# 클라우드 디자인 패턴

총 42개의 클라우드 디자인 패턴

* DB성능개선: 1개
* 읽기 최적화: 4개
* 핵심업무 집중: 6개
* 안정적 현대화: 2개
* 효율적 분산처리: 13개
* 회복성: 6개
* 보안: 3개
* 운영: 7개

| **No.** | **분류** | **목적** | **패턴명** | **패턴 설명** |
| --- | --- | --- | --- | --- |
| 1 | DB 성능 개선 | 양 줄이기 | [Sharding](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2bn6wsx) | 데이터 저장소를 수평적으로 분할(shard)하여 대규모 데이터 저장 및 접근 시 확장성을 높이는 패턴 |
| 2 | 읽기 최적화 | NoSQL DB Query 최적화 | [Index Table](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1fob9te) | 데이터 저장소에서 자주 참조되는 필드에 대한 인덱스를 생성하여 쿼리 성능을 개선하는 패턴 |
| 3 | 읽기 최적화 | 성능 향상 및 데이터 일관성 유지 | [Cache-Aside](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.4d34og8) | 데이터 저장소에서 캐시에 데이터를 필요에 따라 로드하여 성능을 개선하고, 캐시와 데이터 저장소 간의 일관성을 유지하는 패턴 |
| 4 | 읽기 최적화 | 쿼리 성능 최적화 | [Materialized View](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3iwdics) | 데이터를 미리 변환하여 준비된 뷰를 생성함으로써 쿼리 성능을 높이고 데이터 추출을 효율화하는 패턴 |
| 5 | 읽기 최적화 | 읽기/쓰기 분리 | [CQRS](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1y1nskl) | 데이터 저장소의 읽기와 쓰기 작업을 분리하여 성능, 확장성, 보안성을 높이는 패턴 |
| 6 | 핵심업무 집중 | 횡단관심사 분리 | [Gateway Offloading](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.4i1bb8e) | SSL 인증서 관리, 인증, 로깅 등의 공통 기능을 게이트웨이로 분리하여 애플리케이션의 복잡도를 낮추는 패턴 |
| 7 | 핵심업무 집중 | 라우팅 중앙 처리 | [Gateway Routing](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2x6llg7) | 단일 엔드포인트를 통해 요청을 받아 백엔드 서비스나 인스턴스로 라우팅하는 패턴 |
| 8 | 핵심업무 집중 | 클라이언트 요청 수 줄이기 | [Gateway Aggregation](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1cbvvo0) | 단일 엔드포인트에서 클라이언트 요청을 받아 여러 백엔드 서비스로 분배하고 응답을 취합하는 패턴 |
| 9 | 핵심업무 집중 | 프론트엔드 유형별 전용처리 | [Backends for Frontends](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3wbjebt) | 특정 프런트엔드에 특화된 백엔드 서비스를 별도로 구축하는 패턴 |
| 10 | 핵심업무 집중 | 공통 기능 분리 | [Sidecar](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2bgtojm) | 애플리케이션의 일부 컴포넌트를 별도 프로세스나 컨테이너로 분리하여 격리와 확장성을 제공하는 패턴 |
| 11 | 핵심업무 집중 | 네트워크 통신의 안정성과 보안 강화 | [Ambassador](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.qm3yrf) | 클라이언트를 대신해 네트워크 요청을 처리하는 헬퍼 서비스를 생성하는 패턴 |
| 12 | 안정적 현대화 | 현대화의 위험 최소화와 점진적 전환 | [Strangler Fig](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3alrhf8) | 레거시 시스템을 점진적으로 새로운 애플리케이션 및 서비스로 교체하는 패턴 |
| 13 | 안정적 현대화 | 시스템 간 안정적 인터페이스 | [Anti-Corruption Layer](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1pr1rn1) | 서로 다른 하위 시스템 간의 의미적 차이를 조정하기 위해 중간 계층을 구현하는 패턴 |
