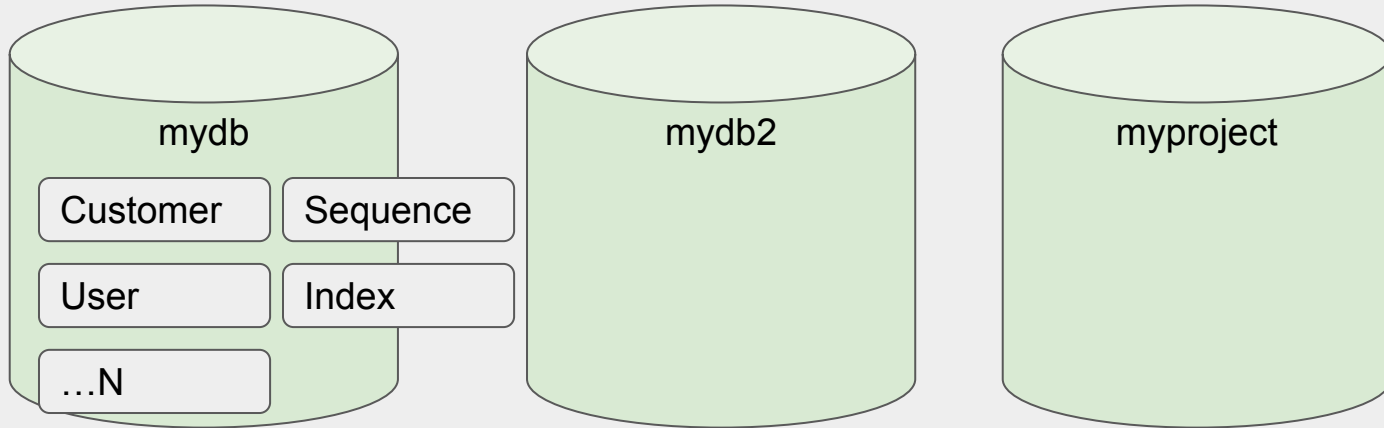


설명 PPT

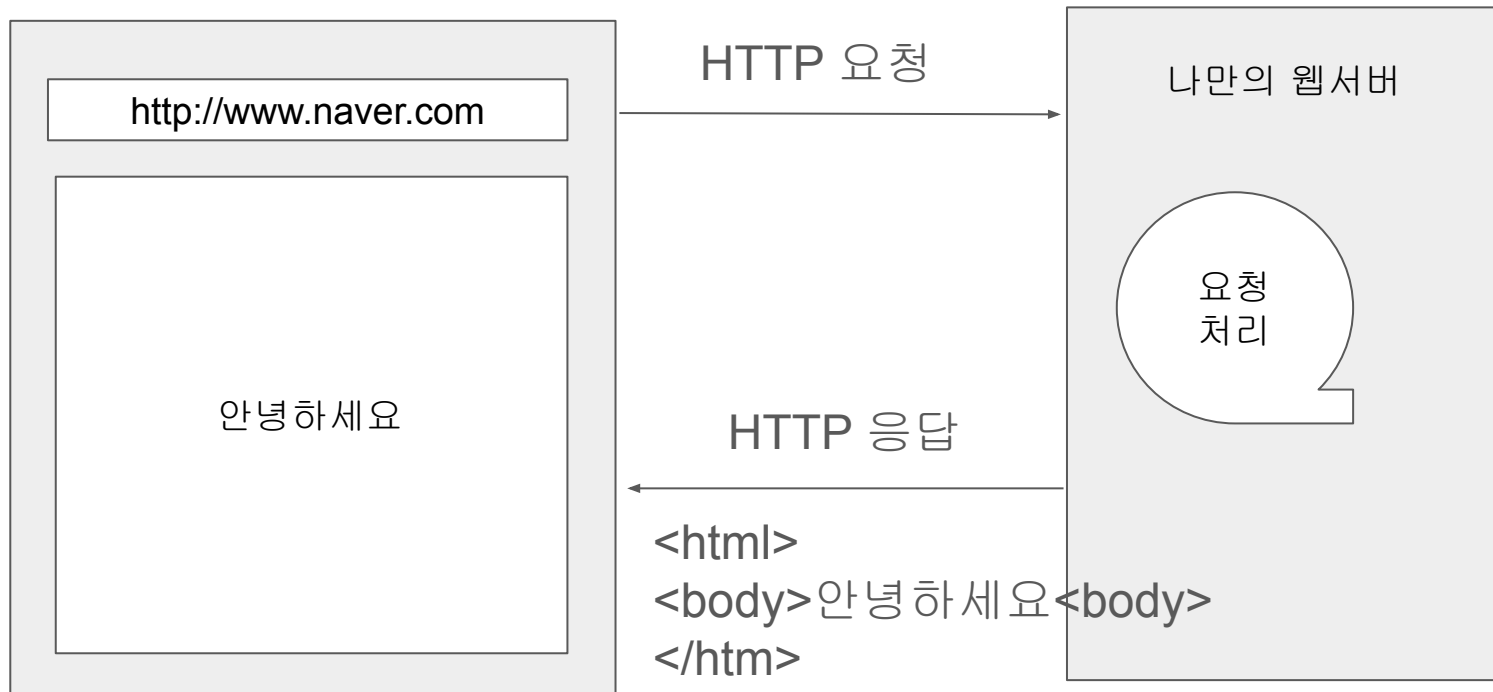
남부_웹개발실무

Posgresql

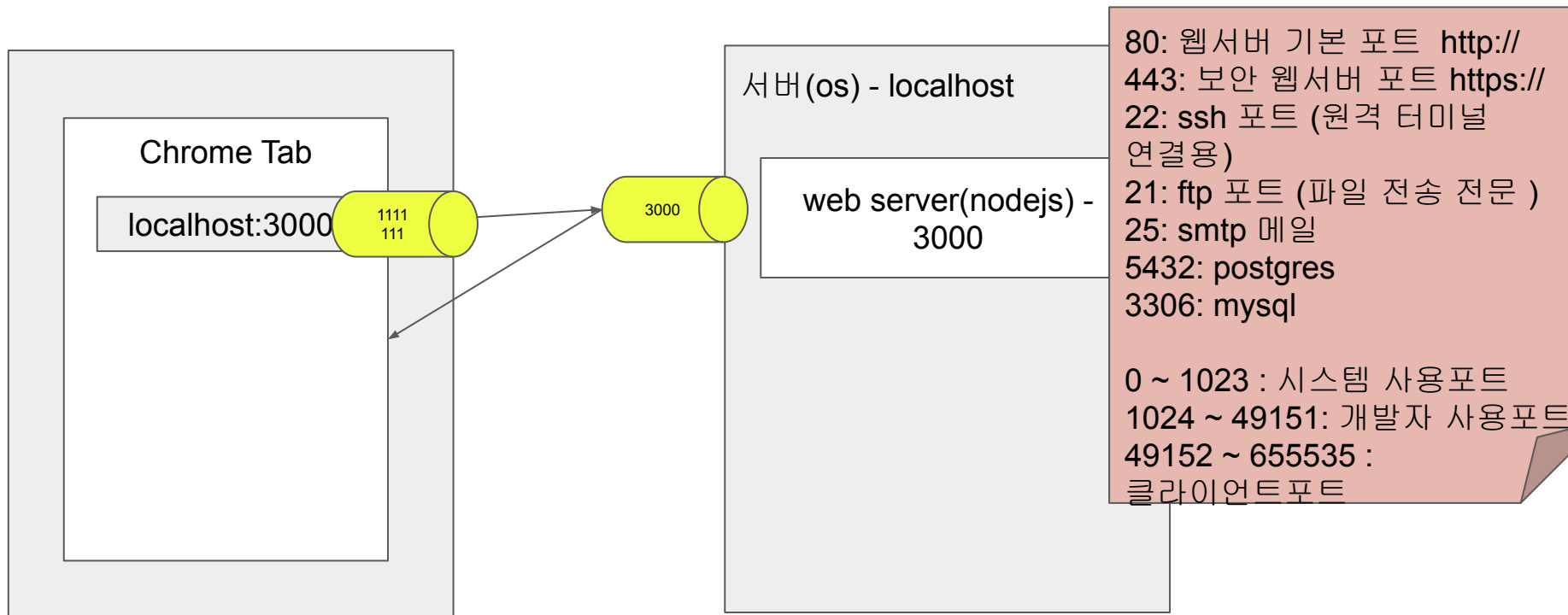
Postgresql @ 14, 16



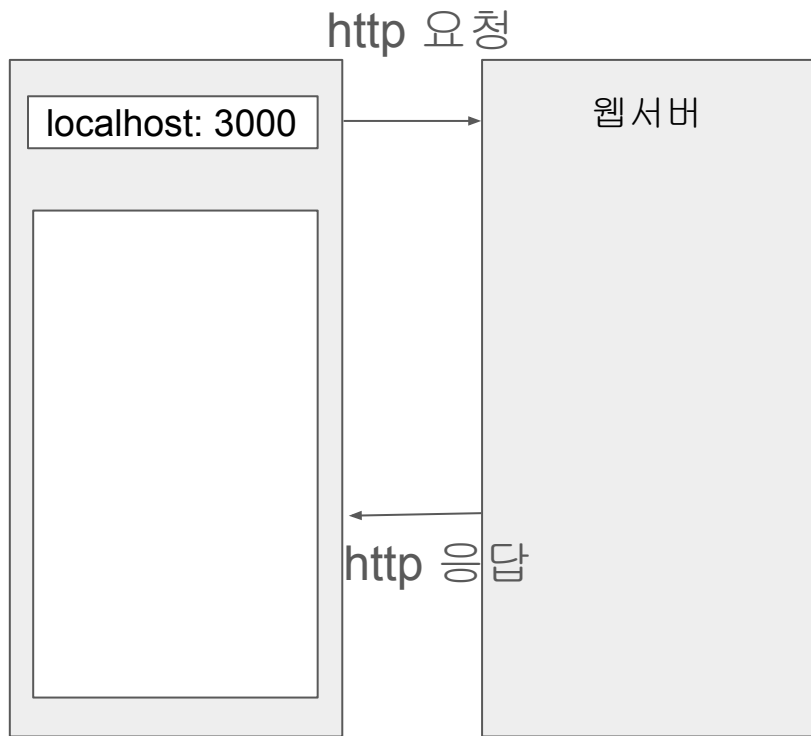
Web Server



포트 번호



HTTP 요청과 응답



http 요청

http 응답

항목	의미	예시
주소	무엇을 요청할지	주소?bn=1
헤더	부가정보	인증정보
바디	데이터를 실어보내는 곳	POST, PUT
항목	의미	예시
상태코드	성공 실패	
응답헤더	부가정보	Content Type
응답본문	결과데이터	html , json, text

응답 코드

- 200
 - 200 : 요청이 성공적으로 처리되었어요
 - 201 : 요청이 성공적으로 처리되고, 리소스가 생성됐어요
 - 204 : 요청은 성공했는데 너한테 줄 본문이 없어요.
- 300
 - 301 : 요청한 리소스가 영구적으로 다른데로 갔어요
- 400
 - 400 : 요청이 잘못되어 서버가 요청을 알아들수 없어요
 - 401 : 인증이 필요해요 (로그인 같은게 필요해요)
 - 404 : 요청한 리소스가 없어요
- 500
 - 500 : 인터널 서버에러 서버에거 처리중에 오류가 났어요 (try catch 안해서 발생)
-

