

# 구현부터 테스트까지

## 대용량 트래픽 처리 시스템

---

작성자 : 이경일

소속팀 / 상위부서 : Cell TF / SmartStudio

# 소개

---

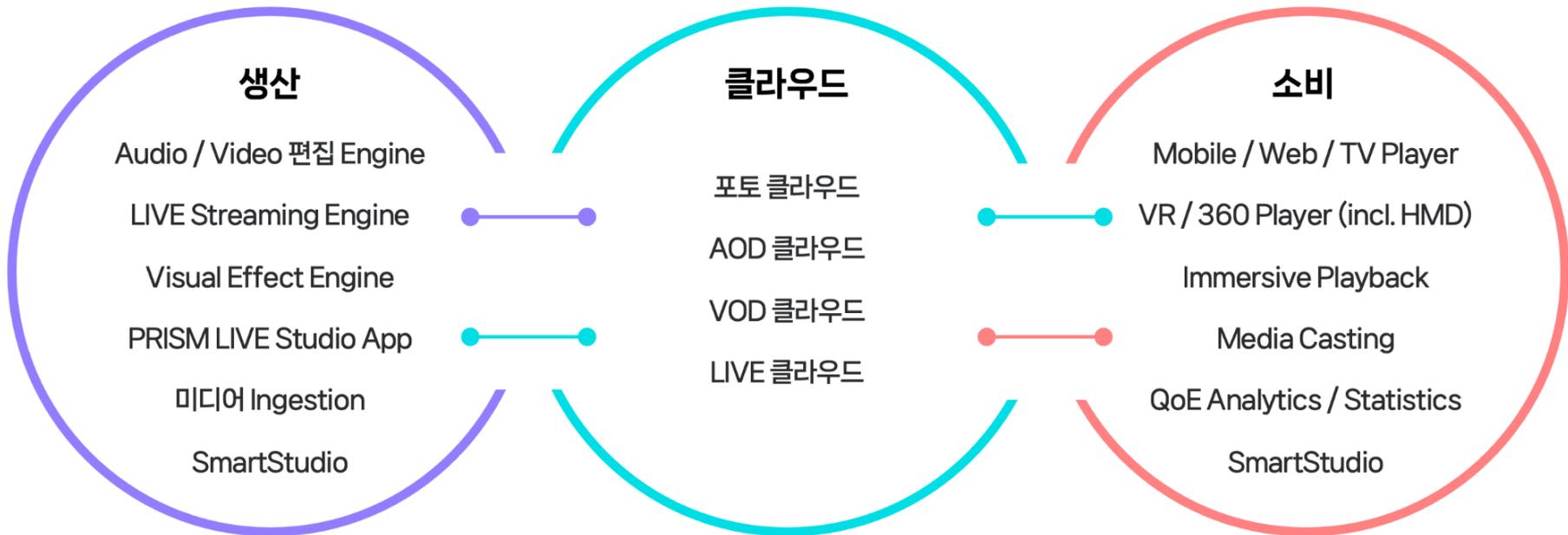


- **이경일**
  - [leekyoungil@gmail.com](mailto:leekyoungil@gmail.com)
  - [kyoungil.lee@navercorp.com](mailto:kyoungil.lee@navercorp.com)
- **(현) Naver / ETECH - SmartStudio**
- **Server 개발, 잡일, 막노동, 매니징 등**
  - 네이버 쇼핑 플랫폼 개발
  - 네이버 플레이스 O2O 플랫폼 개발
  - 카카오 커머스 플랫폼 개발
  - CJ오쇼핑 커머스 플랫폼 개발

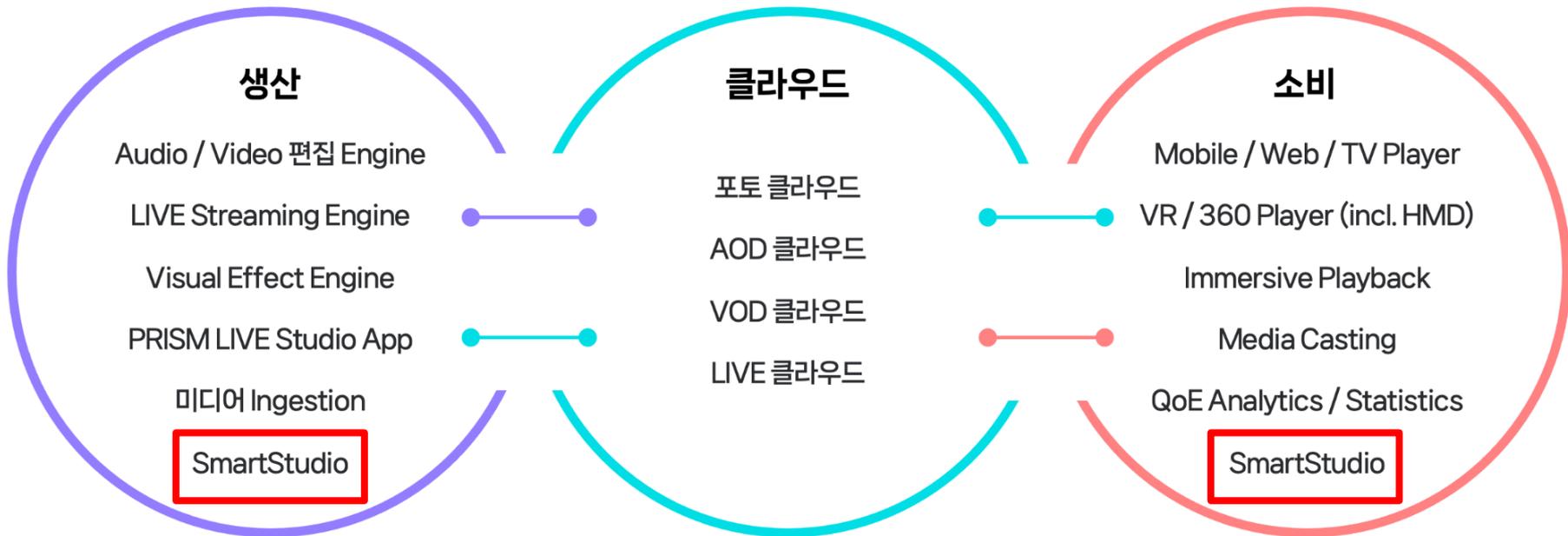
**Cache**  
**Concurrency**  
**Spring Cloud Config, Bus**  
**K8S**  
**Pinpoint**  
**Locust**

**지금부터 이야기를 시작해 보겠습니다.**

# Naver ETECH



# SmartStudio



# SmartStudio - 저작도구 플랫폼 개발 조직



## Cell Platform

It is a platform that provides infrastructure and service API that can be easily developed regardless of environment when developing services.

Documentation

GitHub



## ELF framework

Editor framework for building collaborative editing applications

Documentation

GitHub

## Moment

It is a short form video editor that allows you to edit photos and videos easily and simply.

Documentation

GitHub

## SmartEditor

Smart Editor provides users with high editing freedom and content completion. It is a creative tool that satisfies you at the same time.

Documentation

GitHub

## nBilly

NCDP(No Code Development Platform) that uses IN/OUT APIs to make websites look like Lego blocks. Users can quickly and easily create a website at a low cost.

Documentation

GitHub

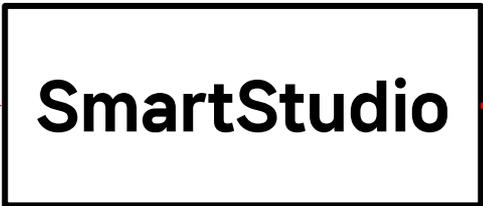


## Yorkie

Document store for building collaborative editing applications

Documentation

GitHub





### Cell Platform

It is a platform that provides infrastructure and service API that can be easily developed regardless of environment when developing services.

 Documentation

 GitHub



### ELF framework

Editor framework for building collaborative editing applications

 Documentation

 GitHub

### Moment

It is a short form video editor that allows you to edit photos and videos easily and simply.

 Documentation

 GitHub

### SmartEditor

Smart Editor provides users with high editing freedom and content completion. It is a creative tool that satisfies you at the same time.

 Documentation

 GitHub

### nBilly

NCDP(No Code Development Platform) that uses IN/OUT APIs to make websites look like Lego blocks. Users can quickly and easily create a website at a low cost.

 Documentation

 GitHub



### Yorkie

Document store for building collaborative editing applications

 Documentation

 GitHub



### Cell Platform

It is a platform that provides infrastructure and service API that can be easily developed regardless of environment when developing services.

 Documentation

 GitHub



### ELF framework

Editor framework for building collaborative editing applications

 Documentation

 GitHub

### Moment

It is a short form video editor that allows you to edit photos and videos easily and simply.

 Documentation

 GitHub

### SmartEditor

Smart Editor provides users with high editing freedom and content completion. It is a creative tool that satisfies you at the same time.

 Documentation

 GitHub

### nBilly

NCDP(No Code Development Platform) that uses IN/OUT APIs to make websites look like Lego blocks. Users can quickly and easily create a website at a low cost.

 Documentation

 GitHub



### Yorkie

Document store for building collaborative editing applications

 Documentation

 GitHub

The logo icon for nBilly consists of a vertical purple bar on the left, followed by a light blue semi-circle on top and a blue circle on the bottom, both partially overlapping the purple bar.

**nBilly** NO-Code  
Development Platform

- Text
- Image
- Button
- Tab
- Video
- Share
- Container
- Section

# Cartier

WATCHES AND WONDERS

JEWELRY

WATCHES

ART OF LIVING



## LA PANTHÈRE

Through this new fragrance campaign, Cartier returns to the origins of its beloved animal, the panther, and is guided by her through the heart of a tropical forest. The British actress Vanessa Kirby appears from within this green and abundant hidden land, a world full of life and beauty.



### Foundation Style

Fill  Add... ▾

### Animation

None ▾

### Advanced

[Edit CSS >](#)

### ID

Add ID for anchoring... [Copy](#)

### nClick Global

ccsrv

g\_ssc

The logo icon consists of a vertical purple bar on the left, followed by a light blue semi-circle at the top and a blue circle at the bottom, resembling a stylized letter 'B'.

# nBilly SDK



혼자 BE 개발



- 마케팅
- 이벤트 페이지
- 간단한 웹사이트



**고민이 생겼습니다.**

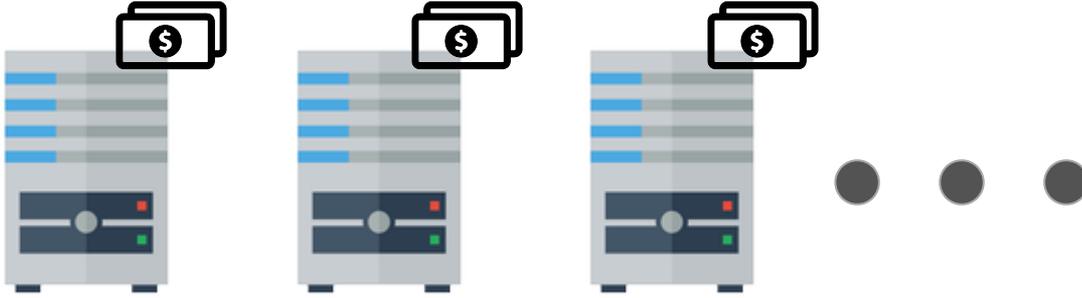
**대용량 트래픽 대응...**

**그럼 어떻게 해야 할까요?**



# 개발자 x N





# 요즘 정말 어려운지... 리소스 압박이 좀...



박진영 아님

**리소스?**

개발자



오늘이 무슨 요일인지도 몰라요. 날짜도 모르구요.  
전 그냥 코딩만 해요.

디자인, 기획



난 사실 아무 생각이 없어



인프라



개발자



오늘이 무슨 요일인지도 몰라요. 날짜도 모르구요.  
전 그냥 코딩만 해요.

디자인, 기획



난 사실 아무 생각이 없어



인프라





Java에서 Map 객체를 이용해서 Local Cache 만드는 코드를 짜줘  
필요한 기능은 set, get, purge, LRU, LFU 기능이야

아래는 Java에서 Map 객체를 이용해서 Local Cache를 구현하는 예시 코드입니다.

```
java Copy code

import java.util.LinkedHashMap;
import java.util.Map;

public class LocalCache<K, V> {

    private final int maxCapacity;
    private final boolean isLRU;
    private final boolean isLFU;
    private final Map<K, V> cache;

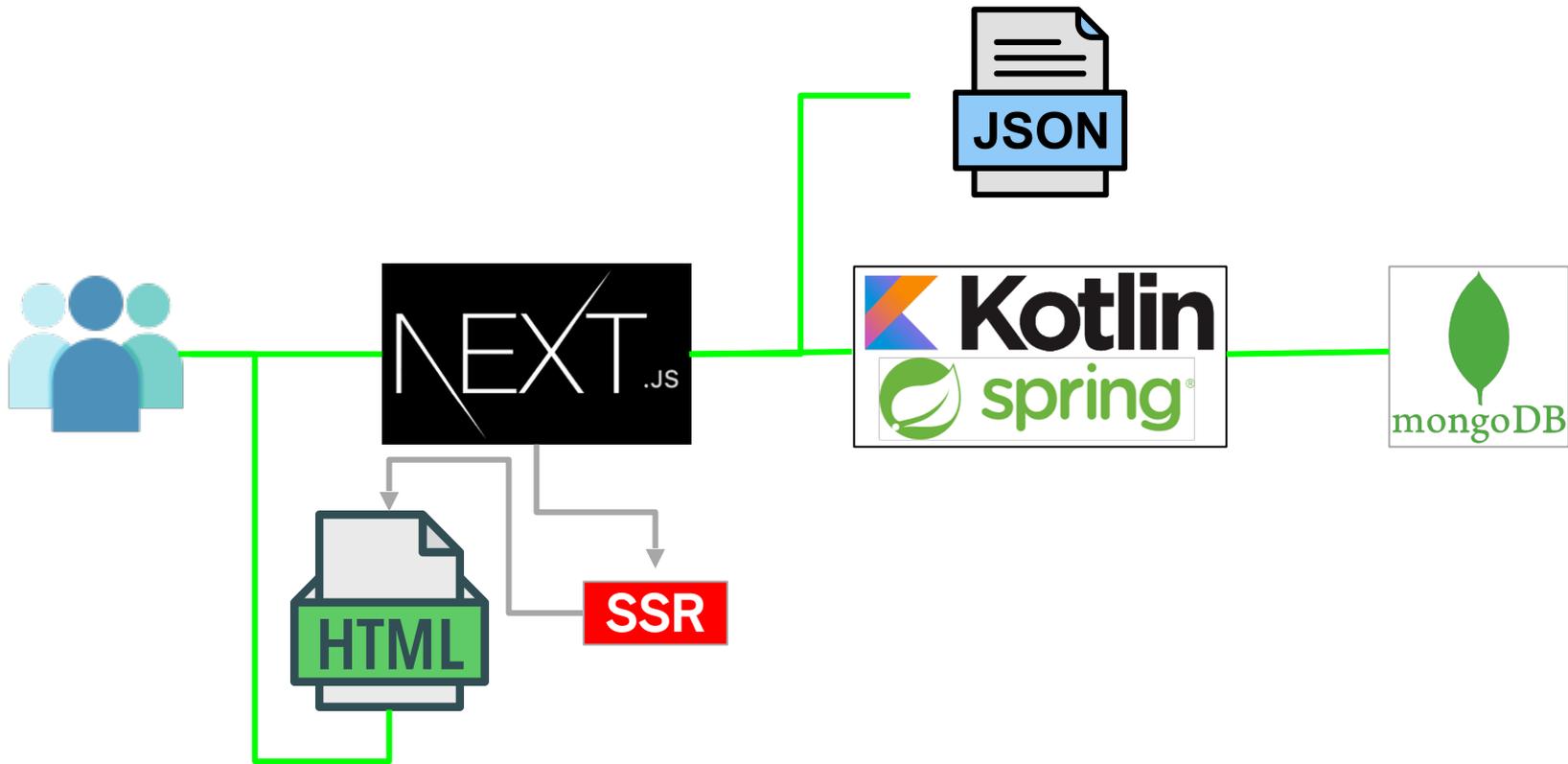
    public LocalCache(int maxCapacity, boolean isLRU, boolean isLFU) {
        this.maxCapacity = maxCapacity;
        this.isLRU = isLRU;
        this.isLFU = isLFU;

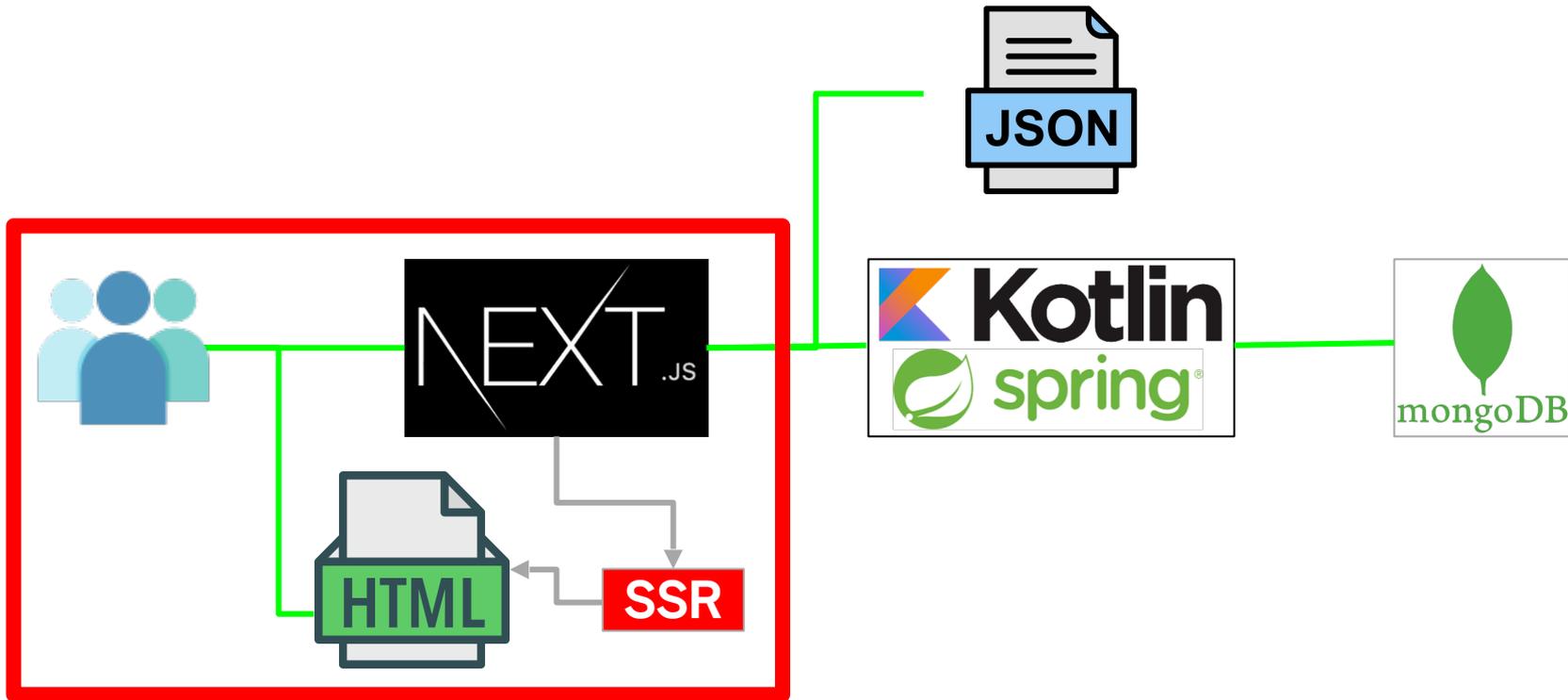
        // LinkedHashMap을 사용하여 LRU 또는 LFU 캐시를 구현합니다.
        if (isLRU) {
            this.cache = new LinkedHashMap<K, V>(maxCapacity, 0.75f, true) {
                @Override
                protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
                    return size() > maxCapacity;
                }
            };
        } else if (isLFU) {

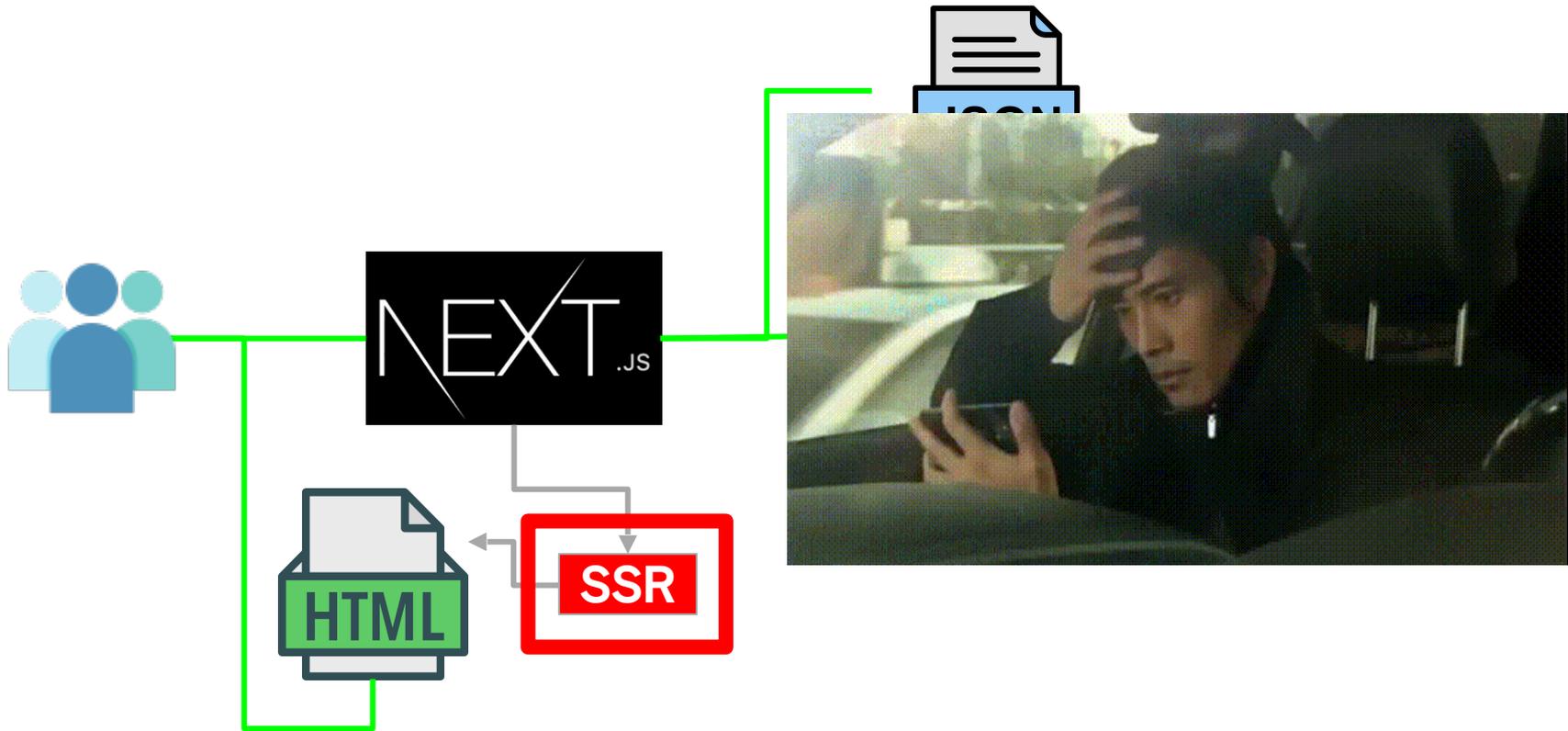
```

이런 발 이제 폐지 주워야겠네

**자.. 다시 돌아와서 구조를 보면**







**이걸 어떻게 해야 할까요?**

# Chat GPT에게 물어봤습니다.





대용량 트래픽을 대응하려면 어떻게 해야해?



대용량 트래픽을 대응하는 방법은 여러 가지가 있습니다. 일부 중요한 전략은 다음과 같습니다.



1. 서버 및 인프라 확장: 높은 트래픽을 처리하기 위해 서버 및 인프라를 확장해야 합니다. 서버를 추가하거나 클라우드 컴퓨팅과 같은 인프라를 이용하여 처리 능력을 높일 수 있습니다.
2. 캐시 및 로드 밸런싱: 캐시와 로드 밸런싱을 사용하면 서버 부하를 줄일 수 있습니다. 캐시를 사용하면 반복적으로 요청되는 내용을 저장하고 다시 보여줌으로써 대역폭을 줄일 수 있습니다. 로드 밸런싱은 여러 서버로 부하를 분산하여 대기 시간을 줄일 수 있습니다.
3. 콘텐츠 배포 네트워크(CDN): CDN은 전 세계에 있는 여러 지역의 서버를 사용하여 콘텐츠를 배포합니다. 이를 통해 지리적으로 분산된 사용자들에게 빠른 속도로 콘텐츠를 제공할 수 있습니다.
4. 데이터베이스 최적화: 데이터베이스 최적화를 통해 대용량 트래픽에 대한 응답 시간을 줄일 수 있습니다. 인덱스를 최적화하거나 더 높은 성능의 데이터베이스 엔진을 사용할 수 있습니다.
5. 코드 최적화: 코드 최적화를 통해 더 빠른 실행 속도를 얻을 수 있습니다. 코드를 최적화하여 처리 속도를 높이는 것은 트래픽 대응에서 중요한 전략 중 하나입니다.

이러한 전략을 통해 대용량 트래픽을 처리할 수 있습니다. 그러나 트래픽이 증가하면서 문제가 발생하면 시스템의 복잡성을 고려하여 전문가의 도움을 받는 것이 좋습니다.



대용량 트래픽을 대응하려면 어떻게 해야해?



대용량 트래픽을 대응하는 방법은 여러 가지가 있습니다. 일부 중요한 전략은 다음과 같습니다.



1. 서버 및 인프라 확장: 높은 트래픽을 처리하기 위해 서버 및 인프라를 확장해야 합니다. 서버를 추가하거나 클라우드 컴퓨팅과 같은 인프라를 이용하여 처리 능력을 높일 수 있습니다.
2. 캐시 및 로드 밸런싱: 캐시와 로드 밸런싱을 사용하면 서버 부하를 줄일 수 있습니다. 캐시를 사용하면 반복적으로 요청되는 내용을 저장하고 다시 보여줌으로써 대역폭을 줄일 수 있습니다. 로드 밸런싱은 여러 서버로 부하를 분산하여 대기 시간을 줄일 수 있습니다.
3. 콘텐츠 배포 네트워크(CDN): CDN은 전 세계에 있는 여러 지역의 서버를 사용하여 콘텐츠를 배포합니다. 이를 통해 지리적으로 분산된 사용자에게 빠른 속도로 콘텐츠를 제공할 수 있습니다.
4. 데이터베이스 최적화: 데이터베이스 최적화를 통해 대용량 트래픽에 대한 응답 시간을 줄일 수 있습니다. 인덱스를 최적화하거나 더 높은 성능의 데이터베이스 엔진을 사용할 수 있습니다.
5. 코드 최적화: 코드 최적화를 통해 더 빠른 실행 속도를 얻을 수 있습니다. 코드를 최적화하여 처리 속도를 높이는 것은 트래픽 대응에서 중요한 전략 중 하나입니다.

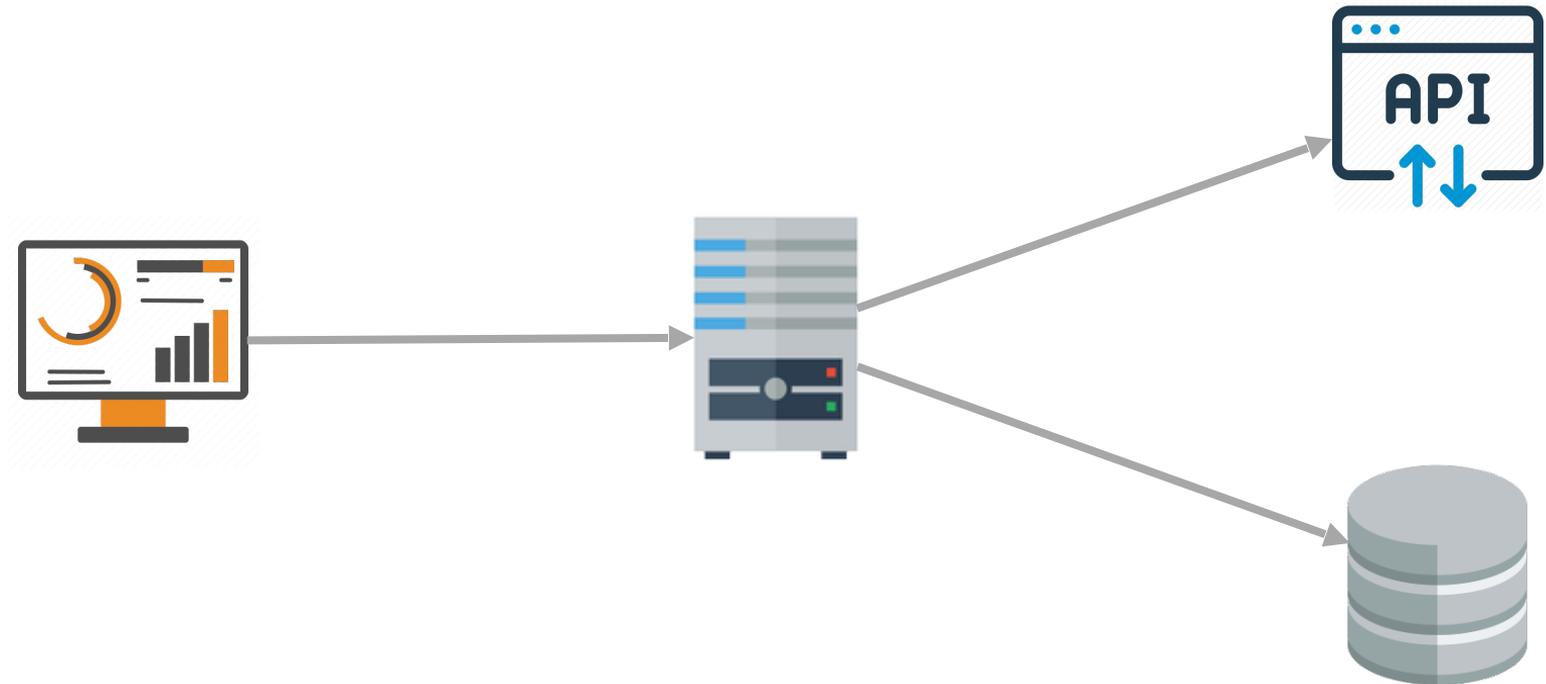
이러한 전략을 통해 대용량 트래픽을 처리할 수 있습니다. 그러나 트래픽이 증가하면서 문제가 발생하면 시스템의 복잡성을 고려하여 전문가의 도움을 받는 것이 좋습니다.

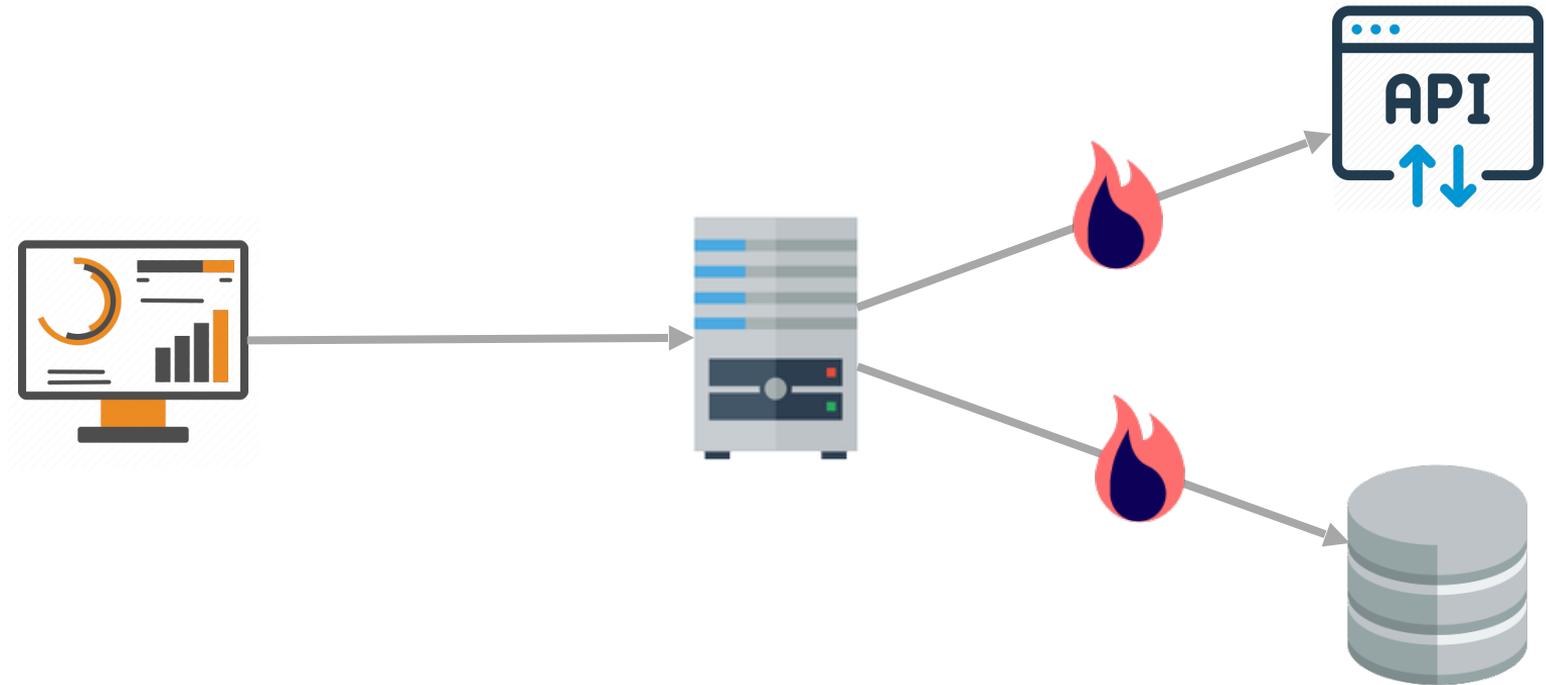
# 캐시를 이용해서 대용량 트래픽을 대응한다.

