4장. 처리율 제한 장치의 설계

API GateWay란 무엇일까?

제이 (jay-so)

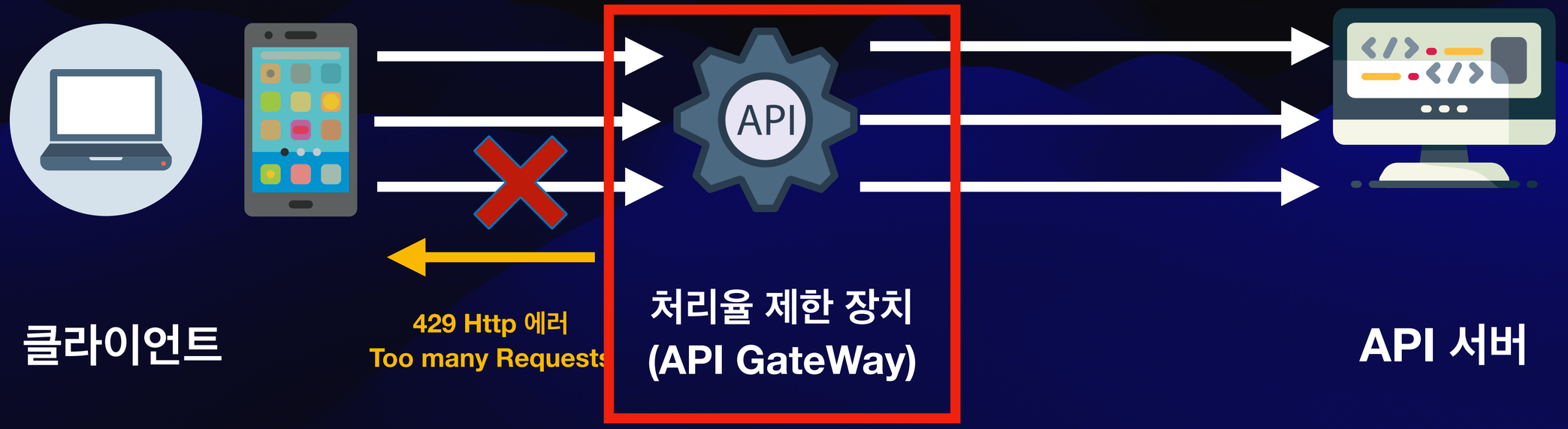
목차

1. API GateWay란 무엇일까?

- API GateWay 프레임워크
- API GateWay 패턴

1. API GateWay란 무엇일까?

API GateWay



1. API GateWay란 무엇일까?

API GateWay 프레임워크 vs API GateWay 패턴

API GateWay 프레임워크

Spring Cloud

AWS API GateWay

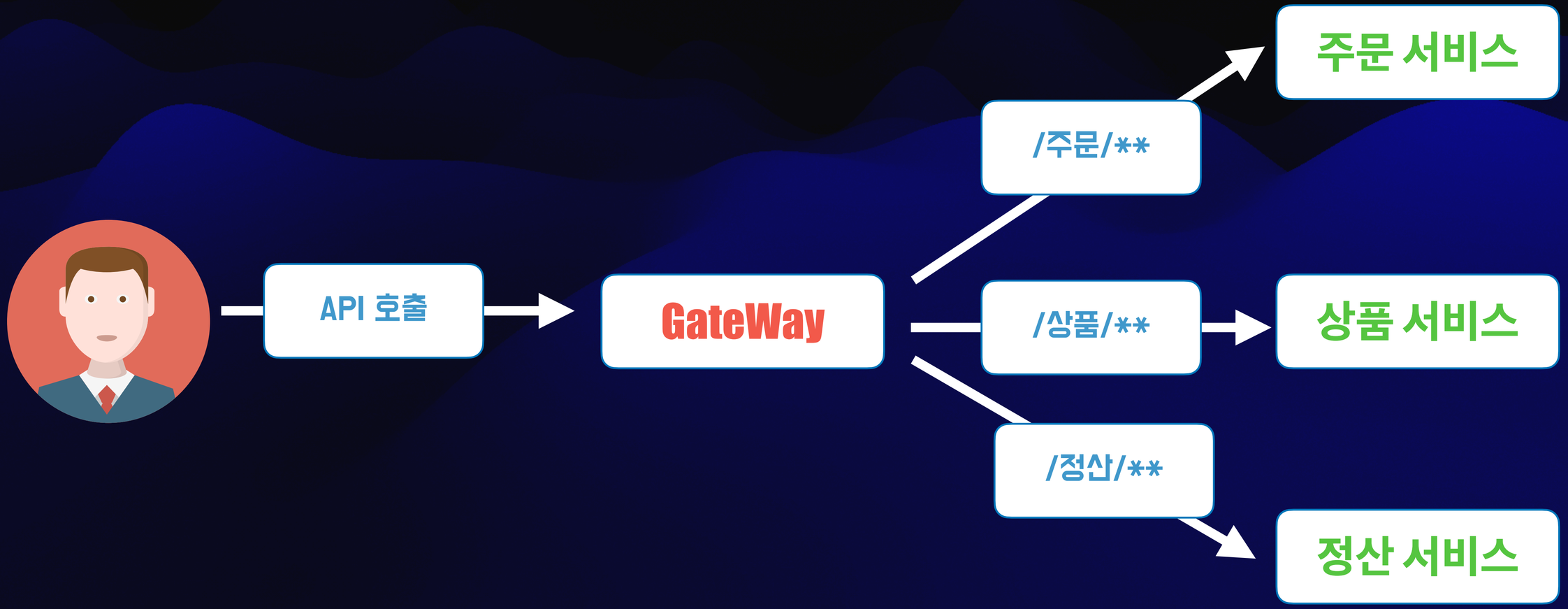
Kong(콩)