URL 설명

<https://www.82cook.com:8080/entiz/enti.php?bn=15&bt=23&>

https → 프로토콜을 의미

www.82cook.com → 도메인

8080 → 포트번호

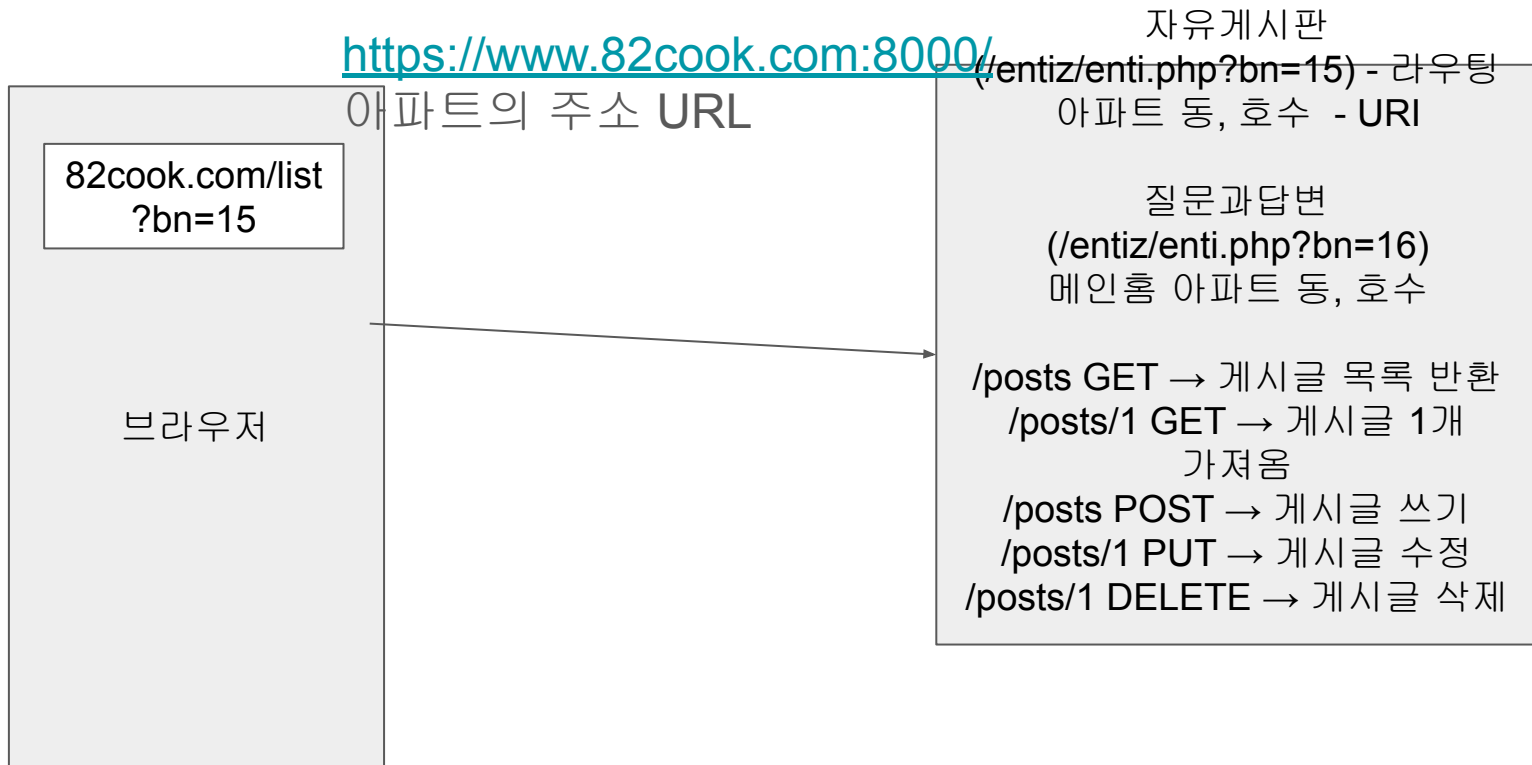
[/entiz/enti.php](https://www.82cook.com:8080/entiz/enti.php) → path , uri

? → 쿼리 파라미터를 시작한다는 의미

bn=15 → bn 키의 값은 15야 의미

URL 과 URI

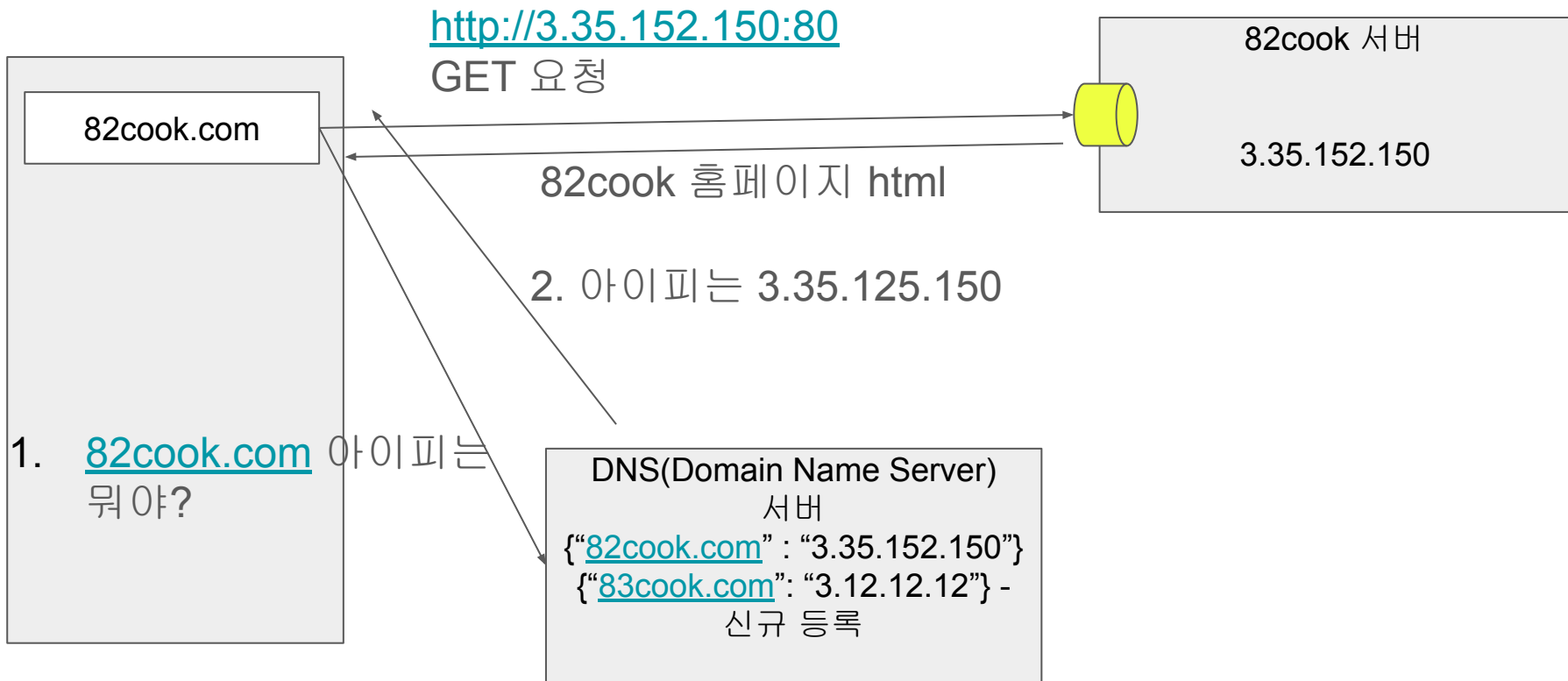
82cook web server (아파트)



