Logcat 도구 진행상황 및 논문 계획

2025.02.11

모바일시스템공학과 조민혁
소프트웨어학과 이승민, 정성원

INDEX

01 도구 진행 상황 및 논문 계획

도구 진행 상황 및 논문 계획

❖특허 인터뷰 일정

- 2/19 (수) 오후 1시 15분 ~
- 장소: ICT 관 510호 (교수님 연구실)

❖저널 투고 계획

- Digital Investigation
- Computers & Security
- 논문 제목은 아직 미정 ..

1. Introduction

① 디지털 포렌식과 안티 포렌식+ 로그 수집 및 분석의 중요성 언급

② 안드로이드 시스템에서는 logcat을 통한 로그 수집 및 분석이 가능함을 언급 ; logcat의 중요성 언급

③ logcat의 단점을 언급 (휘발성, 링버퍼, 전원 재부팅 및 꺼짐) ; 연구 배경

④ 이에 따라 휘발성 로그를 비휘발성 로그로써 저장할 필요 있음을 언급 ; 연구 목표 및 기여점

⑤ 기존 논문들의 한계점 언급

⑥ 논문의 기여도를 언급 ; 이 논문을 통해 포렌식 적으로 어떠한 기여가 있을 것이다.

⑦ 각 섹션 언급 ; 논문의 구성

*궁금한 점: bugreport 언급을 해야할 지?

2. Related Work

① Android forensics_ Automated data collection and reporting from a mobile device

-> Digital Investigation 논문

② A design science approach to developing an integrated mobile app forensic framework

-> ScienceDirect 논문

③ Forensic_Analysis_of_Popular_Social_Media_Applicat

-> ELECTRICAL & COMPUTER ENGINEERING 논문

④ The Android Forensics Automator (AnForA)_ A tool for the Automated Forensic Analysis of Android Applications

-> Computers & Security 논문

3. An Effective Logging Technique (DroidMonitor || DroidLogger)

3-1) Our Logging System Structure (전체적인 구성)

① Android Phone, Remote Storage Server, (if exist, Analyzer), Communication

② 도구의 프로세스

③ Back-End Architecture

3. An Effective Logging Technique (DroidMonitor || DroidLogger)

3-2) A process for collecting and analyzing log data

① 수집 가능한 Data Set (Timestamp 조작, logcat -c, Shutdown, File MetaData, Calling, SMS, BT, …)

② 각 Data Set에 대한 브로드캐스트 존재여부, 우회 방법 테이블로 제시

③ 어떻게 로그를 수집할 것인지? (Broadcast, Observer, 우회 방법)

④ 각 이벤트들에 대한 탐지 및 복구가 가능하다면 프로세스 제시, 불가능하면 단순히 탐지 프로세스 제시

⑤ Anti-Forensic 탐지 후 Anti-Forensic Behavior를 무력화하는 방안 제시

⑥ 분석 프로세스 제시 ; (로그 결과를 통한 분석, 해시값을 이용한 유효성 검증, 기기와 서버의 교차 검증 …)

4. Experiment and Evaluation

① 실험 시스템 제시 (Android 14, Rooting x)

② 포렌식 관점에서 앱 관련 데이터를 수집하고 분석하는 실험 수행하여 결과 보이고 평가

③ 각 안티 포렌식 행위들에 대한 시나리오 작성하여 각각에 대해 실험

- 타임스탬프 별로 시나리오 제시 및 실험
- 기법을 탐지하는 실험 수행하여 결과 보이고 평가
- 기기 내부 저장소에 대해 저장 후 실시간으로 서버로 전송됨을 보임; 성공적으로 실험이 이루어졌음을 보임

