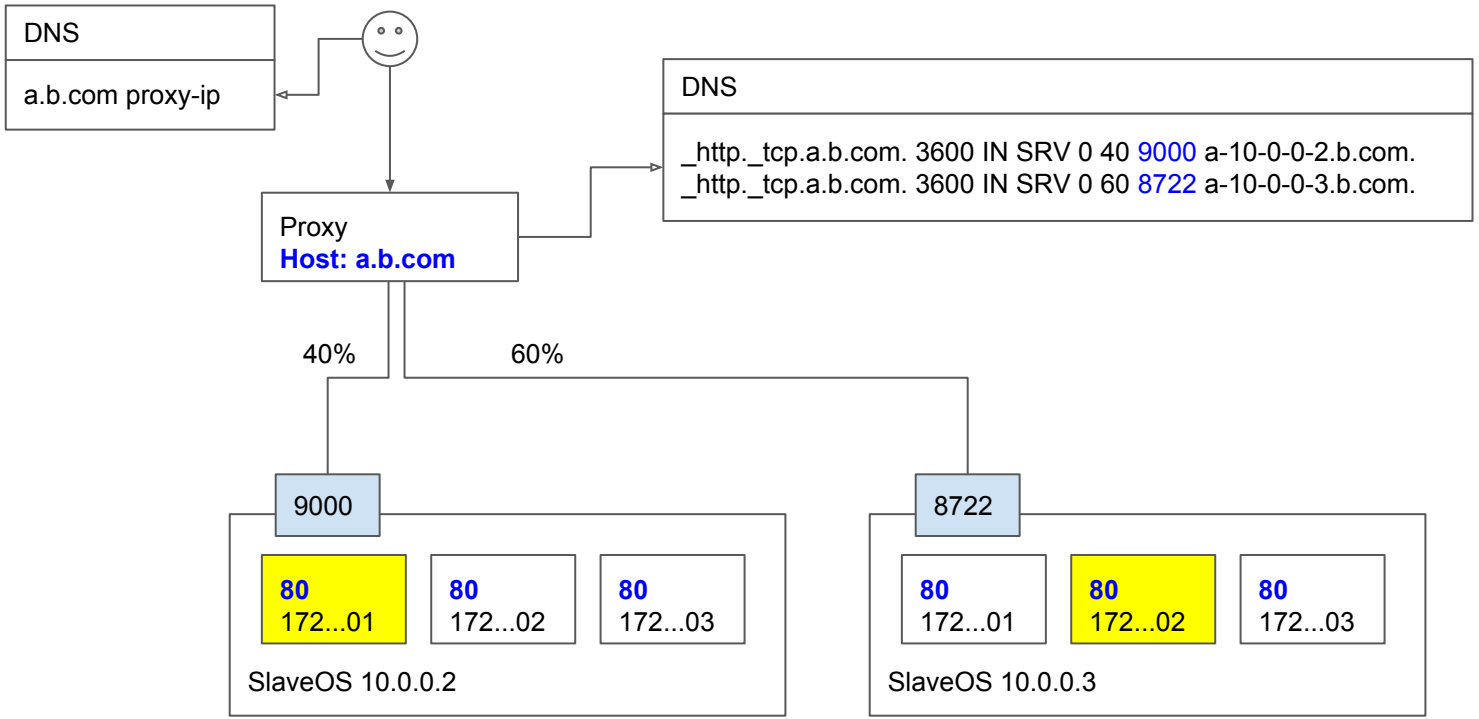


- DNS-SRV(Service Record)를 이용해서 서비스를 찾는다.
- Priority와 Weight를 이용 service backup과 QoS를 설정 할 수 있다.

```
# dig _http._tcp.example.com SRV @127.0.0.1
......
;_http._tcp.example.com.          IN    SRV

;; ANSWER SECTION:
_http._tcp.example.com.   86400   IN   SRV   20 0 8080 backup.example.com.
_http._tcp.example.com.   86400   IN   SRV   10 10 8080 smallbox2.example.com.
_http._tcp.example.com.   86400   IN   SRV   10 10 8086 smallbox2.example.com.
_http._tcp.example.com.   86400   IN   SRV   10 20 8080 smallbox1.example.com.
_http._tcp.example.com.   86400   IN   SRV   10 60 8080 bigbox.example.com.
```

