ZOOKEEPER

발표자 : 이준광

목차

1. 주키퍼란?

2. 주키퍼의 기본구조

3. 주키퍼 설치

4. 주키퍼 znode에 데이터 넣기

5. 주키퍼 활용법 및 분산 락 구현

1. 주키퍼란?

1.1 주키퍼의 역할

-최근 나온 오픈소스들 무리의 네이밍이 동물이름을 채택하고 있는 아파치.. 대표적인 하둡,콤캣,카멜,하이브,피그등 동물이름을 붙여진 동물들의 분산처리를 관리하는 역할을 하기위해 고안했다.(주키퍼는 별도의 장비로 설치해도 되지만 Hadoop의 Datanode/Namenode에 설치해도 무방하다)

1.2 주키퍼의 사용 이유

- 분산처리 환경에서는 기본적으로 서버가 몇 대에서 수십대, 수백 대까지도 갈 수 있다.

분산 처리 환경에서는 예상치 못하는 예외적인 부분이 많이 발생하게 되는데 주로 **네트워크 장애, 일부 서비스/기능 예상치 못한 처리로 중지나 장애, 서비스 업그레이드, 서버 확장 등에 문제가 발생 할 수 있다**.

쉽게 이해가 안되는경우 프로그램적으로 생각 해도 된다. 예로 싱글 쓰레드만 존재하는 프로그램에서 멀티 쓰레드 프로그램을 하게 될 경우 싱글 쓰레드에서 이상없던 동기성 문제등이 나타나기 시작합니다. 분산처리도 마찬가지이다. 하나만 처리하던 싱글 서버에서는 문제가 되지 않으나 멀티서버를 관리르 해야 한다면 여러 문제점들이 발생 할 수 있다.

따라서 이러한 점들을 쉽게 해결 할 수 있는 시스템이 바로 주키퍼 이다.

1.주키퍼란?(특징)

