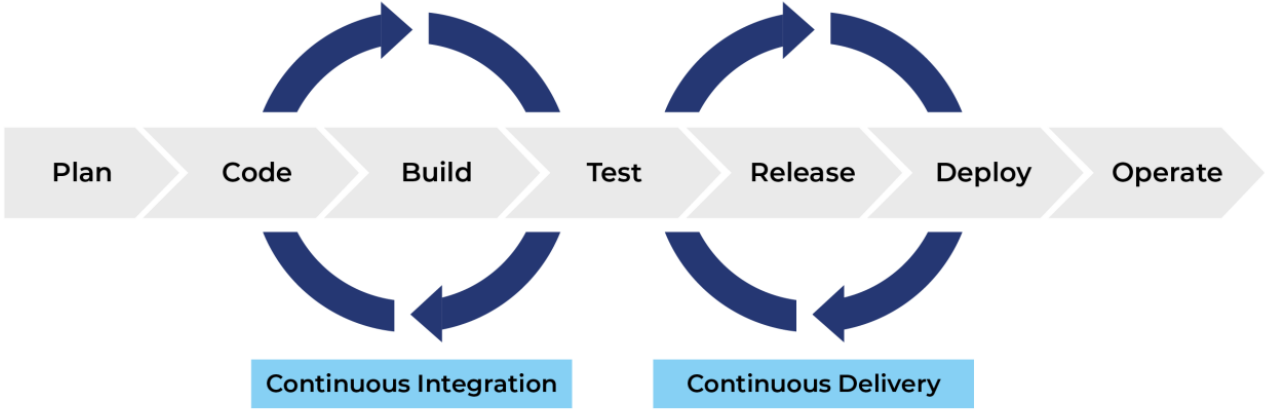
# 겨울방학 스터디
# 6회차

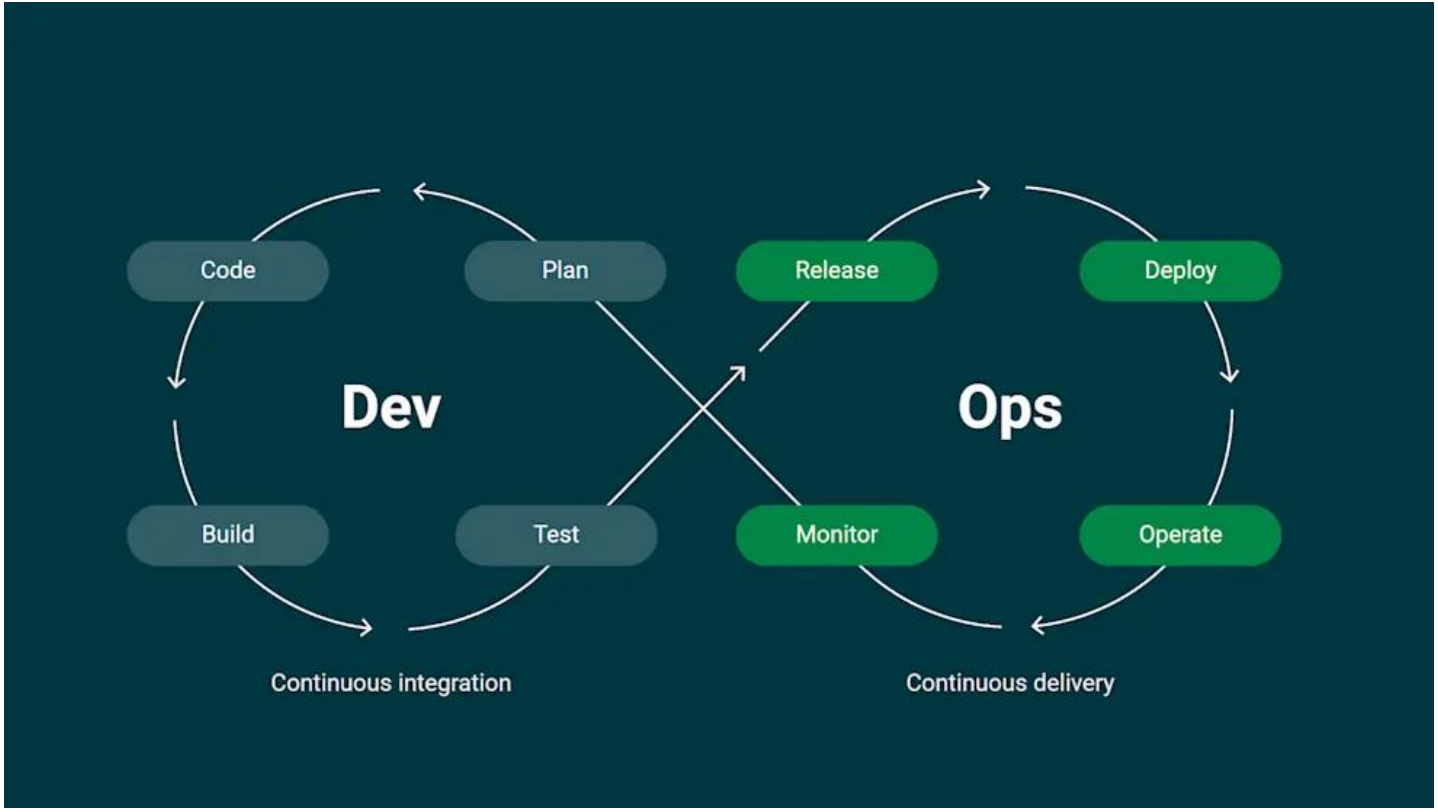
College of Art & Technology
Chung-Ang University

Complex Intelligent Systems Laboratory

https://cislab.cau.ac.kr

# CI/CD


Plan
Code
Build
Test
Release
Deploy
Operate
Continuous Integration
Continuous Delivery


Code
Plan
Release
Deploy
Dev
Ops
Build
Test
Monitor
Operate
Continuous integration
Continuous delivery

## Continuous Integration

지속적 통합

- 개발자가 코드를 공유 저장소에 자주 병합
- 병합할 때마다 자동으로 빌드 & 테스트 실행
- 버그를 조기에 발견하고 코드 품질 유지

## Continuous Delivery

지속적 전달(배포)

- 테스트 통과된 코드를 자동으로 배포 준비
- Delivery: 수동 승인 후 프로덕션 배포
- Deployment: 프로덕션까지 완전 자동화
- 배포 주기 단축, 안정적 릴리스 보장

## 수동 배포의 문제점

- 배포할 때마다 수작업 → 휴먼 에러 발생
- 테스트 안 돌리고 머지 → 프로덕션 장애
- 배포 주기 길어짐 → 피드백 루프 느림
- 롤백 어려움 → 장애 복구 시간 증가

## CI/CD 도입 후

- 코드 푸시 → 자동 빌드 & 테스트
- 테스트 실패 시 즉시 알림 → 빠른 수정
- 배포 자동화 → 일관된 배포 프로세스
- 빠른 릴리스 주기 → 빠른 피드백
- 자동 롤백 → 장애 복구 시간 단축


Push
코드 변경사항
원격 저장소 푸시
Build
소스 코드 빌드
의존성 설치
Test
자동화 테스트
실행 및 검증
Deploy
스테이징/프로덕션
서버에 배포
Monitor
서비스 상태
지표 모니터링
CI 영역
CD 영역

**GitHub Actions**

GitHub 기반 CI/CD
PR 자동 테스트, 자동 배포

**Jenkins**

자체 호스팅 자동화 서버
대규모 파이프라인 실행

**Prometheus + Grafana + Loki**

메트릭 수집 + 시각화 + 로그
배포 후 서비스 모니터링

## GitHub에 내장된 CI/CD 자동화 플랫폼

별도 서버 없이 GitHub 저장소에서 바로 워크플로우를 정의하고 실행할 수 있습니다.

### YAML 기반 설정

.github/workflows/ 폴더에
YAML 파일로 파이프라인 정의

### 클라우드 러너 제공

GitHub이 제공하는 가상머신에서
별도 서버 구축 없이 실행

### GitHub 완전 통합

PR, Issue, Release 등
GitHub 이벤트와 직접 연동

| | |
|---|---|
| **Workflow** | 자동화 프로세스 전체 단위. 하나의 YAML 파일 = 하나의 Workflow |
| **Event** | Workflow를 실행시키는 트리거. push, pull_request 등 |
| **Job** | Workflow 내 독립 실행 단위. 기본적으로 병렬 실행 |
| **Step** | Job 내 순차 실행되는 개별 작업. 명령어 또는 Action 실행 |

.github/workflows/ci.yml

```yaml
name: CI Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  build-and-test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Install dependencies
        run: npm install
      - name: Run tests
        run: npm test
```

| | |
|---|---|
| **name** | 워크플로우 이름 지정 |
| **on** | 트리거 이벤트 정의<br>(push, PR 등) |
| **jobs** | 실행할 작업 단위 정의 |
| **steps** | 각 Job의 세부 단계<br>순서대로 실행 |

| Event | 설명 | 사용 사례 |
| --- | --- | --- |
| `push` | 브랜치에 코드가 푸시될 때 실행 | 빌드/배포 자동화 |
| `pull_request` | PR 생성/업데이트 시 실행 | 코드 리뷰 전 자동 테스트 |
| `schedule` | cron 표현식으로 주기적 실행 | 정기 헬스체크, 백업 |
| `workflow_dispatch` | GitHub UI에서 수동 실행 | 긴급 배포, 수동 트리거 |
| `release` | 릴리스 생성/발행 시 실행 | 릴리스 빌드, 배포 |

# GitHub Actions - 사용 예시

PR 올리면 자동으로 Lint + Test 실행 → 통과해야 Merge 가능


```yaml
on:
  pull_request:
    branches: [main]
jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - run: npm run lint
      - run: npm test
```

# GitHub Actions - 사용 예시

main 브랜치에 Push 시 Docker 이미지 빌드 → 서버에 자동 배포

```yaml
on:
  push:
    branches: [main]
jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Build Docker image
        run: docker build -t app .
      - name: Push to Registry
        run: docker push registry/app
      - name: Deploy to Server
        run: ssh server 'docker pull && restart'
```

1. main에 코드 머지
2. Docker 이미지 빌드
3. 레지스트리에 Push
4. 서버에서 Pull & 재시작

# GitHub Actions - 사용 예시

```yaml
name: AI Code Review

on:
  pull_request:
    types: [opened, ready_for_review]

permissions:
  contents: read
  pull-requests: write
  issues: read

jobs:
  code-review:
    if: github.event_name == 'pull_request' && !github.event.pull_request.draft
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: |
          pip install anthropic PyGithub pyyaml

      - name: Get PR diff
        id: diff
        env:
          GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: |
          PR_NUMBER=${{ github.event.pull_request.number }}
          gh pr diff $PR_NUMBER > pr_diff.txt
          echo "pr_number=$PR_NUMBER" >> $GITHUB_OUTPUT
```

