

OBSERVABILITY AS CODE

우리FISA 3기 클라우드 엔지니어링 기술세미나 - 이아영

목차

1. 주제 선정 배경 및 개요
2. **Observability as Code의 개념**
3. **Observability as Code를 위한 도구 및 기술 스택**
4. **Observability as Code의 구현**
 - * **Dashboard as Code**
 - * **Alert as Code**
5. 결론

1. 주제 선정 배경 및 개요

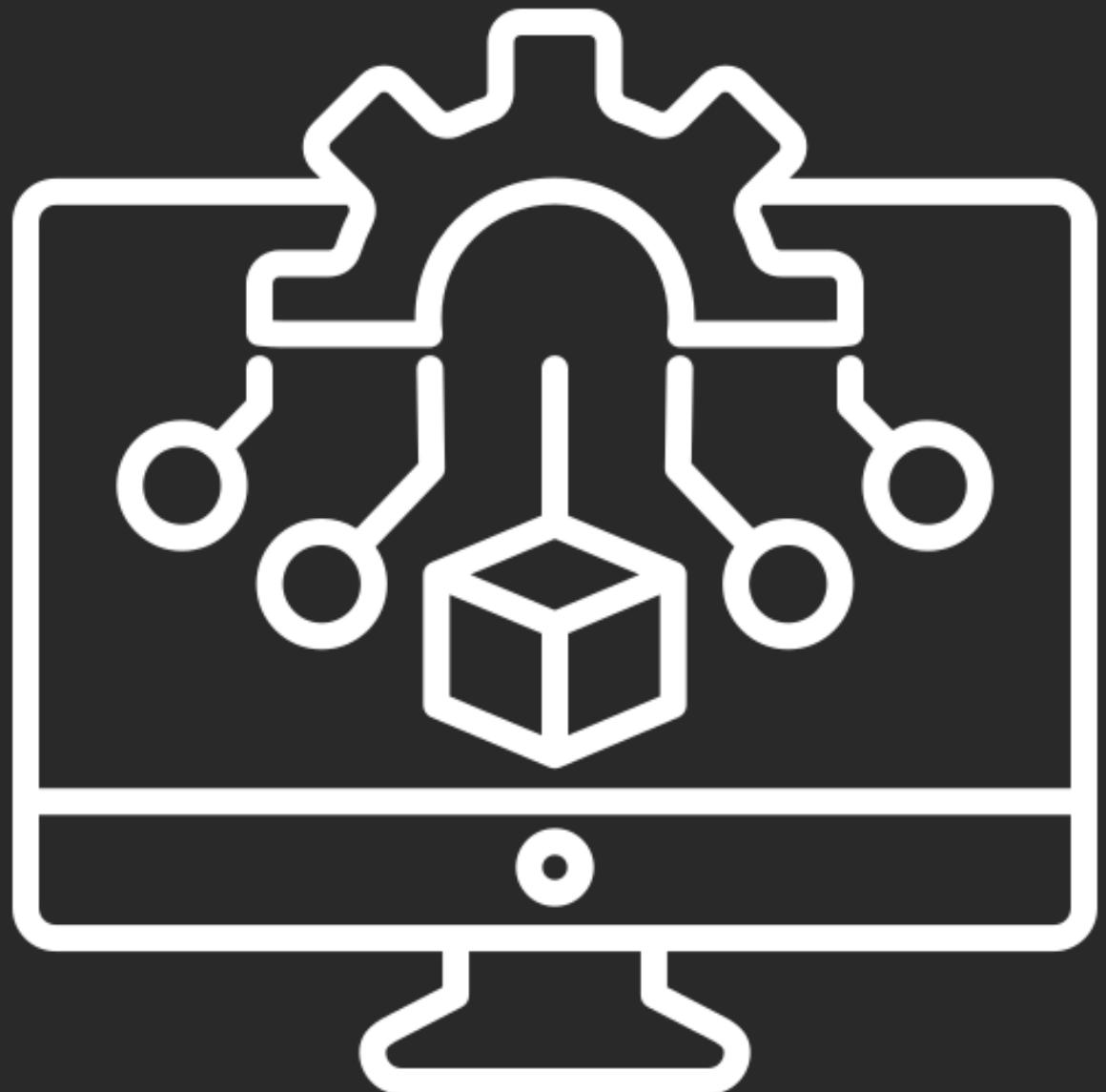
| 1. 주제 선정 배경 및 개요

주제 선정 배경

- **Observability** (Metric/Logs/Tracing)에 대한 기본 지식과 경험이 있는 분
- 서비스 모니터링(APM), 로그, 알람, 추적 도구(Prometheus, Grafana, Loki 등) 툴에 대한 사용 경험이 있으신 분
- 오픈소스 소프트웨어 및 도구를 적극적으로 활용하시는 분
- AWS Public Cloud 환경 서비스 운영 경험이 있으신 분
- 원활한 커뮤니케이션 능력이 있으신 분

