AI SRE Auto-Remediator

Google Gemini AI 기반 실시간 시스템 장애 탐지 및 자동 복구 에이전트

Choi Seong Gi linux1547@hanmail.net

목차

개요

목적

본 문서의 목적은 AI system auto remediator 도구의 목적을 정의하고, 해당 툴의 사용방법을 정리합니다.
AI System auto remediator는 서버의 시스템 로그를 모니터링하여, 장애가 감지되면 AI가 즉각적으로 분석하고 해결 명령어를 제안합니다.
관리자는 슬랙(Slack)을 통해 보고를 받고 승인을 통해 서버를 복구할 수 있습니다.

주요기능

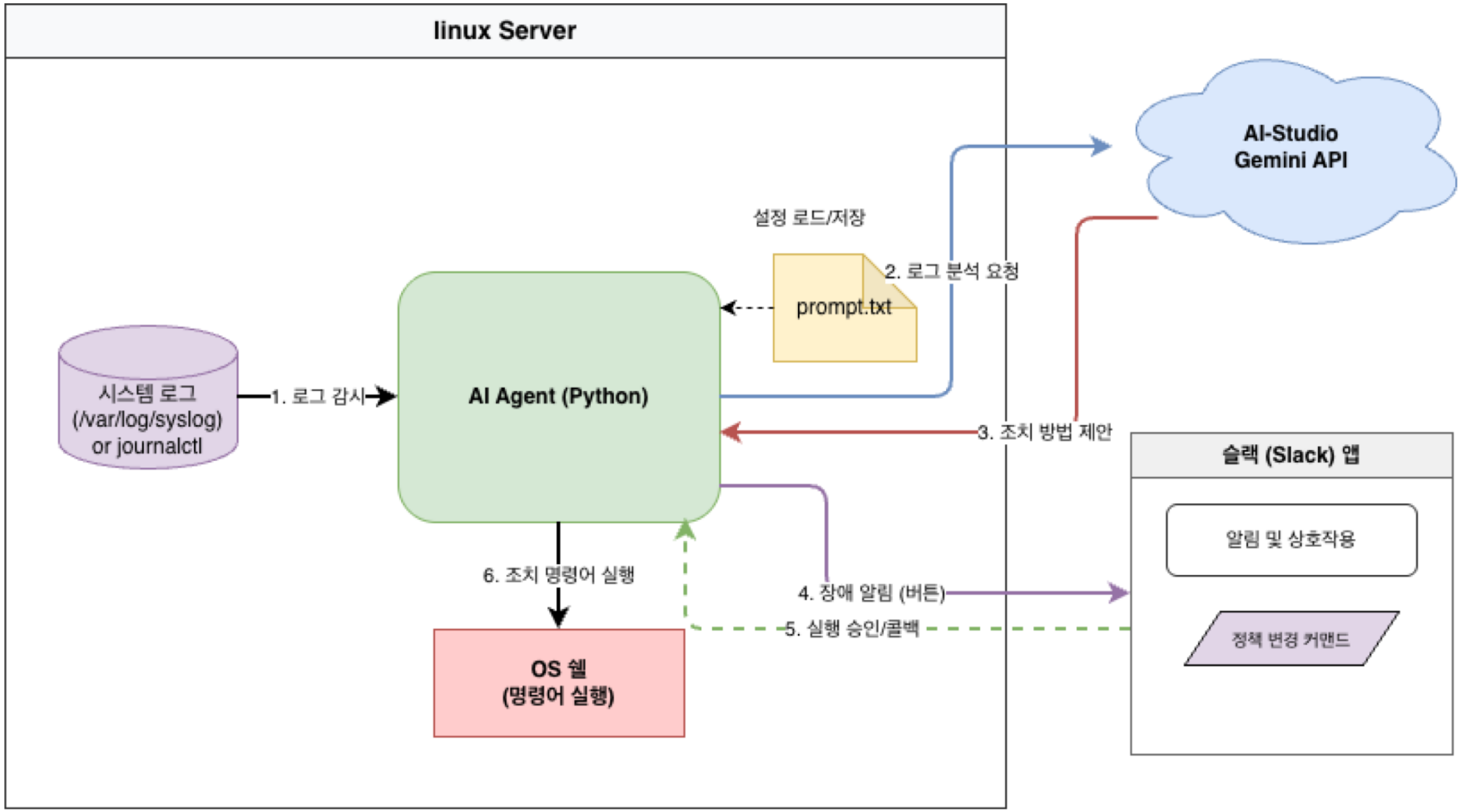
장애 감지: Journald 또는 특정 로그 파일을 감시하여 시스템 에러를 즉시 포착합니다.
AI 상황 분석: Google Gemini 모델이 에러 로그의 문맥을 분석하여 가장 적절한 시스템 명령어를 생성합니다.
보안 필터링 (Blacklist): rm, mkfs 등 시스템에 치명적인 명령어는 실행 전 자동으로 차단합니다.
자동 조치 (Auto-Execute): DISK FULL과 같이 사전에 정의된 중요 키워드는 관리자 승인 없이 AI가 즉시 복구합니다.
인터랙티브 슬랙 연동: 슬랙 버튼을 통해 조치 명령어를 실행/거절하고, 결과까지 슬랙에서 바로 확인합니다.