---

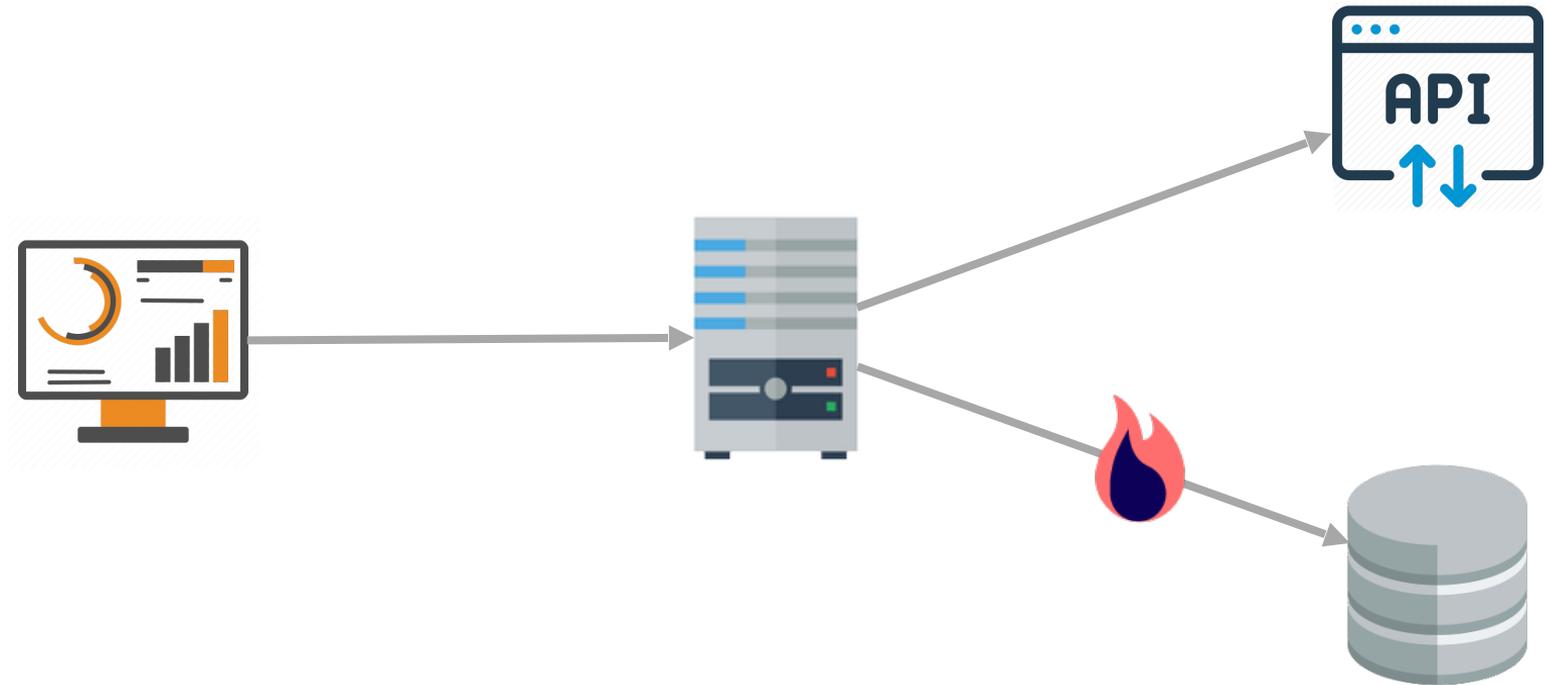


**일단 캐시 관련해서 알아보까요?**



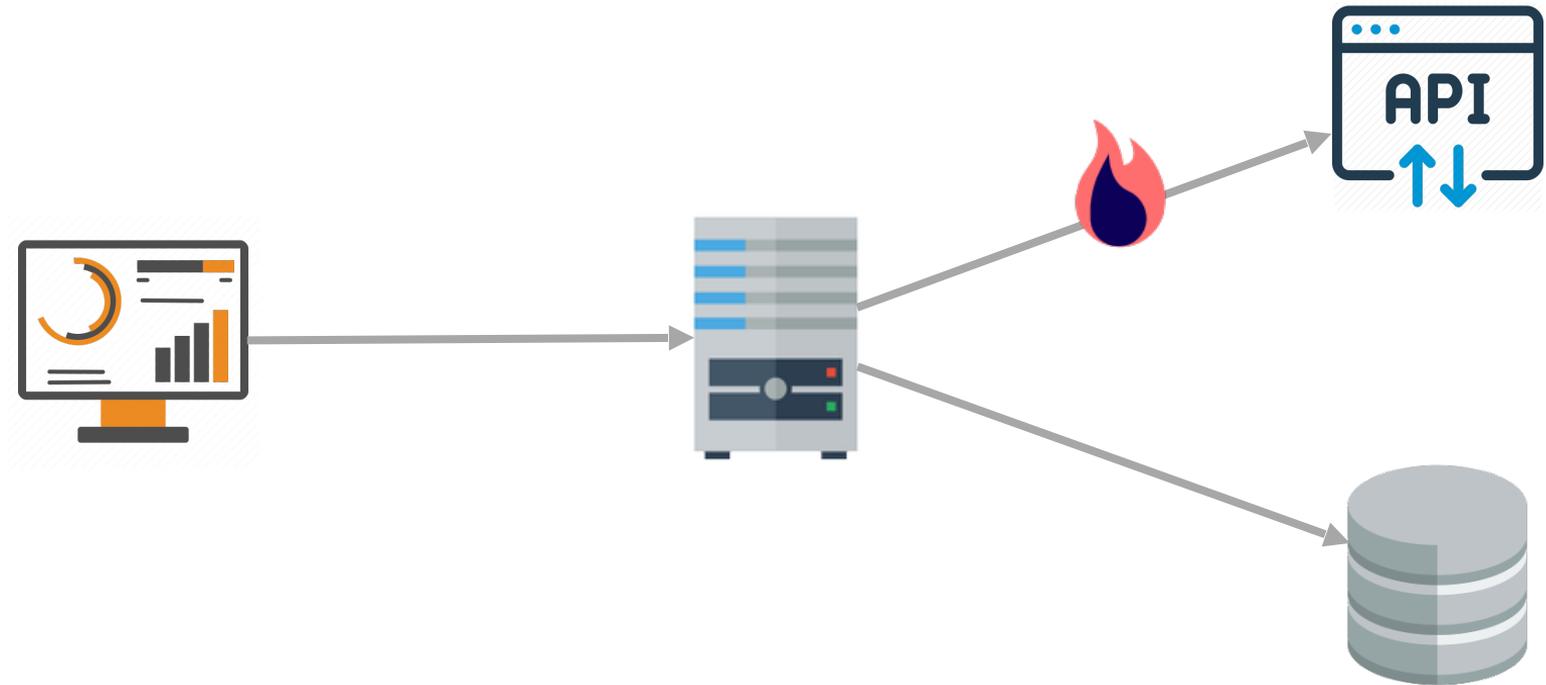


**왜 느낄까요?**



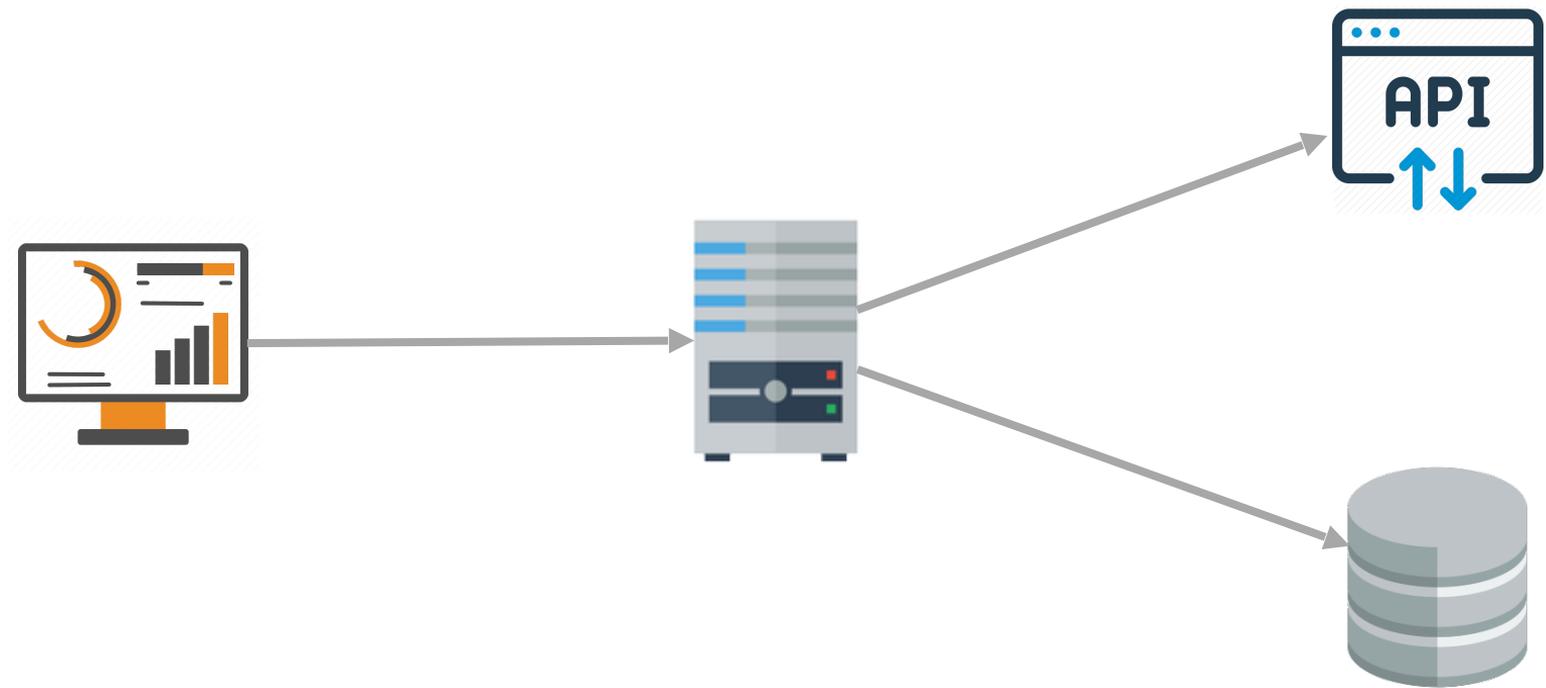
데이터에 **접근**하는 시간이 오래 걸림

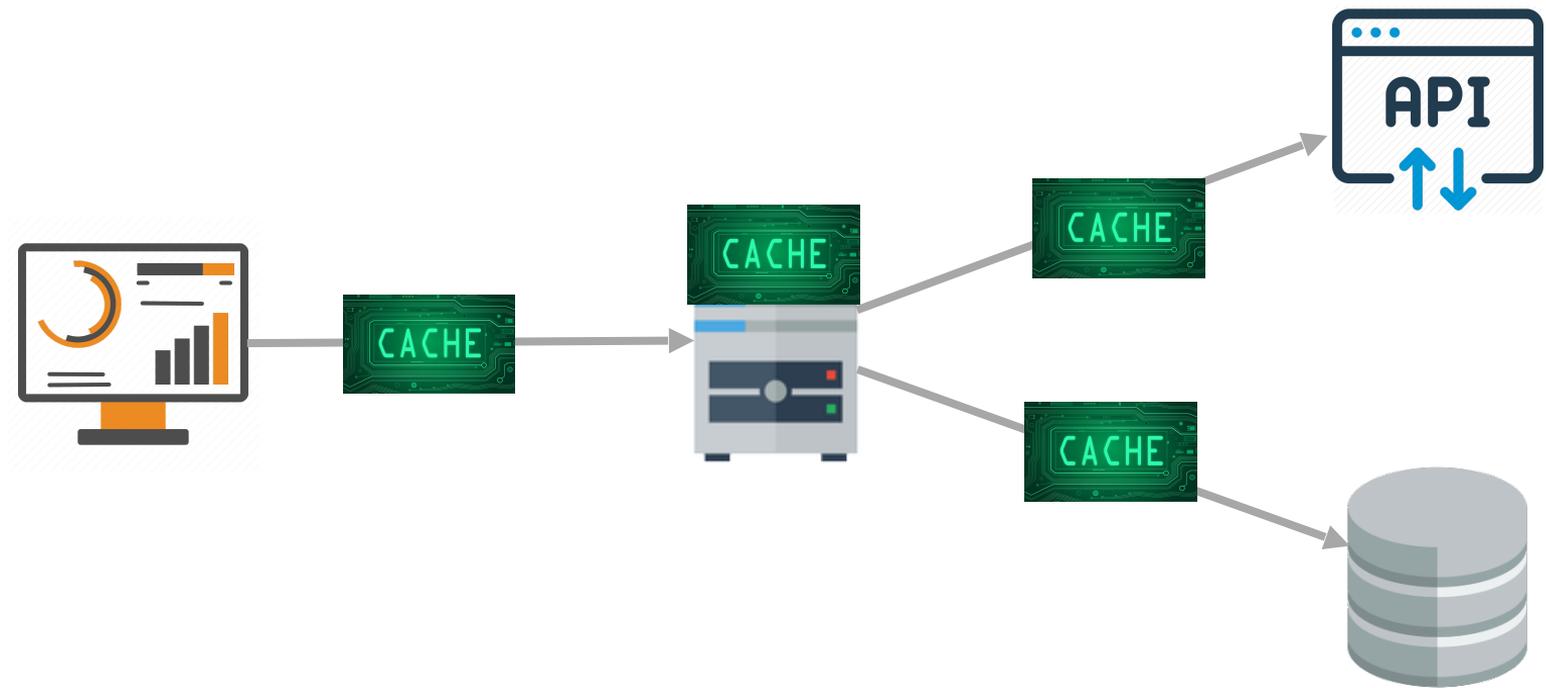
# Network, Disk IO

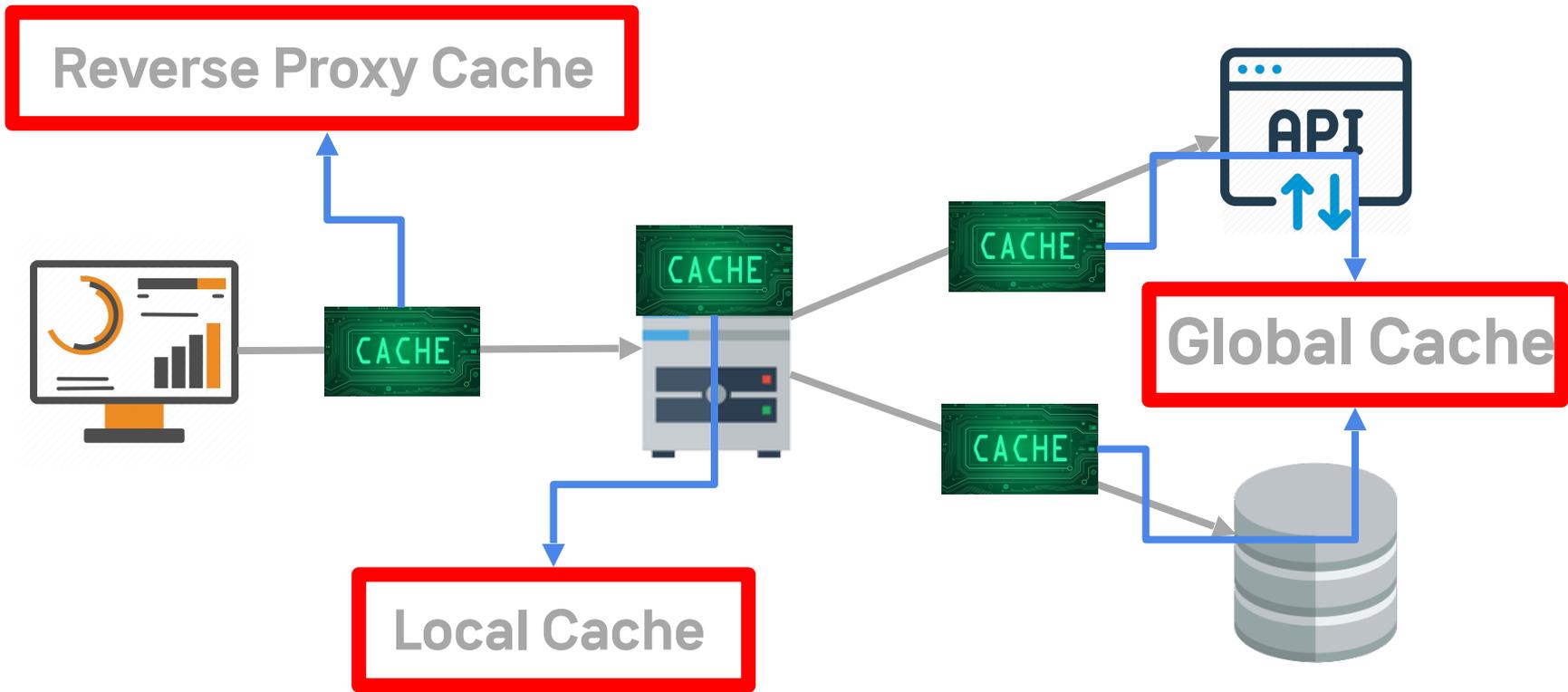


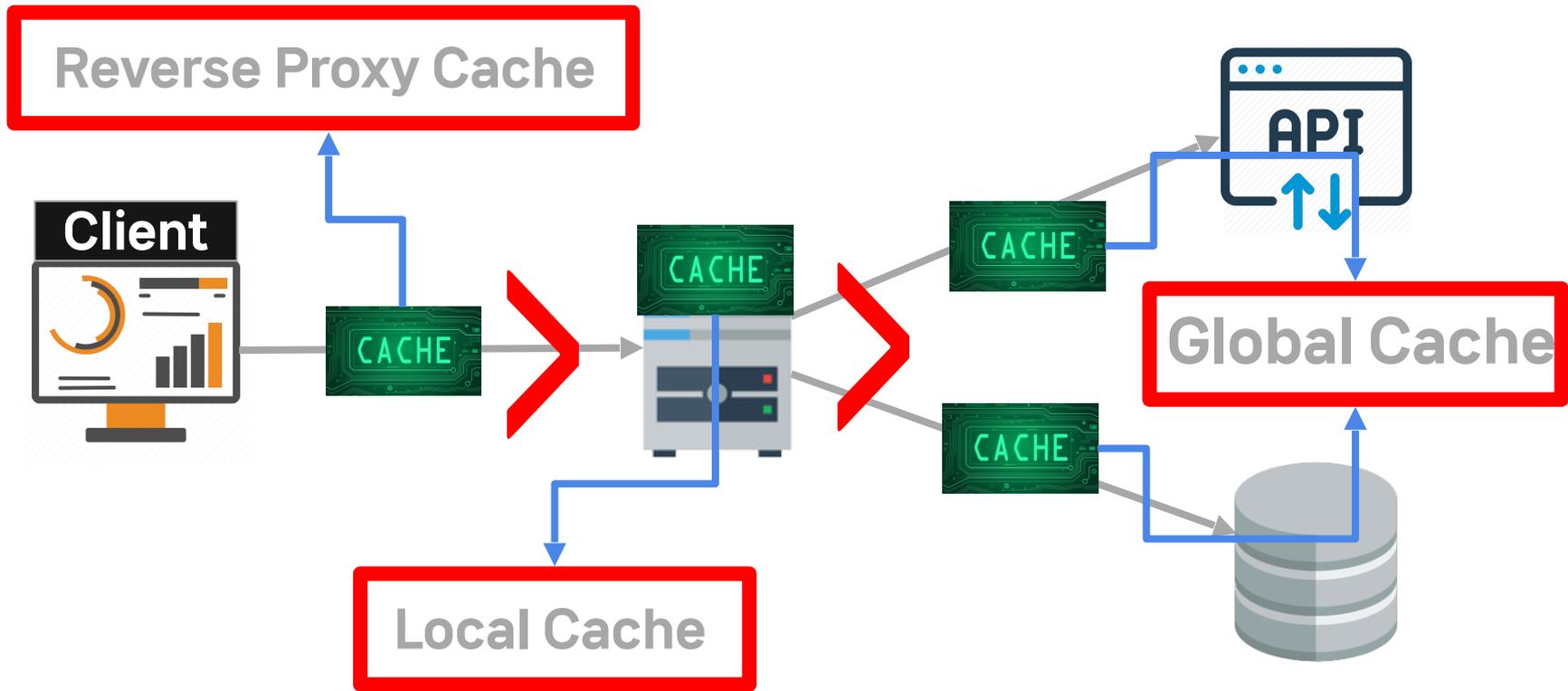
# 외부 API 로직 실행, 인프라 연동

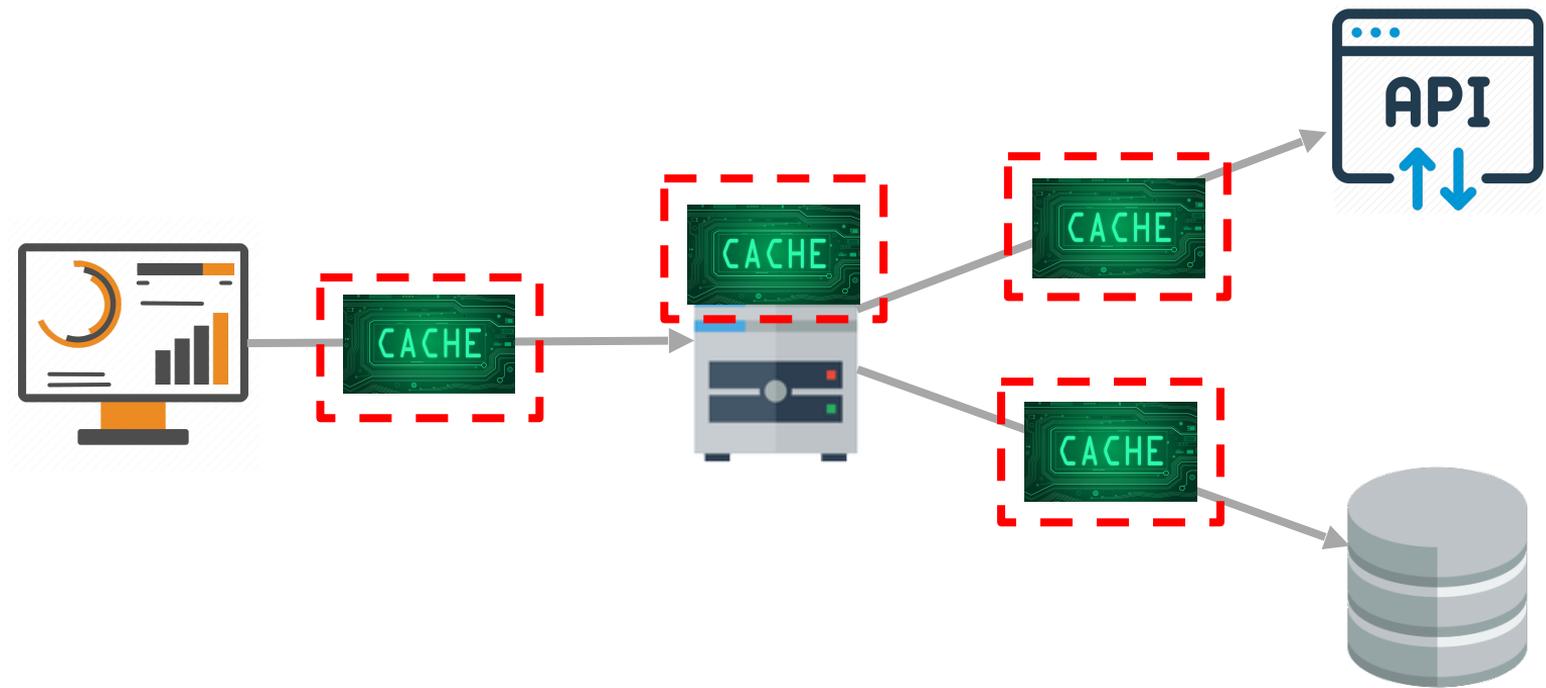
이럴 때 **캐시**를 사용하면 좋습니다.











**미리 처리한 결과를 저장소에 저장하여 사용**



**메모리**를 저장소로 사용

왜 **메모리**를 저장 공간으로 사용할까요?

**빨라서요...**



그럼 전부 다 메모리에 때려 넣으면 되겠네?

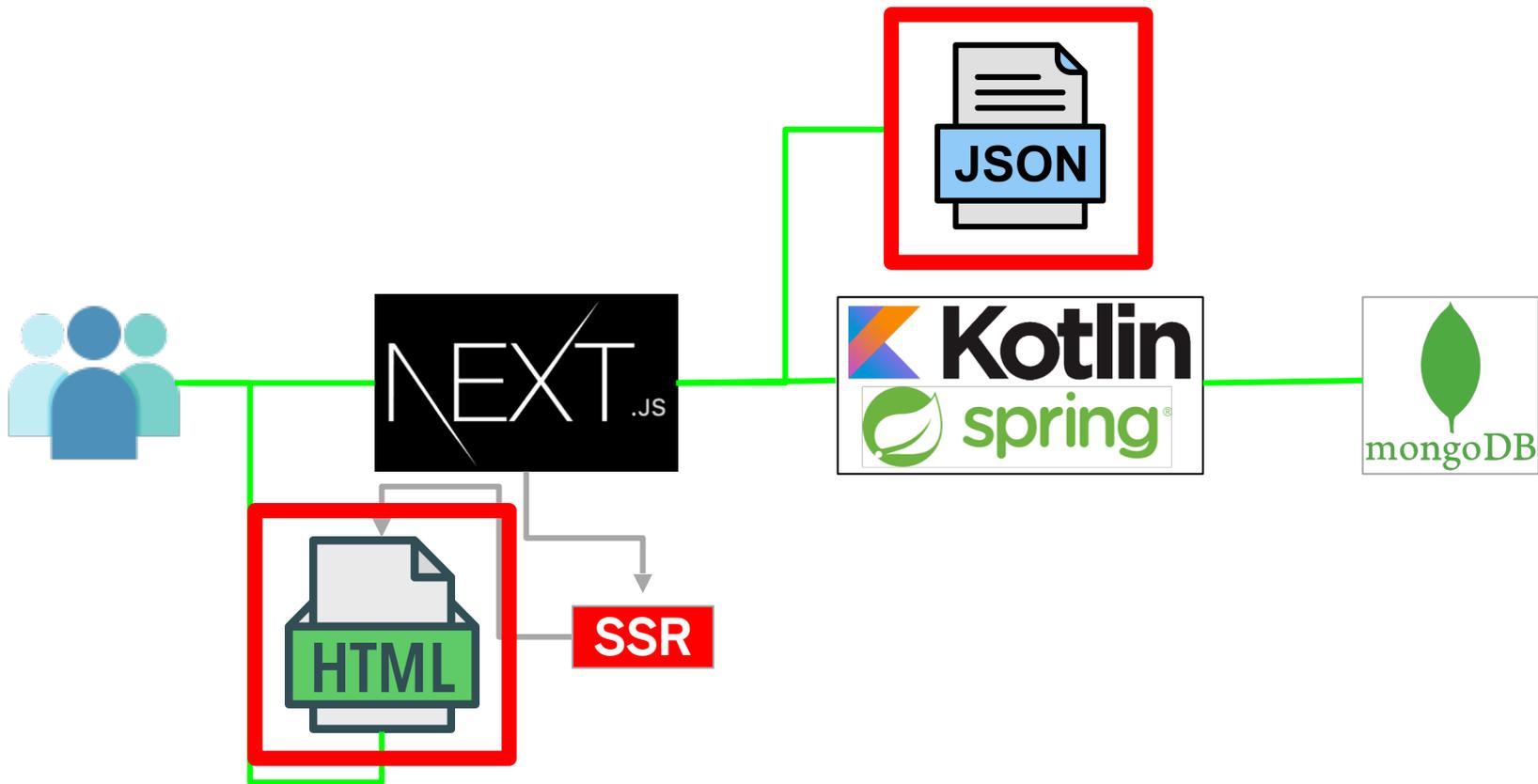
근데 **비싸요...**



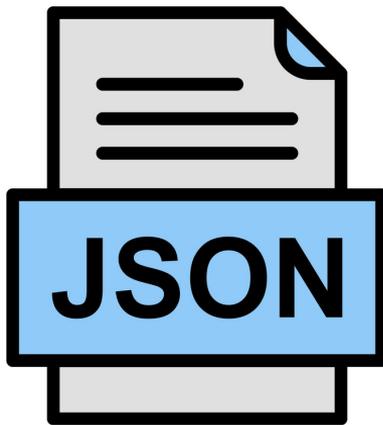
**부자**는 가능할 수도...



**캐시 할 대상을 생각을 좀 해 보면 ...**

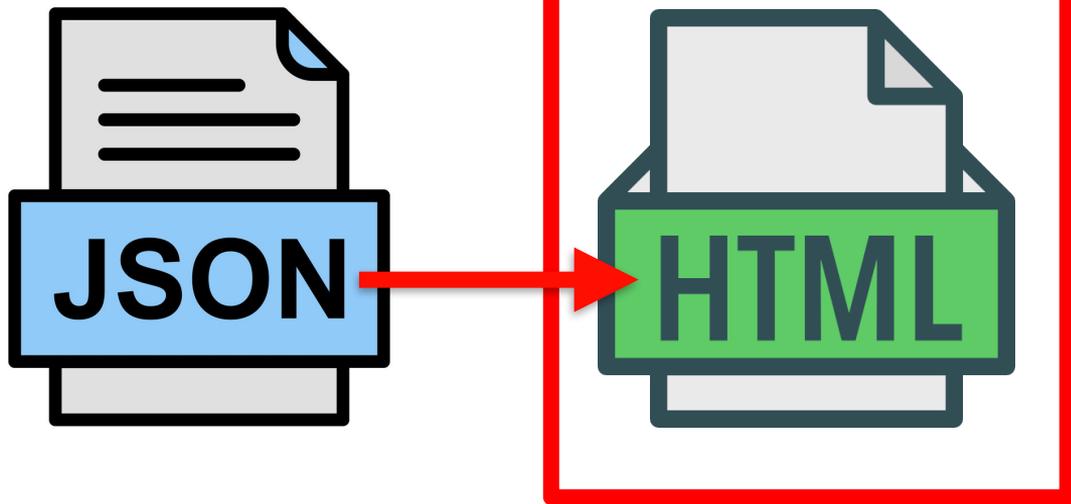


- DataSource Json
- SSR 결과인 Html



- DataSource Json

- SSR 결과인 Html



**주 연산의 유형은?**

**READ 연산이 80% 이상**



**메모리에 올려봅시다.**

**몇 가지 방법이 있습니다.**

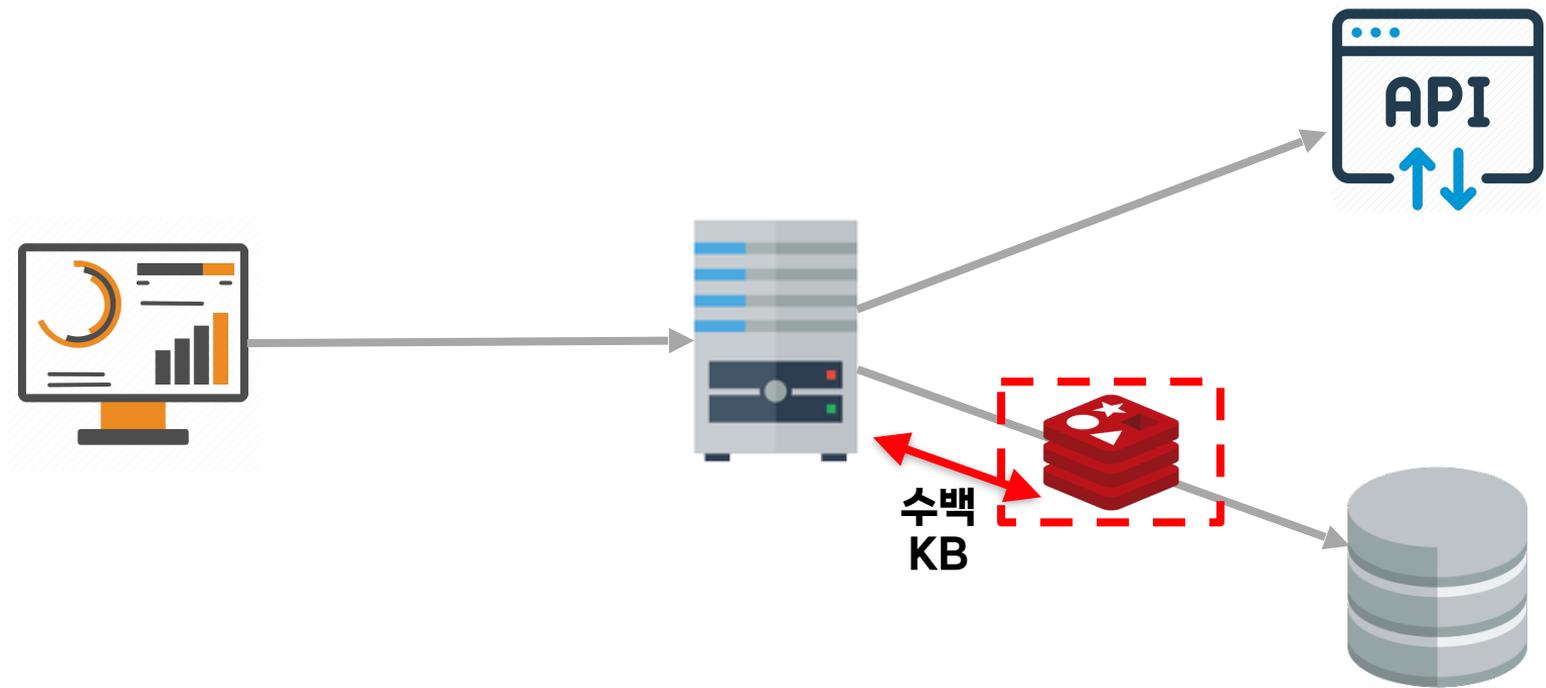


Redis 국내 1인자의 흔한 길거리 코딩





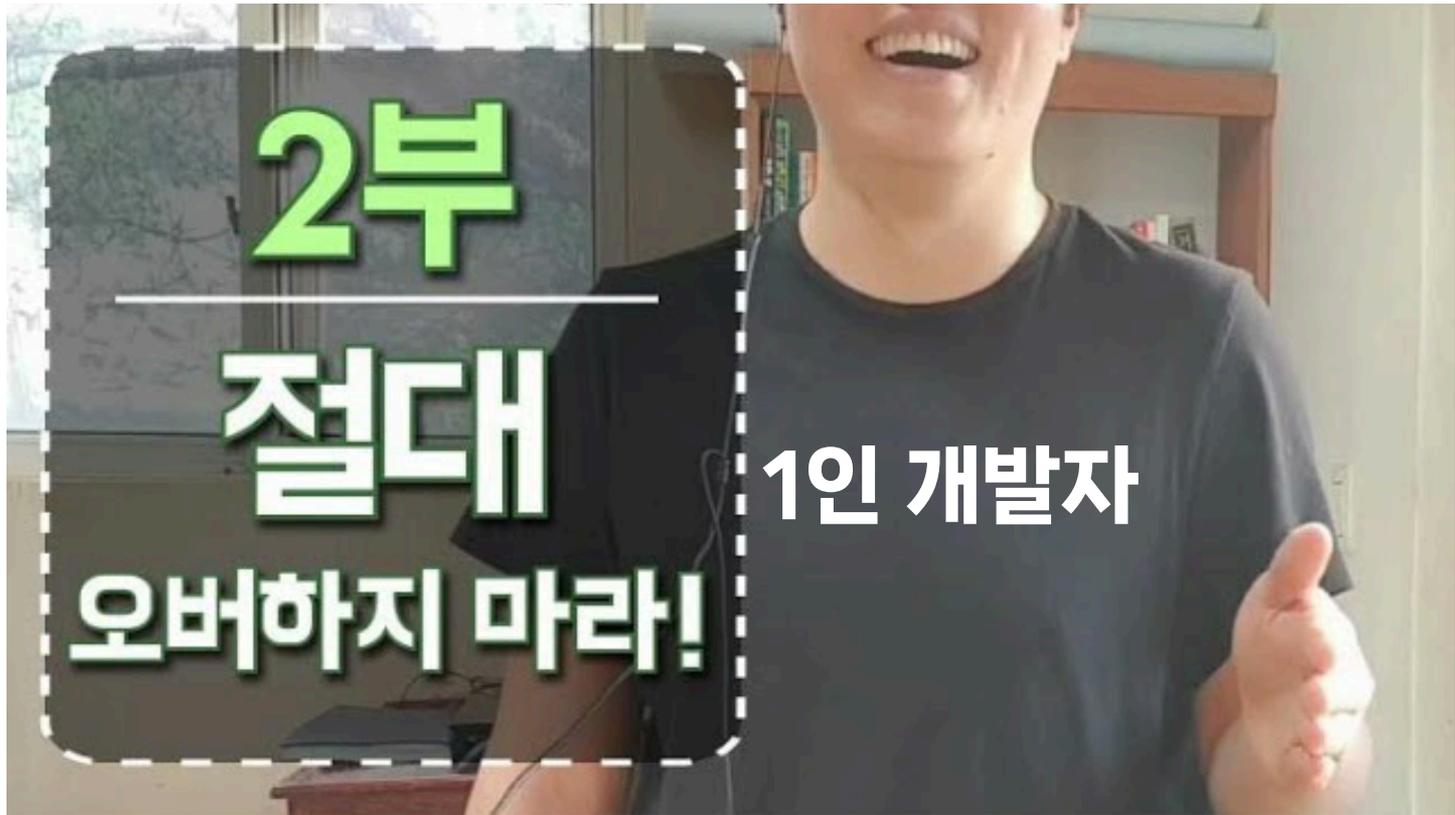




It's no mystery that Japanese mechanical pencils are a cut above the rest,



0:02 / 5:24 • Intro > Screenshot      



2부

절대

오버하지 마라!

