1차 프로젝트 데모 발표

LOG MATE

팀원 : 강찬욱, 양승원, 정주연	2025.04.16

Content

목차

Part 1

프로젝트 소개

1. 개요

- 서비스 이름: Log Mate
- 한줄 요약: 간단하게 구성하는 서버 로그 모니터링 & AI 분석 서비스

1. 에이전트 서버 (로그 수집기)

- 로그 파일 수집 및 파싱, 필터링 기능
- 로그 데이터 전송 기능
- 동적 설정 변경 기능

2. 웹 애플리케이션 서버 (데이터 처리 서버)

- 로그 데이터 영구 저장 기능
- 대용량 로그 데이터 수집 기능
- 에이전트 서버 설정 데이터 전송 기능
- 사용자 인증 및 권한 관리 기능 - 팀 대시보드 기능

2. 웹 애플리케이션 서버 (데이터 처리 서버)

- AI 분석 기능 - 웹 로그 이상탐지
- 알림 기능

3. 웹 프론트엔드 서버 (UI 제공 서버)

- 사용자 친화적 UI/UX 제공
- 기능에 대한 각종 UI 제공

기존 솔루션과 비교

항목	**Loggly**	**Datadog**	**ELK Stack (Beats + Logstash)**	**Grafana Loki (Promtail)**	**우리 서비스 (LogMate)**
설정 방식	스크립트 기반	YAML 설정 파일	수동 구성 파일 (다단계)	직접 YAML 작성	**웹 UI로 설정 전송**
초기 필수 설정 항목 수	2~3개	5~6개	수십 개 이상	5~10개	**1개 (수집할 파일 경로)**
고급 설정 기능	제한적 (Syslog 기반)	다양함 (regex, 마스킹 등)	매우 다양 (Grok, Ruby 등)	제한적 (regex, json 파싱 등)	**UI 기반 고급 옵션 제공**
설정 변경 반영 방식	일부 자동 반영	수동 재시작 필요	수동 재시작 필요	수동 재시작 필요	**HTTP 호출로 실시간 반영**
자체 시각화 대시보드 제공	**Saas UI 제공**	**Saas UI 제공**	❌ (Kibana 필요)	❌ (Grafana 별도 필요)	**Saas UI 제공**
AI 기반 기능 제공 여부	없음	이상 탐지, 로그 패턴 인식 등 지원	없음	없음	AI 기반 웹로그 이상탐지 기능 제공

기존 솔루션과 비교

Loki + Promtail 설정 예시

```yaml
server:
    http_listen_port: 9080
    grpc_listen_port: 0

positions:
    filename: /tmp/positions.yaml

clients:
    - url: http://localhost:3100/loki/api/v1/push

scrape_configs:
    - job_name: system
      static_configs:
          - targets: [localhost]
            labels:
                job: varlogs
                host: myhost
                __path__: /var/log/*.log
      pipeline_stages:
          - multiline:
                firstline: '^\d{4}-\d{2}-\d{2}'
          - regex:
                expression: '^(?P<level>\w+): (?P<msg>.*)'
          - labels:
                level:
          - output:
                source: msg
```

