Web 서비스에 있어서 서버 구성과 그 목적

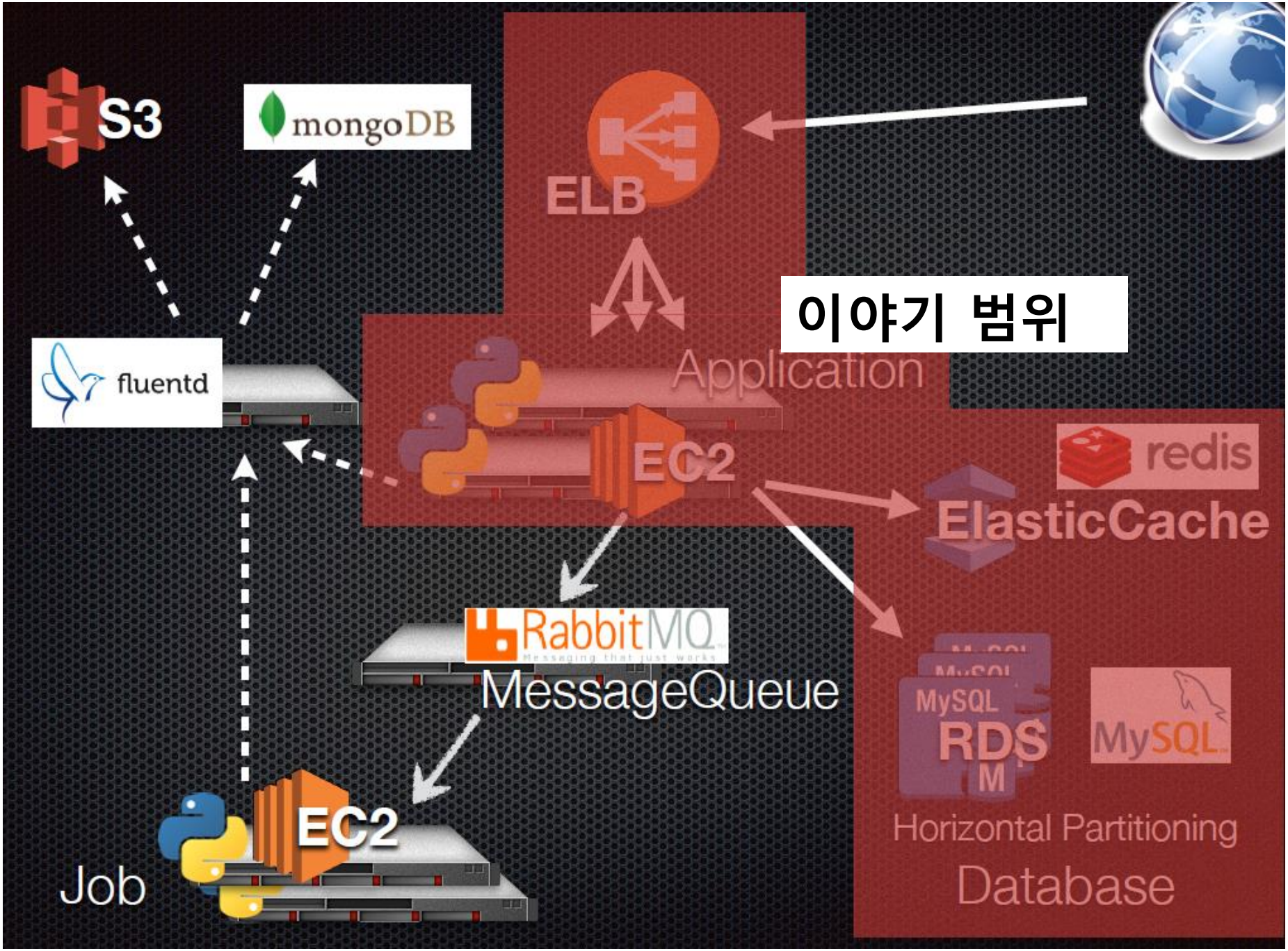
최흥배
https://github.com/jacking75/choiHeungbae