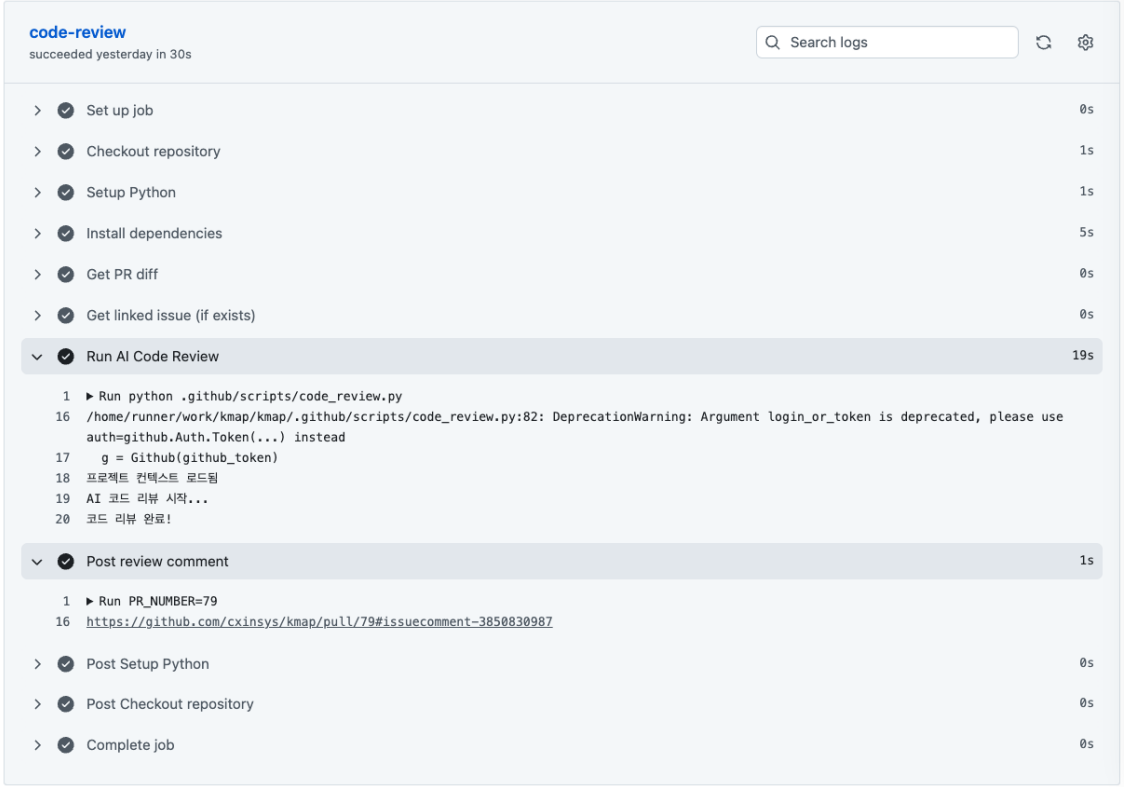
```yaml
      - name: Run AI Code Review
        env:
          ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }}
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          PR_NUMBER: ${{ steps.diff.outputs.pr_number }}
          REPO_NAME: ${{ github.repository }}
          ISSUE_CONTENT: ${{ steps.issue.outputs.issue_content }}
        run: |
          python .github/scripts/code_review.py

      - name: Post review comment
        env:
          GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: |
          PR_NUMBER=${{ steps.diff.outputs.pr_number }}

          if [ -f review_result.md ]; then
            gh pr comment $PR_NUMBER --body-file review_result.md
          fi
```

# GitHub Actions - 사용 예시

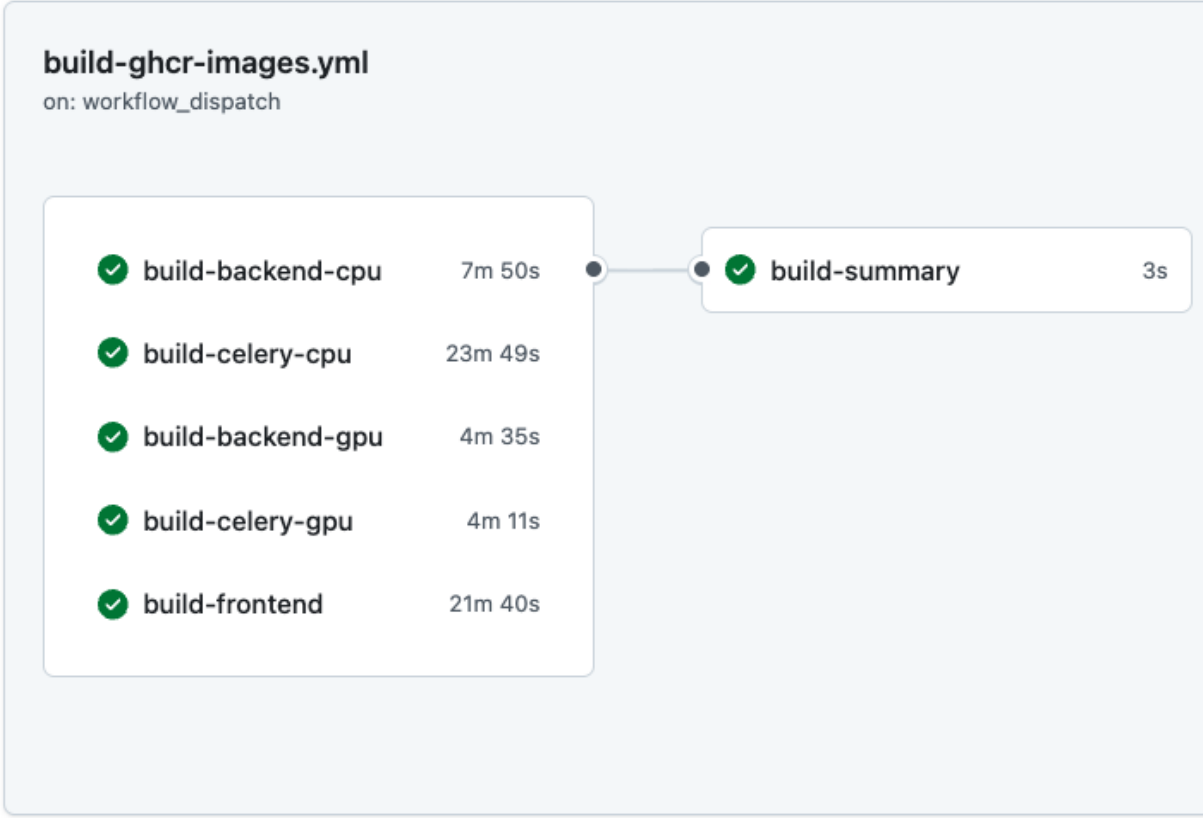
```yaml
name: AI Code Review

on:
  pull_request:
    types: [opened, ready_for_review]

permissions:
  contents: read
  pull-requests: write
  issues: read

jobs:
  code-review:
    if: github.event_name == 'pull_request' && !github.event.pull_request.draft
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4
        with:
          fetch-depth: 0

      - name: Setup Python
        uses: actions/setup-python@v5
        with:
          python-version: '3.11'

      - name: Install dependencies
        run: |
          pip install anthropic PyGithub pyyaml

      - name: Get PR diff
        id: diff
        env:
          GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: |
          PR_NUMBER=${{ github.event.pull_request.number }}
          gh pr diff $PR_NUMBER > pr_diff.txt
          echo "pr_number=$PR_NUMBER" >> $GITHUB_OUTPUT
```

```yaml
      - name: Run AI Code Review
        env:
          ANTHROPIC_API_KEY: ${{ secrets.ANTHROPIC_API_KEY }}
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          PR_NUMBER: ${{ steps.diff.outputs.pr_number }}
          REPO_NAME: ${{ github.repository }}
          ISSUE_CONTENT: ${{ steps.issue.outputs.issue_content }}
        run: |
          python .github/scripts/code_review.py

      - name: Post review comment
        env:
          GH_TOKEN: ${{ secrets.GITHUB_TOKEN }}
        run: |
          PR_NUMBER=${{ steps.diff.outputs.pr_number }}

          if [ -f review_result.md ]; then
            gh pr comment $PR_NUMBER --body-file review_result.md
          fi
```

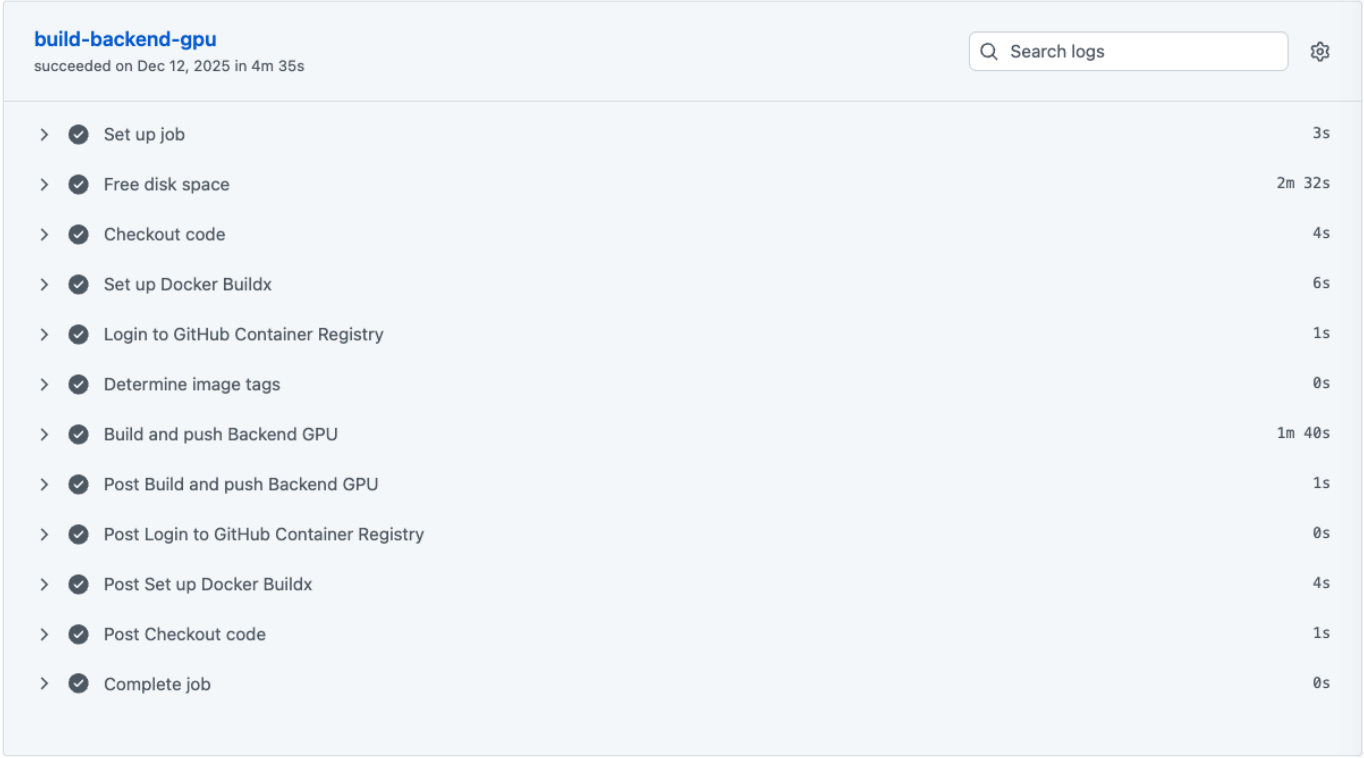
# GitHub Actions - 사용 예시



# GitHub Actions - 사용 예시



# GitHub Actions - 사용 예시



# GitHub Actions - 사용 예시

13 packages

| | | |
|---|---|---|
| **frontend**<br>Published on Sep 11, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft** | 587 |
| **backend-cpu**<br>Published on Sep 23, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft** | 142 |
| **celery-cpu**<br>Published on Sep 23, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft** | 129 |
| **cellcraft-genie3**<br>Published on Aug 23, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft-plugin** | 129 |
| **cellcraft-grnviz**<br>Published on Aug 25, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft-plugin** | 127 |
| **cellcraft-grnboost2**<br>Published on Aug 23, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft-plugin** | 127 |
| **cellcraft-leap**<br>Published on Aug 23, 2025 by Complex Intelligent Systems Lab. in **cxinsys/cellcraft-plugin** | 124 |

## 장점

- GitHub과 완벽한 통합 (PR, Issue 연동)
- 무료 러너 제공 (공개 저장소 무제한)
- YAML 기반 간단한 설정
- 커뮤니티 액션으로 빠른 구축

## 한계

- 복잡한 파이프라인 구성 시 한계
- 무료 러너 스펙 제한 (2 core, 7GB RAM)
- 비공개 저장소 월 사용량 제한
- 세밀한 인프라 제어 어려움

## 가장 널리 사용되는 오픈소스 자동화 서버

자체 서버에 설치하여 빌드, 테스트, 배포를 자동화하는 도구. 1800개 이상의 플러그인 생태계.

