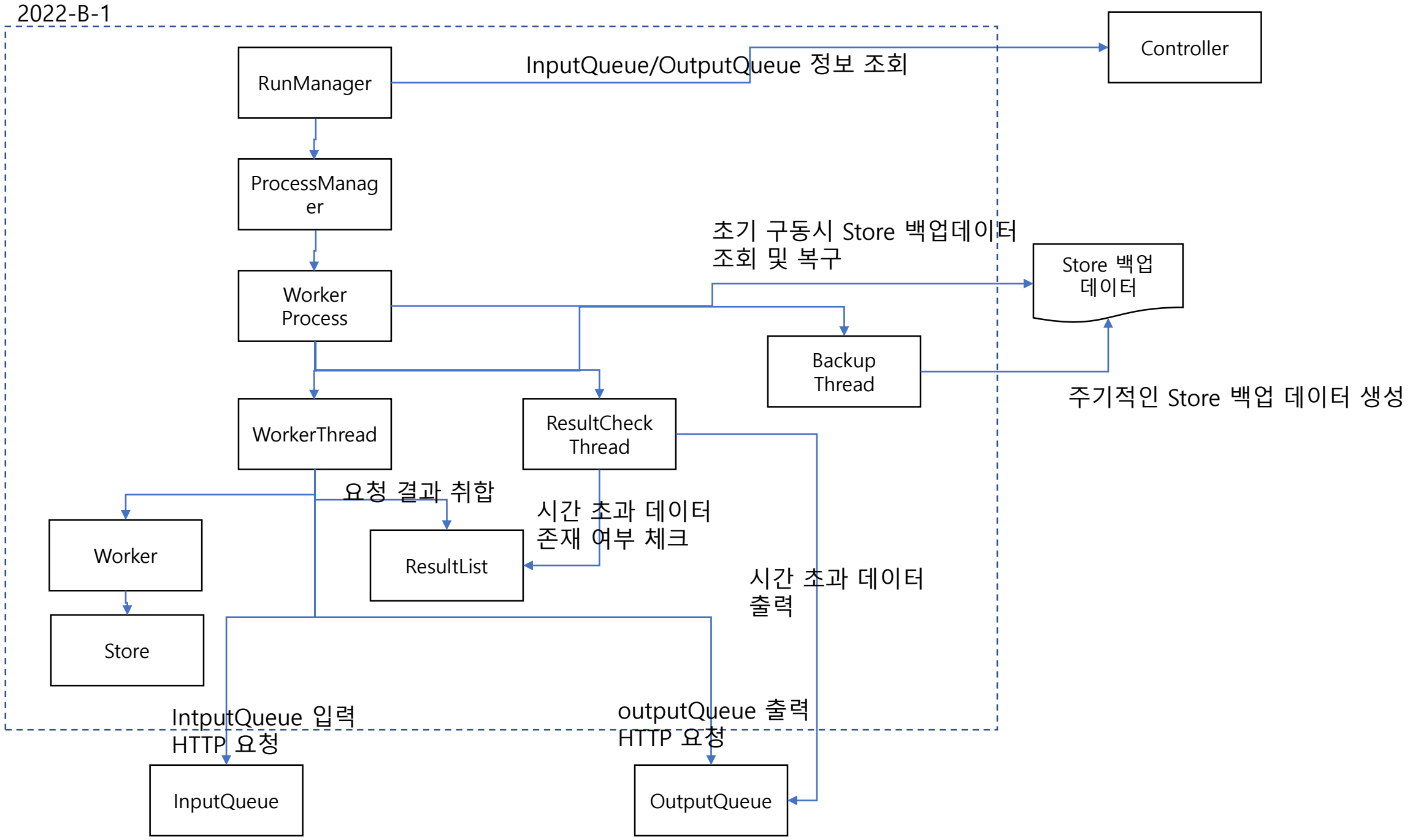
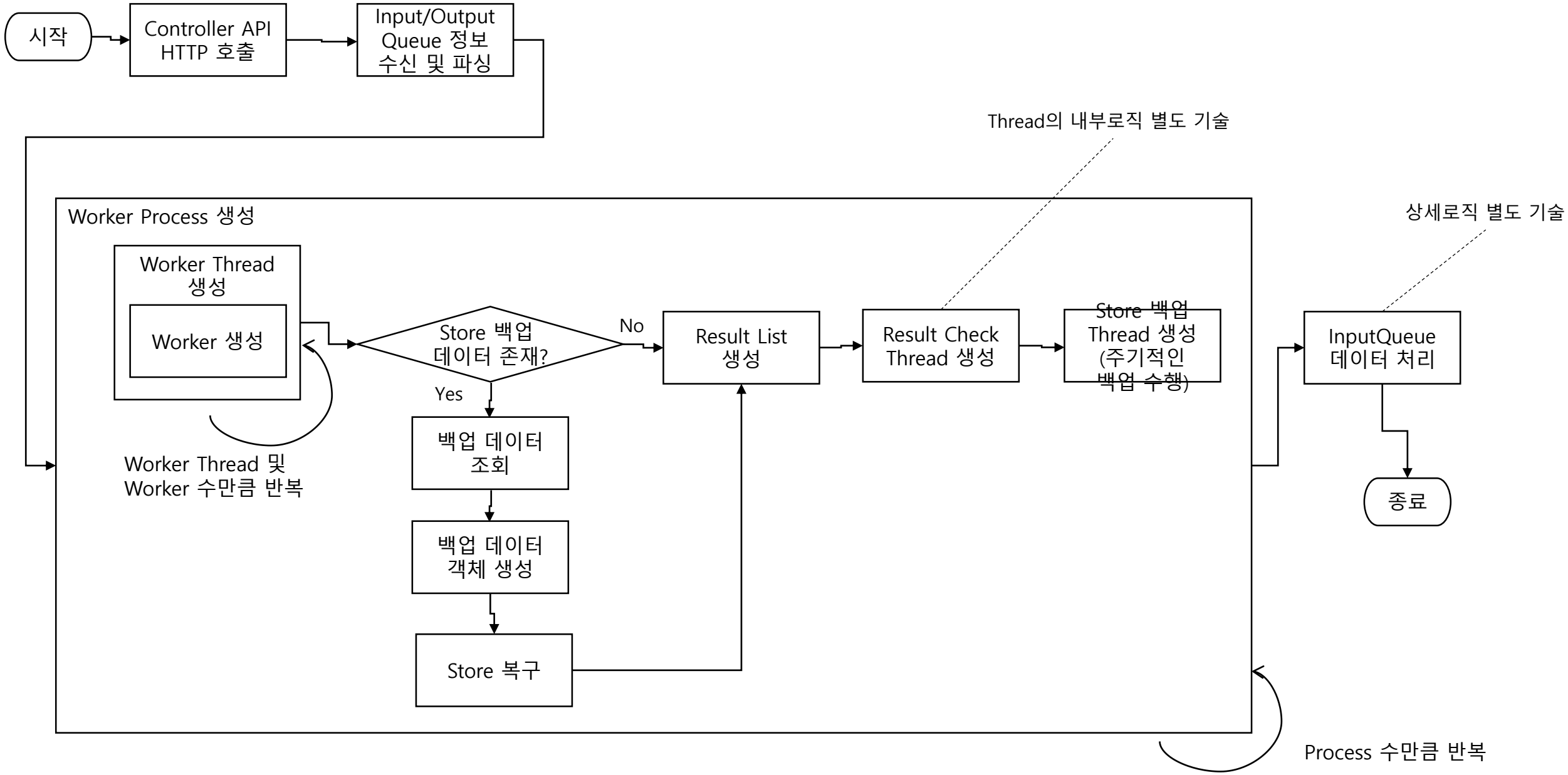
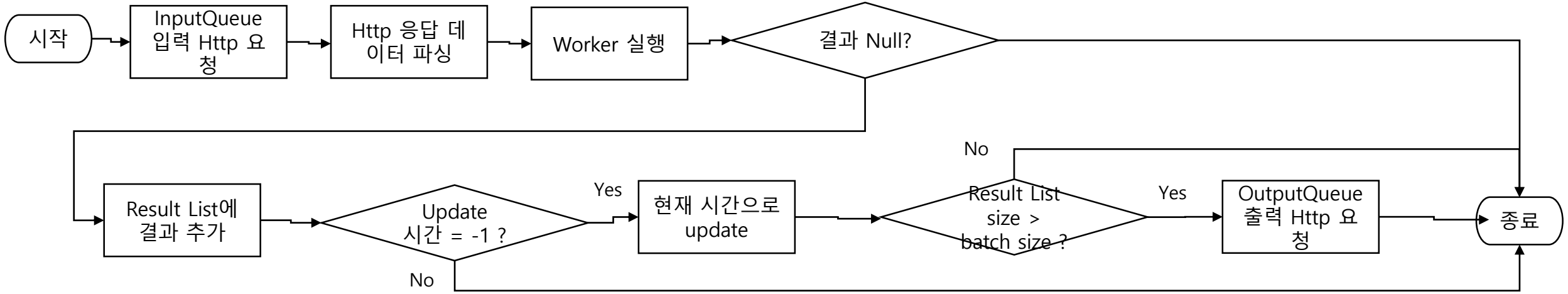