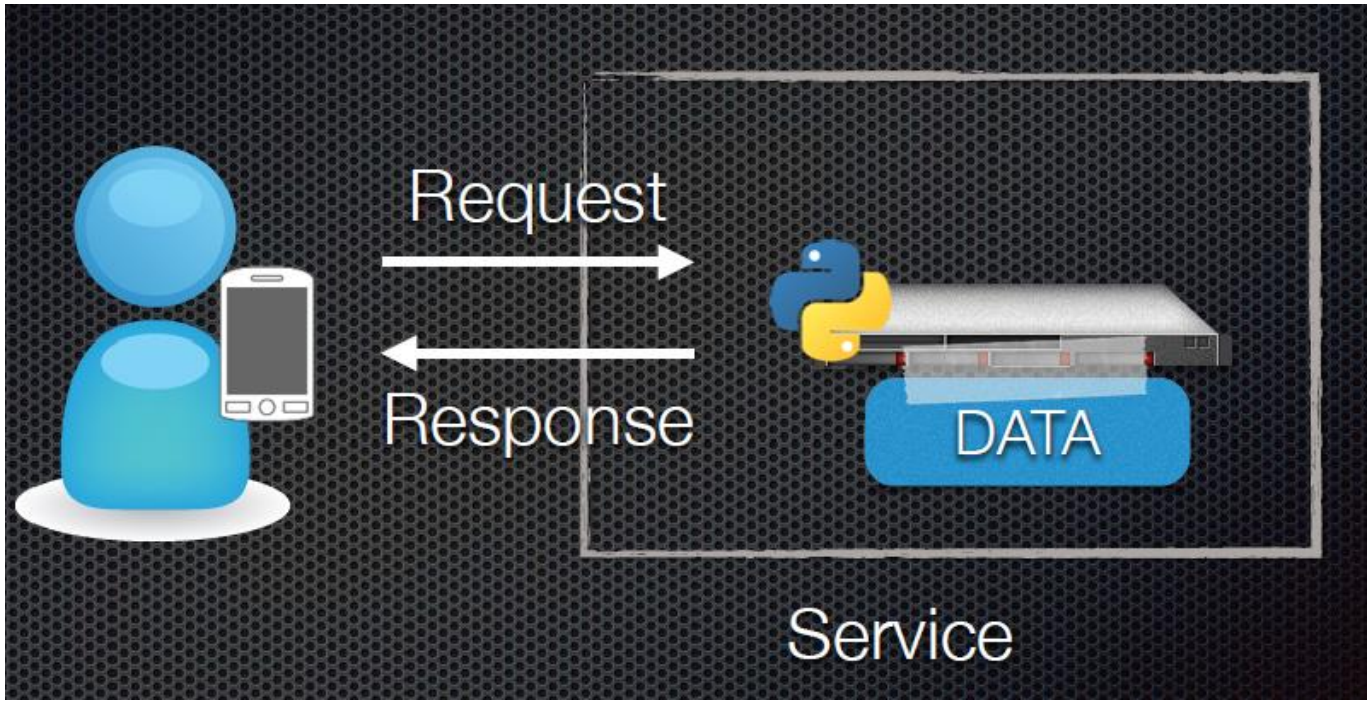
http://www.slideshare.net/yugoshimizu/ss-31022108


django


amazon
web services™


python™


S3
mongoDB
ELB
이야기 범위
fluentd
Application
EC2
redis
ElasticCache
RabbitMQ
MessageQueue
MySQL
RDS
MySQL
EC2
Job
Horizontal Partitioning
Database

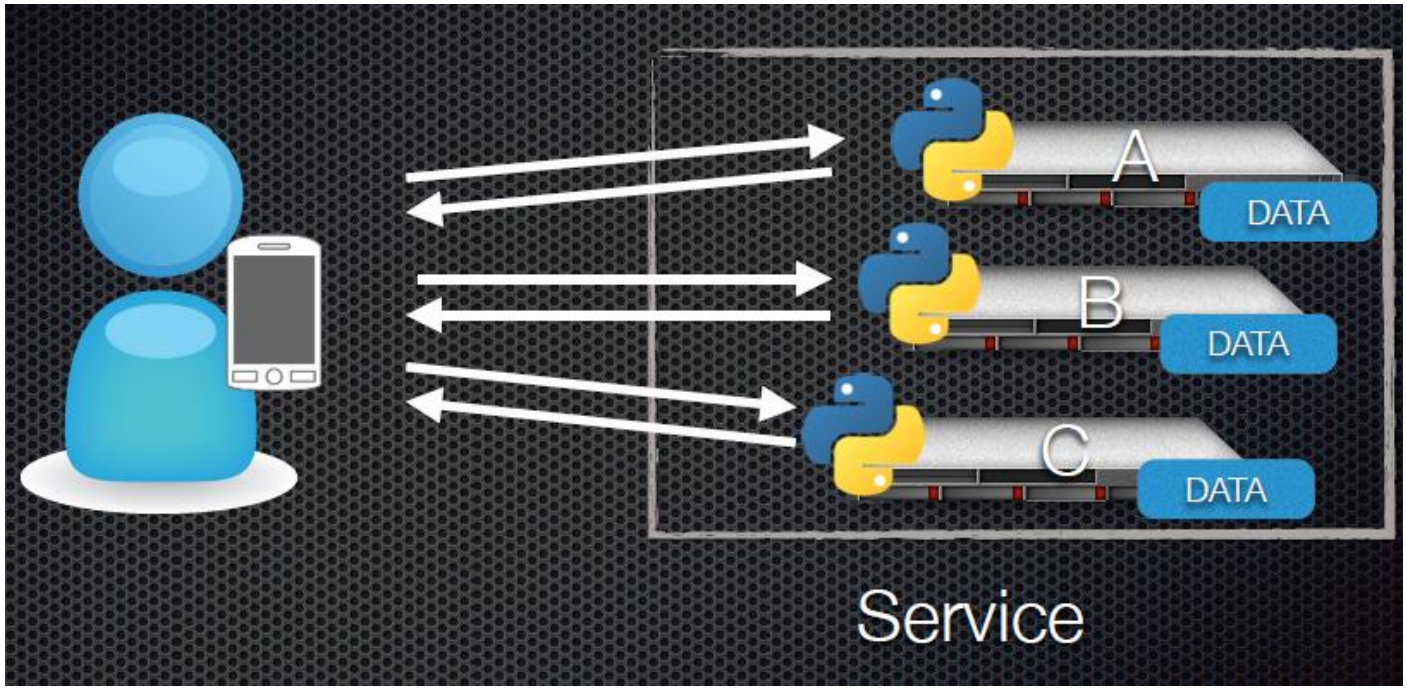


- 가장 간단한 형태.
- 서버가 죽으면 데이터 소멸.
- 서비스도 중지.
- 쓰기 작업 중 죽으면 이상한 데이터가 남는다.