ELK Stack - Filebeat + Logstash 예시

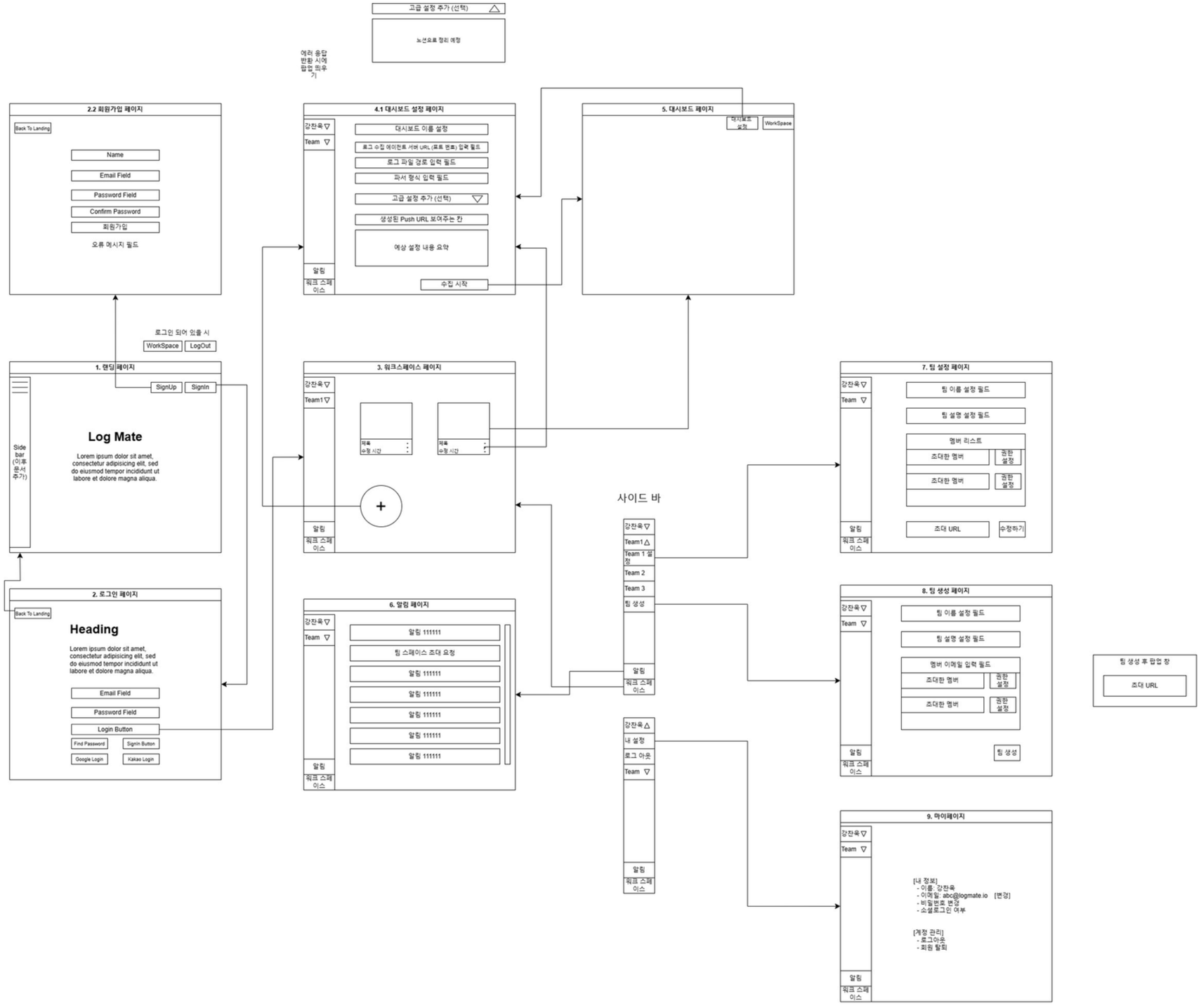
```yaml
filebeat.inputs:
    - type: log
      enabled: true
      paths:
          - /var/log/*.log
      multiline.pattern: '^\['
      multiline.negate: true
      multiline.match: after

output.logstash:
    hosts: ["localhost:5044"]
```

```
input {
    beats {
        port => 5044
    }
}

filter {
    grok {
        match => { "message" => "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:level} %{GREEDYDATA:msg}" }
    }

    mutate {
        remove_field => [ "beat", "host", "input_type" ]
    }

    date {
        match => [ "timestamp", "ISO8601" ]
    }
}

output {
    elasticsearch {
        hosts => ["localhost:9200"]
        index => "logs-%{+YYYY.MM.dd}"
    }
```