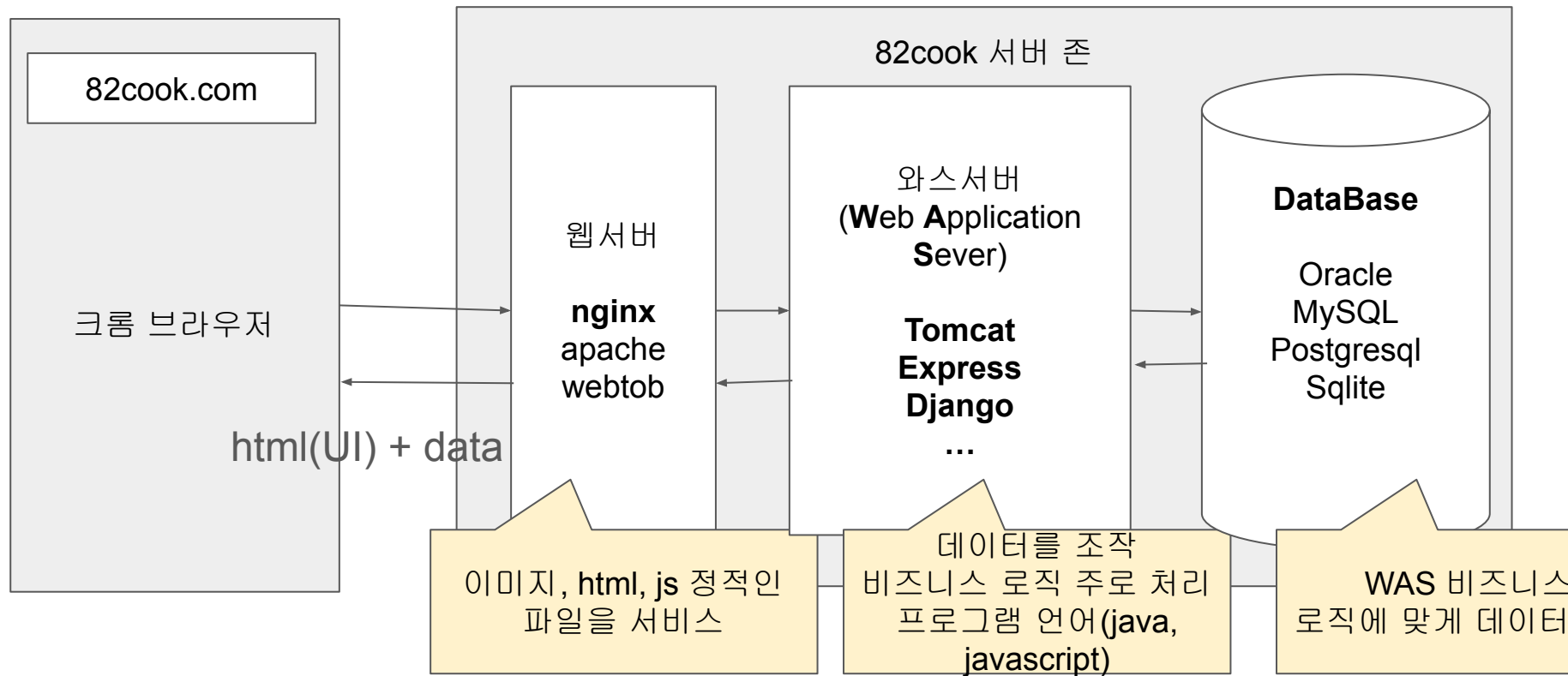
HTTP 메소드

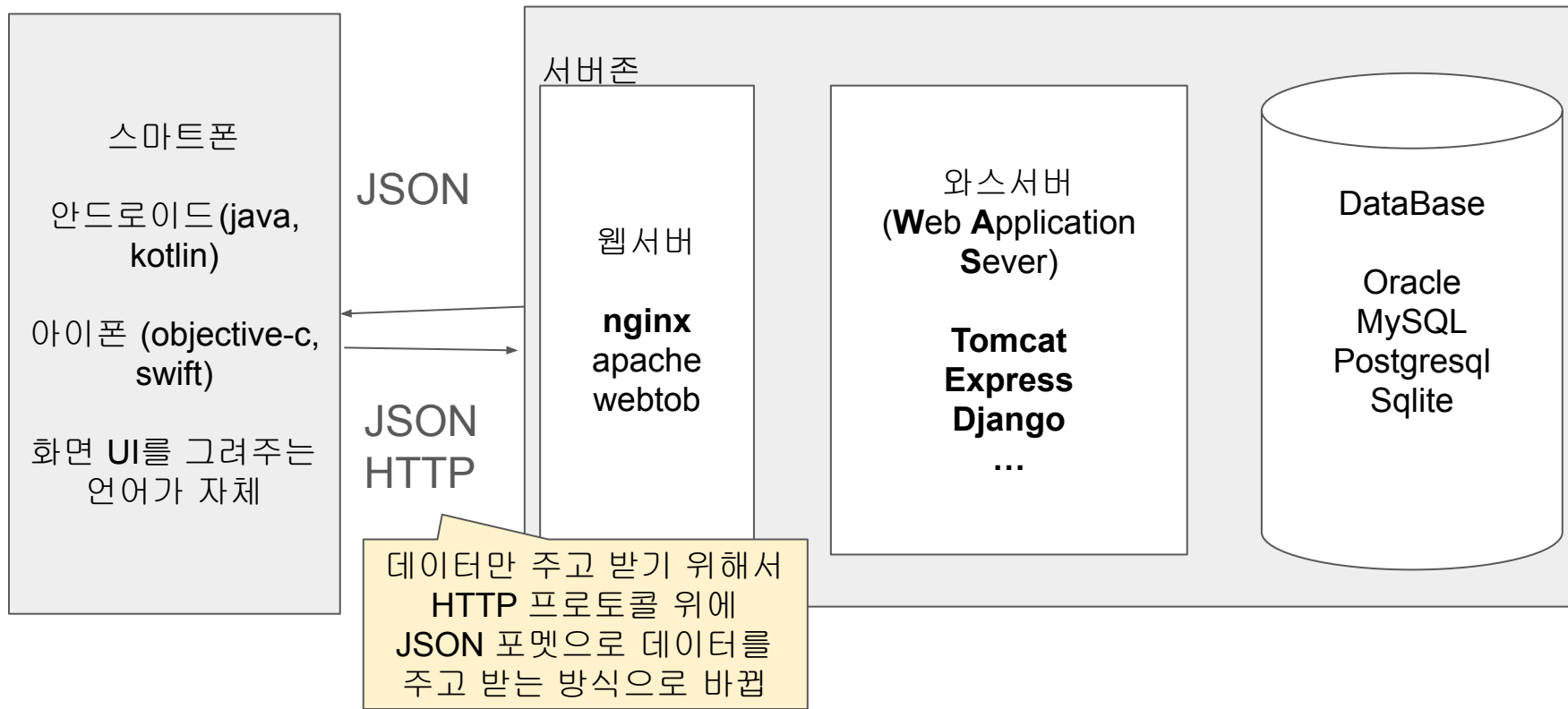
1. GET : 서버에 리소스를 요청합니다.
2. POST : 서버에 데이터를 전송할 때 사용합니다. 클라이언트 → 서버 , 게시글 쓰기
3. PUT : 기존 리소스를 수정할 때 사용합니다.
4. PATCH : 기존 리소스를 부분적으로 수정할 때 사용합니다.
5. DELETE : 기존 리소스를 삭제할 때 사용합니다.