요청이 늘어난다면

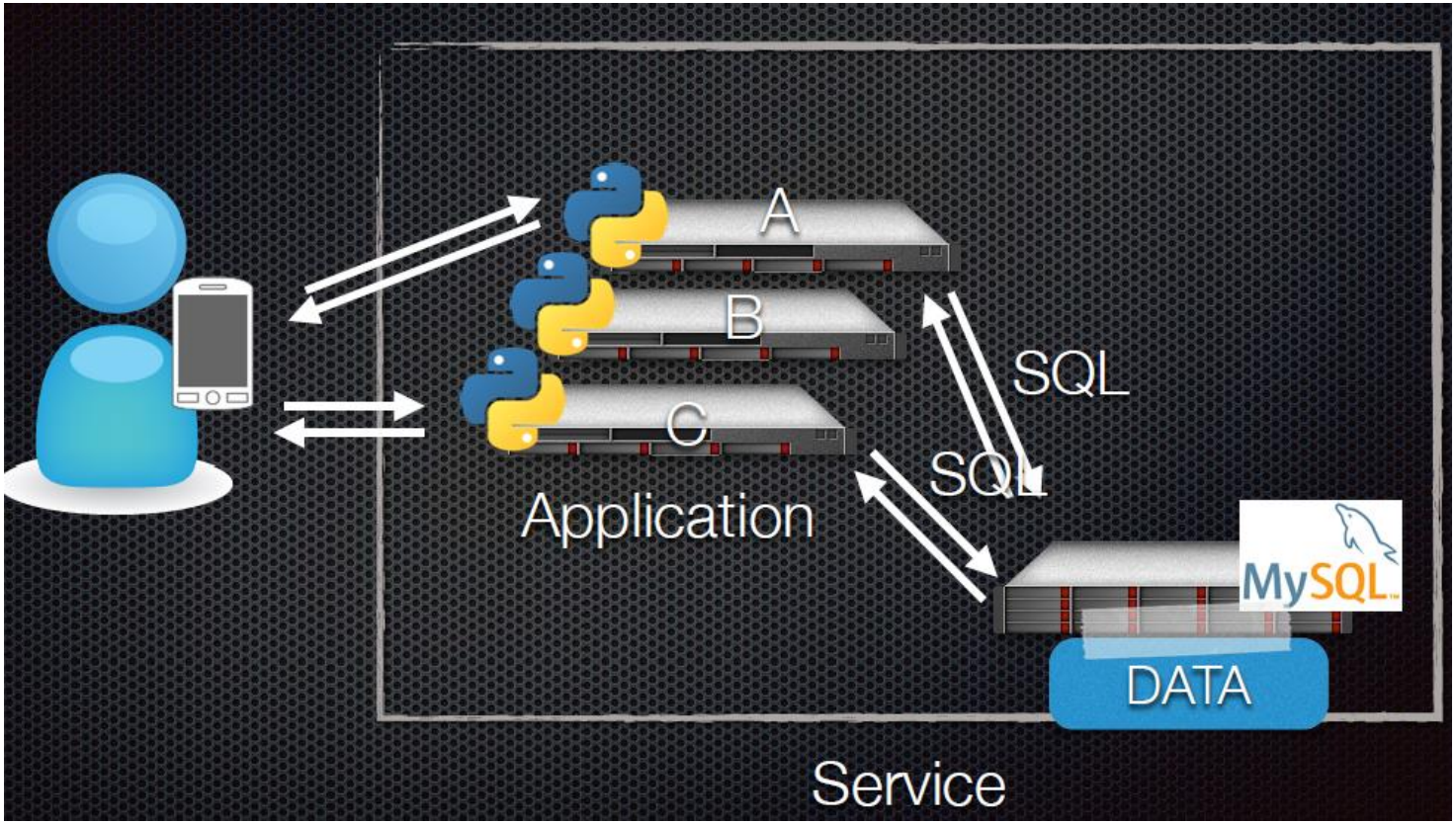
- 스케일 업
  스펙이 좋은 서버로 바꾸어서 처리 능력 UP

- 스케일 아웃
  대수를 늘려서 처리 능력 UP

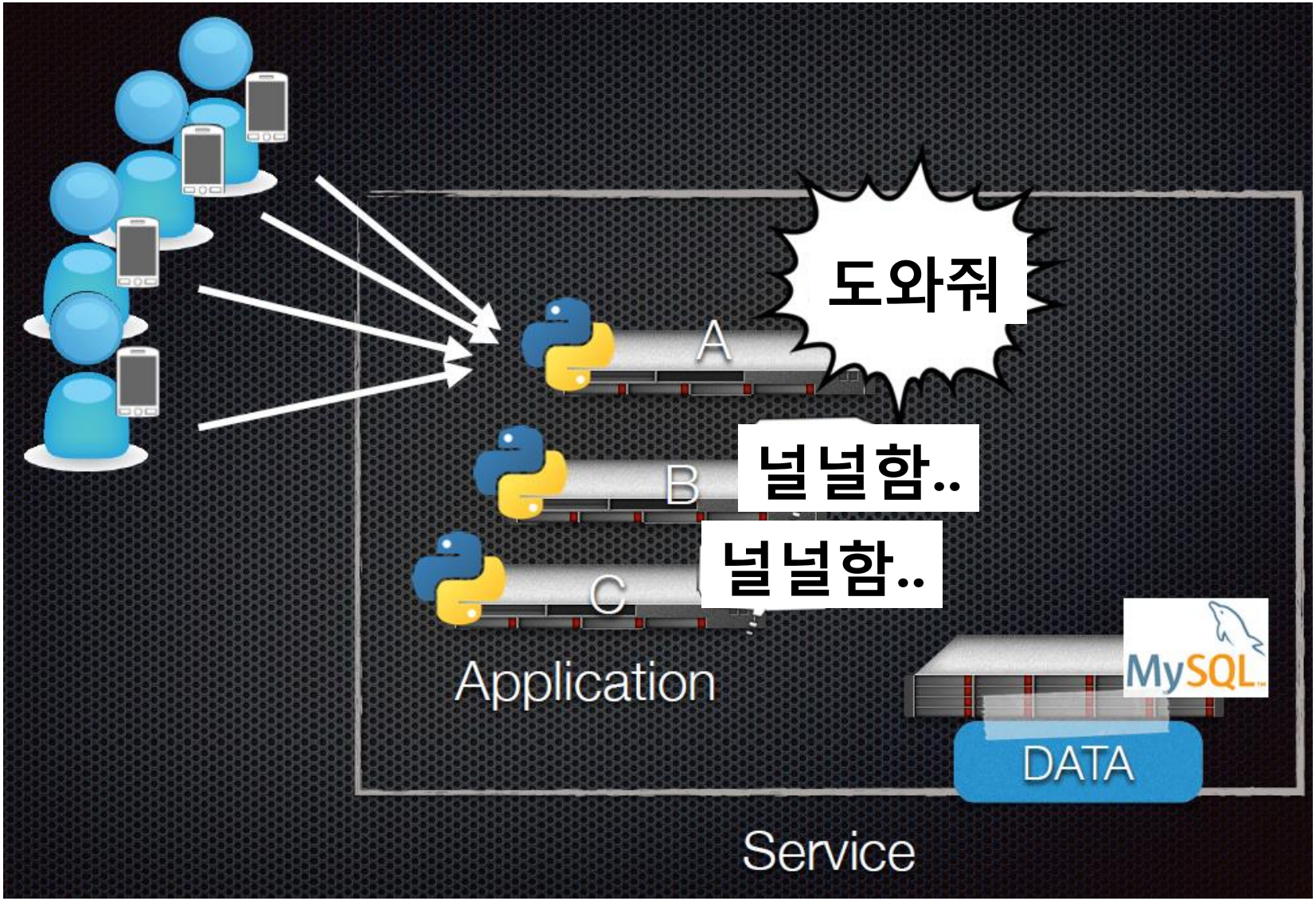