기존 솔루션과 비교

사용자 UI를 통한 원격 설정

화면 설계


고급 설정 추가 (선택)
노션으로 정리 예정
여러 응답 반환 시에 팝업 띄우기
2.2 회원가입 페이지
Back To Landing
Name
Email Field
Password Field
Confirm Password
회원가입
오류 메시지 필드
4.1 대시보드 설정 페이지
강잔욱
Team
대시보드 이름 설정
로그 수집 에이전트 서버 URL (포트 번호) 입력 필드
로그 파일 경로 입력 필드
파서 형식 입력 필드
고급 설정 추가 (선택)
생성된 Push URL 보여주는 칸
예상 설정 내용 요약
수집 시작
알림
워크 스페이스
5. 대시보드 페이지
대시보드 설정
WorkSpace
로그인 되어 있을 시
WorkSpace
LogOut
1. 랜딩 페이지
SignUp
SignIn
Side bar (이후 문서 추가)
Log Mate
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
3. 워크스페이스 페이지
Team1
제목
수정 시간
+
7. 팀 설정 페이지
팀 이름 설정 필드
팀 설명 설정 필드
멤버 리스트
초대한 멤버
권한 설정
초대 URL
수정하기
2. 로그인 페이지
Back To Landing
Heading
Email Field
Password Field
Login Button
Find Password
Signin Button
Google Login
Kakao Login
6. 알림 페이지
알림 111111
팀 스페이스 초대 요청
사이드 바
Team1
Team 1 설정
Team 2
Team 3
팀 생성
내 설정
로그 아웃
8. 팀 생성 페이지
팀 이름 설정 필드
팀 설명 설정 필드
멤버 이메일 입력 필드
팀 생성
팀 생성 후 팝업 창
초대 URL
9. 마이페이지
[내 정보]
- 이름: 강잔욱
- 이메일: abc@logmate.io [변경]
- 비밀번호 변경
- 소셜로그인 여부
[계정 관리]
- 로그아웃
- 회원 탈퇴

Part 2

1차 데모 구현 내용 및 목표

1. 트래픽 처리

- 2명의 사용자가 동시에 접속하여 로그를 전송하고, 웹에서 확인 가능
- 향후 추가적인 사용자 확장을 고려하여 설계

2. 구현

- 로그를 수집하고 서버로 전송 처리하는 에이전트 서버 구현
- 수집된 로그를 데이터베이스에 저장
- 실시간 로그 확인 대시보드 구현
- 웹 UI 로 에이전트 서버의 설정 변경 후 런타임 중 자동 적용되는 구조 구현

