9장. 센티널

> 장애 상황을 대비하기 위한 레디스 자체 고가용성 기능인 센티널에 대해 알아봅시다.

센티널은 . . .

센티널 자체의 SPOF를 방지하기 위해 **최소 3대 이상일 때 정상 동작하도록 설계**됨

클라이언트는 센티널에 먼저 연결해 마스터의 정보를 받고 마스터로 직접 연결하여 통신함

- **모니터링** : 마스터, 복제본 인스턴스의 상태를 실시간으로 확인
- **자동 페일오버** : 마스터 장애 시 복제본 중 하나를 마스터로 승격시킴
  - 레디스가 SPOF가 되는 것을 방지하기 위함
- **인스턴스 구성 정보 안내** : 클라이언트에게 현재 마스터 정보를 알려줌

쿼럼

마스터가 비정상 동작을 한다는 것에 동의해야 하는 센티널의 수

- 쿼럼을 만족하면 페일오버를 시작
- 일반적으로 센티널 인스턴스가 3이면 쿼럼은 2로 설정
- 쿼럼을 이용한 `과반수 선출 개념`을 사용하기 때문에 **센티널 인스턴스를 3대 이상의 홀수로 구성하는 것을 권장**

센티널 인스턴스

배치하기

물리적으로 서로 영향받지 않는 서버에서 실행되는 것을 권장

복제본이 2대까지 필요하지 않다면 1대 서버는 센티널 프로세스만 실행시키도록 배치 가능

이 때 센티널 프로세스만 실행하는 서버는 최저 사양의 스펙으로 구성해도 👌

실행하기

1. 마스터 - 복제본 간 복제 연결이 된 상태로 만들기

```
REPLICAOF 192.168.0.11 6379
```

2. `sentinel.conf` 구성 파일 설정하기

```
# sentinal.conf
port 26379
# 모니터링 할 마스터 지정
# sentinel monitor (마스터 이름) (마스터 주소) (마스터 포트) (쿼럼)
sentinel monitor master-test 192.168.0.11 6379 2
```

- 센티널 인스턴스를 실행시킬 **모든 서버에서 해당 파일을 작성**해야 한다.

3. 센티널 인스턴스 시작하기

```
# redis-sentinel 이용
redis-sentinel /path/to/sentinel.conf

# redis-server 이용
redis-server /path/to/sentinel.conf --sentinel
```

2개 방법 모두 `sentinel.conf` 파일을 이용해 동일하게 동작하며 센티널 인스턴스를 시작시킨다.

지정된 위치에 파일이 없거나, 해당 경로에 데이터를 쓸 수 없으면 시작되지 않는다.

접근하기

```
# 센티널 인스턴스 직접 접근
$ redis-cli -p  26379
```

센티널 인스턴스에 접근하여 확인할 수 있는 정보

- 센티널이 모니터링하고 있는 마스터의 정보
- 센티널이 모니터링하고 있는 복제본의 정보

- 복제본을 함께 모니터링하고 있는 다른 센티널 인스턴스에 대한 정보

→ 레디스 인스턴스가 가진 데이터는 확인할 수 없어요 …

마스터 정보 얻기

```
SENTINEL master (마스터 이름)
```

센티널이 정상적으로 구성됐는지 알아볼 수 있다.

- **num-other-sentinel**: 마스터를 모니터링하고 있는 다른 센티널의 수
- **flags**: 마스터 상태 (`master`:정상/`s_down`:주관적 다운/`o_down`:객관적 다운)
- **num-slaves**: 마스터에 연결된 복제본 개수

복제본 정보 얻기

```
# 마스터에 연결된 복제본 정보 얻기
sentinel> SENTINEL replicas master-test

# 마스터에 연결된 센티널 정보 얻기
sentinel> SENTINEL sentinels master-test

# 마스터에 연결된 센티널 인스턴스가 설정한 쿼럼 값보다 큰지 확인
sentinel> SENTINEL ckquorum master-test
# 정상
OK 3 usable Sentinels. Quorum and failover authorization can
# 비정상
(error) NOQUORUN 1 usable Sentinels. Not enough available Sen
the specified quorum for this master. Not enough available Se
reach the majority and authorize a failover
```

`정상 센티널 수 < 설정 쿼럼 값` 이면 비정상으로 판단. **자동 페일오버가 불가능**하다.