단순히 서버를 늘린다면

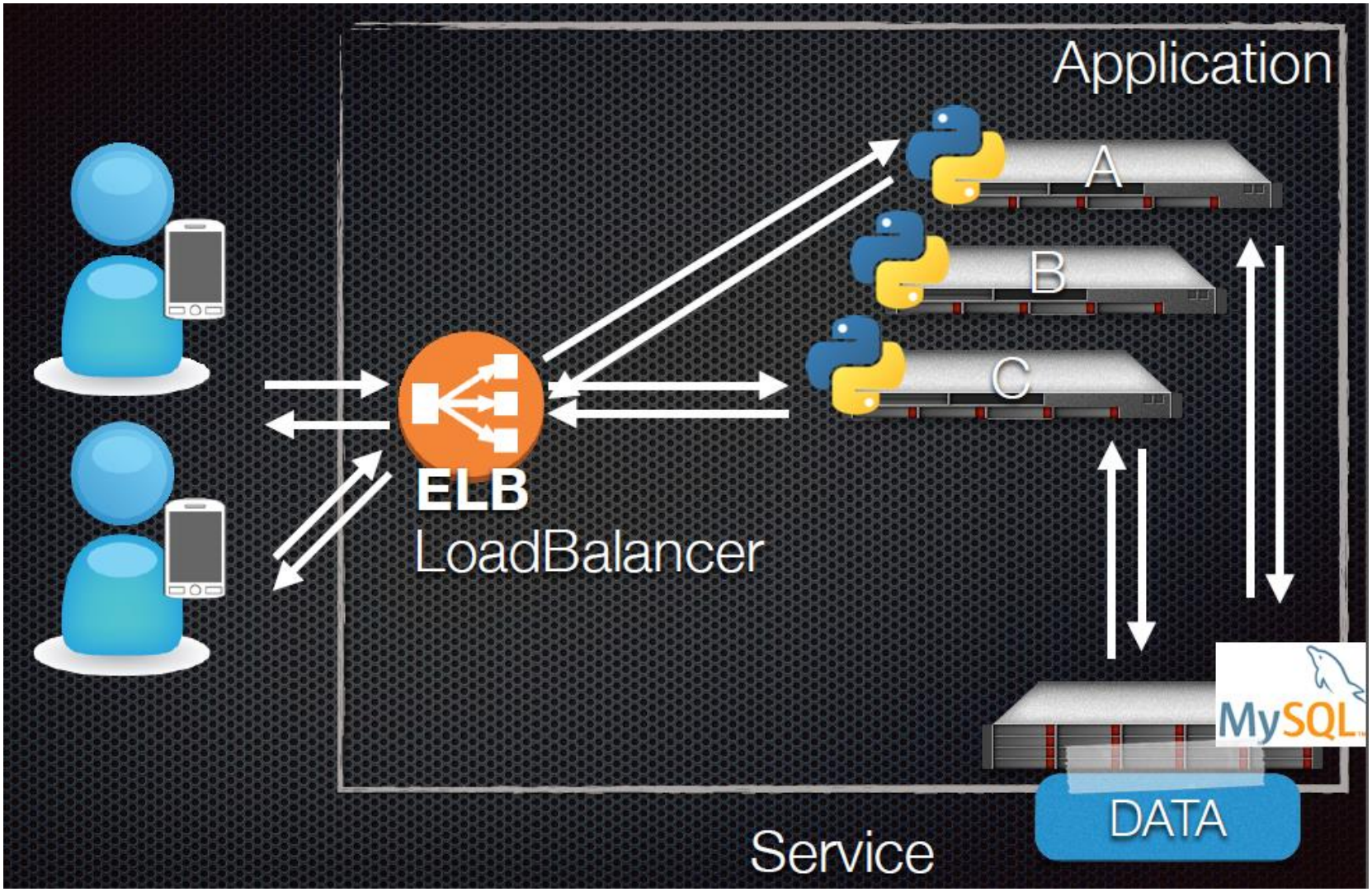
- 특정 서버에 부하가 집중할 수 있다.
- 모든 데이터를 볼려면 각각의 서버에 접속해야 한다.
- 가용성, 정합성 문제를 해결하지 않고 있다.