- 개발 경력 3년 이상, 전산 관련 학과 학사이상 또는 동일한 자격을 보유하신 분을 찾습니다.
- Java / Kotlin, Spring Framework 관련 개발 경험과 문제 해결능력을 보유하신 분을 찾습니다.
- **Observability** (메트릭/로그/트레이싱)에 대한 기본 지식과 경험이 있는 분을 찾습니다.
- 서비스 모니터링, 로그, 얼럿, Tracing 도구(Prometheus, Grafana, Loki 등)에 대한 사용 경험이 있으신 분을 찾습니다.
- SRE 문화에 대한 이해, 경험이 있으신 분을 찾습니다.

Observability의 필요성



현대 시스템의 복잡성 증가
정상적인 운영을 위한 현상태 파악의 중요성

Observability

Observability

IT 인프라에 대한 근본적인 장애 원인을 분석하기 위한 방법론
시스템에서 발생하는 데이터를 분석해 내부 상태를 파악

Observability

IT 인프라에 대한 근본적인 장애 원인을 분석하기 위한 방법론
시스템에서 발생하는 데이터를 분석해 내부 상태를 파악



METRICS

정량적 측정
시간 기반
예시 : CPU 사용률



EVENTS

비정량적
상관 관계
예시 : 배포 이벤트



LOGS

상세 정보
텍스트 기반
예시 : 서버 상태 로그



TRACES

노드 간 추적
상호 연결성
예시 : API 호출

2. Observability as Code의 개념

Observability as Code

Observability 도구의 설정을 일관되고 통제된 방식으로 자동화

Observability as Code

Observability 도구의 설정을 일관되고 통제된 방식으로 자동화

UI를 통한 관리

- 환경마다 일관성 유지하기 어려움
- 변경 사항을 추적하기가 어려움

OaC를 통한 관리

- 설정 파일을 통한 일관성 및 추적의 용이성 보장
- 자동으로 설정 배포 가능
- 반복 작업이 줄어듬
- 협업 효율 증가

3. Observability as Code를 위한 도구 및 기술 스택

| 2. Observability as Code를 위한 도구 및 기술 스택



PROMETHEUS

메트릭 수집 및 저장
실시간 성능 데이터 수집



GRAFANA

시각화 도구
다양한 데이터 소스 지원



TERRAFORM

Observability 도구와의 통합
관측 도구의 설정 코드화

| 2. Observability as Code를 위한 도구 및 기술 스택