Apigee(애피지)

1. API GateWay란 무엇일까?

API GateWay 프레임워크 vs API GateWay 패턴

API GateWay 패턴



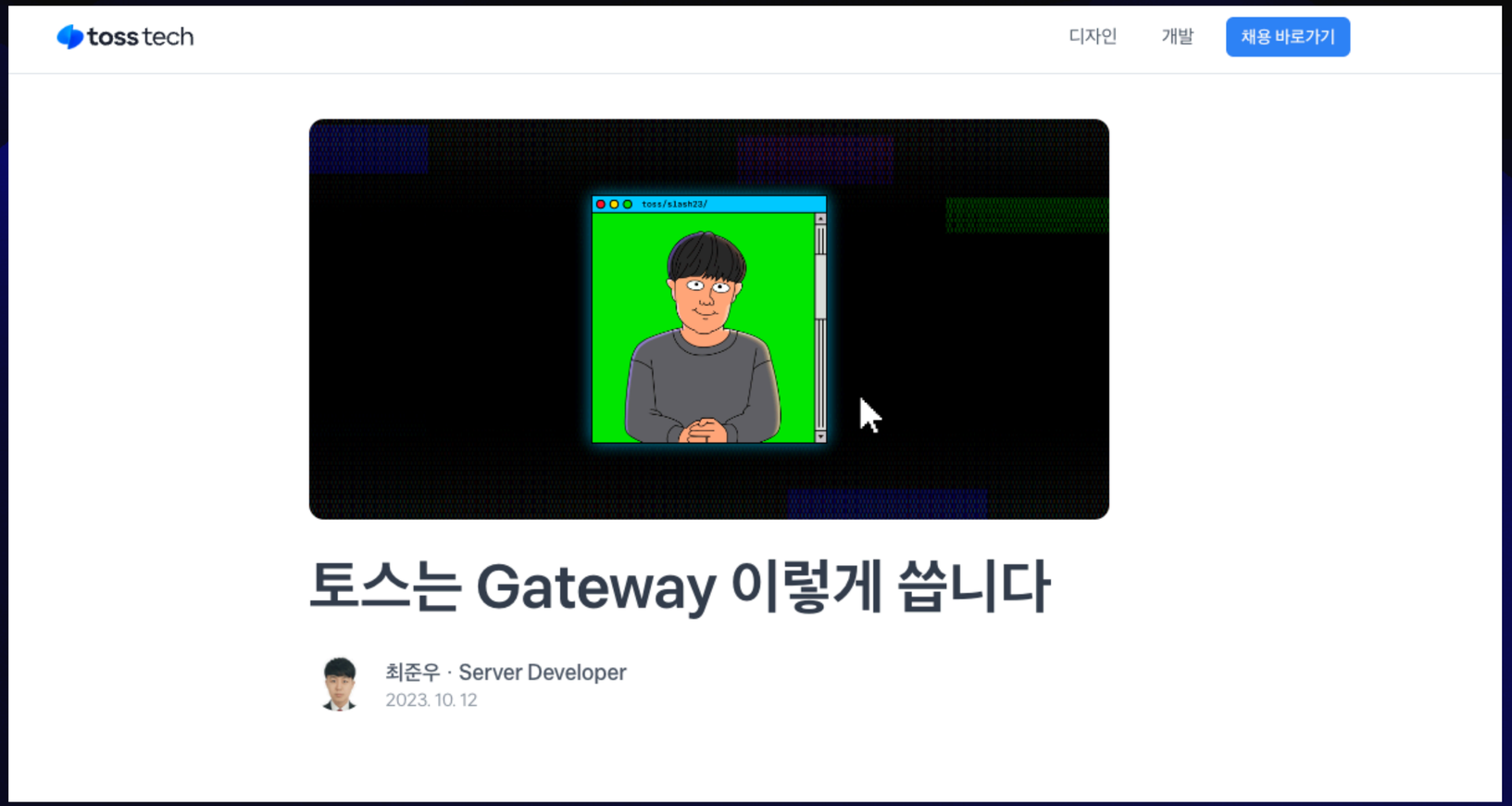
2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