### 자체 호스팅

자체 서버에 설치하여 운영
인프라에 대한 완전한 제어

### 플러그인 생태계

1800개+ 플러그인으로
거의 모든 도구와 연동

### 파이프라인 as Code

Jenkinsfile로 파이프라인을
코드로 관리 (버전 관리 가능)

| | |
|---|---|
| **Pipeline** | 빌드~배포까지의 전체 자동화 프로세스를 정의하는 단위 |
| **Stage** | Pipeline 내 논리적 단계 구분 (Build, Test, Deploy 등) |
| **Agent / Node** | Pipeline이 실행되는 머신. Master-Agent 구조로 분산 빌드 가능 |
| **Jenkinsfile** | Pipeline을 코드로 정의하는 파일. 저장소에 함께 버전 관리 |
| **Plugin** | Jenkins 기능을 확장하는 모듈. Docker, Slack, Git 등 연동 |

## Declarative Pipeline 기본 구조

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                sh 'npm install'
                sh 'npm run build'
            }
        }
        stage('Test') {
            steps {
                sh 'npm test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'docker build -t app .'
                sh 'docker push registry/app'
            }
        }
    }
}
```

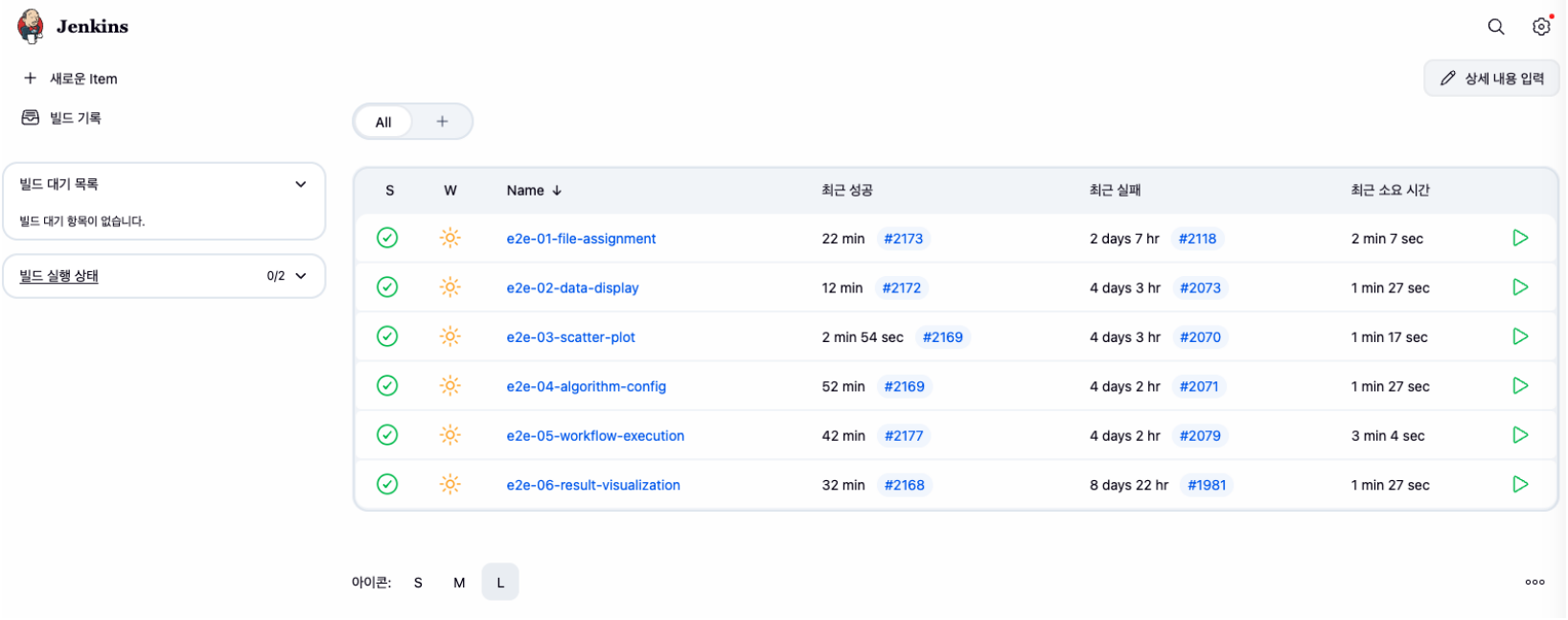
| | |
|---|---|
| **pipeline** | 최상위 블록<br>전체 파이프라인 정의 |
| **agent** | 실행 환경 지정<br>any, docker, label 등 |
| **stages** | 단계(Stage) 목록<br>순차적으로 실행 |
| **steps** | 각 Stage의 실행 명령<br>sh, bat, script 등 |

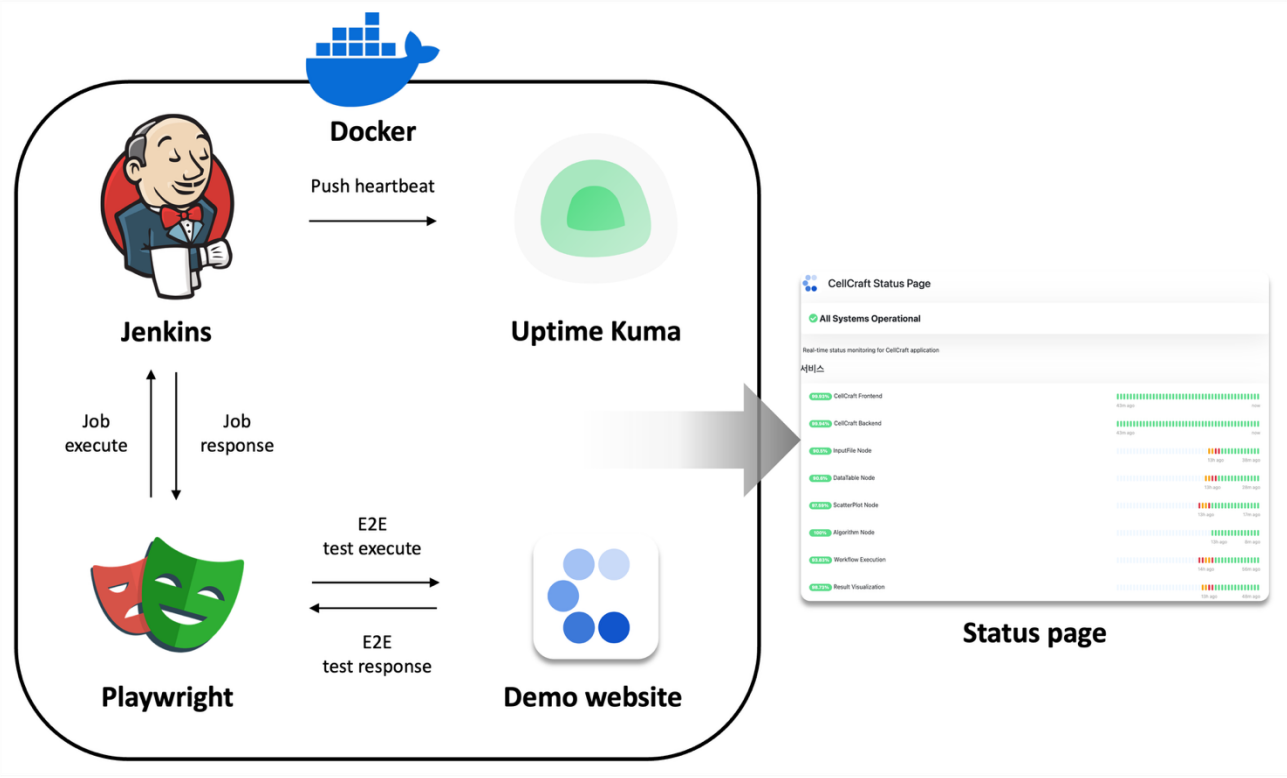
## Jenkins가 적합한 경우

- 자체 인프라에서 운영해야 할 때
- 복잡한 멀티 스테이지 파이프라인
- 세밀한 빌드 환경 제어가 필요할 때
- 대규모 팀, 많은 프로젝트 관리

## GitHub Actions가 적합한 경우

- 소~중규모 프로젝트
- GitHub 중심 워크플로우
- 빠르게 CI/CD 시작하고 싶을 때
- 서버 관리 부담을 줄이고 싶을 때
- 오픈소스 프로젝트


Jenkins
새로운 Item
빌드 기록
상세 내용 입력
All
빌드 대기 목록
빌드 대기 항목이 없습니다.
빌드 실행 상태 0/2
S W Name ↓ 최근 성공 최근 실패 최근 소요 시간
e2e-01-file-assignment 22 min #2173 2 days 7 hr #2118 2 min 7 sec
e2e-02-data-display 12 min #2172 4 days 3 hr #2073 1 min 27 sec
e2e-03-scatter-plot 2 min 54 sec #2169 4 days 3 hr #2070 1 min 17 sec
e2e-04-algorithm-config 52 min #2169 4 days 2 hr #2071 1 min 27 sec
e2e-05-workflow-execution 42 min #2177 4 days 2 hr #2079 3 min 4 sec
e2e-06-result-visualization 32 min #2168 8 days 22 hr #1981 1 min 27 sec
아이콘: S M L


Docker
Push heartbeat
Jenkins
Uptime Kuma
Job execute
Job response
E2E test execute
E2E test response
Playwright
Demo website
CellCraft Status Page
All Systems Operational
Status page

*"배포했는데... 잘 돌아가고 있는 거 맞아?"*

### 장애 감지

서비스 다운, 에러 급증을
실시간으로 감지

### 성능 병목 파악

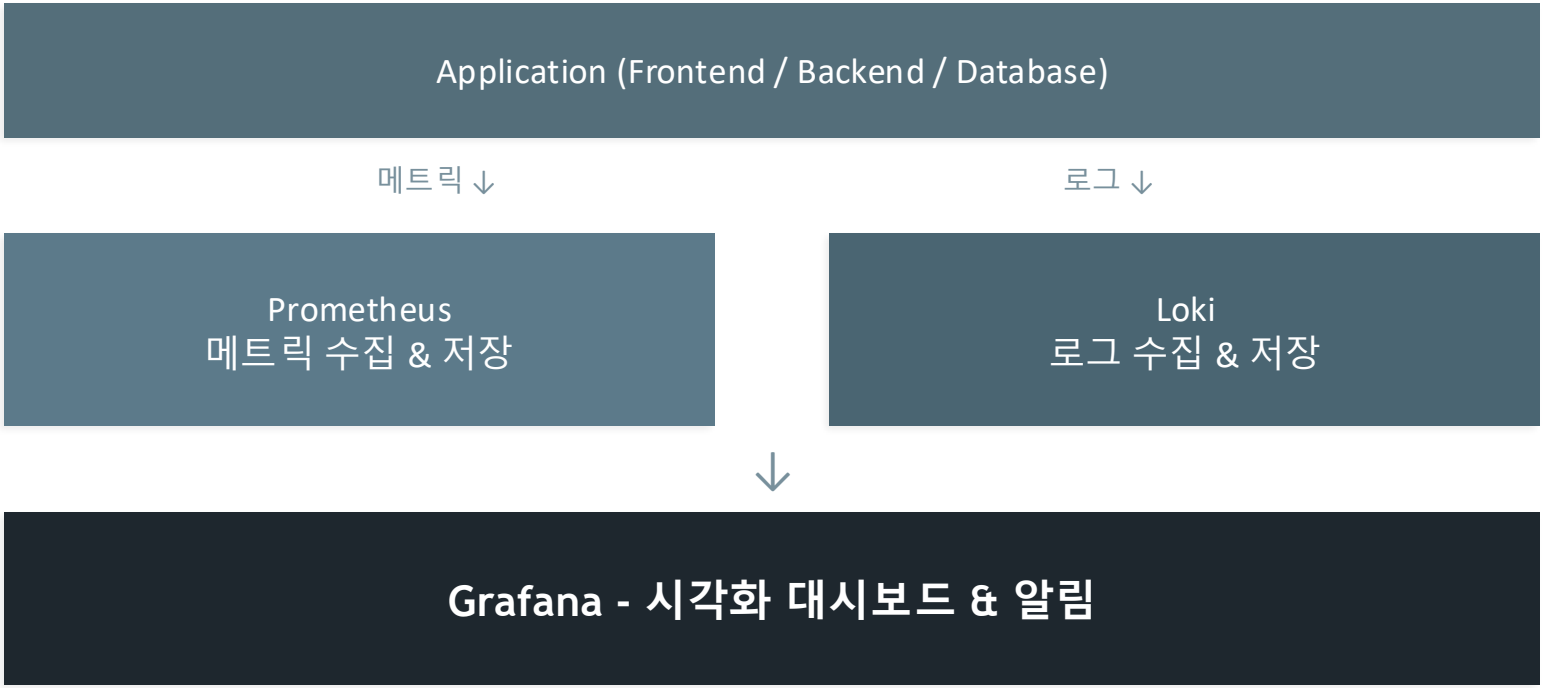
느린 API, 높은 CPU 사용률 등
병목 지점 식별

### 사전 예방

메모리 증가 추이 등
문제 발생 전 미리 대응

### 데이터 기반 의사결정

트래픽 패턴, 사용량 분석으로
인프라 계획 수립


Application (Frontend / Backend / Database)
메트릭 ↓
로그 ↓
Prometheus
메트릭 수집 & 저장
Loki
로그 수집 & 저장
Grafana - 시각화 대시보드 & 알림


Prometheus


Grafana


Grafana loki


## 시계열(Time-series) 메트릭 수집 및 저장 도구

숫자로 표현되는 시스템 지표를 주기적으로 수집하고 시계열 DB에 저장합니다.

### Pull 방식 수집

Prometheus가 타겟에 직접 접근하여
메트릭을 가져오는 방식 (Scrape)

### PromQL

강력한 쿼리 언어로
메트릭 조회/집계/분석

### Alert Rules

조건 기반 알림 규칙 설정
(예: CPU > 80% 5분 지속)

## 메트릭의 4가지 타입

| | | |
|---|---|---|
| **Counter** | 단조 증가하는 누적 값 | 총 HTTP 요청 수, 에러 발생 횟수 |
| **Gauge** | 올라가거나 내려가는 현재 값 | 현재 CPU 사용률, 메모리 사용량 |
| **Histogram** | 값의 분포를 버킷으로 측정 | 요청 응답 시간 분포 (p50, p95, p99) |
| **Summary** | 클라이언트 측 분위수 계산 | 요청 지속 시간의 중앙값, 95th 백분위 |


## 데이터 시각화 및 모니터링 대시보드 도구

Prometheus, Loki 등 다양한 데이터소스를 연결하여 통합 대시보드를 구성합니다.

### 다양한 데이터소스

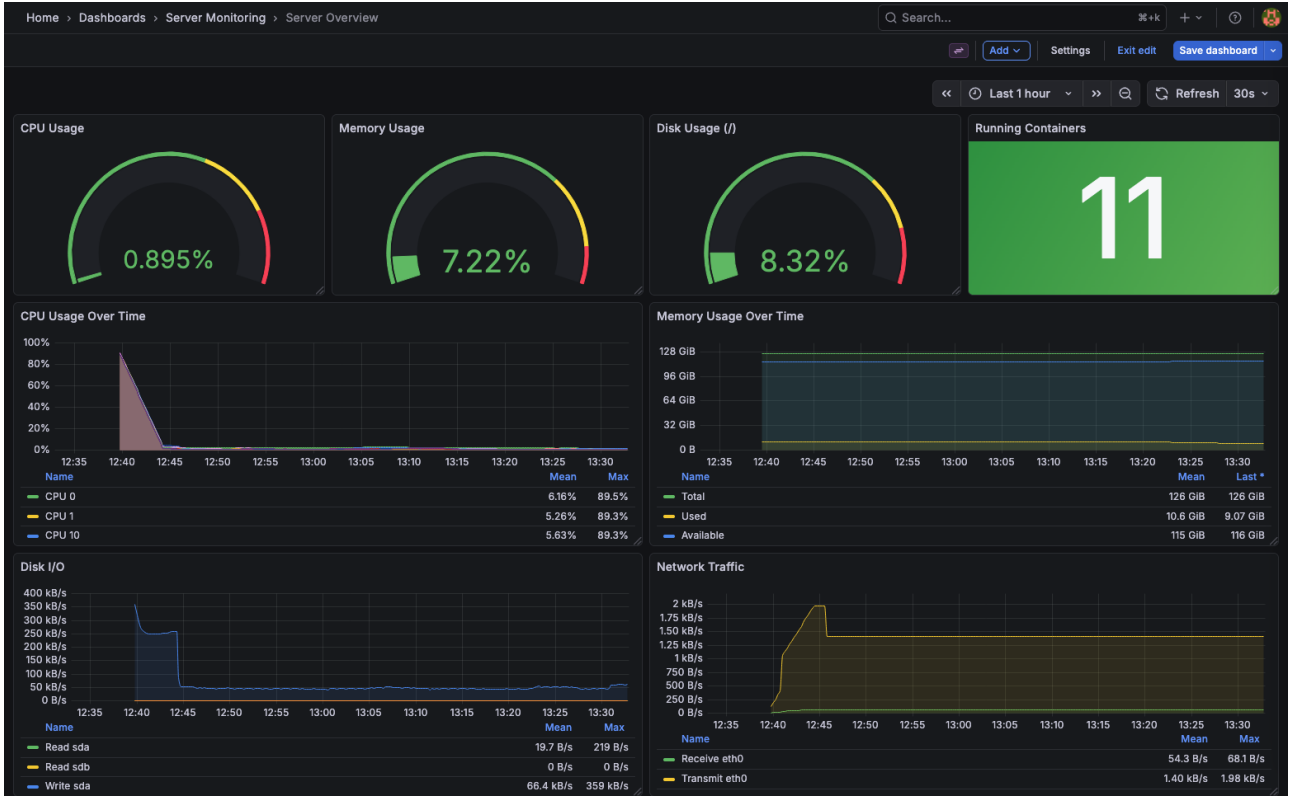
Prometheus, Loki, MySQL,
Elasticsearch 등 40개+ 지원

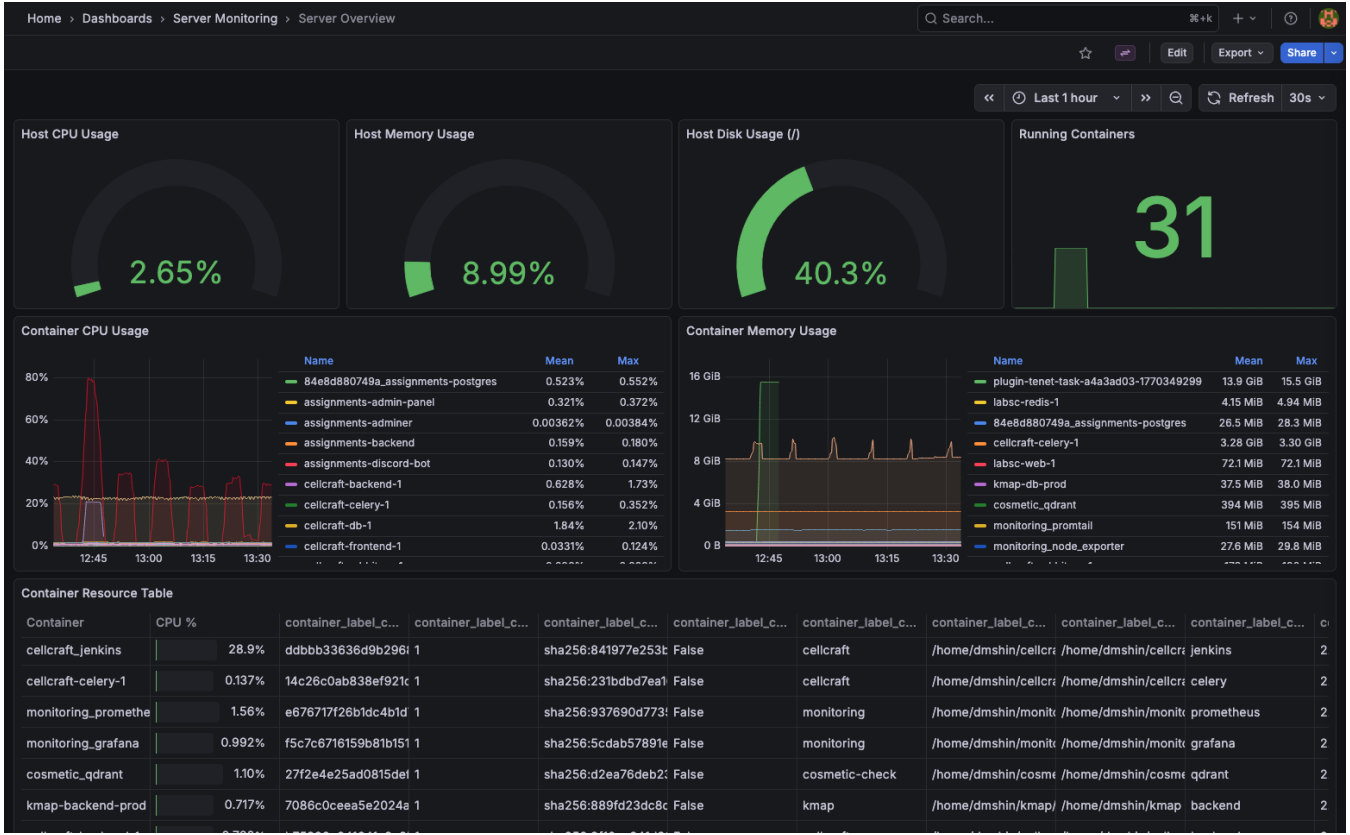
### 알림 설정

조건 기반 알림 → Slack,
이메일, 웹훅 전송

### 시각화

그래프, 게이지, 히트맵,
테이블 등 다양한 패널


Home › Dashboards › Server Monitoring › Server Overview
Search...
Add
Settings
Exit edit
Save dashboard
Last 1 hour
Refresh
30s
CPU Usage
0.895%
Memory Usage
7.22%
Disk Usage (/)
8.32%
Running Containers
11
CPU Usage Over Time
100%
80%
60%
40%
20%
0%
12:35 12:40 12:45 12:50 12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30
Name Mean Max
CPU 0 6.16% 89.5%
CPU 1 5.26% 89.3%
CPU 10 5.63% 89.3%
Memory Usage Over Time
128 GiB
96 GiB
64 GiB
32 GiB
0 B
12:35 12:40 12:45 12:50 12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30
Name Mean Last *
Total 126 GiB 126 GiB