LOKI

로그 집계 및 저장
효율적 스토리지 및 Grafana 통합



OPENTELMETRY

표준화된 데이터 수집 프레임워크
다양한 언어 및 플랫폼 지원



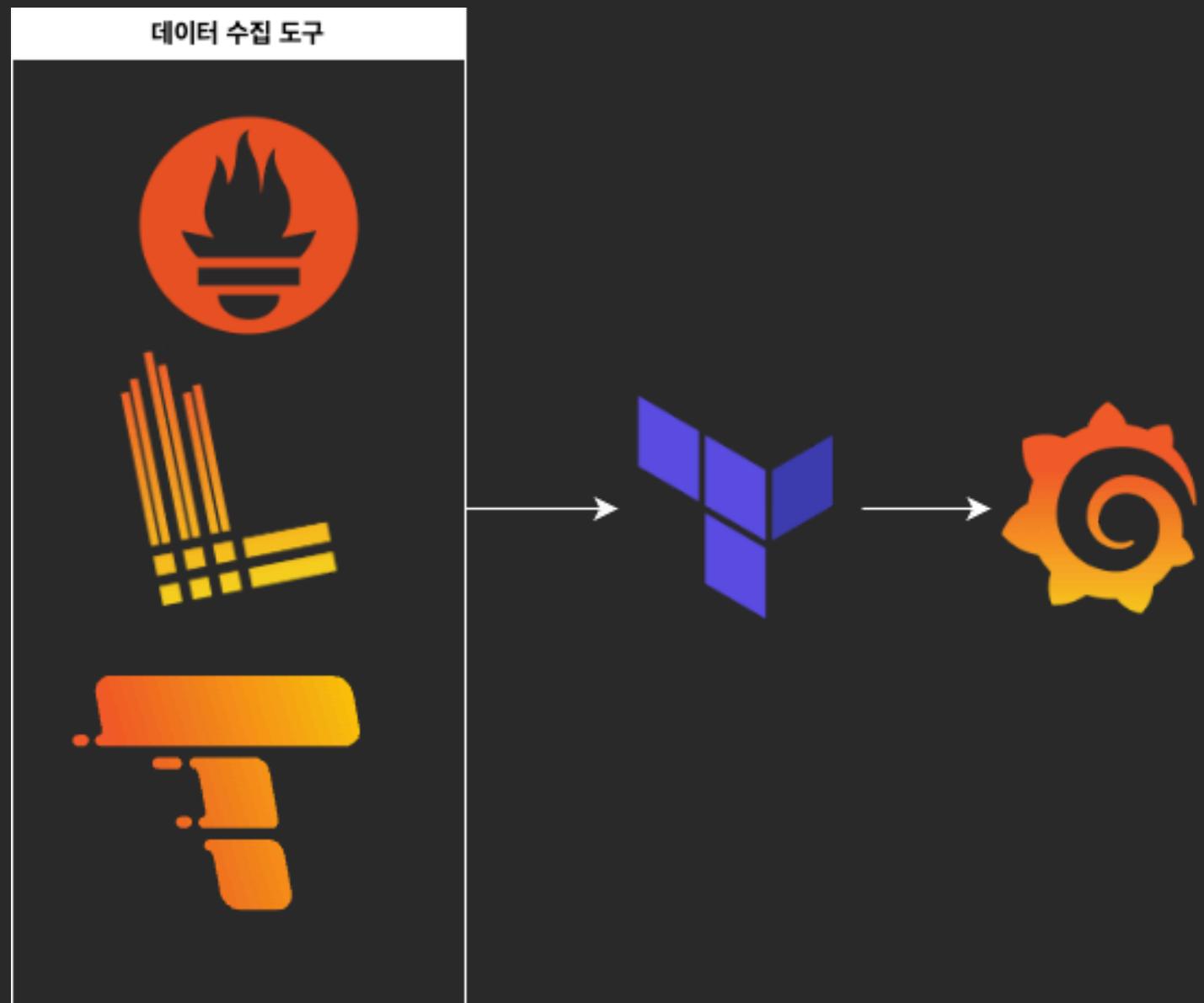
TEMPO

대규모 시스템에서의 효율적 관리
Grafana와의 시각화 및 분석 통합

4. Observability as Code의 구현

Dashboard as Code

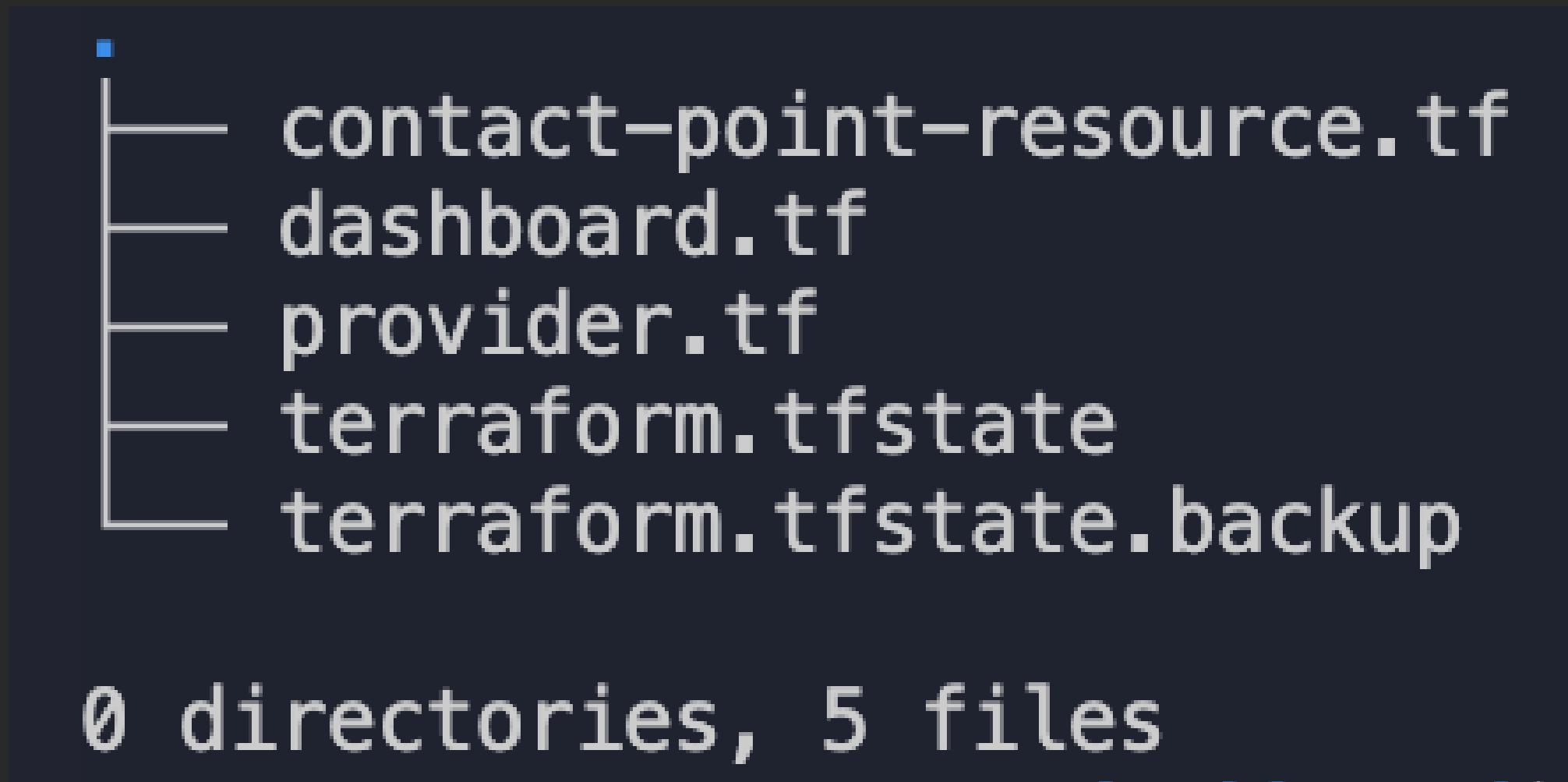
목표 : MELT를 한 번에 볼 수 있는 Dashboard 구현하기



- MELT 데이터 추출
- Terraform으로 연결
- Grafana로 시각화

Dashboard as Code

목표 : MELT를 한 번에 볼 수 있는 Dashboard 구현하기



```
└── contact-point-resource.tf
    ├── dashboard.tf
    ├── provider.tf
    └── terraform.tfstate
        └── terraform.tfstate.backup

0 directories, 5 files
```