1.3 주키퍼의 특징

- 네임서비스를 통한 부하분산

하나의 클라이언트(하나의 서버)만 서비스를 수행하지 않고 알맞게 분산하여 각각의 클라이언트들이 동시 작업할 수 있도록 지원

- 분산락이나 동화 문제 해결

하나의 서버에서 처리된 결과가 또 다른 서버들과 동기화하여 데이터 안정성 보장

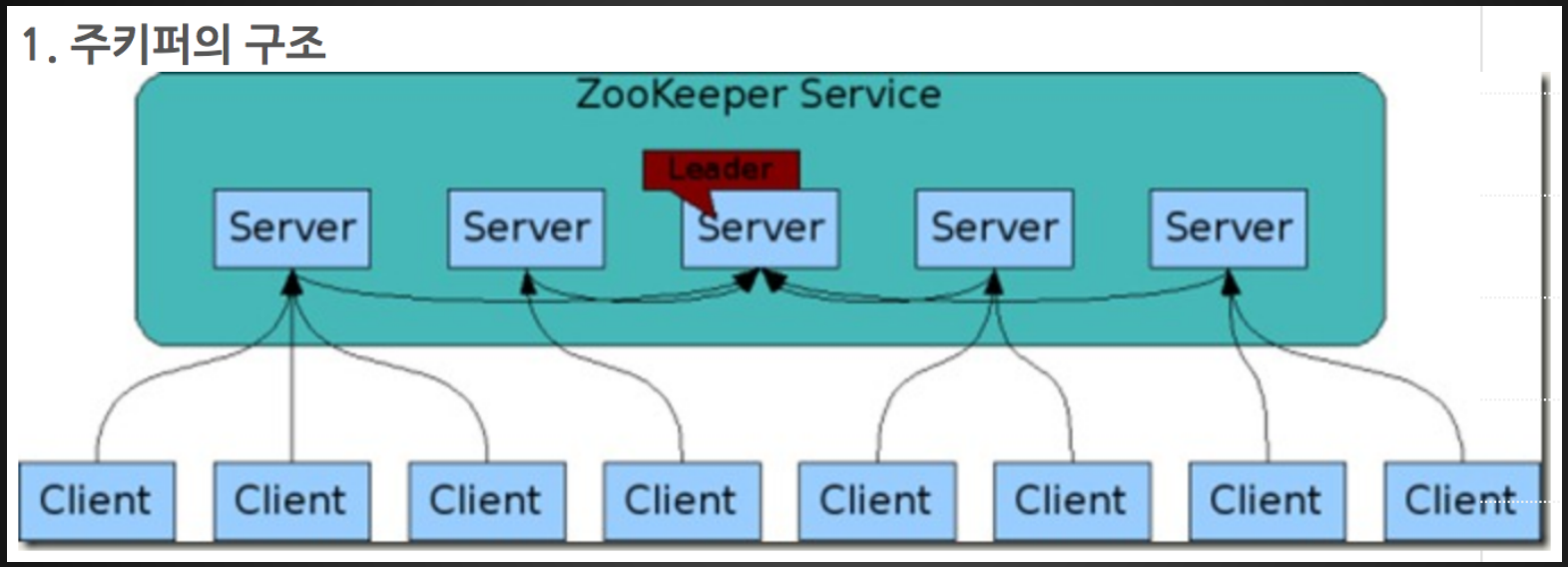
- 장애상황 판단 및 복구

액티브(일반서버)서버가 예기치 못한 상황으로 문제가 발생하여 서비스를 지속적으로 처리를 못 할 경우 스텐바이(일반서버)서버가 액티브 서버로 바뀌어서 기존에 액티브 서버가 바뀌어서 기존에 액티브 서버가 서비스를 하던 일을 처리하게 된다.

- 환경설정 관리

각각의 다른 서버들을 통합적으로 관리하여 환경설정을 따로 분산하지 않고 주키퍼 자체적으로 관리하게 된다.

2. 주키퍼 기본 구조



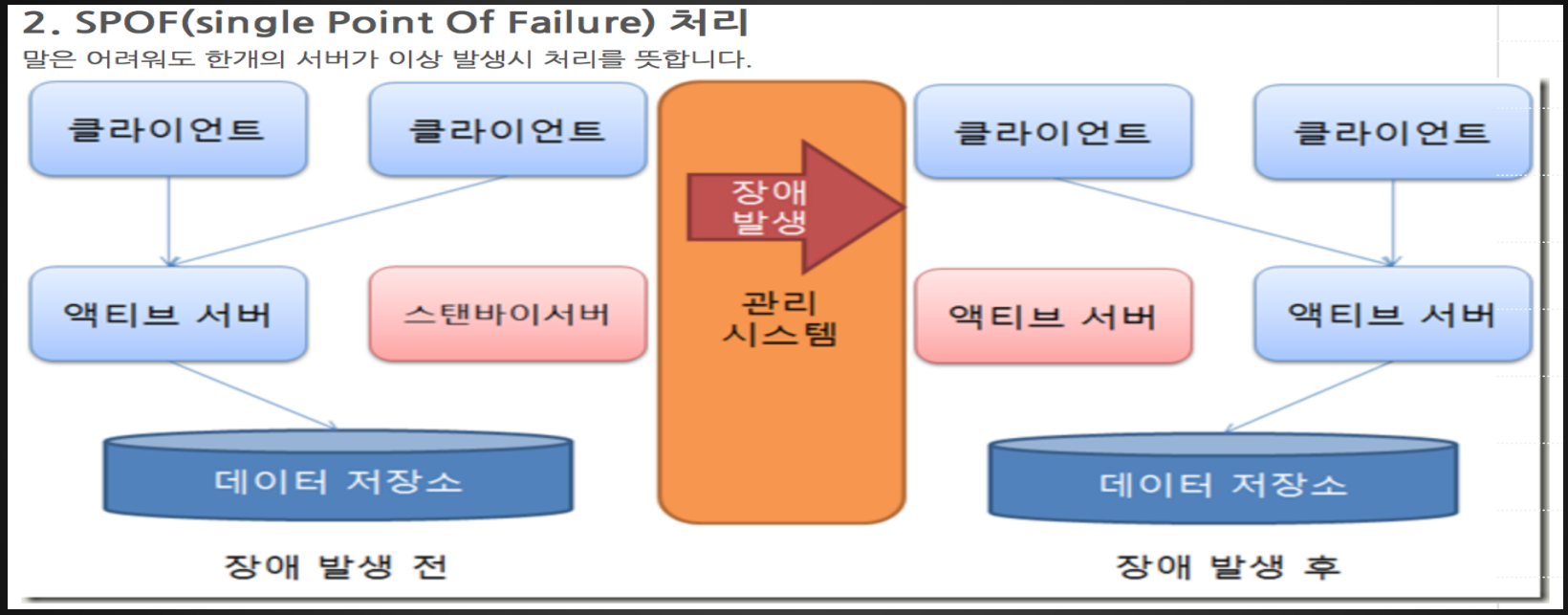
주키퍼는 다중의 서버의 집합을 묶어서 관리해 주는 시스템입니다. 그 중에서 리더가 되는 서버 하나가 존재 합니다.