- Spring Cloud의 GateWay의 구조
- Spring Cloud의 Filter의 구분
- 예시: 토스의 GateWay 사용 예시

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

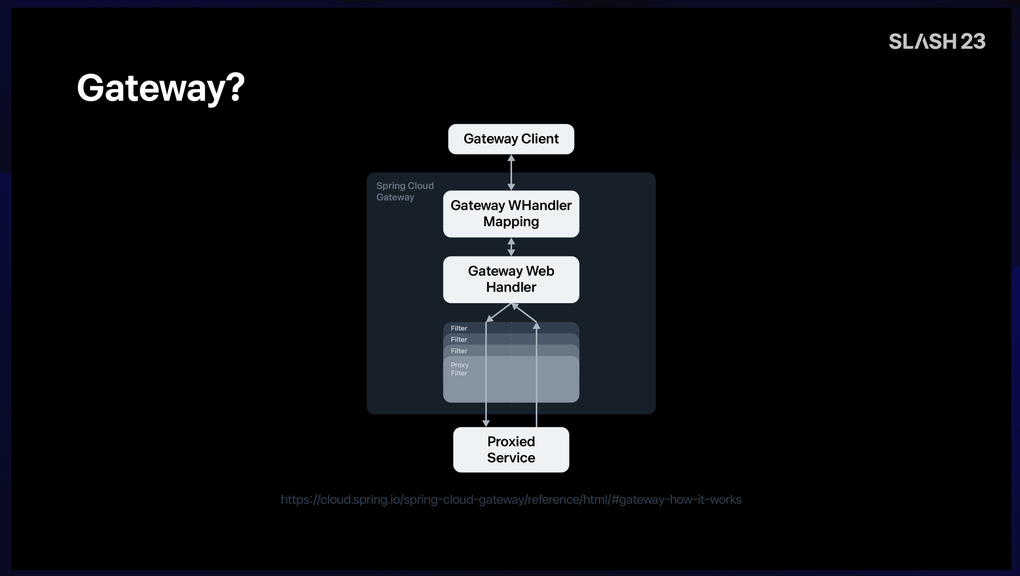
다양한 기능을 가진 API GateWay

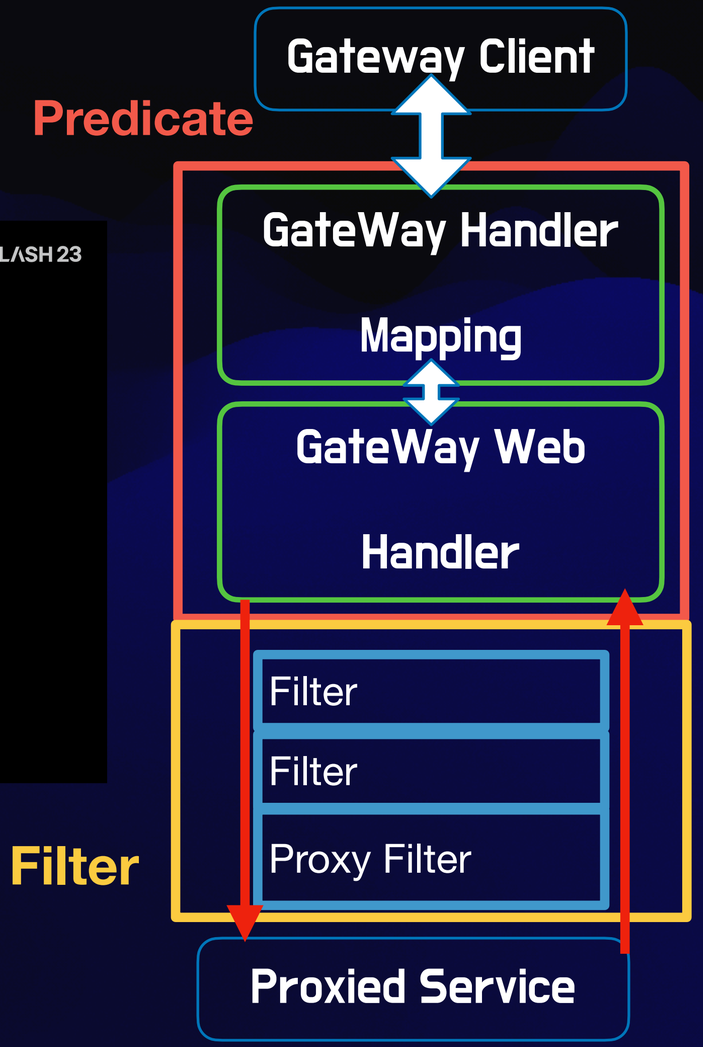
참고: 23년도 토스 테크 블로그_최준우(서버 플랫폼팀)



2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

Spring Cloud의 GateWay 구조





Gateway Handler Mapping

경로와 일치하는지 판단

Gateway Web Handler

요청과 관련된 필터 체인을 통해 요청을 전송

Filter

필터는 프록시 요청이 보내지기 전후에 나뉘어서 로직을 수행

Proxy Filter

프록시 요청이 처리될 때 수행

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

Spring Cloud의 Filter의 구분

글로벌 필터(Global Filter)

- 모든 경로에 적용되며, 로깅과 보안 검사 같은 공통 기능을 수행하는 필터
- 사전 필터(Pre Filter)와 사후 필터(Post Filter)가 속해져 있음

라우터 필터(Router Filter)