1인 개발자



### Cell Platform

It is a platform that provides infrastructure and service API that can be easily developed regardless of environment when developing services.

 Documentation

 GitHub



### ELF framework

Editor framework for building collaborative editing applications

 Documentation

 GitHub

### Moment

It is a short form video editor that allows you to edit photos and videos easily and simply.

 Documentation

 GitHub

### SmartEditor

Smart Editor provides users with high editing freedom and content completion. It is a creative tool that satisfies you at the same time.

 Documentation

 GitHub

### nBilly

NCDP(No Code Development Platform) that uses IN/OUT APIs to make websites look like Lego blocks. Users can quickly and easily create a website at a low cost.

 Documentation

 GitHub



### Yorkie

Document store for building collaborative editing applications

 Documentation

 GitHub

# 전사 캐시 인프라 장애

메모리 캐시





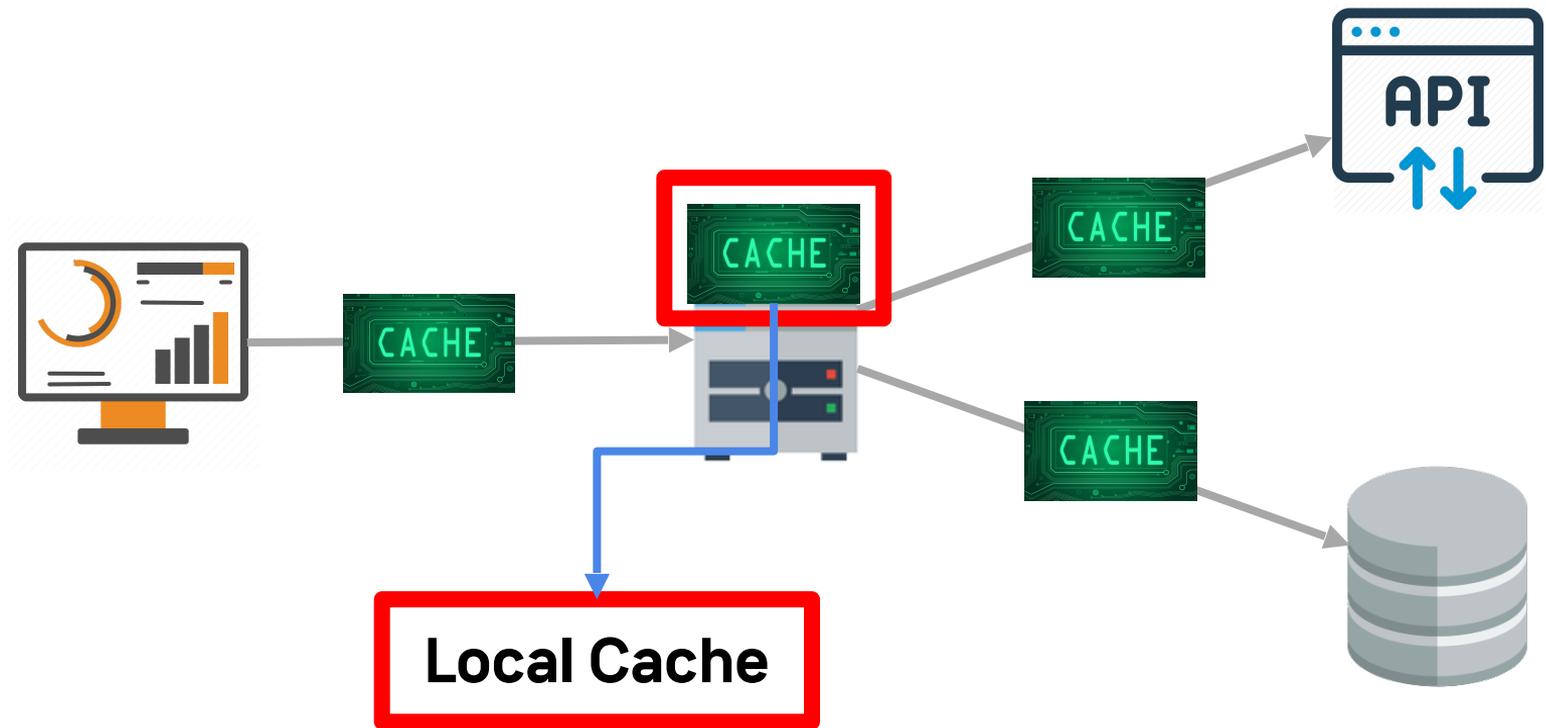
안녕히 계세요 여러분~~~!

저는 이 세상의 모든 굴레와 속박을 벗어던지고

제 행복을 찾아 떠납니다~~~!

**그래서 저는!**

# Application Level Cache



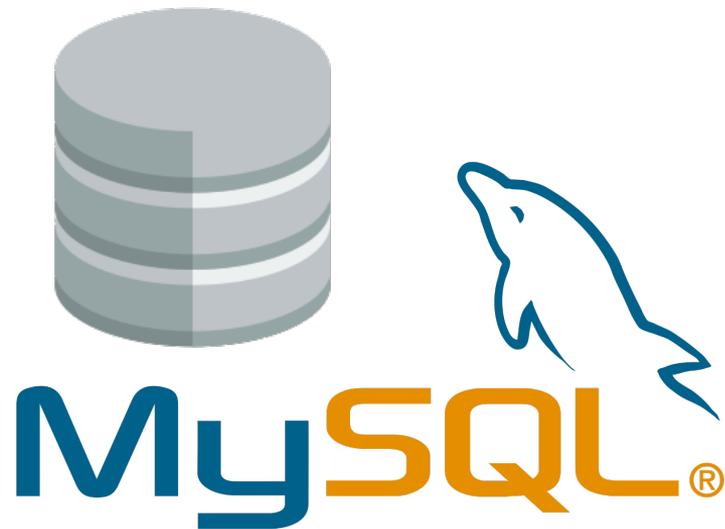
# Keyword **Memory, Cache**



# 데이터를 메모리에 올리려고 노력함



# 주로 사용하는 RDB는 MySQL



# 부자들은 Oracle, SQL Server로 사용



MySQL®

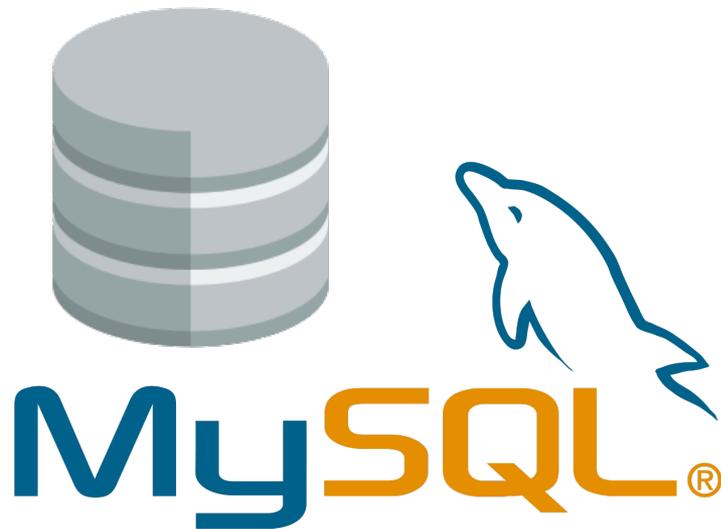
# Memory 엔진이 존재함



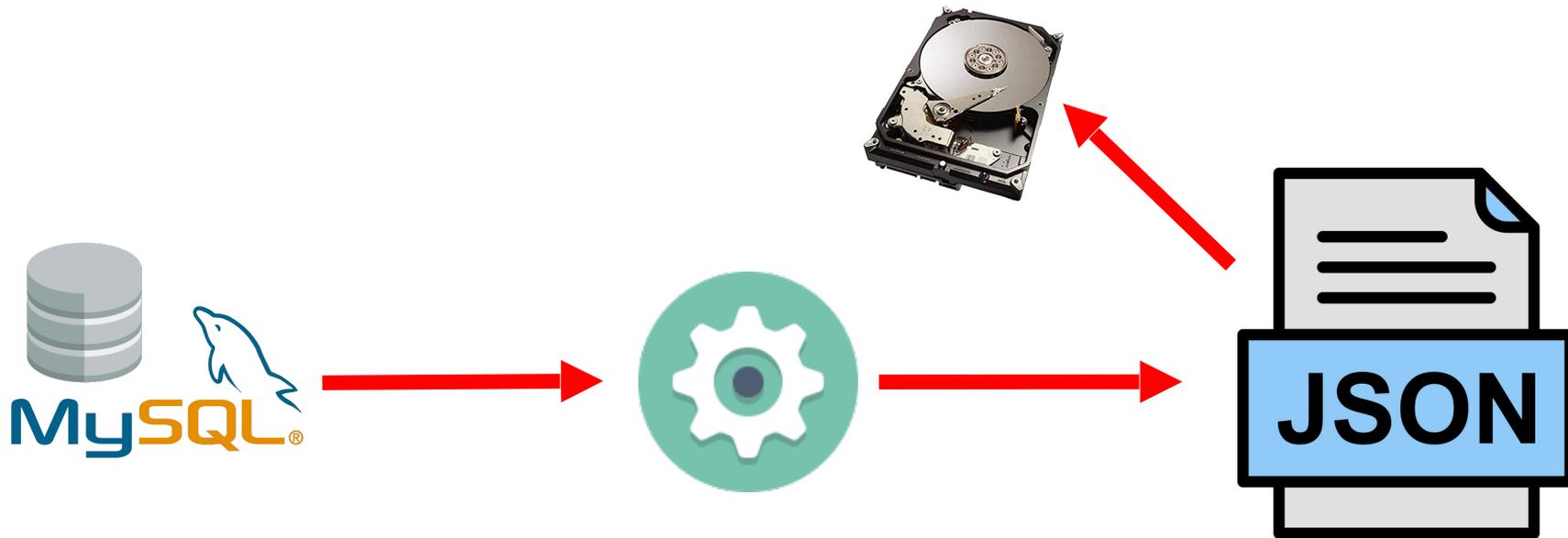
- MYISAM

- INNODB

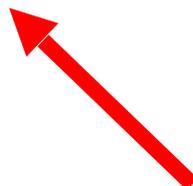
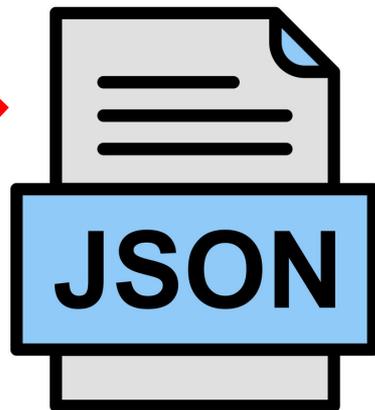
- MEMORY



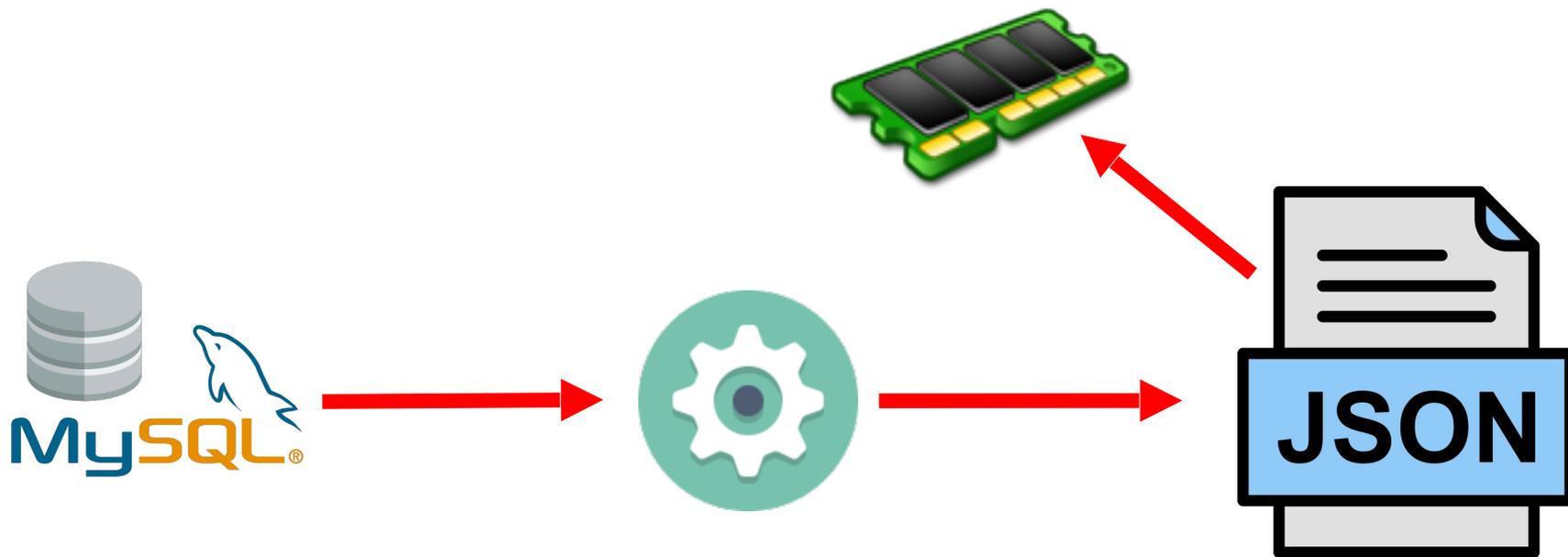
# Json을 Disk에 올려서 사용함



# Disk IO 떡상



# Memory에 올려서 사용



# /dev/shm 사용 가능



```
Filesystem      Size  Used Avail Use% Mounted on
/dev/mmcblk0p1  30G   15G   14G   54% /
none            1.8G   0    1.8G   0% /dev
tmpfs           2.0G   0    2.0G   0% /dev/shm
tmpfs           2.0G   21M   2.0G   2% /run
tmpfs           5.0M   4.0K   5.0M   1% /run/lock
tmpfs           2.0G   0    2.0G   0% /sys/fs/cgroup
tmpfs           397M   0    397M   0% /run/user/1000
```

[https://leekyoungil.github.io/blog/2019/09/29/About\\_the\\_dev\\_shm\\_easy\\_to\\_use\\_the\\_Linux\\_ramdisk.html](https://leekyoungil.github.io/blog/2019/09/29/About_the_dev_shm_easy_to_use_the_Linux_ramdisk.html)

# Local Cache / Global Variable 사용



**요즘도 충분히 사용 가능한 방법**

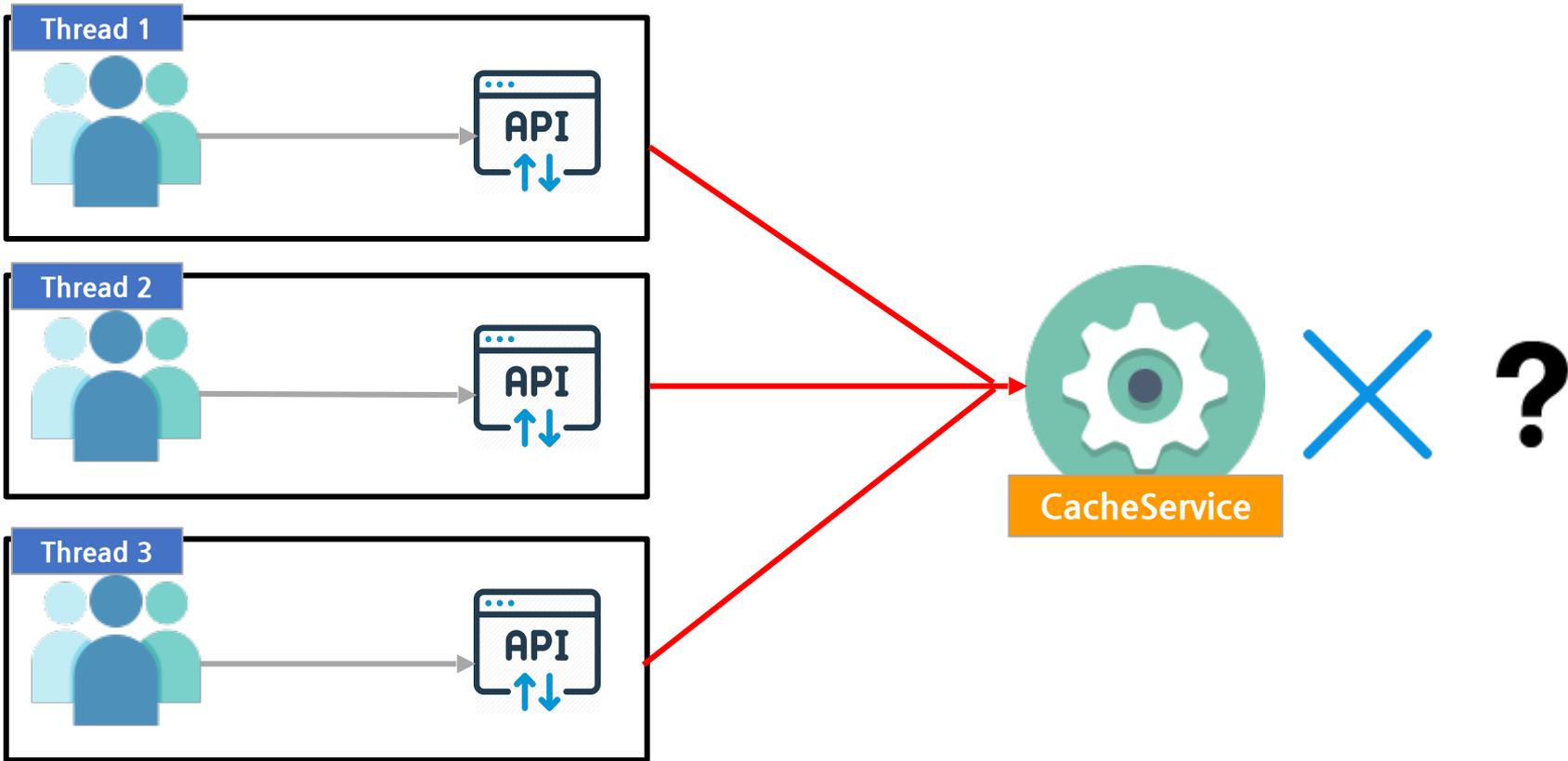
**Map 객체에 저장하면 쉽겠죠?**

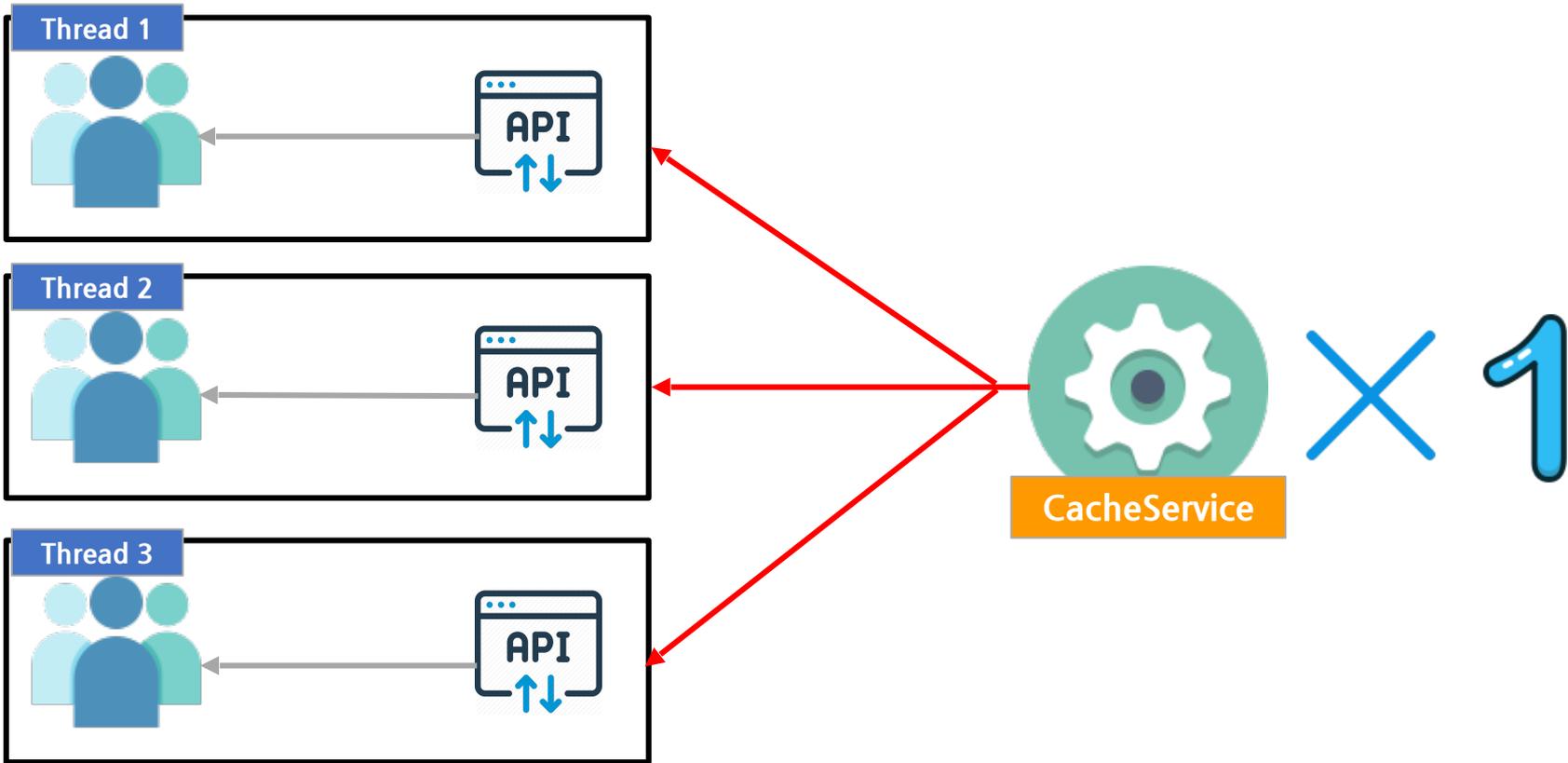
**아니요. 쉽지 않습니다.**

**주의해야 할 점이 있습니다.**

**Spring Bean** 안에 넣어 사용할 때

**CacheService** 라고 해볼게요.

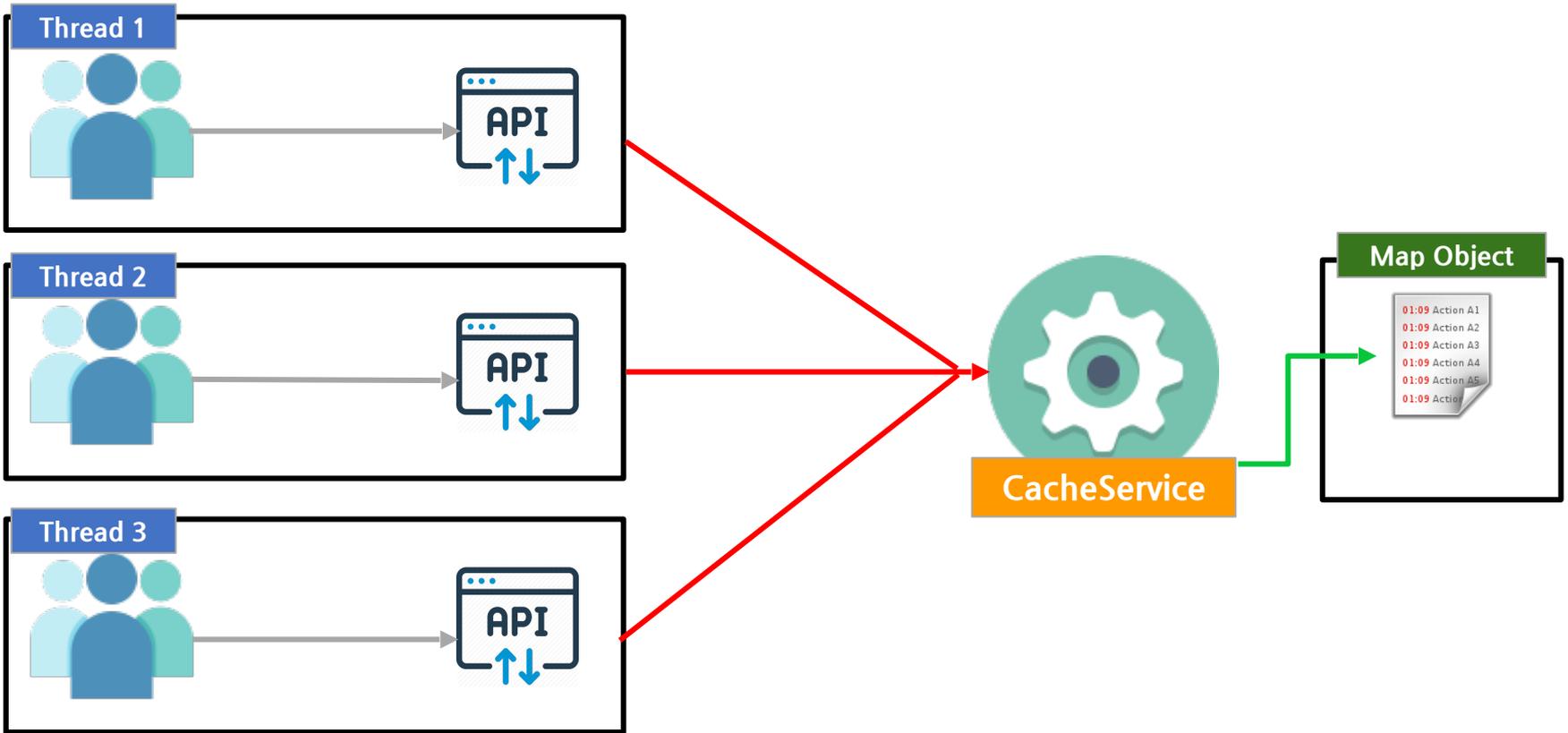


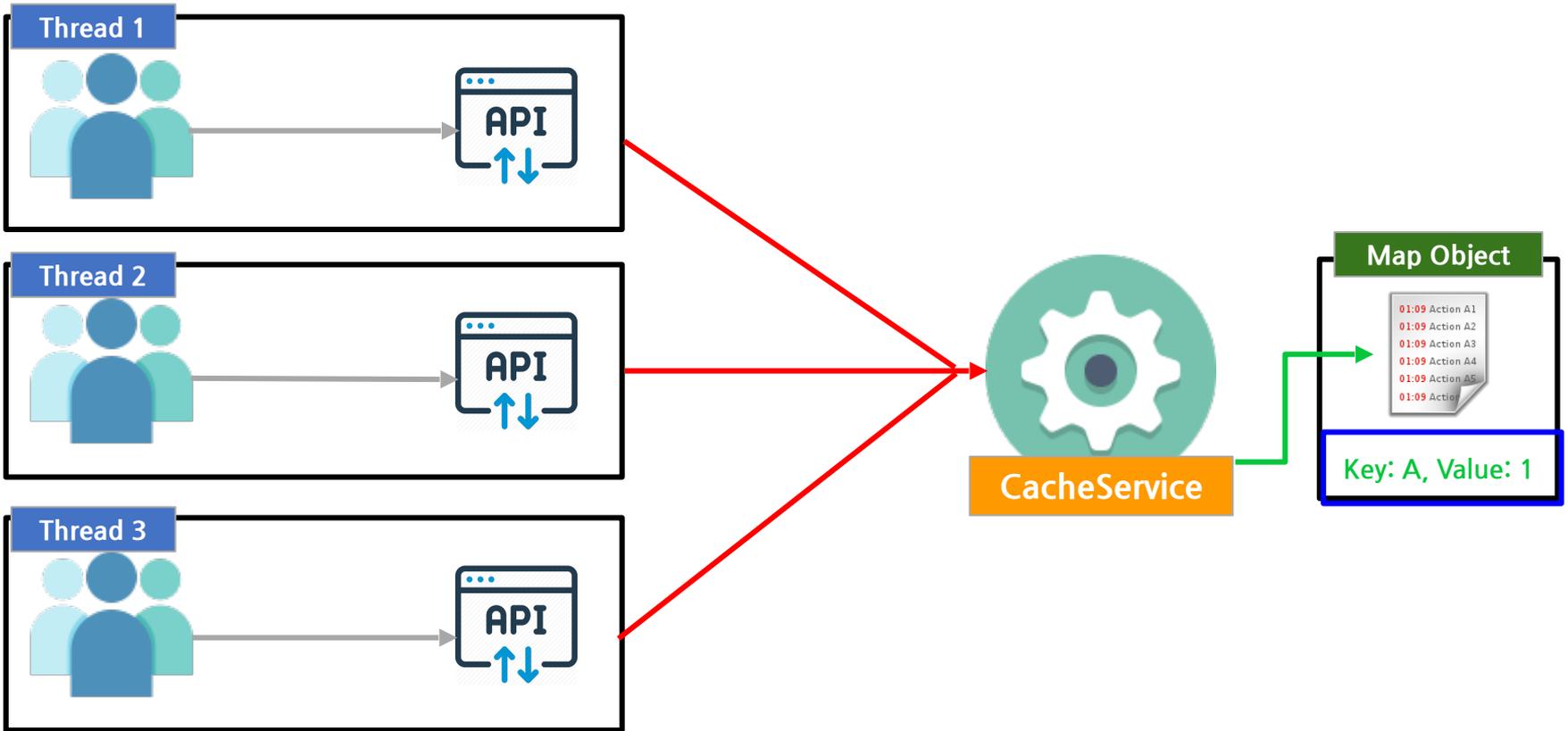


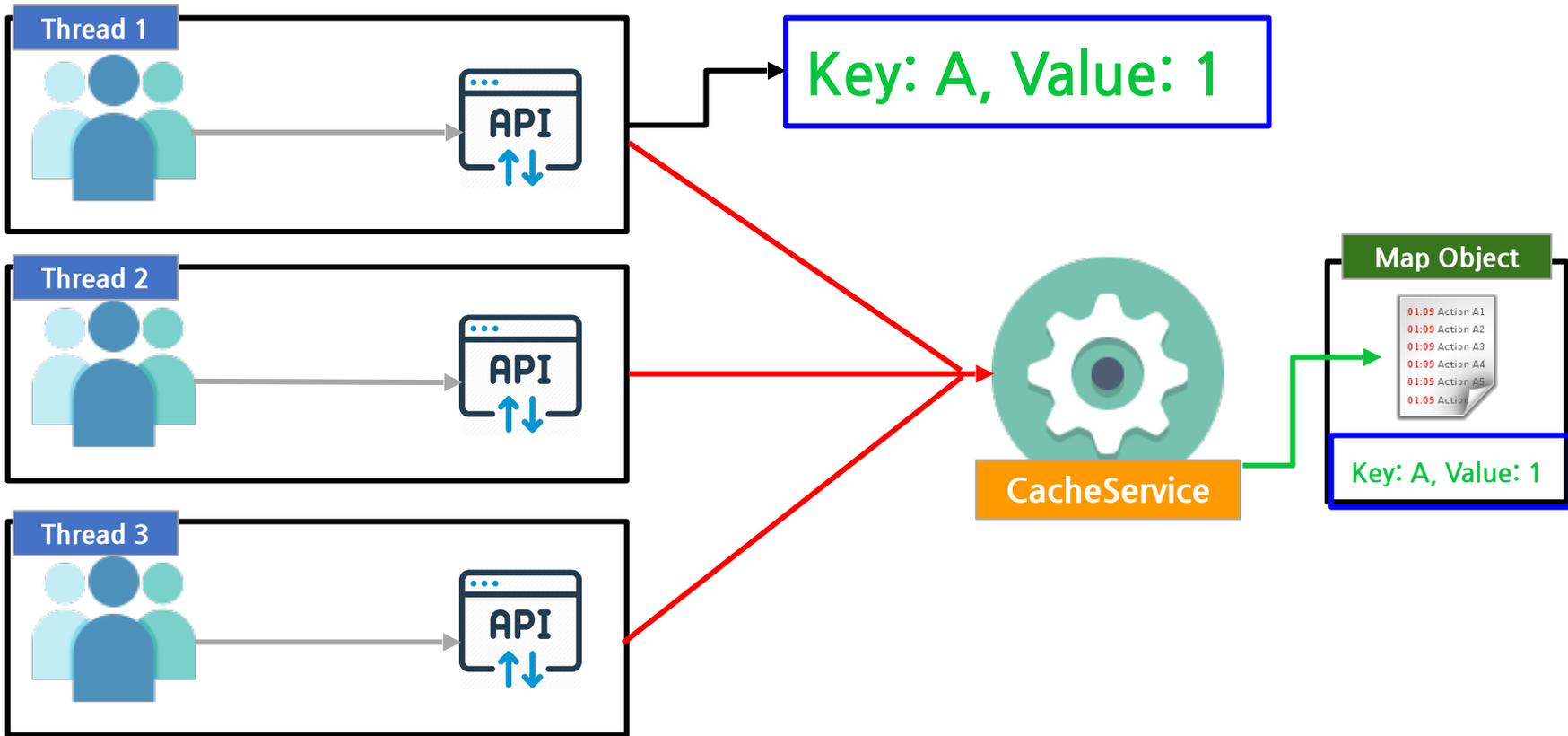


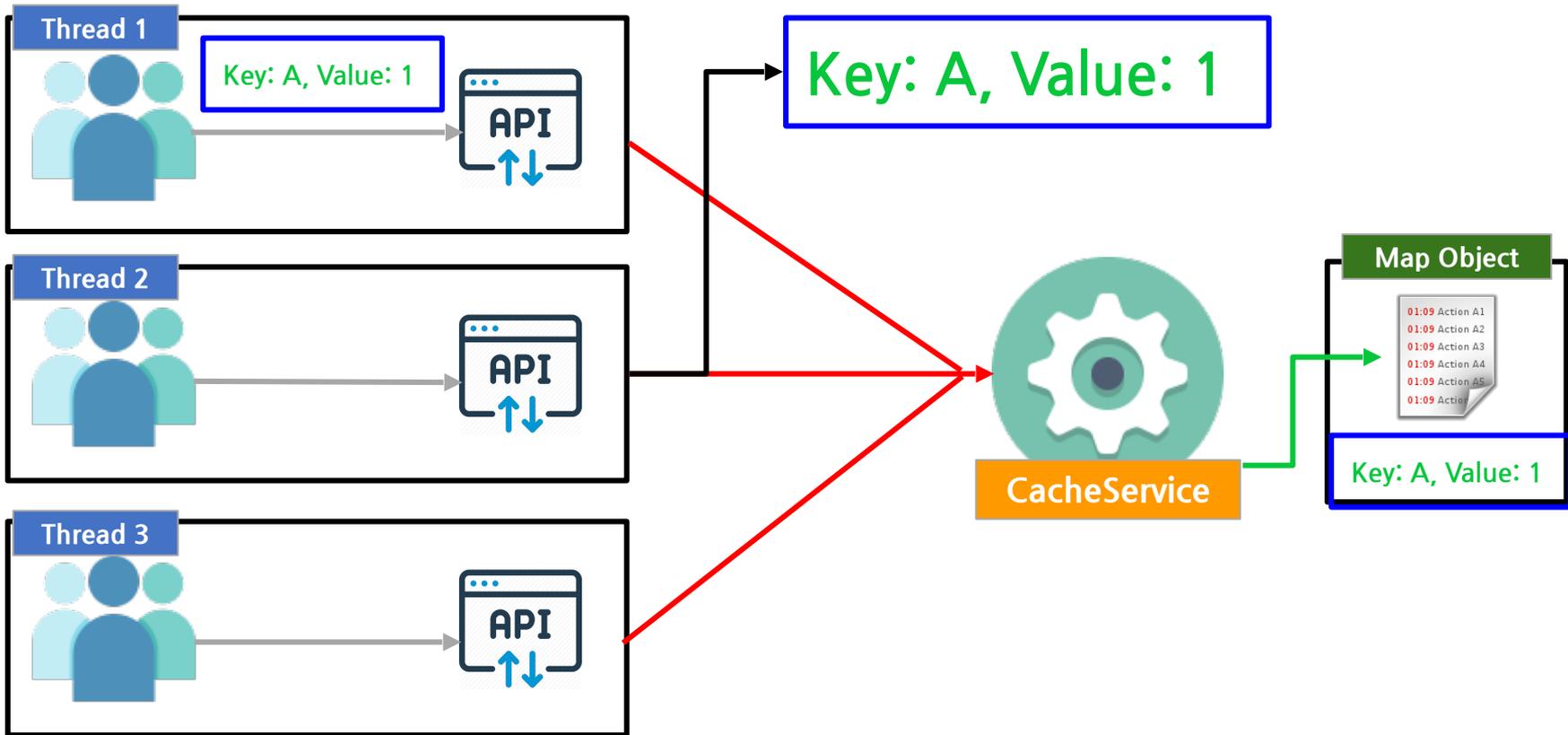
Spring Bean  
객체는  
하나다....