④ 3장에서 제시한 분석 프로세스를 통해 유효성 검증

5. Discussion

① 결과 화면을 통해 수사 시 활용 방안, 또는 필요한 곳 (보험사와 같이 ..) 에서 활용 방안 제시

② 제안 기법과 기존 연구와 비교 분석 ; 장점 언급

③ 제안 기법의 한계점 언급

6. Conclusion and Future Work

① 향후 자동차와 연계를 통해 발전 시킬 것을 언급

- 스마트폰과 자동차와의 통신, 자동차 원격 제어 앱, 내비게이션 APP 등등

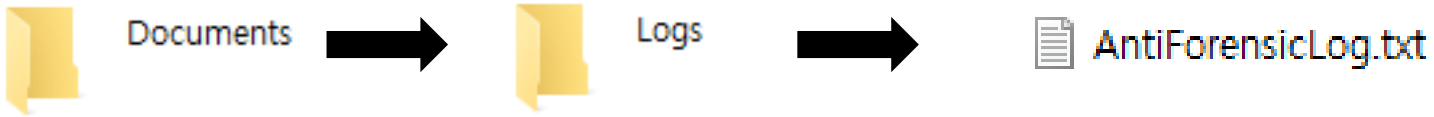
② 또한 커널을 커스텀하여 디폴트 도구로 만들어 발전 시킬 것임을 언급

새롭게 추가된 기능

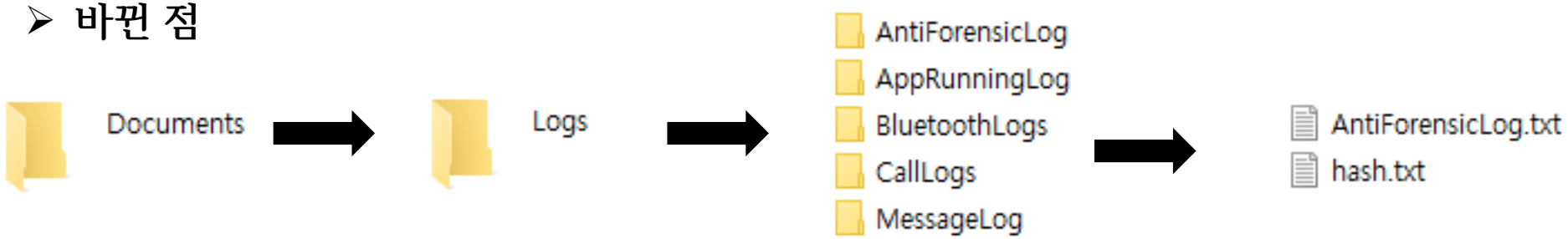
❖ 로그의 디렉토리 저장 + 해시값(SHA-512) 파일 추가

✓ 기기 내부 저장소

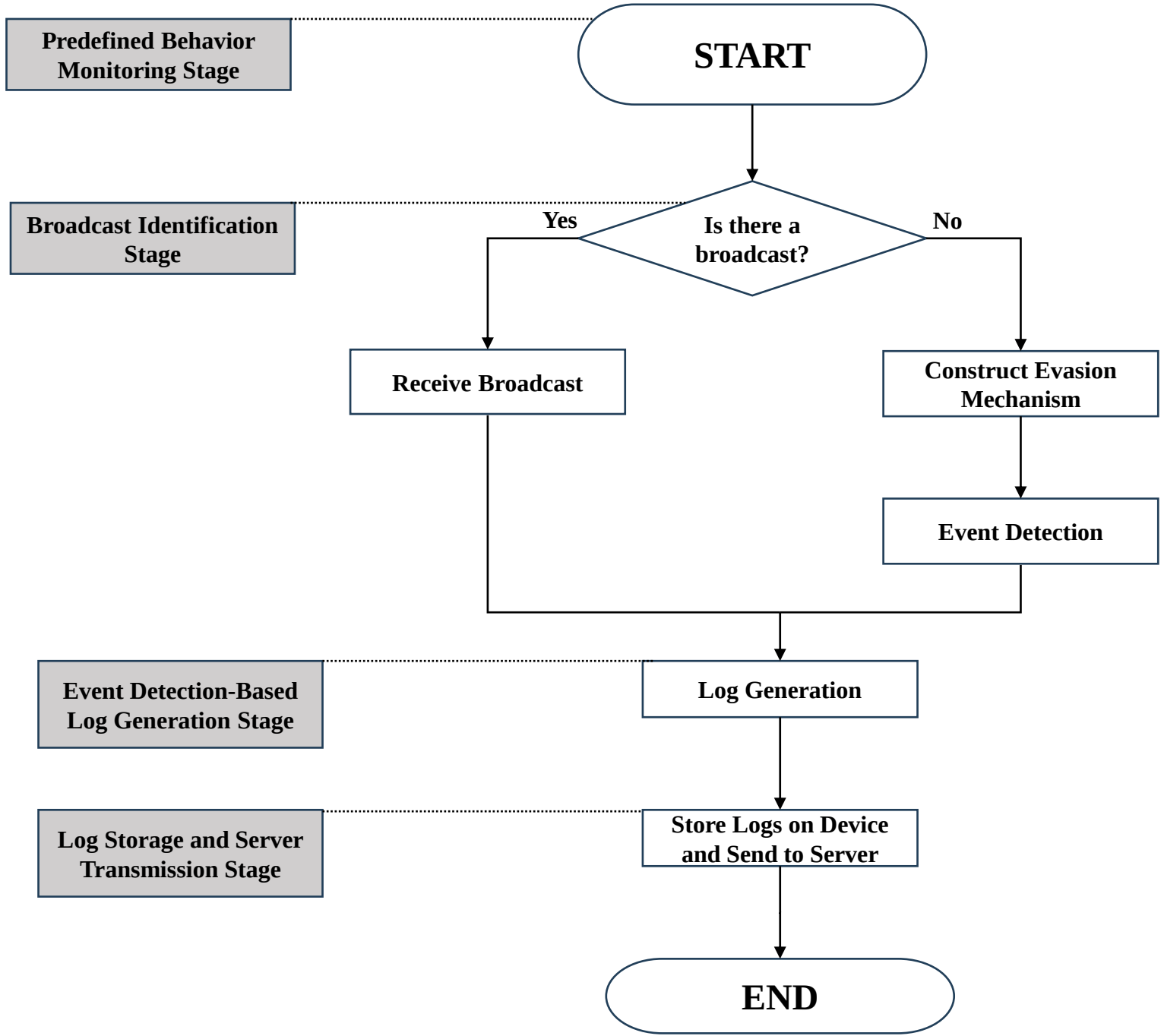
➢ 기존



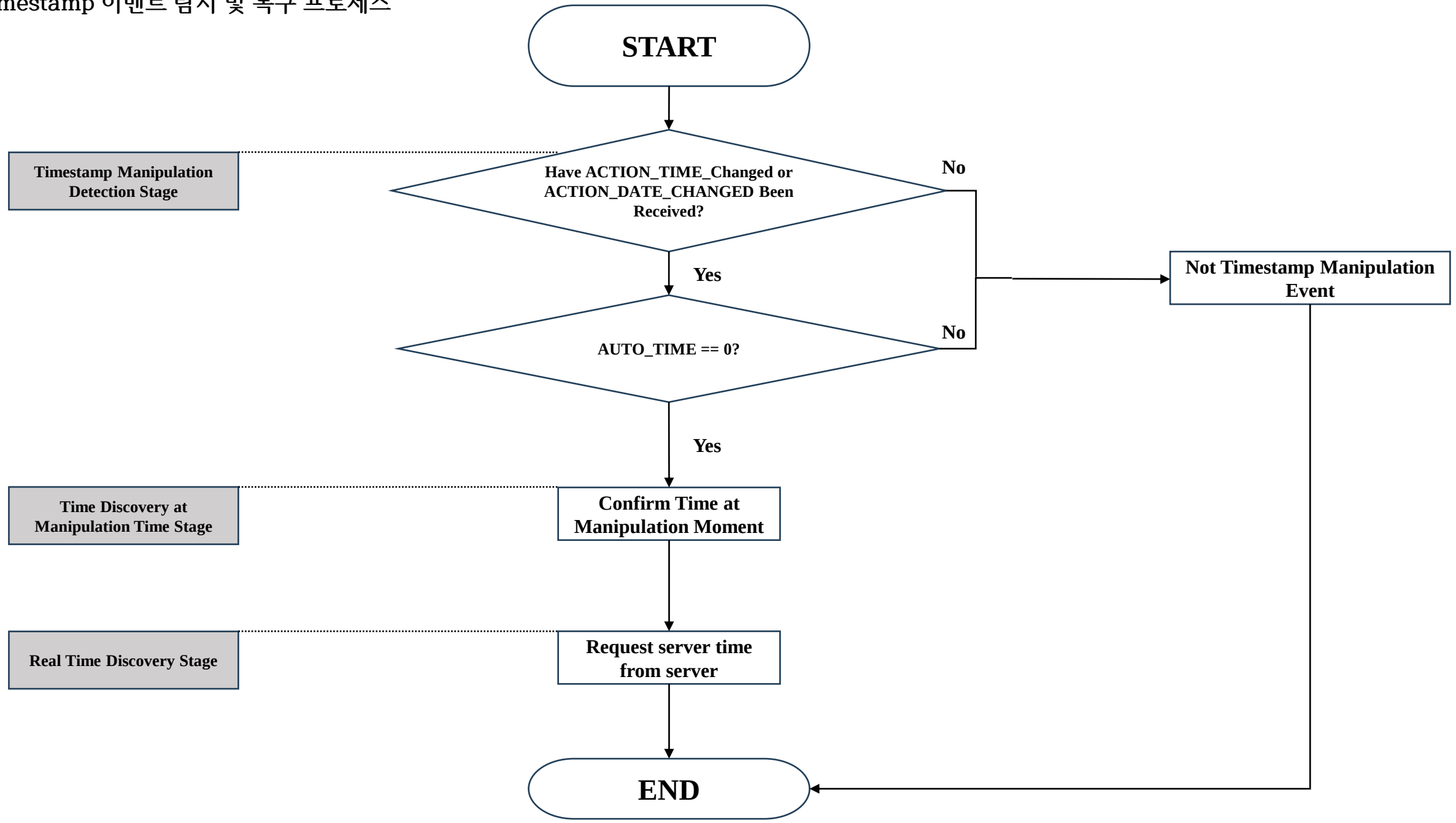
➢ 바뀐 점



✓ Logcat 도구의 프로세스



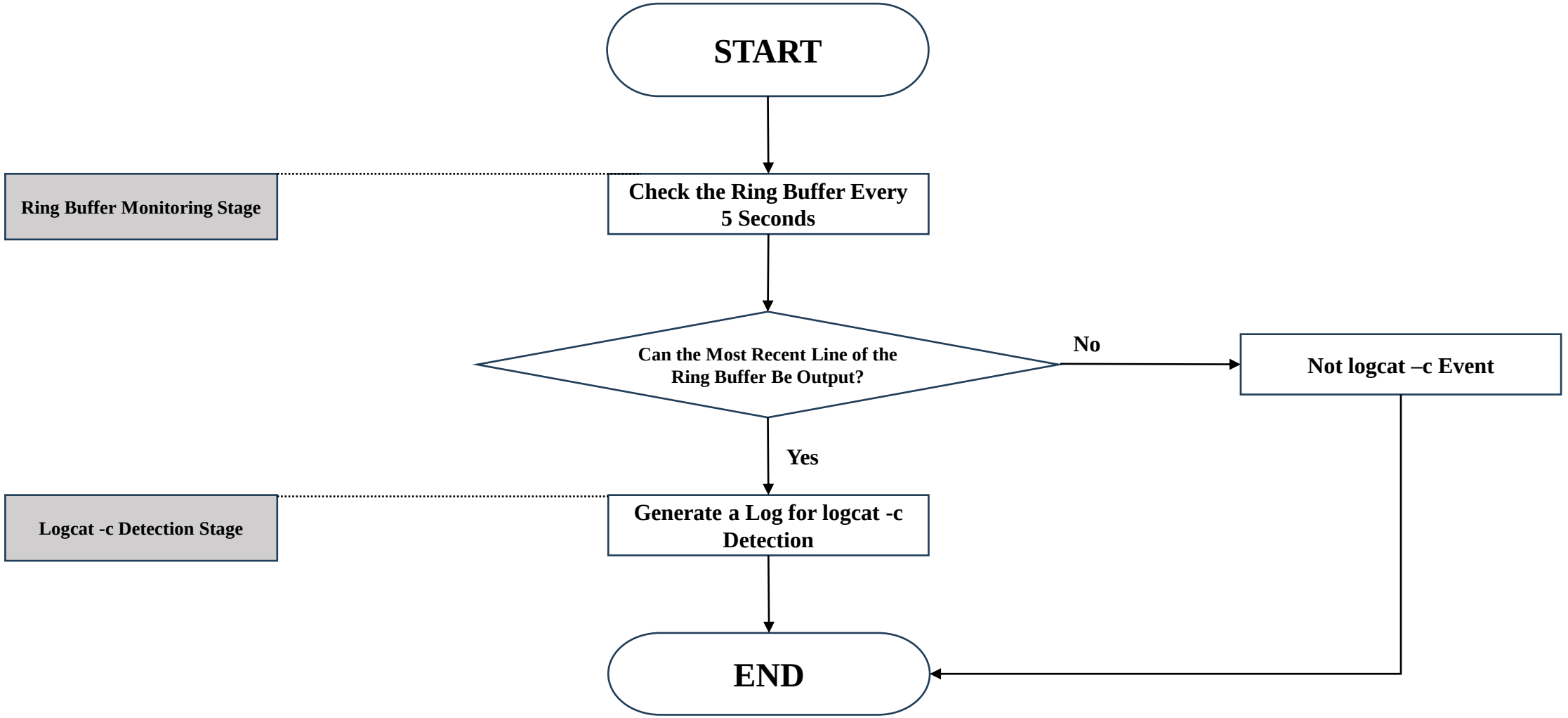
✓ Timestamp 이벤트 탐지 및 복구 프로세스



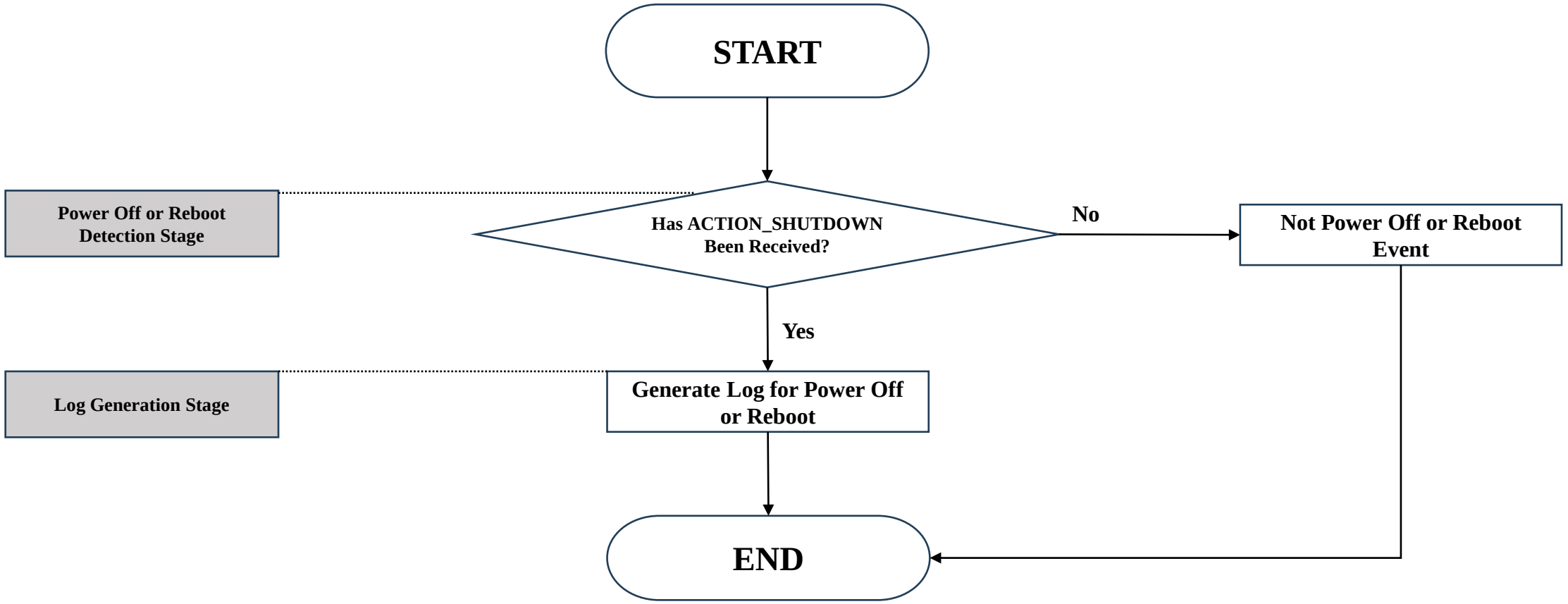
✓ Logcat -c 프로세스

-> 오탐 존재 가능.