코드 확인 : <https://github.com/ayleeee/Observability-as-Code>

Dashboard as Code

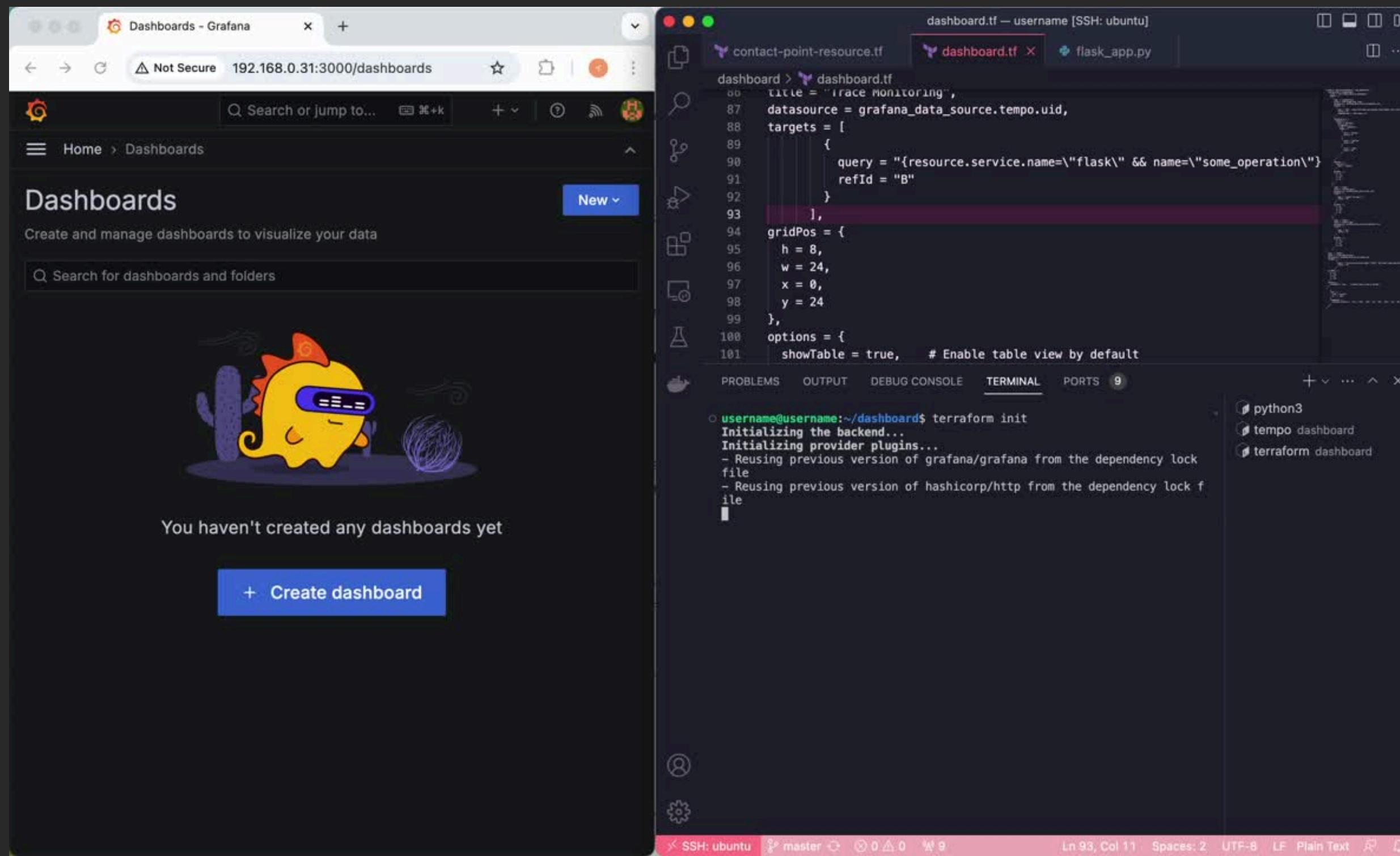
목표 : MELT를 한 번에 볼 수 있는 Dashboard 구현하기

```
dashboard > Observability-as-Code > observability > 🐾 provider.tf
1  terraform {
2    required_providers {
3      grafana = {
4        source  = "grafana/grafana"
5        version = "3.10.0"
6      }
7      http = {
8        source = "hashicorp/http"
9        version = "~>3.0"
10     }
11    }
12  }
13
14  provider "grafana" {
15    url  = ""
16    auth = "AUTH TOKEN"
17  }
18
19  provider "http" {}
```

```
dashboard > Observability-as-Code > observability > 🐾 dashboard.tf
1  resource "grafana_dashboard" "cpu_dashboard" {
2    config_json = jsonencode({
3      title = "Full Observability Dashboard",
4      panels = [
5        {
6          type  = "timeseries",
7          title = "CPU Usage Over Time",
8          datasource = grafana_data_source.prometheus.uid,
9          targets = [
10            {
11              expr = "100 - (avg(irate(node_cpu_seconds_total{mode='idle'}[5m])) * 100)",
12              refId = "A",
13              legendFormat = "CPU Usage (%)"
14            }
15          ],
16          fieldConfig = {
17            defaults = {
18              unit = "percent",
19              thresholds = {
20                mode = "absolute",
21                steps = [
22                  {
23                    color = "green",
24                    value = 0
25                  },
26                  {
27                    color = "yellow",
28                    value = 80
29                  },
29                  {
31                    color = "red",
32                    value = 90
33                  }
34                ]
35              }
36            },
37            options = {
38              tooltip = {
39                mode = "single"
40              }
41            }
42          }
43        }
44      ]
45    }
46  }
```