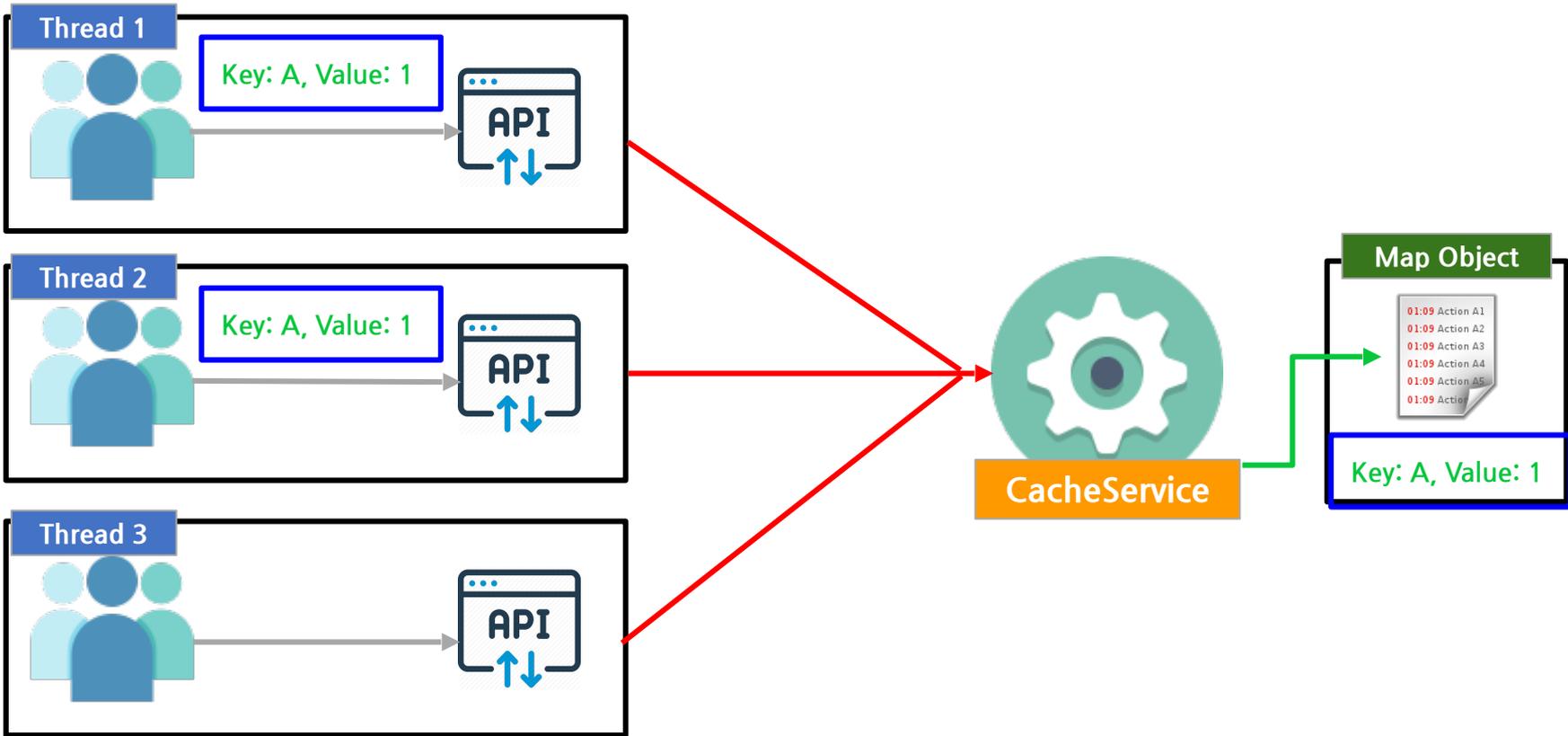


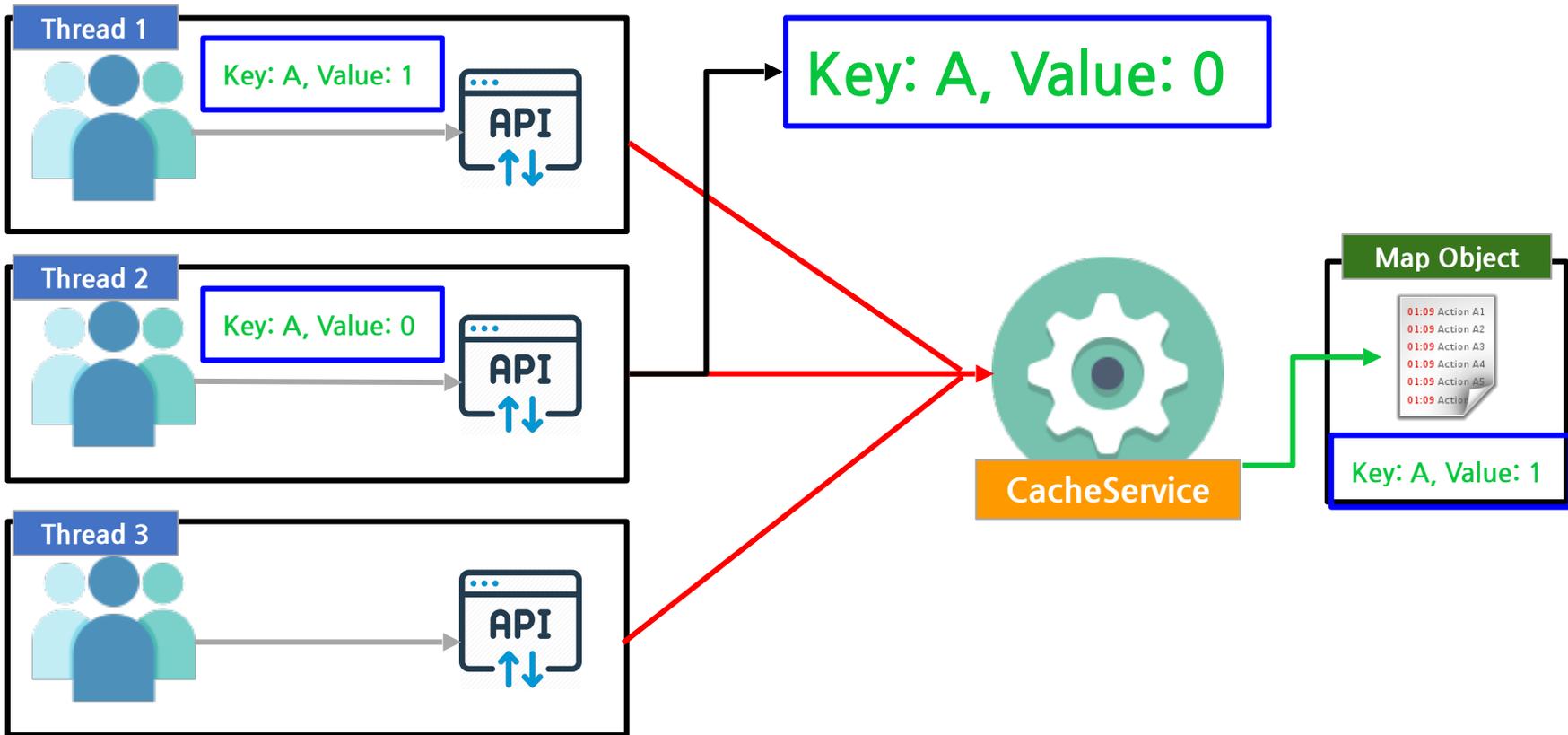


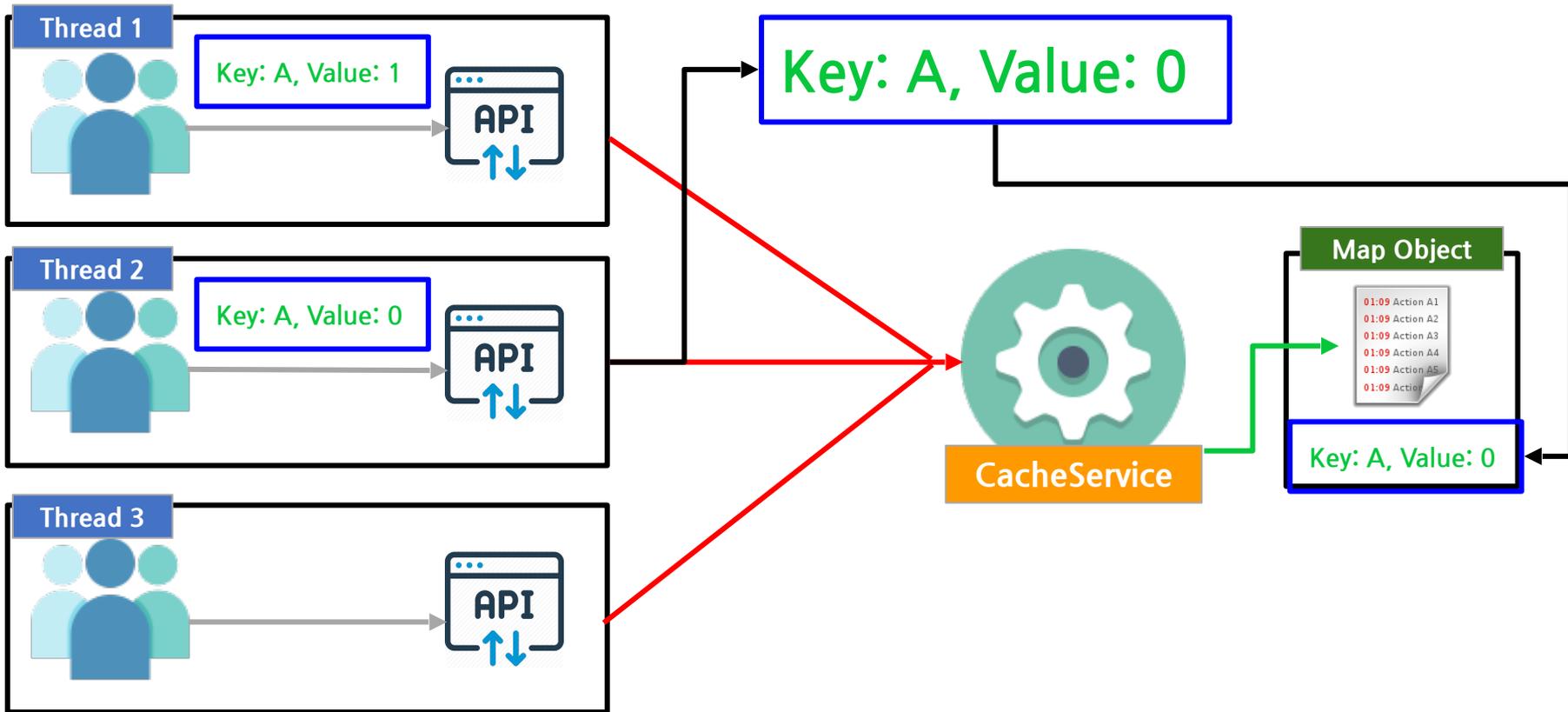


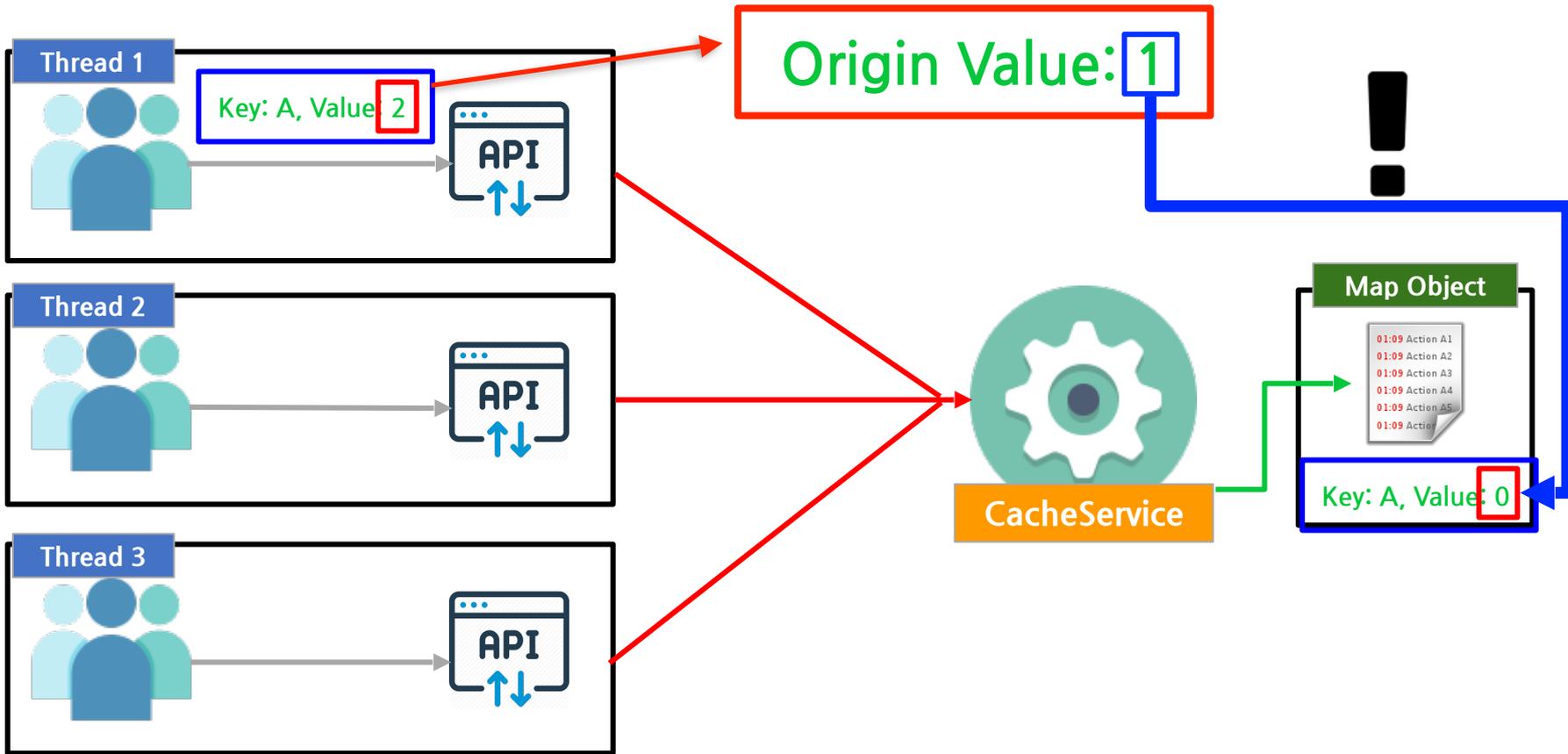












**Java에서는 동시성 문제를  
어떻게 해결할까요?**

# Synchronized Volatile Atomic

**Synchronized**  
**Volatile**  
**Atomic**

@Component

```
public class CellCache {
```

```
    private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```

```
    public Map<String, CacheDetailDataVO> getGlobalCacheMap() {
```

```
        return this.globalCacheMap;
```

```
    }
```

```
    public void setGlobalCacheMapDetail(String key, CacheDetailDataVO data) {
```

```
        this.globalCacheMap.set(key, data);
```

```
    }
```

```
}
```

@Component

```
public class CellCache {
```

```
    private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```

```
    public Map<String, CacheDetailDataVO> getGlobalCacheMap() {
```

```
        return this.globalCacheMap;
```

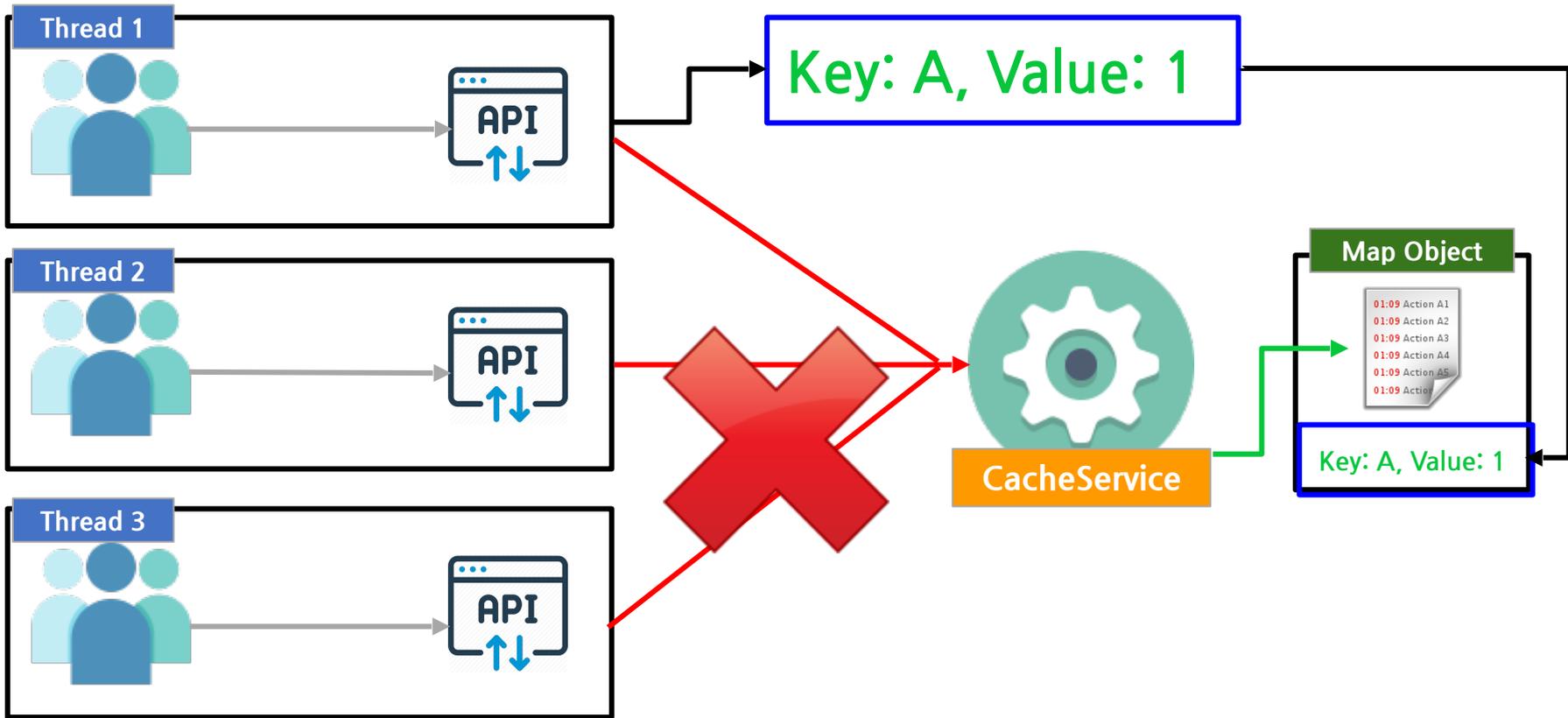
```
    }
```

```
    public synchronized void setGlobalCacheMapDetail(String key, CacheDetailDataVO data) {
```

```
        this.globalCacheMap.set(key, data);
```

```
    }
```

```
}
```



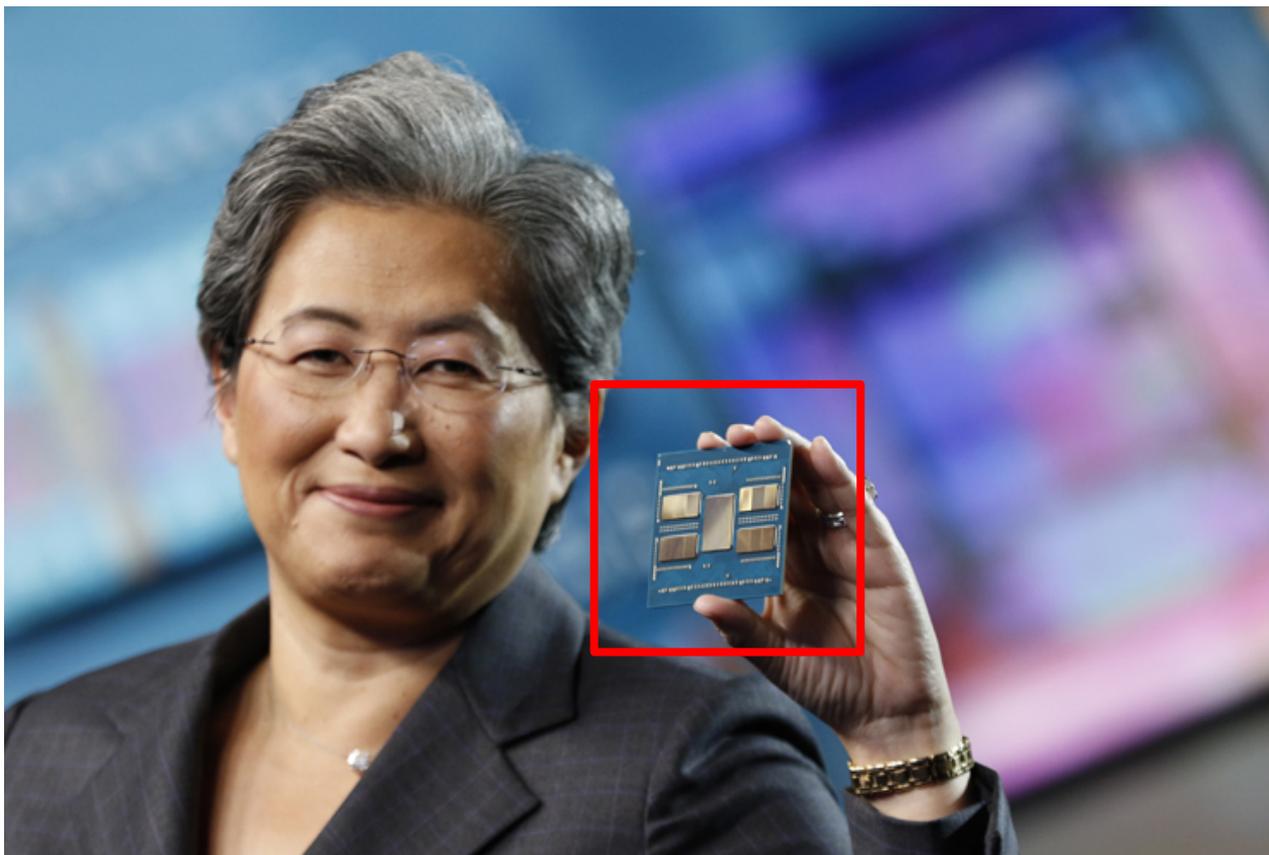


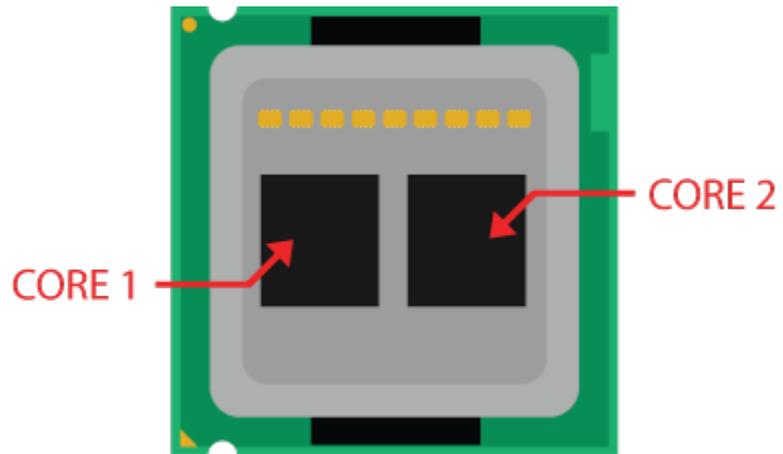
**최고로 안전합니다.**



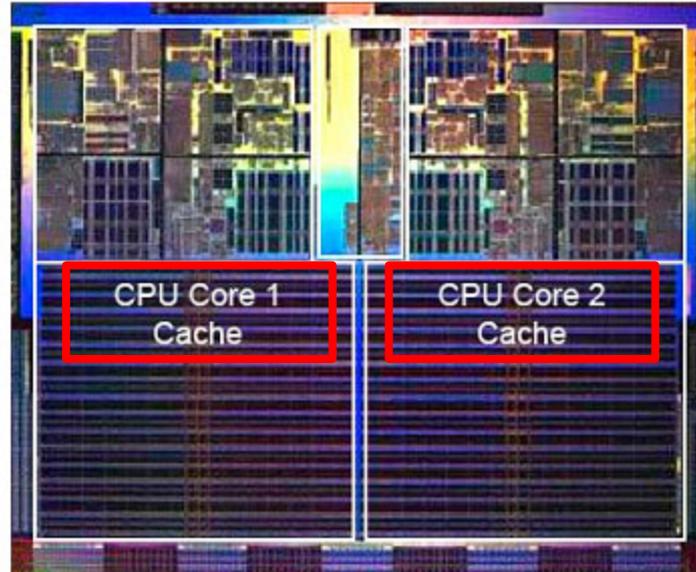
새누리 백세인생 로고송 포기  
**"너무 비싸 포기했다 전해라"**

**Synchronized**  
**Volatile**  
**Atomic**





**MULTI-CORE CPU**



@Component

**public class** CellCache {

```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```

```
public Map<String, CacheDetailDataVO> getGlobalCacheMap() {
```

```
    return this.globalCacheMap;
```

```
}
```

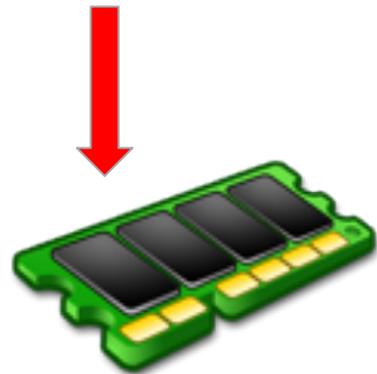
```
public void setGlobalCacheMapDetail(String key, CacheDetailDataVO data) {
```

```
    this.globalCacheMap.set(key, data);
```

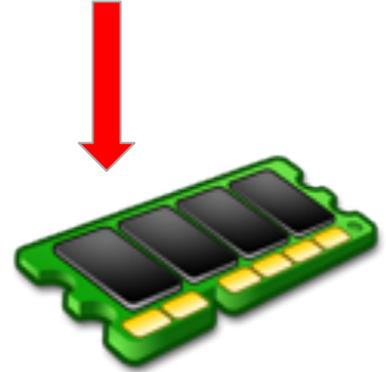
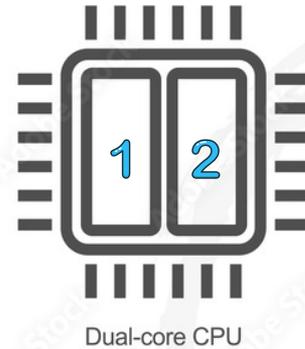
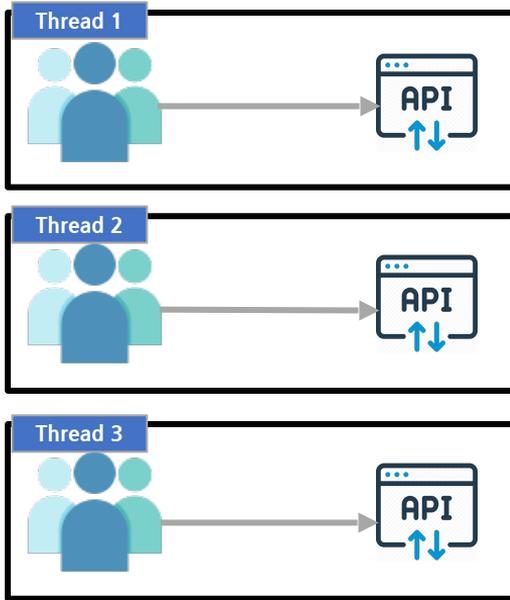
```
}
```

```
}
```

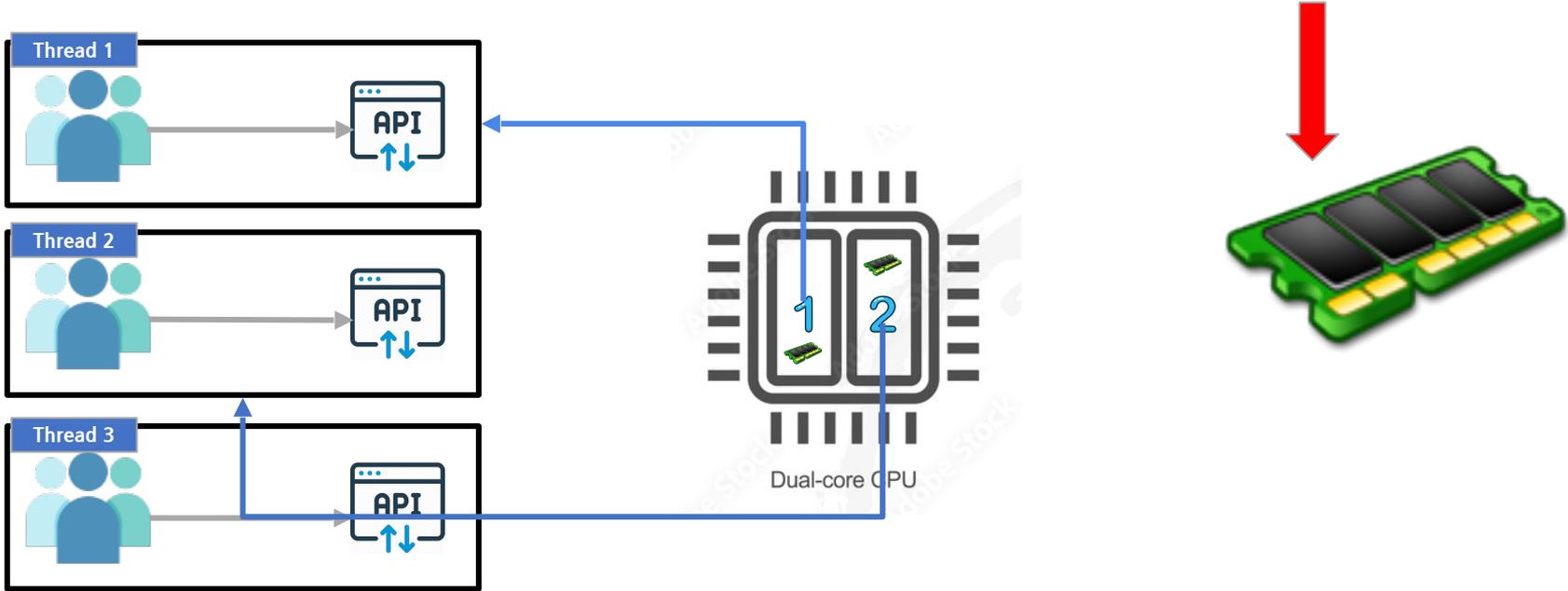
```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



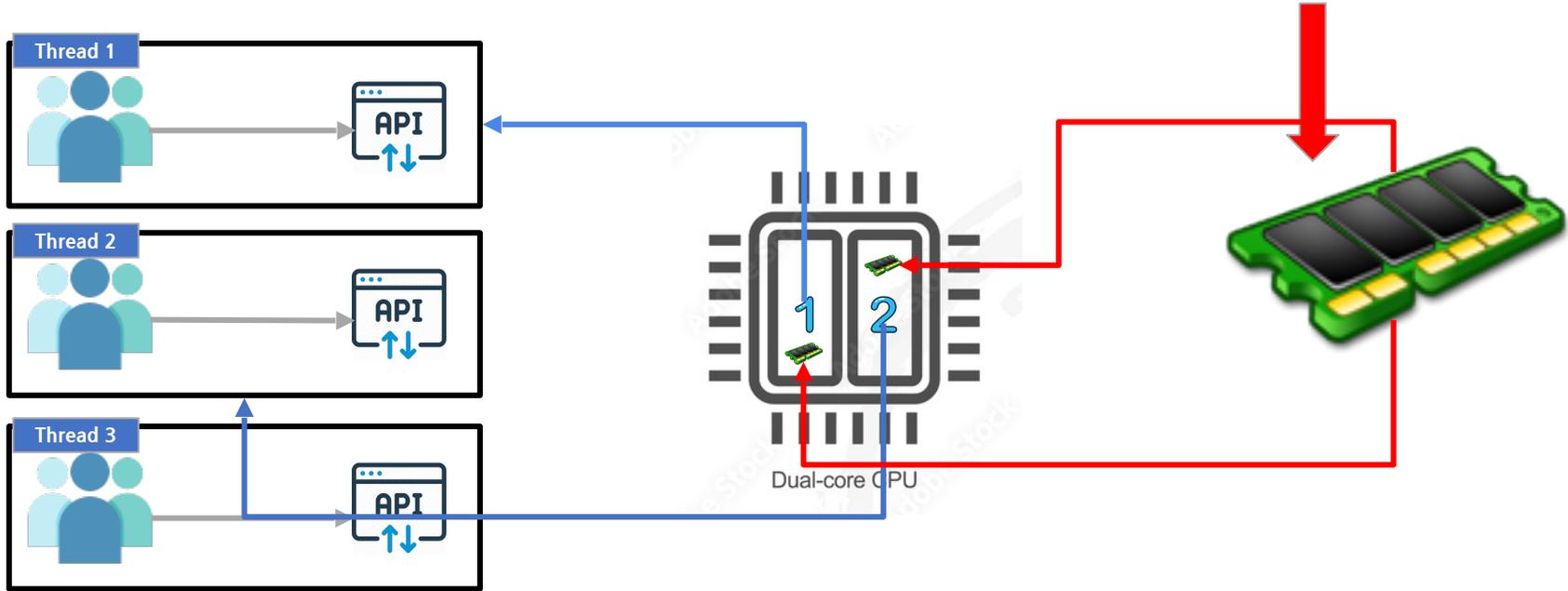
```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



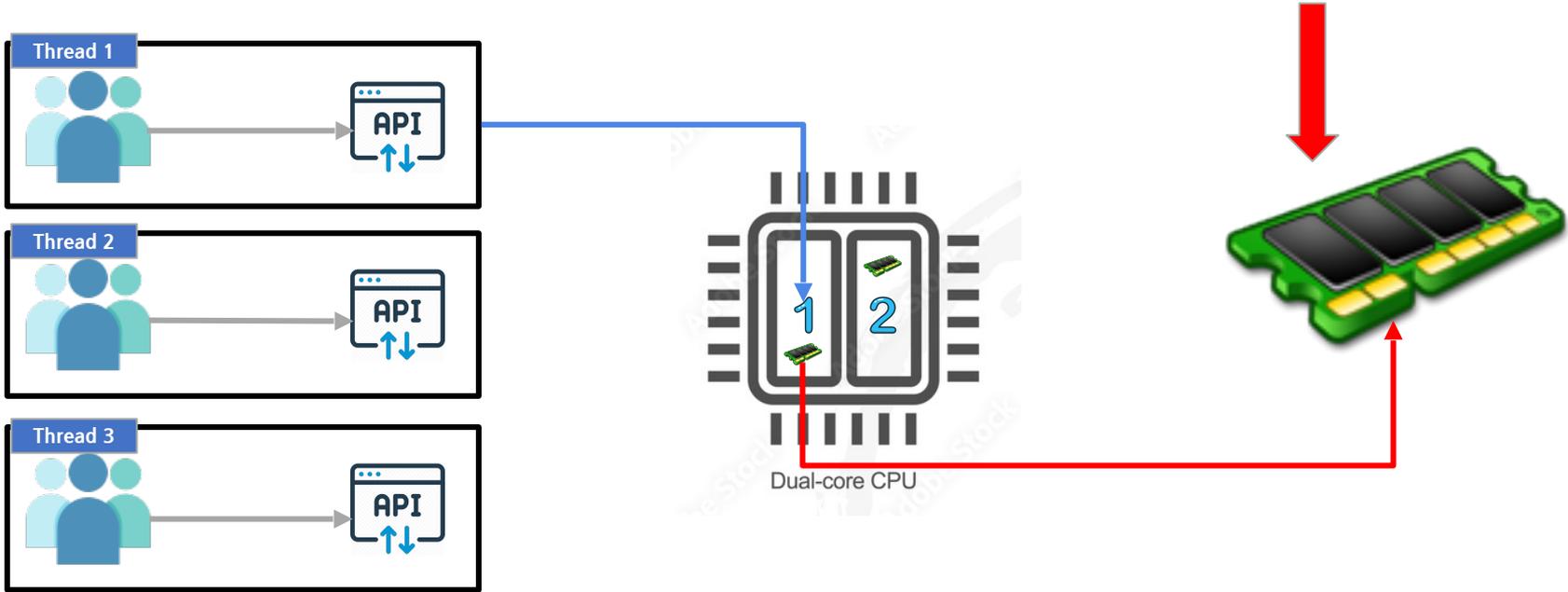
```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