- 특정 경로에만 적용되며, 특정 서비스의 요청에 헤더를 추가하는 등의 기능을 하는 필터
- 프록시 필터(Proxy Filter)가 속해져 있음

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay

1. 요청에 대한 전처리 및 후처리 작업

2. 유저 정보를 이용한 로직 수행

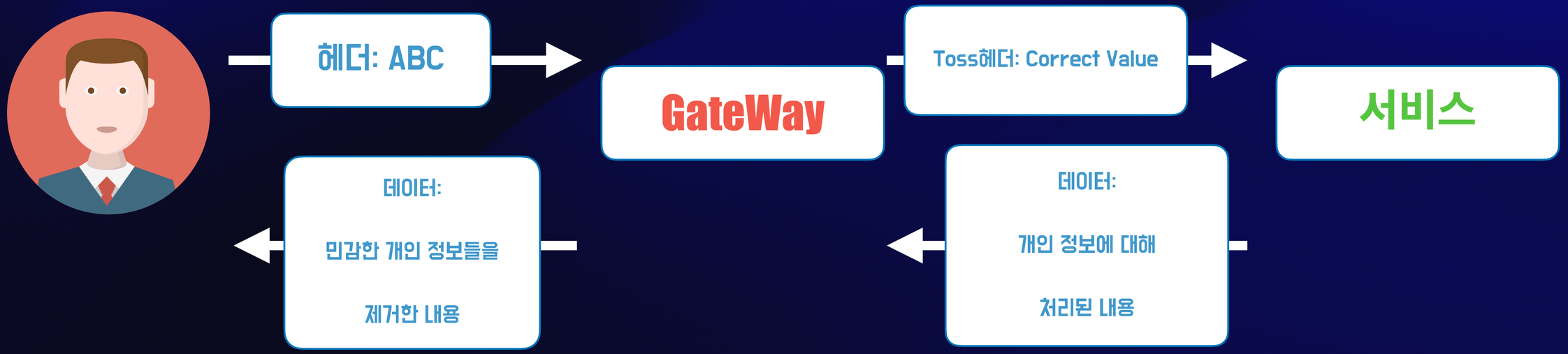
3. 보안과 서비스 안정화

4. 로깅과 모니터링

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 1. 요청에 대한 전처리 및 후처리 작업

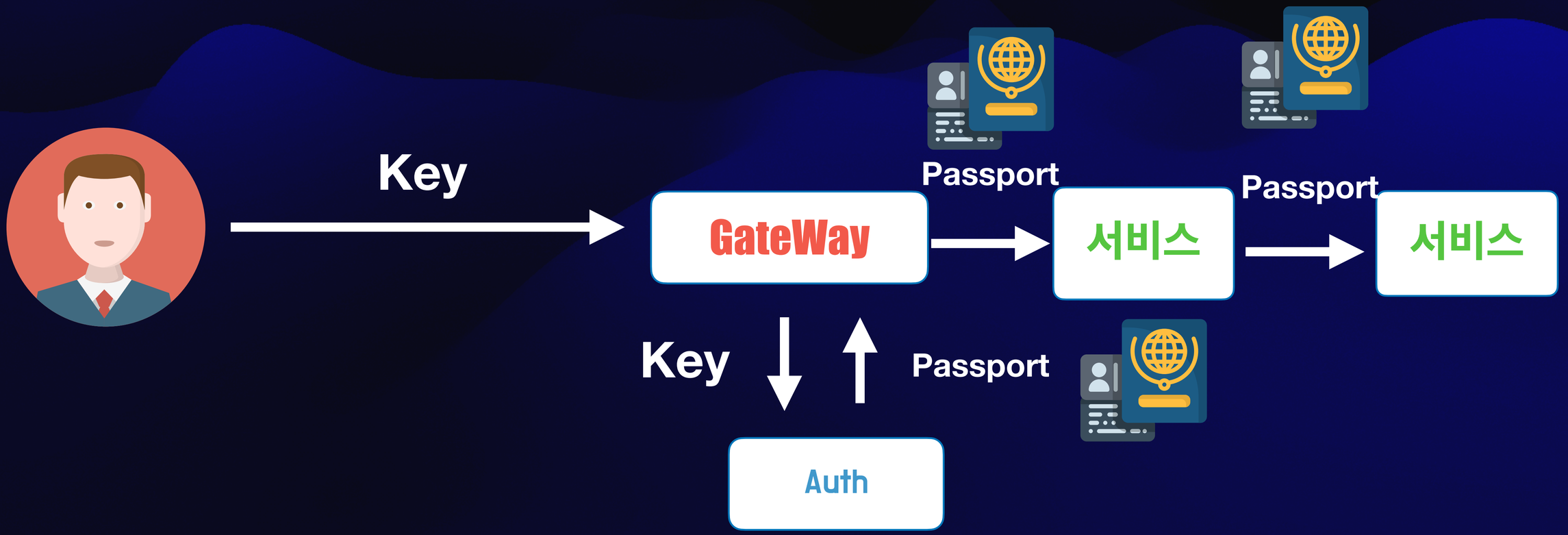
Sanitize 작업



2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 2. 유저 정보를 이용한 로직 수행

Passport 정보



2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

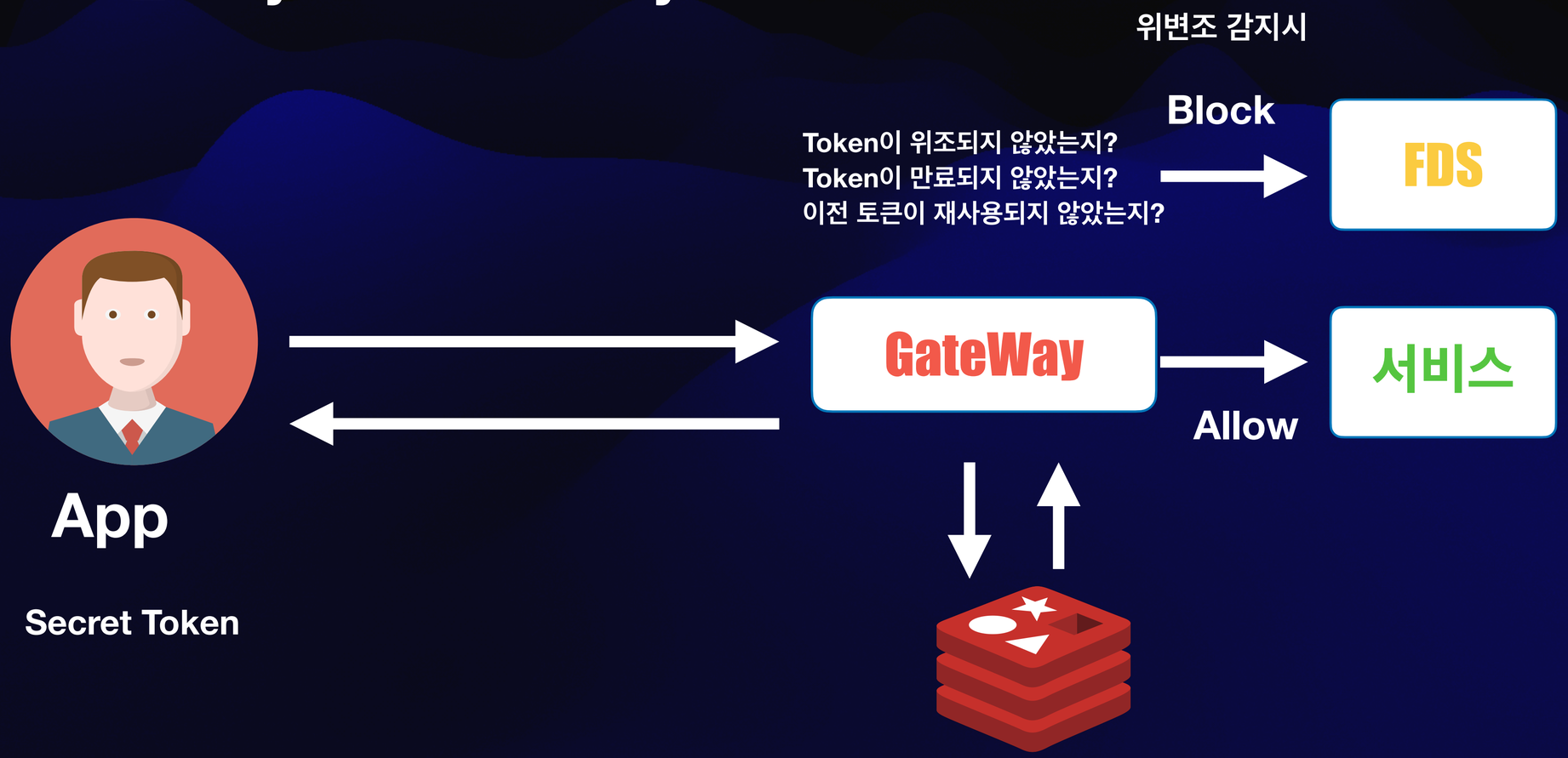
토스의 GateWay - 3. 보안과 서비스 안정화

보안 - 종단 간 암호화

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 3. 보안과 서비스 안정화

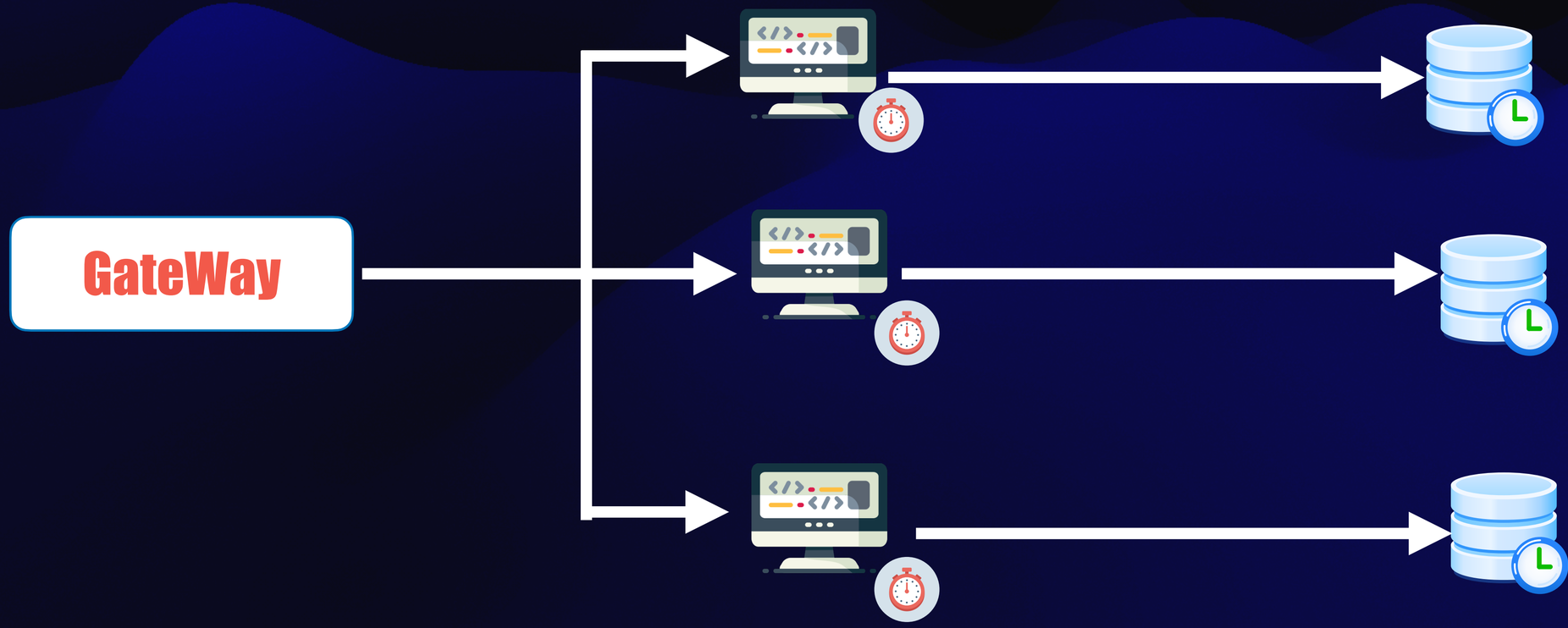
보안 - Dynamic Security



2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 3. 보안과 서비스 안정화

