6단원. 응용 계층: HTTP와 DNS (최빈출 단원)

> 대상: 전공자 · 정보처리기사 취득자 | 목적: 인프라 직무 기술 면접 대비 중요도: ★★★ 최빈출 · ★★ 자주 출제 · ★ 기본 개념

Q1. HTTP의 특징(무상태성, 비연결성)과 그 극복 방법을 설명하세요. ★★★

답안 HTTP는 요청-응답 구조의 텍스트 기반(HTTP/1.x) 프로토콜로, **무상태(stateless)** — 서버가 이전 요청을 기억하지 않음 — 가 핵심 특징입니다. 덕분에 서버 확장(어느 서버가 받아도 됨)이 쉽지만, 로그인 유지 같은 상태가 필요한 기능은 **쿠키/세션/토큰**으로 해결합니다. 비연결성(요청마다 연결 종료)은 HTTP/1.1의 **keep-alive**(연결 재사용)로 개선되었습니다. 무상태성이 로드밸런싱·오토스케일링을 가능하게 하는 설계 기반이라는 점을 인프라 관점에서 연결해 답하면 좋습니다.

관련 개념 무상태와 수평 확장, 쿠키/세션/JWT, keep-alive, 멱등성

Q2. 쿠키와 세션의 차이, 그리고 JWT 토큰 방식과의 비교를 설명하세요. ★★★

답안 쿠키는 서버가 Set-Cookie로 내려주고 브라우저가 저장해 매 요청에 자동 첨부하는 클라이언트 측 데이터입니다. **세션**은 상태를 서버(메모리/Redis)에 저장하고 클라이언트에는 세션 ID만 쿠키로 주는 방식으로, 민감 정보가 서버에 있어 안전하고 즉시 무효화가 가능하지만, 서버가 여러 대면 세션 공유(Redis 등 중앙 저장소)가 필요합니다. **JWT**는 서명된 토큰 자체에 사용자 정보를 담아 서버가 상태를 저장하지 않는 방식으로, 확장에 유리하고 MSA에 적합하지만 발급 후 강제 만료가 어렵습니다(블랙리스트나 짧은 만료+리프레시 토큰으로 보완). 보안 속성(HttpOnly, Secure, SameSite)도 함께 언급하면 좋습니다.

관련 개념 세션 스토어(Redis), 스티키 세션, JWT 서명·만료 전략, HttpOnly/SameSite

Q3. HTTP 메서드와 멱등성, 안전성을 설명하세요. ★★★

답안 주요 메서드는 GET(조회), POST(생성/처리), PUT(전체 교체), PATCH(부분 수정), DELETE(삭제), HEAD(헤더만), OPTIONS(지원 메서드 확인, CORS 프리플라이트)입니다. **안전(safe)**은 서버 상태를 바꾸지 않는 성질(GET, HEAD), **멱등(idempotent)**은 여러 번 실행해도 결과가 같은 성질(GET, PUT, DELETE는 멱등, POST는 아님)입니다. 멱등성이 중요한 이유는 **재시도 안전성** 때문입니다. 네트워크 오류 시 멱등 요청은 안심하고 재시도할 수 있지만 POST 재시도는 중복 결제 같은 사고가 되므로, 실무에서는 멱등성 키(Idempotency-Key)로 보완합니다. 게이트웨이·프록시의 자동 재시도 정책도 멱등 메서드에만 적용하는 것이 원칙입니다.

관련 개념 safe/idempotent 분류표, 멱등성 키, 재시도 정책, REST 설계

Q4. 주요 HTTP 상태 코드를 계열별로 설명하세요. ★★★

답안 2xx 성공: 200 OK, 201 Created, 204 No Content. **3xx 리다이렉션**: 301(영구 이동), 302(임시 이동), 304(Not Modified, 캐시 유효). **4xx 클라이언트 오류**: 400(잘못된 요청), 401(인증 필요), 403(권한 없음 — 401과의 차이가 단골 질문), 404(없음), 429(요청 과다, 레이트 리밋). **5xx 서버 오류**: 500(내부 오류), 502(Bad Gateway — 프록시가 업스트림에서 잘못된 응답 수신), 503(서비스 불가 — 과부하·점검), 504(Gateway Timeout — 업스트림 응답 시간 초과). 인프라 직무에서는 **502/503/504의 구분**이 중요합니다. 502는 백엔드가 죽었거나 연결 거부, 504는 백엔드가 느림, 503은 LB가 보낼 곳이 없거나 서버가 의도적 거부 — 로드밸런서 뒤 장애 진단의 첫 갈림길입니다.