2022-B-1
RunManager
InputQueue/OutputQueue 정보 조회
Controller
ProcessManag
er
초기 구동시 Store 백업데이터
조회 및 복구
Store 백업
데이터
Worker
Process
Backup
Thread
주기적인 Store 백업 데이터 생성
WorkerThread
ResultCheck
Thread
요청 결과 취합
시간 초과 데이터
존재 여부 체크
Worker
ResultList
시간 초과 데이터
출력
Store
IntputQueue 입력
HTTP 요청
outputQueue 출력
HTTP 요청
InputQueue
OutputQueue

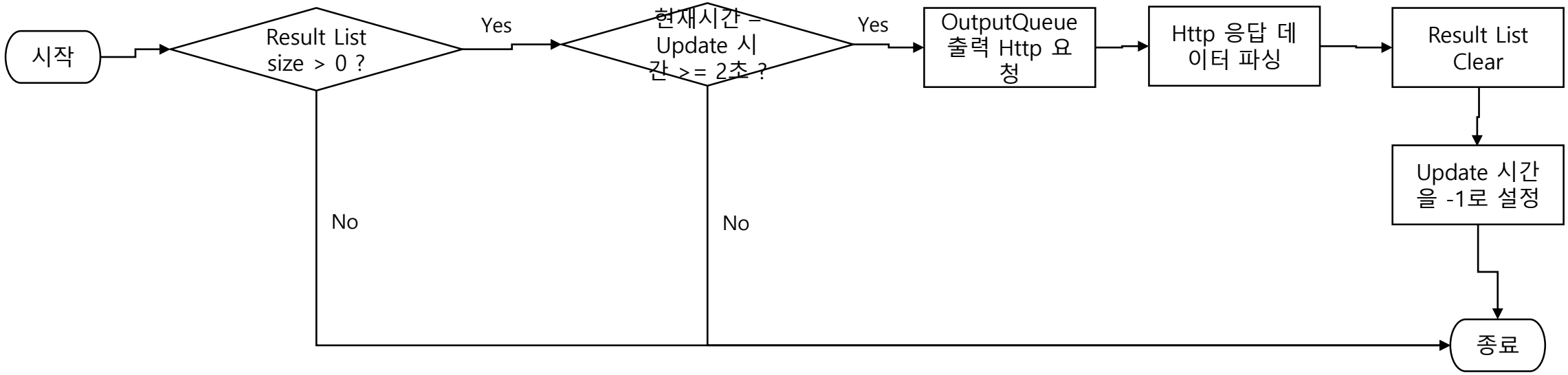
2022-B-2 Main Flow



2022-B-2 InputQueue 데이터 처리 상세



2022-B-2 Result Check Thread 내부 로직 상세



컴포넌트	사용 기술	비고
RunManager, WorkerThead, ResultCheck Thread	HttpClient	사유 : HTTP 요청 송신 및 응답 수신
RunManager	SingleThreadScheduledExecutor	사유 : 주기적인 Process 상태 체크를 통해 Process가 중지되었을 때 재기동하도록 SingleThreadSchedulerExecutore의 scheduleWithFixedDelay 메소드를 사용
ProcessManager	ProcessBuilder	사유 : 프로세스 생성
RunManager	Gson	사유 : HTTP 요청 데이터 생성 및 응답 데이터 파싱
ResultList	Synchronized List	사유 : 여러 Thread에서 접근해야 하므로 동시성 문제 해결을 위해 대체 기술 : Lock이나 synchronized 키워드를 사용해 구현할 수도 있다.
Backup Thread	SingleThreadScheduledExecutor	사유 : scheduleAtFixedRate 메소드를 이용해 주기적인 반복 실행(Store 데이터 백업) 구현
Backup Thread	Serializable	사유 : Store 데이터를 파일 형태로 저장하기 위해
WorkerThread, Backup Thread	ObjectInputStream ObjectOutputStream FileInputStream FileOutputStream	사유 : Store 데이터 객체와 파일 간의 상호 변환을 위해
WorkerThread, ResultCheck Thread, Backup Thread	Thread	사유 : 독립적인 업무를 비동기로 처리

주어진 주제에 대한 기술 전문성 및 문제해결 위주의 서술식 평가
- 범용성(Configurable Customizing)을 보유한 솔루션의 설계/개발을 리딩 할 수 있다.
- 시장/기술 Trend 및 자사 사업 방향에 Align된 신규 솔루션을 기획/발굴 할 수 있다.
- 기반 솔루션/특수 목적 소프트웨어의 시장 경쟁력을 고려한 상용화를 기획 할 수 있다.

1. 구성도 : 감점 없게, 적당히, 요건 누락 없이
2. 순서도 : 개략적, 핵심은 강조하고 상세하게, 요건 누락 없이
3. 사용한 기술에 대해 설명, 어느 구성에 어떤 기술을 왜 썼는지, 선택사유(유사 기술 중에...), 디자인패턴
   5~7개
4. 신기술 적용 포인트, 기술/시장 경쟁력, 기술/운영적 측면, 추가하고 싶은 기능, 대체할 수 있는 기술
   kafka, web, socket, msa, ...
   4~5개

2차 workflow
Json에 workflow 정보

3차 state
Action state
replace

기본 요구사항

- 파일 처리
    - 파일 read : 압축/암호화할 파일 읽기
    - 파일 write : 압축/암호화된 파일 생성
- 파일 압축
    - 줄 압축 : 동일한 내용의 line이 반복될 경우 압축
    - 내용 압축 : 한 line에서 동일한 문자가 반복될 경우 압축
- 파일 전송
    - Line별 전송
    - 오류 발생시 재전송
    - 특정 line 전송

추가 요구사항

- 확장성
    - 압축 알고리즘은 변경 가능해야 한다.
    - 암호화 알고리즘은 변경 가능해야 한다.
- 성능
    - 다중 사용자에 대한 요청을 처리할 수 있어야 한다.
      : 서버의 처리량보다 사용자 요청이 많을 경우에도 처리가 가능해야 한다.
    - 동일 파일에 대한 압축은 가능한 최소화한다.
      : 동일한 파일에 대해 동일한 알고리즘으로 압축/암호화 하는 것을 최소화하기 위해 압축/암호화 파일은 별도로 관리한다.
      단, 관리하는 파일의 양은 적정 수준을 유지해야 한다.
    - 대용량 파일 저장소는 확장 가능해야 한다.
    - 다국적으로 서비스를 제공해야 할 경우, 지역별(예. 아시아, 북미, 유럽 등)로 서버 배치
      : 지역간 파일 및 데이터 동기화를 고려해야 함
- 보안
    - 무결성 확인을 위한 체크섬 관리 (SHA-1, MD5 등)
      : 체크섬을 제공하여 파일이 정상적으로 다운로드 되었는지 확인할 수 있도록 한다.
    - 블랙리스트 관리
      : 요청 이력(IP, 건수, 용량) 등을 기반으로 시스템에 부하가 갈 정도로 악의적 요청을 보내는 클라이언트 차단
- 신뢰성
    - 서비스 중단을 방지하도록 시스템을 이중화해야 한다.