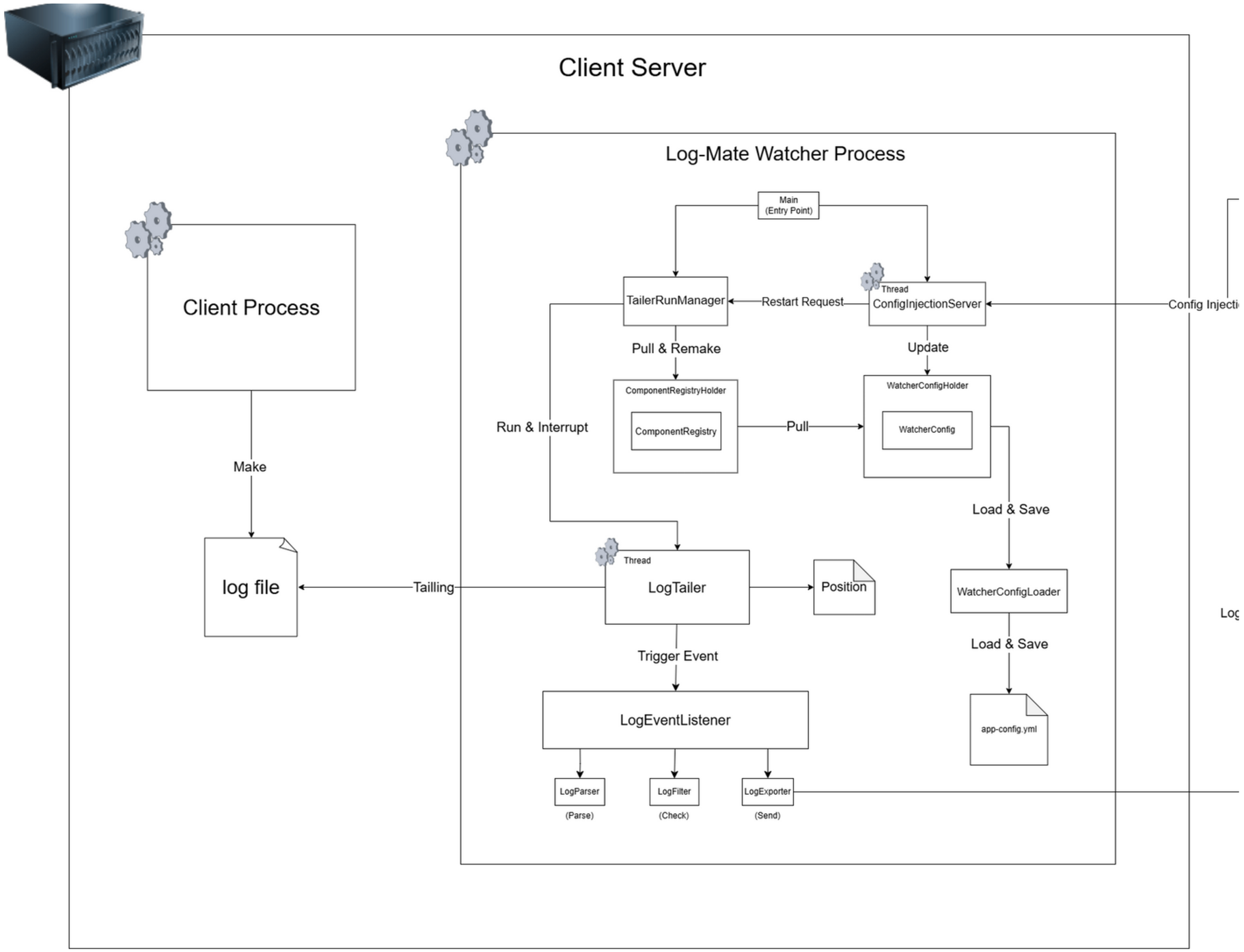
1. 에이전트 서버 구현 - Log Watcher

1) 1차 데모 구현 기능

- 로그 파일 Tail 처리
- 사용자 친화적 웹 UI 를 통한 설정 연동
- 런타임 중 설정 변경사항 적용
- HTTP 기반 로그 전송
- 무설정으로 실행

2) 기술 스펙

- **언어:** Java 17
- **파일 읽기 방식:** RandomAccessFile 기반 처리
- **설정 방식:** HTTP API 요청으로 동적 설정 수신
- **설정 항목:** logPushURL, logFilePath
- **로그 오프셋 저장:** 별도 메타파일 생성 및 쓰기 처리
- **로그 전송 방식:** HTTP POST, Json Body 전송
- **로그 전송 형식:** Jackson 라이브러리를 이용한 객체 직렬화
- **외부 의존성:** Jackson (객체 직렬화), SLF4J (로깅)
- **Docker 지원:** 로그 파일 마운트 시 사용 가능. 도커 이미지 지원


Client Server
Log-Mate Watcher Process
Main
(Entry Point)
TailerRunManager
Restart Request
Thread
ConfigInjectionServer
Config Injecti
Client Process
Pull & Remake
Update
ComponentRegistryHolder
ComponentRegistry
Pull
WatcherConfigHolder
WatcherConfig
Run & Interrupt
Make
Load & Save
log file
Tailling
Thread
LogTailer
Position
WatcherConfigLoader
Log
Load & Save
Trigger Event
LogEventListener
app-config.yml
LogParser
(Parse)
LogFilter
(Check)
LogExporter
(Send)