관련 개념 401 vs 403, 301 vs 302, 502/503/504 진단, 304와 조건부 요청

Q5. HTTP/1.1, HTTP/2, HTTP/3의 차이를 설명하세요. ★★★

답안 HTTP/1.1은 텍스트 기반이며 연결당 한 번에 하나의 요청-응답만 처리해(파이프라이닝은 실패), 브라우저가 도메인당 6개 연결을 열어 병렬화합니다. **HTTP/2**는 바이너리 프레이밍으로 **하나의 연결에 여러 스트림을 다중화**하고, 헤더 압축(HPACK), 우선순위를 지원해 연결 수를 줄이고 성능을 높였습니다. 하지만 TCP 위라서 패킷 손실 시 모든 스트림이 함께 멈추는 **TCP HOL 블로킹**이 남습니다. **HTTP/3**은 전송을 UDP 기반 **QUIC**으로 바꿔 스트림별 독립 전달로 HOL을 해소하고, TLS 1.3 내장, 0-RTT 재연결, 연결 마이그레이션을 제공합니다. 요약: 1.1=연결 병렬화, 2=스트림 다중화, 3=전송 계층 교체.

관련 개념 멀티플렉싱, HPACK/QPACK, TCP HOL 블로킹, QUIC, Alt-Svc

Q6. DNS의 계층 구조와 이름 해석(resolution) 과정을 설명하세요. ★★★

답안 DNS는 루트 → TLD(.com) → 권한 네임서버(example.com)의 계층 구조입니다. 클라이언트가 **재귀 리졸버**(ISP, 8.8.8.8 등)에 질의하면, 리졸버가 **반복 질의**로 ① 루트 서버에 물어 .com TLD 서버 주소를 받고 ② TLD 서버에 물어 example.com의 권한 서버 주소를 받고 ③ 권한 서버에서 최종 A 레코드를 받아 클라이언트에 응답합니다. 각 단계 결과는 **TTL 동안 캐시**되어 대부분의 질의는 캐시에서 끝납니다. 클라이언트 관점 순서는 브라우저 캐시 → OS 캐시/hosts 파일 → 리졸버입니다. TTL은 변경 전파 속도와 질의 부하의 트레이드오프로, 서버 이전 전에 TTL을 미리 낮추는 것이 실무 관행입니다.

관련 개념 재귀 vs 반복 질의, 루트/TLD/권한 서버, TTL과 캐시, dig +trace

Q7. 주요 DNS 레코드 타입(A, AAAA, CNAME, MX, TXT, NS, SOA)을 설명하세요. ★★

답안 A는 도메인→IPv4, **AAAA**는 도메인→IPv6 매핑입니다. **CNAME**은 도메인의 별칭으로 다른 도메인을 가리키며(www→example.com), 루트 도메인(apex)에는 표준상 쓸 수 없어 클라우드 LB 연결 시 ALIAS/ANAME 레코드가 등장했습니다. **MX**는 메일 수신 서버, **TXT**는 임의 텍스트로 SPF/DKIM/DMARC(메일 인증)와 도메인 소유 확인에 쓰입니다. **NS**는 그 존을 관리하는 네임서버, **SOA**는 존의 기본 정보(시리얼, 갱신 주기)입니다. 실무에서는 "CNAME과 A의 차이", "apex에 CNAME이 안 되는 이유"가 자주 나옵니다.