추가 요구사항 (이어서)

- 기능
  - 파일 업로드 및 삭제
    : 전송할 대용량 파일 관리 (저장소에 업로드, 저장소에서 삭제)
  - 파일 목록 검색 및 페이징
    : 전송할 대용량 파일을 검색할 수 있는 기능. 건수가 많을 경우 페이징으로 출력하는 기능.
  - 전송 중 오류가 발생한 경우 재전송이 가능해야 한다.
    : 파일 전송 도중 오류가 발생하여 다시 전송할 경우, 처음부터 다시 전송 받는 것이 아니라 오류가 발생한 부분부터 이어서 전송한다.
  - 사용자 관리
  - 로그인/로그아웃
  - 그룹 관리
    : 사용자를 그룹 단위로 관리할 수 있는 기능
  - 권한 관리
    : 사용자 또는 그룹 단위로 다운로드 가능한 파일 권한 관리
      파일을 관리(업로드 및 삭제)할 수 있는 사용자 권한 관리
  - 사용 이력 로깅
    : 파일별, 사용자별 전송 이력을 로깅한다.
      시스템 부하를 막기 위해 특정 기간(예. 6개월)이 지난 이력은 주기적으로 삭제한다.
  - 사용량 측정
  - 스케쥴링 (예약 전송)
    : 사용자가 원하는 시간 또는 주기적으로 특정 파일을 지정하여 예약 전송할 수 있는 기능
  - 보고서 생성
    : 관리자를 위해 특정 기간에 대한 보고서 생성
  - 과금 관리
    : 사용자별 과금 관리

*** 메커니즘 분석 - 압축 유틸리티**
성능 테스트를 수행하여 목적에 맞는 압축 유틸리티를 선정한다.
- 측정 항목
    - 압축 속도 및 압축률
        - 압축 속도
        - 압축 해제 속도
        - 압축 사이즈 / 원본 사이즈 비율
        - 리소스 사용률 (CPU 및 메모리)
- 기타 고려 사항
    - 라이선스
    - 오픈 소스인 경우 릴리즈, 커밋 건수 등 커뮤니티 활동 상황
    - 지원 언어
- 대상 유틸리티의 예
    - gzip, bzip2, 7za 등

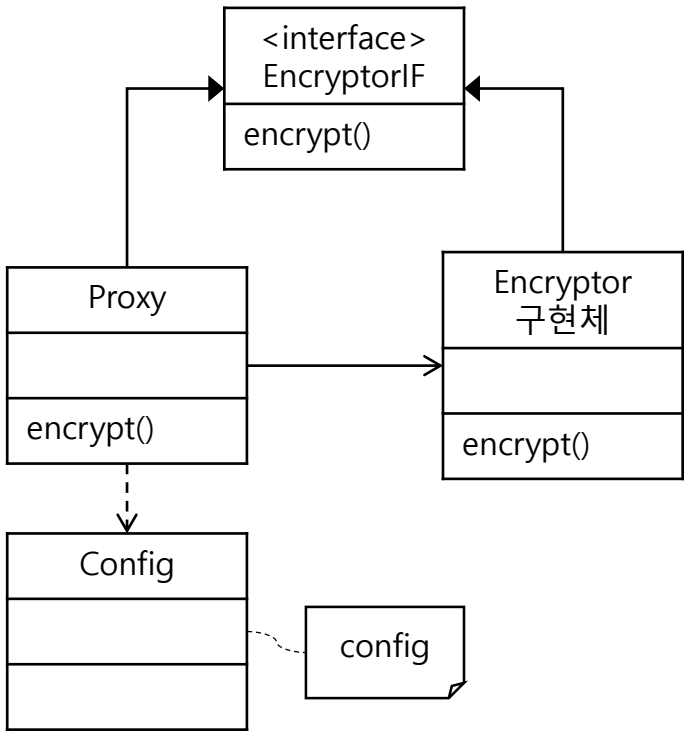
*** 메커니즘 분석 - 암호화**
암호화 알고리즘을 선정한다.
전송된 파일을 사용자가 압축 해제할 수 있는 양방향 암호화 알고리즘 중 선택한다.

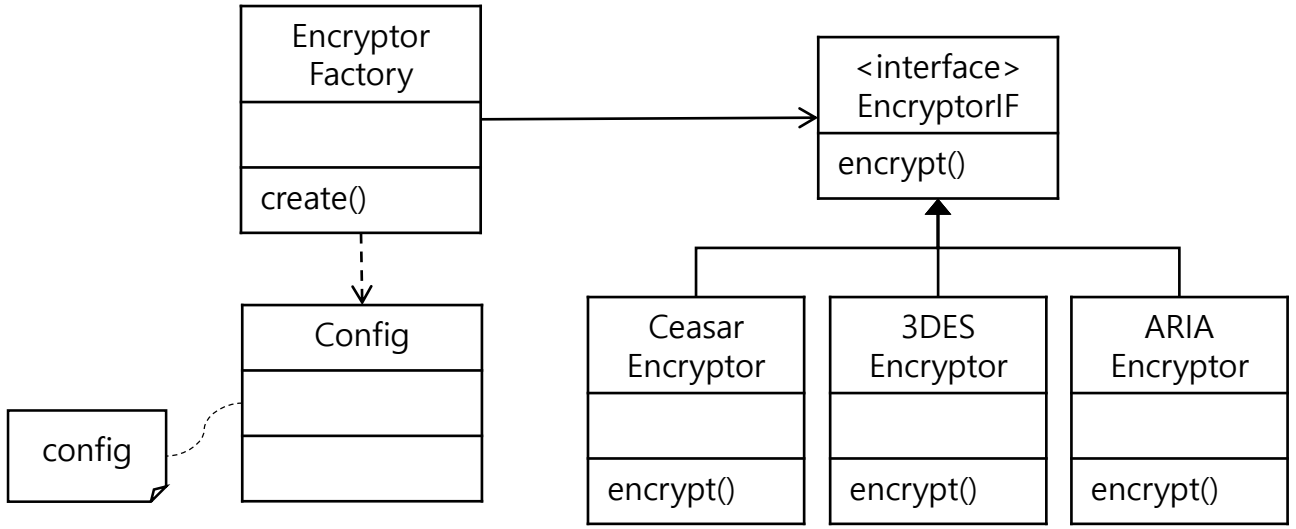
- 대칭키 알고리즘
    - 특징 : 암복호화에 서로 동일한 키가 사용되는 암호화 방식
    - 장점 : 암호화 속도가 빠르다
    - 단점 : 송신 측에서 수신 측에 암호키를 전달하는 과정에서 노출 우려가 있다.
    - 알고리즘의 예 : 3DES, AES, ARIA, SEED 등
- 비대칭키 알고리즘
    - 특징 : 암복호화에 서로 다른 키가 사용되는 암호화 방식. 하나의 키는 공개키로 사용.
    - 장점 : 암호키 전달 과정이 불필요하므로 안정성이 높다.
    - 단점 : 대칭키 방식에 비해 느리다
    - 알고리즘의 예 : RSA, ECC 등

후보 알고리즘을 선정 후 암호화 성능을 측정한다.
시스템에 감내할 수 있는 최소 수준의 성능에 매치하면서 보안성이 높은 알고리즘을 선정한다.
보다 높은 수준으로 보안을 강화해야 할 경우에는 비대칭키 알고리즘을 선정하고, 인증서 발급을 위한 CA 도입을 고려한다.

*** 메커니즘 분석 – 압축/암호화 메커니즘**

시스템 전체에서 단일 알고리즘을 사용할 경우에는 proxy patter을 이용해 구현하고, 복수 개의 알고리즘을 사용할 경우에는 factory pattern을 사용한다.





알고리즘별 구현체를 Config 컴포넌트를 이용해 설정으로 관리한다.
Config 컴포넌트는 별도의 설정 파일을 통해 시스템 전반의 설정을 관리하며, reload 수행시 config에 대한 조회를 block함으로써 일관성을 유지한다.

제약 사항
- 클라이언트 영역에도 동일하게 압축 해제 및 복호화 매커니즘이 적용되어 있어야 한다.

*** 메커니즘 분석 – 다중 사용자 처리**

다중 사용자의 요청을 처리할 수 있어야 한다.
대용량 파일 전송의 경우 메모리와 CPU를 많이 사용하는데, 과도한 요청으로 대기열이 쌓여서 시스템 성능이 저하되고 시스템이 다운되는 현상을 방지하기 위해 비동기 메시지 처리 방식으로 구성한다.

- Message Queue 특징
    - 비동기 (Asynchronous) : 비동기 처리
    - 비동조 (Decoupling) : 어플리케이션과 분리 가능
    - 탄력성 (Resilience) : 일부 실패시 전체 영향을 받지 않음
    - 반복 (Redundancy) : 실패할 경우 재실행 가능
    - 보증 (Guarantees) : 작업이 처리된 것을 확인할 수 있음
    - 확장성 (Scalable) : 다수의 프로세스들이 큐에 메시지를 보낼 수 있음
- Message Queue 제품 예
    - ActiveMQ
        - 다양한 언어 지원
        - 클러스트링 가능
        - 모니터링 도구 미지원
    - RabbitMQ
        - 다양한 언어 지원
        - 클러스트링 가능
        - 실시간 모니터링 및 관리 용이
    - Kafka
        - 대용량 실시간 처리에 특화되어 설계된 메시지 시스템
        - 분산 시스템을 기본으로 설계
- 선정을 위한 판단 근거
    - 성능
    - 지원 언어
    - 설치를 위한 최소 사양
    - 충분한 레퍼런스가 존재하는지
    - 오픈 소스인 경우, 릴리즈, 커밋 건수 등 커뮤니티 활동 상황
    - 상용인 경우, 기술 지원

*** 메커니즘 분석 – 압축/암호화 최소화**

방안 1) 대용량 파일을 저장하는 시점에 미리 압축/암호화를 하여 저장한다.