| 14 | 효율적 분산처리 | 작업 단계 모듈화로 재사용성과 성능 향상 | [Pipes and Filters](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2eclud0) | 복잡한 작업을 독립적인 단계(필터)로 분리하고 메시지(파이프)로 연결하여 모듈성과 유연성을 높이는 패턴 |
| 15 | 효율적 분산처리 | 워크플로우의 신뢰성 향상 | [Scheduler Agent Supervisor](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2ovzkin) | 작업 단계를 스케줄러, 에이전트, 감독자로 분리하여 신뢰성과 확장성을 높이는 패턴 |
| 16 | 효율적 분산처리 | 분산 작업의 충돌 방지와 안정성 향상 | [Leader Election](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1419uqg) | 분산 시스템에서 여러 작업 인스턴스 중 하나를 리더로 선출하여 조정 역할을 맡기는 패턴 |
| 17 | 효율적 분산처리 | 데이터 일관성 보장 | [Saga](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3o0xde9) | 각 서비스의 로컬 트랜잭션을 사용하여 분산 트랜잭션의 일관성을 보장하는 패턴 |
| 18 | 효율적 분산처리 | 오류 복구로 데이터 일관성 보장 | [Compensating Transaction](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2367nm2) | 분산 트랜잭션에서 실패한 작업을 보상하기 위해 이전 작업을 취소하거나 상쇄하는 트랜잭션을 실행하는 패턴 |
| 19 | 효율적 분산처리 | 중요 작업의 우선 처리 보장 | [Priority Queue](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.393x0lu) | 메시지의 우선순위에 따라 처리 순서를 조정하는 큐를 사용하는 패턴 |
| 20 | 효율적 분산처리 | 부하의 균등한 분산으로 안정성 확보 | [Queue-Based Load Leveling](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.ibhxtv) | 메시지 큐를 사용하여 작업과 서비스 간의 부하를 균등하게 분산시키는 패턴 |
| 21 | 효율적 분산처리 | 처리순서 보장 | [Sequential Convoy](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.32b5gho) | 관련 메시지 집합을 순서대로 처리하되 다른 메시지 처리를 차단하지 않도록 하는 패턴 |
| 22 | 효율적 분산처리 | 메시지 크기 최소화 및 성능과 보안 향상 | [Claim Check](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1hgfqph) | 메시지에서 페이로드를 분리하여 외부 저장소에 저장하고 참조키(클레임 체크)를 사용하는 패턴 |
| 23 | 효율적 분산처리 | 단일 이벤트 메시지의 복수 서비스 처리 보장 | [Publisher-Subscriber](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.41g39da) | 다수의 소비자(Consumer)에게 이벤트 발행 |
| 24 | 효율적 분산처리 | 장시간 처리 작업의 응답시간 단축 | [Asynchronous Request-Reply](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2gldjl3) | **프런트엔드(클라이언트)와 백엔드 간** 비동기로 요청과 응답을 분리하여 응답 시간을 단축 |
| 25 | 효율적 분산처리 | 병렬처리로 작업 처리 속도 향상 | [Competing Consumers](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.vqntsw) | 동일 메시지 채널에서 여러 소비자가 경쟁적으로 메시지를 처리하여 병렬성을 높이는 패턴 |
| 26 | 효율적 분산처리 | 중앙집중 처리의 병목현상 방지 | [Choreography](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3fqbcgp) | 중앙 조정자 없이 각 서비스가 자율적으로 이벤트를 구독하고 반응하여 전체 워크플로를 수행하는 패턴 |
| 27 | 안정성 | 요청 폭주 방지로 안정성 유지 | [Rate Limiting](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1uvlmoi) | 일정 기간 동안 허용되는 요청 수를 제한하여 과부하를 방지하고 서비스 안정성을 높이는 패턴 |
| 28 | 안정성 | 요청 폭주 방지로 안정성 유지 | [Throttling](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.4ev95cb) | 시스템의 부하 상태에 따라 요청 처리량을 **동적으로 조절**하여 과부하를 방지하는 패턴 |