리더라고 불리는 서버는 모든 서버에 중심이 되는 곳입니다.

또한 하나의 서버에서 처리가 되어 데이터가 변경되면 모든 서버에 동기화를 하게 됩니다.

주키퍼는 하나의 분산처리 서버 입니다.

2.2 주키퍼 기본 구조

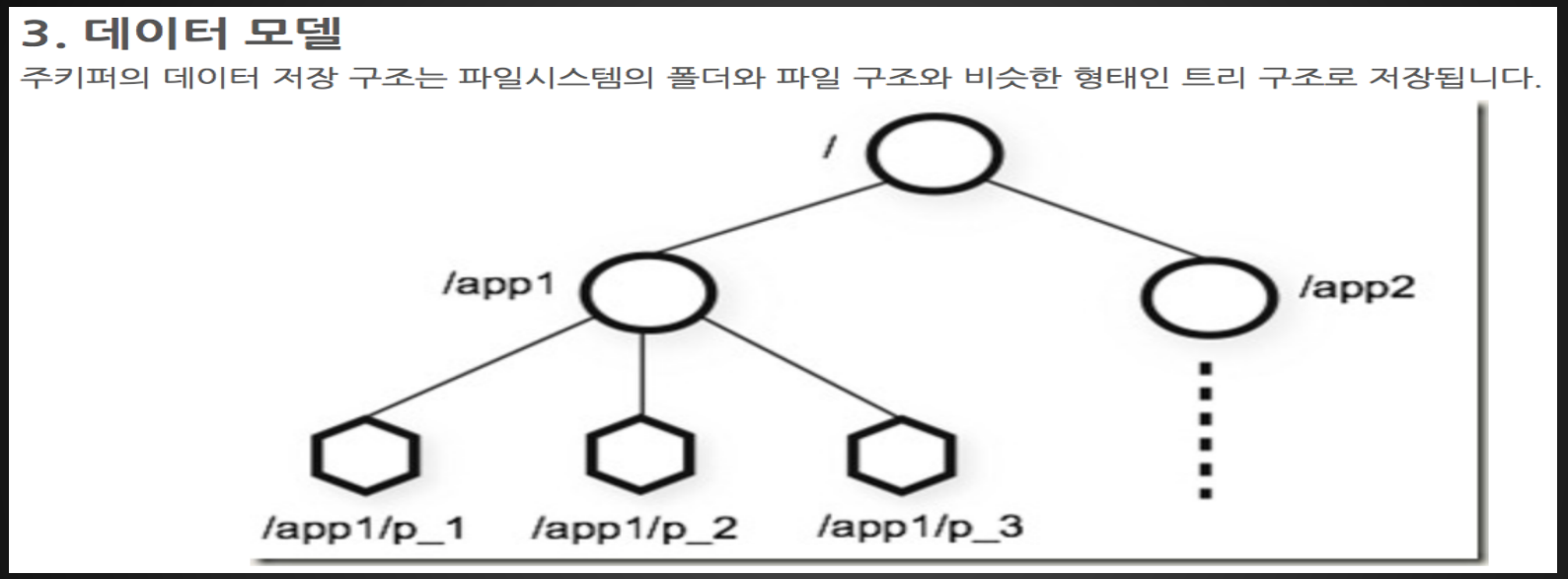


액티브 서버 : 현재 서비스를 하고있는 서버

스탠바이 서버 : 장애 발생시 대처하기 위한 대기 서버

동작중이던 액티브 서버가 장애가 발생 할 경우 관리시스템이 판단하여 기존의 스탠바이 서버를 액티브 서버로 전환하여 서비스를 처리하는 모습입니다.