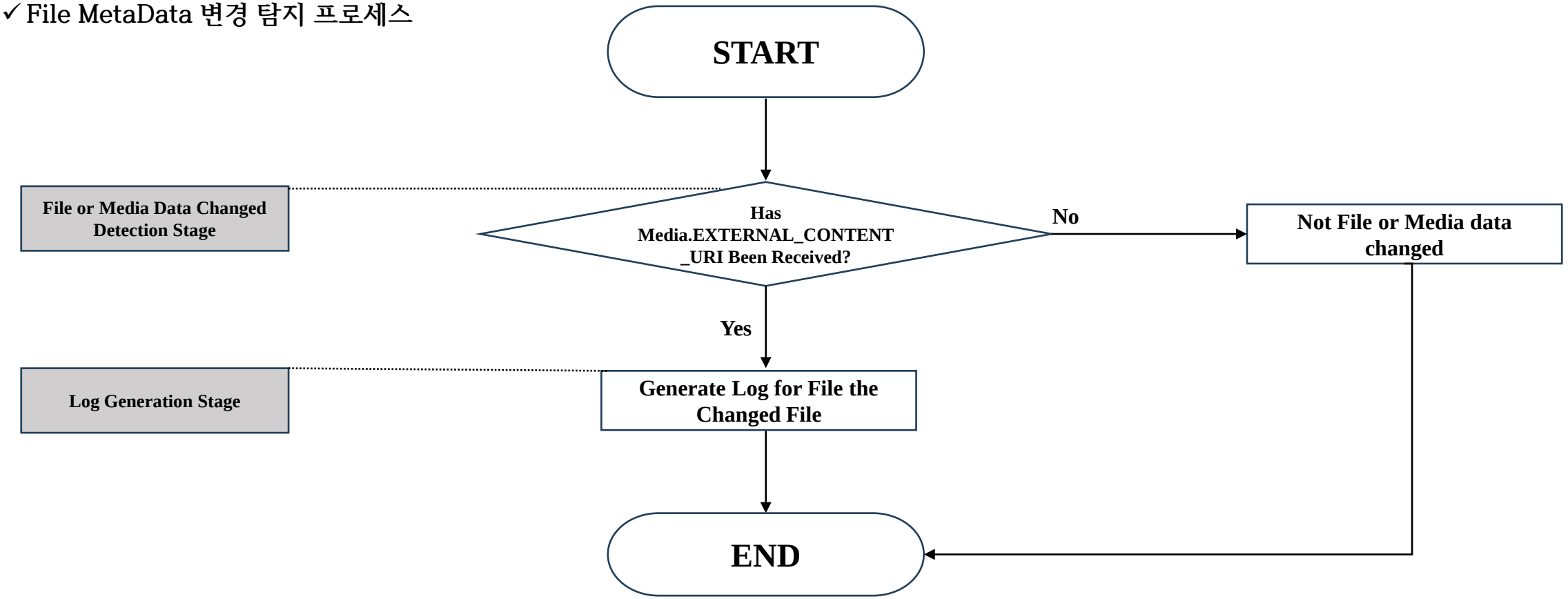
-> 한번에 링버퍼를 꽉차게 하는 이벤트 있었는 지 검증 필요



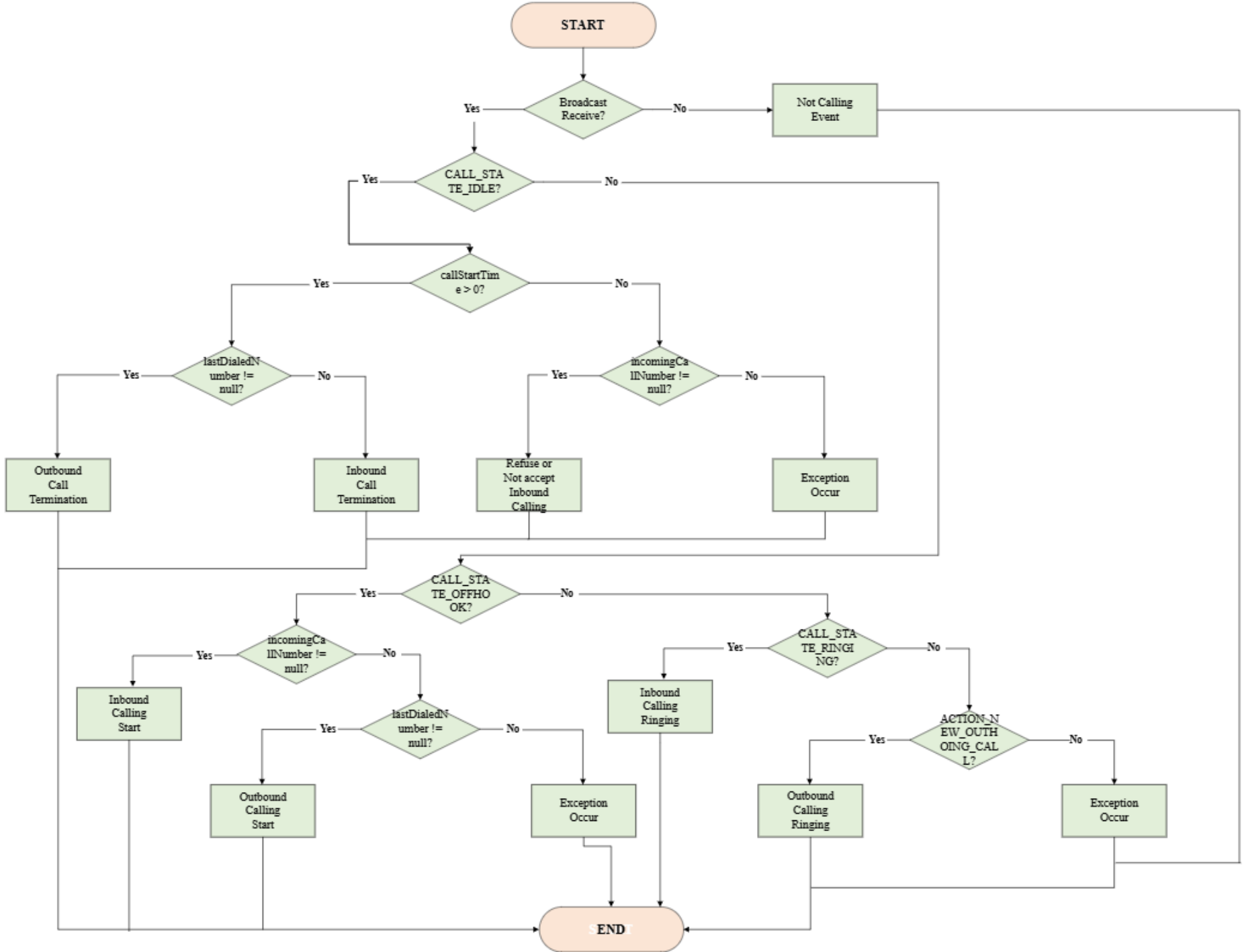
✓전원 꺼짐 및 재부팅 프로세스



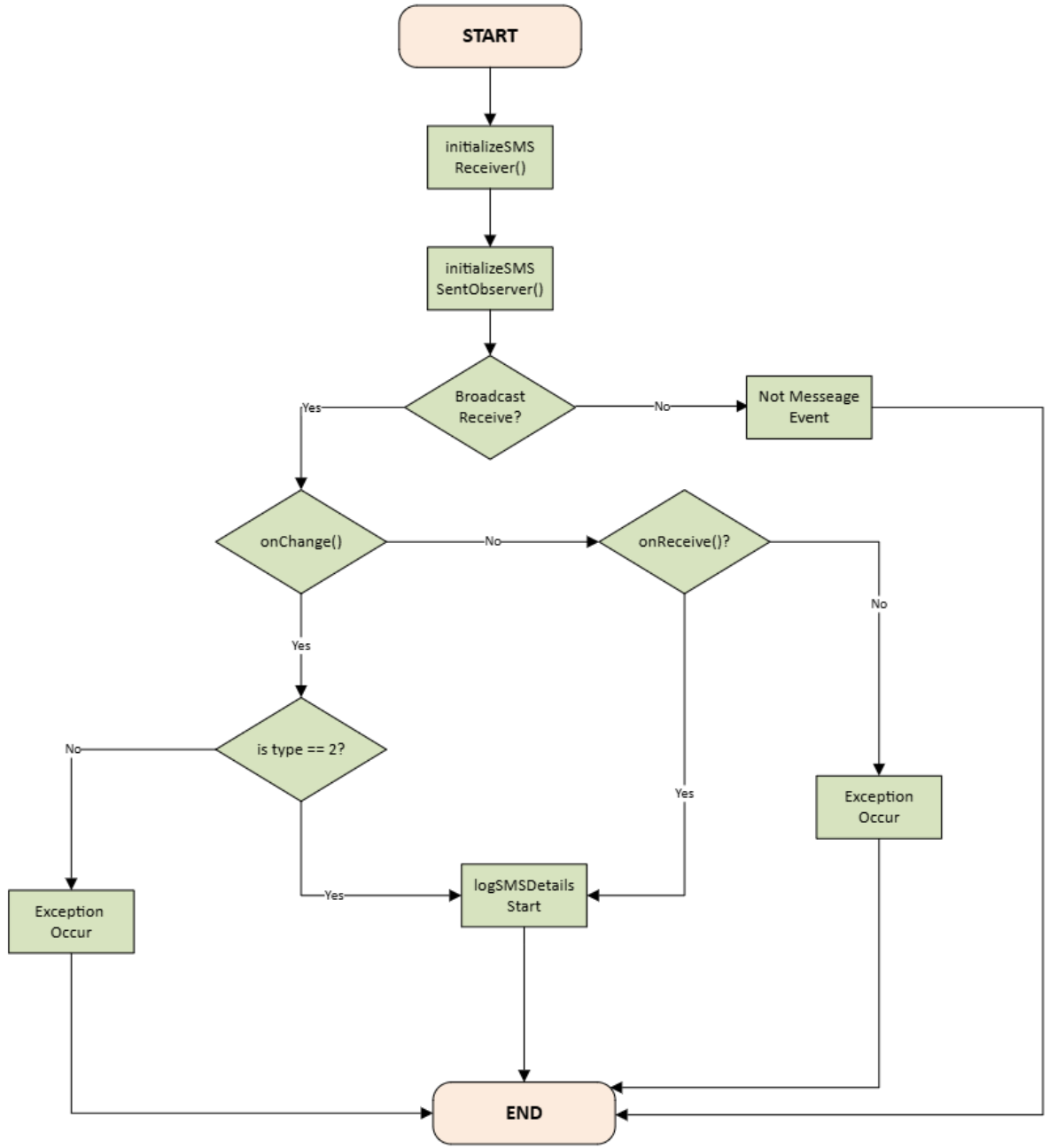
✓ File MetaData 변경 탐지 프로세스



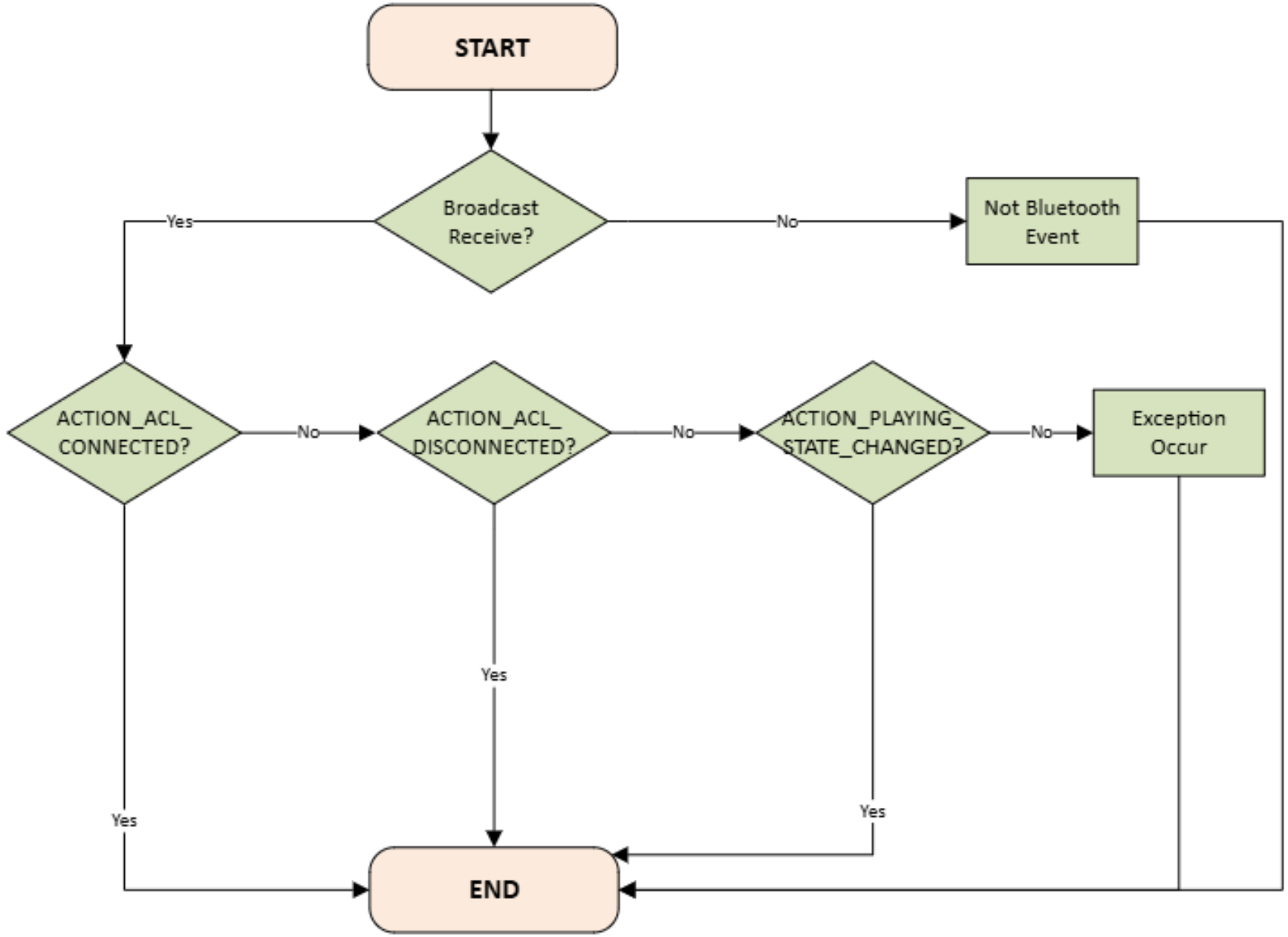
✓전화 프로세스


START
Broadcast Receive?
Yes
No
Not Calling Event
CALL_STATE_IDLE?
Yes
No
callStartTime > 0?
Yes
No
lastDialedNumber != null?
Yes
No
incomingCallNumber != null?
Yes
No
Outbound Call Termination
Inbound Call Termination
Refuse or Not accept Inbound Calling
Exception Occur
CALL_STATE_OFFHOOK?
Yes
No
incomingCallNumber != null?
Yes
No
CALL_STATE_RINGING?
Yes
No
Inbound Calling Start
lastDialedNumber != null?
Yes
No
Inbound Calling Ringing
ACTION_NEW_OUTHOING_CALL?
Yes
No
Outbound Calling Start
Exception Occur
Outbound Calling Ringing
Exception Occur
END

✓문자 프로세스


START
initializeSMS Receiver()
initializeSMS SentObserver()