단점 : 모든 파일에 대해 압축/암호화된 파일을 저장하려면 (전체 파일 용량 * 압축률)만큼의 저장소가 필요함
압축 또는 암호화 알고리즘이 변경되면 전체에 대해 다시 압축/암호화를 수행해야 하므로 해당 시점에 부하가 상당함

→ drop

방안 2) 자주 요청되는 파일에 대해서 압축/암호화된 파일을 별도로 저장한다.

사용자의 파일 사용 특성에 따라 캐쉬 알고리즘과 캐쉬 사이즈를 선정하고, 압축/암호화된 파일을 캐쉬한다.

- 캐쉬 대상
  - 파일 정보 : DB에 저장 (알고리즘 ID도 함께 저장되어야 함)
    - 원본 파일명
    - 암호화 알고리즘 ID
    - 압축 알고리즘 ID
    - 압축/암호화된 파일 경로
    - 최근 요청일시
    - 요청 건수
  - 압축/암호화된 파일 : 별도의 파일 저장소에 저장
- 캐쉬 알고리즘 (LRU 또는 LFU 선호)
  - LRU (Least Recently Used)
  - LFU (Least Frequently Used)
  - FIFO (First In First Out)
  - Random
- 캐쉬 메커니즘 구현시 고려해야 하는 사항
  - 파일 정보 삭제시 파일 저장소에서 압축/암호화된 파일도 함께 삭제한다.
  - 원하는 시점에 캐쉬를 clean 할 수 있는 API를 제공한다.
  - 압축 또는 암호화 알고리즘이 변경되면, 캐쉬 파일 정보 및 파일을 모두 삭제한다.
  - 알고리즘 및 캐쉬 사이즈는 별도의 설정 파일로 관리하고 실시간으로 변경할 수 있도록 한다.
- 제약사항
  - 멀티 서버에서 동일한 파일 저장소를 사용할 수 있도록 NAS 등의 적용이 고려되어야 한다. 그렇지 않을 경우에는 DB로 저장하는 파일 정보에 서버 정보가 추가되어야 한다.

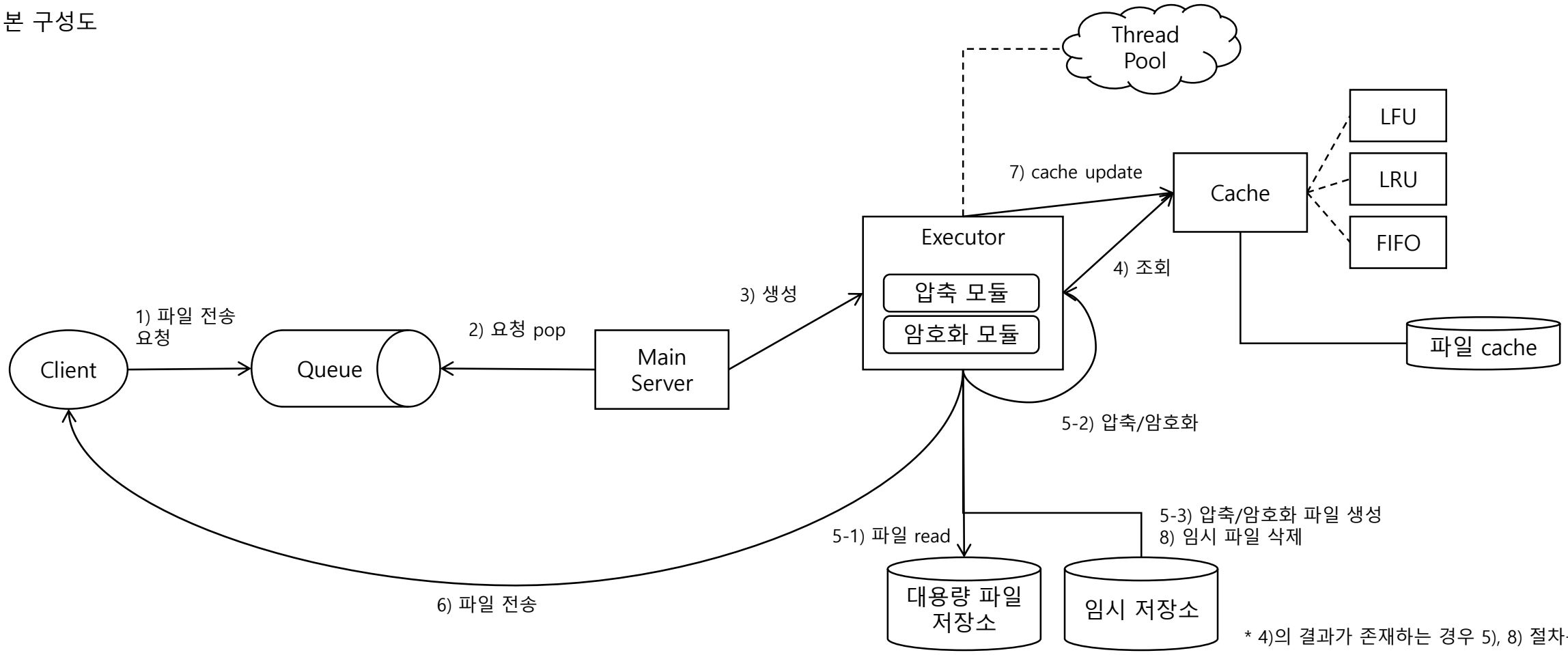
*** 메커니즘 분석 - 체크섬**

전송된 파일의 유효성을 클라이언트에서 검사할 수 있도록 체크섬을 제공한다.
단방향 암호화을 통해 생성한 해쉬를 체크섬으로 제공한다.

- 체크섬 생성 알고리즘
    - MD5 : 취약점이 알려져서 사용을 권장하지 않음
    - SHA : SHA-0, SHA-1은 취약점이 알려져있으므로 SHA-2 권장. 통상 SHA-256 이상의 알고리즘을 사용하도록 권장됨.
- 구현 방안
    - Java의 MessageDigest 활용
    - com.google.common.hash의 File.asByteSource(file).hash(hashFunction) 사용

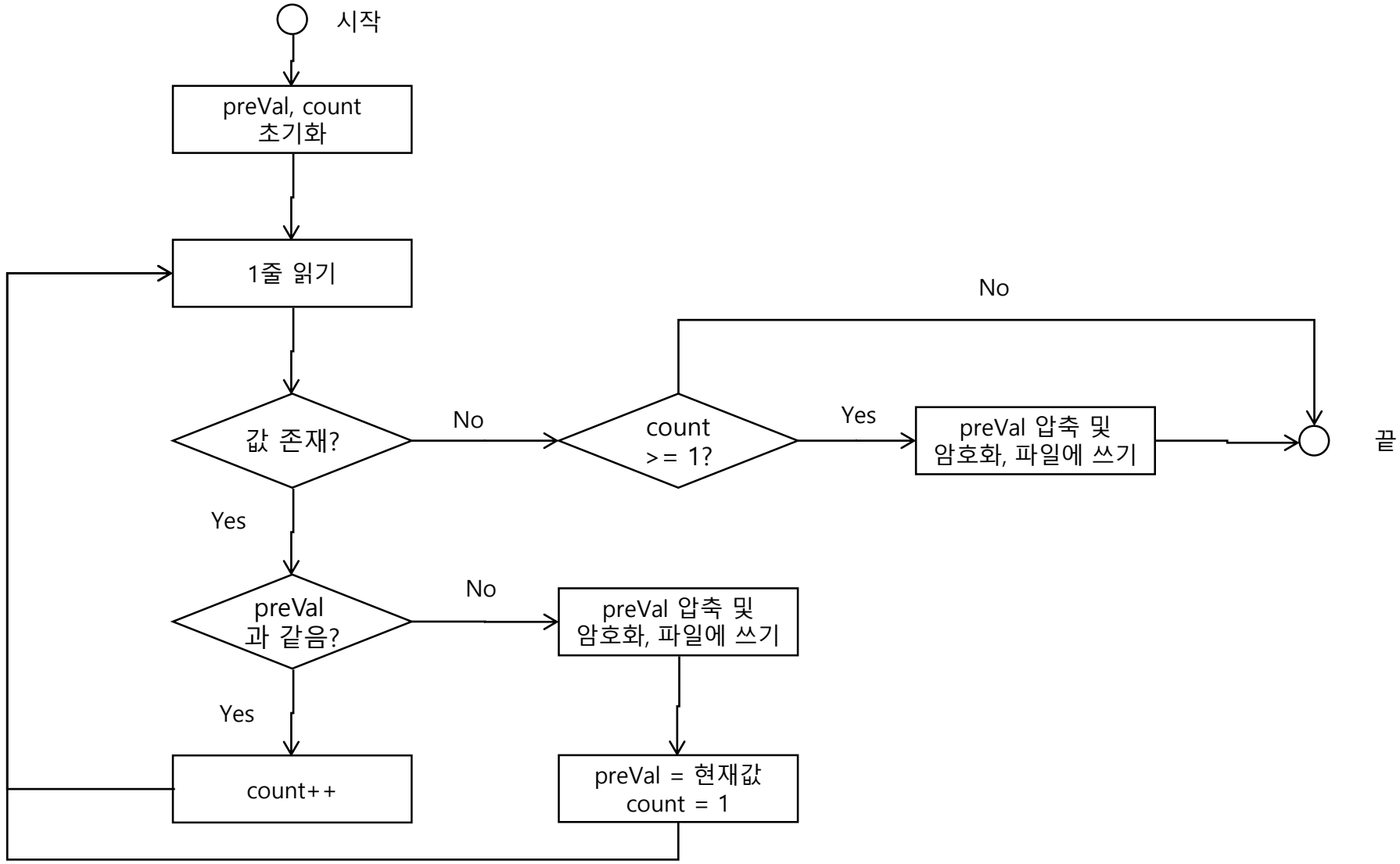
    → 대용량 파일에 대한 성능 비교 및 사용 편의성을 고려하여 선정