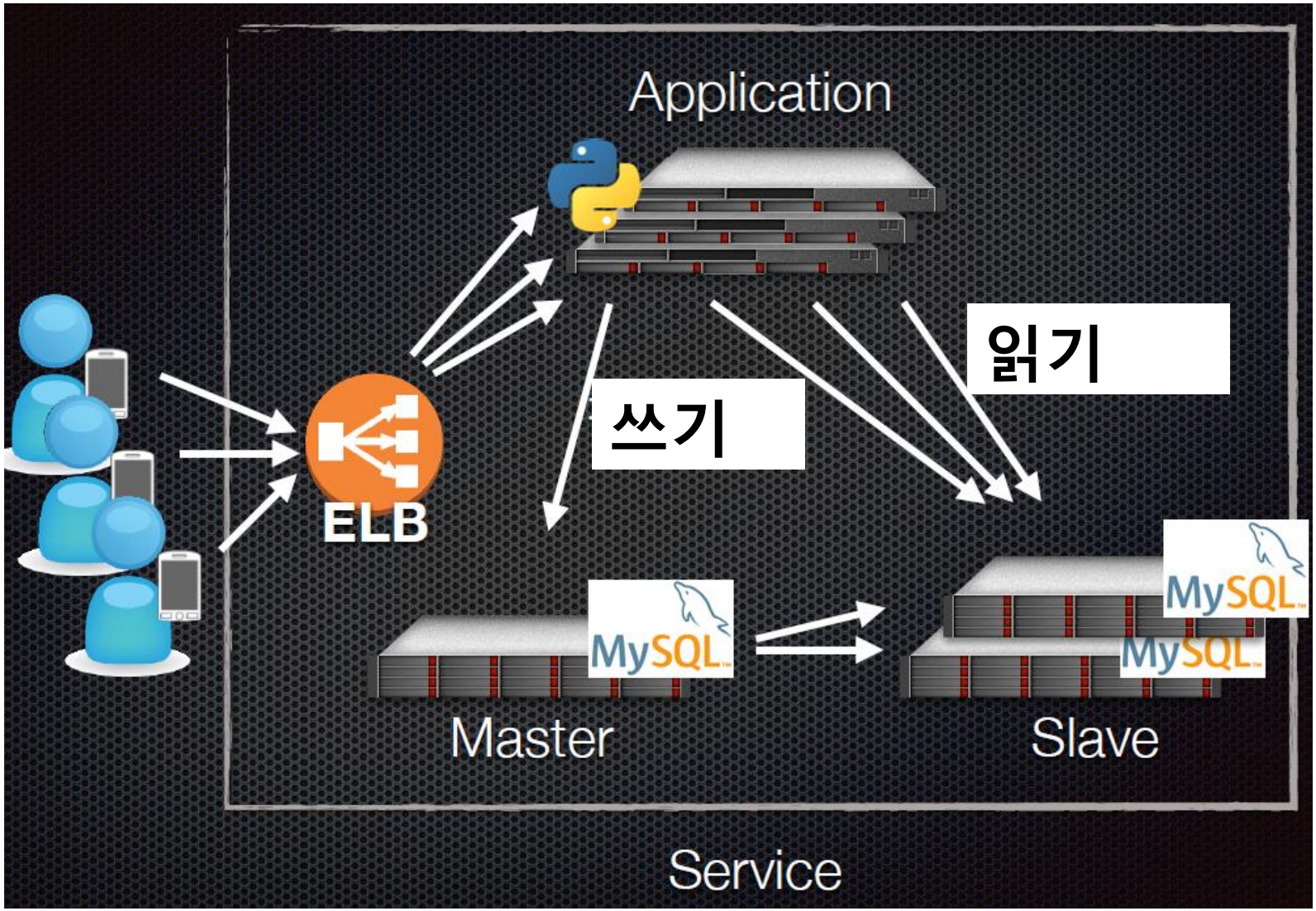
- 데이터 저장에 RDB 사용
- 데이터를 다루는 방법이 통일
- 어떤 애플리케이션 서버라도 같은 데이터를 다룬다
- 유저는 어떤 서버 하나에만 접속해도 괜찮다