고려사항

- 시스템별 제공하는 서비스와 환경에 맞추어서 프롬프트 설정이 필요합니다.
- rm, dd등 시스템에 즉각적으로 영향을 주는 명령어는 사전에 정의하여 차단해야 합니다.

구성도

Google Gemini AI 기반 실시간 시스템 장애 탐지 및 자동 복구 에이전트



시스템 아키텍처

시스템 동작 흐름

시스템은 다음과 같은 흐름으로 동작합니다:

1. 모니터링 (Detection Phase) Journald 또는 지정된 로그 파일을 실시간으로 감시(tail -F 방식)합니다. 시스템 에러 레벨(err, crit, alert, emerg) 또는 사용자가 지정한 에러 키워드가 포함된 로그만 추출합니다.

2. AI 상황 분석 (Analysis Phase) 탐지된 로그와 prompt.txt에 정의된 SRE 전문가 지침을 함께 Gemini AI에게 전달합니다. AI는 발생한 장애를 해결하기 위한 최적의 단일 Bash 명령어를 생성합니다.

3. 보안 검증 (Security Filter) AI가 제안한 명령어가 실행되기 전, blacklist.txt를 실시간으로 읽어 대조합니다. rm, mkfs 등 시스템에 위험한 단어가 포함되어 있다면 즉시 실행을 차단하고 슬랙으로 보안 경고를 전송합니다.

4. 조치 및 보고 (Action & Notification)
자동 조치: 로그 내용이 auto_keywords.txt에 등록된 키워드(예: DISK FULL)와 일치하면 승인 없이 즉시 명령어를 실행합니다. 수동 조치: 그 외의 일반 에러는 슬랙 버튼을 통해 관리자에게 '실행' 또는 '거절' 여부를 묻습니다. 결과 보고: 명령어 실행 결과(Standard Output/Error)를 다시 슬랙으로 전송하여 최종 조치 여부를 확인합니다.

Agent 설치 및 동작 프로세스

Agent 설치

- 설치 주소 : https://github.com/SeongGi/AI-SRE-System

쉘 스크립트를 실행하면 아래의 6단계 과정을 통해 AI SRE 환경이 구축됩니다.

- 환경변수 입력

Gemini API Key
Slack Webhook URL
Gemini 모델버전
시스템 로그 파일 경로 (Journald, Logfile)

1. 환경 초기화 및 클린업 (Cleanup Phase) 새로운 설치를 위해 기존에 구동 중인 서비스와 데이터를 정리합니다. 기존 ai-remediator 서비스 중지 및 삭제 에이전트 사용 포트(예: 7788) 점유 프로세스 종료 기존 설치 디렉토리(/opt/ai-agent) 및 전용 유저 삭제