2.3 주키퍼 기본 구조



/app1 : 하나의 폴더로 생각

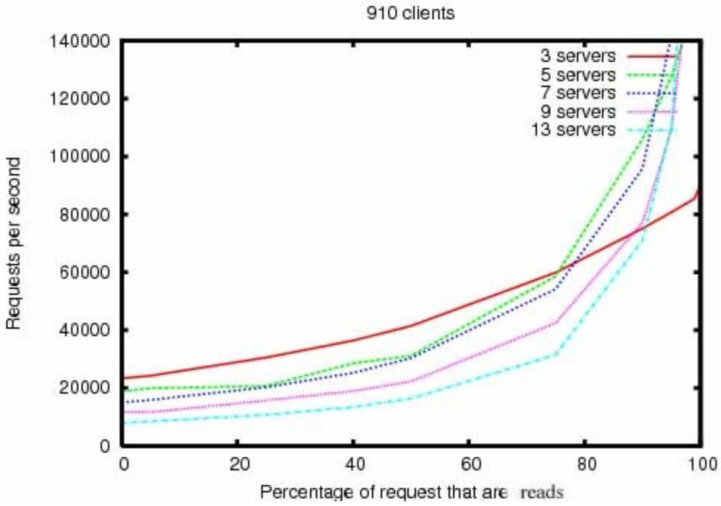
/app1/p_1 : 하나의 노드로 데이터가 저장되는 znode

계층적인 네임스페이스이며 각각의 노드들은 z노드 (znode)라고 합니다. znode는 다음과 같은 특징이 있다

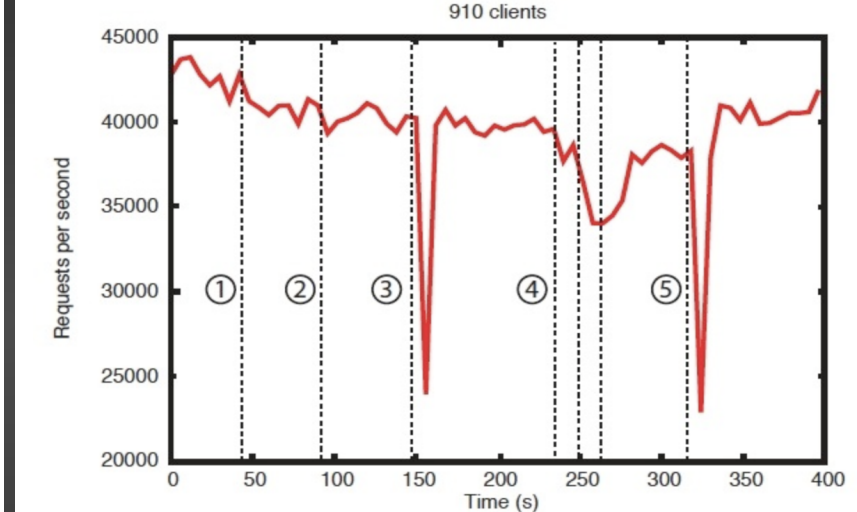
- 모든 노드에 저장 할 수 있다.(부모)
- 크기가 작은 데이터로 저장된다.
- znode는 저장되는 순간부터 버전을 가지고 있다.
- 접근 권한이 있다.

2.4 주키퍼 기본 구조

4. 성능



5. 신뢰성



- 주키퍼는 매우 빠른속도로 동작 됩니다. 하지만 서버수가 증가함에 따라서 성능은 저하되는 것을 볼 수 있다.

- 주키퍼의 서버가 이상 발생시 SPOF문제를 제대로 처리 하는지 나타내는 그래프 이다. 위의 그래프에서 이상하게 처리 능력이 하라 하는 부분이 있는데 바로 저 부분이 문제가 발생되어 주키퍼가 복구하여 지속적인 서비스를 하게 되는 부분이다.

3. 주키퍼 설치

1. 다운로드
 - wget http://mirror.apache-kr.org/zookeeper/stable/zookeeper-3.4.5.tar.gz
   tar zxvf zookeeper-3.4.5.tar.gz
2. 환경설정
 - cd zookeeper-3.4.5
 - cp conf/zoo_sample.cfg conf/zoo.cfg
 - vi zoo.cfg
3. Standalone모드
 - tickTime = 2000
 - dataDir =/zookeeper
 - clientPort=2181
4.Replicated모드
 - standalone모드와 윗부분은 동일
 - initLimit = 5
 - syncLimit = 2
 - server.1 = 192.168.0.1.2888:3888
 - server.2 = 192.168.0.2.2888:3888 ……

4. 주키퍼 znode에 데이터 넣기

1. /zk_test node를 생성하고 test_data이름을 가지게 된다.
 - create /zk_test test_data

2. ls로 zk_test목록에 추가된 것을 확인 합니다.