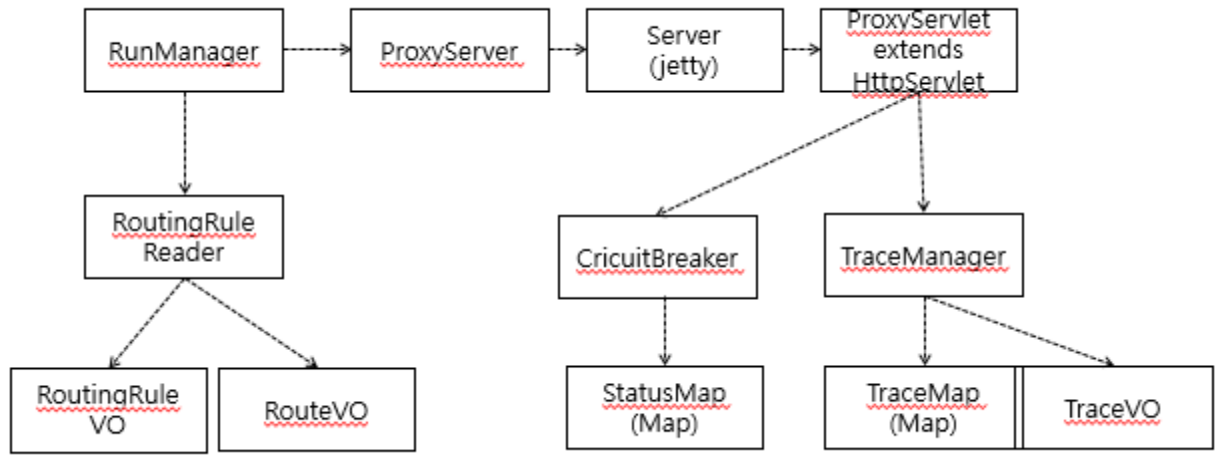
* 기본 구성도

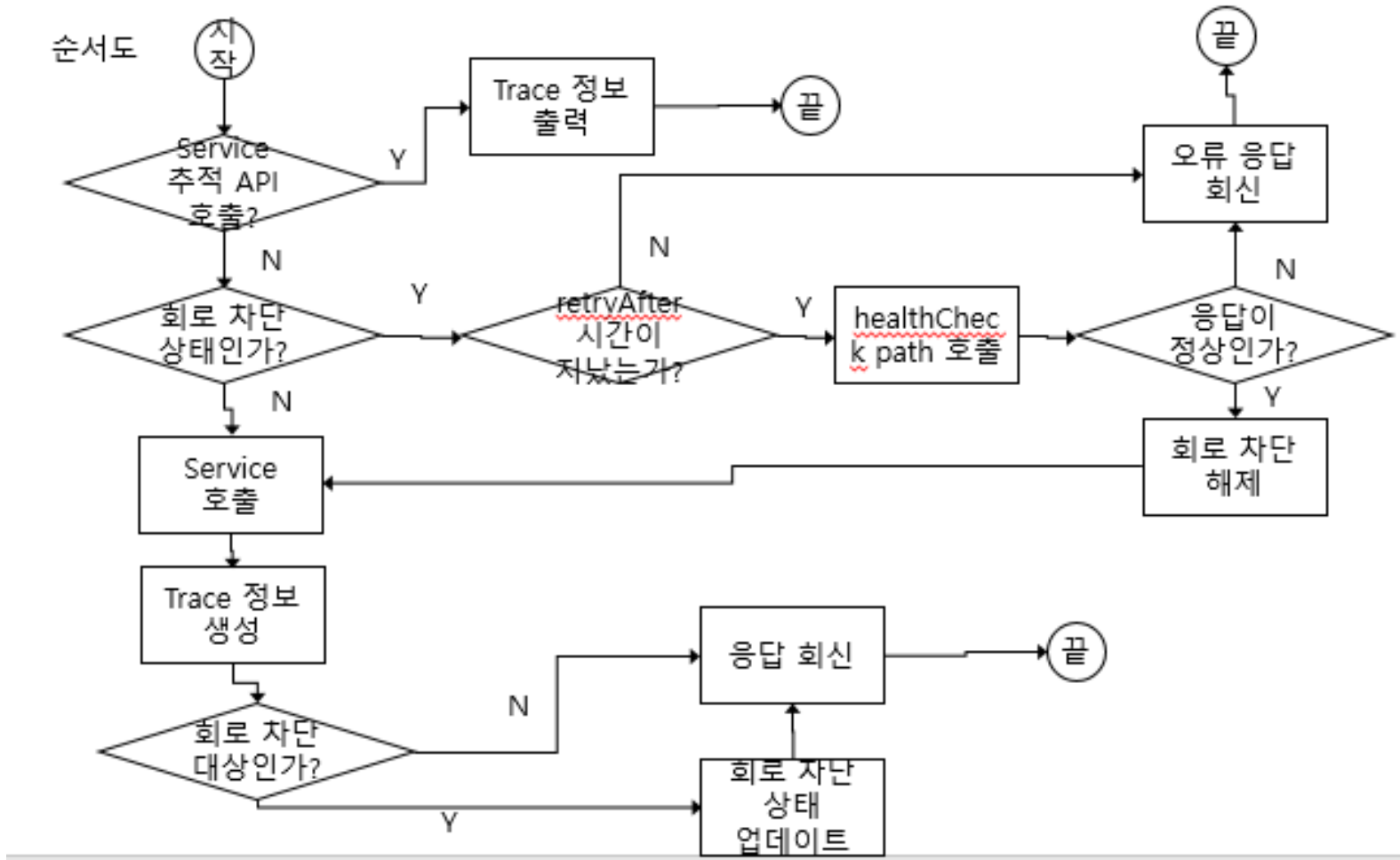


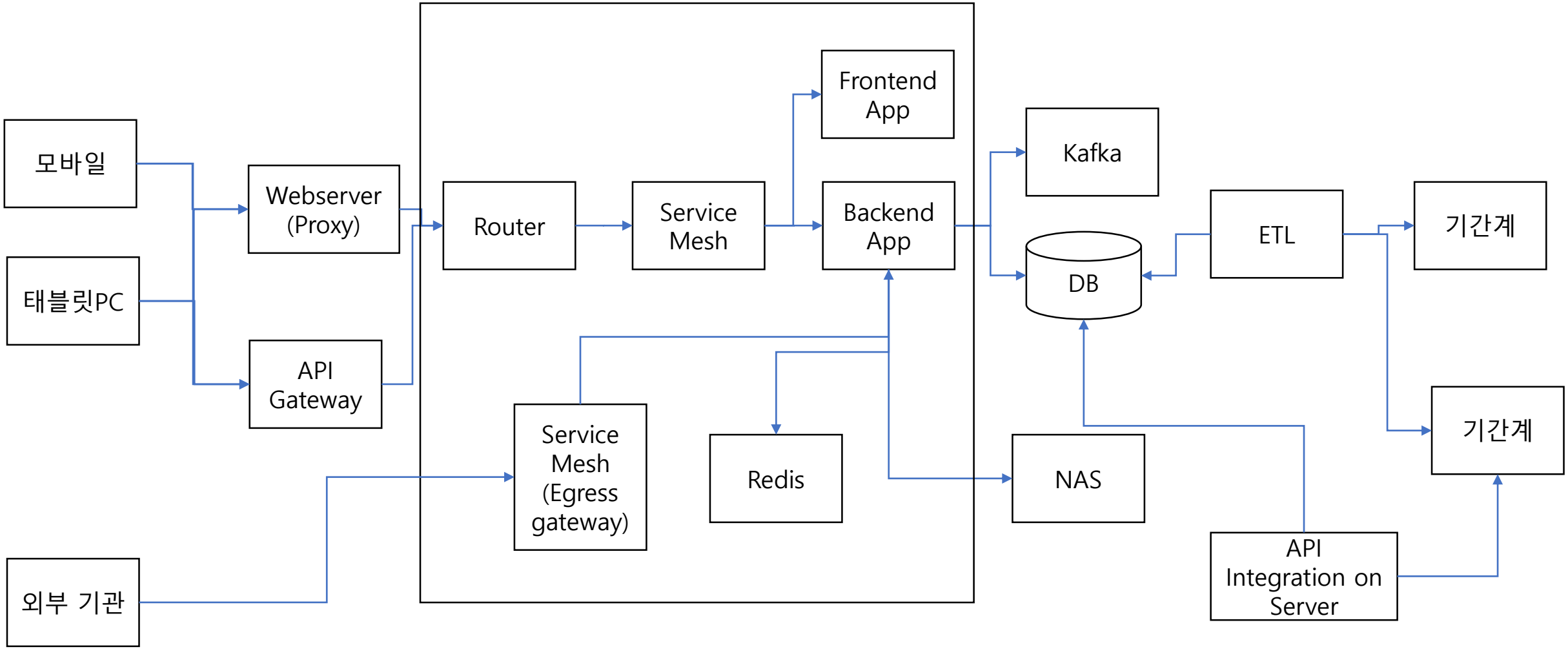
Client	관리 기능		부가 기능	기타
스케줄링 (예약전송)	사용자 관리	사용 이력 로깅	DoS 방지	이중화
재전송 (이어서 전송)	과금	보고서 생성	Client 간 전송	localization