2. 전용 유저 및 권한 설정 (Security Setup) 보안을 위해 최소 권한을 가진 전용 시스템 유저를 생성합니다. ai-agent 시스템 유저 생성 시스템 로그를 읽을 수 있도록 adm 및 systemd-journal 그룹에 유저 추가 작업 디렉토리 생성 및 소유권 부여

```
seonggi1547@tesalamate-seonggi:~$ bash ai-auto.sh
=== AI SRE 에이전트 설정 시작 ===
✔ Gemini API Key:
✔ Slack Webhook URL:
✔ Gemini 모델 버전 (예: gemini-1.5-flash): gemini-3-flash-preview
✔ 서비스 포트 번호 (기본: 5000):
------------------------------------------------
모니터링 방식을 선택하세요:
1) Journald (시스템 전체 에러 감시 - 권장)
2) Log File (특정 파일 경로 지정 감시)
선택 (1 또는 2): 2
✔ 감시할 로그 파일 경로 (예: /var/log/syslog):
```

```
--- [1/5] 환경 초기화 (Clean Up) ---
+ sudo systemctl stop ai-remediator.service
+ sudo systemctl disable ai-remediator.service
+ sudo rm -f /etc/systemd/system/ai-remediator.service
+ sudo fuser -k 5000/tcp
+ sudo rm -rf /opt/ai-agent
+ sudo userdel -r ai-agent
+ echo --- [2/5] 유저 및 권한 설정 ---
--- [2/5] 유저 및 권한 설정 ---
+ sudo useradd -m -s /bin/bash ai-agent
+ sudo usermod -aG adm,systemd-journal ai-agent
+ sudo mkdir -p /opt/ai-agent
+ sudo chown -R ai-agent:ai-agent /opt/ai-agent
```

Agent 설치 및 동작 프로세스

3. 관리용 설정 파일 생성 (Configuration Setup)
AI의 행동을 제어하는 3대 핵심 텍스트 파일을 기본값으로 생성합니다.
prompt.txt: AI에게 부여하는 SRE 전문가 지침서
blacklist.txt: 실행을 금지할 위험 명령어 목록 (rm, mkfs 등)
auto_keywords.txt: 승인 없이 즉시 조치할 장애 키워드 (DISK FULL 등)

```
--- [3/5] 필수 파일 및 가상환경 생성 ---
+ sudo -u ai-agent tee /opt/ai-agent/prompt.txt
+ sudo -u ai-agent tee /opt/ai-agent/auto_keywords.txt
+ sudo -u ai-agent tee /opt/ai-agent/blacklist.txt
+ sudo apt update
Hit:1 http://kr.archive.ubuntu.com/ubuntu noble InRelease
```

4. Python 가상환경 구축 및 패키지 설치 (Python Environment) 시스템 라이브러리와 분리된 독립적인 실행 환경을 구성합니다. python3 -m venv venv: 독립 가상환경 생성 pip install: 필수 라이브러리(flask, requests, google-genai) 설치

```
python3-venv is already the newest version (3.12.3-0ubuntu2.1).
coreutils is already the newest version (9.4-3ubuntu6.1).
psmisc is already the newest version (23.7-1build1).
0 upgraded, 0 newly installed, 0 to remove and 65 not upgraded.
+ sudo -u ai-agent python3 -m venv /opt/ai-agent/venv
+ sudo -u ai-agent /opt/ai-agent/venv/bin/pip install flask requests google-genai
Collecting flask
  Downloading flask-3.1.2-py3-none-any.whl.metadata (3.2 kB)
Collecting requests
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting google-genai
  Downloading google_genai-1.57.0-py3-none-any.whl.metadata (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.3/53.3 kB 2.4 MB/s eta 0:00:00
```

5. 메인 로직(main.py) 및 서비스 등록 (Deployment) 에이전트의 핵심 엔진을 배치하고 백그라운드 서비스로 등록합니다. main.py 파일 생성: 로그 감시, AI 분석, 슬랙 통신 로직 포함 systemd 유닛 파일 생성: 서버 부팅 시 자동 실행 및 프로세스 다운 시 자동 재시작 설정

6. 서비스 가동 및 상태 확인 (Activation) 모든 설정을 마치고 모니터링을 시작합니다. systemctl start: 에이전트 서비스 즉시 가동 공백이나 특수문자가 제거된 안전한 환경 변수 주입 확인 최종 접속 IP 및 포트 정보 출력

Agent 설치 및 동작 프로세스

5. 메인 로직(main.py) 및 서비스 등록 (Deployment) 에이전트의 핵심 엔진을 배치하고 백그라운드 서비스로 등록합니다. main.py 파일 생성: 로그 감시, AI 분석, 슬랙 통신 로직 포함 systemd 유닛 파일 생성: 서버 부팅 시 자동 실행 및 프로세스 다운 시 자동 재시작 설정

```
+ echo --- [4/5] 메인 코드(main.py) 생성 ---
--- [4/5] 메인 코드(main.py) 생성 ---
+ sudo -u ai-agent tee /opt/ai-agent/main.py
```