Broadcast Receive?
Yes
No
Not Messeage Event
onChange()
No
onReceive()?
Yes
No
Yes
is type == 2?
No
Exception Occur
Yes
logSMSDetails Start
Exception Occur
END

✓블루투스 프로세스


START
Broadcast Receive?
Yes
No
Not Bluetooth Event
ACTION_ACL_ CONNECTED?
No
ACTION_ACL_ DISCONNECTED?
No
ACTION_PLAYING_ STATE_CHANGED?
No
Exception Occur
Yes
Yes
Yes
END

02 논의 사항

논의 사항

1. 언제부터 논문을 쓰는 게 좋을까요? (초안)

2. Anti-Forensic을 무력화 하는 기법이 잘 생각나지 않는데 .. 어떻게 하는게 좋을까요?

3. 문자, 파일 메타데이터는 조금 더 수정 필요해서 수정할 예정입니다.

03 Spring Boot/Docker 기반 백엔드 구성

추가 구현 및 변경 사항

❖ **Multi-User와 Log 데이터 저장을 위한 MongoDB(NoSQL) 연결**

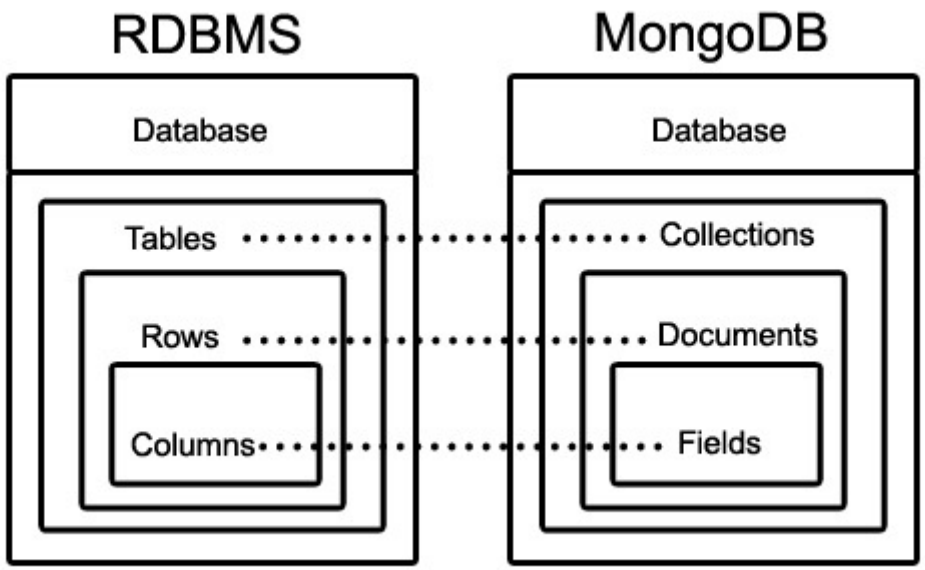
- ❖ DeviceId에 따라 다른 Collection에 저장

❖ **로그 전송 요청 시 ResponseData – Timestamp**

MongoDB

❖ MongoDB는 JSON 기반 문서(Document) 지향 데이터 모델 DB

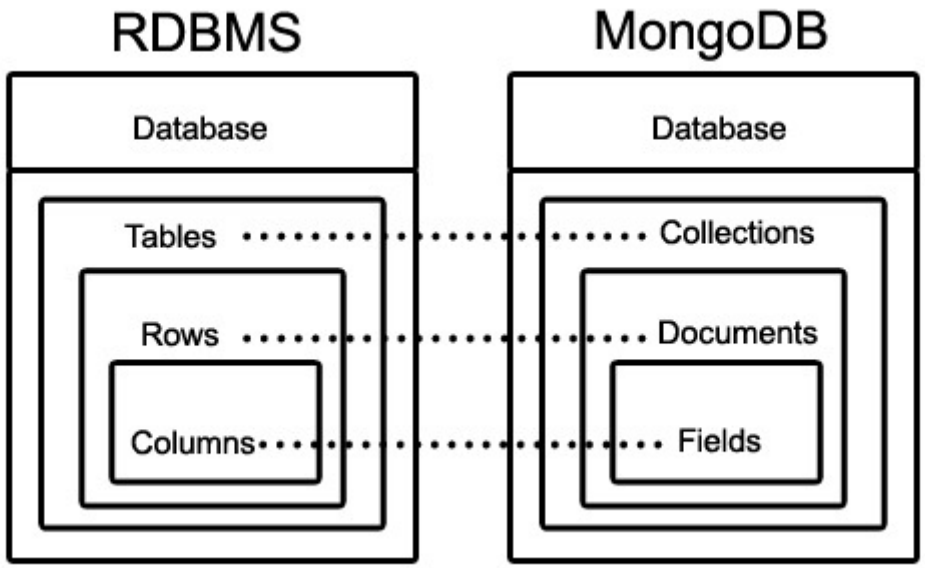
- 반정규화 기반의 동적 스키마→ Document : 한 쌍 이상의 Key:Value로 이루어지고, Document들이 모여 Collection을 이룬다