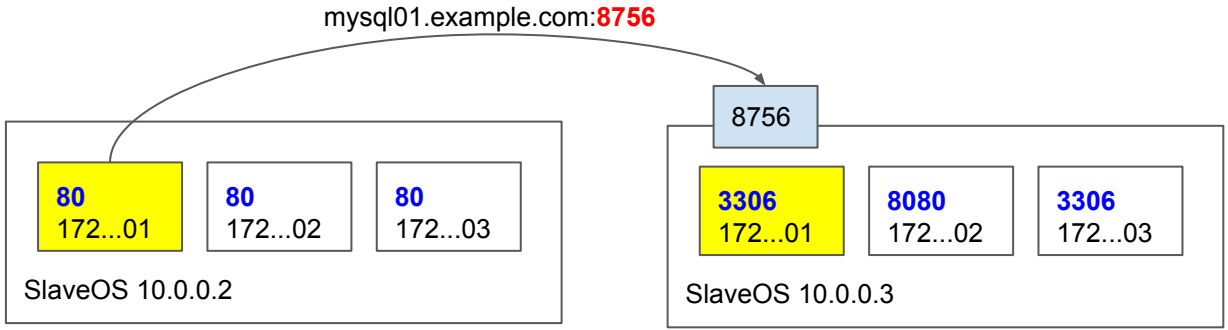
Service Discovery 모델 - Port forwarding



Service Discovery 모델 - Port forwarding

장점 및 단점

- 구현이 쉽다.
- HTTP 기반의 서비스에는 잘 작동한다 : 프로토콜에 Domain Name이 명시
- 응용 애플리케이션의 Port 설정이 어렵다.
  - mysql, mongodb, redis ....
  - Proxy server를 개발 할 수 없다.

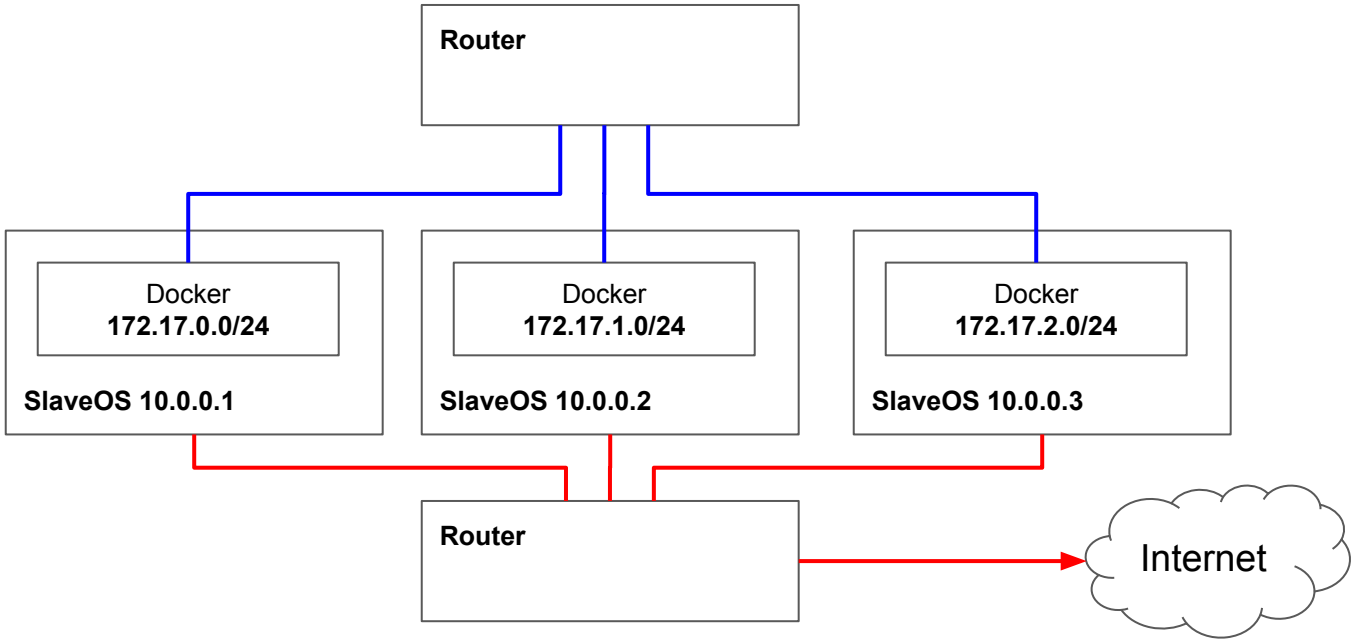


Service Discovery 모델 - Direct IP

- 컨테이너에 IP를 직접 할당.
- IPv4 자원의 한계
  - VM과 달리 많은 수의 컨테이너를 만들 수 있다.
- IPv6
  - 사용이 제한 적이다. (ex. AWS)