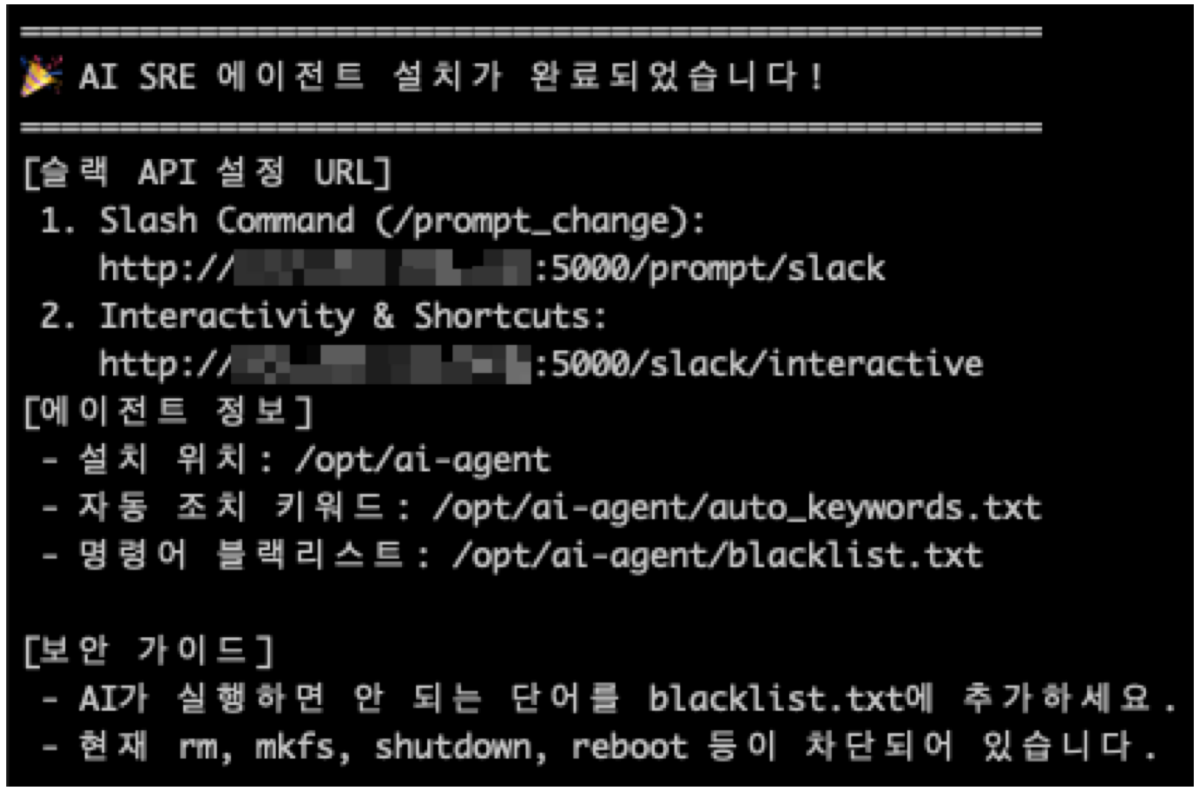
6. 서비스 가동 및 상태 확인 (Activation) 모든 설정을 마치고 모니터링을 시작합니다.

```
--- [5/5] 서비스 등록 및 시작 ---
+ sudo tee /etc/systemd/system/ai-remediator.service
+ sudo systemctl daemon-reload
+ sudo systemctl enable ai-remediator.service
Created symlink /etc/systemd/system/multi-user.target.wants/ai-remediator.service → /etc/systemd/system/ai-remediator.service.
+ sudo systemctl restart ai-remediator.service
+ curl -s ifconfig.me
+ PUBLIC_IP=211.[redacted]
+ echo
```

Agent 설치 및 동작 프로세스

에이전트 서비스 실행

공백이나 특수문자가 제거된 안전한 환경 변수 주입 확인
최종 접속 IP 및 포트 정보 출력



Agent 관리 및 설정 방법

Agent 관리

Agent 설치 후, 모든 설정은 /opt/ai-agent/ 디렉토리 내의 텍스트 파일을 수정하여 서비스 재시작 없이 즉시 반영할 수 있습니다.