MongoDB 선택 이유

❖ MongoDB는 JSON 기반 문서 지향 데이터 모델 DB

- 적은 검색과 트랜잭션, 문자열의 같은 디바이스가 문자열 데이터를 중첩 저장하는 경우가 많음
- 디바이스마다 쌓이는 로그 데이터를 한 타입별로 묶는 것은 Document 방식의 저장이 유리
- 샤딩 기법을 이용한 확장 가능



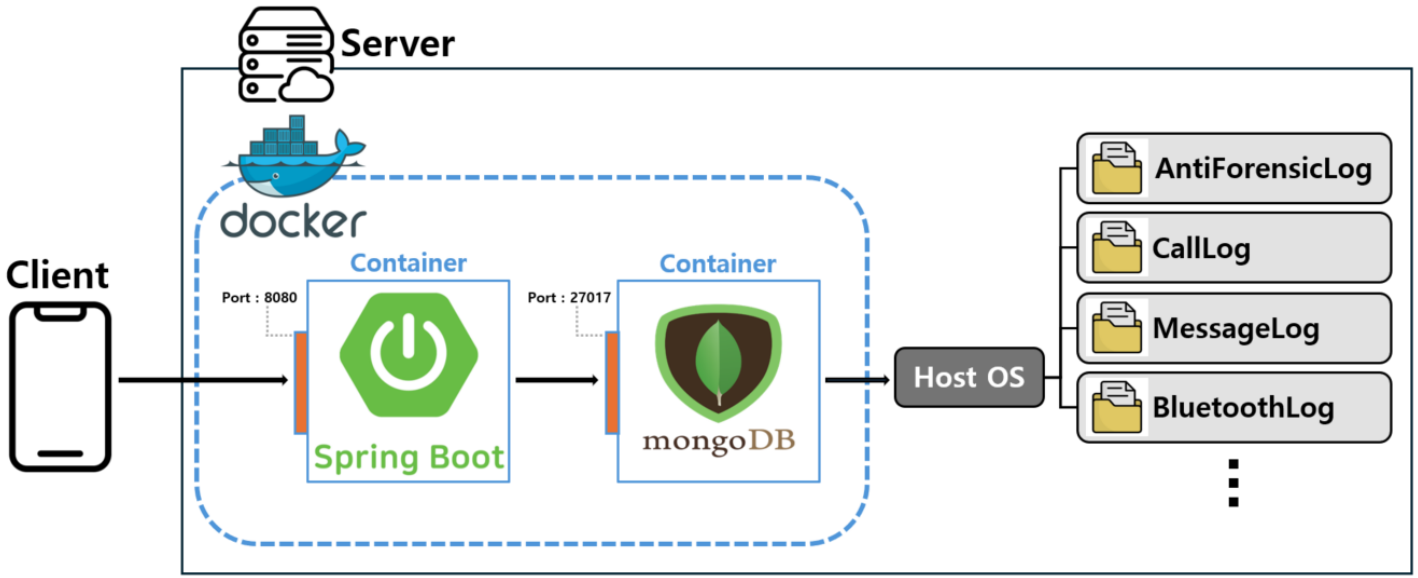
MongoDB Aggregation 기법 예시

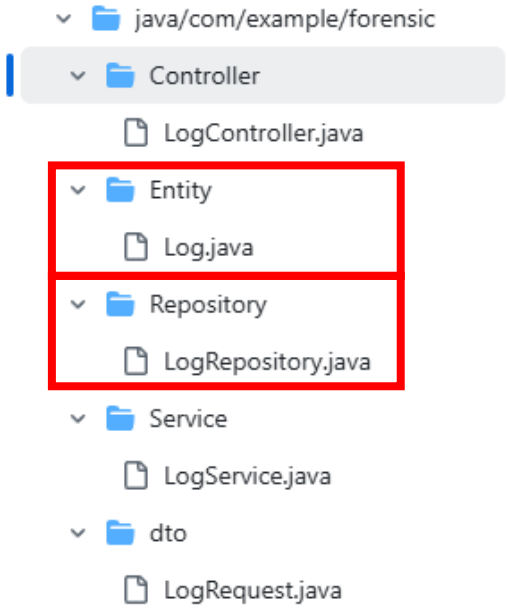
```
{ "deviceId": "12345", "timestamp": "2025-02-11T10:00:00Z", "logType": "INFO", "message": "Device started" }
{ "deviceId": "12345", "timestamp": "2025-02-11T10:05:00Z", "logType": "ERROR", "message": "Connection lost" }
```

```
db.deviceLogs.aggregate([
  { $match: { "deviceId": "12345" } },
  { $group: {
      _id: "$deviceId",
      logs: { $push: { "timestamp": "$timestamp", "logType":
"$logType", "message": "$message" } }
  }}
]);
```

```
{
  "_id": "12345",
  "logs": [
    { "timestamp": "2025-02-11T10:00:00Z", "logType": "INFO", "message": "Device started" },
    { "timestamp": "2025-02-11T10:05:00Z", "logType": "ERROR", "message": "Connection lost" }
  ]
}
```

스프링 부트를 이용한 백엔드 구성



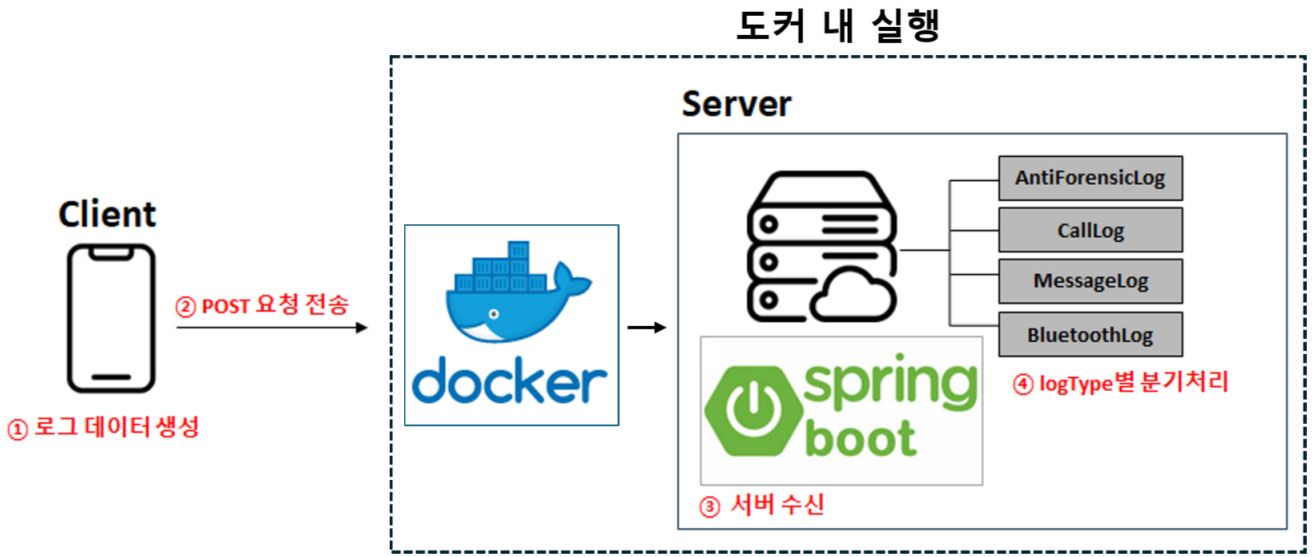


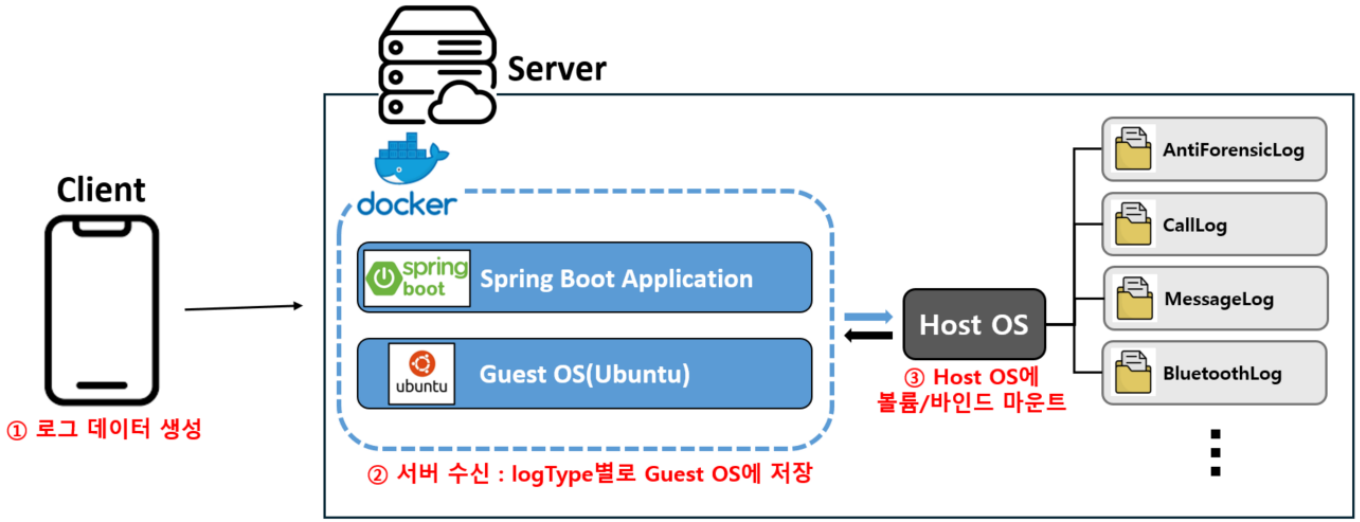
❖ HTTP/HTTPS(REST API)를 이용하여 클라이언트 디바이스에서 서버로 JSON 형식의 데이터를 POST 방식으로 전송 후 **DeviceId마다 Collection을 별도 저장**