2. 웹어플리케이션 서버 구현

1) 1차 데모 구현 기능

- 수집된 로그의 HTTP 요청을 통한 수신 및 영구 저장 처리
- WebSocket을 통한 실시간 로그 스트리밍
- 사용자별 대시보드 제공
- 사용자 에이전트 서버에 HTTP 요청을 통한 설정 정보 전달

2) 기술 스펙

- **언어/프레임워크:** Java 17 / Spring Boot 3.3.10
- **웹 서버:** Apache Tomcat (Spring Boot Embedded)
- **라이브러리:** Spring Web, Spring WebSocket, JPA, Thymeleaf
- **데이터베이스:** MySQL
- **로그 수신**: POST API 를 통해 파싱된 로그 배열 수신 처리
- **실시간 통신:** Spring WebSocket + STOMP 통한 메시지 중계
- **동적 대시보드 제작:** 템플릿엔진을 이용하여 동적 HTML 생성

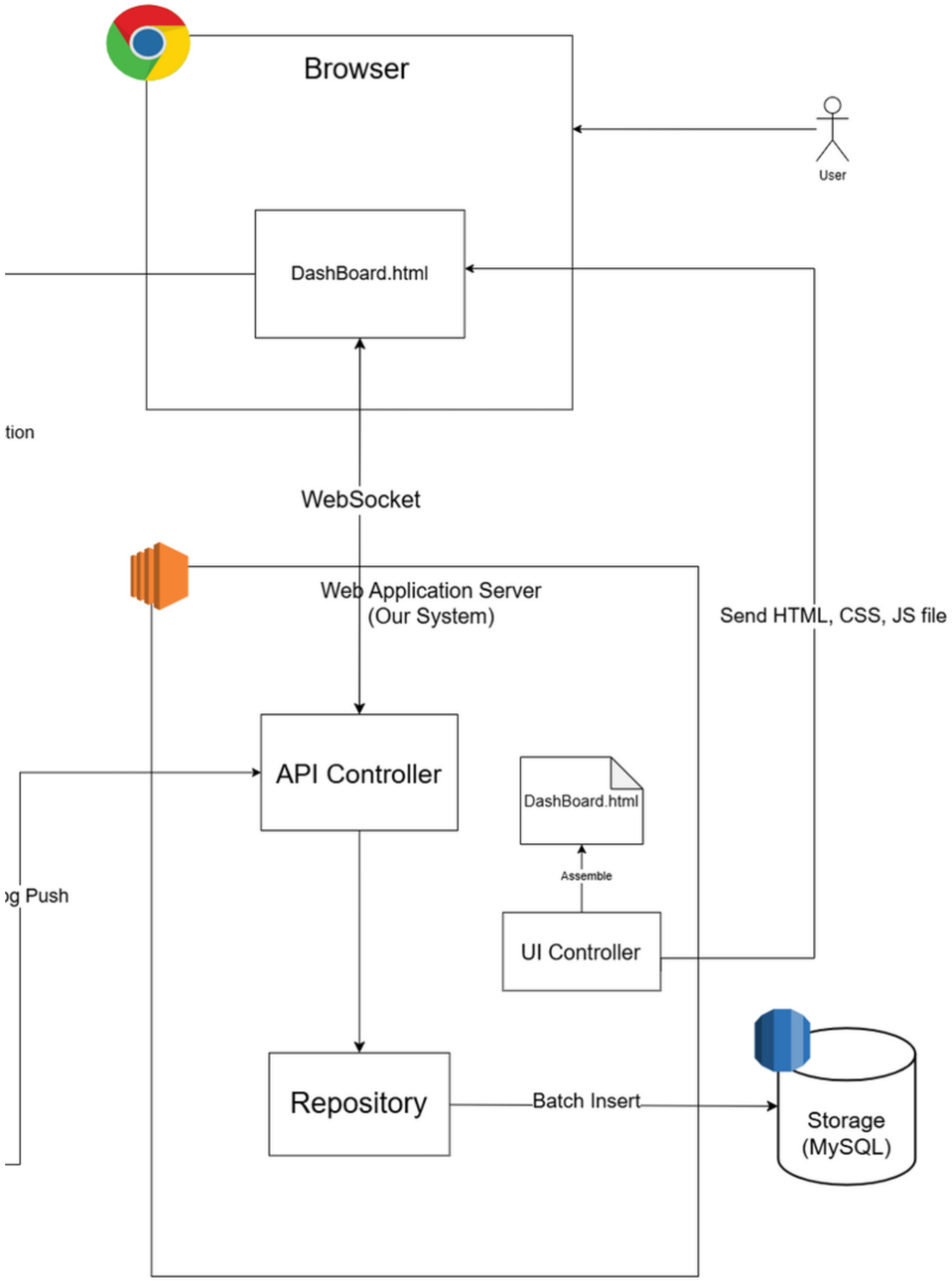
3. 프론트엔드 구현

1) 1차 데모 구현 기능

- 사용자별 대시보드 구현
- 설정 입력 UI 구현
- 실시간 로그 표시

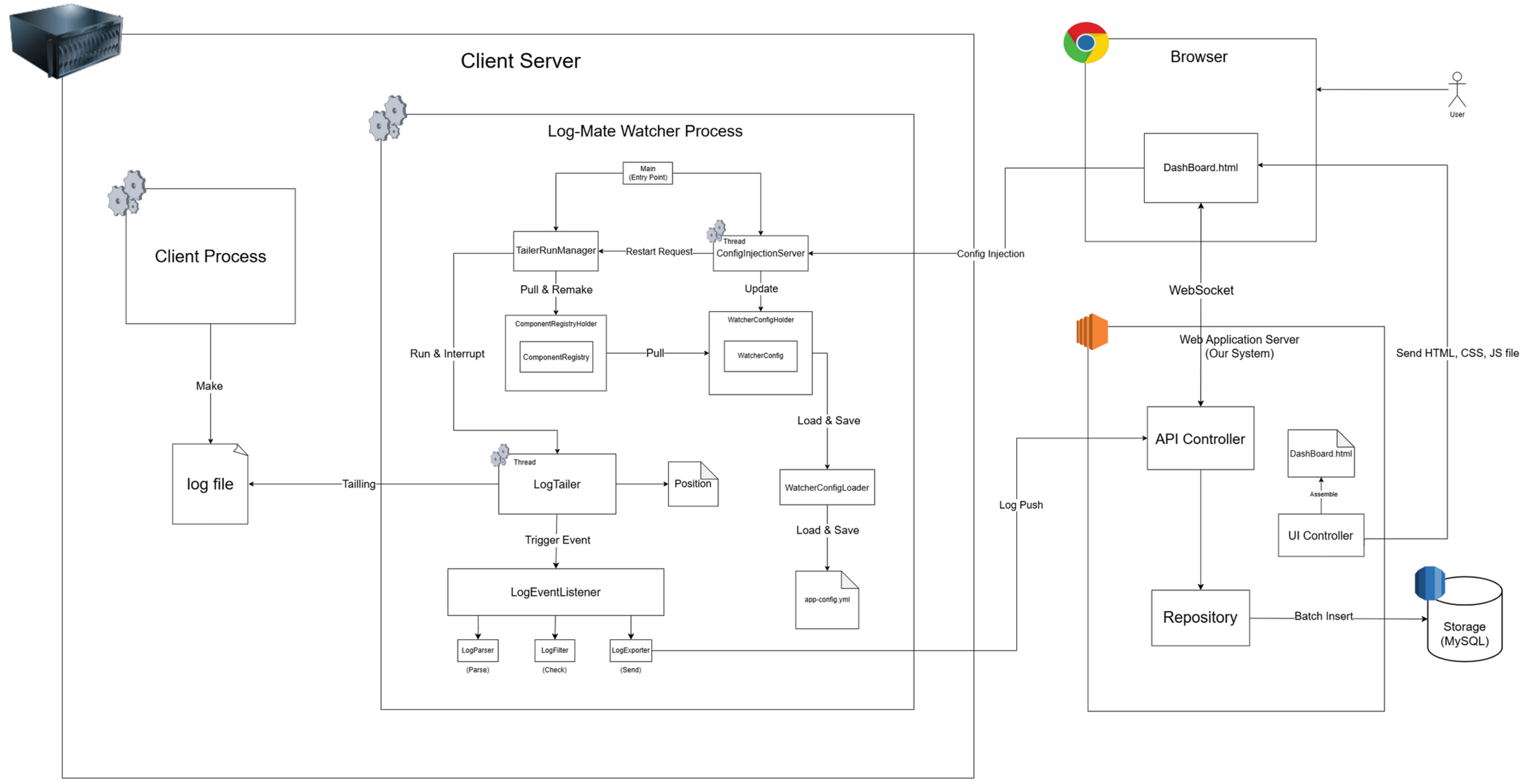
2) 기술 스펙

- **UI 기술 및 언어:** HTML5, JavaScript, CSS
- **실시간 통신**: STOMP, SockJS
- **설정 변경 요청:** HTTP API


Browser
User
DashBoard.html
tion
WebSocket
Web Application Server
(Our System)
Send HTML, CSS, JS file
API Controller
DashBoard.html
Assemble
g Push
UI Controller
Repository
Batch Insert
Storage
(MySQL)