서비스 안정화 -서킷 브레이커



2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 4. 로깅과 모니터링

로깅 - Route 응답로그

Time ↓	status.extraData.routeId	status.extraData.routeMethod	status.extraData.routeUri	message
Jan 19, 2023 @ 22:50:36.353	slash-demo	POST	http:///slash-demo.service/slash-demo/hello	POST /slash-demo/hello -> 200 took 212ms
Jan 19, 2023 @ 22:50:36.229	sample-service-a	GET	http://sample-service-a.service/sample-service-a/foo	POST /sample-service-a/foo -> 200 took 15ms
Jan 19, 2023 @ 22:50:36.189	sample-service-b	POST	http://sample-service-b.service/sample-service-b/bar	POST /sample-service-b/bar -> 200 took 20ms
Jan 19, 2023 @ 22:50:35.245	sample-service-c	POST	http:///sample-service-c.service/sample-service-a/uno	POST /sample-service-a/uno -> 200 took 21ms
Jan 19, 2023 @ 22:50:33.094	sample-service-d	GET	http://sample-service-d.service/sample-service-d/dos?hi=bi	POST /sample-service-d/dos -> 200 took 42ms

2. 처리율 제한 장치 외의 다양한 기능을 가진 API GateWay

토스의 GateWay - 4. 로깅과 모니터링

메트릭 - 시스템 메트릭(System Metric)

메트릭 - 게이트웨이 메트릭(Gateway Metric)

3. 처리율 제한 장치로써의 API GateWay

- 처리율 제한 알고리즘의 종류
- 토큰 버킷 알고리즘
- 누출 버킷 알고리즘
- 고정 윈도 카운터 알고리즘
- 이동 윈도 카운터 알고리즘

3. 처리율 제한 장치로써의 API GateWay

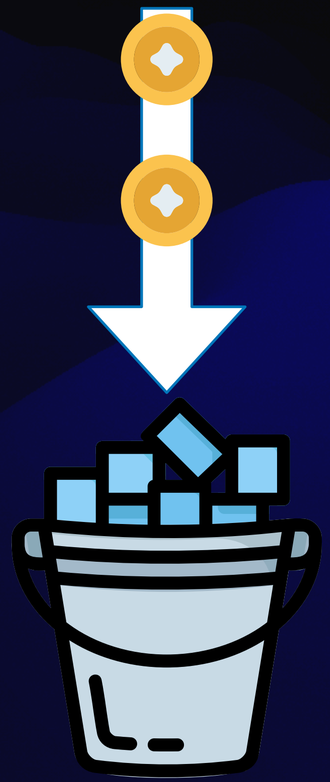
처리율 제한 알고리즘의 종류

	토큰 버킷 알고리즘	누출 버킷 알고리즘	고정 윈도 카운터 알고리즘	이동 윈도 로깅 알고리즘	이동 윈도 카운터 알고리즘
장점	구현이 쉬움. 짧은 시간에 집중되는 트래픽에 대응가능	트래픽의 일정한 속도를 유지.	구현이 간단하고 이해하기 쉬움	시간에 따른 정확한 요청 속도 측정 가능	짧은 시간에 집중되는 트래픽에 대응 가능. 메모리 효율이 좋음
단점	버킷의 크기와 토큰 공급률이라는 2개의 인자를 튜닝하는 것이 어려움	갑작스런 트래픽 증가에 유연하게 대응하기 어려움. 버킷의 크기와 토큰 공급률이라는 2개의 인자를 튜닝하는 것이 어려움	윈도 경계에서의 요청 집중으로 인한 한계 초과 가능성	거부된 요청의 타임스탬프도 보관하기 때문에 메모리 사용량이 증가함	구현이 어려움