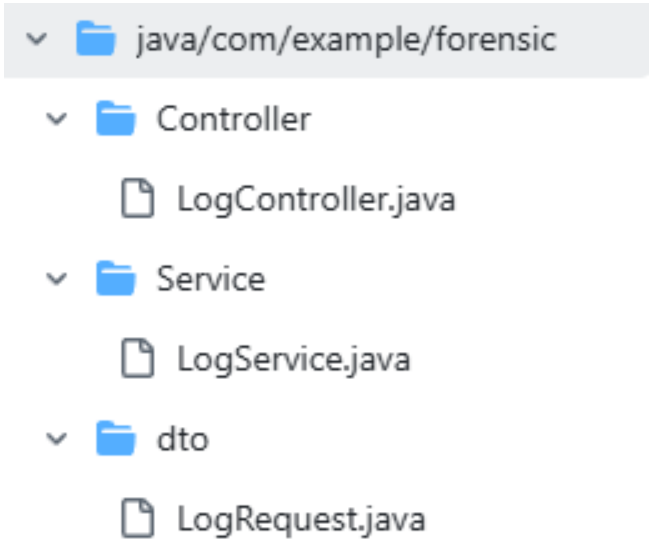
- 서버는 스프링 부트가 제공하는 REST API 엔드포인트에서 데이터를 수신

❖ 스프링 컨트롤러에서는 클라이언트가 보낸 JSON 파일에서 로그 데이터의 라벨(logType)을 읽어, **각 라벨에 해당하는 파일을 MongoDB에 저장**

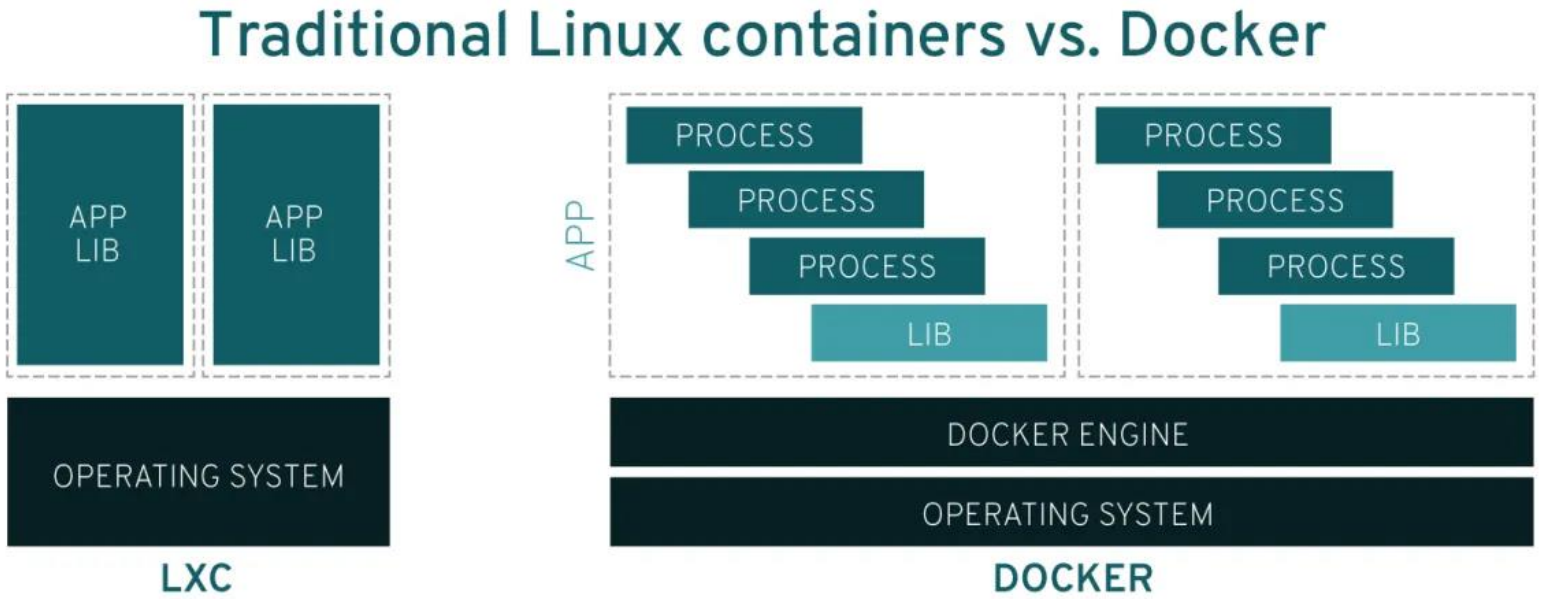
스프링 부트를 이용한 백엔드 구성



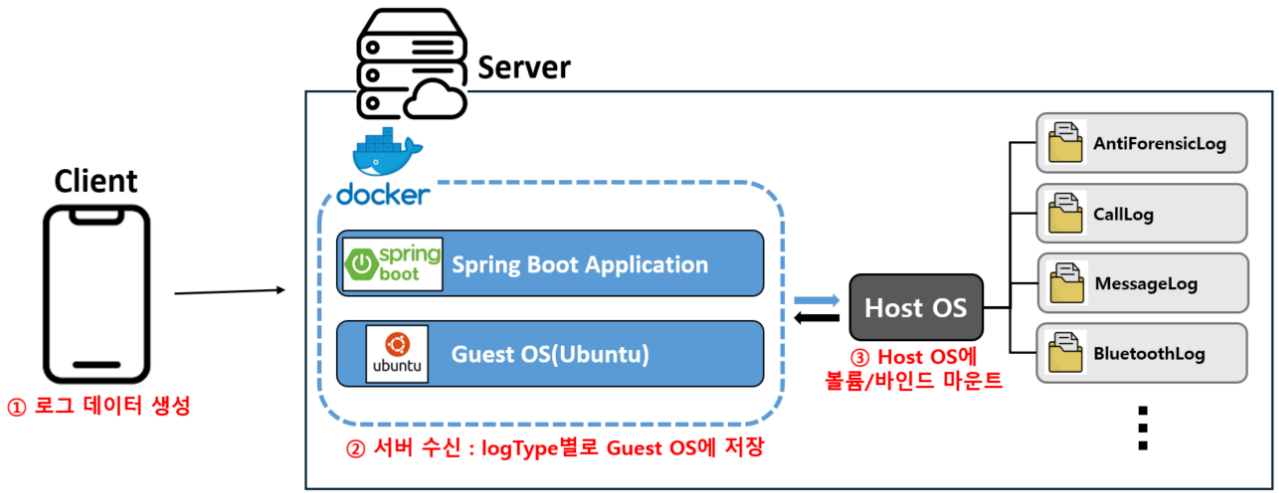




Feedback



컨테이너는 애플리케이션을 실행하는 데 필요한 최소한의 바이너리, 라이브러리를 포함 → Ubuntu



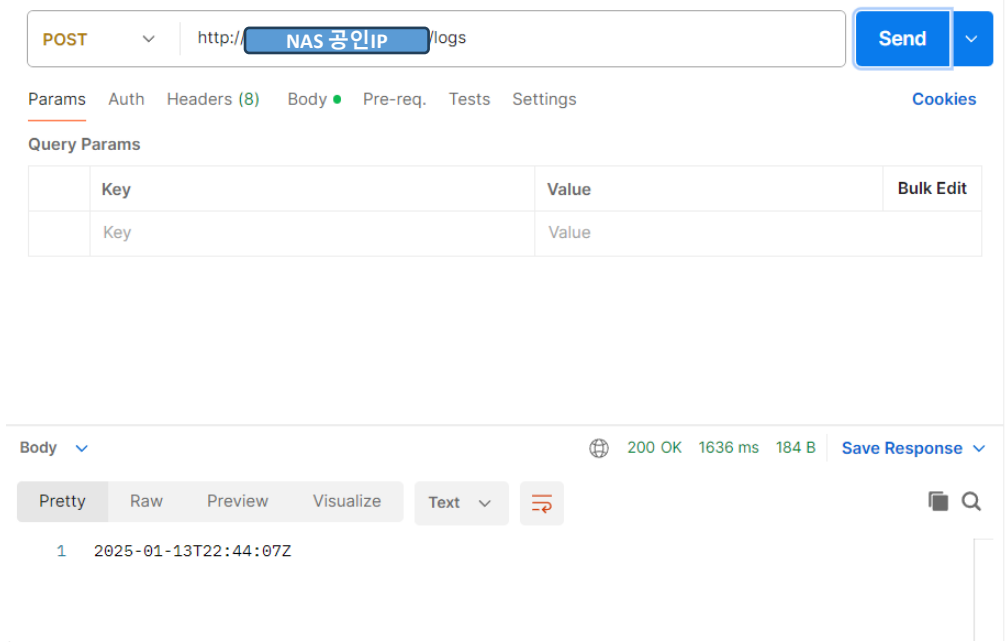
API 명세

❖ POST - http://{SeverIP}/logs

❖ 로그 추가 기능 (appendLog)