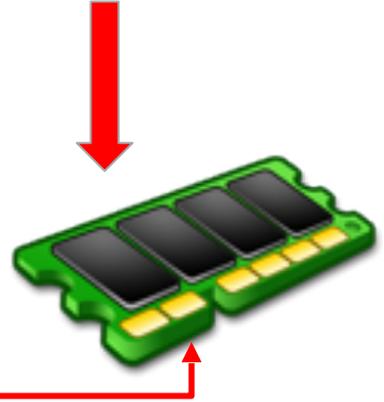
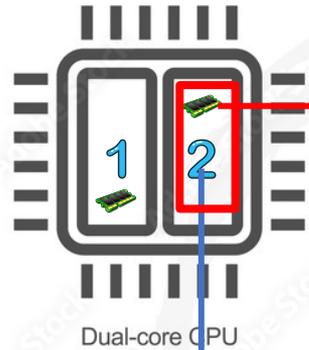
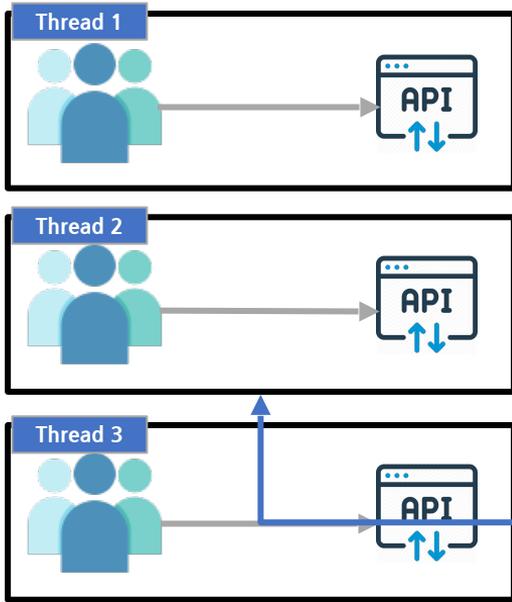
```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



@Component

```
public class CellCache {
```

```
    private volatile Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```

```
    public Map<String, CacheDetailDataVO> getGlobalCacheMap() {
```

```
        return this.globalCacheMap;
```

```
    }
```

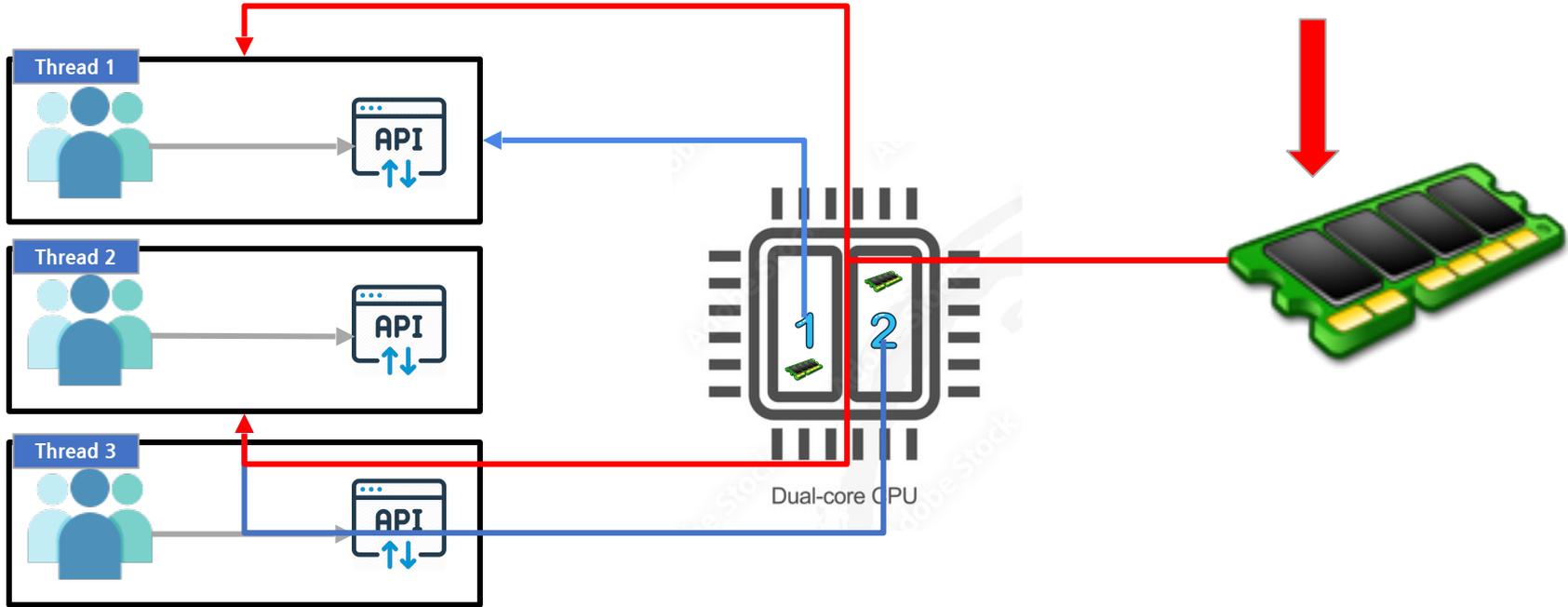
```
    public void setGlobalCacheMapDetail(String key, CacheDetailDataVO data) {
```

```
        this.globalCacheMap.set(key, data);
```

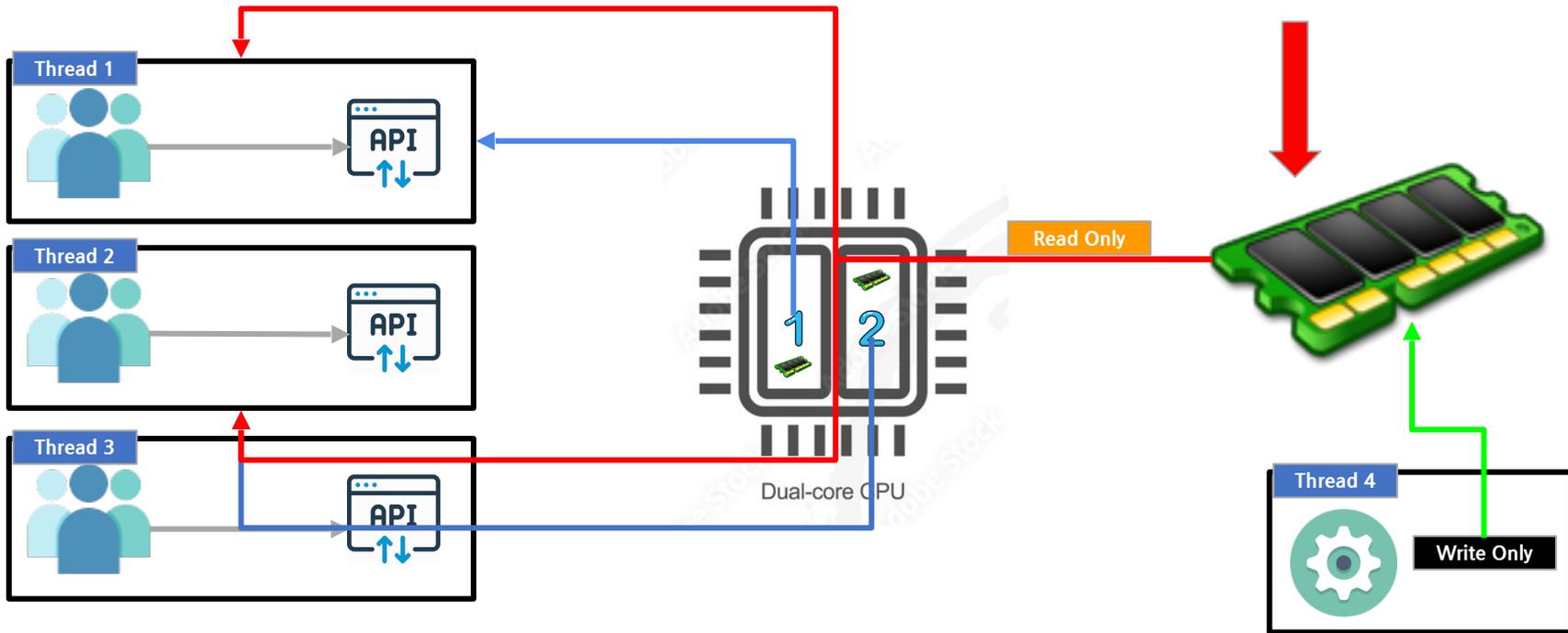
```
    }
```

```
}
```

```
private Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```



```
private volatile Map<String, CacheDetailDataVO> globalCacheMap = new HashMap<>();
```

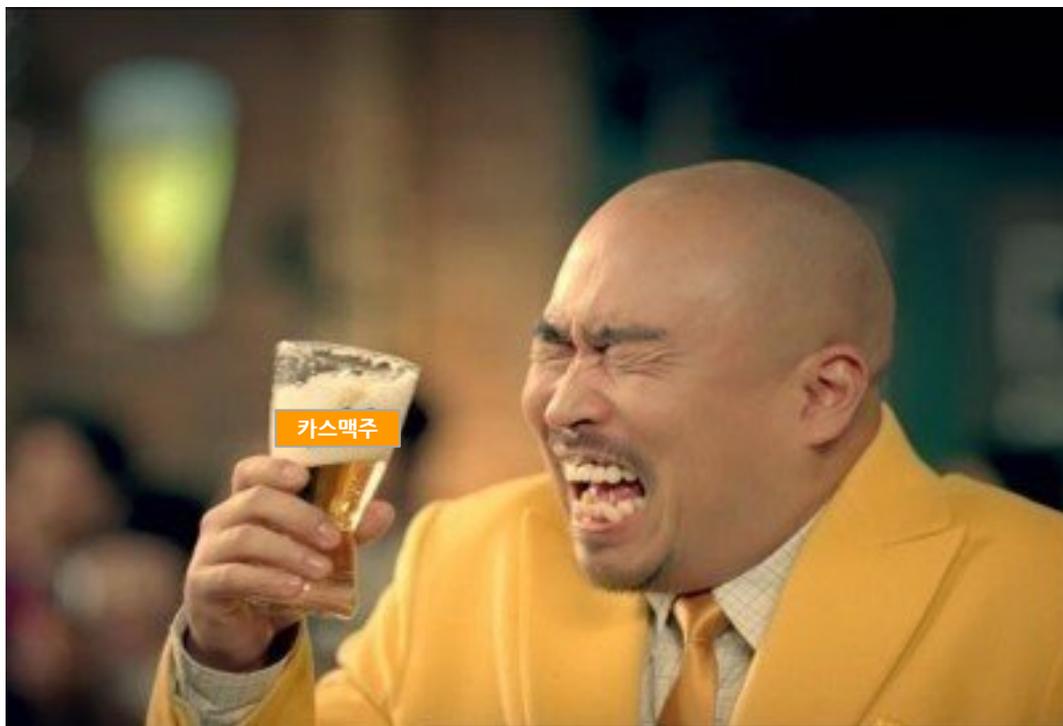


**Synchronized**  
**Volatile**  
**Atomic**

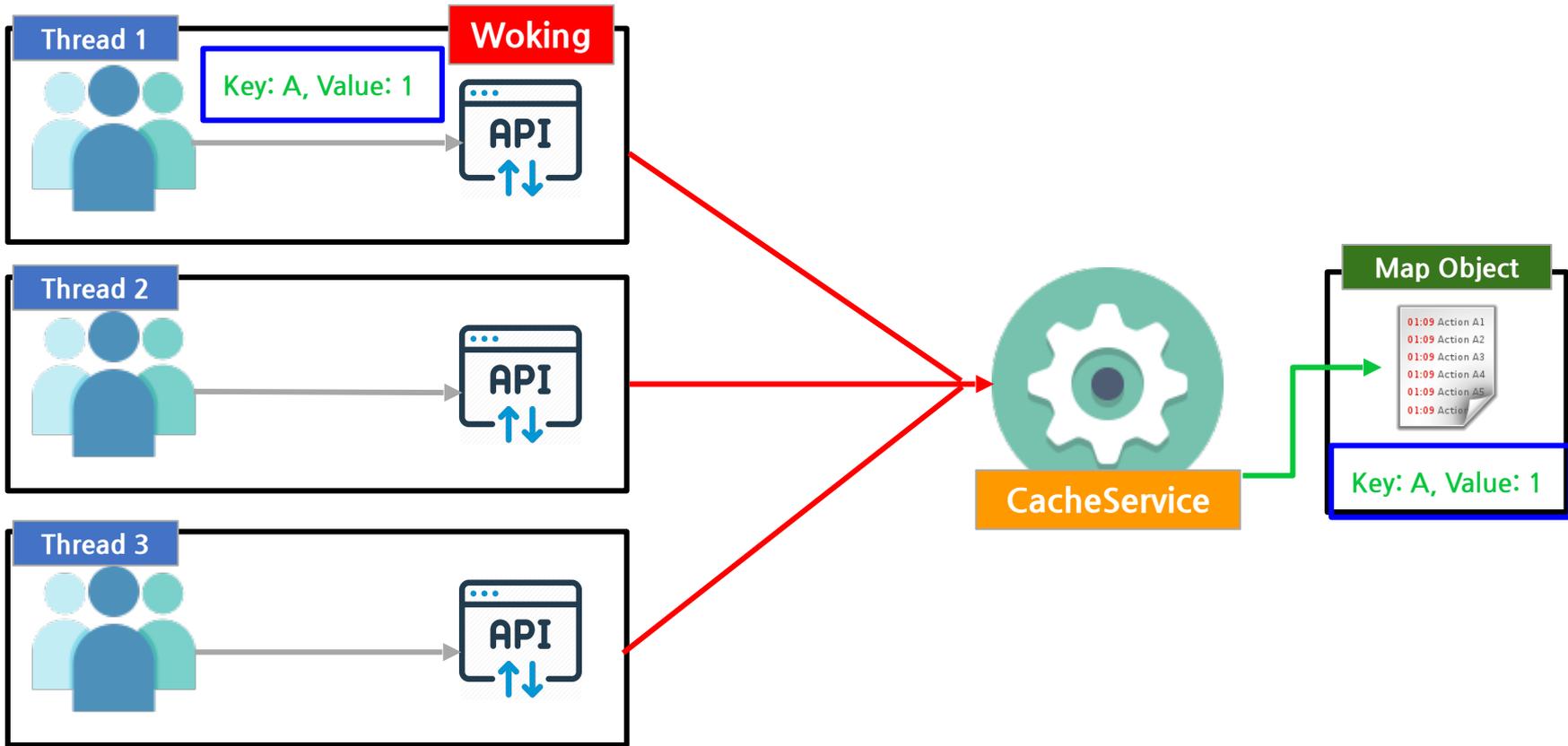
**java.util.concurrent.atomic**

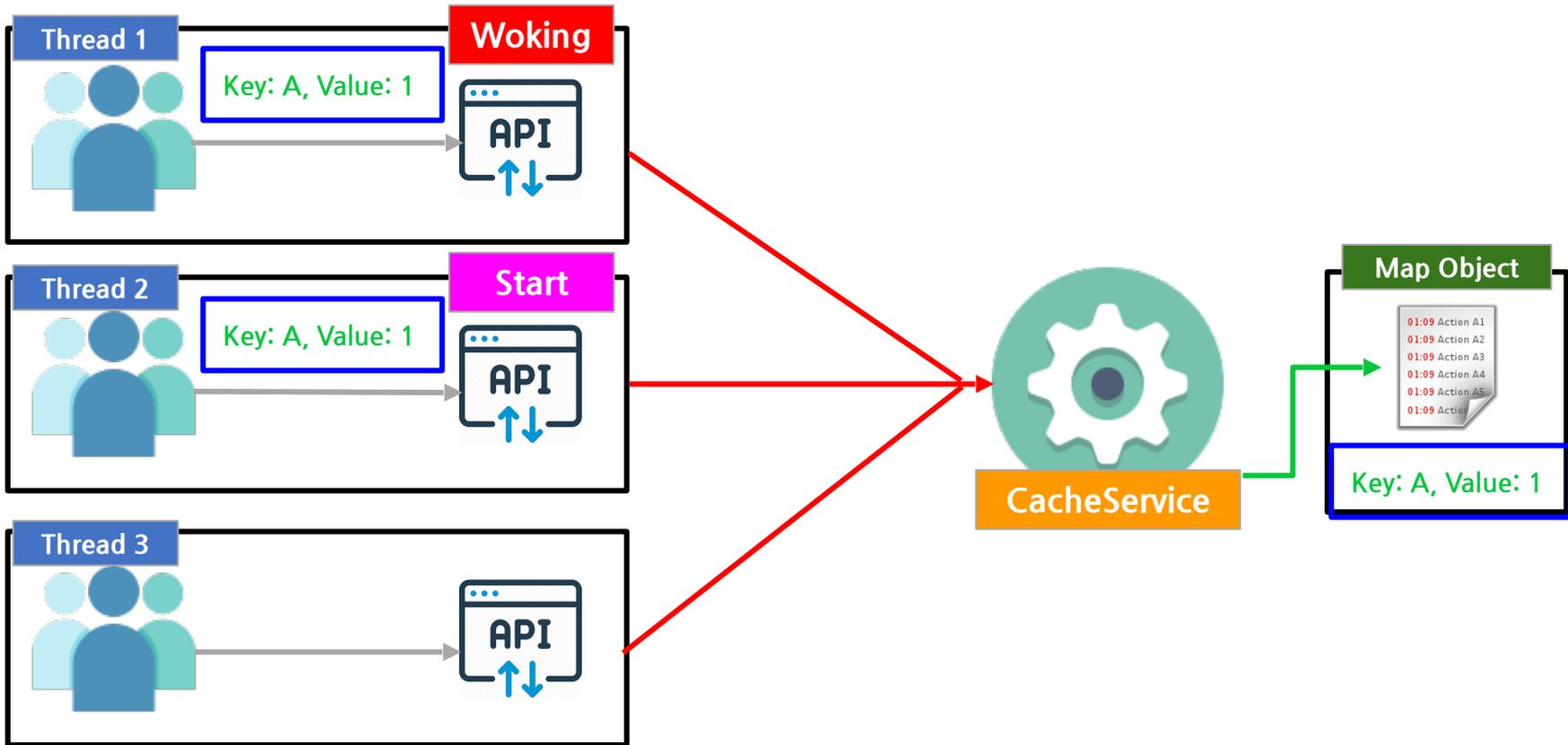
**AtomicInteger**  
**AtomicLong**  
**AtomicString**  
**AtomicBoolean**  
**AtomicReference**

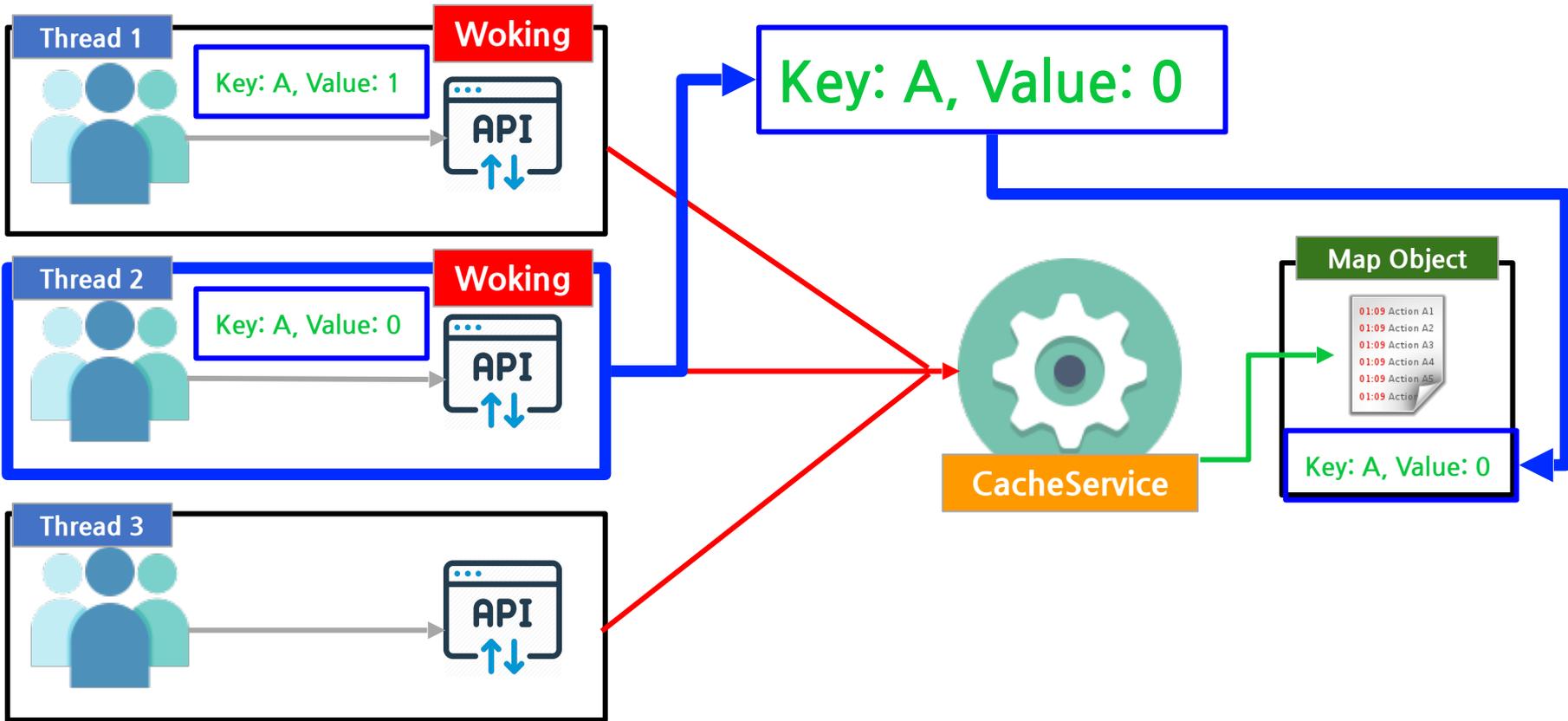
카스(CAS) 연산을 통해  
동시성을 만족합니다.

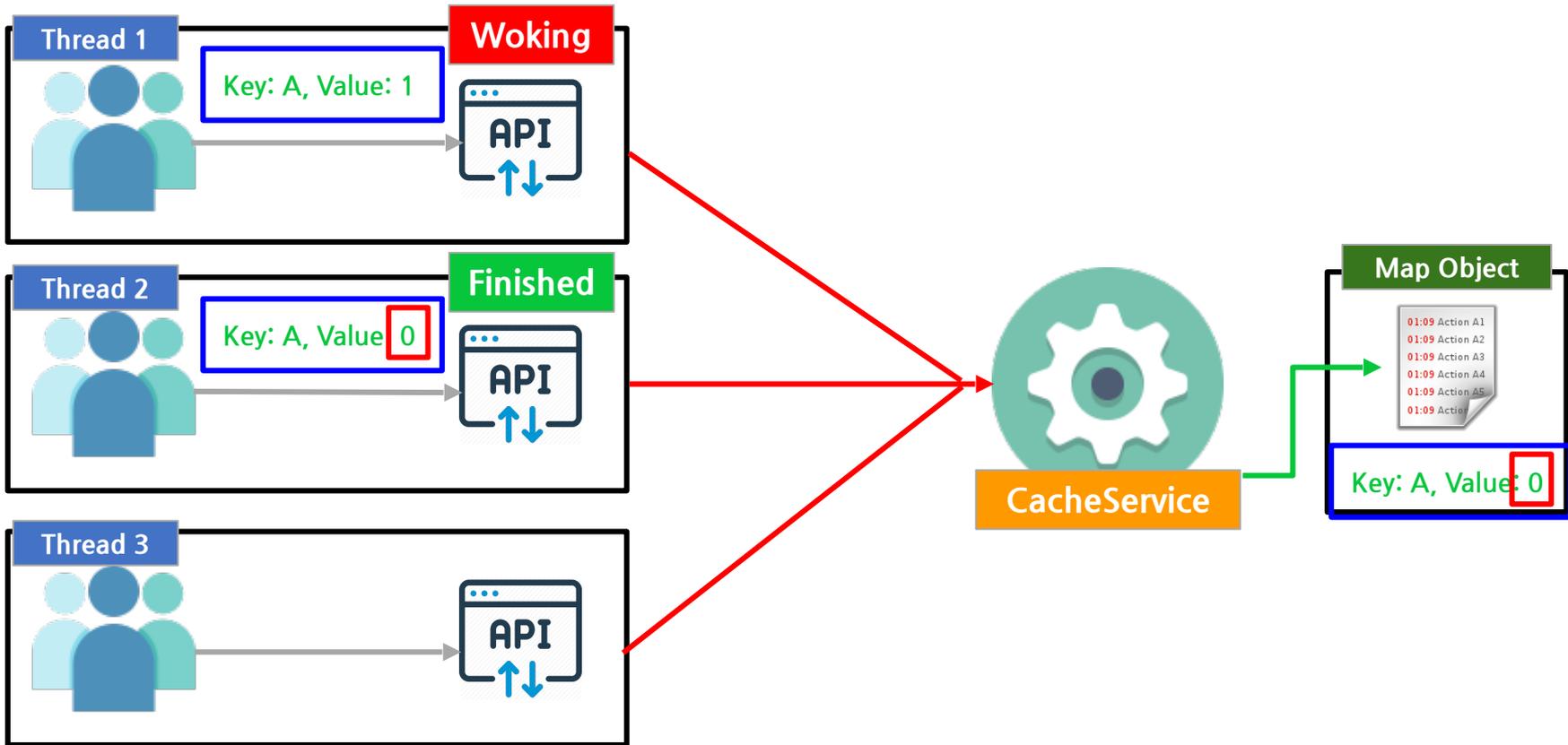


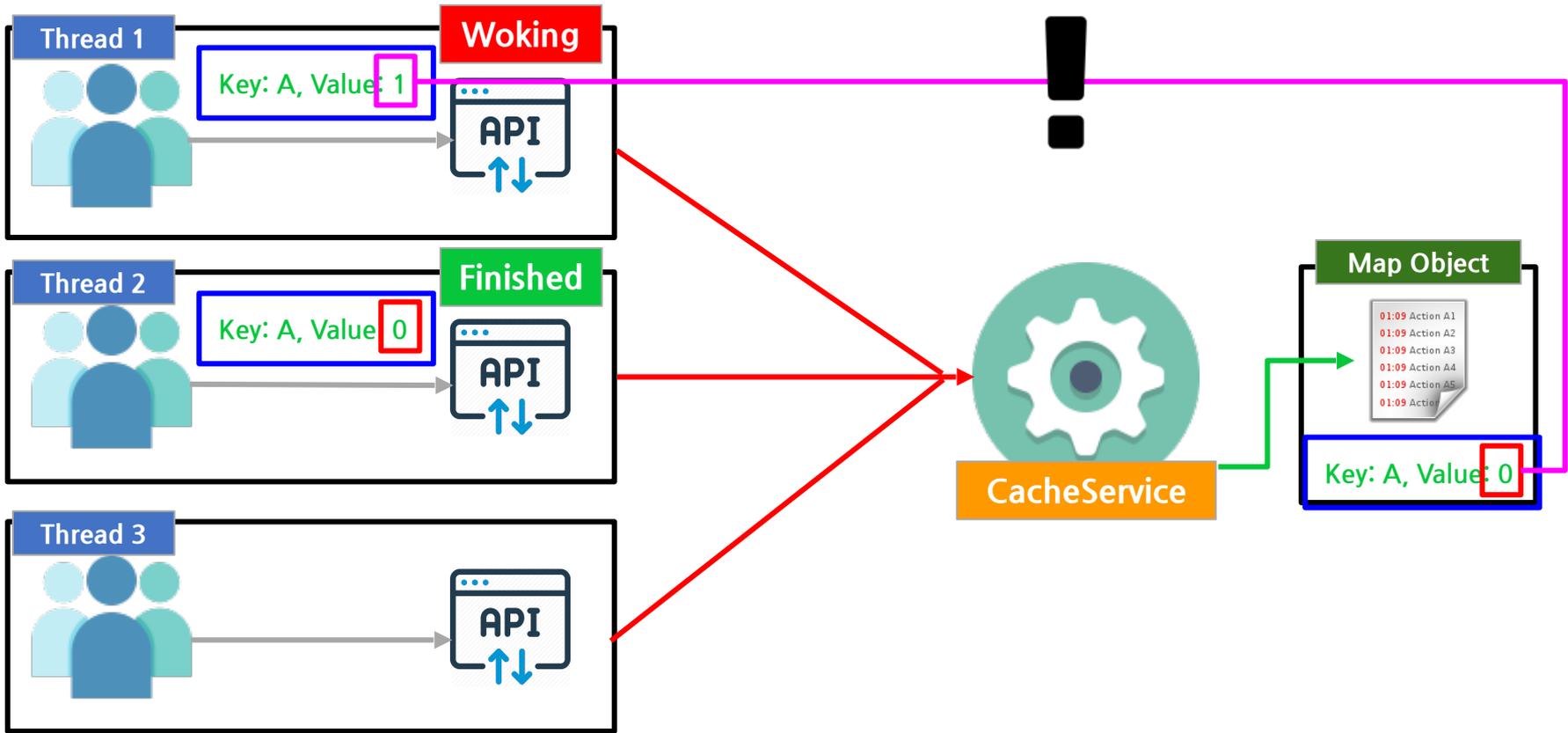
# Compare And Swap

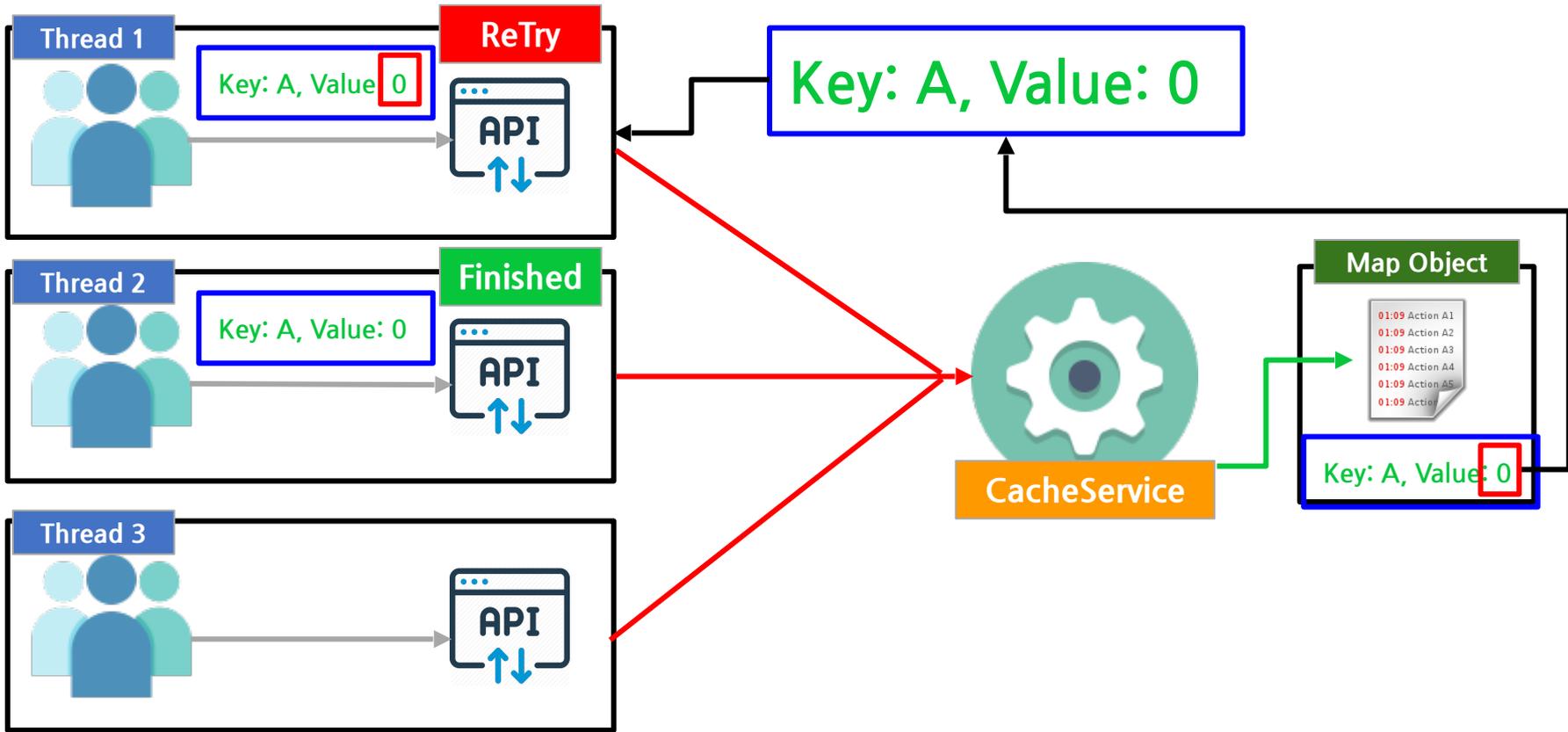














APRIL 1987

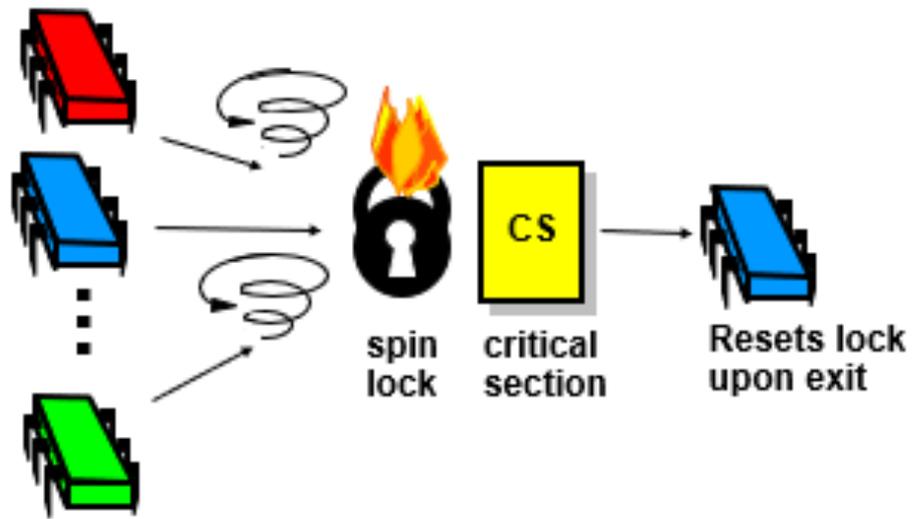
# COMPUTER

## No Silver Bullet

*Fred Brooks  
on  
Avoiding Horrors  
in the Software  
Engineering  
Process*

 THE COMPUTER SOCIETY  
OF THE IEEE

 THE INSTITUTE OF ELECTRICAL AND  
ELECTRONICS ENGINEERS, INC.

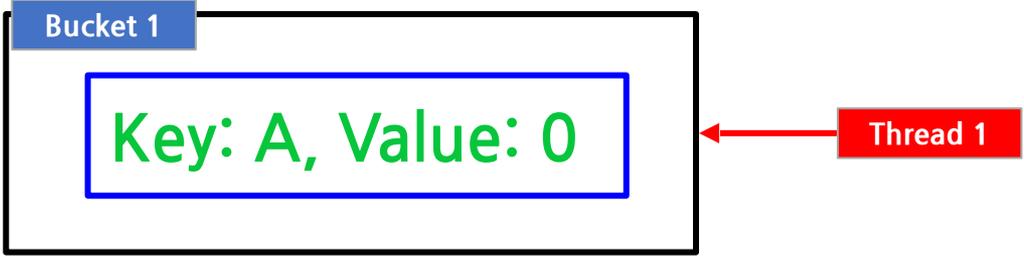


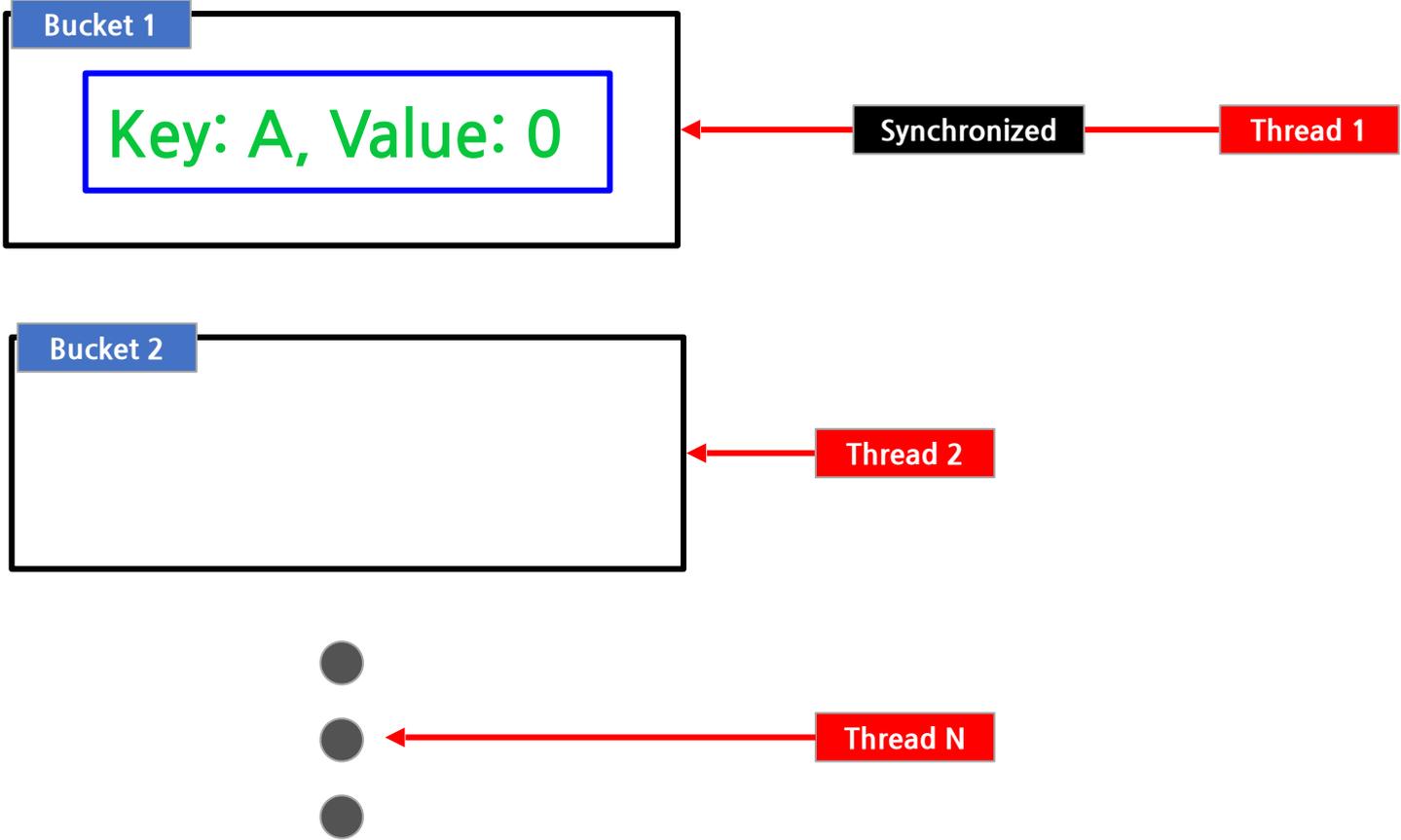
그래도 **Synchronized**보단 빠릅니다.

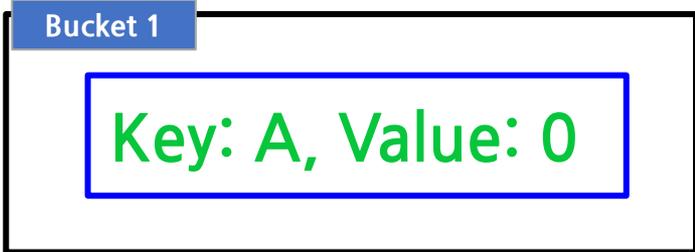
**ConcurrentHashMap은  
CAS를 지원합니다.**



**Bucket**이라는 개념이 있습니다.

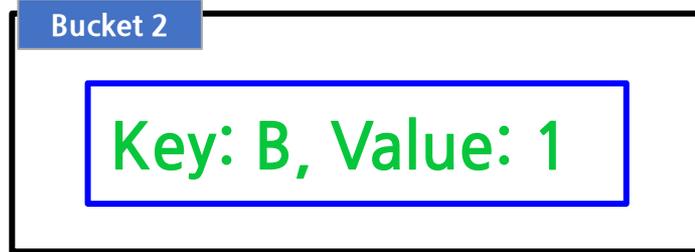






Synchronized

Thread 1



