DC/OS

1. Operational : Linux node에서 작동되는 application의 software-defined configuration과 automation의 위한 system.
2. Development : DC/OS내부의 service(component)를 활용하는 application의 분산 시스템을 제공하는platform

* DC/OS를 node에서 동작하는 유일한 application이라고 생각할 수 있다.
* DC/OS가 설치된 node들은 cluster를 만들기 위해 다른 node와 통신한다
* DC/OS가 설치되면 이를 통해 application을 동작하고 관리할 수 있다.
* 소프트웨어가 configuration을 관리, 정의할 수 있게한 cluster수준의 OS이다.
  + Chef와 Puppet과 같은 provisioning tool과는 다르다.
* Mesos attribute 와 Mesos ROLE은 따로 설치 중 설정할 수 있다.
  + Scheduling 관련
* Task scheduling과 resource allocate에 Mesos를 사용

작동방식

* Master와 agent로 구성(설치시 결정)
* 여러 component로 구성되며 각각은 하나의 application으로 systemd unit이다.
* Master와 agent는 다른 component set을 가지지만 각각은 모두 동일한 component set를 가진다.(DC/OS Configuration 은 두가지만 존재)

Component

* Zookeeper
* Admin Router
* Metronome
* Marathon
* Mesos DNS
* Cosmos

Master node

* Cluster의 중재자 역할을 하며 cluster의 configuration을 저장함.
* Master node들의 leader는 Zookeeper의 election을 통해 결정
* Leader는 Cluster의 상태를 변경시킬 수 있으며 이를 다른 master(quorum)에 복사
* Leader가 죽으면 다른 master 중 leader를 선택
  + 일은 leader만, leader가 아닌 master는 cluster에 관여하지 않음
* DC/OS의 master는 Mesos의 master이기도 함
* Master에는 충분한 memory가 필요
  + Master는 cluster의 상태를 memory에 올린다.

Agent node

* Agent는 master에게 가능한 resource를 제공
* Master는 resource에 task를 할당
* Resource에 특정 role을 지정 가능
  + 특정 application만 해당 resource 사용가능
* 언제든 cluster에 새로 추가될 수 있다.

Mesos

* agent node에서 low-level 의 task assignment와 execution을 담당
* cgroup을 사용하여 scheduler가 정한 resource만 task에 할당
  + - * cgroup : resource활용에 관련된 Linux kernel기술, 동일 node에 다양한 task가 구동할 수 있게 함.
  + 여러개의 workload를 하나의 Cluster에서 구동 가능.
  + Framework를 사용하면 자체 failure handling사용 가능.
* Task의 복구와 loss of agent를 처리하지 않음
* Agent node resource를 추상적인 집합으로 제공

Framework

* Framework를 통해서 Mesos는 task를 받는다
* Resource offer를 받고 그에 따른 task를 할당하는데 Mesos API를 사용.
* Master가 agent의 resource를 확인 > Framework로 master는 resource offer를 한다.

> Framework는 master에게 agent가 task를 실행하도록 함

Mesos Attribute and Role

* Attribute는 key-value형태로 resource offer와 같이 framework에 전달

(Framework에서 사용여부 결정 가능)

ex) rack : 물리적인 rack 이름, 동일 task를 같은 이름의 rack에 부여 X

* + - * One-point failure
* Role은 해당 node resource와 같은 role의 Framework 에게만 제공
  + Offer에 같이 제공
  + Network portioning과 같은 static portioning이 가능

DC/OS task instance

* 공통된 requirement에 따른 정형화된 실행방식을 제공

1. Apps

* Marathon에 의해서 장시간 실행을 보장
* Fail-over을 보장

1. Job

* Metronome에 의해서 일회성 실행

1. Package

* DC/OS의 common service를 위해 만들어진 app definition 이다.
* CLI plug-in을 가지고 있어 DC/OS CLI를 통해 관리 가능
* High-Level service abstraction을 지원(database, framework)
* DC/OS GUI or CLI를 통해서 설치 가능
* Scheduler application은Marathon app처럼 동작하며, 이후 master와 직접 통신하며 Marathon과 독립적으로 task를 관리한다.
* 어느 시점에서든 설치가능
* DC/OS CLI
  + DC/OS를 관리하기 위해 어느 pc에나 설치가능한 application

(cluster외부에서 가능)

* + 직접 사용과 Scrip로 관리 가능
  + node 관리 package 설치/삭제 cluster 상태 확인을 할 수 있다.