Service Discovery 모델 - Direct IP

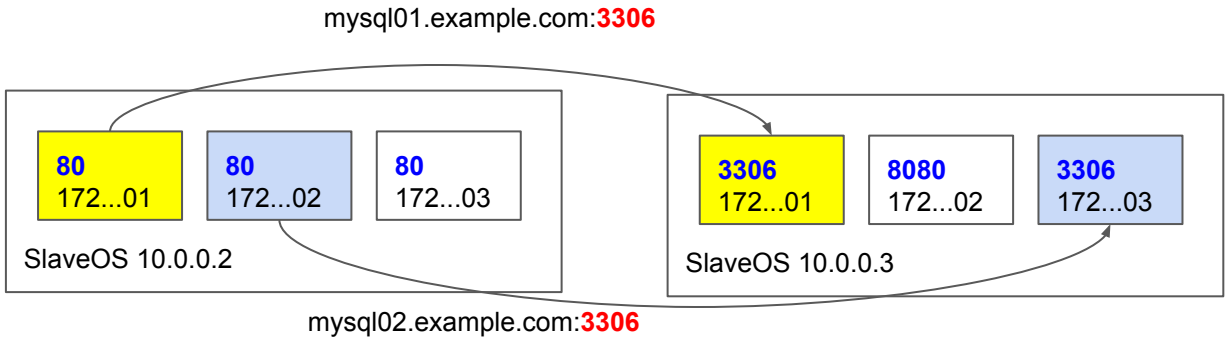
네트워크 구성 수정



Service Discovery 모델 - Direct IP

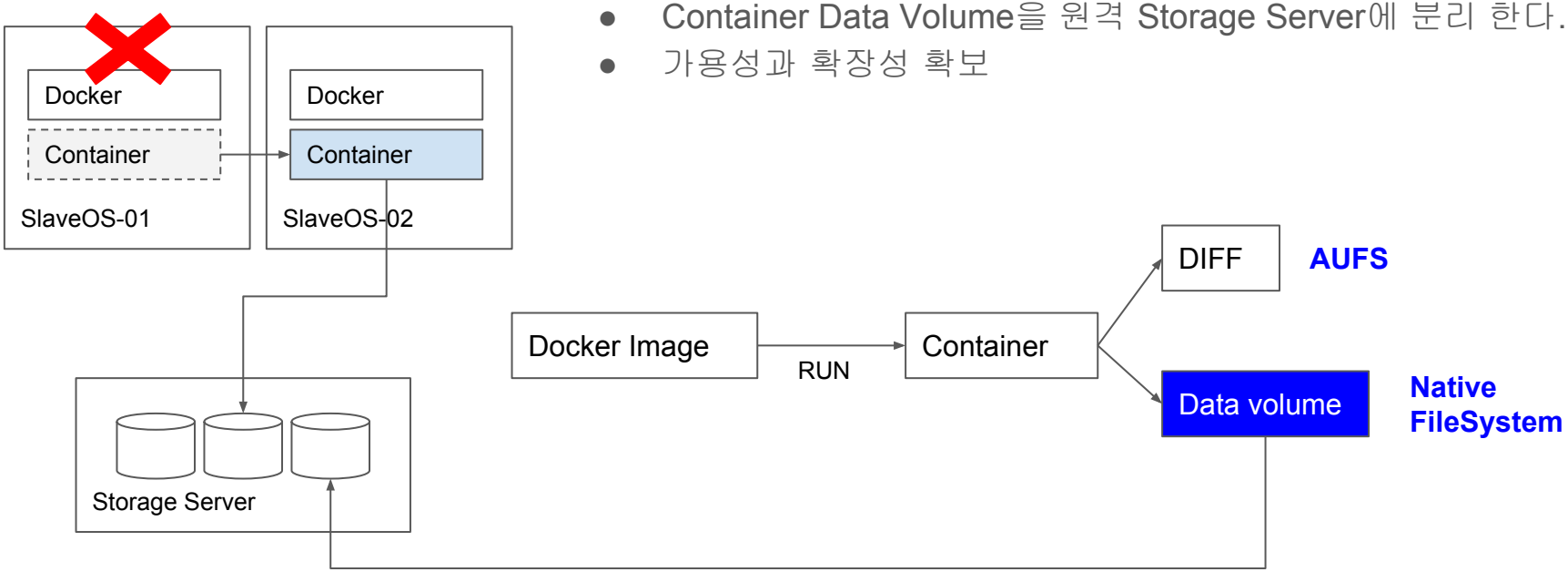
장점 & 단점

- IP가 직접 할당 된다. VM 처럼 자유롭게 Port를 설정 할 수 있다.
- DNS-SRV lookup이 필요 없다.
- Routing Table 관리가 필요하다.



Storage Service 모델

- Container Data Volume을 원격 Storage Server에 분리 한다.
- 가용성과 확장성 확보

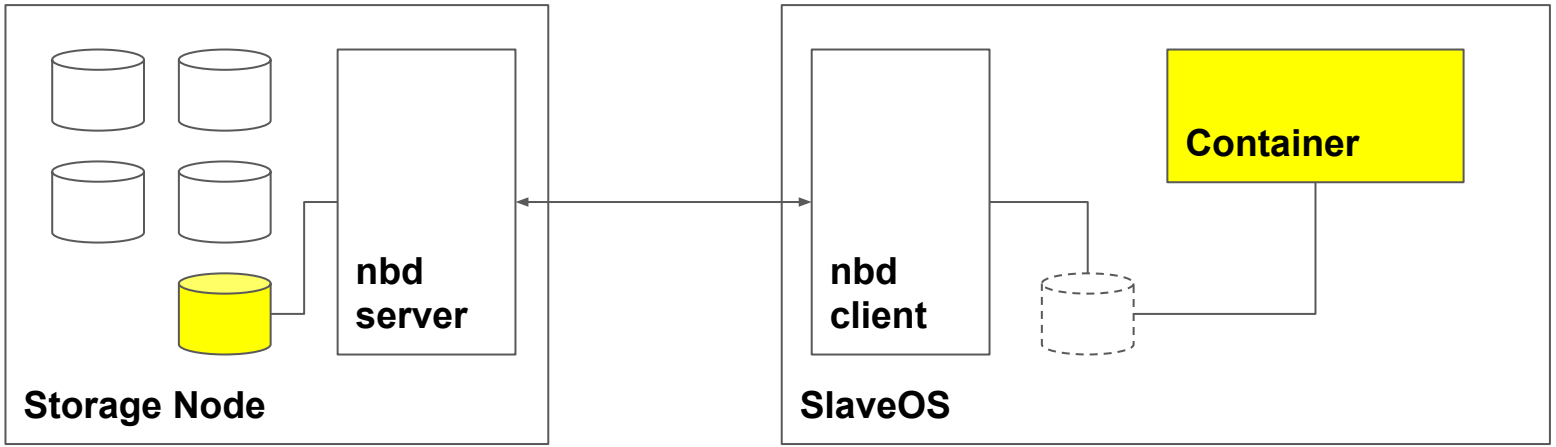


Storage Service 모델 - NFS

- Container는 Process다. 따라서 파일 시스템이 무난해 보인다.
- 그러나 아래의 문제점이 있을 수 있다.
  - 격리가 어렵다.
  - Quota 설정이 어렵다.
  - ZFS나 Btrfs등을 이용 해서 격리 & Quota 설정이 가능하지만 관리가 어려워진다.

Storage Service 모델 - Block device

- Object file 단위로 컨테이너에 제공
- 격리 & Quota 설정이 자유롭다.
- 백업 & 복구 & 마이그레이션등 관리가 쉽다.
- 서버 & 클라이언트 모델의 NBD(Network Block Device)를 적용 할 수 있다.



Storage Service 모델 - Block device

QEMU - qcow2 Image

- Sparse file
- 증분 Snapshot
- Linked clone
- Transparent decompression

```
server # qemu-img create -f qcow2 container-name.img 5G
server # qemu-nbd --bind=0.0.0.0 --port=4000 container-name.img -x container-name
client  # nbd-client -N container-name 10.x.x.x 4000  /dev/nbd0
client  # mount /dev/nbd0  /containers/volumes/container-name
client  # docker run --name=container-name -v /containers/volumes/container-name:/opt/mysql ubuntu
mysql-server
```

끝