3. get 으로 zk_test를 확인 한다.
 - get /zk_test

4. 테스트로 만든 /zk_test를 삭제한다.
 - delete /zk_test

```
[zk: localhost:2181(CONNECTED) 2] get /zookeeper

cZxid = 0x0
ctime = Thu Jan 01 09:00:00 KST 1970
mZxid = 0x0
mtime = Thu Jan 01 09:00:00 KST 1970
pZxid = 0x0
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 0
numChildren = 1
[zk: localhost:2181(CONNECTED) 3]
```

```
[zk: localhost:2181(CONNECTED) 5] get /zk_test
test_data
cZxid = 0x10
ctime = Thu Jan 27 14:10:41 KST 2011
mZxid = 0x10
mtime = Thu Jan 27 14:10:41 KST 2011
pZxid = 0x10
cversion = 0
dataVersion = 0
aclVersion = 0
ephemeralOwner = 0x0
dataLength = 9
numChildren = 0
[zk: localhost:2181(CONNECTED) 6]
```

5. 주키퍼 활용 및 분산락 구현

* 주키퍼는 서버를 등록하고 znode를 활용하여 등록 삭제 등 하게 되는데 그게 전부 이다.

* 주키퍼를 사용하는 이유는 락 기능, 서버 분산 기능을 직접적으로 지원을 안 한다. 그냥 그런 기능을 구현 할 수 있도록 일부 기능들을 제공 해주고 그 기능을 사용하여 우리가 직접 분산 락, 서버 구현을 해야 한다.