Running Application

* Marathon(app) : always-on application(Web server, CRUD service)
* Metronone(job) : one-off application

공통점 : docker support, health check, failure handling, change management

Mesos sandbox의 code를 실행할 수 도 있고 container를 실행할 수 있다

* data를 남기거나(debuge목적) garbage collector에 의해 삭제할 수 있다.
* Marathon Pods
* 하나의 agent에서 실행될 여러 app을 Pod라는 것으로 묶을 수 있다
* 상호의존적인 app끼리 동시에 실행되며 resource를 공유할 수 있게 한다
* Pod의 task는 network interface를 공유하며 localhost interface로 통신할 수 있다
* Fail Handling in Marathon
* Marathon을 통해 health check가 가능하다.
* Health check 설정되어 있지 않으면 Mesos state를 health check로 사용한다
* 다음의 protocol을 사용할수 있다
  + HTTP, HTTPS
  + TCP
  + Command
  + MESOS\_HTTP, MESOS\_HTTPS
* Marathon은 설정된 수의 app instance를 healthy state로 두기 위해 failing task를 교체한다.
  + Crash looping : task가 fail과 replace의 loop에 고착
* Node와 통신이 끊어졌을 때 , Marathon은 reschedule될 lost node의 task의 비율을 제한
  + 일시적 문제인지 영구적인지 판별 X, 과부화 방지
  + 다수agent의 shut down상황에서 cluster state를 회복하기위해 많은 시간이 소요
    - * Marathon을 사용하지 않는 instance사용
  + Marathon에 의해서 실행되는 task는 Marathon과 통신이 끊나도 실행 지속
* Marathon의 change Management
  + Marathon은 app의 change handling을 위해서 Mesos로 얻은 정보와 health check를 사용
    - App configuration change가 발생시, 기존 app을 종료하지 않고 새로운 app을 실행
    - 새 app이 실행되고 health check를 보낼 시 기존의 app을 제거
      * downtime없이 version change가 가능

Writing Application

* Communication
* Storage
* Service Discovery(communication)
  + 실행 중 target service를 동적으로 확인 통신하는 mechanism ex) DNS
  + DC/OS 내부에 있는 어느 node가 target service를 실행하고 있을지 모른다
    - VIP(virtual IP)
    - Mesos DNS
* VIP
  + IP + PORT -> Single Virtual IP
  + Name-based (numeric X)
  + Private Virtual IP + PORT + Service name

Ex) <service-name>. marathon.l4lb.thisdcos.directory:<PORT> (Marathon app)

* + Minuteman에 의해서 제공
  + 핵심 component로서, node의 routing table을 cluster state에 맞춤
  + property
    - Load Balancing(TCP only)
    - Allow drop-in replace for address in legacy system
    - Do no check health care
    - Task change 반응에 10초 정도 소요(failed node에 대해 반응하는데 10초 걸림)
    - Minuteman이 설치되어 있어야함
* Mesos DNS
  + 모든 Mesos Master master는 Mesos DNS를 실행
  + agent는 DNS로 master를 사용하도록 설정됨
  + Mesos DNS는 master에 대한 polling을 30초마다 진행하여 record를 채운다
    - **폴링**(polling) : 하나의 장치(또는 프로그램)가 충돌 회피 또는 동기화 처리 등을 목적으로 다른 장치(또는 프로그램)의 상태를 주기적으로 검사하여 일정한 조건을 만족할 때 송수신 등의 자료처리를 하는 방식을 말한다.
    - Ex) leader.mesos -> leader 찾는데 사용

<app id>.marathon.mesos ->marathon app (A reord)

<app id>.<tcp|upd>.marathon.mesos -> app IP + PORT(SRV record)

* + load balancing 지원 X
  + SRV record의 사용은 별도의 process를 처리할 여분의 Mesos DNS가 필요
  + A record는 PORT고정 (hard cording configuration)이 필요
  + Health check X
  + 30초 Polling -> change handle is slow
  + 모든 agent가 하나의 master를 DNS로 사용 -> scalability 한계 존재
  + 다수의 A Record를 받게 되더라고 failure handling X
  + 모든 task의 주소를 알 수 있다(Minuteman만 가능한 VIP)
  + DNS quary 대신 HTTP request를 날릴 수 있다.
    - * 추천 : Marathon LB

Persistent State

* DC/OS는 Mesos Persist Volume을 Package로 제공
  + 내부 상태를 기본적으로는 저장 X(sandbox, container)
  + 독립적으로 작동, database나 file system모두 사용 가능

Local Persistent

* + Task가 최초로 launch가 될 때, 요청한 만큼의 저장공간을 제공
  + 이후 이 공간을 사용하는 task는 해당 node에서만 실행
    - * Flexibility, fail handling 감소
  + Marathon을 통해 쉽게 사용가능
  + CPU or RAM과 같은 다른 resource가 부족하면 실행 X