❖ Sever에 로그 데이터를 logType별로 전송

❖ ResponseData로 서버의 시간을 String으로 받아 옴

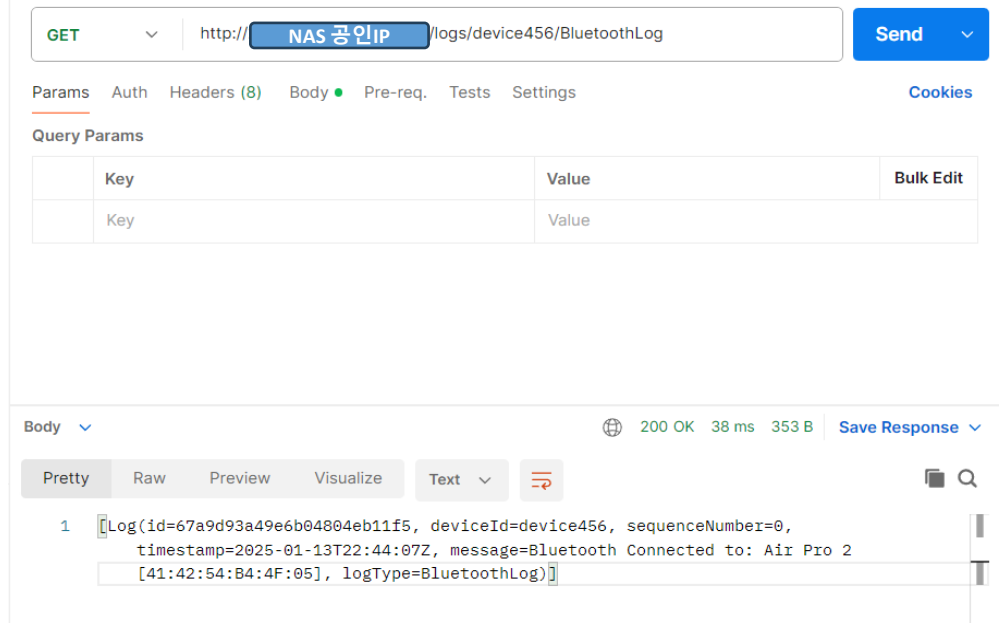


❖ GET - http://{SeverIP}/logs/{DeviceId}/{logType}

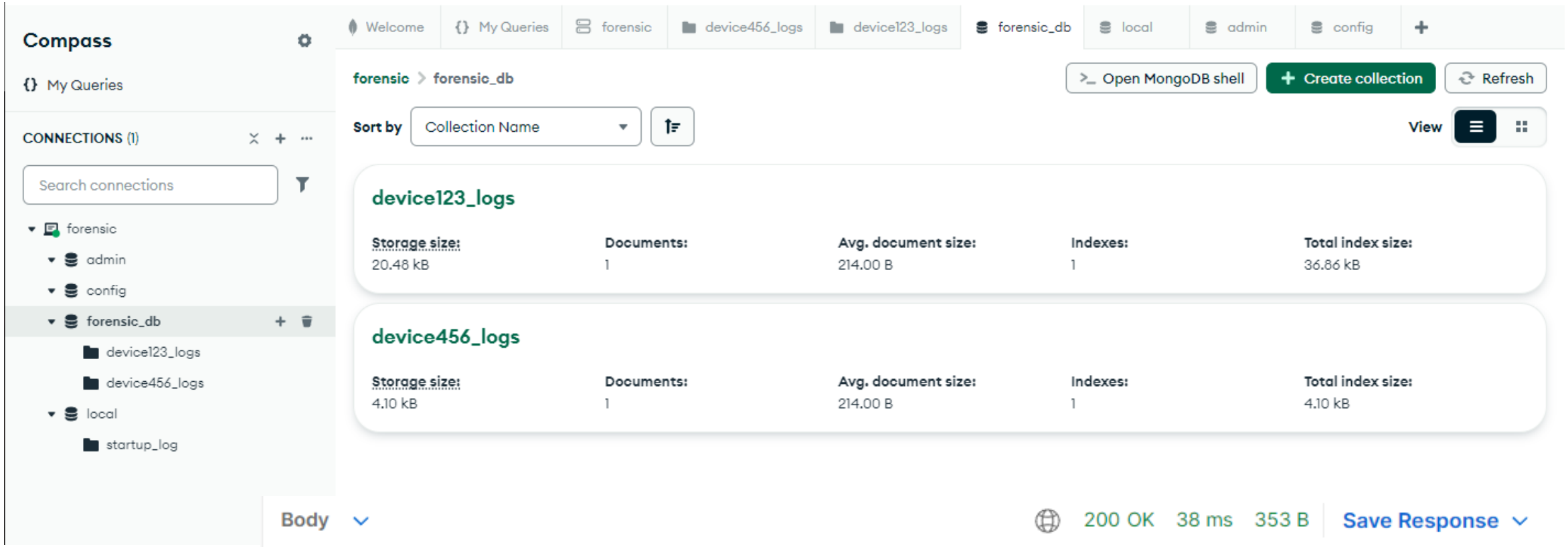
❖ 로그 조회 기능 (readLog)

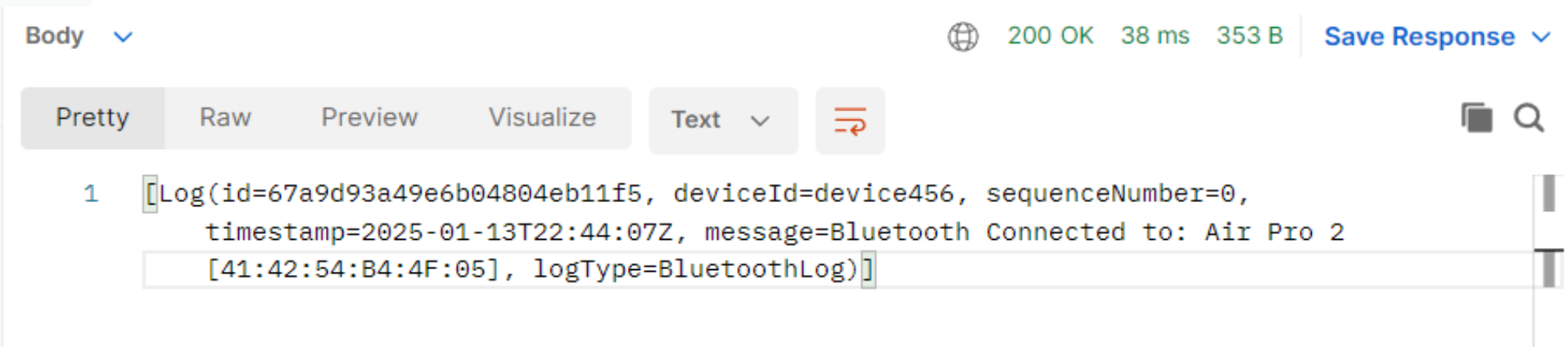
❖ Device별로 MongoDB에 저장된 로그를 Logtype별로 조회

❖ 특정 로그 타입에 해당하는 컬렉션을 읽어 반환



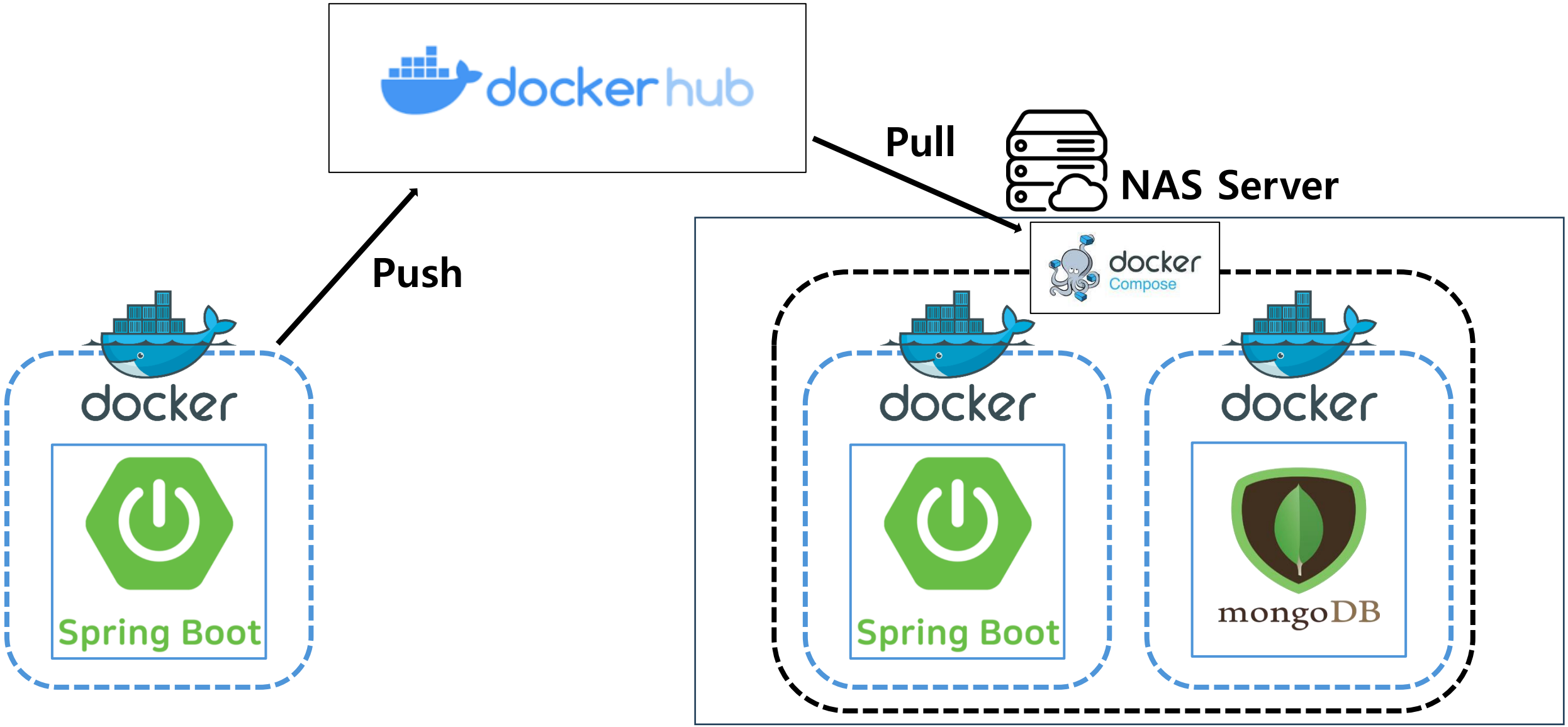
API 명세


Compass
My Queries
CONNECTIONS (1)
Search connections
forensic
admin
config
forensic_db
device123_logs
device456_logs
local
startup_log
Welcome
My Queries
forensic
device456_logs
device123_logs
forensic_db
local
admin
config
forensic > forensic_db
Open MongoDB shell
Create collection
Refresh
Sort by
Collection Name
View
device123_logs
Storage size:
20.48 kB
Documents:
1
Avg. document size:
214.00 B
Indexes:
1
Total index size:
36.86 kB
device456_logs
Storage size:
4.10 kB
Documents:
1
Avg. document size:
214.00 B
Indexes:
1
Total index size:
4.10 kB