도와줘
A
널널함..
B
널널함..
C
Application
MySQL
DATA
Service


Elastic Load Balancing


Application
A
B
C
ELB
LoadBalancer
MySQL
DATA
Service

레플리케이션

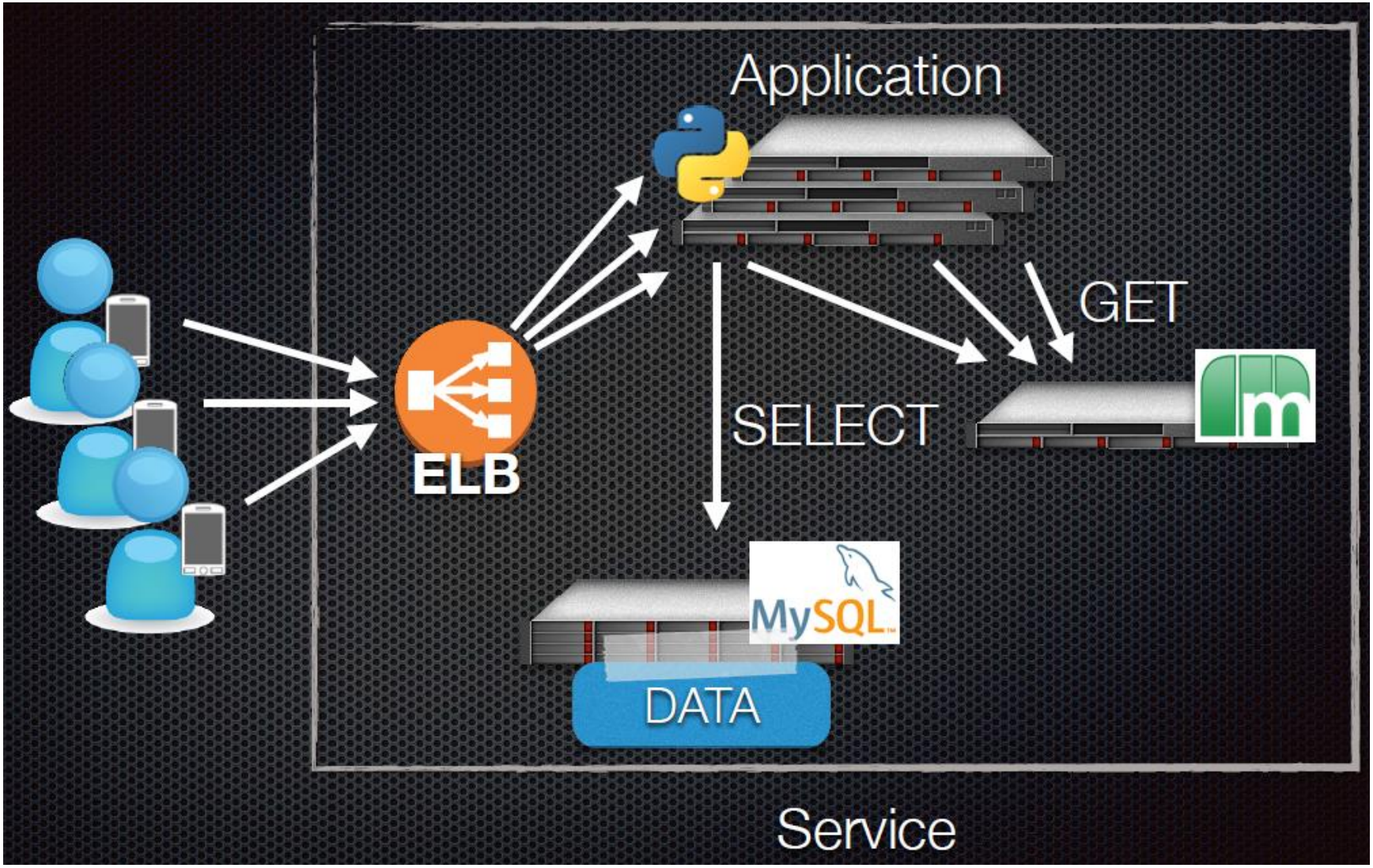

Application
읽기
쓰기
ELB
MySQL
MySQL
MySQL
Master
Slave
Service

- **장점**
  기존 로직에 큰 변경 없이 도입 가능
  자동으로 데이터 갱신
  재 기동해도 사라지지 않는다.