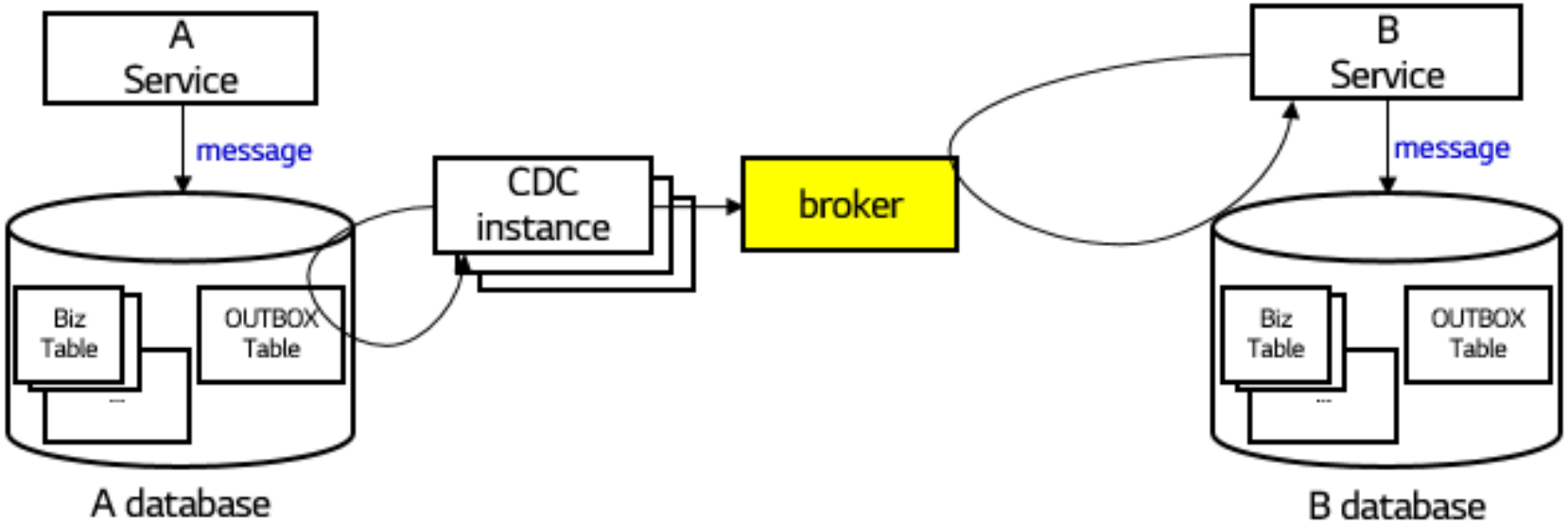
* Executor 처리 흐름

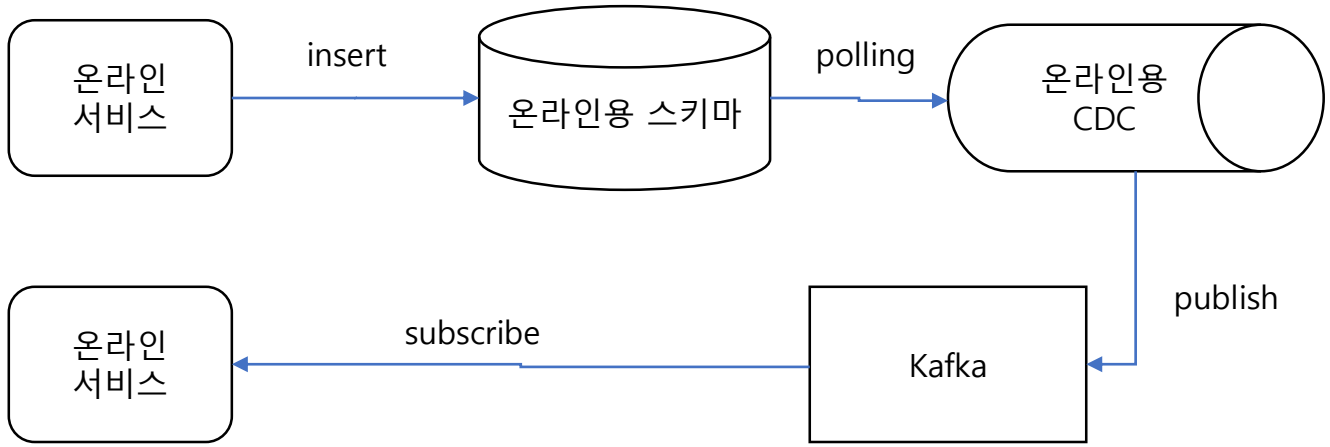


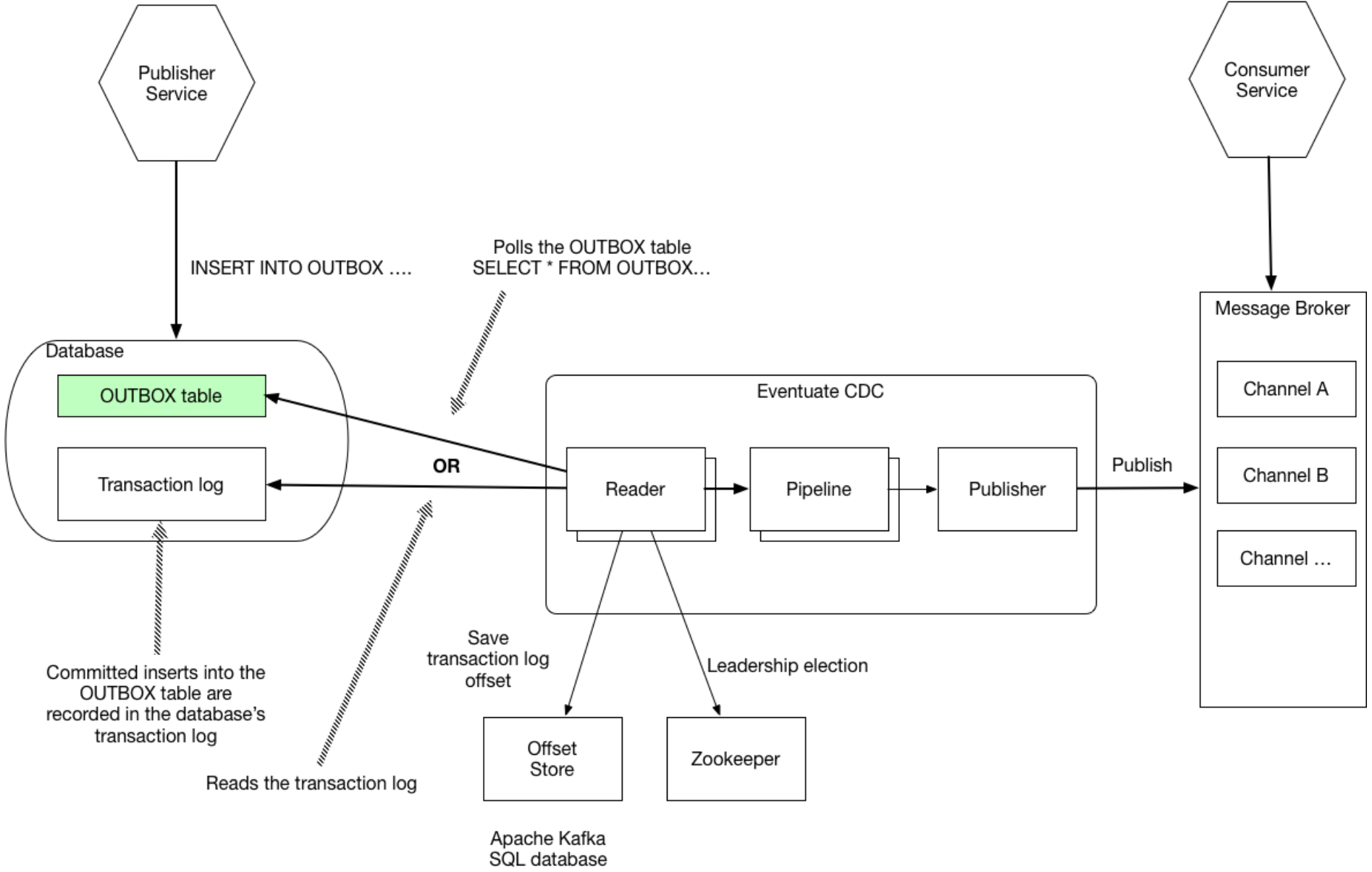

RunManager
ProxyServer
Server
(jetty)
ProxyServlet
extends
HttpServlet
RoutingRule
Reader
CricuitBreaker
TraceManager
RoutingRule
VO
RouteVO
StatusMap
(Map)
TraceMap
(Map)
TraceVO


순서도
시작
Service 추적 API 호출?
Y
Trace 정보 출력
끝
N
회로 차단 상태인가?
Y
retryAfter 시간이 지났는가?
N
Y
healthCheck path 호출
응답이 정상인가?
N
Y
오류 응답 회신
끝
회로 차단 해제
N
Service 호출
Trace 정보 생성
회로 차단 대상인가?
N
Y
응답 회신
끝
회로 차단 상태 업데이트


모바일
태블릿PC
외부 기관
Webserver (Proxy)
API Gateway
Router
Service Mesh