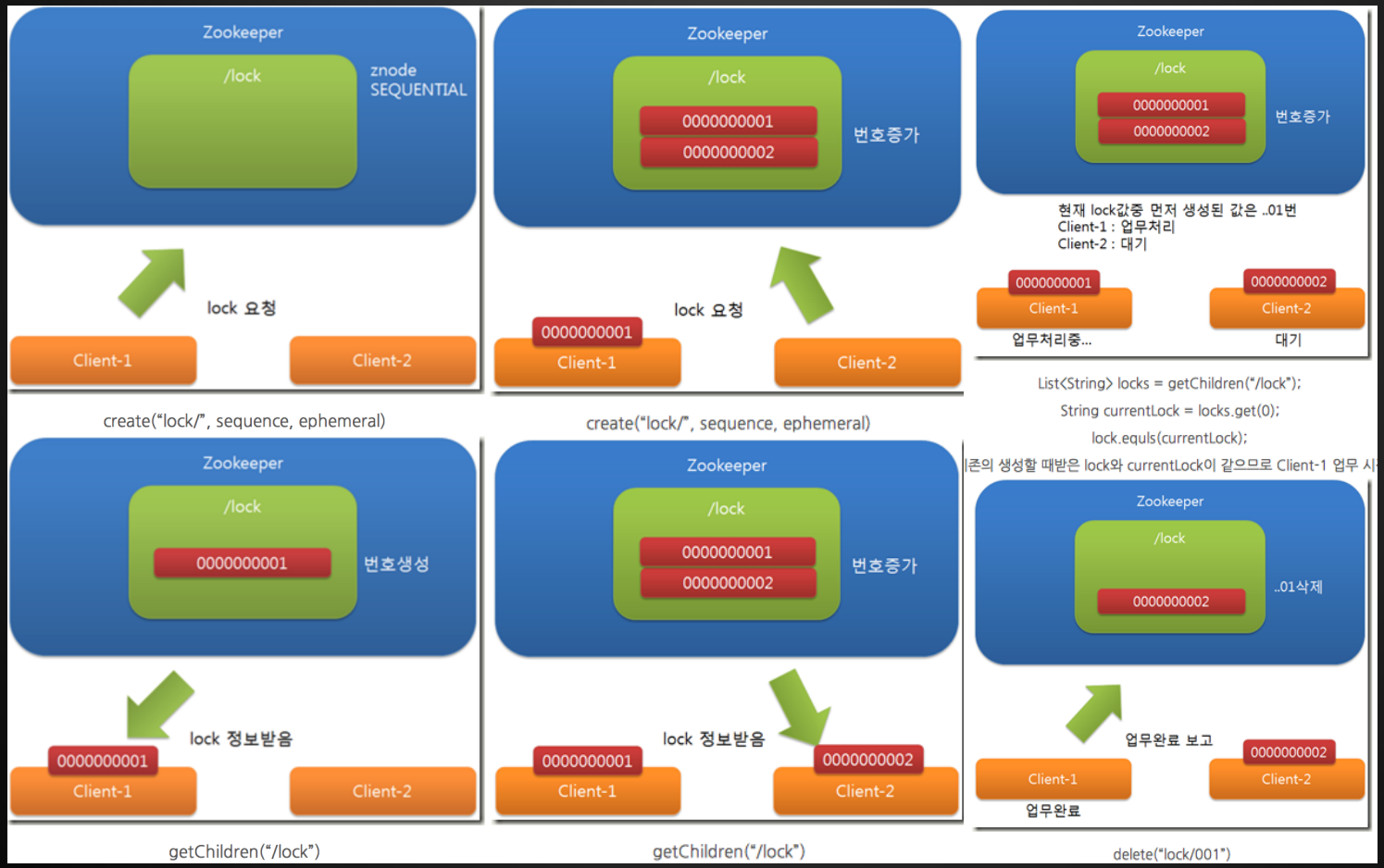
5.1 분산락 구현 법

- 분산 락을 구현하는 순서를 lock이라는 부모 노드에서 하나씩 만들어가면서 추가되는 방식을 예로 보겠다.

순서	Client-1	Client-2	Directory
1	부모 노드 lock 생성		
2	create("lock/", sequence, ephemeral) -001 생성	create("lock/", sequence, ephemeral) -002생성	/lock /001 /002
3	getChildren("/lock")	getChildren("/lock")	
	List("001", "002")	List("001", "002")	
4	작업 처리	exist("lock", watcher)	
5	delete("lock/001")		/lock /002
		3번 작업 으로 시작	

5. 주키퍼 활용 및 분산락 구현

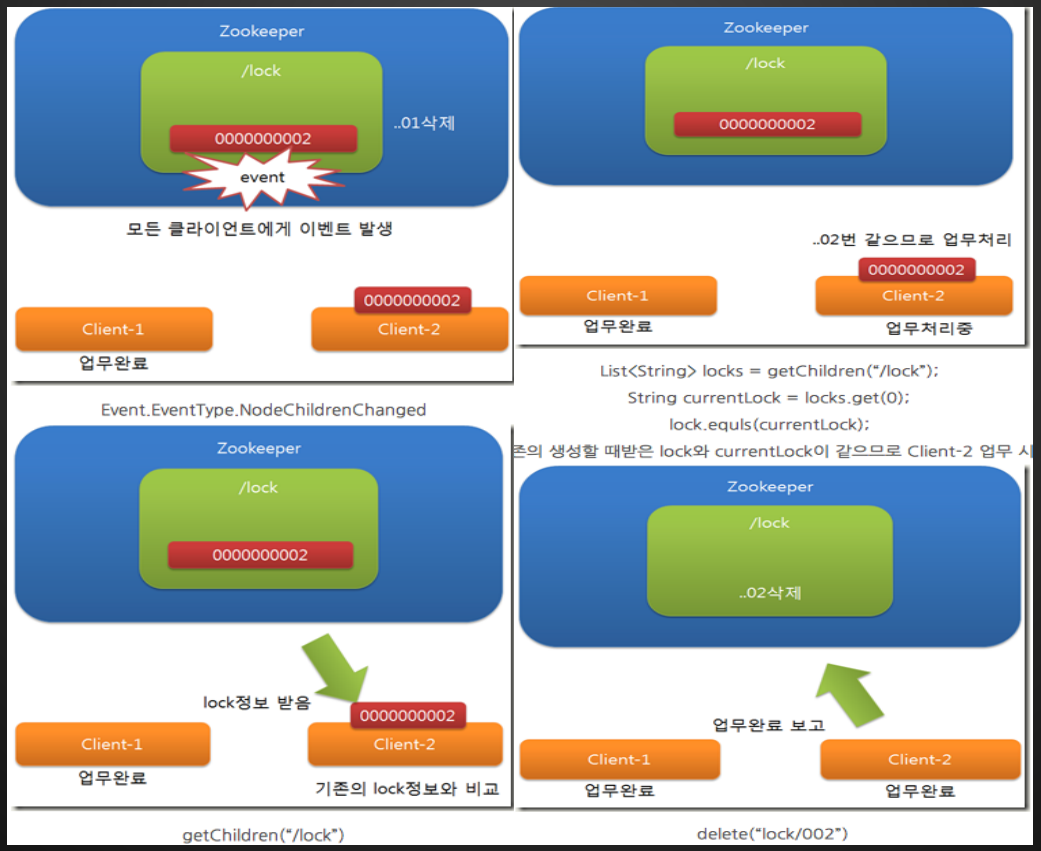
5.2 분산락 작동 순서



5. 주키퍼 활용 및 분산락 구현

5.3 결과

- 주키퍼는 락을 직접적으로 지원을 해주지 않는다. 바로 node와 watcher기능 이용 하여 분산락을 직접 구현을 해야한다.



감사합니당~