External Persistent

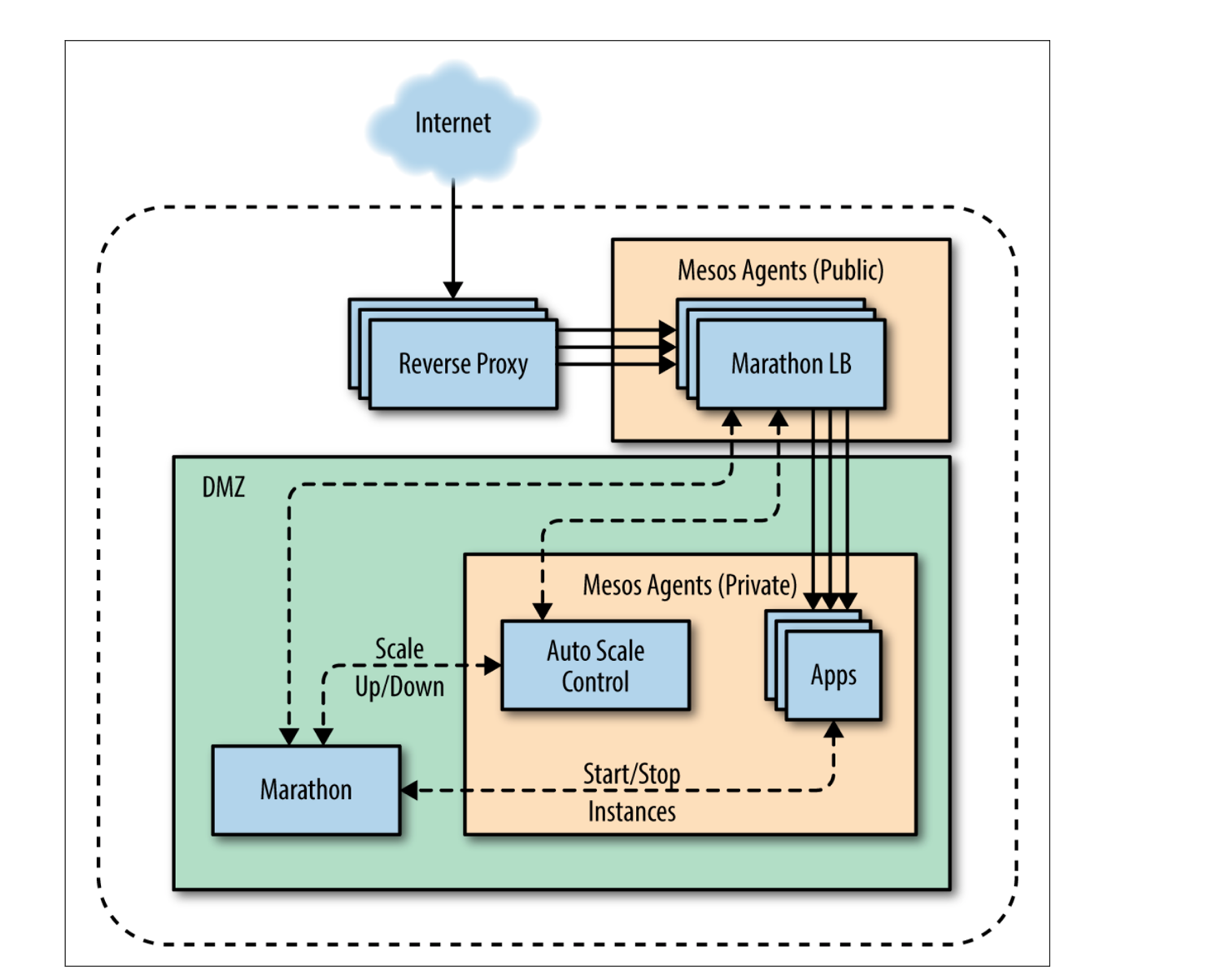
* + 특정 node에 종속되지 않고 task reschedule이 가능
  + runtime에 Docker나 Mesos container에 mount할 수 있는 REX-Ray기술 사용
  + Marathon을 사용하면 Persistent에 접근하는 instance 하나만 생성

Storage Pacakge

* + Redundancy를 위해 여러 node에 저장
  + Redundancy를 app에 database나 file system으로 공개
  + 여러 persistent volume을 만들어 하나가 고장시 이를 다른 것으로 대체하는 것이 일반적
  + lost persistent volume을 교체하는 mechanism을 가지고 있다.
  + Lost node를 교체하는 것은 수동적으로 이루어진다
    - 해당 node의 data를 가진 다른 node로 부터 transfer해야함
    - 장시간이 소요, 일시적인 손실인지 영구적인 손실인지 판별 불가
* self-healing이 안됨

Publishing Application

* Revers poxy server 사용을 추천
  + task의 위치가 수시로 바뀔 수 있음



* Revers Proxy server
  + Server와 Client 사이에서Client의 요청을 중개
  + 요청정보를 cache하기 때문에 재요청시 시간 감소.
  + DMZ로의 유일한 통로로 보안 강화
  + Load balancing을 구현