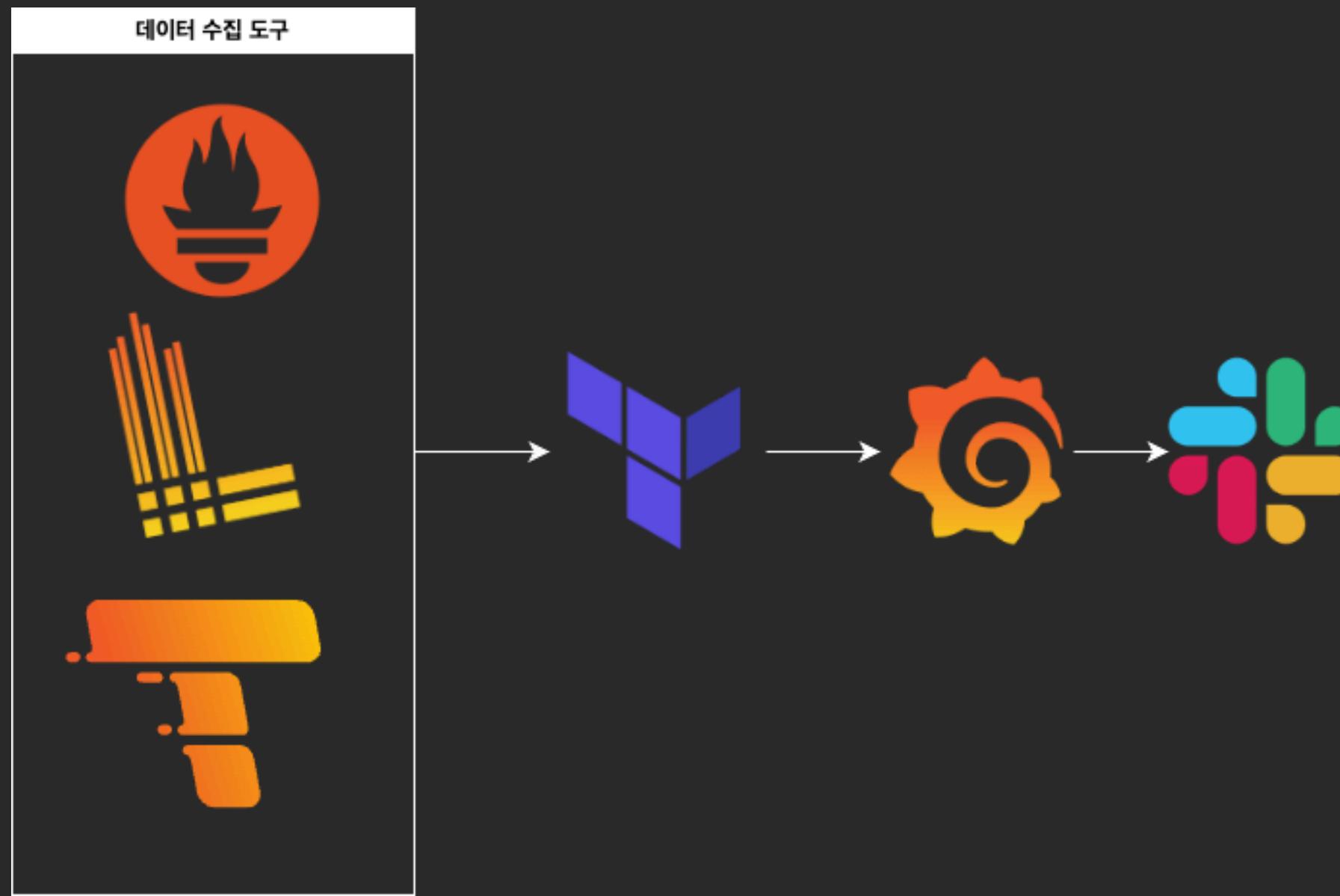
코드 확인 : <https://github.com/ayleeeee/Observability-as-Code>