Frontend App
Backend App
Service Mesh (Egress gateway)
Redis
Kafka
DB
NAS
ETL
기간계
기간계
API Integration on Server

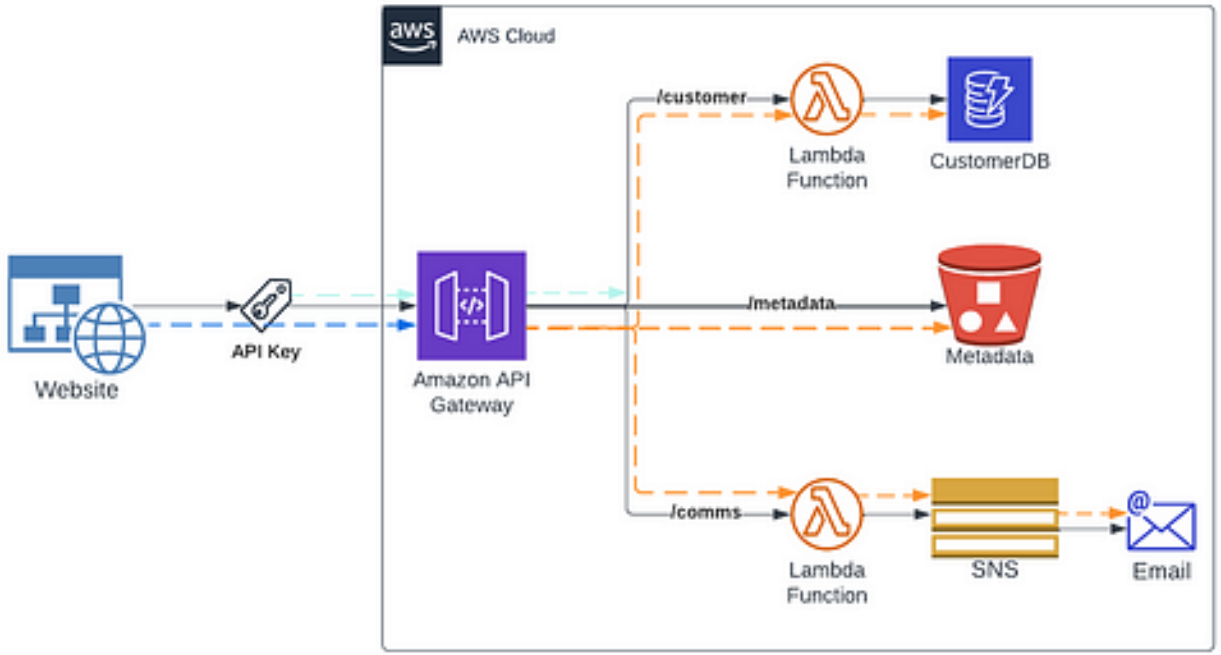

A
Service
message
CDC
instance
broker
B
Service
message
Biz
Table
OUTBOX
Table
...
A database
Biz
Table
OUTBOX
Table
...
B database

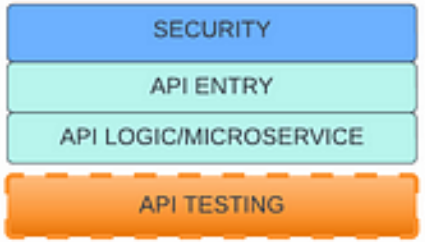

온라인
서비스
insert
온라인용 스키마
polling
온라인용
CDC
publish
온라인
서비스
subscribe
Kafka



https://eventuate.io/docs/manual/eventuate-tram/latest/cdc-configuration.html

API-Driven Architecture with AWS resources
이 예에서는 메타데이터를 캡처하고 고객 데이터를 데이터베이스에 삽입하고 이메일 알림을 보내는 API 엔드포인트를 보여줍니다.