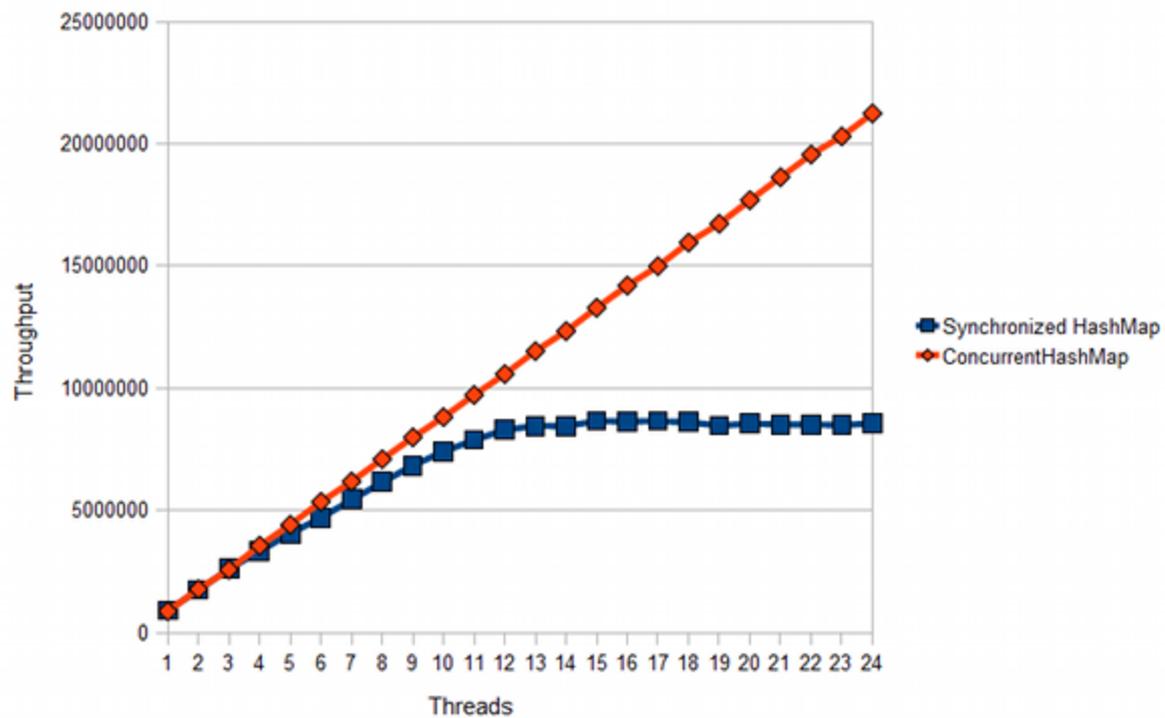
Thread 2



Thread N

# SynchronizedMap

### Scalability of synchronized HashMap vs ConcurrentHashMap

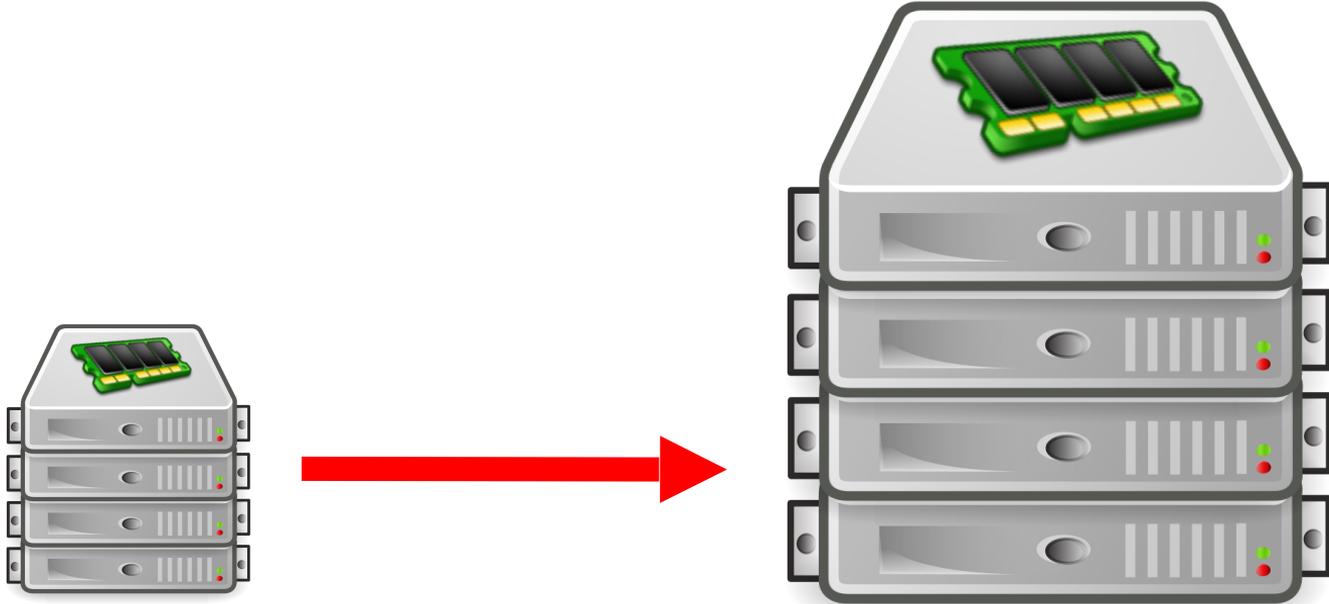


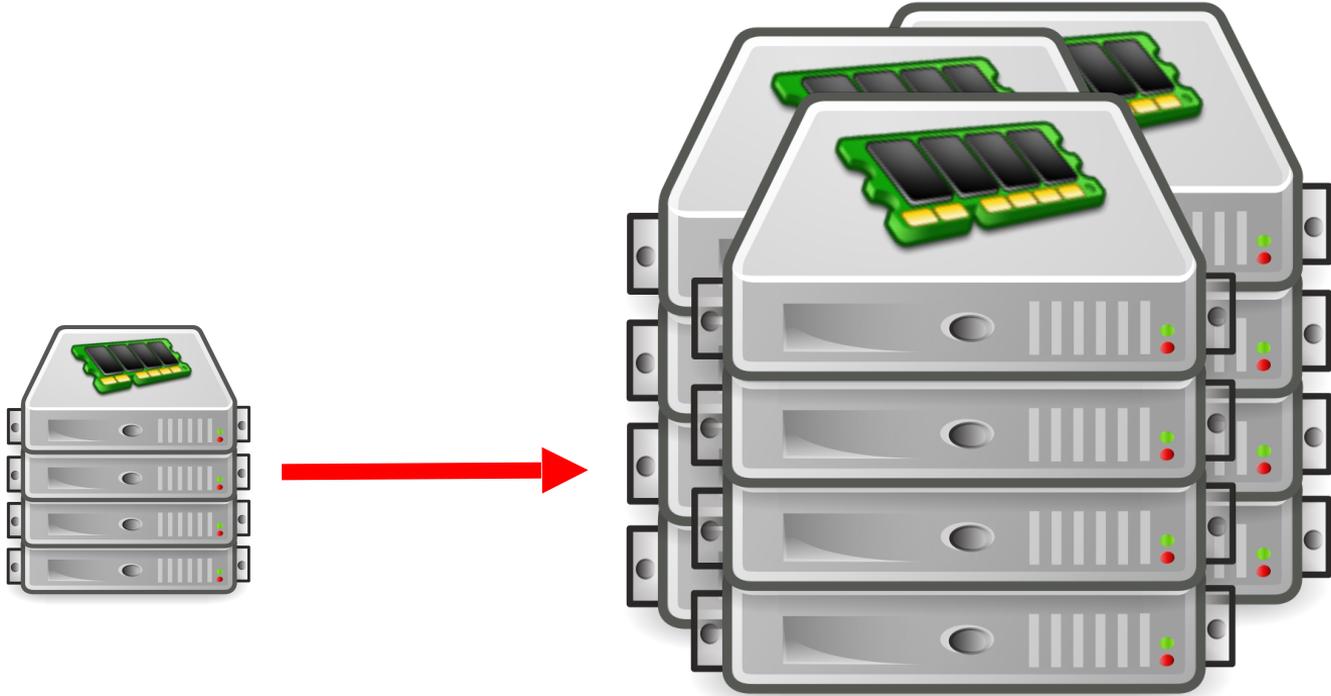
# ConcurrentHashMap

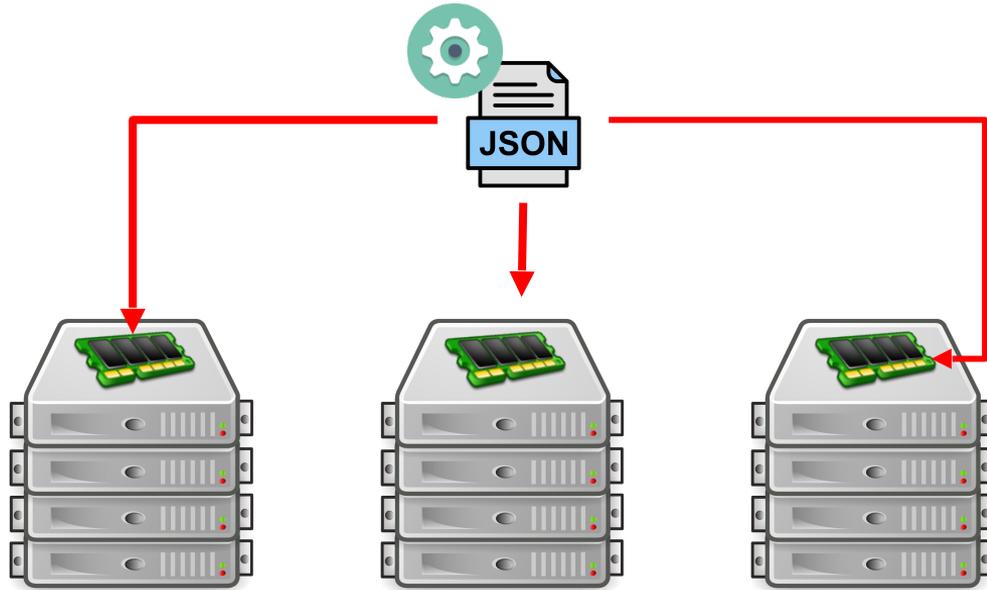
**그런데 큰 문제가 있습니다.**

**이건 서버 1대로 사용할 때 이야기고...**

**처리량이 더 필요할 때**



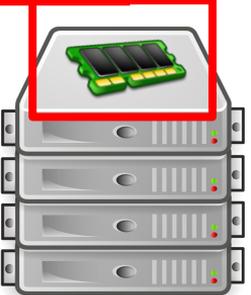






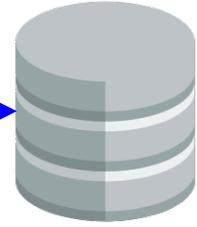
**서버 간 동기화 문제가 발생합니다.**

- Version: 1



- Version: 1

- Version: 1.1로 데이터가 업데이트됨



Version: 1.1

- Version: 1

- Version: 1.1로 데이터가 업데이트됨



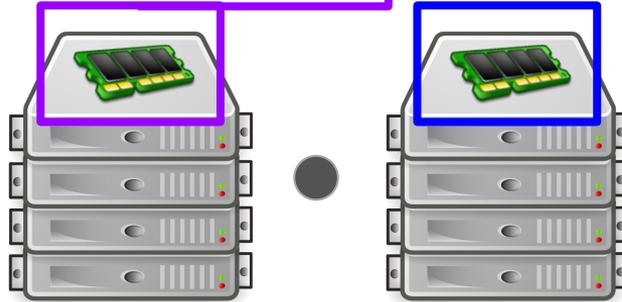
- Version: 1

- Version: 1.1로 데이터가 업데이트됨

- Version: 1.2로 데이터가 업데이트됨



Version: 1.2



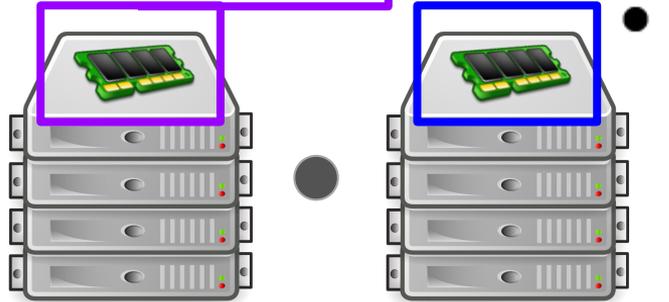
- Version: 1

- Version: 1.1로 데이터가 업데이트됨

- Version: 1.2로 데이터가 업데이트됨



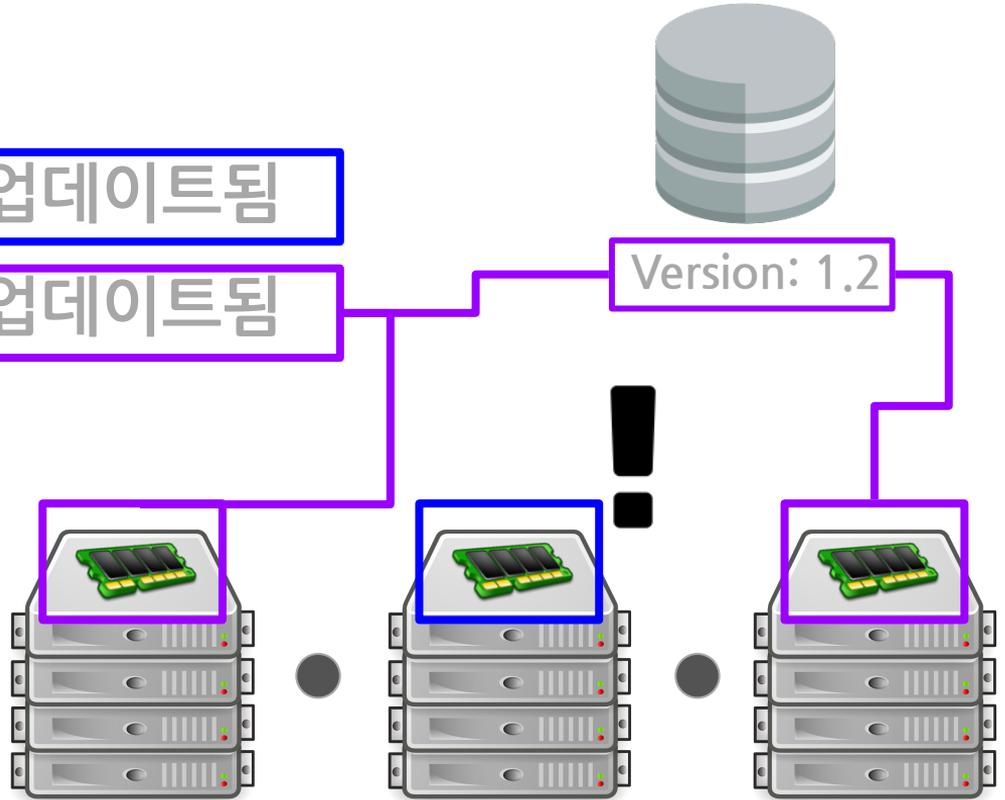
Version: 1.2

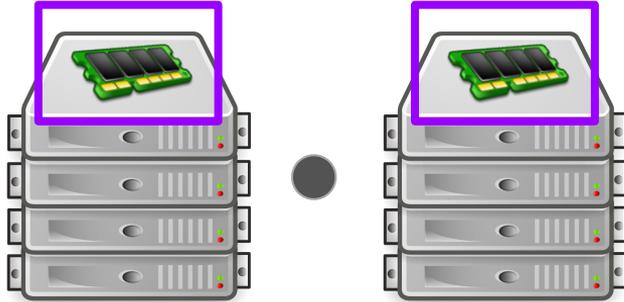


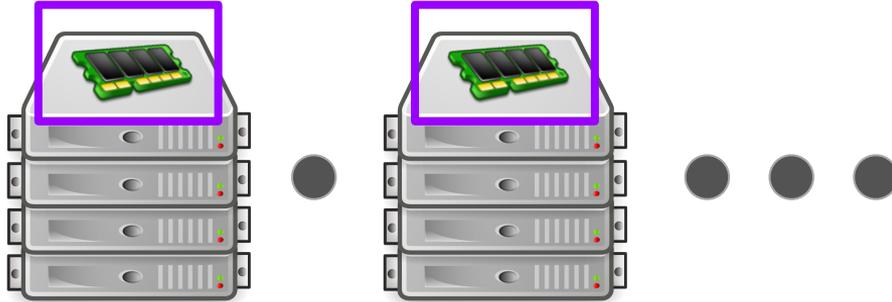
- Version: 1

- Version: 1.1로 데이터가 업데이트됨

- Version: 1.2로 데이터가 업데이트됨







# Eventual Consistency

**동기화가 완료될 때까지 접근을 막지 않고  
Near Real Time Sync(+<2s)를 보장**

근데 이걸 어떻게 하죠? 배치

귀찮네요...

사실 별로 하는거 없지만



오늘은 더 적극적으로 안할거야

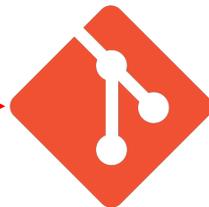


- **application.properties**
- **mysql.properties**



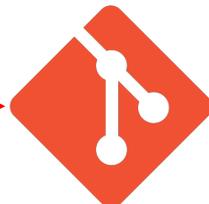
- application.properties

- mysql.properties



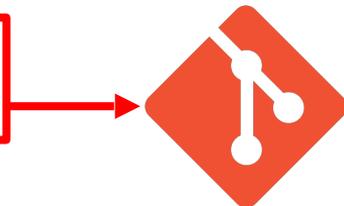
- application.properties

- mysql.properties



- application.properties

- mysql.properties

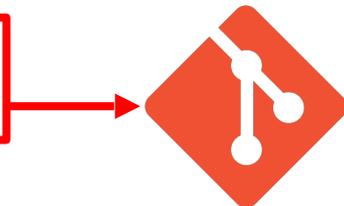


- bootstrap.properties



- application.properties

- mysql.properties



- bootstrap.properties

- config server address



- application.properties

- mysql.properties

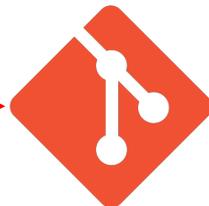
- bootstrap.properties



- application.properties

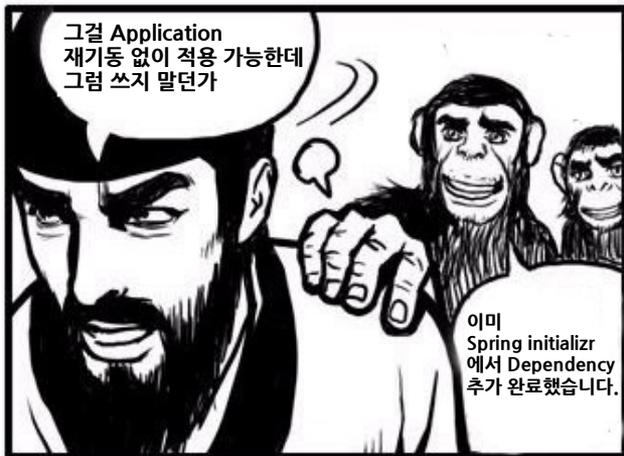
- mysql.properties

- bootstrap.properties



**그래서 특성을 정리하면...**





**배포 없이** 설정을 적용할 수 있습니다.

**2가지** 방법이 있습니다.