| 29 | 안정성 | 자원풀 격리로 장애 전파 방지 | [Bulkhead](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2u0jfk4) | 애플리케이션 요소를 격리된 풀로 분할하여 하나의 장애가 전체로 전파되는 것을 방지하는 패턴 |
| 30 | 안정성 | 장애전파 방지 | [Circuit Breaker](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2o52c3y) | 장애가 발생한 구성 요소를 빠르게 감지하고 요청 실패를 최소화하는 패턴 |
| 31 | 안정성 | 일시적 오류시 처리 보장 | [Retry](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.10v7oud) | 일시적인 오류에 대해 실패한 요청을 재시도하여 복원력을 높이는 패턴 |
| 32 | 안정성 | 데이터 멱등성 보장과 변경 기록 제공 | [Event Sourcing](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.195tprx) | 데이터에 대한 모든 변경사항을 이벤트로 저장하고, 이벤트를 재생하여 데이터의 상태를 복원하는 패턴 |
| 33 | 보안 | 사용자 인증 및 관리 효율화 | [Federated Identity](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3t5h8fq) | 인증을 외부 ID 제공자에 위임하여 사용자 관리를 간소화하고 SSO를 구현하는 패턴 |
| 34 | 보안 | 데이터 접근 제어와 보안 강화 | [Gatekeeper](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1w363f7) | 신뢰할 수 있는 호스트에 보안 관련 기능을 집중시켜 스토리지나 서비스의 보안을 강화하는 패턴 |
| 35 | 보안 | 네트워크 대역폭 감소 | [Valet Key](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.28arinj) | 클라이언트가 특정 리소스에 제한된 직접 접근을 할 수 있도록 토큰을 사용하는 패턴 |
| 36 | 운영 | 글로벌 서비스 가용성과 성능 최적화 | [Geodes](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.ng1svc) | 백엔드 서비스를 여러 지역에 분산 배치하여 지연 시간을 줄이고 가용성을 높이는 패턴 |
| 37 | 운영 | 멀티 테넌트 관리 | [Deployment Stamps](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1fhy1k3) | 리소스 그룹을 복제하여 작업이나 테넌트 단위로 격리된 운영 환경을 제공하는 패턴 |
| 38 | 운영 | 서비스 가용성 상태 점검 | [Health Endpoint Monitoring](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.37fpbj5) | 애플리케이션의 상태를 모니터링하기 위한 전용 API 엔드포인트를 노출하는 패턴 |
| 39 | 운영 | 자원 사용 효율성과 비용 절감 | [Compute Resource Consolidation](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.1mkzlqy) | 여러 작업이나 운영을 단일 컴퓨팅 단위로 통합하여 효율성과 비용을 최적화하는 패턴 |
| 40 | 운영 | 정적 자원 제공 비용절감과 성능 향상 | [Static Content Hosting](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.46kn4er) | 정적 콘텐츠를 클라우드 스토리지에 배포하여 클라이언트에 직접 제공함으로써 컴퓨팅 인스턴스 사용을 줄이는 패턴 |
| 41 | 운영 | 환경설정 중앙관리와 재배포 없이 설정 변경 적용 | [External Configuration Store](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.2lpxemk) | 애플리케이션의 설정 정보를 중앙화하여 관리 효율성을 높이고 설정 값 변경 시 재배포 없이 적용하는 패턴 |
| 42 | 운영 | 엣지컴퓨팅 의 효율적 관리 | [Edge Workload Configuration](https://docs.google.com/document/d/1tsUTXRqdb0jxn0_V3naNFiJv9FYHGoCJ/edit#heading=h.3s49zyc) | 장치와 시스템이 혼재된 엣지 환경에서 워크로드 구성을 효율적으로 관리하여 지연 시간 단축과 네트워크 비용 절감을 하는 패턴 |