Used 10.6 GiB 9.07 GiB
Available 115 GiB 116 GiB
Disk I/O
400 kB/s
350 kB/s
300 kB/s
250 kB/s
200 kB/s
150 kB/s
100 kB/s
50 kB/s
0 B/s
12:35 12:40 12:45 12:50 12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30
Name Mean Max
Read sda 19.7 B/s 219 B/s
Read sdb 0 B/s 0 B/s
Write sda 66.4 kB/s 359 kB/s
Network Traffic
2 kB/s
1.75 kB/s
1.50 kB/s
1.25 kB/s
1 kB/s
750 B/s
500 B/s
250 B/s
0 B/s
12:35 12:40 12:45 12:50 12:55 13:00 13:05 13:10 13:15 13:20 13:25 13:30
Name Mean Max
Receive eth0 54.3 B/s 68.1 B/s
Transmit eth0 1.40 kB/s 1.98 kB/s


Home › Dashboards › Server Monitoring › Server Overview
Search...
Edit
Export
Share
Last 1 hour
Refresh
30s
Host CPU Usage
2.65%
Host Memory Usage
8.99%
Host Disk Usage (/)
40.3%
Running Containers
31
Container CPU Usage
80%
60%
40%
20%
0%
12:45
13:00
13:15
13:30
Name
Mean
Max
84e8d880749a_assignments-postgres 0.523% 0.552%
assignments-admin-panel 0.321% 0.372%
assignments-adminer 0.00362% 0.00384%
assignments-backend 0.159% 0.180%
assignments-discord-bot 0.130% 0.147%
cellcraft-backend-1 0.628% 1.73%
cellcraft-celery-1 0.156% 0.352%
cellcraft-db-1 1.84% 2.10%
cellcraft-frontend-1 0.0331% 0.124%
Container Memory Usage
16 GiB
12 GiB
8 GiB
4 GiB
0 B
12:45
13:00
13:15
13:30
Name
Mean
Max
plugin-tenet-task-a4a3ad03-1770349299 13.9 GiB 15.5 GiB
labsc-redis-1 4.15 MiB 4.94 MiB
84e8d880749a_assignments-postgres 26.5 MiB 28.3 MiB
cellcraft-celery-1 3.28 GiB 3.30 GiB
labsc-web-1 72.1 MiB 72.1 MiB
kmap-db-prod 37.5 MiB 38.0 MiB
cosmetic_qdrant 394 MiB 395 MiB
monitoring_promtail 151 MiB 154 MiB
monitoring_node_exporter 27.6 MiB 29.8 MiB
Container Resource Table
Container
CPU %
container_label_c...
container_label_c...
container_label_c...
container_label_c...
container_label_c...
container_label_c...
container_label_c...
container_label_c...
cellcraft_jenkins 28.9% ddbbb33636d9b296i 1 sha256:841977e253b False cellcraft /home/dmshin/cellcra /home/dmshin/cellcra jenkins 2
cellcraft-celery-1 0.137% 14c26c0ab838ef921c 1 sha256:231bdbd7ea1 False cellcraft /home/dmshin/cellcra /home/dmshin/cellcra celery 2
monitoring_promethe 1.56% e676717f26b1dc4b1d 1 sha256:937690d773 False monitoring /home/dmshin/monit /home/dmshin/monit prometheus 2
monitoring_grafana 0.992% f5c7c6716159b81b151 1 sha256:5cdab57891e False monitoring /home/dmshin/monit /home/dmshin/monit grafana 2
cosmetic_qdrant 1.10% 27f2e4e25ad0815de 1 sha256:d2ea76deb2 False cosmetic-check /home/dmshin/cosme /home/dmshin/cosme qdrant 2
kmap-backend-prod 0.717% 7086c0ceea5e2024a 1 sha256:889fd23dc8c False kmap /home/dmshin/kmap/ /home/dmshin/kmap backend 2

**cellcraft monitoring** 앱 2026. 2. 4. 오후 8:51

**Firing**

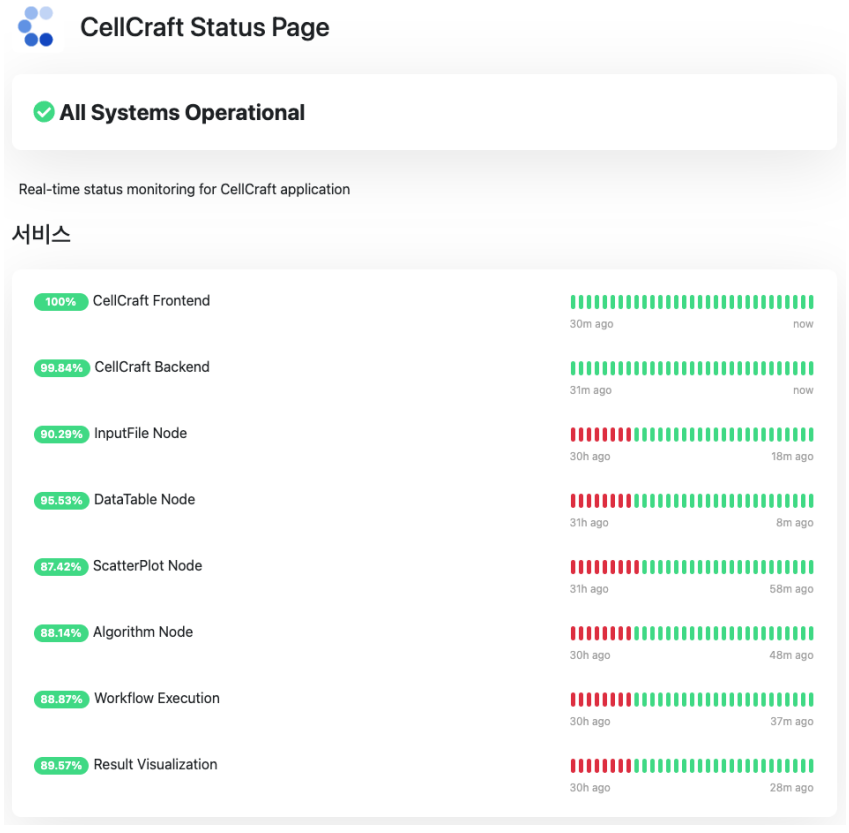
Value: B=14.39456558227539, C=1
Labels:

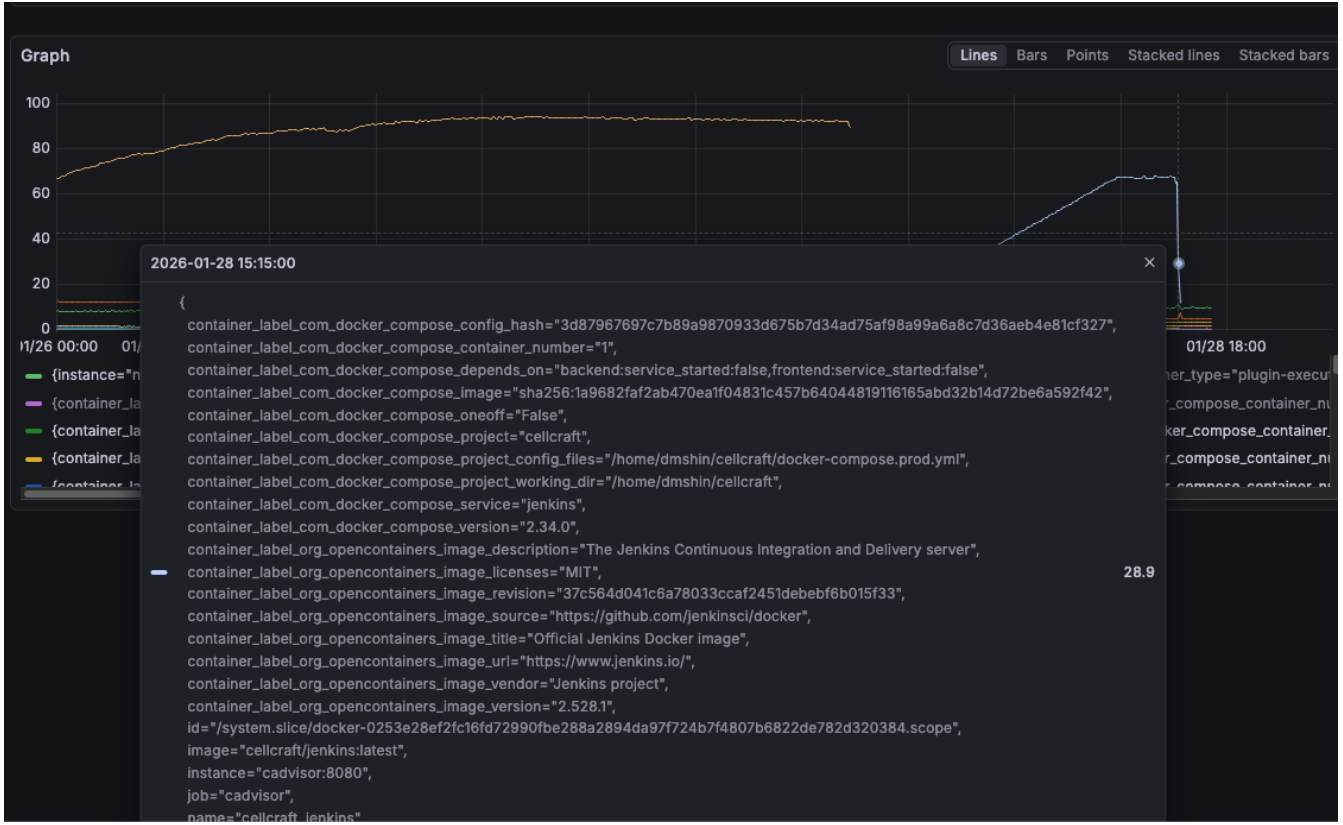
- alertname = Container High Memory Usage
- container_label_celery_task_id = 3b4fd90d-ce43-442a-860b-516b54224536
- container_label_container_type = plugin-execution
- container_label_plugin_name = TENET
- grafana_folder = Server Monitoring
- id = /system.slice/docker-42eef32f469b2951fbf4b3777100c1c6b083ce4d644d73cccdf4e0e2a644dfcf.scope
- image = ghcr.io/cxinsys/cellcraft-tenet:1.0
- instance = cadvisor:8080
- job = cadvisor
- name = plugin-tenet-task-3b4fd90d-1770205306
- severity = warning

Annotations:

- description = 최대 컨테이너 메모리 사용량: 14.4GB
- summary = 컨테이너 메모리 사용량이 8GB를 초과했습니다

Source: http://localhost:3002/alerting/grafana/container-high-memory/view?orgId=1
Silence: http://localhost:3002/alerting/silence/new?alertmanager=grafana&matcher=__alert_rule_uid__%3Dcontainer-high-memory&matcher=container_label_celery_task_id%3D3b4fd90d-ce43-442a-860b-516b54224536&matcher=container_label_container_type%3Dplugin-execution&matcher=container_label_plugin_name%3DTENET&matcher=id%3D%2Fsystem.slice%2Fdocker-42eef32f469b2951fbf4b3777100c1c6b083ce4d644d73cccdf4e0e2a644dfcf.scope&matcher=image%3Dghcr.io%2Fcxinsys%2Fcellcraft-tenet%3A1.0&matcher=instance%3Dcadvisor%3A8080&matcher=job%3Dcadvisor&matcher=name%3Dplugin-tenet-task-3b4fd90d-1770205306&matcher=severity%3Dwarning&orgId=1