- 단점
  분산 가능한 것은 읽기만
  비동기의 경우 이전 데이터가 최신 데이터로 인식된다
  동기의 경우 Master 성능이 나빠진다

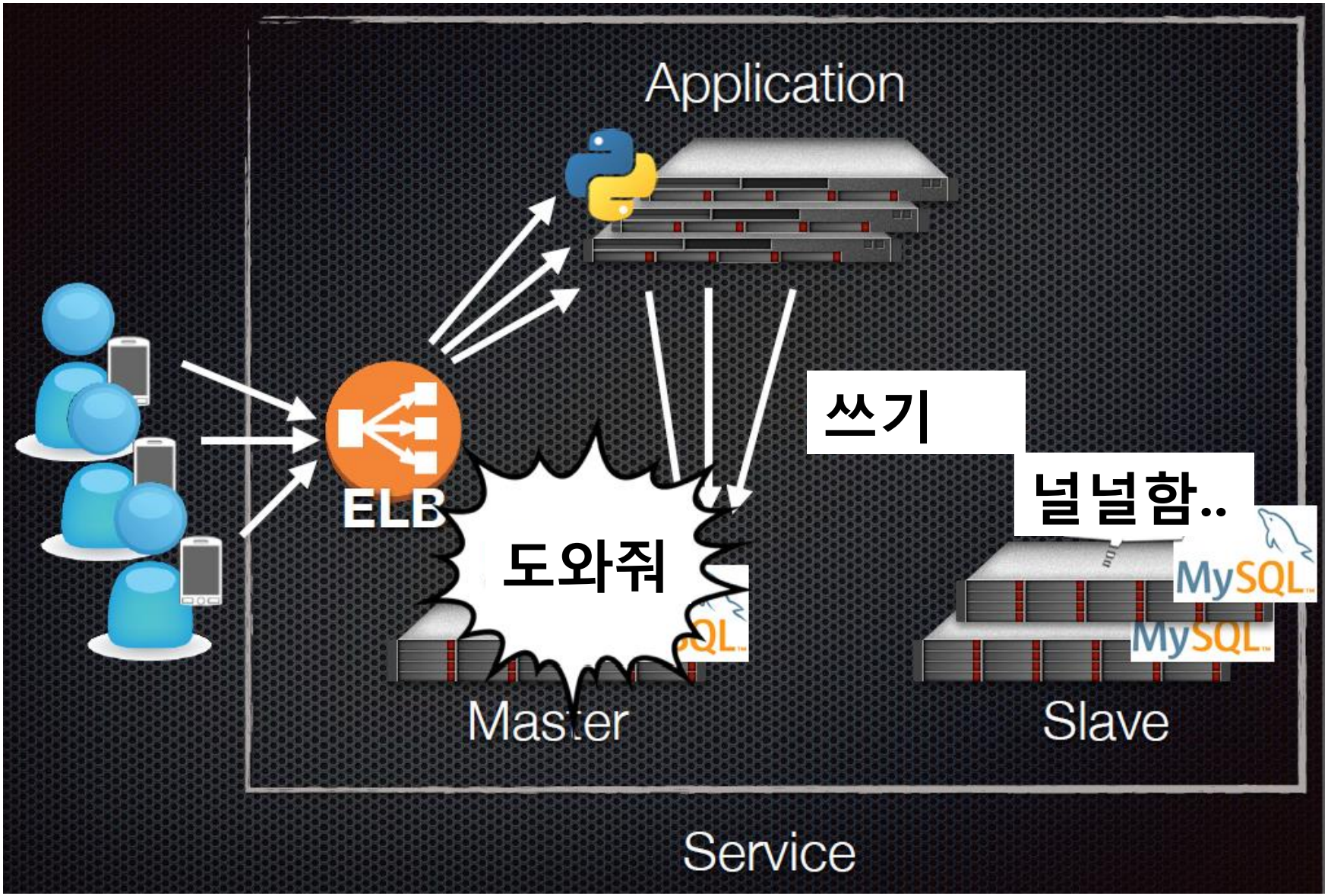
KeyValueStire (KVS)

- Key와 Value 쌍으로 데이터를 관리하는 DB
- Memcache, DynamoDB, Riak, Redis… etc
- 각각의 DB는 특징, 사용 방법, 용도가 다르다.


Application
GET
SELECT
ELB
MySQL
DATA
Service

Redis

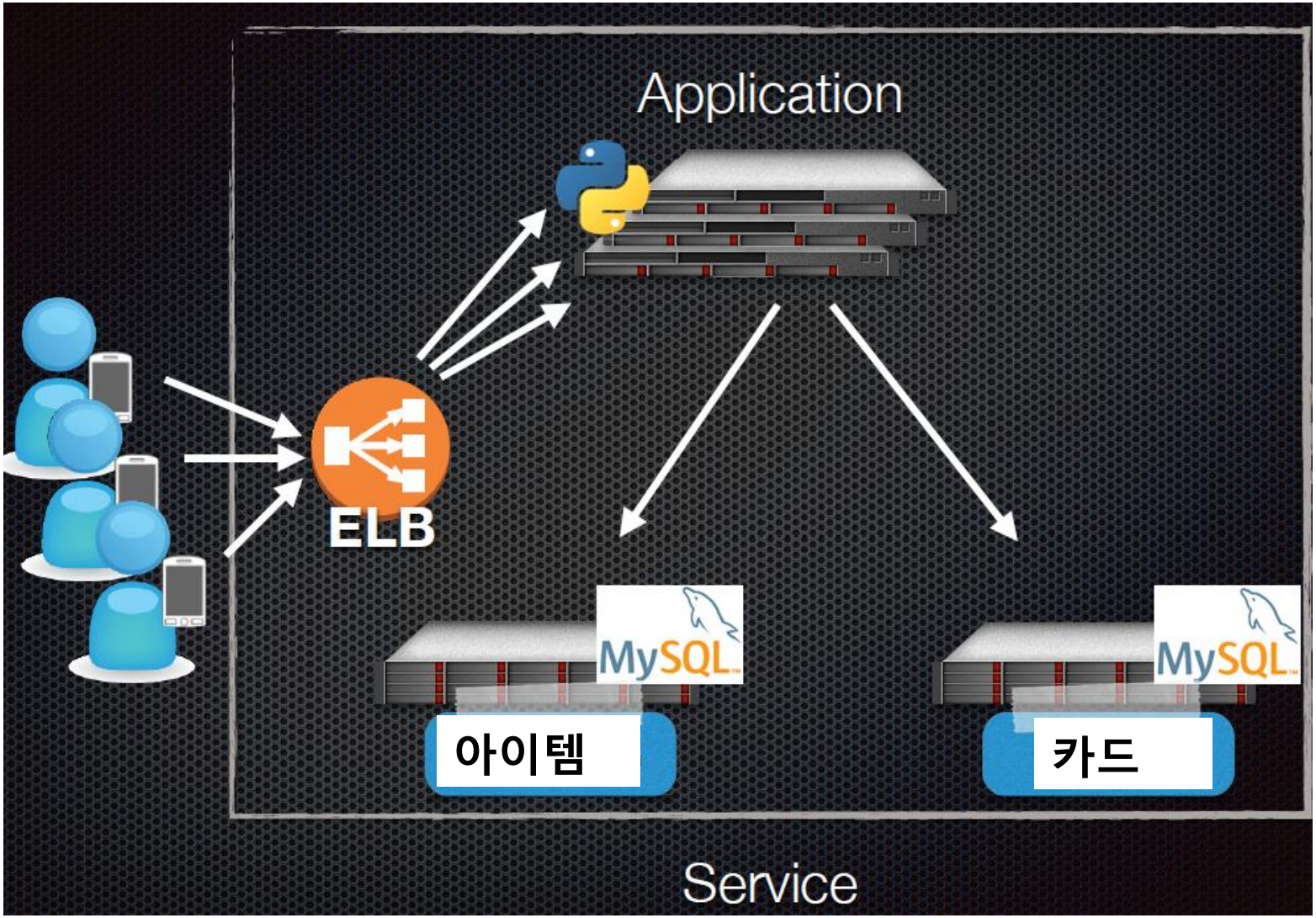
- 인 메모리 형 KVS
- 영구 저장 가능
- 레플리케이션 가능
- 데이터 형(LIST, SET, SORTED SET, HASH)
- 복잡한 조작을 atomic하게 실행 가능