3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 토큰 버킷 알고리즘

토큰 = 요청



꽉 차 있다면, 토큰을 버림

토큰 버킷
= 요청을 허용할 수 있는 처리율 양

3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 토큰 버킷 알고리즘(코드 레벨)

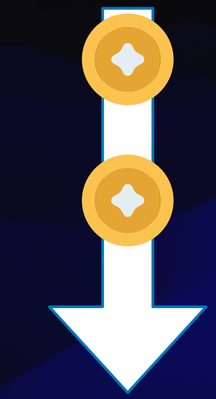
```java
@RestController
public class TestController {

  private final Bucket bucket;

  public TestController() {
    // 충전 간격을 10초로 지정하며, 한 번 충전할 때마다 2개의 토큰을 충전한다.
    Refill refill = Refill.intervally(2, Duration.ofSeconds(10));

    // Bucket의 총 크기는 3
    Bandwidth limit = Bandwidth.classic(3, refill);

    // 총 크기는 3이며 10초마다 2개의 토큰을 충전하는 Bucket
    this.bucket = Bucket.builder()
        .addLimit(limit)
        .build();

  }

  @GetMapping(value = "/api/test")
  public ResponseEntity<String> test() {
    if (bucket.tryConsume(1)) {
      return ResponseEntity.ok("success!");
    }
    return ResponseEntity.status(HttpStatus.TOO_MANY_REQUESTS).build();
  }

}
```

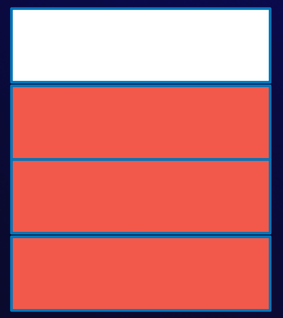
```
~ curl -v -X GET http://localhost:8080/api/test
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /api/test HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.86.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200
< Content-Type: text/plain;charset=UTF-8
< Content-Length: 8
< Date: Wed, 21 Jun 2023 11:34:06 GMT
<
* Connection #0 to host localhost left intact
success!%
~ curl -v -X GET http://localhost:8080/api/test
Note: Unnecessary use of -X or --request, GET is already inferred.
*   Trying 127.0.0.1:8080...
* Connected to localhost (127.0.0.1) port 8080 (#0)
> GET /api/test HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.86.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 429
< Content-Length: 0
< Date: Wed, 21 Jun 2023 11:34:06 GMT
<
* Connection #0 to host localhost left intact
~
```

3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 누출 버킷 알고리즘

토큰 = 요청





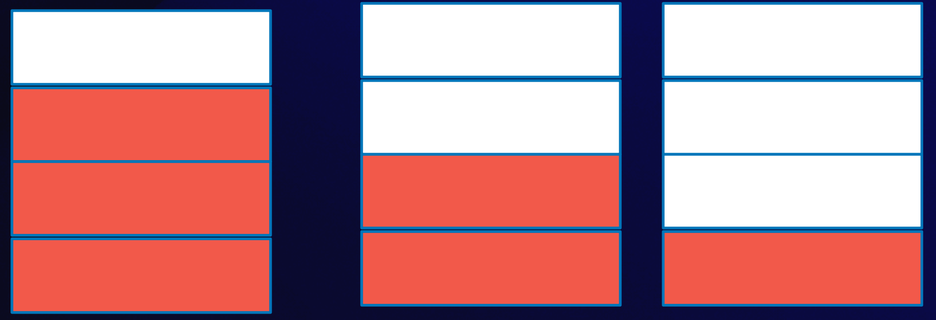
꽉 차 있다면, 토큰을 버림

토큰 버킷
= 요청을 허용할 수 있는 처리율 양

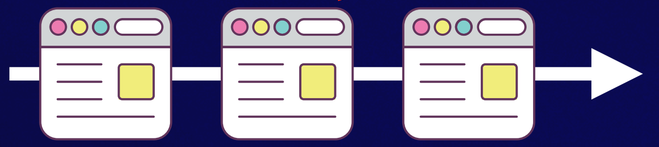
3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 고정 윈도 카운터 알고리즘