CellCraft Status Page
All Systems Operational
Real-time status monitoring for CellCraft application
서비스
100% CellCraft Frontend
30m ago now
99.84% CellCraft Backend
31m ago now
90.29% InputFile Node
30h ago 18m ago
95.53% DataTable Node
31h ago 8m ago
87.42% ScatterPlot Node
31h ago 58m ago
88.14% Algorithm Node
30h ago 48m ago
88.87% Workflow Execution
30h ago 37m ago
89.57% Result Visualization
30h ago 28m ago


Graph
Lines
Bars
Points
Stacked lines
Stacked bars
100
80
60
40
20
0
01/26 00:00
01/28 18:00
2026-01-28 15:15:00
{
container_label_com_docker_compose_config_hash="3d87967697c7b89a9870933d675b7d34ad75af98a99a6a8c7d36aeb4e81cf327",
container_label_com_docker_compose_container_number="1",
container_label_com_docker_compose_depends_on="backend:service_started:false,frontend:service_started:false",
container_label_com_docker_compose_image="sha256:1a9682faf2ab470ea1f04831c457b64044819116165abd32b14d72be6a592f42",
container_label_com_docker_compose_oneoff="False",
container_label_com_docker_compose_project="cellcraft",
container_label_com_docker_compose_project_config_files="/home/dmshin/cellcraft/docker-compose.prod.yml",
container_label_com_docker_compose_project_working_dir="/home/dmshin/cellcraft",
container_label_com_docker_compose_service="jenkins",
container_label_com_docker_compose_version="2.34.0",
container_label_org_opencontainers_image_description="The Jenkins Continuous Integration and Delivery server",
container_label_org_opencontainers_image_licenses="MIT",
28.9
container_label_org_opencontainers_image_revision="37c564d041c6a78033ccaf2451debebf6b015f33",
container_label_org_opencontainers_image_source="https://github.com/jenkinsci/docker",
container_label_org_opencontainers_image_title="Official Jenkins Docker image",
container_label_org_opencontainers_image_url="https://www.jenkins.io/",
container_label_org_opencontainers_image_vendor="Jenkins project",
container_label_org_opencontainers_image_version="2.528.1",
id="/system.slice/docker-0253e28ef2fc16fd72990fbe288a2894da97f724b7f4807b6822de782d320384.scope",
image="cellcraft/jenkins:latest",
instance="cadvisor:8080",
job="cadvisor",
name="cellcraft_jenkins"


## 로그 수집 및 조회 도구 (Prometheus의 로그 버전)

Grafana와 완전 통합되어, 메트릭 그래프에서 바로 해당 시점의 로그를 조회할 수 있습니다.

| | |
|---|---|
| **라벨 기반 인덱싱** | 로그 내용 전체가 아닌 라벨(메타데이터)로 인덱싱하여 저장 비용 절감 |
| **Grafana 통합** | Grafana에서 Prometheus 메트릭과 Loki 로그를 나란히 조회 가능 |
| **LogQL 쿼리** | PromQL과 유사한 문법으로 로그 필터링/집계 |


Home › Dashboards › Server Monitoring › Log Explorer
Search...
Edit
Export
Share
Container
All
Last 1 hour
Refresh
10s
Log Volume by Container
400
350
300
250
200
150
100
50
0
12:45
12:50
12:55
13:00
13:05
13:10
13:15
13:20
13:25
13:30
13:35
13:40
assignments-backend
cellcraft-backend-1
cellcraft-celery-1
cellcraft-db-1
cellcraft-frontend-1
cellcraft_uptime_kuma
cosmetic_api
Error Count (Selected Range)
315
Warning Count (Selected Range)
83
Total Log Lines (Selected Range)
11212
Log Stream
2026-02-06 13:42:45.169 """
2026-02-06 13:42:45.169 billiard.exceptions.Terminated: 15
2026-02-06 13:42:45.169 raise Terminated(-(signum or 0))
2026-02-06 13:42:45.169 File "/opt/micromamba/envs/cellcraft-test/lib/python3.10/site-packages/billiard/pool.py", line 1771, in _set_terminated
2026-02-06 13:42:45.169 Traceback (most recent call last):
2026-02-06 13:42:45.169 """
2026-02-06 13:42:45.169 WARN [2026-02-06 13:42:45,168: WARNING/MainProcess] Failure detected in task workflow_task:process_data_task:
2026-02-06 13:42:45.140 WARN [2026-02-06 13:42:45,140: WARNING/ForkPoolWorker-134] Container unregistered: Task 471d7d4a-73e4-46ce-a6c4-7815dbee633e

## Prometheus

*"얼마나?" → 숫자 메트릭*

- CPU 사용률: 85%