관리 대상	파일 경로	설명
보안 필터링	blacklist.txt	실행을 차단할 위험 단어 목록 (예: rm, mkfs)
자동 조치	auto_keywords.txt	AI가 승인 없이 즉시 조치할 로그 키워드 (예: DISK FULL)
AI 페르소나	prompt.txt	AI에게 부여할 지침 (슬랙 /prompt_change로도 수정 가능)

Agent 관리 및 설정 방법

관리명령어
모니터링 : 서버 터미널에서 에이전트의 상태를 확인하고 관리할 때 사용합니다.

“sudo journalctl –u ai-remediator.service –f

```
Jan 05 18:58:53 seonggi-ubuntu-899541 systemd[1]: Started AI SRE Agent Final.
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]: [*] 모니터링 시작 (JOURNAL)
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]:  * Serving Flask app 'main'
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]:  * Debug mode: off
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]: WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]:  * Running on all addresses (0.0.0.0)
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]:  * Running on http://127.0.0.1:7788
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]:  * Running on http://10.0.0.5:7788
Jan 05 18:58:58 seonggi-ubuntu-899541 python3[6022]: Press CTRL+C to quit
Jan 05 19:03:55 seonggi-ubuntu-899541 python3[6022]: 100.24.209.157 - - [05/Jan/2026 19:03:55] "POST /prompt/slack HTTP/1.1" 200 -
```

현재 시스템 데몬의 상황과 error 발생시 slack 메시지 전송여부 (http 200 메시지)를 확인할 수 있습니다.

서비스 상태 확인 및 재시작

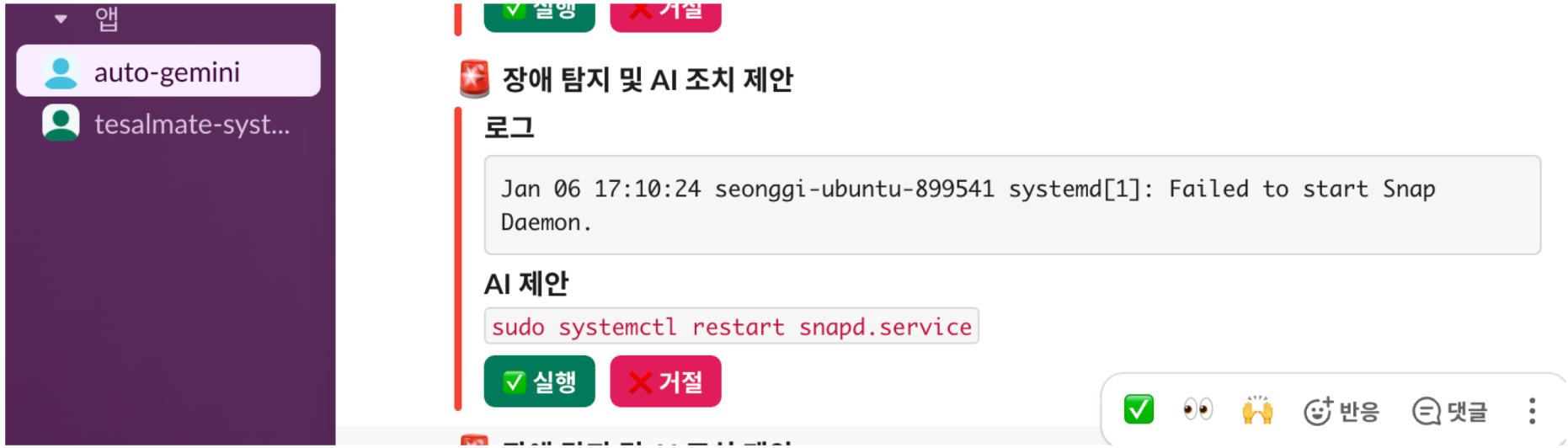
“sudo systemctl status ai-remediator.service”
“sudo systemctl restart ai-remediator.service”