Application
ELB
쓰기
넓널함..
도와줘
MySQL
MySQL
Master
Slave
Service

수직 샤딩

ID	체력	EXP	직업
Player A	100	100	전사
Player B	98	200	격투가
Player C	70	50	마법사
Player D	66	120	승려

DB1 DB2 DB3


Application
ELB
MySQL
MySQL
아이템
카드
Service

장점
분할 해도 집계가 용이
유니트 제약이 유지 된다.

단점
아이템에 부하가 집중 된다면 방법이 없다

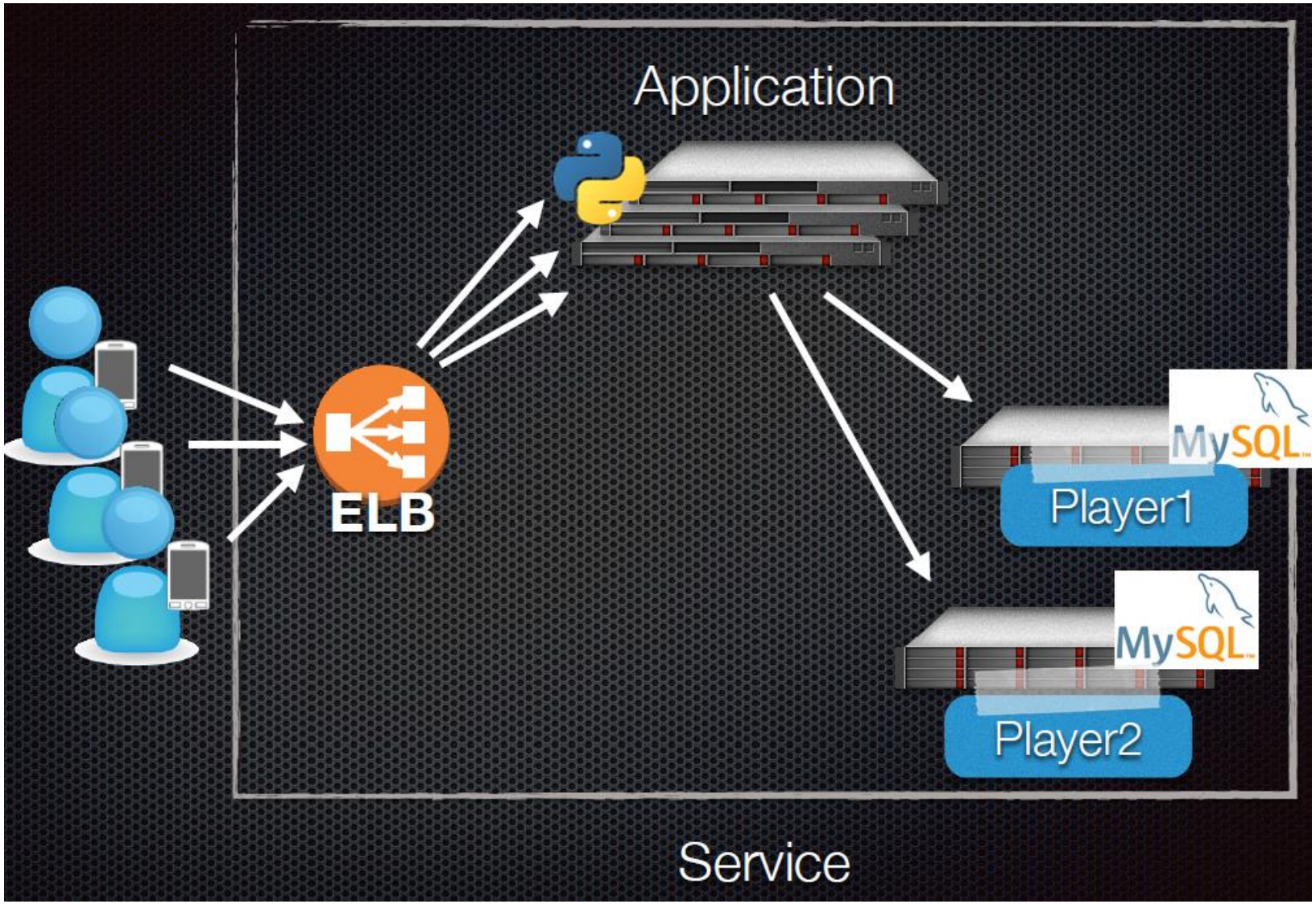
수평 샤딩

ID	체력	EXP	직업
Player A	100	100	전사
Player B	98	200	격투가
Player C	70	50	마법사
Player D	66	120	승려

DB1

DB2

DB3


Application
ELB
MySQL
Player1
MySQL
Player2
Service

장점
거의 균등하게 부하가 분산된다
플레이에 대한 처리를 여러 DB를 걸치지 않아도 된다.

단점
집계가 어렵다
기술적 난이도
유니크 제약이 부서질 수 있다