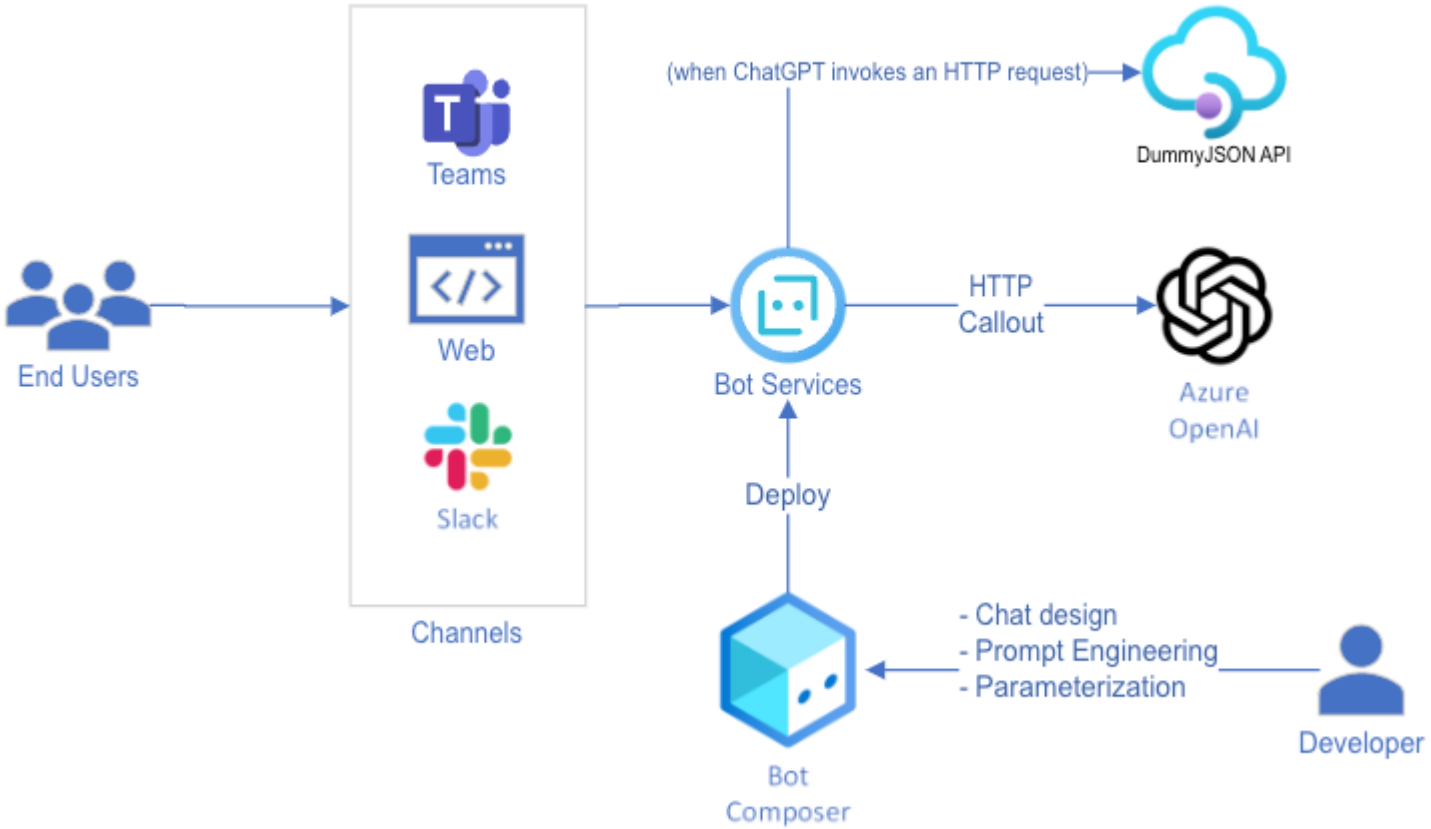




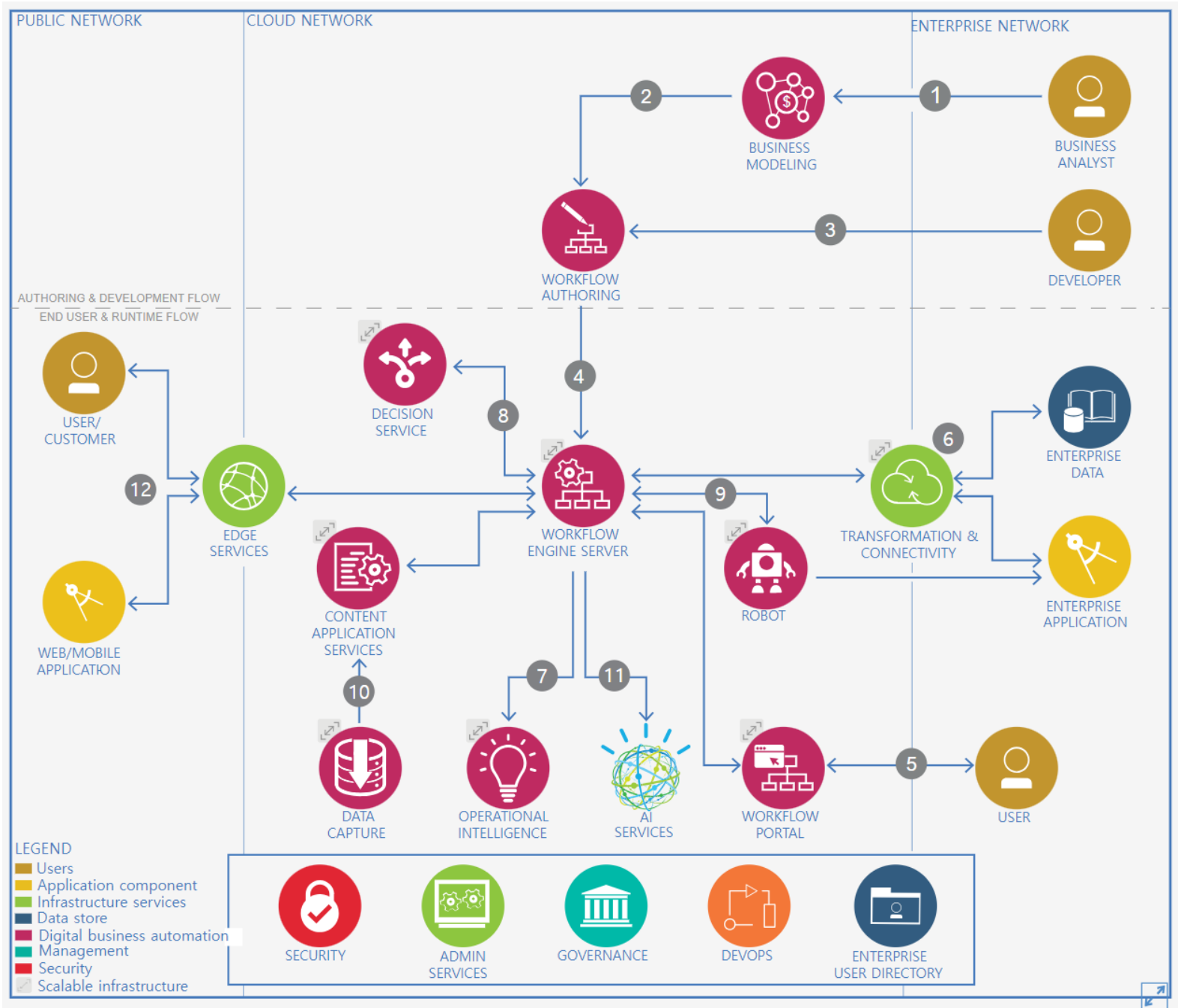
중요한 사항
1. 보안: API 인증
2. API Entry Point: 검증을 통해 올바른 매개변수로 구현된 REST API
3. API 테스트: 각 엔드포인트 경로를 테스트하여 API 응답을 검증
   도구 Postman이 도움.
   이는 API에 배포되어서는 안 됨

ChatGPT를 포함한 OpenAI의 GPT-3는 다음과 같은 방법으로 API 기반 아키텍처에 통합될 수 있습니다.
1. 자연어 처리(NLP) API: 인간과 유사한 텍스트를 생성하고 자연어 입력을 이해하는 ChatGPT의 기능을 NLP API로 활용하여 애플리케이션에서 사용자 요청을 처리하고 이해할 수 있습니다.
2. 텍스트 생성 API: 프롬프트를 기반으로 텍스트를 생성하는 ChatGPT의 기능을 텍스트 생성 API로 활용하여 챗봇, 언어 번역, 콘텐츠 생성과 같은 애플리케이션에서 응답을 생성할 수 있습니다.
3. 질문 응답 API: ChatGPT의 질문 답변 기능은 질문 응답 API로 사용되어 고객 서비스 챗봇이나 가상 비서와 같은 애플리케이션에서 사용자 질문에 대한 답변을 제공할 수 있습니다.

How ChatGPT Plugins (could) work
검색어 : chatgpt integration architecture



https://www.ibm.com/cloud/architecture/architectures/workflowDomain/reference-architecture/



1. 비즈니스 분석가는 LOB(Line of Business) 대표와 협력하여 휴먼 워크플로, 임시 사례 프로세스 또는 직접 프로세스와 같은 비즈니스 프로세스를 검색하고 모델링합니다.
2. 보다 자세한 모델링을 위한 시작점으로 초기 모델을 워크플로 작성 환경으로 가져올 수 있습니다.
3. 워크플로우 작성 및 개발 환경 내에서 개발자는 비즈니스 객체, 사용자 인터페이스, 외부 서비스에 대한 인터페이스, 애플리케이션 및 기록 시스템을 정의합니다.
4. 플랫폼의 DevOps 기능을 사용하면 워크플로 애플리케이션이 워크플로 엔진을 실행하는 서버에 배포됩니다.
5. 워크플로 솔루션 사용자로서 지식 근로자는 워크플로 포털을 통해 비즈니스 프로세스에 참여합니다. 포털은 작업 청구, 작업 및 릴리스, 케이스 작업을 위한 우선순위가 지정된 작업 목록과 사용자 인터페이스를 제공합니다. 동일한 포털 내에서 관리자는 팀의 성과 대시보드와 KPI에 액세스할 수 있습니다.
6. 워크플로 애플리케이션은 일반적으로 외부 기록 시스템, 엔터프라이즈 애플리케이션 또는 서비스와 상호 작용합니다.
7. 운영 인텔리전스 기능은 워크플로 실행과 관련된 이벤트를 캡처하고 확장 가능한 방식으로 기록하며 대화형 도구 및 대시보드를 통해 비즈니스 성과 분석을 제공합니다. 워크플로 애플리케이션은 선택적으로 8~12단계에 표시된 인접 기능을 사용할 수 있습니다.
8. 선택적으로 워크플로 애플리케이션은 의사결정 서비스를 호출하여 의사결정 태스크를 자동화할 수 있습니다.
9. 선택적으로 워크플로 애플리케이션은 로봇의 활성화 및 실행을 조정하여 워크플로의 수동 작업을 자동화할 수 있습니다.
10. 선택적으로 워크플로 애플리케이션은 콘텐츠 관리 플랫폼에서 관리되는 문서와 상호 작용하거나 일부 데이터 캡처 기능을 통해 디지털화된 데이터를 사용하여 상호 작용할 수 있습니다.
11. 선택적으로 워크플로 애플리케이션은 AI 서비스를 호출하여 사용자를 지원하거나 비즈니스 데이터를 강화할 수 있습니다.
12. 대부분의 워크플로 애플리케이션은 내부용이지만 특정 워크플로에는 보안 에지 서비스를 통해 인터넷을 통해 워크플로 솔루션과 상호 작용할 수 있는 외부 참가자나 시스템이 포함됩니다.