주제

Github

코드를 저장하고 공유하고, 협업하는 서비스.

버전관리 가능 -> 롤백 가능.
협업 -> 오픈 프로젝트에 기여가능.

```
Repository   - 프로젝트 하나.
Commit       - 변경 사항 저장.
Branch       - 코드 분기.
Merge        - 분기 합치기.
Pull Request - 코드 합치기 요청
Issue        - 버그, 할 일 관리.
Push         - 변경 사항 Github에 적용
Pull         - 내 컴퓨터로 Github에 있는 코드 가져오기.
```

분기를 하는 이유 -> 미완성 코드가 main 코드에 섞이면 이슈가 생길 수 있음.

Server

Client - 요청 하는 쪽
Server - 응답 하는 쪽
웹서버 -> 웹 페이지를 클라이언트에 전달해주는 서버.
애플리케이션 서버 -> API 서버 -> 로그인, 계산, 알고리즘, AI 등을 해서 결과를 반환하는 서버. 웹, 모바일 앱과 통신. -> 화면은 없고 데이터만 반환.
데이터베이스 서버 -> 데이터를 저장 및 조회 (SQL, NoSQL)
파일서버: FTP서버, NAS -> 파일 업로드, 다운로드 -> 내 컴퓨터의 어떤 폴더의 접근 권한을 열어서 웹상에서 접근 가능하게 하는 서버.
캐시 서버 -> 빠른 응답을 위한 임시저장소.
인증 서버 -> 로그인, 토큰 발급, 권환 관리.
게임 서버 -> 유저 상태 동기화, 실시간성 중요 -> 웹소켓
메일 서버 -> 이메일 송수신 하는 서버.
DNS 서버 -> 도메인 주소 -> IP 주소 변환. : google dns, cloudflare DNS
프록시 서버 / 로드 밸런서 -> 트래핑 분산, 중계
스트리밍 서버 -> 영상 음성 실시간 송수신.

소켓 -> tcp/udp 방식이 있고 tcp는 신뢰성이 있는, udp는 확실히 메시지가 갔는지 확인을 안하는.
웹소켓 -> 웹상에서 소켓.

클라우드 플레어.

캐시 - 이미 로드된 무언가를 다시 어렵게 로드 하지 않고 캐시라는 구조에 저장해 두어 다시 요청이 있을 때 꺼내 쓰는.
HTTP
HTTPS

SQL -> 표 -> 트리
JSON -> NoSQL
dict같은 자료구조

```
{
    "▭":23,
    23:true
}
```

AWS = Amazon Web Services
클라우드 컴퓨팅 서비스

원래는 서버, 저장소, 네트워크를 직접 사서
→ 서버실에 두고
→ 전기·관리·보안까지 다 직접

클라우드는 이걸 인터넷으로 빌려 쓰는 것이야.

- 서버 컴퓨터
- 데이터베이스
- 파일 저장 공간
- 네트워크
- AI / 빅데이터 / 보안 등

✅ 장점

- **확장성**: 트래픽 폭증해도 자동 확장
- **신뢰성**: 장애 나도 다른 리전에 백업
- **속도**: 전 세계 데이터센터
- **생태계**: 거의 없는 게 없음

❌ 단점

- 구조 복잡
- 잘못 쓰면 **요금 폭탄**

- 러닝 커브 있음

Service
EC2 -> 가상 리눅스 서버
S3 -> 파일 저장소.
RDS / DynamoDB
->DB
VPC 가상 네트워크 -> 라우팅, 방화벽, 서브넷 설정 가능.
Lambda -> serverless -> 서버 없이 preload된 html js css등만 사용.