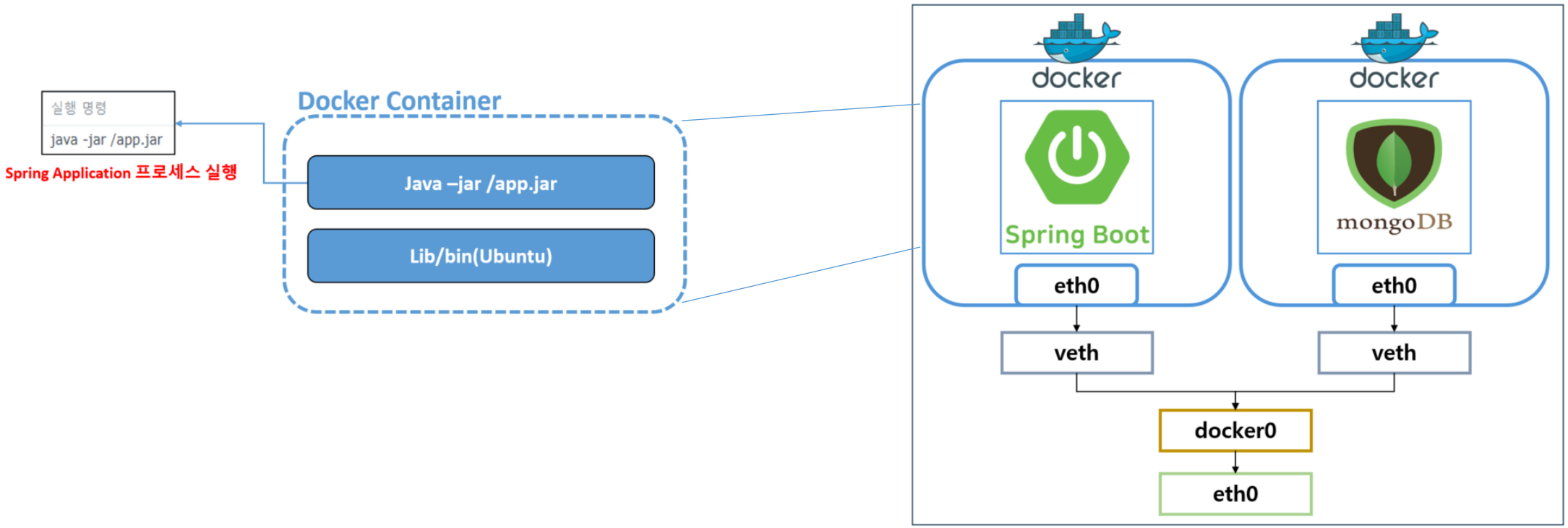

Body
200 OK
38 ms
353 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1
[Log(id=67a9d93a49e6b04804eb11f5, deviceId=device456, sequenceNumber=0,
timestamp=2025-01-13T22:44:07Z, message=Bluetooth Connected to: Air Pro 2
[41:42:54:B4:4F:05], logType=BluetoothLog)]

04 NAS 서버 배포

Docker-compose.yml 파일 작성



Container & Bridge Network



❖ Spring Boot 컨테이너에서 MongoDB와 통신하려면 네트워크를 통해 접근

- Docker 네트워크(docker0)를 통해 Spring Boot 컨테이너와 MongoDB 컨테이너가 연결
  - Spring Boot 애플리케이션은 MongoDB 컨테이너의 eth0 IP 주소로 요청을 보냄
  - MongoDB 컨테이너는 요청을 받아 데이터베이스 작업을 수행하고 응답을 반환

Docker-compose.yml 파일 작성

❖ NAS 서버의 volume1/docker에서 forensic 디렉토리 생성

```
csos_admin@TMAX:/volume1/docker$ cd forensic
csos_admin@TMAX:/volume1/docker/forensic$ ll
total 4
drwxrwxrwx+ 1 csos_admin users  36 Feb 10 21:23 .
drwxrwxrwx+ 1 root       root   28 Feb 10 18:28 ..
-rwxrwxrwx  1 csos_admin users 744 Feb 10 21:23 docker-compose.yml
```

❖ docker-compose.yml 파일 실행

- Docker compose는 여러 개의 컨테이너(Spring, MongoDB)를 하나의 서비스로 정의하여 관리
  - docker-compose up 명령으로 실행

```
version: '3.8'

services:
  # Spring Boot 애플리케이션
  app:
    image: woniwory/forensic_spring
    container_name: forensic-app
    ports:
      - "52346:8080"
    environment:
      - SPRING_DATA_MONGODB_URI=mongodb://root:example@mongodb:27017/forensic_d
    depends_on:
      - mongodb
    networks:
      - forensic_net

  # MongoDB 컨테이너
```

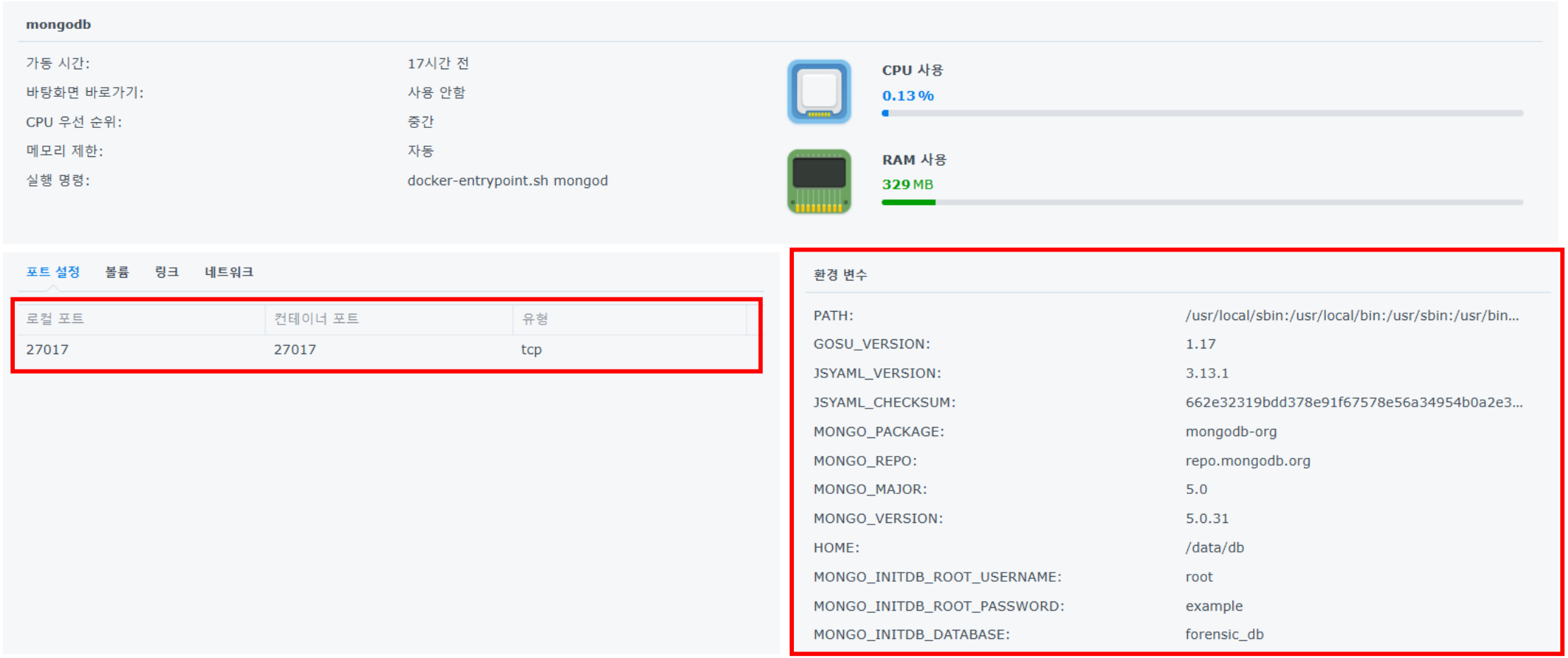
```
  # MongoDB 컨테이너
  mongodb:
    image: mongo:5.0
    container_name: mongodb
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: root
      MONGO_INITDB_ROOT_PASSWORD: example
      MONGO_INITDB_DATABASE: forensic_db
    networks:
      - forensic_net

networks:
  forensic_net:
    driver: bridge
```

NAS 서버에서의 컨테이너 실행 (1)

❖Docker hub에 Push된 Spring Application 이미지를 NAS에서 Pull 하여 컨테이너를 실행

NAS 서버에서의 컨테이너 실행 (2)

❖Docker hub의 공식 MongoDB 이미지를 NAS에서 Pull 하여 컨테이너를 실행



감사합니다