- 메모리: 2.1GB / 4GB
- HTTP 요청: 1,200 req/s
- 응답 시간 p95: 250ms
- 에러율: 2.5%

## Loki

*"무슨 일이?" → 로그 텍스트*

- ERROR: DB connection timeout
- WARN: Memory usage high
- INFO: User login from 1.2.3.4
- ERROR: NullPointerException
- DEBUG: Query took 3.2s

두 도구는 상호 보완적 → Grafana에서 메트릭 이상 감지 후 Loki로 원인 로그 추적

## CPU 사용률

**정상 범위**
일반적으로 40~60% 유지

**주의 신호**
80% 이상 5분 이상 지속

**위험 신호**
95%+ 지속 → 서비스 응답 지연/장애

**확인 포인트**
어떤 프로세스가 CPU를 많이 쓰는지 확인

## Memory 사용량

**정상 범위**
전체 대비 60~70% 이하

**메모리 누수 패턴**
시간이 지나도 계속 증가 (해제 안 됨)

**OOM Kill**
메모리 한도 초과 → 컨테이너 강제 종료

**확인 포인트**
재시작 없이 메모리가 계속 올라가는지 추이 확인

## RED 메서드 (Rate, Errors, Duration)

### Rate (RPS)

초당 요청 수

트래픽 규모 파악. 평소 대비 급증/급감 시 이상 징후.
예: 평소 500 RPS → 갑자기 50 RPS = 문제 발생 가능

### Errors (에러율)

4xx / 5xx 비율

4xx: 클라이언트 오류 (잘못된 요청)
5xx: 서버 오류 (코드 버그, DB 장애 등)
정상: 에러율 1% 미만

### Duration (Latency)

응답 시간

p50: 절반의 요청이 이 시간 내 응답
p95: 95%의 요청이 이 시간 내 응답
p99: 꼬리 지연 (tail latency) 확인

**컨테이너 재시작 횟수**

재시작이 잦다면 OOM Kill, 크래시 등 문제 발생 중.
정상: 0회. 반복 재시작은 즉시 원인 조사 필요.

**컨테이너별 리소스 사용량**

각 컨테이너의 CPU/Memory 사용량 개별 추적.
어떤 서비스가 리소스를 많이 쓰는지 파악.

**네트워크 I/O**

컨테이너 간 통신량, 외부 요청량 모니터링.
비정상적 트래픽 급증 시 공격 또는 장애 의심.

## "응답 시간이 갑자기 늘었다"

| | | |
|---|---|---|
| 1 | Grafana 대시보드 확인 | 어느 시점부터 Latency가 증가했는지 확인<br>특정 엔드포인트에서만 느린지 전체적인지 파악 |
| 2 | CPU / Memory 확인 | 리소스 부족으로 인한 지연인지 확인<br>CPU 100% 또는 메모리 부족 여부 체크 |
| 3 | DB 쿼리 시간 확인 | Slow Query 급증 여부 확인<br>커넥션 풀 고갈 여부 체크 |
| 4 | Loki 로그 확인 | 해당 시간대 에러 로그 검색<br>Timeout, Connection refused 등 단서 탐색 |

## "에러율이 급증했다"

| | | |
|---|---|---|
| 1 | 에러 유형 분류 | 4xx vs 5xx 비율 확인<br>4xx: 클라이언트 문제 / 5xx: 서버 문제 |
| 2 | 배포 시점 확인 | 최근 배포 이후 에러가 시작되었는지 확인<br>배포 직후라면 새 코드가 원인일 가능성 높음 |
| 3 | Loki 로그로 원인 추적 | 5xx 에러의 스택 트레이스 확인<br>NullPointer, Timeout, DB 오류 등 구체적 원인 파악 |
| 4 | 대응 | 원인 파악 시 핫픽스 배포 또는 롤백<br>알림 규칙 추가하여 재발 방지 |

## "메모리가 계속 올라간다"

| | | |
|---|---|---|
| 1 | 추이 그래프 확인 | 메모리 사용량이 계단식이 아닌 우상향 직선이면 누수 의심<br>재시작 후에도 같은 패턴 반복되는지 확인 |
| 2 | 컨테이너별 확인 | 어떤 컨테이너(Frontend/Backend/DB)에서 증가하는지 특정<br>특정 서비스만 문제면 해당 서비스 코드 점검 |
| 3 | GC / 캐시 확인 | Java: GC 로그 확인 / Node.js: 힙 메모리 추적<br>캐시 만료 정책 미설정 시 무한 증가 가능 |
| 4 | OOM Kill 방지 | 메모리 제한(limit) 설정 확인<br>OOM Kill 발생 전 알림 설정 (예: 메모리 80% 초과 시) |

## 알림 채널

- Prometheus AlertManager → 규칙 기반 알림 발송
- Grafana Alert → 대시보드 패널 조건 기반
- 전달: Slack, 이메일, 웹훅, PagerDuty 등

## 권장 알림 조건

- CPU > 80% (5분 이상 지속)
- 메모리 > 85%
- 5xx 에러율 > 5%
- 컨테이너 재시작 발생

## Code

개발자가 코드 작성
Git으로 관리

## CI

자동 빌드 & 테스트
GitHub Actions / Jenkins

## CD

자동 배포
스테이징 → 프로덕션

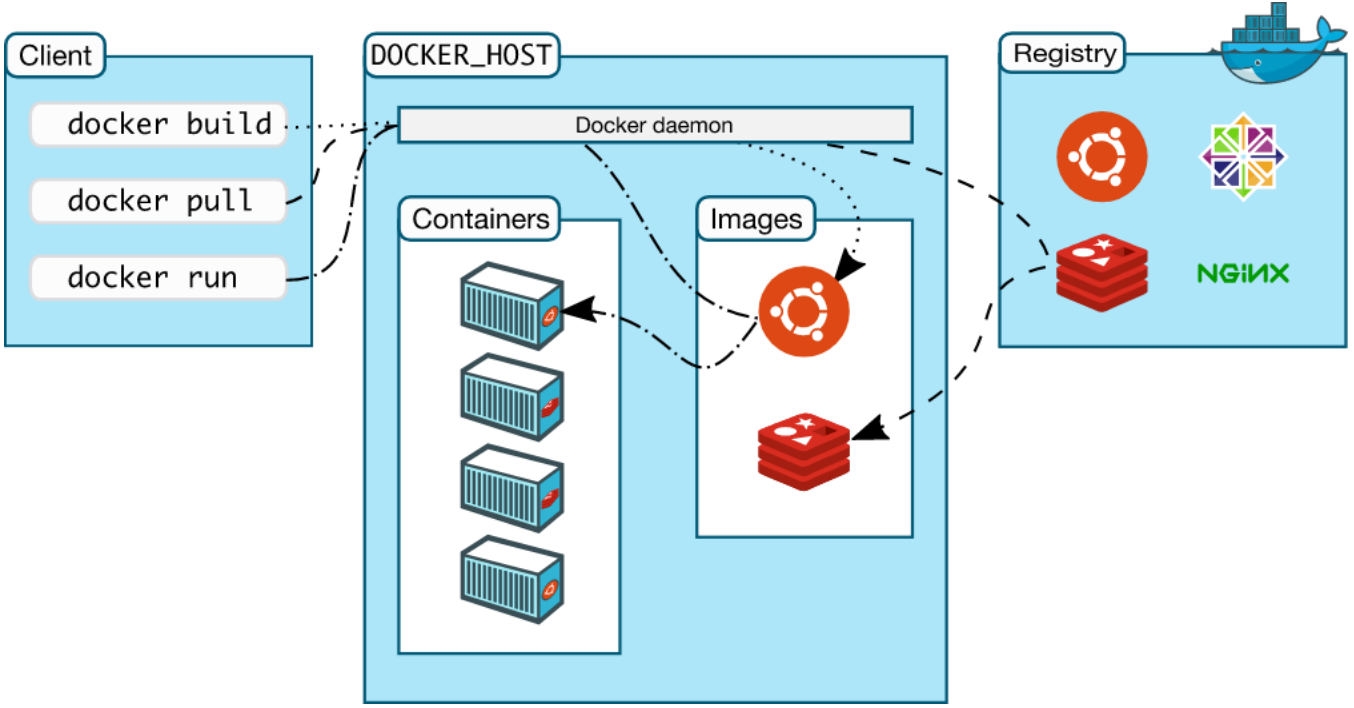
## Monitor

지표 수집 & 시각화
Prometheus + Grafana + Loki

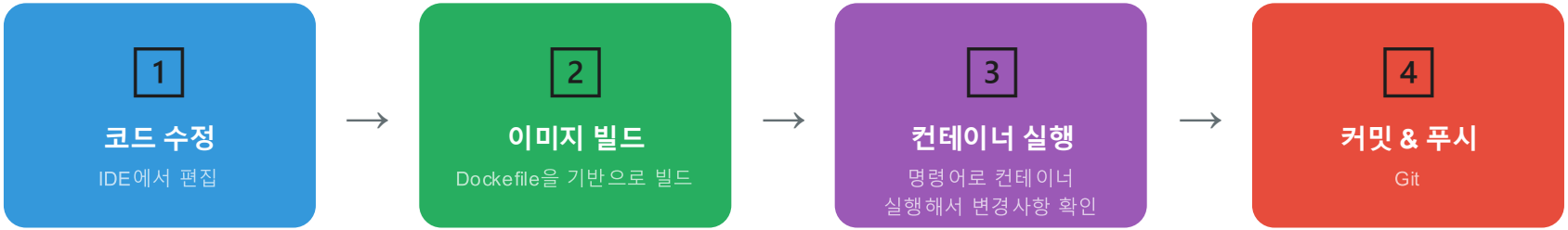
**코드 변경부터 배포, 운영까지**

## KMAP에 적용한다면?

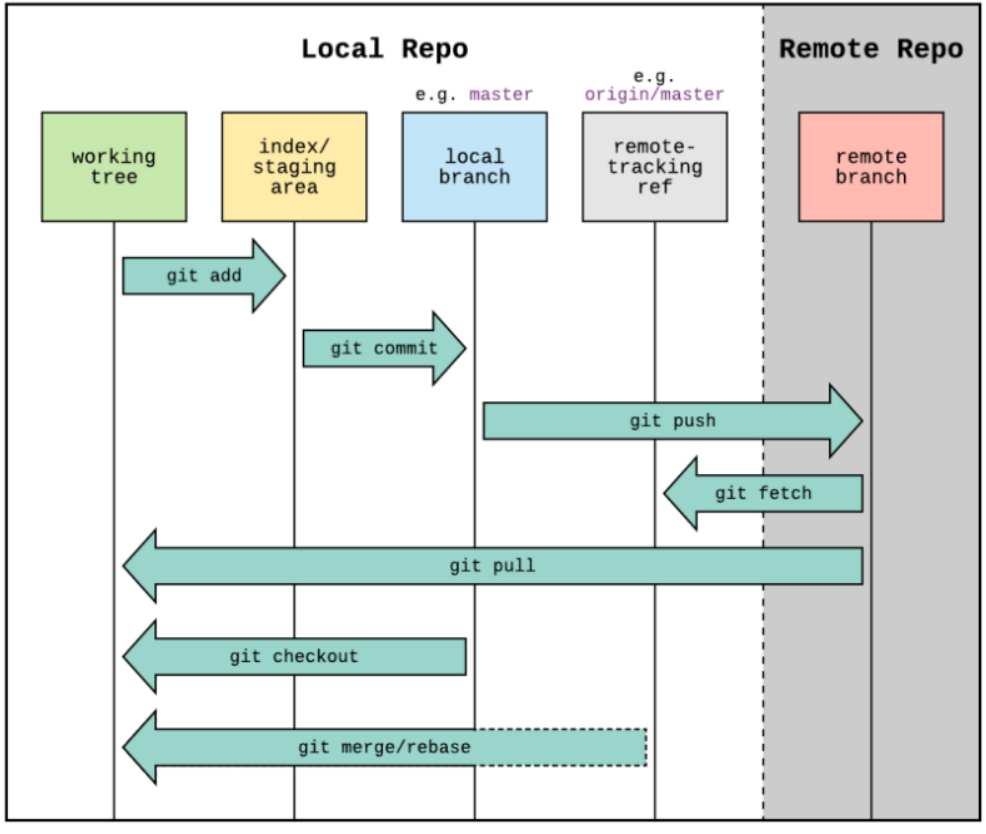
| | |
|---|---|
| **Frontend (React)** | CI: PR 시 lint + Playwright E2E 테스트 자동 실행<br>CD: main 병합 시 Docker 빌드 → Nginx 배포<br>Monitor: 페이지 로드 시간, JS 에러율, 사용자 트래픽 |
| **Backend (FastAPI)** | CI: PR 시 pytest 자동 실행 + 타입 체크<br>CD: main 병합 시 Docker 빌드 → 서버 배포<br>Monitor: API 응답 시간, 5xx 에러율, RPS |
| **Database (PostgreSQL)** | CI: 마이그레이션 스크립트 검증<br>CD: 스키마 변경 자동 적용<br>Monitor: 커넥션 풀, Slow Query, 디스크 사용량 |


Client
docker build
docker pull
docker run
DOCKER_HOST
Docker daemon
Containers
Images
Registry
NGINX

# Docker Compose 개발 시나리오




💡 Docker 이미지는 수정 불가능하기 때문에 **무조건 빌드해서 새로 생성**해야 컨테이너에 코드 변경사항 적용됨


Local Repo
Remote Repo
e.g. master
e.g. origin/master
working tree
index/ staging area
local branch
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge/rebase

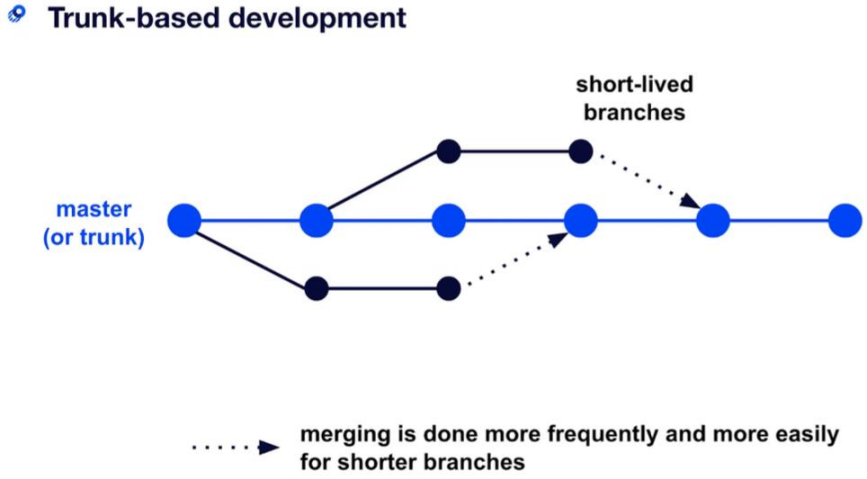
# Trunk-Based Development (TBD)

### 핵심 개념

단일 브랜치(trunk/main)에 모든 개발자가 직접 커밋하거나,
아주 짧은 수명의 브랜치만 사용

### 주요 특징

- 브랜치 수명: **최대 2-3일**
- 작은 단위의 빈번한 통합
- main 머지 = 즉시 배포
- CI/CD 필수



# AI 시대의 코드 리뷰

### AI가 대신할 수 있는 것

- 문법/스타일 검사
- 일반적인 버그 패턴 탐지
- 엣지 케이스와 예외 처리

### 사람이 집중해야 할 것

- 비즈니스 로직의 정확성
- 아키텍처와 설계 결정
- 팀 컨텍스트 기반 판단

src/components/layout/useObserveSize.ts