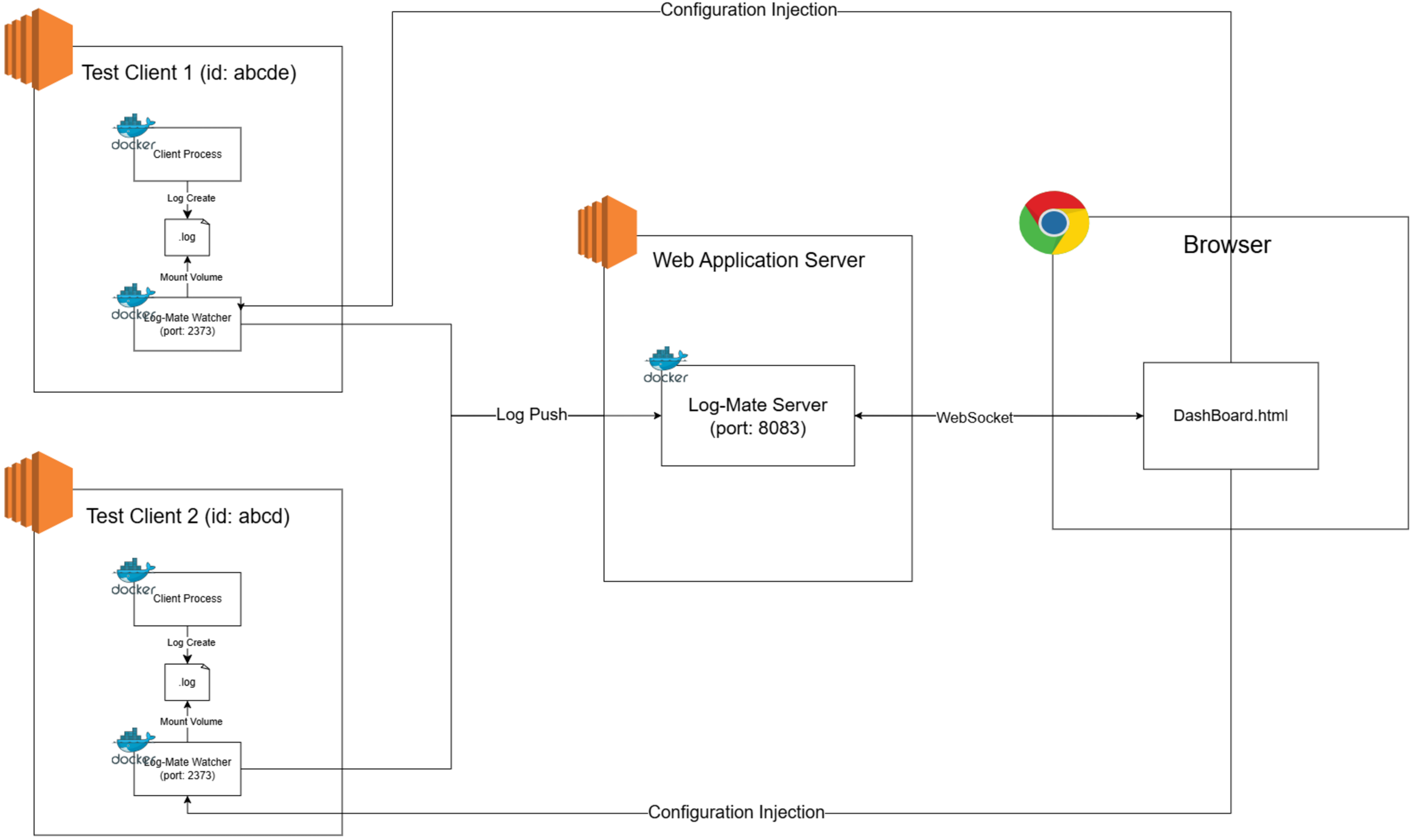
Part 3

1차 데모 구현 아키텍쳐


Client Server
Log-Mate Watcher Process
Main
(Entry Point)
TailerRunManager
Restart Request
Thread
ConfigInjectionServer
Pull & Remake
Update
ComponentRegistryHolder
ComponentRegistry
Pull
WatcherConfigHolder
WatcherConfig
Run & Interrupt
Load & Save
Client Process
Make
log file
Tailling
Thread
LogTailer
Position
WatcherConfigLoader
Trigger Event
Load & Save
LogEventListener
app-config.yml
LogParser
(Parse)
LogFilter
(Check)
LogExporter
(Send)
Config Injection
Browser
User
DashBoard.html
WebSocket
Web Application Server
(Our System)
Send HTML, CSS, JS file
API Controller
DashBoard.html
Assemble
UI Controller
Log Push
Repository
Batch Insert
Storage
(MySQL)

Part 4

1차 데모 시연


Configuration Injection
Test Client 1 (id: abcde)
Client Process
Log Create
.log
Mount Volume
Log-Mate Watcher
(port: 2373)
Web Application Server
Log-Mate Server
(port: 8083)
Log Push
WebSocket
Browser
DashBoard.html
Test Client 2 (id: abcd)
Client Process
Log Create
.log
Mount Volume
Log-Mate Watcher
(port: 2373)
Configuration Injection

Part 5

개선점 및 2차 데모 목표

1) 개선점

- **보안:** 설정정보, 로그 등 민감한 정보등이 네트워크를 통해 전달되다 보니 보안 이슈가 발생할 수 있을거라 생각
- **성능 최적화:** 로그 수집에 있어 비동기 처리가 필요
- **AI 기능 기획 추가**: AI 활용 방안 추가 기획 필요
- **주입된 설정 저장 기능 추가:** 설정 정보를 영구적으로 저장하는 기능 추가 필요
- UI 로 설정 주입 시 에러 메시지 UI로 출력

2) 목표

- 웹어플리케이션 서버 데이터 처리 서비스 추가 구현
- 웹 UI 추가 구현

감사합니다