- 장애가 발생해도 쿼럼 수 이상의 센티널 인스턴스에게 동의를 받을 수 없기 때문

페일오버 테스트

실제 레디스를 운영 서비스에 투입하기 전 테스트를 해보자

커맨드를 이용한 수동 페일오버 발생

```
# SENTINEL FAILOVER (마스터 이름)
SENTINEL FAILOVER master-test
```

다른 센티널의 동의를 구하지 않고 바로 페일오버 발생

확인 가능한 정보

- `센티널 - 복제본 노드` 간 네트워크 단절 등의 이슈로 페일오버가 실패하는지
- 센티널에 연결된 애플리케이션의 커넥션이 정상적으로 승격된 마스터에 연결되는지

마스터 동작을 중지시켜 자동 페일오버 발생

직접 마스터 노드에 장애를 발생시켜 페일오버가 발생하는지 확인

```
# redis-cli -h (마스터 주소) -p (마스터 포트) shutdown
$ redis-cli -h 192.168.0.11 -p 6379 shutdown
```

센티널이 주기적으로 마스터 노드에 ping을 보내 응답 확인

- `sentinel.conf`에 설정한 `down-after-milliseconds`(기본값:30000, 30초) 동안 응답이 오지 않으면 페일오버 트리거

센티널 운영하기

패스워드 인증

장애 상황에 센티널이 자동으로 페일오버를 시키기 때문에 복제 구성 내 **모든 레디스 노드는 잠재적 마스터 노드**다. (복제 파트에서 패스워드를 모두 같은 걸로 설정하는 이유인듯 !)

패스워드가 걸린 레디스를 모니터링할 경우 `sentinel.conf`에 패스워드 지정이 필요하다.

```
# sentinel.conf
# sentinel auth-pass (마스터 이름) (패스워드)
```

```
sentinel auth-pass master-test 123456
```

복제본 우선순위

`replica-priority` (기본값 100)

페일오버 진행 시 각 복제본 노드의 우선순위를 확인하여 **가장 작은 노드를 마스터로 선출**한다.

값이 0인 복제본은 절대 마스터로 선출되지 않는다.

운영 중 센티널 구성 정보 변경

운영 중 모니터링할 마스터를 추가/제거/변경 할 수 있다.

센티널이 여러 대라면 각각의 센티널에 적용해야 한다. **(변경한 설정이 다른 센티널로 전파되지 않음)**

```
# 새로운 마스터 모니터링 추가
# SENTINEL MONITOR (마스터 이름) (마스터 주소) (마스터 포트) (쿼럼)
ㅇ

# 마스터 모니터링 삭제
# SENTINEL REMOVE (마스터 이름)
ㅇ

# 특정 마스터에 대한 파라미터 변경
# SENTINEL SET (마스터 이름) [(파라미터명) (파라미터값) ...]
sentinel> SENTINEL SET master-test down-after-milliseconds 10
OK
sentinel> SENTINEL SET master-test quorum 1
OK

# 마스터에 종속되지 않는 센티널의 고유 설정값 변경 (레디스 6.2 ~)
# SENTINEL CONFIG SET (설정명) (설정값)
# SENTINEL CONFIG GET (설정명)
```

센티널 초기화

접근할 수 없는 복제본 노드도 계속 모니터링한다. 다시 돌아올지도 몰라 …

```
# 모니터링 중단
SENTINEL REMOVE (마스터 이름)

# 센티널 초기화 (마스터에 대한 모니터링)
SENTINEL RESET (패턴)
```

- 센티널 인스턴스 상태 정보 초기화
- 센티널이 모니터링하는 마스터, 복제본, 다른 센티널 인스턴스 정보 새로고침
- 전체 마스터 정보 초기화(*)도 가능

센티널 노드 추가/제거

1. 마스터를 모니터링하도록 설정하기
2. 센티널 인스턴스 실행하기
3. 자동으로 다른 센티널에 `known-list`에 추가 **(with 자동 검색 메커니즘)**

주의사항

- 여러 대 센티널을 한 번에 추가해야 한다면 **천천히 추가**하는 것이 오류 발생 가능성을 줄인다.

> 🔍 **왜?**
>
> 새로운 센티널 노드 추가 시 네트워크 부하가 발생할 수 있기 때문

- 센티널 노드 끼리는 오랜 시간동안 응답이 없어도 센티널의 `known-list`에서 지우지 않는다.