```
22  +       observer.disconnect();
23  +     };
24  +   }, [observer, options, target]);
25  + }
```

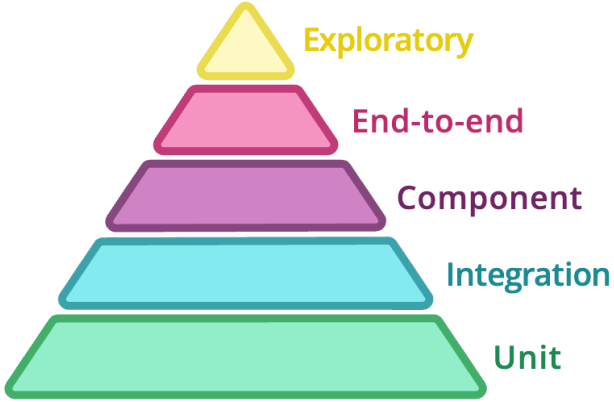
**sanchez-moon** 2 weeks ago — Author

코드 리뷰를 진행하겠습니다.

1. **메모리 누수 위험**: `useMemo`를 사용하여 `observer`를 생성하고 있지만, 컴포넌트가 언마운트될 때 `observer.disconnect()`를 호출하여 리소스를 해제하고 있습니다. 하지만, `useMemo`는 컴포넌트 업데이트 시 재사용될 수 있으므로, `observer`가 더 이상 필요하지 않을 때 안전하게 정리될 수 있도록 조건을 명확히 해야 합니다.

2. **ResizeObserver의 대표적인 부작용**: `ResizeObserver`는 비동기적으로 크기 변화를 감지하므로, 콜백 함수가 불필요하게 여러 번 호출될 수 있습니다. 불필요한 렌더링을 방지하기 위해, 콜백 함수를 debounce하거나 throttle하는 방법을 고려해볼 수 있습니다.

# 테스트 피라미드




**위로 갈수록**

신뢰도 높음 / 유지비용 높음 / 속도 느림

**아래로 갈수록**

비용 낮음 / 속도 빠름 / 작성 용이

> **효율적인 전략: 하위 단계에 집중, E2E는 핵심만**

# Playwright 소개

Playwright

Microsoft Open Source

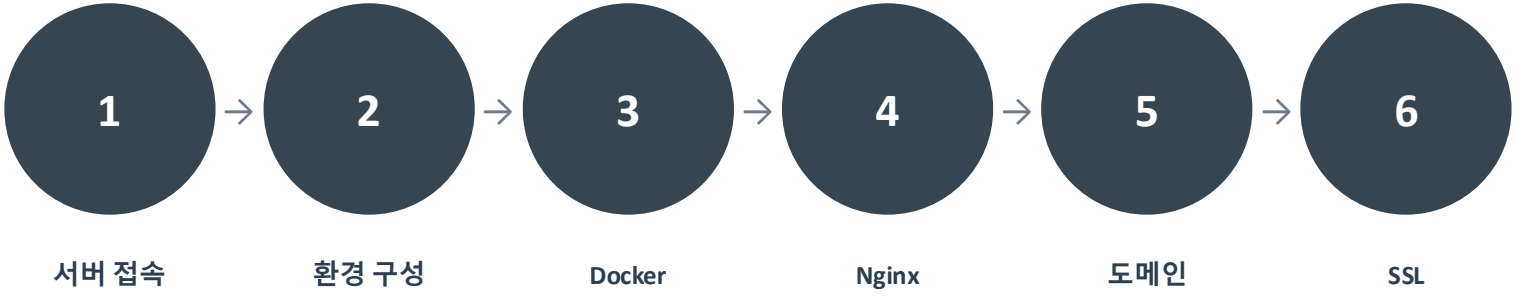
**브라우저 자동화 프레임워크**

웹 테스트 및 자동화를 위한 오픈소스 라이브러리

| Chromium | Firefox | WebKit |
| --- | --- | --- |

Windows, Linux, macOS + 모바일 환경 지원

# 배포 프로세스




'취업 치트키' 컴공과의 배신...이제 '문송' 아니고 '컴송합니다' (자막뉴스) / SBS

조회수 4만회 • 3일 전

SBS 뉴스

한때 취업 시장에서 높은 취업률을 자랑하던 컴퓨터공학부 졸업생들의 취업률이 하락한 걸로 나타났습니다. 대학들이 공개한 '2025년 ...

새 동영상


2026년 신입 개발자 취업이 헬게이트인 이유

조회수 1.4만회 • 2주 전

CODER X DOX 코더엑스독스

슈퍼코더 되는 법: https://www.youtube.com/@coderxdox/join 기술 블로그: https://coderxdox.com [00:00] 인트로 - AI가 개발자

챕터 7 인트로 - AI가 개발자를 대체할까? | AI가 개발자 시장을 양극화시키고 있다 | 시스템 아키텍트는 경험이 필요한 고급

# 1. 외부 행사 참여

README

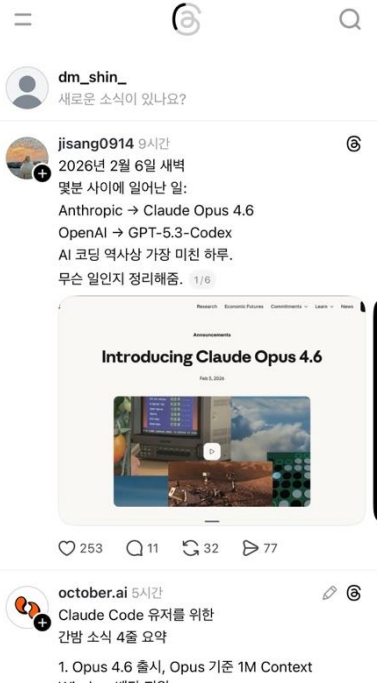
all contributors 192

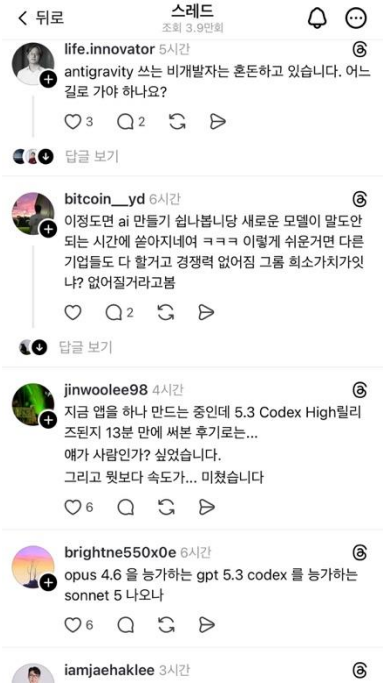

문의: covenant.ko@kakao.com

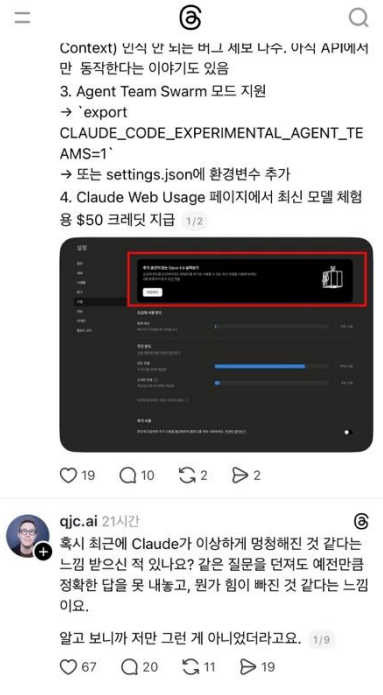
Web Android iOS 인스타그램 Email 구독 명예의 전당

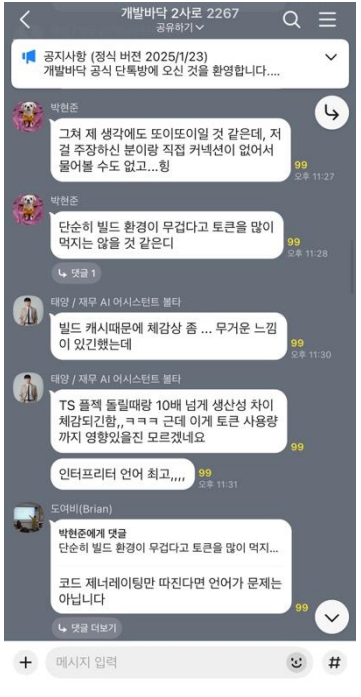
배너 클릭시 연관 링크로 이동합니다.

# 2. 커뮤니티 적극 활용


dm_shin_
새로운 소식이 있나요?
jisang0914 9시간
2026년 2월 6일 새벽
몇분 사이에 일어난 일:
Anthropic → Claude Opus 4.6
OpenAI → GPT-5.3-Codex
AI 코딩 역사상 가장 미친 하루.
무슨 일인지 정리해줌. 1/6
Introducing Claude Opus 4.6
253 11 32 77
october.ai 5시간
Claude Code 유저를 위한
간밤 소식 4줄 요약
1. Opus 4.6 출시, Opus 기준 1M Context


뒤로
스레드
조회 3.9만회
life.innovator 5시간
antigravity 쓰는 비개발자는 혼돈하고 있습니다. 어느 길로 가야 하나요?
3 2
답글 보기
bitcoin__yd 6시간
이정도면 ai 만들기 쉽나봅니당 새로운 모델이 말도안되는 시간에 쏟아지네여 ㅋㅋㅋ 이렇게 쉬운거면 다른 기업들도 다 할거고 경쟁력 없어짐 그롬 회소가치가잇냐? 없어질거라고봄
2
답글 보기
jinwoolee98 4시간
지금 앱을 하나 만드는 중인데 5.3 Codex High릴리즈된지 13분 만에 써본 후기로는...
얘가 사람인가? 싶었습니다.
그리고 뒷보다 속도가... 미쳤습니다
6
brightne550x0e 6시간
opus 4.6 을 능가하는 gpt 5.3 codex 를 능가하는 sonnet 5 나오나
6
iamjaehaklee 3시간


Context) 인식 안 되는 버그 제보 나수. 아직 API에서만 동작한다는 이야기도 있음
3. Agent Team Swarm 모드 지원
→ `export CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1`
→ 또는 settings.json에 환경변수 추가
4. Claude Web Usage 페이지에서 최신 모델 체험용 $50 크레딧 지급 1/2
19 10 2 2
qjc.ai 21시간