관련 개념 apex와 ALIAS, SPF/DKIM/DMARC, 존 파일, PTR(역방향 조회)

Q8. DNS는 왜 UDP를 쓰고, 언제 TCP로 전환하나요? ★★

답안 일반 질의는 작은 요청-응답 한 쌍이라 연결 수립 오버헤드가 없는 **UDP 53**이 효율적입니다. **TCP 53**을 쓰는 경우는 ① 응답이 UDP 크기 제한을 넘어 **잘림(TC 비트)** 표시가 왔을 때 재시도(전통 512바이트, EDNS0으로 확장 가능), ② **존 전송**(AXFR, 주-보조 서버 간 존 복제), ③ 최근의 DNS over TLS(853)입니다. 방화벽에서 "DNS는 UDP만 열면 된다"고 TCP 53을 막으면 큰 응답(DNSSEC 등)과 존 전송이 실패하는 장애가 생기므로 둘 다 열어야 한다는 것이 실무 포인트입니다.

관련 개념 TC 비트, EDNS0, AXFR/IXFR, DoT/DoH

Q9. GSLB와 DNS 기반 로드밸런싱을 설명하세요. (인프라 연계) ★★

답안 DNS 라운드 로빈은 하나의 도메인에 여러 A 레코드를 등록해 응답 순서를 바꿔가며 분산하는 가장 단순한 방법이지만, 서버 장애를 감지하지 못하고 캐시 때문에 제어가 부정확합니다. **GSLB**는 DNS 응답을 지능화한 것으로, **헬스체크**로 살아있는 사이트만 응답하고, 지리적 근접성·지연시간·가중치 기반으로 **데이터센터/리전 단위 트래픽을 분배**합니다. 재해 복구(DR) 시 DNS 응답을 백업 사이트로 바꿔 페일오버하는 것이 대표 활용입니다. 한계는 TTL 캐시로 인한 전환 지연이며, 그래서 TTL을 짧게(30~300초) 둡니다. AWS Route 53의 라우팅 정책(지연시간, 지역, 페일오버)이 GSLB의 구현입니다.

관련 개념 헬스체크, 페일오버와 TTL, Route 53 라우팅 정책, 애니캐스트와의 비교

Q10. REST API란 무엇이고 설계 원칙은 무엇인가요? ★★

답안 REST는 HTTP의 의미론을 활용한 아키텍처 스타일로, **자원을 URI로 표현**하고(/users/123), **행위를 HTTP 메서드로** 표현하며(GET 조회, POST 생성, PUT/PATCH 수정, DELETE 삭제), 표현(주로 JSON)을 주고받습니다. 원칙은 무상태성(요청에 필요한 정보 포함), 균일한 인터페이스, 캐시 가능성, 계층화입니다. 설계 관행으로 명사형 복수 URI, 적절한 상태 코드 사용, 버저닝(/v1), 페이지네이션, HATEOAS(엄밀한 REST)가 있습니다. 대안 기술로 GraphQL(클라이언트가 필요한 필드 지정, 오버페칭 해소)과 gRPC(HTTP/2+프로토버프, 내부 서비스 간 고성능 통신)를 비교해 답할 수 있으면 좋습니다.

관련 개념 자원/표현, 상태 코드 활용, GraphQL/gRPC 비교, API 게이트웨이

Q11. 웹 캐싱은 어떻게 동작하나요? (Cache-Control, ETag) ★★

답안 HTTP 캐싱은 응답 헤더로 제어합니다. **Cache-Control: max-age=3600**은 브라우저·프록시가 1시간 동안 재검증 없이 캐시를 쓰게 하고, no-cache는 매번 재검증, no-store는 저장 금지, private/public은 공유 캐시(CDN) 저장 여부를 정합니다. 만료 후에는 **조건부 요청**으로 재검증합니다. 서버가 준 **ETag**(콘텐츠 해시)를 If-None-Match로 보내거나 Last-Modified를 If-Modified-Since로 보내면, 변경이 없을 때 서버가 **304 Not Modified**(본문 없음)로 응답해 대역폭을 절약합니다. 정적 자산은 파일명에 해시를 넣고 max-age를 길게(캐시 버스팅), HTML은 no-cache로 두는 것이 표준 전략입니다.