```
no usages new *
@RefreshScope
@Configuration
@ConfigurationProperties(prefix = "cell-cache")
public class CellCacheMap {

    no usages
    private Map<String, Map<String, CacheStatusDetailVO>> cellCacheMap;
}
```

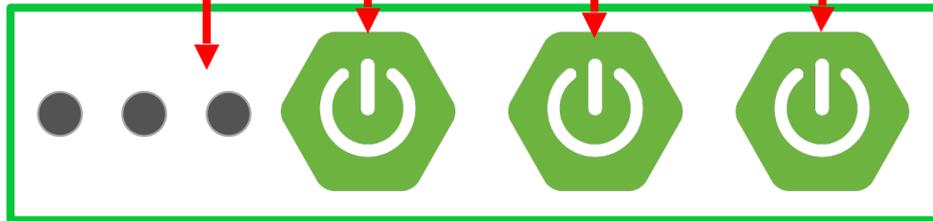
```
@Component
public class CellCacheRefreshEventListener implements ApplicationListener<EnvironmentChangeEvent> {

    @Override
    public void onApplicationEvent(EnvironmentChangeEvent event) {
```

**배포 없이 Post로 /refresh를 호출**

- 1. 설정 파일 수정**
- 2. GIT Push**
- 3. /refresh Post 호출**

**/refresh**



Springcloudbus

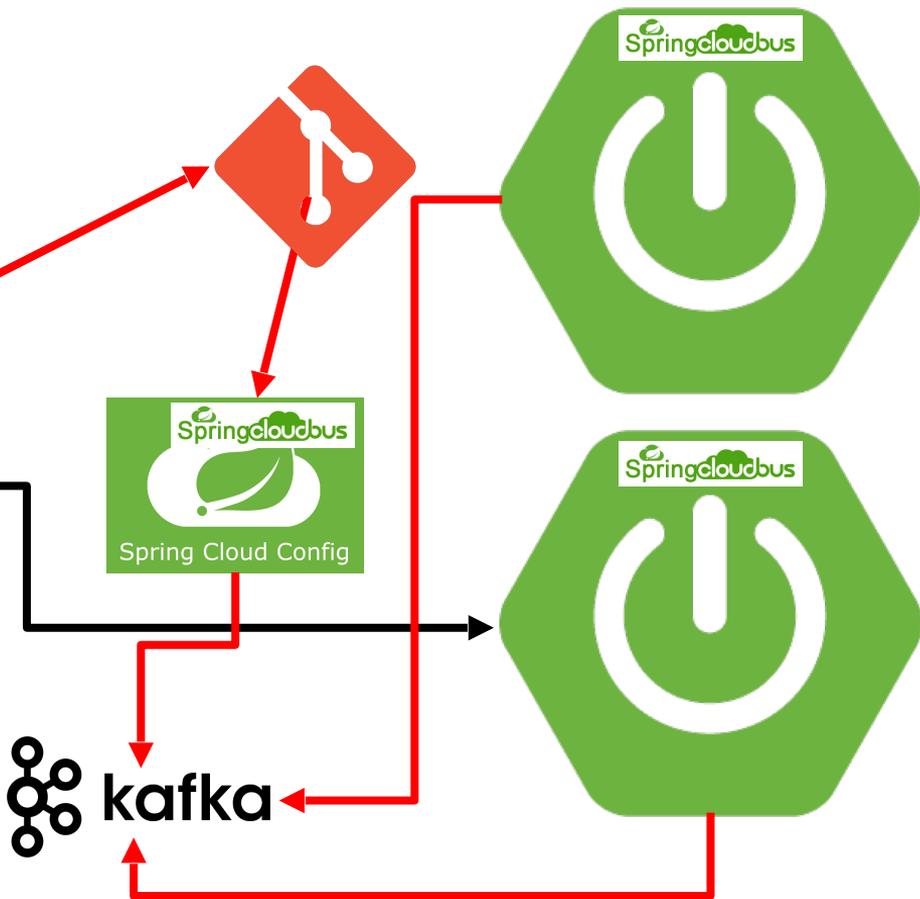




- application.properties

- mysql.properties

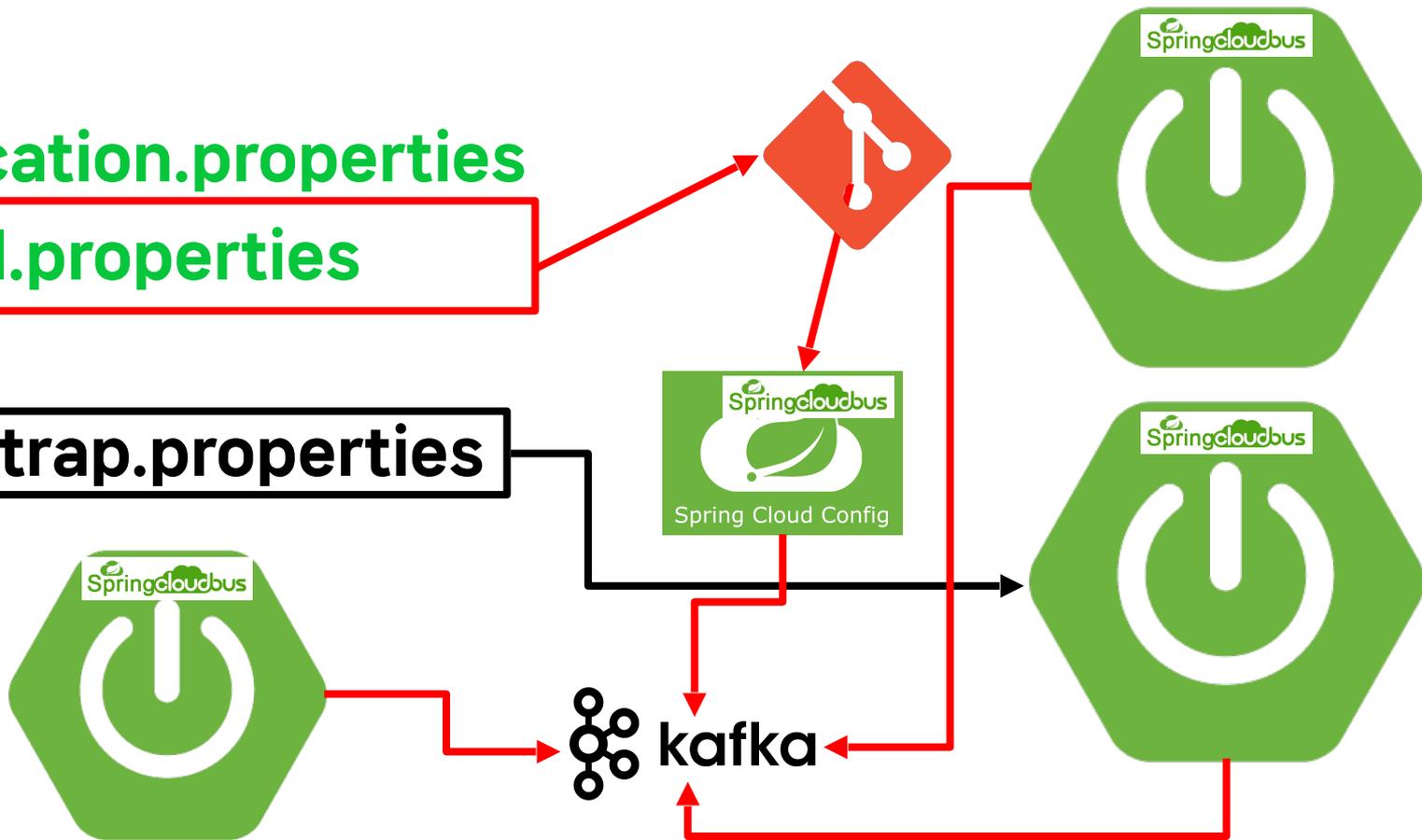
- bootstrap.properties



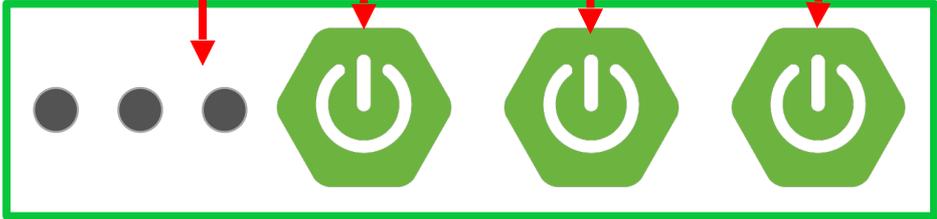
- application.properties

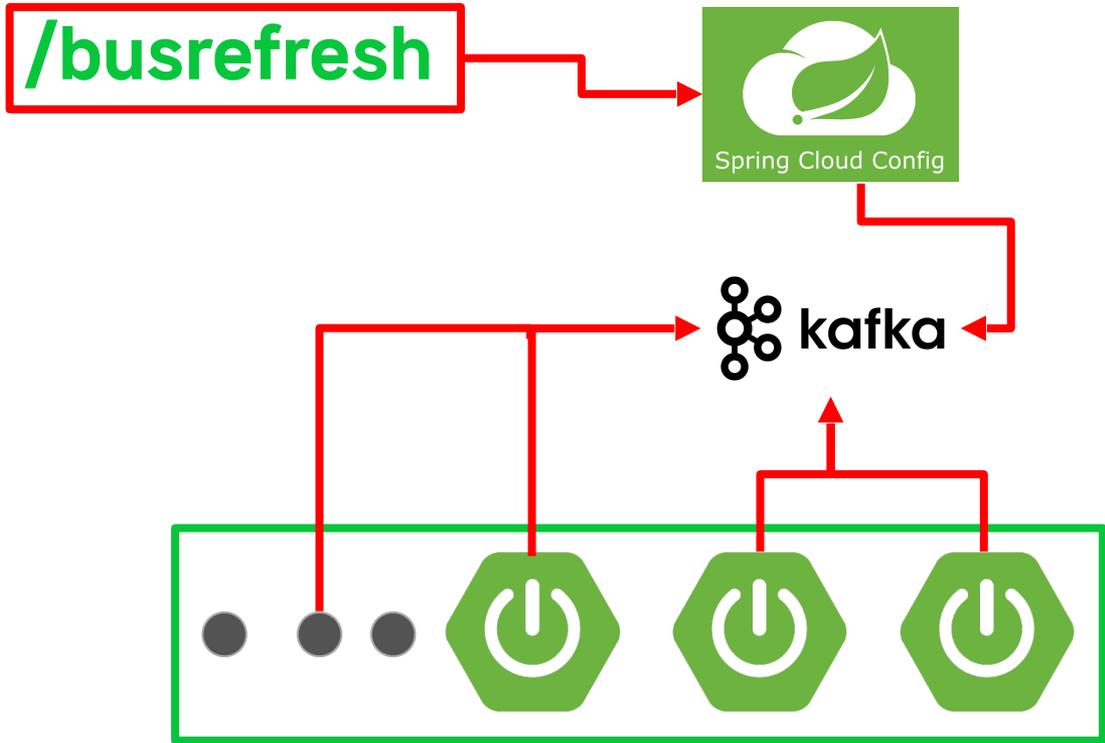
- mysql.properties

- bootstrap.properties



**/refresh**





**그런데 또 문제가 있네요...**

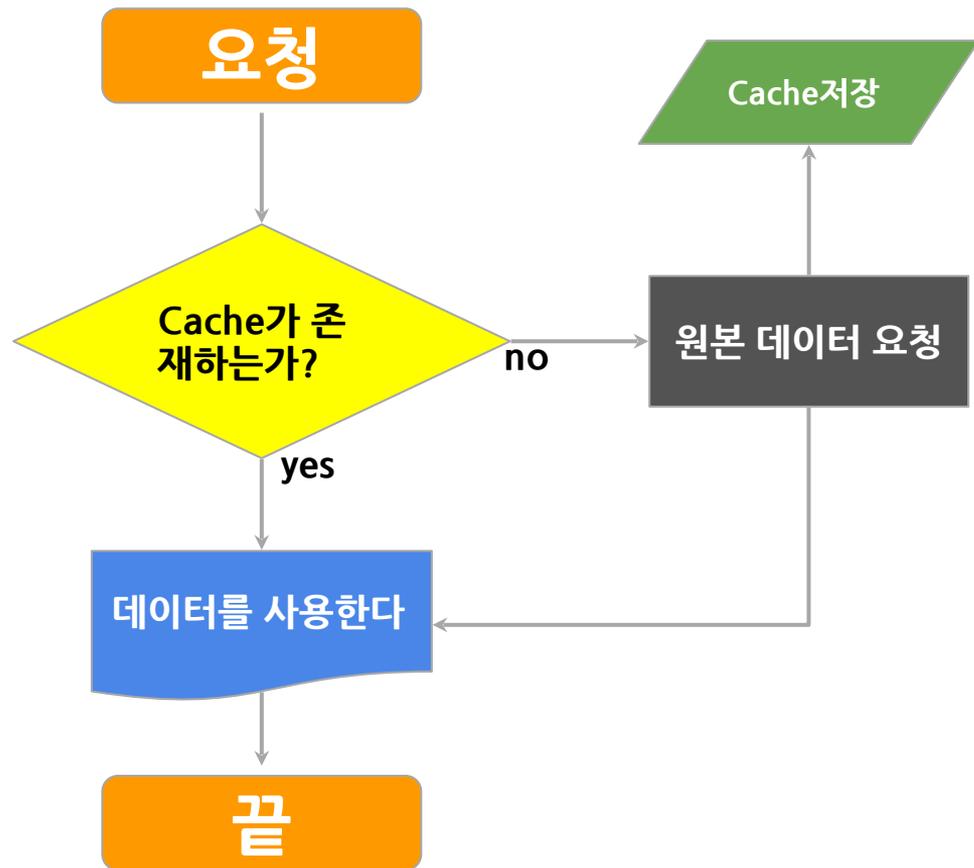
- Git Push 하기 귀찮아
- 자동으로 갱신해 줘

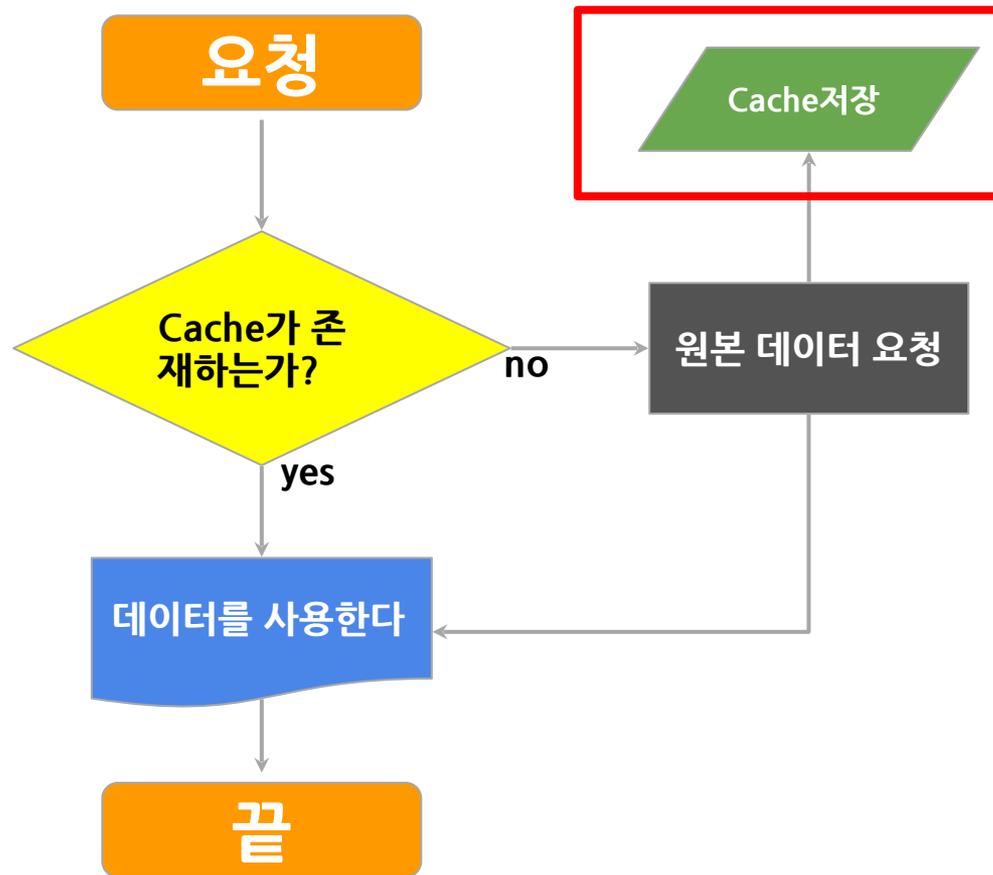
사실 별로 하는거 없지만



오늘은 더 적극적으로 안할거야

**설계를 변경합니다.**





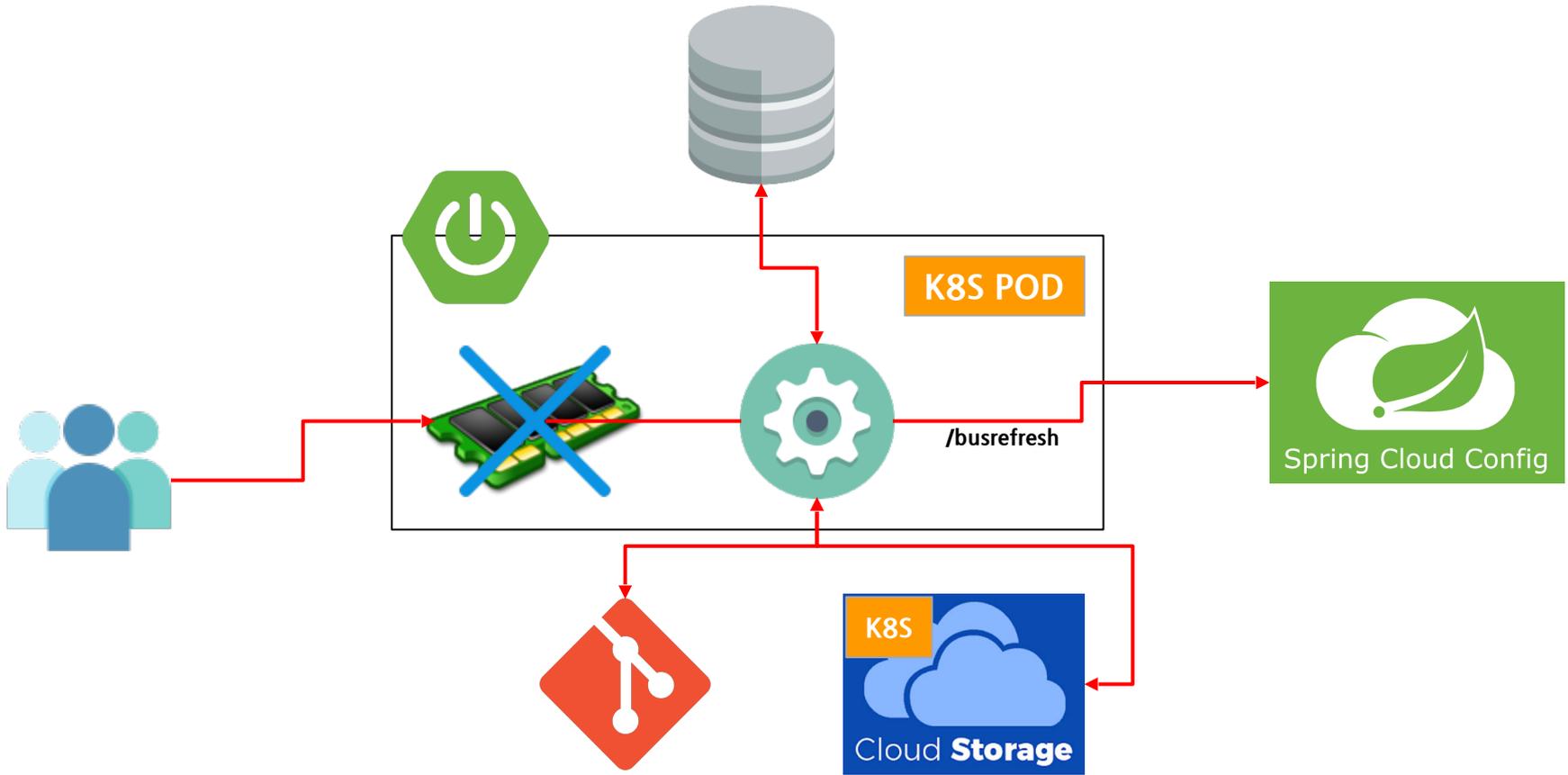
# Config Server는 기본적으로...

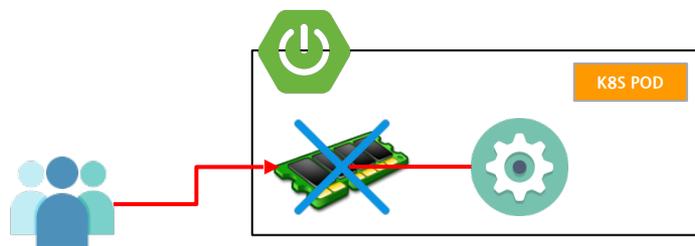
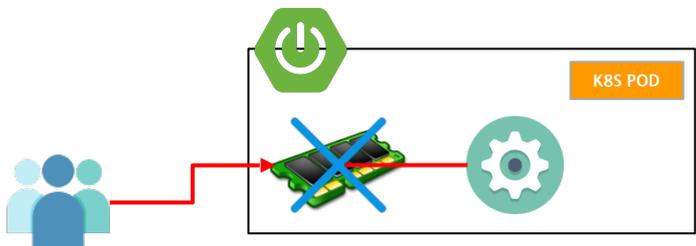
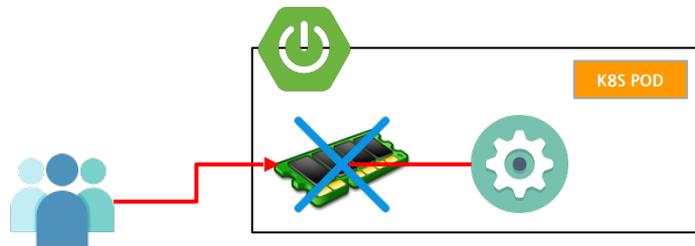
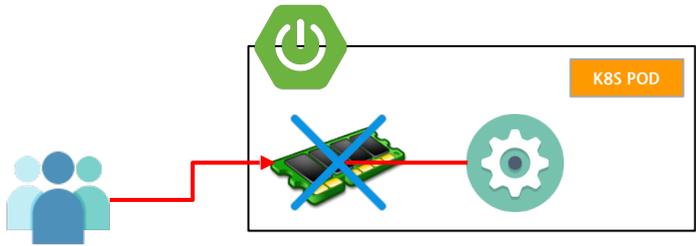
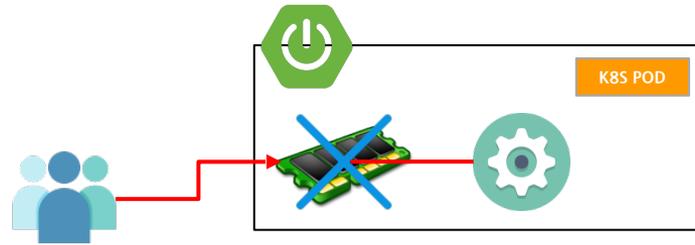
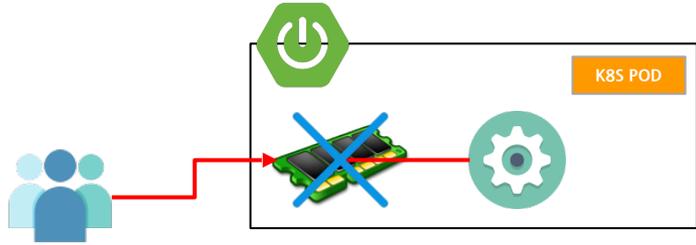
**한 번의 수동 조작으로  
설정을 여러 Application에  
자동으로 적용하고  
배포 없이 수정사항을 적용하는 것**

**그리고 전제 조건은...**

**설정을 GIT에서 Clone 받고  
수정을 하고  
GIT에 다시 PUSH 한 Event가 발생**







- Network Storage I/O 문제
- GIT Conflict
- Cache Sync 복잡도 문제



**또 문제점이 있는데...**

**Properties의 특성을 생각해 보면**

# HashTable

**객체가 들어갈 수 있습니다.**

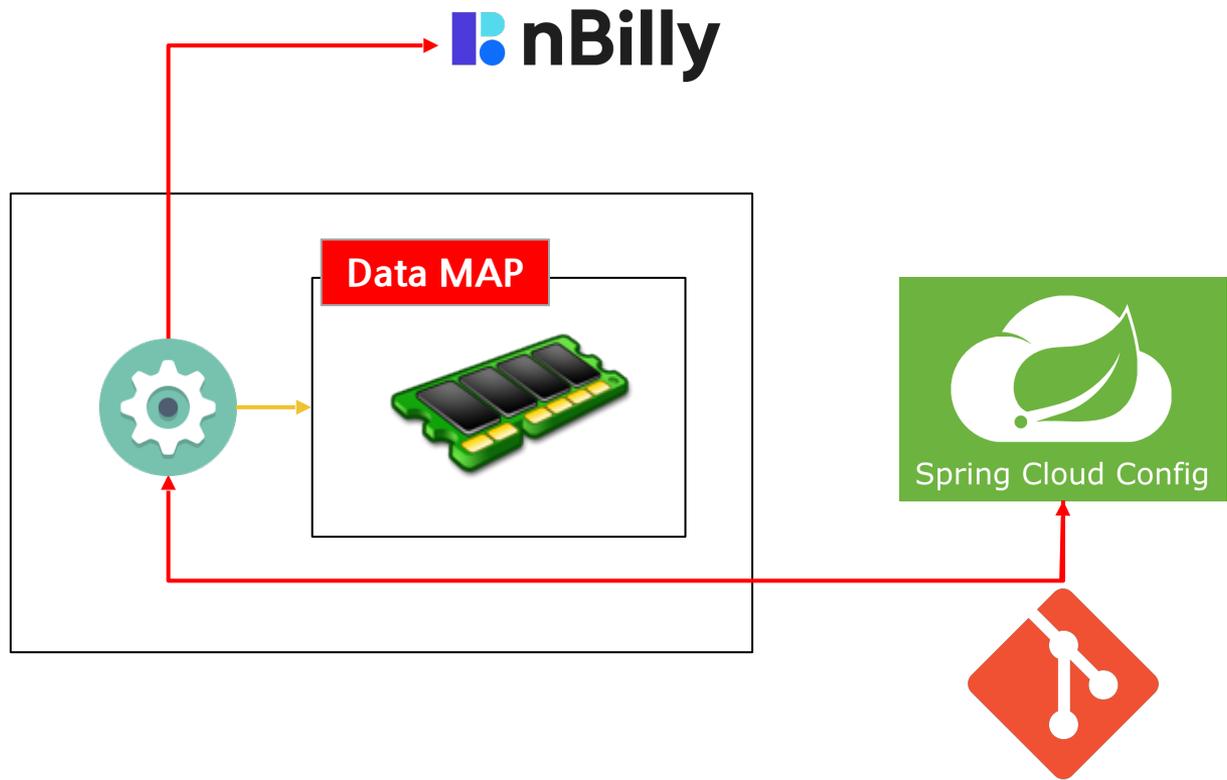
```
cell-cache:
  dataMap:
    samanda: |
      {
        "_id": {
          "$oid": "630de80b5c29b3656091eb17"
        },
        "uuid": "fad73a84-00ed-419d-adf1-ce6c1739f832",
        "userId": "manma971",
        "path": "/manma97/fad73a84-00ed-419d-adf1-ce6c1739f832",
        "title": "Hyun-test",
        "description": "test",
        "representativeImage": {
          "src": "https://beta-cell-platform.ssl.phinf.net/MjAyMjA4MzBmJmJy/MDAxNjYxODU1NzUzNjY3.TN8RW83-aI7WAYfD2p1NDlKuD2V2BTsTisZeBLl5HaIg.QPdNA012ZJ1d2S",
          "type": "JPEG",
          "width": 600,
          "height": 420,
          "thumbnail": {
            "small": "https://beta.ssl.phinf.net/dthumb?src=\"https%3A%2F%2Fbeta-cell-platform.ssl.phinf.net%2FMjAyMjA4MzBmJmJy%2FMDAxNjYxODU1NzUzNjY3.TN8RW",
            "medium": "https://beta.ssl.phinf.net/dthumb?src=\"https%3A%2F%2Fbeta-cell-platform.ssl.phinf.net%2FMjAyMjA4MzBmJmJy%2FMDAxNjYxODU1NzUzNjY3.TN8R",
            "large": "https://beta.ssl.phinf.net/dthumb?src=\"https%3A%2F%2Fbeta-cell-platform.ssl.phinf.net%2FMjAyMjA4MzBmJmJy%2FMDAxNjYxODU1NzUzNjY3.TN8RW"
          }
        },
        "pages": [
          {
            "id": "page_1",
            "containers": [
              {
                "id": "d41afd65-935a-4464-beb2-c09cc9158241",
                "rows": [
                  {
                    "id": "a19afa16-5396-4b45-9c4e-956333d1b08f",
                    "columns": [
```

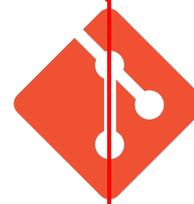
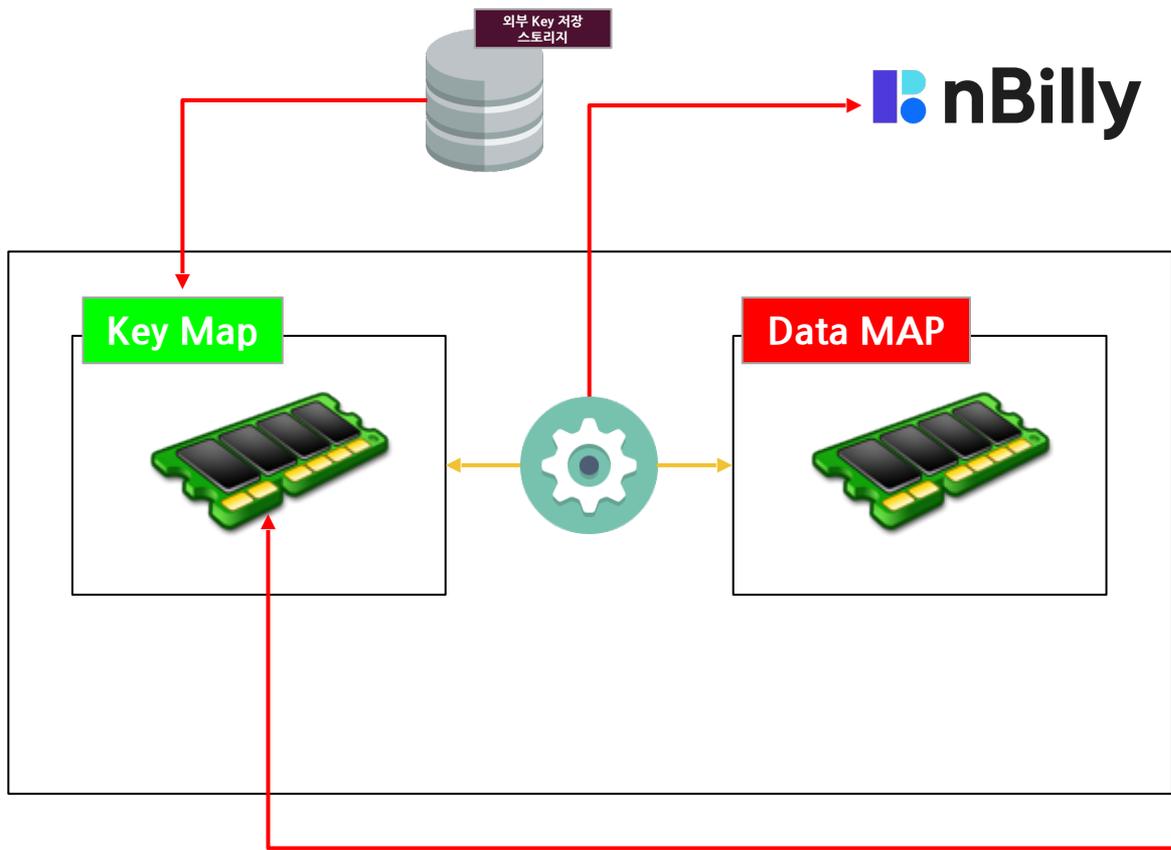
**Cache 대상 객체를 직접 넣지 말고**

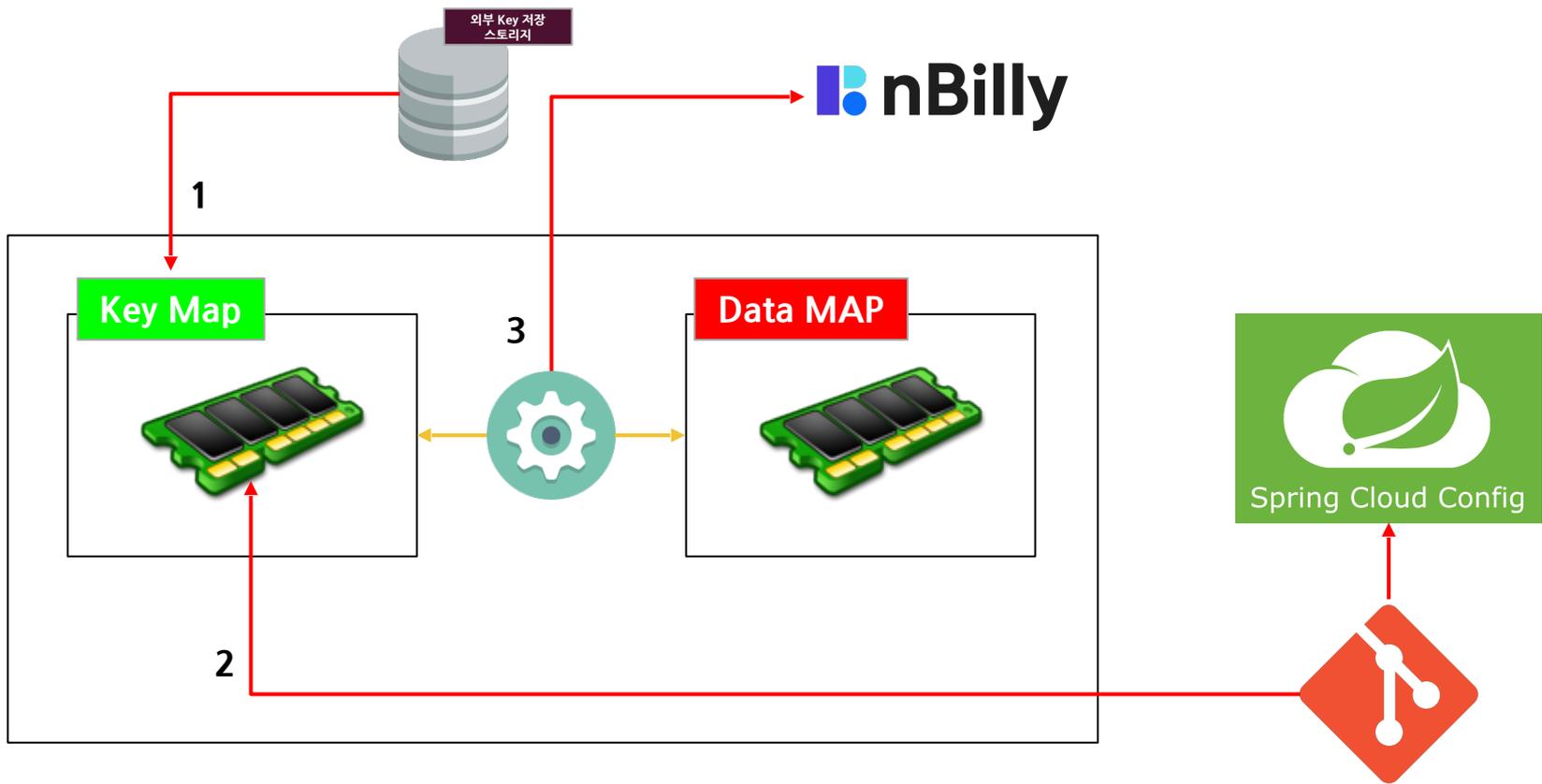
**KeyMap만 따로 관리를 하자!**

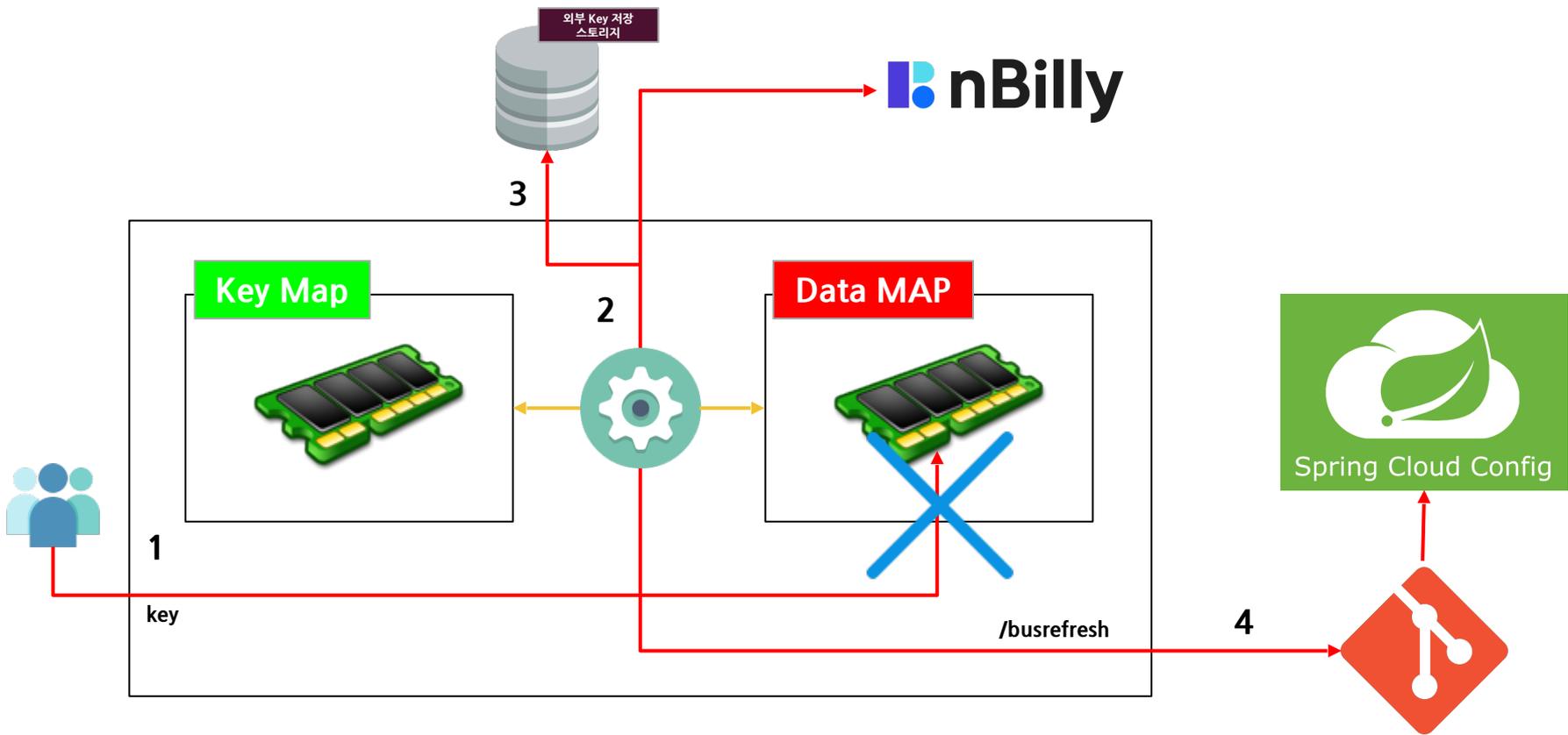
**Dynamic Key + Static Key = Map**

**Config Server는 Trigger로 쓰자**









# 이번 학223123 기는 네이버 필수앱으로 준비 완료!

새학기 생활을 슬기롭게 업그레이드 시켜줄 네이버 필수 앱 총정리  
네이버 필수 앱 총정리 네이버 필수 앱 총정리 앱 총정리



The screenshot shows the Network tab of a browser's developer tools. The 'Headers' sub-tab is active, displaying the request headers for a selected resource. The headers include: 'date: Sun, 09 Apr 2023 05:33:18 GMT', 'expires: 0', 'pragma: no-cache', 'server: nginx', 'vary: Accept-Encoding', 'x-cached: hit' (highlighted with a red box), 'x-content-type-options: nosniff', and 'x-xss-protection: 1; mode=block'. The 'Request Headers' section is partially visible at the bottom.





**운영은  
실전이야  
어딜  
퇴근해**

(테스트나 돌려)

# PERFORMANCE TESTING TOOLS

## (COMPREHENSIVE LOAD TESTING TOOLS LIST)



© [www.SoftwareTestingHelp.com](http://www.SoftwareTestingHelp.com)

# Top Performance Testing Tools

**We have done the research for you. Here is a list of the best web application performance and load testing tools with a detailed comparison:**

1. WebLOAD
2. LoadNinja
3. HeadSpin
4. ReadyAPI Performance
5. LoadView
6. Keysight's Eggplant
7. Apache JMeter
8. LoadRunner
9. Rational Performance Tester
10. NeoLoad
11. LoadComplete
12. WAPT
13. Loadster
14. k6
15. Testing Anywhere
16. Appvance
17. StormForge

# Top Performance Testing Tools

**We have done the research for you. Here is a list of the best web application performance and load testing tools with a detailed comparison:**

1. WebLOAD
2. LoadNinja
3. HeadSpin
4. ReadyAPI Performance
5. LoadView
6. Keysight's Eggplant
7. Apache JMeter
8. LoadRunner
9. Rational Performance Tester
10. NeoLoad
11. LoadComplete
12. WAPT
13. Loadster
14. k6
15. Testing Anywhere
16. Appvance
17. StormForge

# n-Grinder jmeter ab test

# n-Grinder jmeter ab test

**익숙함, 간편한 설정  
쉬운 환경 구성  
스크립트 관리 가능**



**그런데 문제가 좀...**

**혼자 쓰지 못한다.  
Worker 영향이 크다.  
Local 실행이 힘들다.  
구성이 복잡하다.**

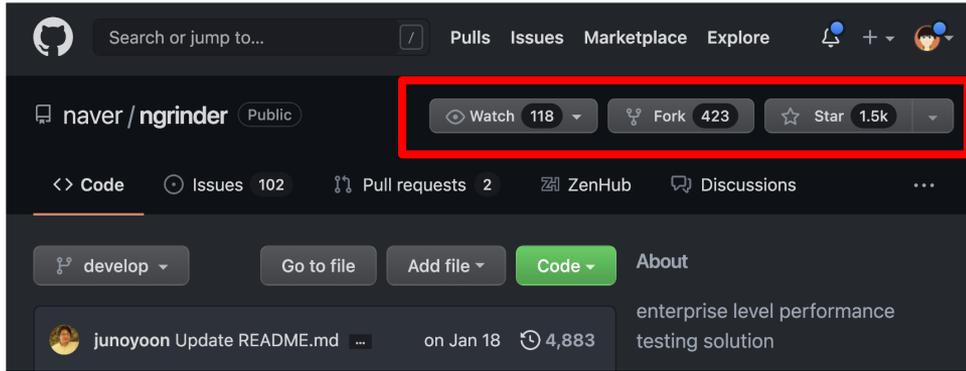




**NAVER**

**Python으로 매우 쉽다.**  
**시나리오 테스트 가능**  
**K8S와 환상의 궁합**  
**이론상 무제한 트래픽 가능**  
**Local에서 손쉬운 실행**





This screenshot shows the GitHub repository page for `naver/ngrinder`. The repository is public and has 118 watchers, 423 forks, and 1.5k stars. The repository is currently on the `develop` branch. The description of the repository is "enterprise level performance testing solution".

Repository: `naver/ngrinder` (Public)

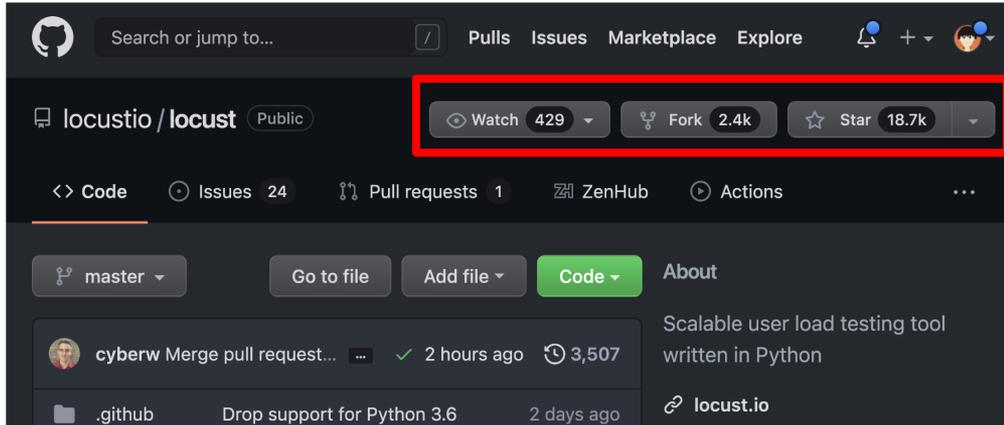
Watch: 118 | Fork: 423 | Star: 1.5k

Code | Issues: 102 | Pull requests: 2 | ZenHub | Discussions

Branch: `develop` | Go to file | Add file | Code | About

junoyoon Update README.md ... on Jan 18 4,883

enterprise level performance testing solution



This screenshot shows the GitHub repository page for `locustio/locust`. The repository is public and has 429 watchers, 2.4k forks, and 18.7k stars. The repository is currently on the `master` branch. The description of the repository is "Scalable user load testing tool written in Python".

Repository: `locustio/locust` (Public)

Watch: 429 | Fork: 2.4k | Star: 18.7k

Code | Issues: 24 | Pull requests: 1 | ZenHub | Actions

Branch: `master` | Go to file | Add file | Code | About

cyberw Merge pull request... ✓ 2 hours ago 3,507

`.github` Drop support for Python 3.6 2 days ago

Scalable user load testing tool written in Python

locust.io

**그리고 진짜 쉬워요...**