혹시 최근에 Claude가 이상하게 멍청해진 것 같다는 느낌 받으신 적 있나요? 같은 질문을 던져도 예전만큼 정확한 답을 못 내놓고, 뭔가 힘이 빠진 것 같다는 느낌이요.
알고 보니까 저만 그런 게 아니었더라고요. 1/9
67 20 11 19


개발바닥 2사로 2267
공유하기
공지사항 (정식 버전 2025/1/23)
개발바닥 공식 단톡방에 오신 것을 환영합니다....
박현준
그쳐 제 생각에도 또이또이일 것 같은데, 저걸 주장하신 분이랑 직접 커넥션이 없어서 물어볼 수도 없고...힝
99
오후 11:27
박현준
단순히 빌드 환경이 무겁다고 토큰을 많이 먹지는 않을 것 같은디
99
오후 11:28
댓글 1
태양 / 재무 AI 어시스턴트 볼타
빌드 캐시때문에 체감상 좀 ... 무거운 느낌이 있긴했는데
99
오후 11:30
태양 / 재무 AI 어시스턴트 볼타
TS 폴젝 돌릴때랑 10배 넘게 생산성 차이 체감되긴함,,ㅋㅋㅋ 근데 이게 토큰 사용량까지 영향있을진 모르겠네요
99
인터프리터 언어 최고,,,,
99
오후 11:31
도여비(Brian)
박현준에게 댓글
단순히 빌드 환경이 무겁다고 토큰을 많이 먹지...
코드 제너레이팅만 따진다면 언어가 문제는 아닙니다
99
댓글 더보기
메시지 입력

# 3. 채용 공고 확인


풀스택 개발자
케어랩스
서울 강남구 · 경력 6년 이상


Site Reliability Engineer (플랫폼 신뢰성 및 자동화 엔지니어)
미소(miso)
서울 종로구 · 경력 5-15년

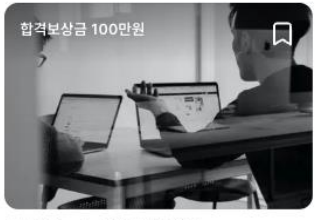

MLOps Engineer
씨이랩
서울 강남구 · 경력 2-8년


Product Engineer (Full Stack)
씨이랩
서울 강남구 · 경력 2-8년


클라우드 인프라/서버/네트워크 3년 이상
이지케어텍
서울 중구 · 경력 3-20년


Backend Engineer
브레이브모바일(숨고,Soomgo)
서울 강남구 · 경력 3-10년


SaaS 플랫폼 풀스택 개발자 (Python/React)
마크스폰
서울 서초구 · 경력 3년 이상




Full Stack 풀스택 개발자
컨워스
서울 서대문구 · 경력 7년 이상


AI/ML 개발자 3년 이상
서치독
경기 성남시 · 경력 3-10년


신용평가모형 개발(2년 이상)
테크핀레이팅스
서울 중구 · 경력 2-15년

# 3. 채용 공고 확인

**이런 분과 함께하고 싶어요**

- React, Vue, Angular 등 SPA 프레임워크 사용에 능숙하신 분이면 좋아요.
- 단순히 주어진 개발을 해내는 것보다, 주도적으로 문제를 발견하고 분석해 솔루션을 제안할 수 있는 분이 필요해요.
- TypeScript, Flow를 이용한 JavaScript 정적 타입 분석을 경험해보신 분이면 좋아요.

**이력서는 이렇게 작성하시는 걸 추천해요**

- 그동안의 경험을 단순 나열하는 것이 아닌, 경험 속에서의 임팩트 및 러닝 포인트를 기술해주세요.
- 고객의 보이스를 기반으로 빠르게 제품의 완성도를 높여가기 때문에, 주어진 문제를 스스로 해결해보려고 시도하는지 보고 있어요.
- 서버 사이드 렌더링(SSR) 및 모바일 앱 내 웹앱 개발 경험이 있으면 기술해주세요.
- 기존 소스 코드를 새로운 코드 베이스로 점진적으로 이관한 경험이 있으면 기술해주세요.

**토스가 사용하는 기술**

- 코어: React, TypeScript, Next.js
- 상태 관리: TanStack Query, Jotai
- 스타일링: Emotion
- 패키지 매니저: Yarn Berry, PNPM
- 빌드: Vite, ESBuild, SWC
- CI/CD: GitHub Actions, CircleCI

# 3. 채용 공고 확인

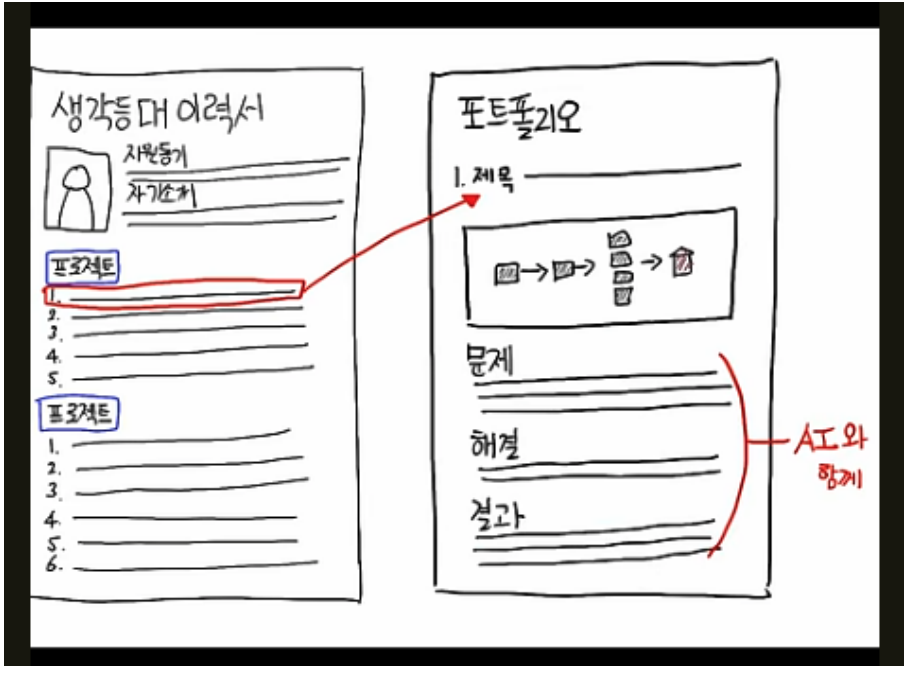
**이런 분과 함께하고 싶어요**

- 고가용성의 확장 가능한 시스템을 설계하고 운영해본 경험이 있는 분이 필요해요.
- 대규모의 실시간 트래픽을 처리하는 시스템 개발 경험이 있는 분이 필요해요.
- 장애를 경험하고 문제를 해결해보신 경험이 있는 분이 필요해요.
- 서비스에 대한 애착이 강해서 '내 서비스'라는 마음으로 일하는 분이면 좋아요.
- 서비스 개발을 하면서 얻게 되는 새로운 인사이트를 공유하며, 끊임없이 기술적인 도전을 하고 싶은 분과 함께하고 싶어요.

**토스에서 사용하는 기술**

- Kotlin, Spring, MySQL, MongoDB, Hadoop, Redis, Kafka

# 4. 문제 해결 경험 기록


생각등대 이력서
지원동기
자기소개
프로젝트
1.
2.
3.
4.
5.
프로젝트
1.
2.
3.
4.
5.
6.
포트폴리오
1. 제목
문제
해결
결과
AI와
함께

## 4. 문제 해결 경험 기록

문제 : 장기 운영 중 메모리 누수로 인한 서버 중단 장애 발생
접근 : SQLAlchemy Engine/DB 연결 미해제가 원인임을 분석하고
해결 : Context Manager 패턴 적용으로 일일 450MB 메모리 누수 해결

문제 : 장시간 작업의 실시간 모니터링 제공을 위해
접근 : SSE 기반 작업 상태 스트리밍 구현, 1시간 타임아웃 및 연결 끊김 시 자동 리소스 정리로
해결 : 장시간 연결에서의 안정성 확보

# Q&A

**마무리 과제 안내**

📝 과제:
- KMAP 서버 접속 해보기
- 회고 블로그 글 작성하기