> 🔍 **왜?**
> 일시적 네트워크 통신 문제로 인해 노드 간 연결이 잠깐 끊어진 경우에도 클러스터 전체의 안정성을 유지하기 위해

자동 페일오버

센티널 노드들은 레디스 인스턴스를 함께 감지하여 오탐을 줄인다.

마스터 장애 상황 감지

1. `down-after-milliseconds` 값 만큼 마스터에 보낸 ping에 응답 받지 못하면 마스터가 다운됐다고 판단.
2. 마스터가 다운됐다고 판단한 센티널 노드는 마스터를 `sdown`(주관적 다운)으로 플래그 변경
3. 다른 센티널 노드들에게 장애 사실 전파
4. 다른 노드에서 장애 인지 여부를 응답함
    - 자신을 포함해 쿼럼 값 이상의 노드가 장애를 인지했다면 `odown`(객관적 다운)으로 플래그 변경
    - 마스터를 제외한 노드에 대해서는 sdown만 함 (페일오버 진행 시 `sdown` 상태 복제본은 승격되지 않게 하기 위해서)
5. 처음으로 `odown`을 인지한 노드가 **페일오버 시작**
    - `에포크` 값 증가
    - 다른 센티널 노드에게 센티널 리더를 선출하기 위한 `증가시킨 에포크`와 `투표 메시지`를 보냄
    - 전달받은 노드들은
        - `자신의 에포크 < 전달받은 에포크`면 자신의 에포크를 증가시키고 리더에게 투표하겠다는 응답을 보냄
        - `자신의 에포크 = 전달받은 에포크`면 **이미 리더로 선출한 센티널 id를 응답**함
    - 하나의 에포크에서 센티널은 하나의 센티널에 투표할 수 있음 (변경 불가)
    - **과반수 이상의 센티널이 페일오버에 동의하면** 리더 센티널이 마스터로 승격시킬 복제본을 선정함

      🕵️ **자격 요건**

        - `redis.conf` 파일에 설정된 `replica-priority`가 낮은 복제본
        - 마스터로부터 더 많은 데이터를 수신(`master-repl`)한 복제본
        - `runID`가 사전 순으로 작은 복제본 (특별한 의미 없음, 하나의 노드를 고르기 위해)

        - runID : 모든 레디스 인스턴스가 가지는 실행의 식별자
6. 선정한 복제본에 기존 마스터로부터의 복제를 끊음 (`slaveof no one` 커맨드 수행)
7. 기존 복제본들을 새로 승격된 마스터의 복제본이 될 수 있도록 복제 연결 변경
8. 복제 그룹의 모든 센티널 노드에서도 레디스 구성 정보 변경
9. 센티널은 새로운 마스터를 모니터링함

에포크

각 마스터에서 발생한 **페일오버의 버전**을 관리

에포크 값을 이용해 페일오버 과정 진행동안 **모든 센티널 노드가 같은 작업을 시도하고 있다는 것을 보장**

센티널 리더 선출

- 페일오버를 실제로 시도하기 전 센티널 리더 선출을 위해 실제 센티널 개수 중 과반수 이상의 센티널의 동의를 얻어야 함
- 즉 쿼럼 값보다 많은 센티널이 동의해도 과반수보다 작다면 페일오버는 발생하지 않음
- 🔍 페일오버를 수행하는 센티널 ≠ 센티널 리더

스플릿 브레인

네트워크 파티션 이슈로 인해 분산 환경의 데이터 저장소가 끊어지고 **끊긴 두 부분이 각각을 정상적인 서비스라고 인식하는 현상**

예시

마스터와 센티널 A 그리고 복제본과 센티널 B, C 노드 간 네트워크 단절이 일어난 경우

- 센티널 B, C는 마스터 노드로의 접근이 비정상임을 감지
- 복제본 노드를 마스터로 승격시킴
    - 과반수 이상의 센티널이 같은 네트워크 파티션에 존재하기 때문
- 하나의 복제본에 2개 마스터가 생김 (!) → **스플릿 브레인**

문제점

- 스플릿 브레인 현상이 발생하면 이전에 연결된 클라이언트는 기존 마스터 인스턴스에 데이터를 입력
- 이후에 연결된 클라이언트는 새롭게 승격된 마스터 인스턴스에 데이터를 입력
- 네트워크 단절이 복구되면 기존 마스터가 새롭게 승격된 마스터의 복제본으로 연결
- 기존 마스터가 네트워크 단절 동안 처리한 모든 데이터가 사라짐 (!)