Dashboard as Code



Alert as Code

목표 : CPU 사용량이 임계치를 넘어가면 슬랙으로 알림 보내기

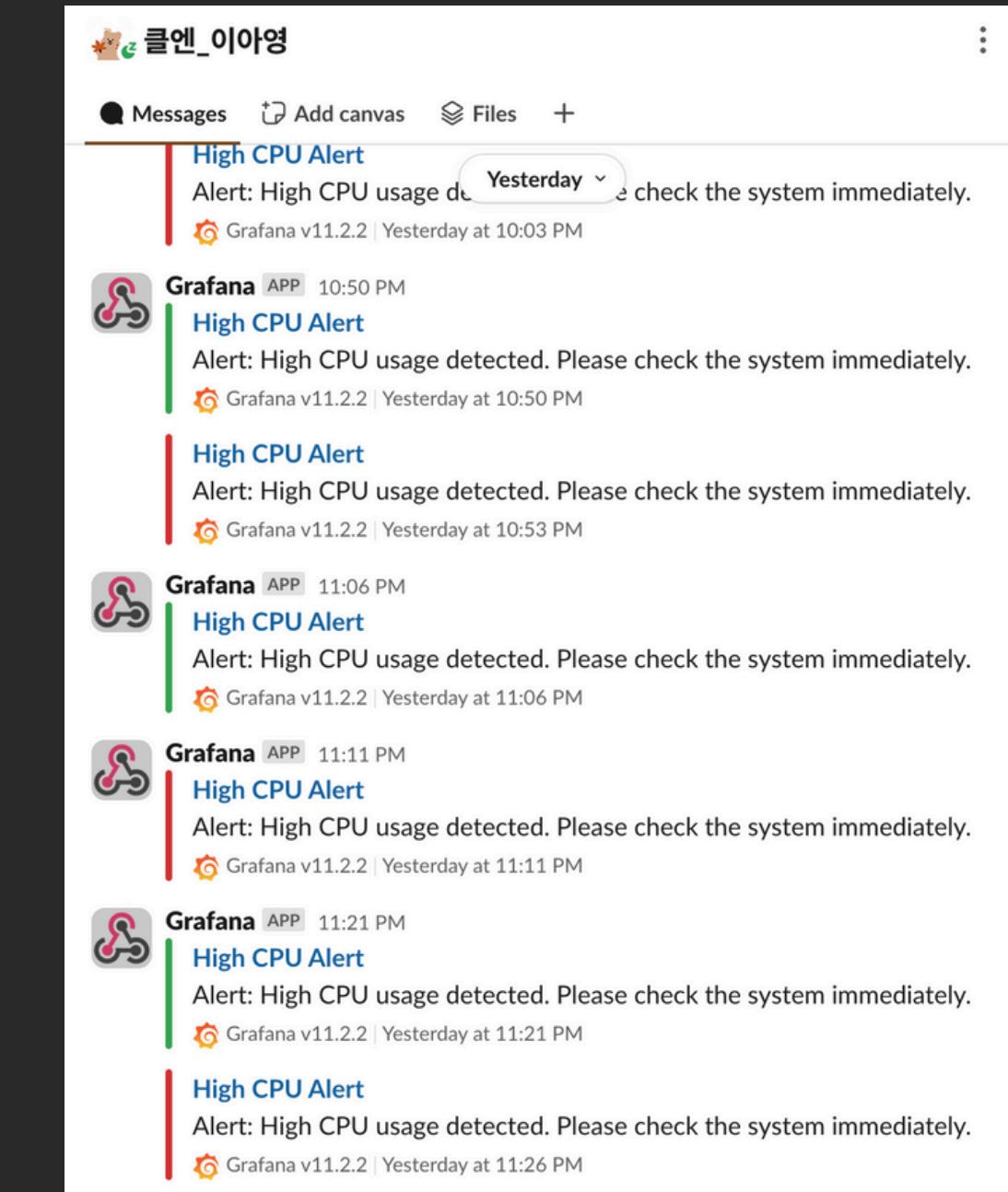


- MELT 데이터 추출
- Terraform 으로 연결
- Grafana로 시각화
- CPU 사용량 기준 초과시 슬랙 알림

Alert as Code

목표 : CPU 사용량이 임계치를 넘어가면 슬랙으로 알림 보내기

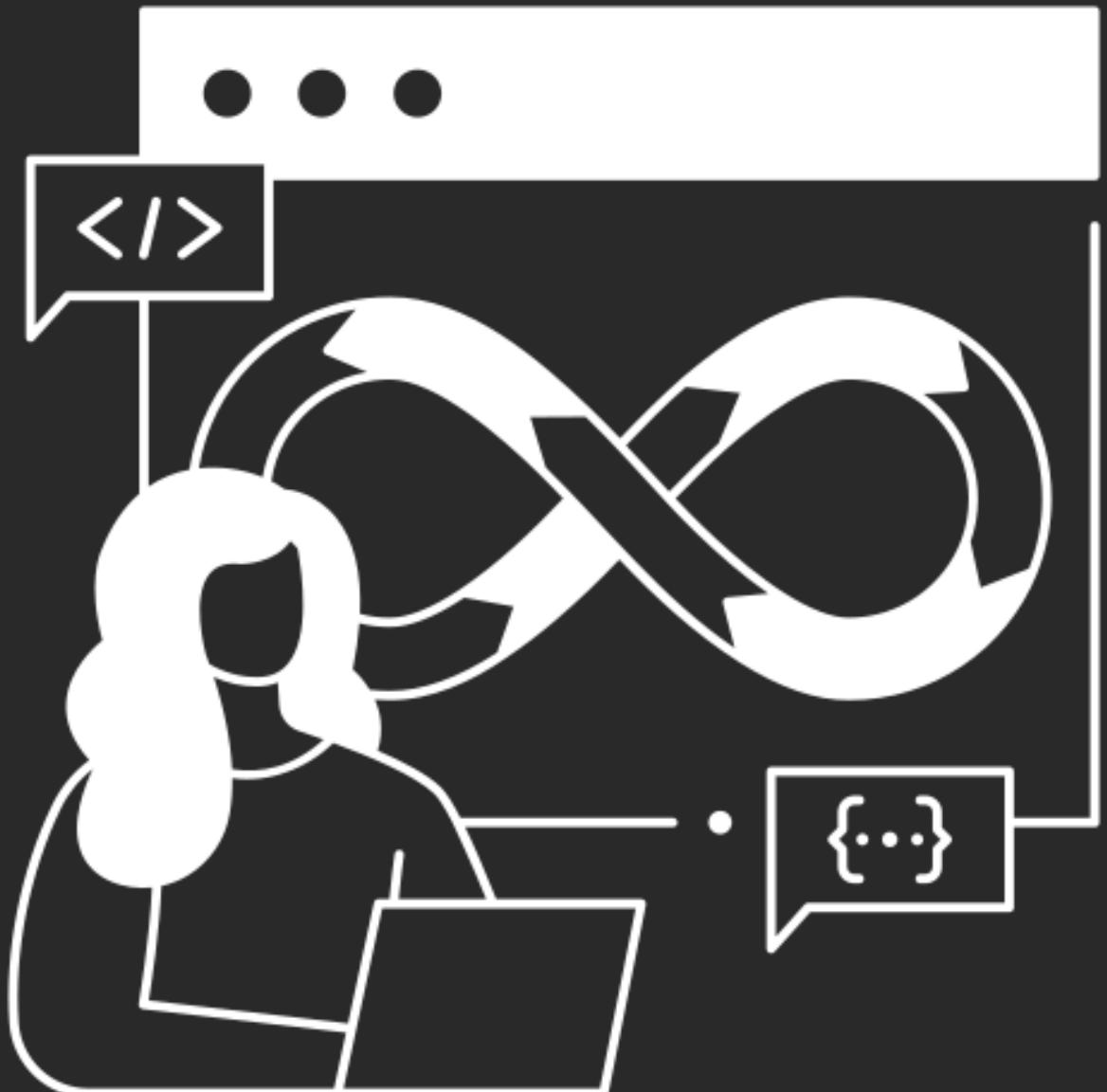
```
dashboard > Observability-as-Code > observability > contact-point-resource.tf
1  resource "grafana_contact_point" "slack_contact_point" {
2    name = "Slack-Alert"
3
4    slack {
5      url = "https://hooks.slack.com/services/"
6      icon_url = "https://grafana.com/static/img/fav32.png"
7      title = "High CPU Alert"
8      text = "Alert: High CPU usage detected. Please check the system immediately."
9    }
10 }
11
12 resource "grafana_notification_policy" "cpu_notification_policy" {
13   group_by      = ["alertname"]
14   contact_point = grafana_contact_point.slack_contact_point.name
15
16   policy [
17     matcher {
18       label = "alertname"
19       match = "="
20       value = "High CPU Usage Alert"
21     }
22     contact_point = grafana_contact_point.slack_contact_point.name
23   ]
24 }
25
26 resource "grafana_folder" "prometheus_alert"{
27   title = "Prometheus Alert Provisioning by Terraform"
28 }
29
30 > resource "grafana_rule_group" "my_rule_group" { ...
58 > { ...
75 }
76 > EOT ...
106 }
```



코드 확인 : <https://github.com/ayleeee/Observability-as-Code>

5. 결론

Observability as Code



- **Observability - MELT**
 - Metrics Events Logs Trace
- **Observability as Code**
 - 일관성 있고 효율적인 운영
 - 휴먼에러 방지
 - 신속하고 전략적인 대응 가능
 - 협업 효율 증가

감사합니다