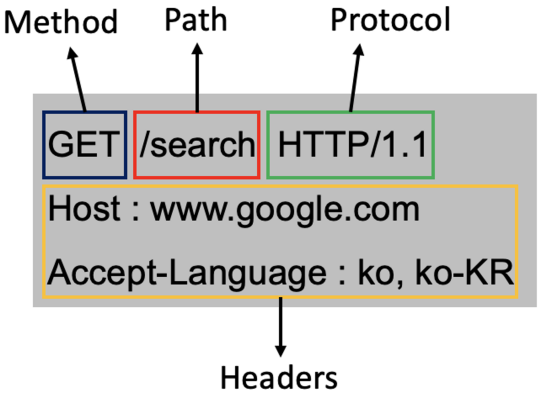
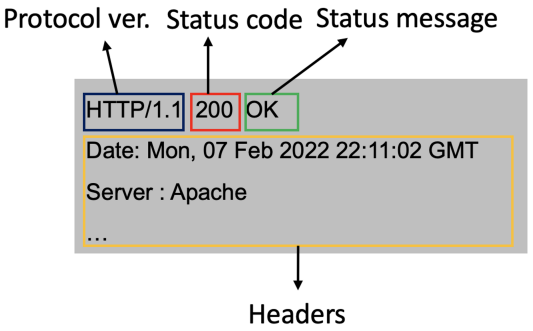
Http, Https, Rest API

💡 HyperText Transfer Protocol의 약자로, 인터넷 상에서 클라이언트와 서버가 자원을 주고 받을 때 쓰는 통신 프로토콜(규약)

- request/response 구조로 웹 상에서 정보를 주고 받음
- TCP/IP 기반으로 작동
- request message
  - start line (method, path, HTTP, version)
  - headers
  - body
- response message
  - status line(HTTP version, status code, status message)
  - headers
  - body



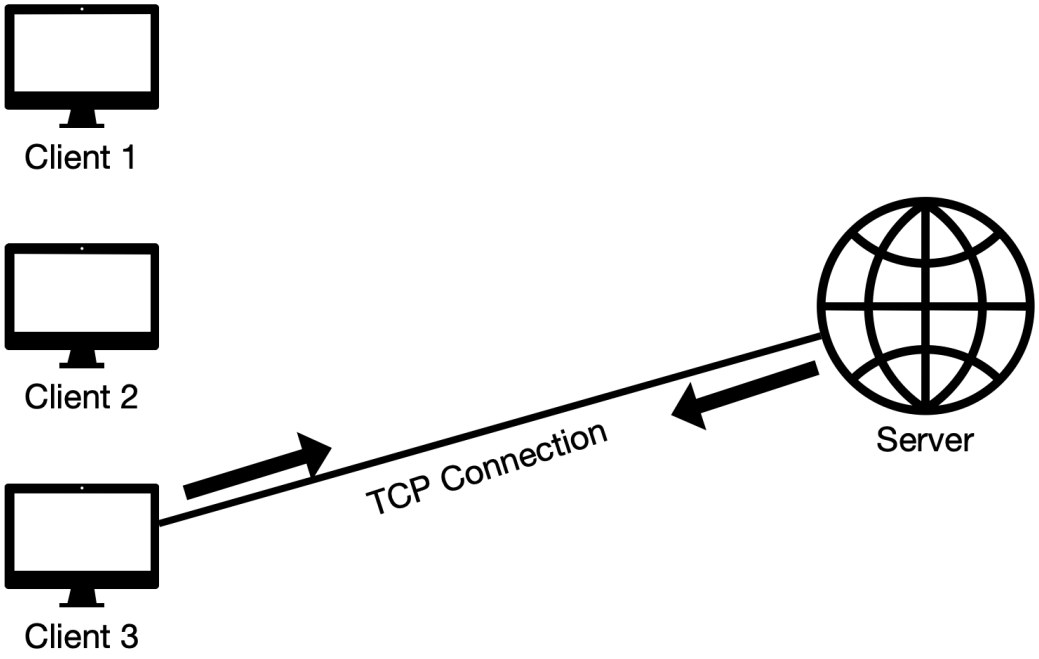


💡 HTTP의 특징

📌 Connectionless

- TCP/IP의 경우 기본적으로 연결을 종료하지 않으면 그 연결은 종료되지 않고 유지

    - 자원 낭비 발생 가능성 ⬆️
- HTTP 모델은 각 클라이언트가 서버에 요청을 하고 응답을 받으면 바로 연결을 끊어버리는 Connectionless 특성을 가짐
    - 동시 접속을 최소화하여 더 많은 요청 처리 가능
    - 하지만 클라이언트의 이전 상태(로그인 유무 등)을 알 수 없는 Stateless 특성 발생