DNS(Domain Name Server) 동작 방식



3티어 서버 구조

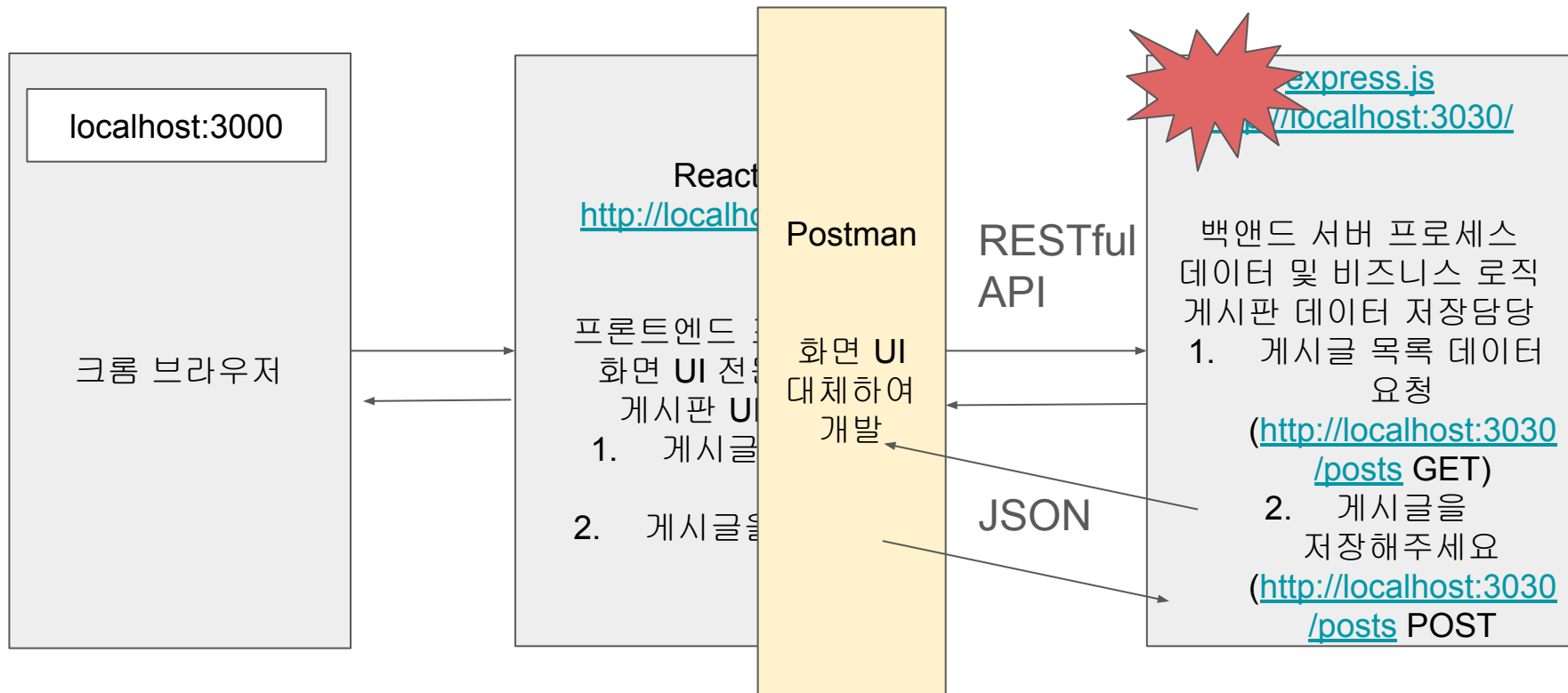




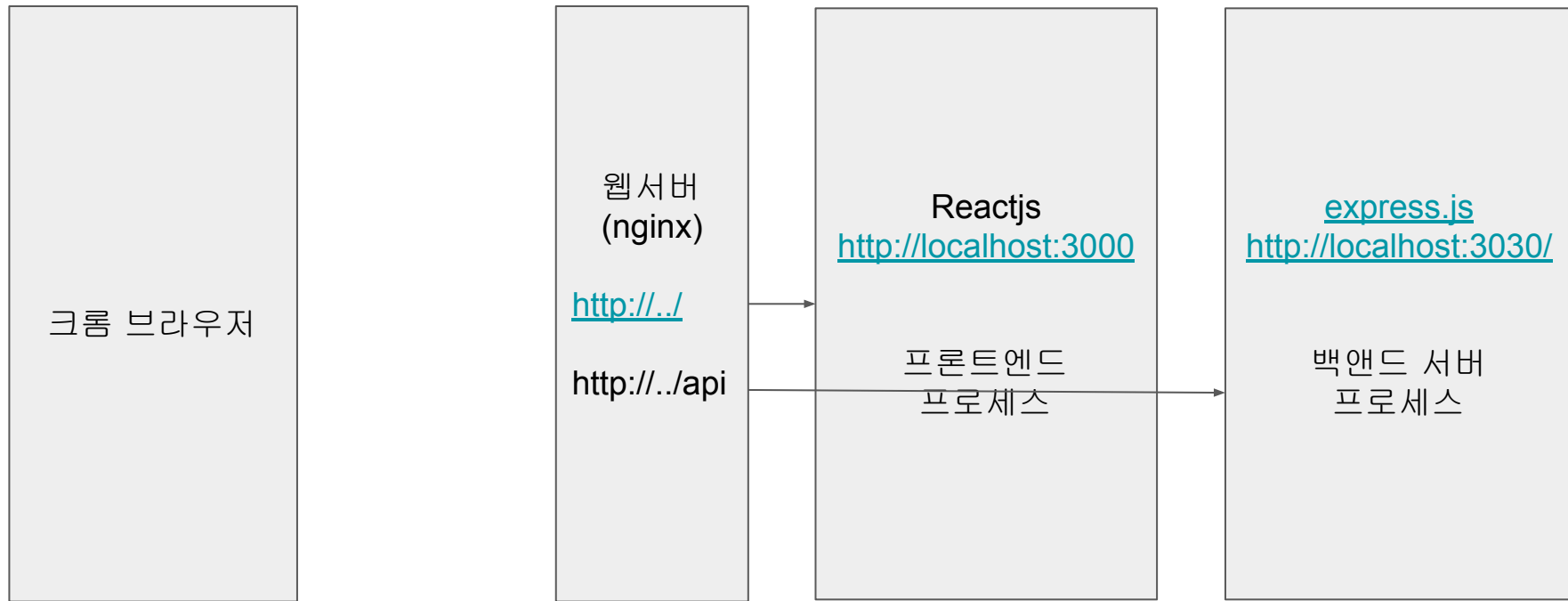
RESTful API

- 포맷 : JSON
- 전송방식 : HTTP
 - GET : 데이터를 읽을때 사용
 - POST : 데이터를 서버에 보낼 때
(클라이언트가, `postman/react` 소스)
 - PUT : 데이터를 수정할 때 사용
 - DELETE : 데이터를 지울 때 사용
- 게시글 목록 : GET /posts
- 게시글 상세 : GET /posts/12
- 게시글 쓰기 : POST /posts
- 게시글 삭제 : DELETE /posts/12

frontend + backend



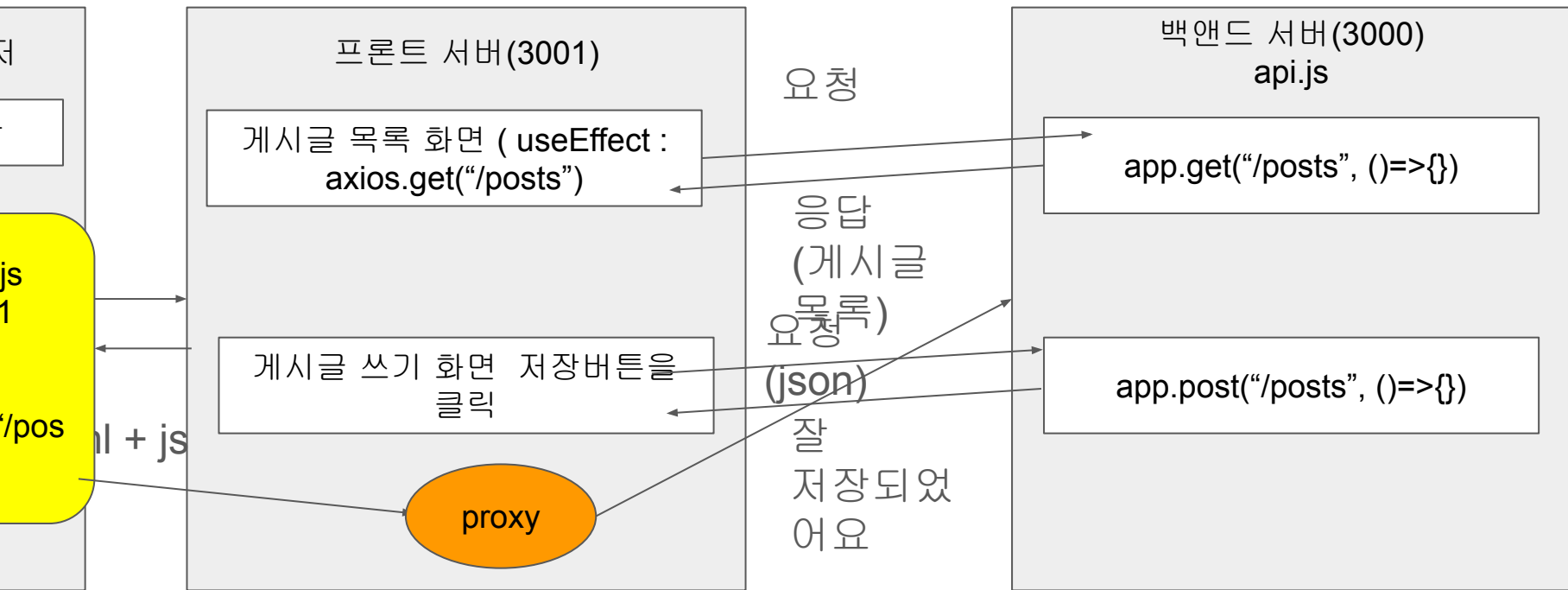
배포했을때(VM 직접배포)



파라미터 설명

<https://www.82cook.com:80/entiz/enti.php?bn=10>

- https → 프로토콜
- www.82cook.com → 도메인 주소 (IP)
- 80 → 82cook 의 서버 포트 번호
- /entiz/enti.php → path, uri
- ? 뒤에는 query → 서버에 간략한 데이터를 전송하기 위해서 사용 → 크기에 제한(255)
 - ?key1=value1&key2=value2&key3=value3 GET, POST, PUT, DELETE
- POST 요청(Request) : Body 곳에다 넣어서 보냅니다 → 게시글 본문 처럼 큰 사이즈, 첨부파일, 이미지파일



페이지 네이션

1번 게시물

2번 게시물

3번 게시물