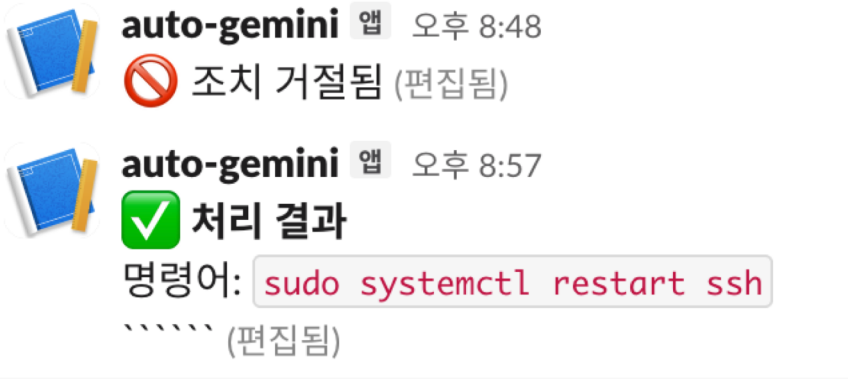
주의사항

API 키 보안: main.py와 서비스 설정에 포함된 Gemini API 키가 외부로 유출되지 않도록 주의하십시오.

권한 관리: 에이전트는 ai-agent 유저 권한으로 실행되나, AI가 제안하는 명령어에 따라 sudo 권한이 필요할 수 있습니다.

최종결과


앱
auto-gemini
tesalmate-syst...
장애 탐지 및 AI 조치 제안
로그
Jan 06 17:10:24 seonggi-ubuntu-899541 systemd[1]: Failed to start Snap Daemon.
AI 제안
sudo systemctl restart snapd.service
✅ 실행
❌ 거절
✅ 👀 🙌 반응 댓글


auto-gemini 앱 오후 8:48
🚫 조치 거절됨 (편집됨)
auto-gemini 앱 오후 8:57
✅ 처리 결과
명령어: sudo systemctl restart ssh
``````
(편집됨)
``````

결론 및 향후 방향

본 프로젝트는 장애 알림 단계를 넘어, 실무 복구 조치(Action)를 수행하는 AI SRE 에이전트를 구축하였습니다.

운영 효율성 혁신: Journald 로그를 실시간으로 분석하고 조치 명령어를 자동 생성함으로써, 관리자가 원인을 파악하고 명령어를 검색하는 시간을 획기적으로 단축했습니다.
안전한 자동화 구현: AI의 오판을 방지하기 위해 금지어 목록(Blacklist) 기반의 보안 필터링과 관리자 승인 절차를 적용하여 운영의 안정성을 확보했습니다.
유연한 하이브리드 대응: 사전에 정의된 치명적 장애는 즉시 자동 복구하고, 미지의 에러는 관리자의 판단을 거치도록 설계하여 속도와 안정성을 가졌습니다.

향후 발전 방향
현재의 단일 서버 에이전트 모델을 기반으로, 대규모 운영 환경에 적용 하는 발전성을 검토하고 있습니다.

중앙 집중형 관제 시스템(Fleet Management) 구축
현재는 서버마다 개별적으로 에이전트가 동작하지만, 향후에는 수십 대의 서버 로그를 중앙에서 수집하고 AI가 통합 분석하는 구조로 확장합니다.
이를 통해 특정 서버군 전체에 동일한 패치를 적용하거나, 다중 서버 동시 장애에 대한 일괄 조치가 가능해집니다.

RAG(검색 증강 생성) 기술을 활용한 사내 지식 연동
현재 Gemini가 가진 범용 리눅스 지식에 더해, 회사의 과거 장애 처리 보고서나 내부 위키(Wiki), 개발 문서를 AI에게 학습시킵니다.
이를 통해 "일반적인 해결책"이 아닌 " 시스템 환경에 맞는 맞춤형 복구 가이드"를 구현합니다.

결론 및 향후 방향

단일 명령어가 아닌, 워크플로우(Workflow) 자동화
현재의 단일 Bash 명령어 실행 방식을 고도화하여, 장애 발생 시 Ansible Playbook을 실행하거나 Terraform 스크립트를 구동하도록 기능을 확장합니다.
예를 들어, 서비스 재시작만으로 해결되지 않는 경우 자동으로 스냅샷을 생성하고 인스턴스를 교체하는 등 복잡한 순차 작업을 자동화해볼 수 있습니다.

장애 예측 및 선제적 대응
단순히 에러 로그가 발생한 직후에 대응하는 것을 넘어, CPU/메모리 사용량의 이상 추이나 로그 패턴의 변화를 감지하여 장애 발생 전에 미리 경고하는 시스템으로 발전시킵니다.
"디스크가 가득 찼습니다"라는 알림 대신, "3시간 뒤 디스크가 가득 찰 것으로 예측됩니다"라는 알림을 통해 장애를 원천 차단

Thank you