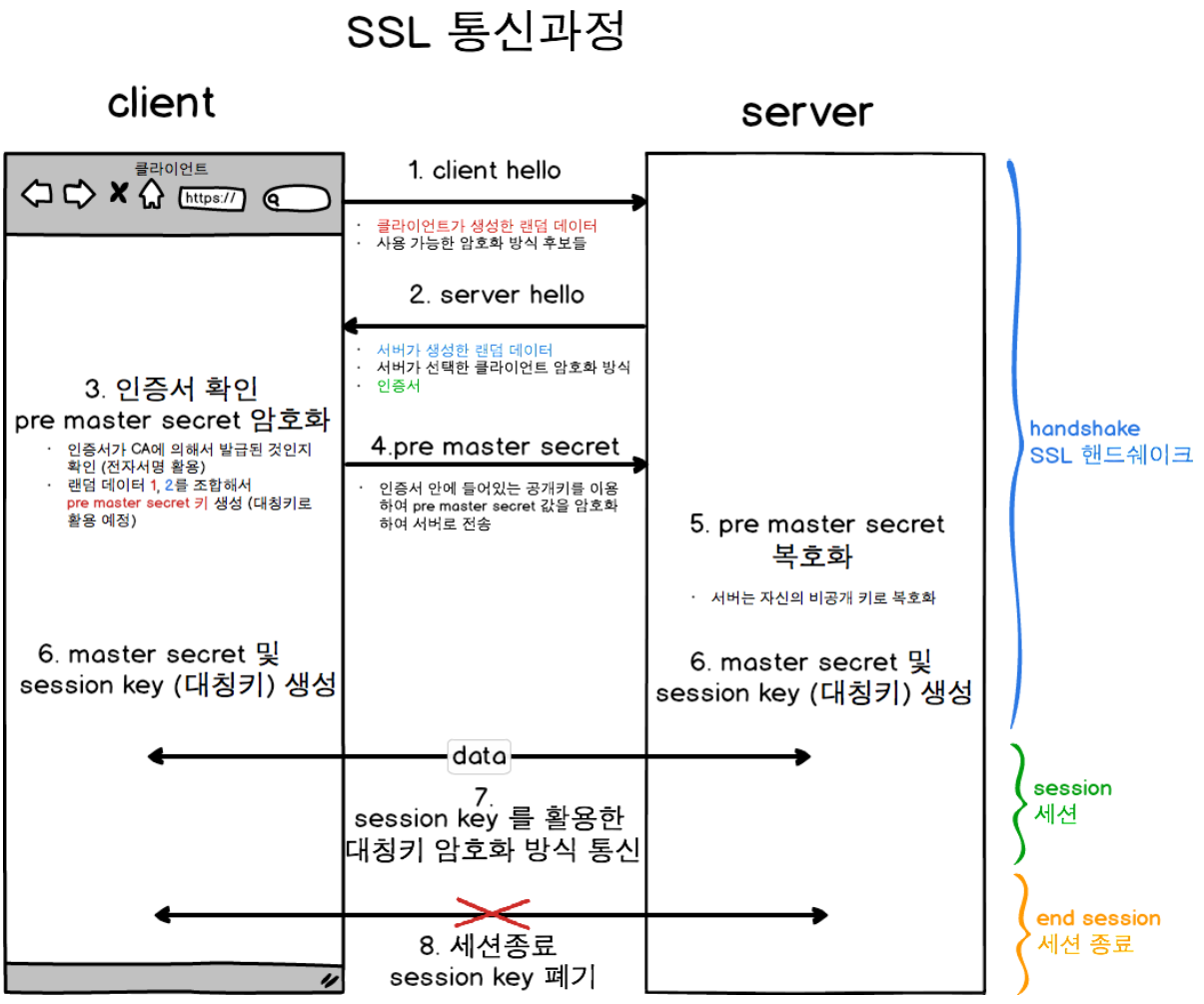
📌 Stateless

- 통신이 끝나면 더 이상 상태를 유지하지 않음
- 발생할 수 있는 문제
    - 사이트에 로그인 시, 이전의 로그인이 유지되지 않아 매번 로그인을 해야함
    - 팝업 보지 않기를 체크했는데, 페이지를 유지할 때마다 매번 팝업창이 뜸
- Connectionless, Stateless 특성을 가진 HTTP의 단점을 해결하기 위해 cookie, session, jwt 등을 도입