4번 게시물

5번 게시물

6번 게시물

7번 게시물

8번 게시물

9번 게시물

10번 게시물

11번 게시물

1페이지

2페이지

3페이지

```
select * from posts limit 5 offset 0
```

$$(\text{page} - 1) * \text{limit} = (1 - 1) * 5 = 0$$

```
select * from posts limit 5 offset 5
```

$$(\text{page} - 1) * \text{limit} = (2 - 1) * 5 = 5$$

```
select * from posts limit 5 offset 10
```

ORM

- **Object** 를 사용하듯이(자바스크립트 프로그래밍 만으로) **SQL**을 대체하여 데이터베이스 데이터를 조작하게하는 도구(라이브러리)
- 자바스크립트 : **Sequelize, TypeORM, Prisma**
- 자바 : **JPA**, 하이버네이트, **MyBatis** - (**select** 문을 전부 별도의 **xml** 로 모아서 관리)
- 루비 : 액티브 레코드
- **ORM**도구를 사용해보면 내부적으로는 전부 **SQL** 변환되어 실행된다 .
- 프로그래머 입장에서 **SQL** 을 볼 수가 없다

Sequelize

```
db.prepare(`select * from posts`).all()
```

```
db.prepare(`select * from posts`).get()
```

```
db.prepare(`update posts set title = ?`  
).get(title)
```

```
const posts = Post.findAll();
```