윈도 = 요청



꽉 차 있다면, 윈도를 버림



설정된 카운터의 양
= 요청을 허용할 수 있는 처리율 양

1시 0분 1시 1분 1시 2분

3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 이동 윈도 로깅 알고리즘

윈도 = 요청

이전 1분

현재 1분

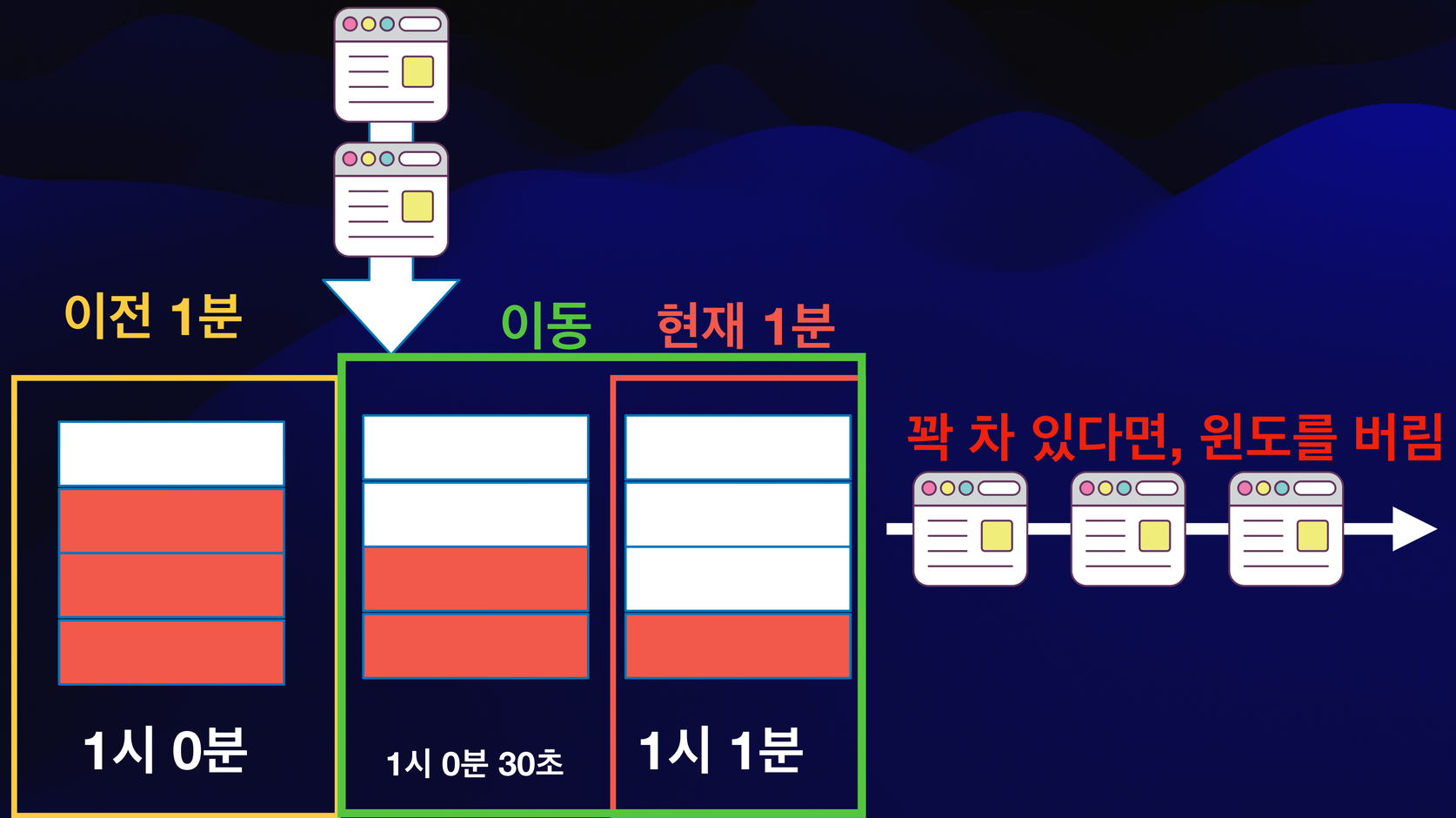
꽉 차 있다면, 윈도를 버림

설정된 카운터의 양
= 요청을 허용할 수 있는 처리율 양

3. 처리율 제한 장치로써의 API GateWay

처리율 제한 알고리즘의 종류 - 이동 윈도 카운터 알고리즘

윈도 = 요청



설정된 카운터의 양
= 요청을 허용할 수 있는 처리율 양

4. 참고 자료

4. 참고 자료

토스(토스는 GateWay를 이렇게 씁니다)

https://toss.tech/article/slash23-server

https://docs.tosspayments.com/resources/glossary/fds

Spring Cloud GateWay 프로젝트

https://velog.io/@ychxexn/Spring-Cloud-Gateway-%ED%94%84%EB%A1%9C%EC%A0%9D%ED%8A%B8-%EC%83%9D%EC%84%B1

https://github.com/ycheese

처리율 제한 알고리즘

https://github.com/jun-labs/spring-cloud

https://hogwart-scholars.tistory.com/entry/Spring-Boot-%EC%9E%90%EB%B0%94-%EC%8A%A4%ED%94%84%EB%A7%81%EC%97%90%EC%84%9C-%EC%B2%98%EB%A6%AC%EC%9C%A8-%EC%A0%9C%ED%95%9C-%EA%B8%B0%EB%8A%A5%EC%9D%84-%EA%B5%AC%ED%98%84%ED%95%98%EB%8A%94-4%EA%B0%80%EC%A7%80-%EB%B0%A9%EB%B2%95

Spring Cloud GateWay의 구조

https://github.com/jun-labs/spring-cloud