관련 개념 max-age/no-cache/no-store, ETag/304, 캐시 버스팅, CDN 캐시 무효화

Q12. CORS는 왜 존재하고 어떻게 동작하나요? ★★

답안 브라우저의 **동일 출처 정책(SOP)**은 스크립트가 다른 출처(스킴+도메인+포트)의 응답을 읽지 못하게 해, 악성 사이트가 사용자의 쿠키 인증을 업고 타 사이트 데이터를 훔치는 것을 막습니다. **CORS**는 서버가 명시적으로 허용한 교차 출처 요청만 브라우저가 허용하도록 완화하는 표준입니다. 서버가 Access-Control-Allow-Origin 헤더로 허용 출처를 선언하고, 단순 요청이 아닌 경우(커스텀 헤더, PUT/DELETE 등) 브라우저가 먼저 **OPTIONS 프리플라이트**로 허용 여부를 확인합니다. 중요한 이해 포인트는 CORS가 **브라우저의 보호 장치**라는 것입니다. 서버 간 호출이나 curl에는 적용되지 않으므로 CORS는 보안 경계가 아니라 브라우저 사용자 보호 장치이며, 서버 측 인증·인가는 별도로 필요합니다.

관련 개념 SOP, 프리플라이트, Allow-Origin/Credentials, CSRF와의 관계

Q13. 웹소켓(WebSocket)과 HTTP 폴링 방식들을 비교하세요. ★★

답안 실시간 양방향 통신 요구에 대한 답들입니다. **폴링**은 클라이언트가 주기적으로 요청하는 방식으로 단순하지만 지연과 낭비가 큽니다. **롱 폴링**은 서버가 이벤트가 생길 때까지 응답을 보류해 지연을 줄이지만 여전히 요청 반복 오버헤드가 있습니다. **SSE**는 하나의 HTTP 연결로 서버→클라이언트 단방향 스트림을 제공합니다(알림, 피드에 적합). **웹소켓**은 HTTP Upgrade 핸드셰이크(101 Switching Protocols)로 시작해 **완전 양방향 지속 연결**을 제공하며 채팅, 게임, 협업 도구에 적합합니다. 인프라 관점에서 웹소켓은 장수 연결이므로 LB의 유휴 타임아웃 설정, 스티키 세션 또는 pub/sub 백플레인(Redis) 설계가 필요합니다.

관련 개념 Upgrade 핸드셰이크, SSE, LB 타임아웃과 장수 연결, 스케일아웃 시 메시지 브로커

Q14. 이메일 전송 프로토콜(SMTP, POP3, IMAP)을 설명하세요. ★

답안 SMTP(25, 제출은 587)는 메일 **발송·중계** 프로토콜로, 발신 클라이언트→발신 서버→수신 서버 구간을 담당합니다. 수신함 읽기는 **POP3**(110/995) — 메일을 로컬로 내려받고 서버에서 삭제하는 단말 중심 방식 — 와 **IMAP**(143/993) — 메일을 서버에 두고 여러 기기에서 동기화 — 가 담당합니다. 현대는 IMAP(또는 웹메일/Exchange)이 표준입니다. 스팸·위조 방지를 위한 SPF(발신 IP 검증), DKIM(서명), DMARC(정책)를 함께 알아두면 "우리 회사 메일이 스팸함에 들어가요" 같은 실무 질문에 대응할 수 있습니다.

관련 개념 MTA/MUA, 포트 25/587/465, SPF/DKIM/DMARC, 오픈 릴레이

Q15. curl로 API 장애를 진단하는 방법을 설명해보세요. (실무 연계) ★★

답안 ① **curl -v**로 상세 과정(DNS 해석, TCP 연결, TLS 핸드셰이크, 요청/응답 헤더)을 단계별로 확인해 어느 단계에서 실패하는지 분리합니다. ② DNS 의심 시 --resolve로 특정 IP를 강제해 DNS와 서버 문제를 분리합니다(LB 뒤 특정 서버 직접 테스트에도 유용). ③ -w 옵션으로 time_namelookup, time_connect, time_starttransfer를 찍어 **지연 구간을 정량화**합니다. ④ TLS 문제는 -k(검증 생략)로 인증서 문제인지 확인 후 openssl s_client로 체인을 봅니다. ⑤ 헤

더·바디를 조작(-H, -d)해 재현 조건을 좁힙니다. "브라우저에선 되는데 서버에선 안 돼요"류 문제는 프록시 환경변수, SNI, User-Agent 차이를 점검합니다.

관련 개념 curl -v/-w/--resolve, openssl s_client, SNI, 단계별 지연 분석