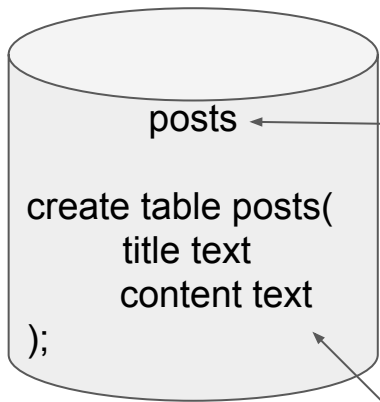
```
const post = Post.findOne();
```

```
const post = Post.update({title: title})
```

Sequelzie

```
db.prepare(`select * from posts`).all()
```

```
const posts = Post.findAll();
```



Post → sequelize 사용되는 모델명

```
const Post = sequelize.define('Post', {  
  title : text,  
  content : text  
})  
sequelize.sync();
```

```
create table posts(  
  title text  
  content text  
);
```

레이어 아키텍처

app.js

POST /posts -> app.post("/posts,()=>{})

GET /posts -> app.get("/posts,()=>>{})

POST /todos -> app.post("/todos,()=>>{})

GET /todos -> app.get("/todos,()=>>{})

GET /notes -> app.get("/notes,()=>>{})

POST /notes -> app.post("/notes,()=>>{})

3000줄

app.js

routes

controllers

라우터

(/posts)

라우터

(/todos)

라우터

(/notes)

컨트롤러

GET
POST
PUT
DELETE
(/posts)

컨트롤러

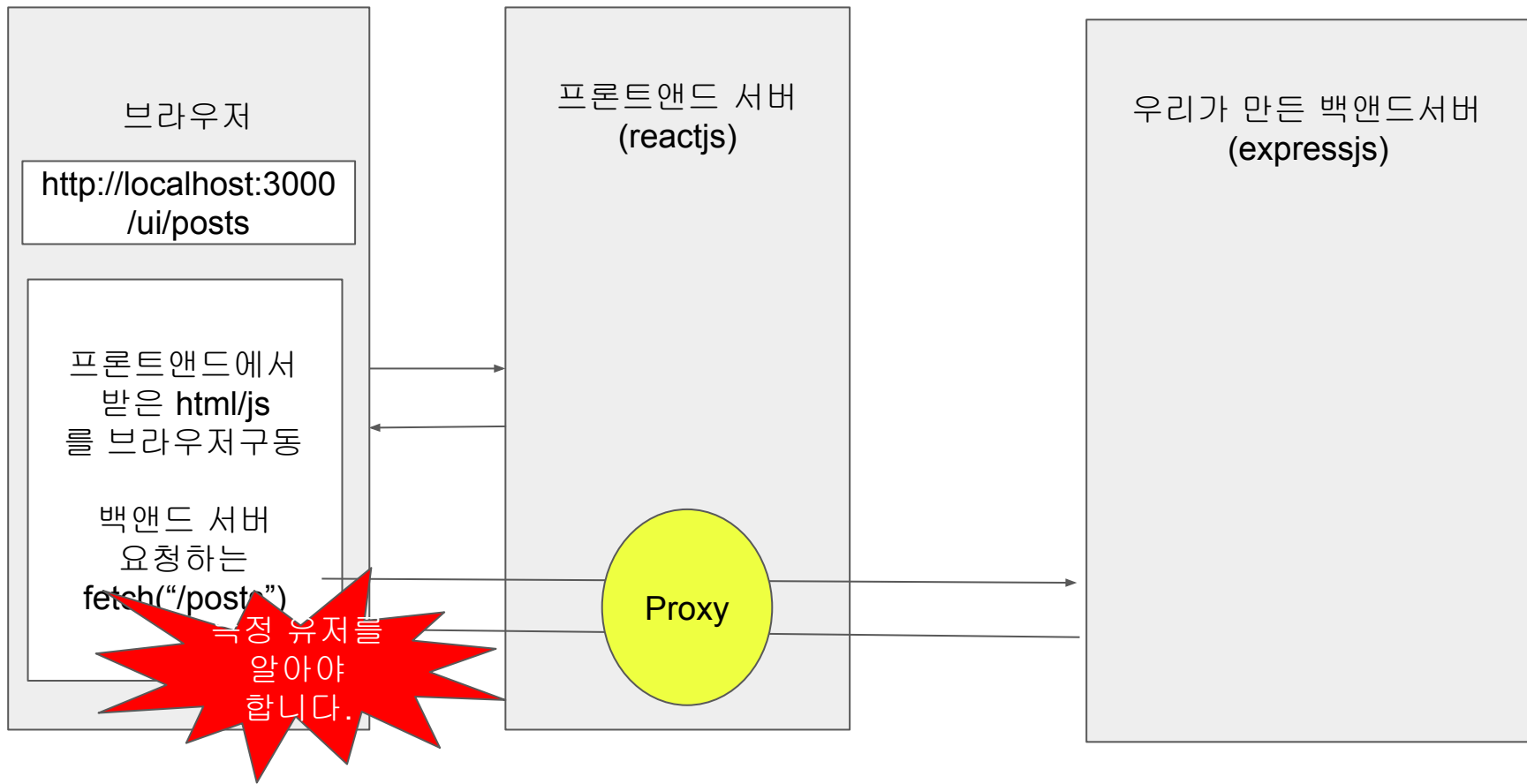
GET
POST
PUT
DELETE
(/todos)