```
from locust import task, HttpUser, between, tag
from random import *
```

```
class BasicTester(HttpUser):
```

```
    channel_id = '166487678472876'
```

```
    def on_start(self):
```

```
        super(BasicTester, self).on_start()
```

```
        self.client.headers['Content-Type'] = 'application/json'
```

```
        self.client.headers['X-CHANNEL-ID'] = self.channel_id
```

```
    @task
```

```
    @tag("configuration-fetch")
```

```
    def configuration_fetch(self):
```

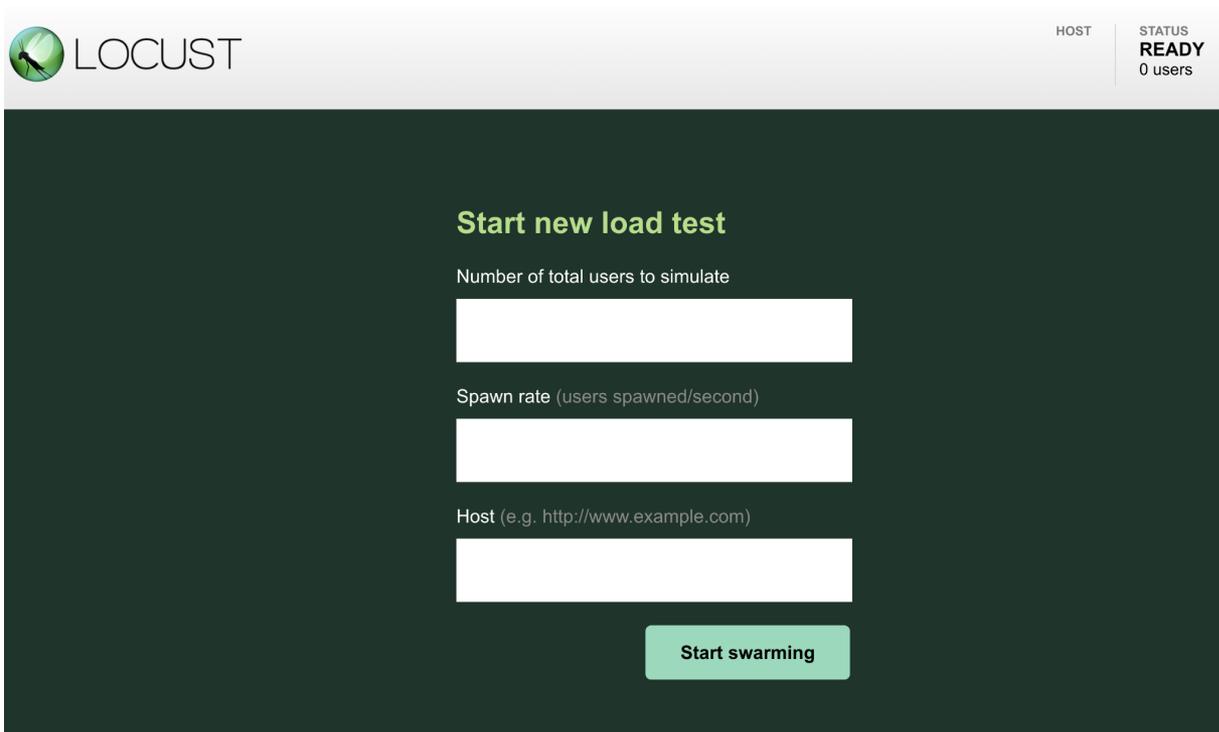
```
        self.client.get("/editor/configurations-fetch")
```

```
# locust -f {스크립트 파일명}.py --tags=configuration-fetch
```

```
->
```

```
[2022-04-28 22:13:46,434] leekyoungils-iMac.local/INFO/locust.main: Starting web interface at http://0.0.0.0:8089 (accepting connections from all network interfaces)
```

```
[2022-04-28 22:13:46,440] leekyoungils-iMac.local/INFO/locust.main: Starting Locust 1.4.3
```



The screenshot shows the Locust web interface. At the top left is the Locust logo, a green circle with a black locust. To its right is the word "LOCUST" in a large, white, sans-serif font. In the top right corner, there is a status bar with "HOST" on the left and "STATUS" on the right. Under "STATUS", it says "READY" in bold and "0 users" below it. The main content area has a dark green background. It features a heading "Start new load test" in a light green font. Below this heading are three white input fields: "Number of total users to simulate", "Spawn rate (users spawned/second)", and "Host (e.g. http://www.example.com)". At the bottom right of this area is a light green button with the text "Start swarming".



HOST  
https://[redacted]  
/homebuilder\_node\_fe/homebuilder-node-api

STATUS  
**RUNNING**  
20 users  
[Edit](#)

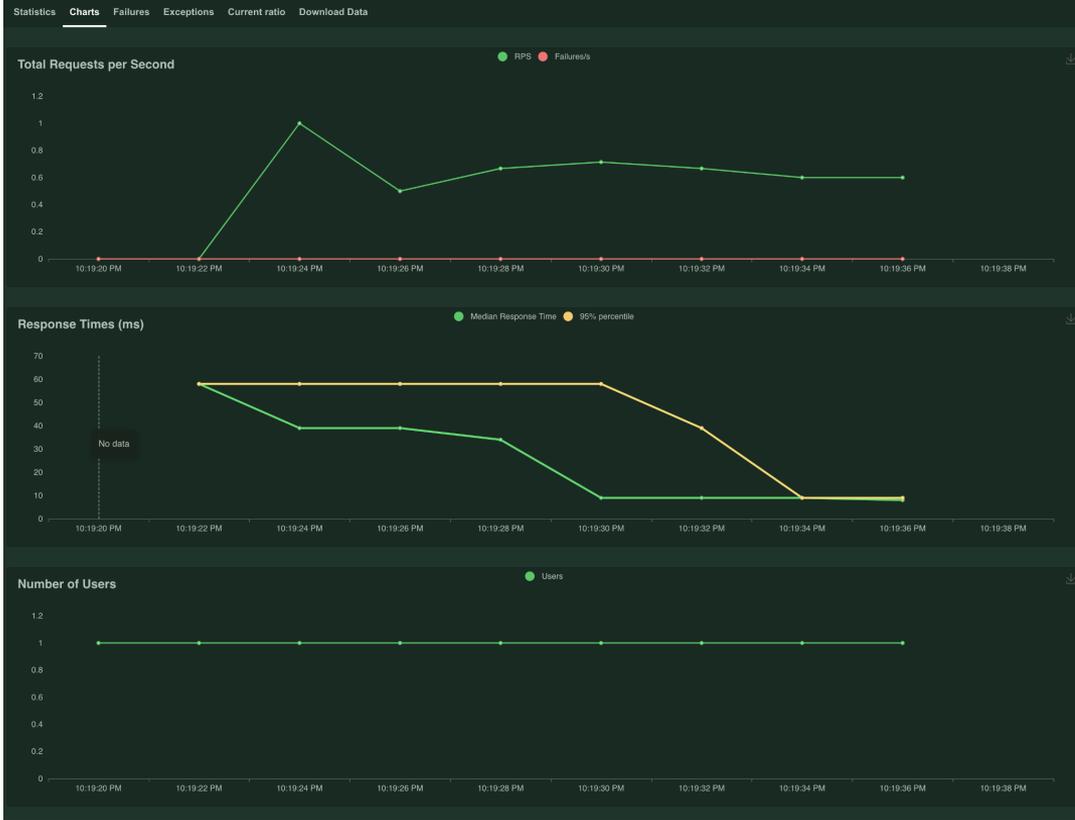
RPS  
**8.7**

FAILURES  
**0%**



[Statistics](#) [Charts](#) [Failures](#) [Exceptions](#) [Download Data](#)

Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/homebuilder_node_fe/homebuilder-node-api/editor/configurations-fetch	144	0	11	34	15	8	75	180417	8.7	0
<b>Aggregated</b>		<b>144</b>	<b>0</b>	<b>11</b>	<b>34</b>	<b>15</b>	<b>8</b>	<b>75</b>	<b>180417</b>	<b>8.7</b>	<b>0</b>



Instance **samanda-dev-locust**



### 인스턴스 서비스

Service name	Cluster IP	Load balancer info
<a href="#">samanda-dev-locust</a>	[REDACTED]	<a href="#">DNS</a> <a href="#">port0</a> <a href="#">port1</a> <a href="#">port2</a> <a href="#">Private VIP</a>

### Installations

[dev-samanda-locust-220428-221438](#) +Deployed Weight 100 Total Pod 2  >\_logs

Pods	Deployment	Monitoring	Age	Replicas
<input type="radio"/>	<a href="#">samanda-dev-locust-dev-samanda-locust-220428-221438-master</a>		4 hours	1
<input type="radio"/>	<a href="#">samanda-dev-locust-dev-samanda-locust-220428-221438-worker</a>		4 hours	1



HOST

mebuilder-node-api

STATUS  
**STOPPED**  
[New test](#)

WORKERS  
**3**

RPS  
**45.5**

FAILURES  
**0%**

[Statistics](#) [Charts](#) [Failures](#) [Exceptions](#) [Current ratio](#) [Download Data](#) [Workers](#)

Worker

samanda-dev-locust-dev-samanda-locust-220429-024350-worker85mzv\_744d29b73b7d44a4ad4ae4fce2f0b07  
samanda-dev-locust-dev-samanda-locust-220429-024350-workerldgzj\_a3e21e0d0b754a0489140f98b32fb438  
samanda-dev-locust-dev-samanda-locust-220429-024350-workerpgddz\_747bb952ea3c4dd6b30ce5888ecda508

State

# users

CPU usage

Memory usage

ready

0

0%

42.59 MB

ready

0

0.2%

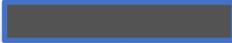
154.79 MB

ready

0

0%

41.16 MB



STATUS	WORKERS	RPS	FAILURES
<b>RUNNING</b>	<b>50</b>	<b>6042.8</b>	<b>0%</b>
7000 users			

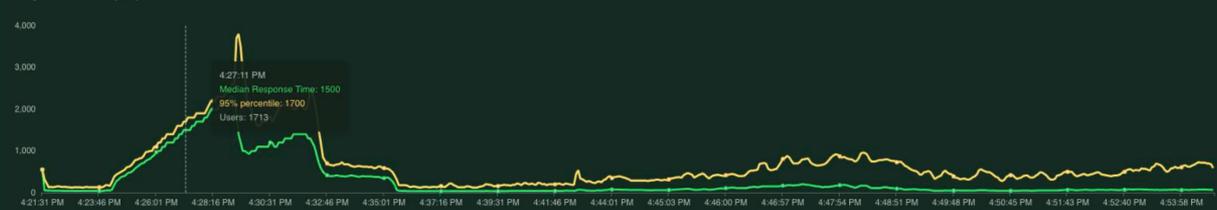


Statistics Charts Failures Exceptions Current ratio Download Data Workers

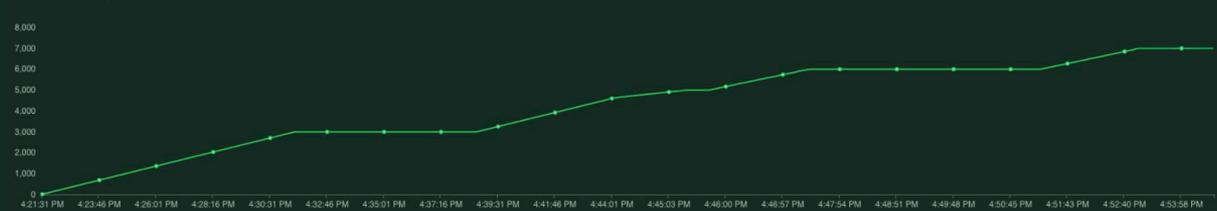
### Total Requests per Second



### Response Times (ms)



### Number of Users



이경일

2022년 12월 12일

지금 만드는 플랫폼 성능 테스트 진행중...

유저 7천에 5900 RPS넘어가고 있음  
테스트는 6천 RPS 찍고 넘어갈때 중지 했는데.

거의 30분동안 부하를 주면서 돌렸는데 오류 한번 안  
났다.  
기분이 좋군.

남자답게 계속 pod 늘리며 1만 RPS 찍어보고 싶었는데  
인프라 담당자에게 연락 올까봐 여기까지만 하고 중  
지 간단히 보기

수정

심천보, 오종인 및 외 30명

좋아요 댓글 달기 공유하기

관련성 높은 순

김현민  
원래 인프라 담당자 연락 올때까지 테  
스트 하는거 아닌가요  
웃겨요 댓글 달기 16주  
↳ 이경일님이 답글을 남겼... 댓글 1개

조희범  
에칭에 금요일에 로그분석 솔루션 들  
리다 부하테스트하다가 보안팀에서 연  
락받음 ddos 공격 하지 말라고 ㅋㅋㅋ  
좋아요 댓글 달기 16주

Bruce Daeyoung Kim  
서버개발의 골쟁포인트 성능테스트군  
여  
좋아요 댓글 달기 16주

전육전  
굿굿!  
좋아요 댓글 달기 16주

댓글을 입력하세요...

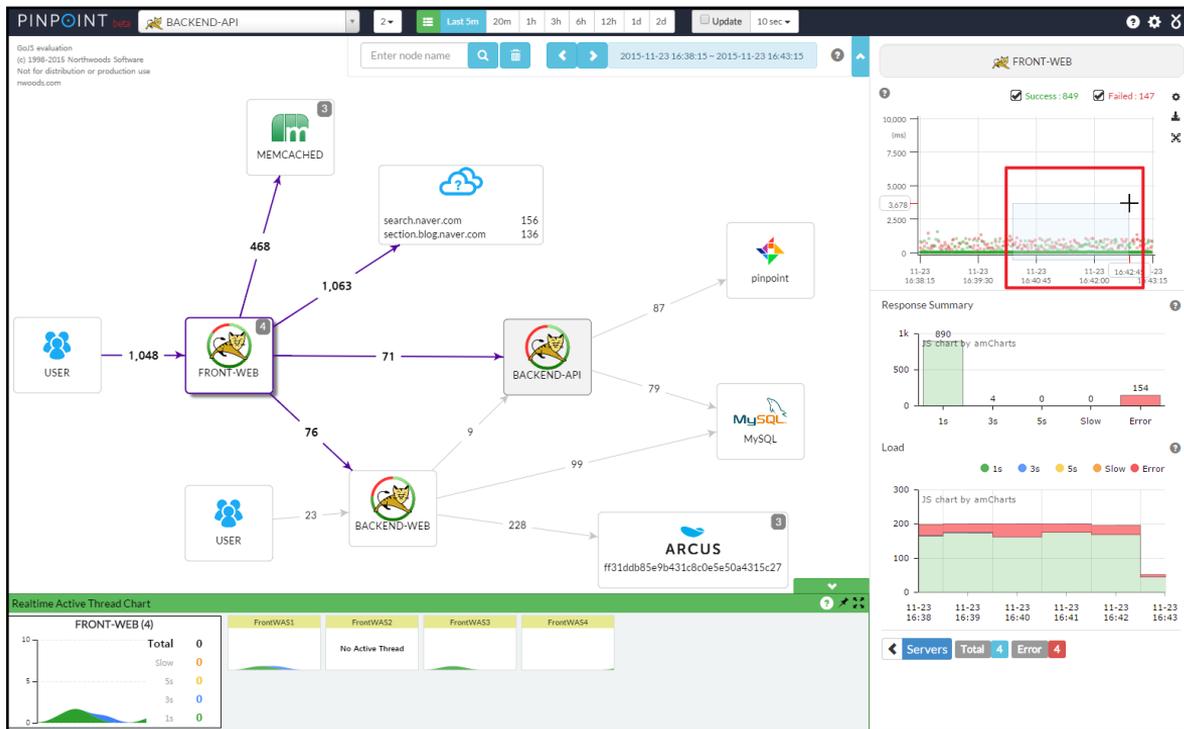
**한 가지 더 필수 툴이 있습니다.**

**APM이 필요해요.**



# Application Performance Management

# - PINPOINT - <https://github.com/pinpoint-apm/pinpoint>



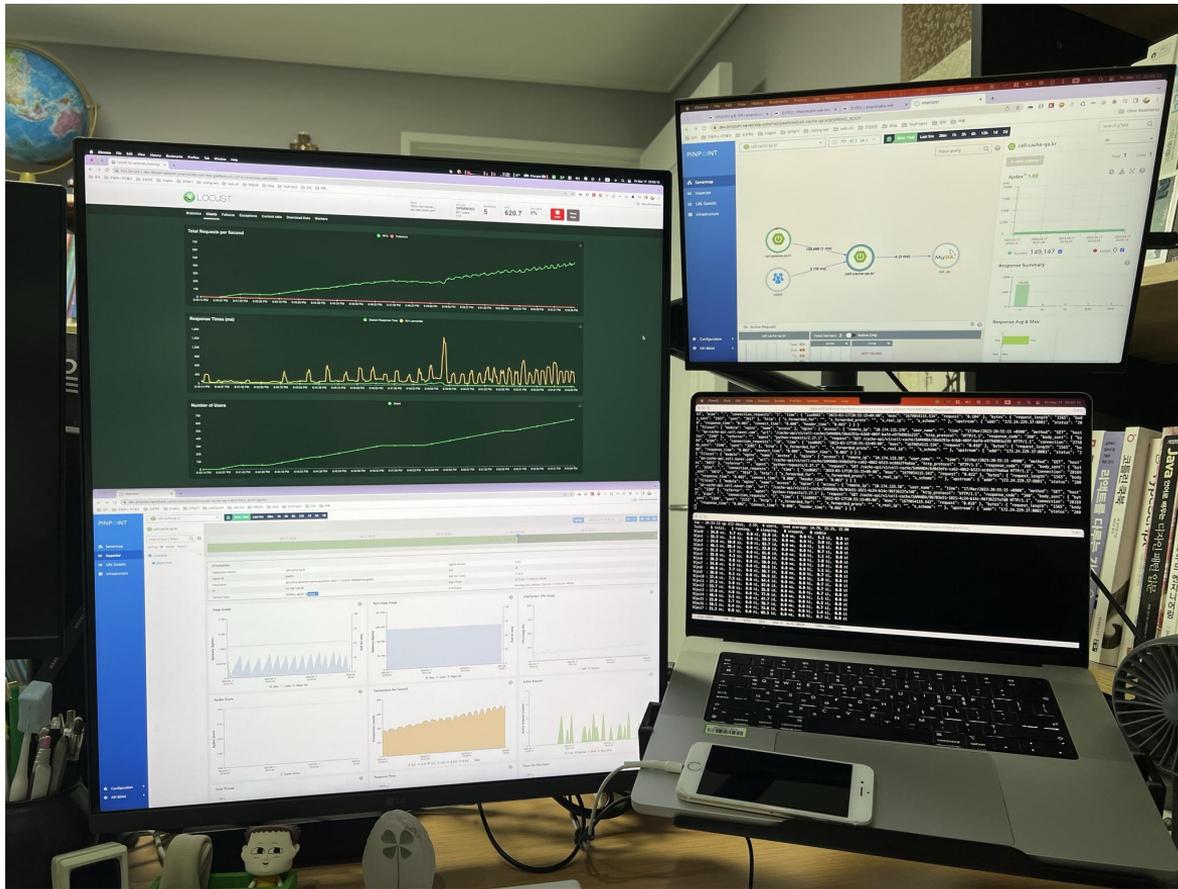
궁금한가?

더 이상의  
자세한 설명은  
생략한다.



설명은  
GPT님이  
하신다

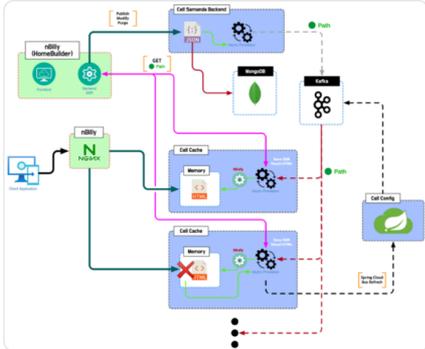
# 저의 성능 테스트에 임하는 자세



**들어주셔서 감사합니다.**

**운영에 적용 후 성과를 보고하는 자리가  
있었는데요.**

**우상훈 (ETECH / Smart Studio) 4:05 PM**  
여기서 어딜 강조할까요?



**이경일 (ETECH / Cell TF, HB) 4:06 PM**  
모든것이 자동으로 이루어지고  
이론상 무제한의 트래픽을 감당 할 수 있다는것

**이경일 (ETECH / Cell TF, HB) 4:06 PM**  
pod 하나하나가 캐시가 되고 그 pod들의 캐시 데이터 싱크가 near real time 으로 보장됨

**우상훈 (ETECH / Smart Studio) 4:08 PM**  
이거 기존의 네이버 방식과 뭐가 달라서 좋아!

**우상훈 (ETECH / Smart Studio) 4:08 PM**  
이런 건 없으?

**우상훈 (ETECH / Smart Studio) 4:08 PM**  
(미안 내가 BE를 잘 모르니 ㅠㅠ)

**우상훈 (ETECH / Smart Studio) 4:10 PM**  
이거 다른 BE들이 들으면 와우 할 포인트 맞췄?

**이경일 (ETECH / Cell TF, HB) 4:13 PM**  
지금 제가 만든 캐시가 일종의 Local Cache 인데 문제가 서버가 늘어나면 싱크하기가 힘들어서 성능은 좋지만 사용하기 어렵다는건데 (새로고침 할때마다 다르게 캐시된 데이터가 나올 수 있음)

제가 그린 아키텍처의 의해서 near real time으로 싱크를 보장합니다.

**이경일 (ETECH / Cell TF, HB) 4:13 PM**  
K8S가 수평확장이 될때도 최신의 Cache데이터를 들고 쓰는것을 보장하구요

**이경일 (ETECH / Cell TF, HB) 4:16 PM**  
정리하면

1. 외부 (혹은전사) 캐시 인프라에 의한 장애로부터 격리가능
2. POD가 모두 Cache 서버가 되기 때문에 수평확장을 하는 K8S(N2C) 위에서 무제한 트래픽 처리가능
3. 모두 자동으로 POD 간 Cache Data Sync가 이루어짐

**이경일 (ETECH / Cell TF, HB) 4:23 PM**  
그리고 우와~ 할 포인트는 맞아요. (미친놈 별짓을 다하네 정도로)

👍 1 🔄 1 🗨️

**이경일 (ETECH / Cell TF, HB) 4:26 PM**  
그리고 의존하는 인프라가 없기에 글로벌 나가도 당연 유리하죠.

**미친 짓을 계속하다 보면 한두 개는  
걸리더라구요.**



**성능 테스트는 왜 해야 할까요?**

**지옥에서 온 CTO님이 하신 말씀이 있죠.**



Gyuwon Yi

3일 · 🌐



“누구나 그럴싸한 아키텍처를 가지고 있다. 높은 트래픽에 쳐맞기 전까지는.”



Gyuwon Yi

2019년 5월 1일 · 🌐

“누구나 그럴싸한 아키텍처를 가지고 있다. 높은 트래픽에 쳐맞기 전까지는.”

트래픽을 초단위가 아니라 분단위로 세는 시장은 아직 많다. 그건 누군가의 잘못이 아니다. 그냥 그런거다. 하지만 안락한 연구개발 센터에만 머물며 개발자 테스트나 데모 정도만 경험하고서 치열한 전장에서 높은 트래픽과 싸우는 사람과 시스템을 과소평가 하진 말았으면 한다.

동일한 기능 요구사항에도 서비스 트래픽이 높아지면 수많은 가정이 뒤틀린다. 우아하게 동작하던 기능들은 숨겨놓은 오류들을 쏟아낸다. 개발자는 밑바닥부터 다시 쌓아올리고 싶은 맘이 굴뚝같다. 하지만 서비스 운영은 그런 사치 따위 허용하지 않는다. 달리는 열차의 사각 바퀴를 오각 바퀴로 갈아 끼우는 임무를 수행해야 한다. 온실에서 꽃피운 아키텍처는 이런 환경에서 몇 초도 견디지 못 할 것이다.



Gyuwon Yi

3일 · 🌐



“누구나 그럴싸한 아키텍처를 가지고 있다. 높은 트래픽에 쳐맞기 전까지는.”



Gyuwon Yi

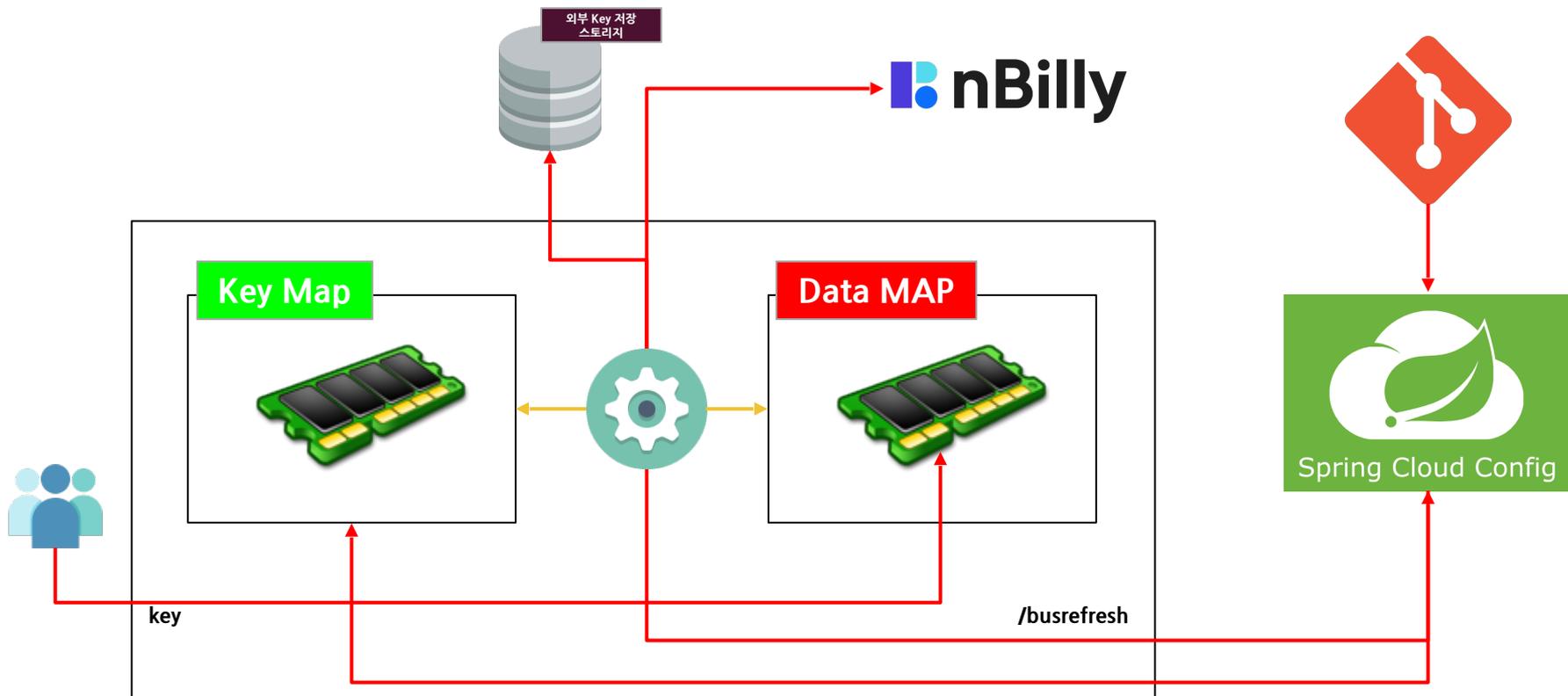
2019년 5월 1일 · 🌐

“누구나 그럴싸한 아키텍처를 가지고 있다. 높은 트래픽에 쳐맞기 전까지는.”

트래픽을 초단위가 아니라 분단위로 세는 시장은 아직 많다. 그건 누군가의 잘못이 아니다. 그냥 그런거다. 하지만 안락한 연구개발 센터에만 머물며 개발자 테스트나 데모 정도만 경험하고서 치열한 전장에서 높은 트래픽과 싸우는 사람과 시스템을 과소평가 하진 말았으면 한다.

동일한 기능 요구사항에도 서비스 트래픽이 높아지면 수많은 가정이 뒤틀린다. 우아하게 동작하던 기능들은 숨겨놓은 오류들을 쏟아낸다. 개발자는 밑바닥부터 다시 쌓아올리고 싶은 맘이 굴뚝같다. 하지만 서비스 운영은 그런 사치 따위 허용하지 않는다. 달리는 열차의 사각 바퀴를 오각 바퀴로 갈아 끼우는 임무를 수행해야 한다. 온실에서 꽃피운 아키텍처는 이런 환경에서 몇 초도 견디지 못 할 것이다.







	CPU	MEM	
	100.00%	79.73%	

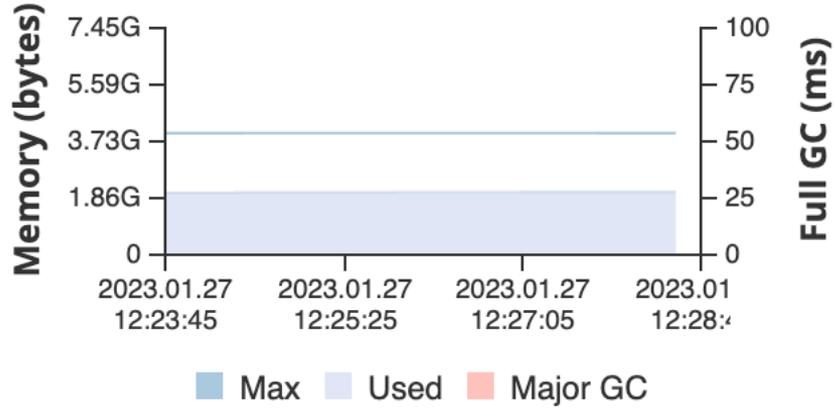


**개발자란 어떤 직업이라고 생각하세요?**

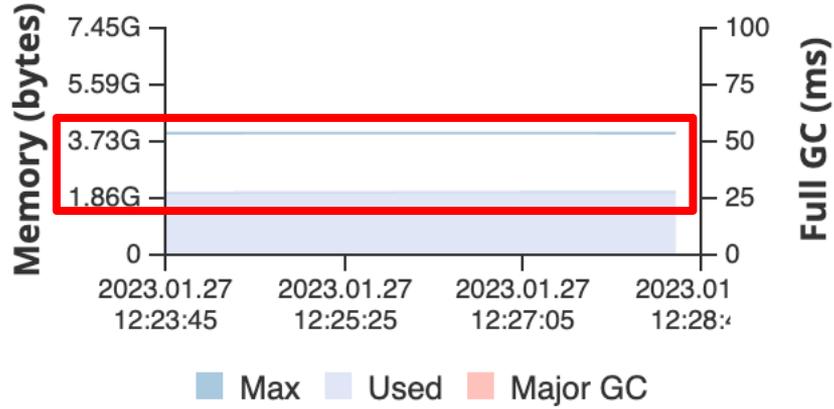


**대용량 트래픽 대응...**

## Heap Usage



## Heap Usage





### Cell Platform

It is a platform that provides infrastructure and service API that can be easily developed regardless of environment when developing services.

 Documentation

 GitHub



### ELF framework

Editor framework for building collaborative editing applications

 Documentation

 GitHub

### Moment

It is a short form video editor that allows you to edit photos and videos easily and simply.

 Documentation

 GitHub

### SmartEditor

Smart Editor provides users with high editing freedom and content completion. It is a creative tool that satisfies you at the same time.

 Documentation

 GitHub

### nBilly

NCDP(No Code Development Platform) that uses IN/OUT APIs to make websites look like Lego blocks. Users can quickly and easily create a website at a low cost.

 Documentation

 GitHub

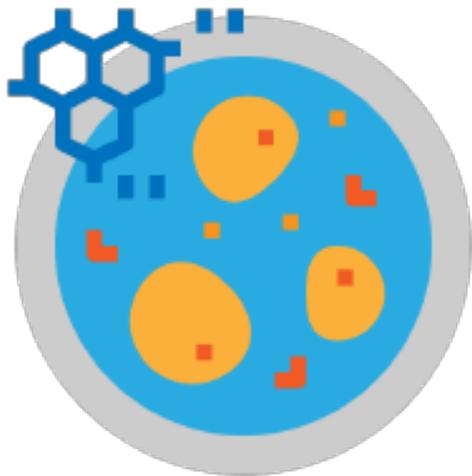


### Yorkie

Document store for building collaborative editing applications

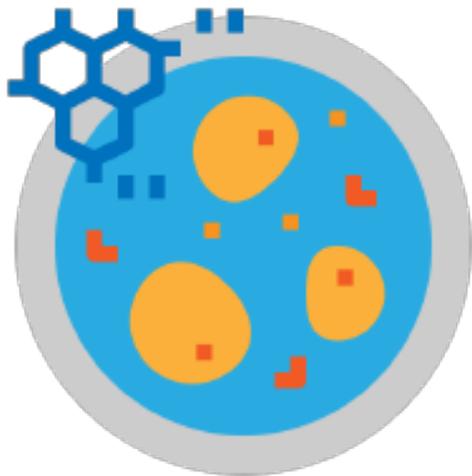
 Documentation

 GitHub



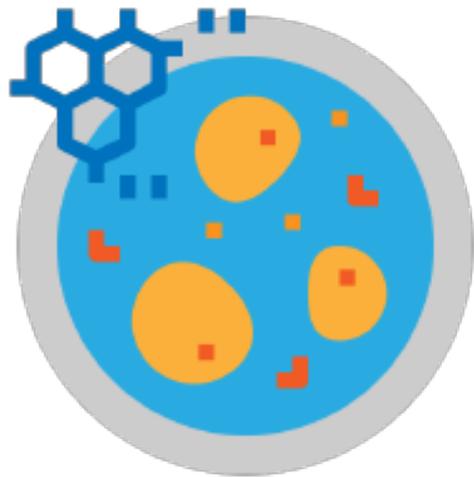
# Cell Platform

SmartStudio Common backend Platform

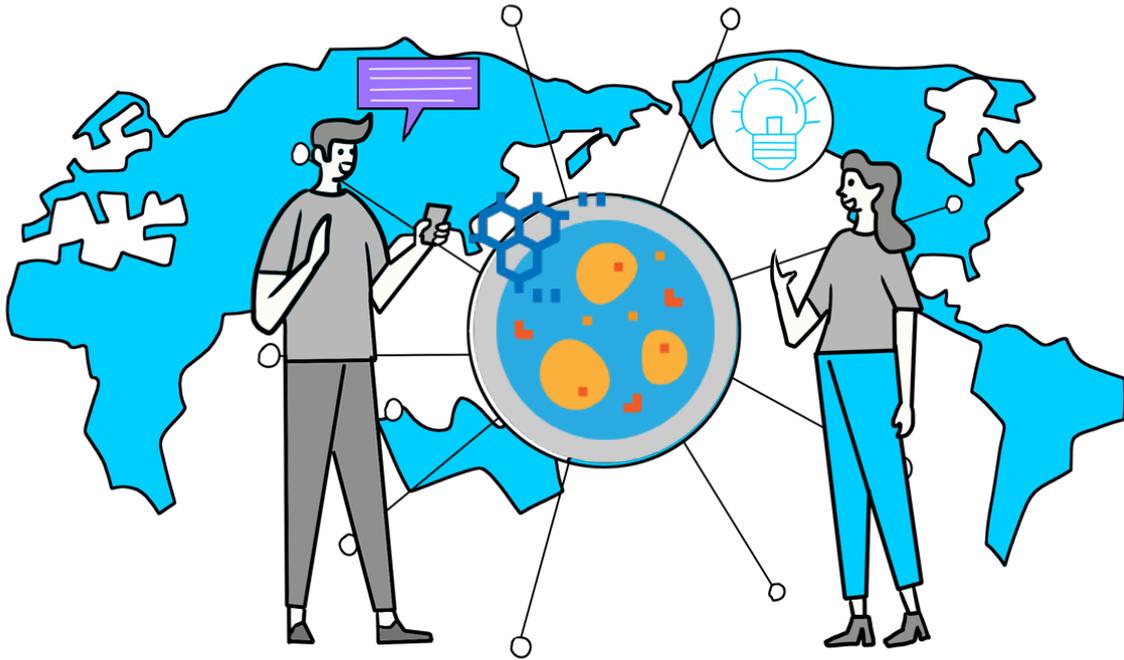


# Cell Platform

SmartStudio Common backend Platform



**조직에서 서비스를 개발할 때  
특정 지역/인프라에 구애받지 않고  
적은 인원으로 빠르게 출시할 수 있는  
통합 플랫폼을 개발한다.**



저와 같이 **Cell Platform**을 만들어 보고  
싶지 않으신가요?

# 기다리고 있겠습니다.

- [leekyoungil@gmail.com](mailto:leekyoungil@gmail.com)
- [kyoungil.lee@navercorp.com](mailto:kyoungil.lee@navercorp.com)



**감사합니다.**