💡 HTTPS

- HyperText Transfer Protocol Secure

- HTTP는 text 교환이므로, 누군가 네트워크에서 인터셉트할 경우 데이터 유출이 발생할 수 있는 보안 이슈 존재 ➡️ 이러한 문제를 해결하기 위해 HTTP에 암호화를 추가한 HTTPS 사용
- SSL(보안 소켓 계층)이나 TSL(전송 계층 보안) 프로토콜을 사용하여 데이터를 암호화
  - TLS는 데이터 무결성을 제공하기 때문에 데이터가 전송 중에 수정, 손상되는 것을 방지
- 초기에 클라이언트와 서버가 통신을 하며 암호화 키를 서로 안전하게 주고받는다. (SSL Handshake)
- 이 때 암호화 키값이 노출되지 않도록 안전하게 해주는 것이 https 서버 인증서



💡 HTTP request method - GET, POST 차이

- GET: 클라이언트가 서버에서 리소스를 요청할 때 사용하는 메소드
- POST: 서버에게 데이터 처리(주로 생성)을 요청할 때 사용하는 메소드
- GET
  - 필요한 정보를 특정하기 위해 URL 뒤에 Query String을 추가하여 정보를 조회
  - Query String: URL 주소 끝에 key-value 쌍으로 parameter를 포함
  - URL에 데이터를 포함해서 전달하므로 전송하는 길이에 제한이 있음
  - 브라우저 히스토리에 남고 캐시가 가능
    - 한 번 서버에 GET 요청을 한 적이 있다면 브라우저가 그 결과를 저장하고 이후 동일한 요청 처리 시 브라우저에 저장된 값 사용

```
https://www.google.com/search?q=get
```

- POST
  - 전달할 데이터를 Body 부분에 포함하여 통신
  - 요청 Body는 길이에 제한이 없기 때문에 대용량 데이터 전송 가능
  - 요청 Header의 Content-Type에 요청 데이터의 타입을 표시해야 함
  - 브라우저 히스토리에 남지 않고 캐시 불가능
- PUT: 리소스를 대체, 해당 리소스가 없으면 생성
- PATCH: 리소스의 일부분을 수정

💡 REST API

- REST를 기반으로 만들어진 API

REST(Representational State Transfer)

- 자원을 이름으로 구분하고 해당 자원의 상태를 주고 받는 모든 것
- HTTP URI(Uniform Resource Identifier)를 통해 자원(Resource)을 명시하고
- HTTP Method(POST, GET, PUT, DELETE, PATCH 등)를 통해
- 해당 자원(URI)에 대한 CRUD Operation을 적용하는 것을 의미
- CRUD Operation

> Create : 데이터 생성(POST)
>
> Read : 데이터 조회(GET)
>
> Update : 데이터 수정(PUT, PATCH)
>
> Delete : 데이터 삭제(DELETE)

- REST의 구성 요소
  1. **자원(Resource) : HTTP URI**
  2. **자원에 대한 행위(Verb) : HTTP Method**
  3. **자원에 대한 행위의 내용 (Representations) : HTTP Message Pay Load**
- REST의 장단점
  - **장점**
    - HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라를 구출할 필요가 없다.
    - HTTP 프로토콜의 표준을 최대한 활용하여 여러 추가적인 장점을 함께 가져갈 수 있게 해 준다.
    - HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용이 가능하다.
    - Hypermedia API의 기본을 충실히 지키면서 범용성을 보장한다.
    - REST API 메시지가 의도하는 바를 명확하게 나타내므로 의도하는 바를 쉽게 파악할 수 있다.
    - 여러 가지 서비스 디자인에서 생길 수 있는 문제를 최소화한다.
    - 서버와 클라이언트의 역할을 명확하게 분리한다.
  - **단점**
    - 표준이 자체가 존재하지 않아 정의가 필요하다.
    - HTTP Method 형태가 제한적이다.
    - 브라우저를 통해 테스트할 일이 많은 서비스라면 쉽게 고칠 수 있는 URL보다 Header 정보의 값을 처리해야 하므로 전문성이 요구된다.
    - 구형 브라우저에서 호환이 되지 않아 지원해주지 못하는 동작이 많다.(익스폴로어)