컨트롤러

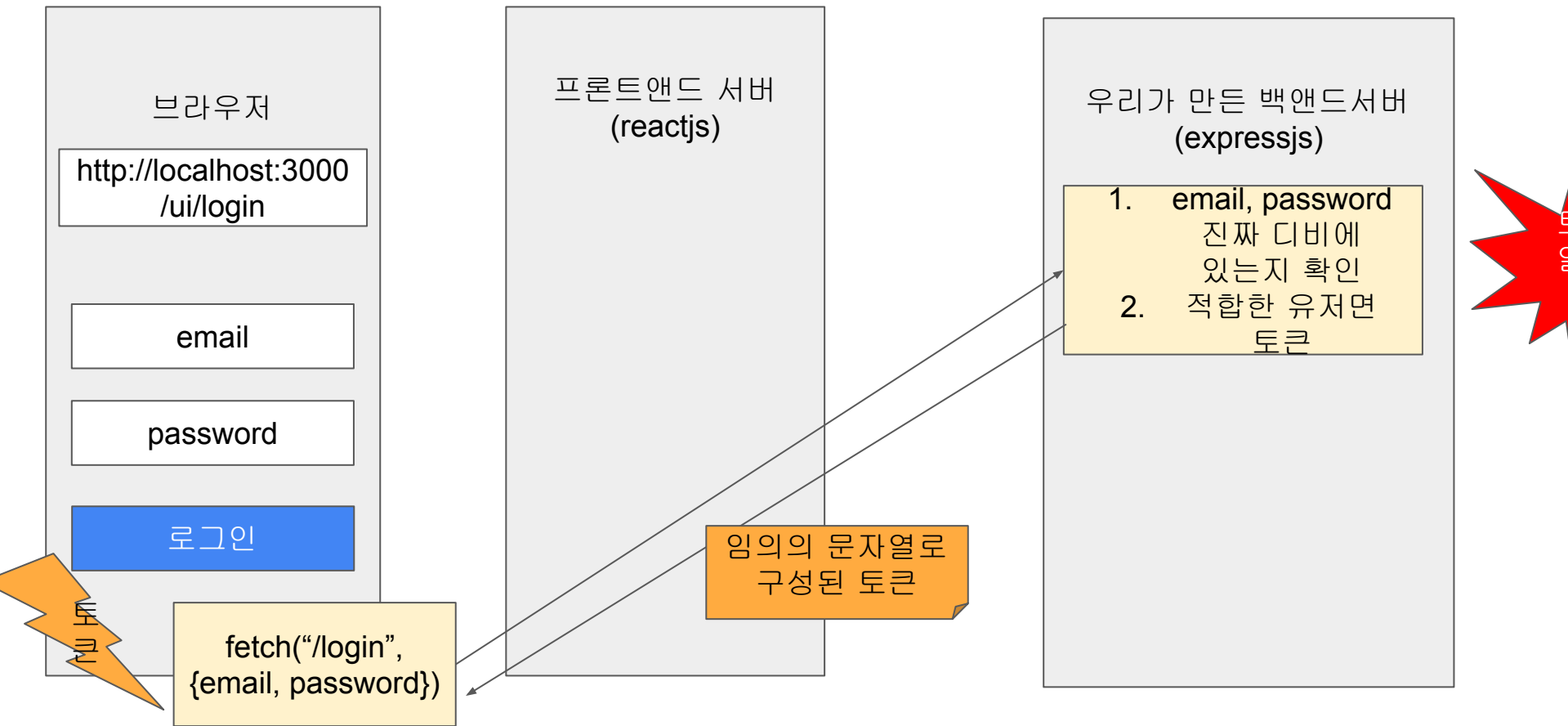
GET
POST
PUT
DELETE
(/notes)

models
/posts
/todos
notes

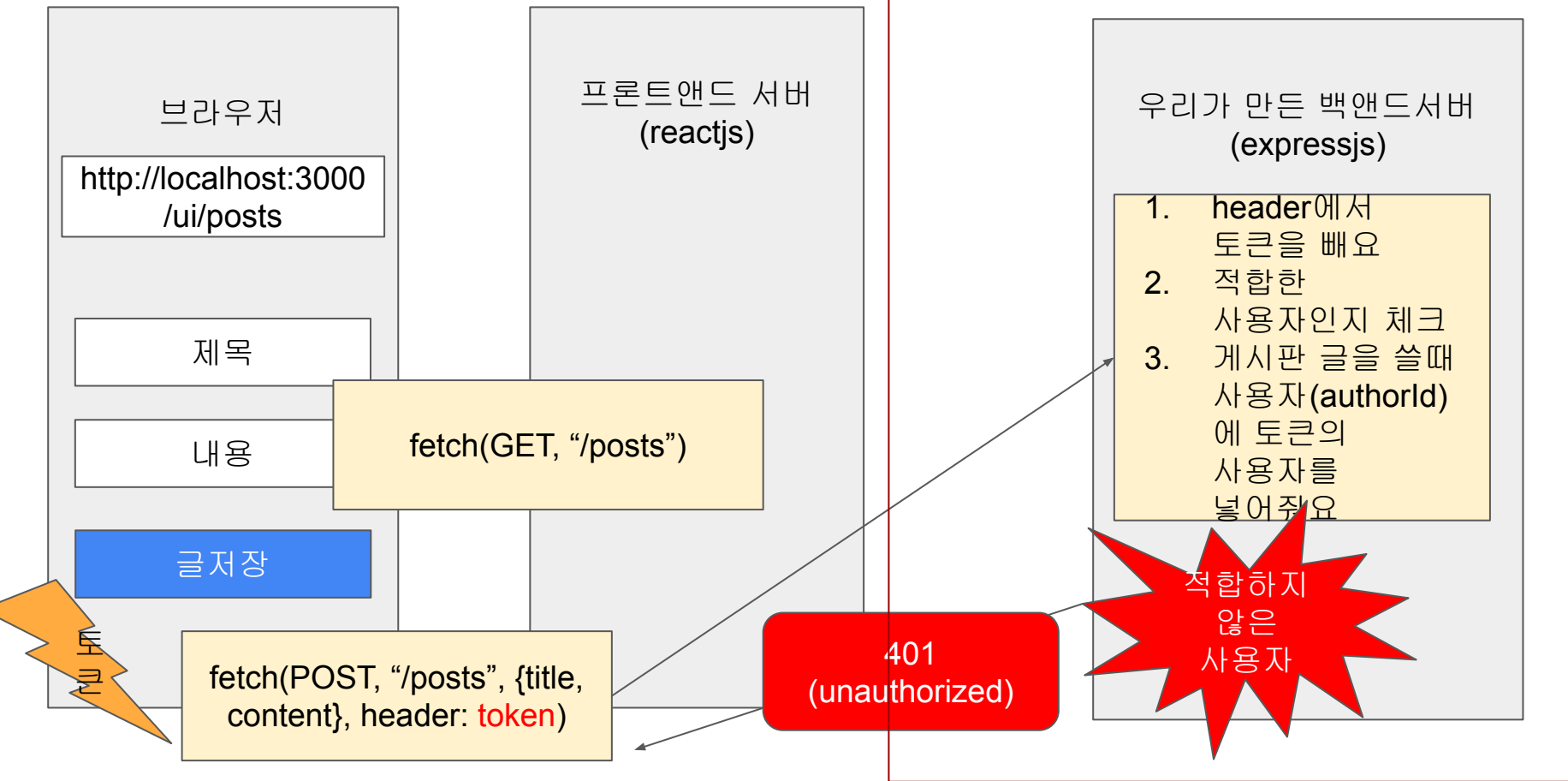
인증



로그인



게시판글쓸때 인증방식



토큰 (열쇠)

- **Access Token** : 보호된 자원 접근할 때 사용 (탈취되면 위험하기 때문에 보통 1시간)
 - 만료기간을 : 30 일, 60일 길게 줄꺼예요
- Refresh Token : Access Token 을 재발급 받기 위해서 사용되는 (14일)
 -