



# DevOps CI/CD on GKE

# Table of Contents

## 1... [What is DevOps?](#)

1.1... DevOps Capabilities

## 2... [What is CI/CD?](#)

## 3... [Diagram](#)

## 4... [Used Tech](#)

4.1... AKS vs EKS vs GKE

4.2... Jenkins vs Gitlab

4.3... What is ArgoCD?

4.4... What is FastAPI?

## 5... [Create FastAPI File](#)

5.1... Install FastAPI

5.2... Code

5.3... Start Server

## 6... [Gitlab Pipeline](#)

6.1... Create Repository

6.2... Create .gitlab-ci.yml

6.3... Edit Repository

6.4... Add Predefined Variables

6.5... Check CI/CD

## 7... [Create Cluster](#)

7.1... Create Cluster

## 8... [Settings Kubernetes](#)

8.1... Settings Kubernetes

8.2... Install ArgoCD

8.3... Patch yaml ArgoCD Service

8.4... Get ArgoCD Admin PWD

8.5... Login ArgoCD

## 9... [Settings ArgoCD](#)

9.1... Create Manifest

9.2... Create APP

9.3... Check Sync

## 10... [Monitoring GKE](#)

10.1... Default Dashboard

10.2... Custom Dashboard

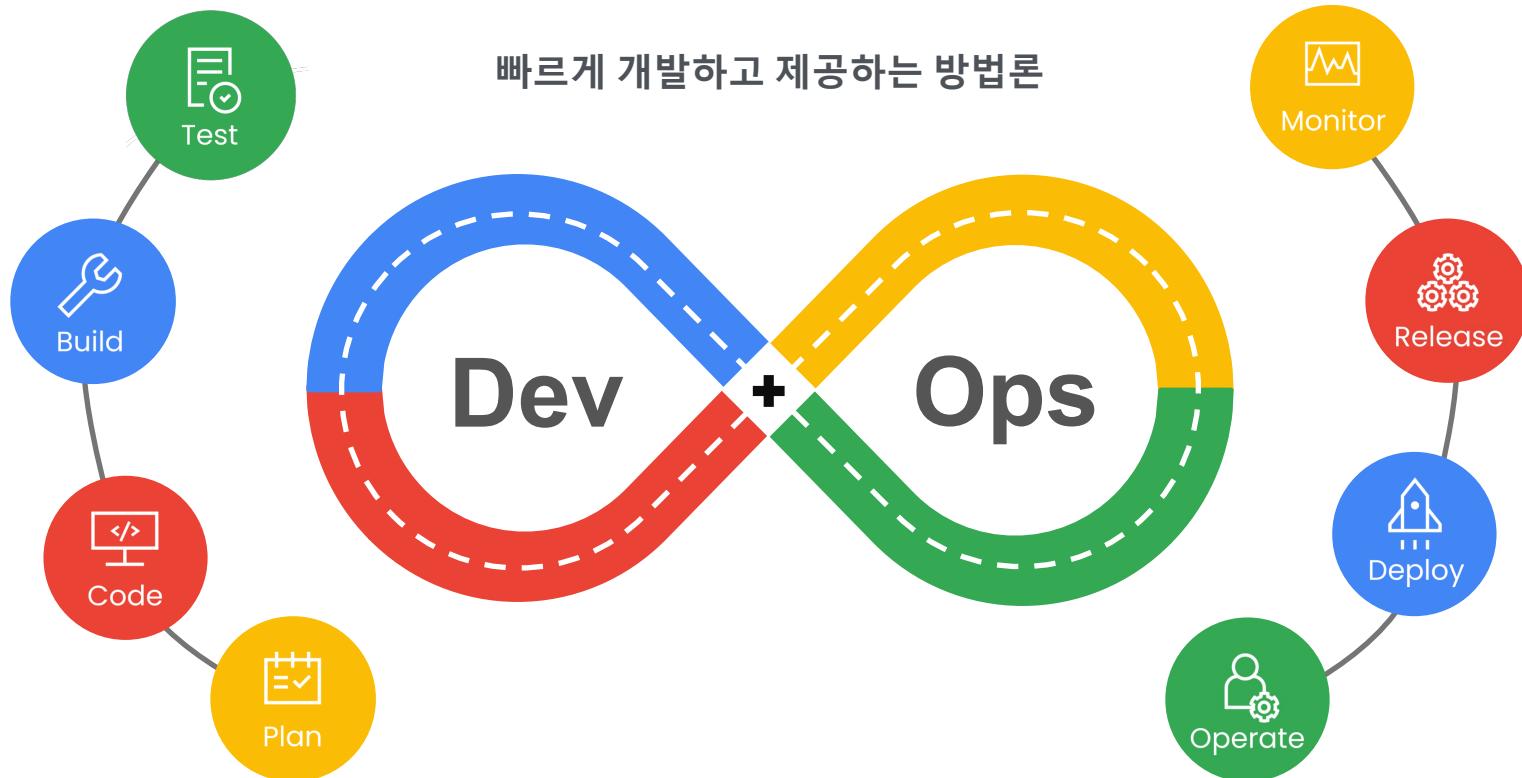
10.3... Metrics Dashboard



# What is DevOps?

개발자와 운영자가 함께 새 소프트웨어 기능 및 서비스를

빠르게 개발하고 제공하는 방법론





# DevOps Capabilities

DevOps 연구 결과 및 평가 ([DORA](#))

## 기술적 역량 기능

- 버전 제어
- 지속적 통합
- 배포 자동화
- 코드 유지 관리성

## 프로세스 기능

- 소규모 배치 작업
- 변경 승인 간소화

## 측정 기능

- 모니터링
- 오류 사전 알림

## 문화적 기능

- 직무 만족도
- 배포 학습



# What is CI/CD?



CI

지속적인 통합 (Continuous Integration) 입니다.

애플리케이션의 새로운 코드의 변경 사항이 정기적으로 빌드 및 테스트 되어 레파지토리에 통합하는 것을 의미합니다.

여러 개발자가 동시의 작업하며 발생할 수 있는 쟁돌 문제를 수시로 확인하고 해결할 수 있습니다. 버그를 빠르게 찾아 해결하고, 소프트웨어 품질을 개선하며 새로운 업데이트와 출시 시간을 단축 시키는 것 입니다.

CD

지속적인 제공(Continuous Delivery) 또는 지속적인 배포(Continuous Deployment) 두 가지 의미로 같이 사용됩니다.

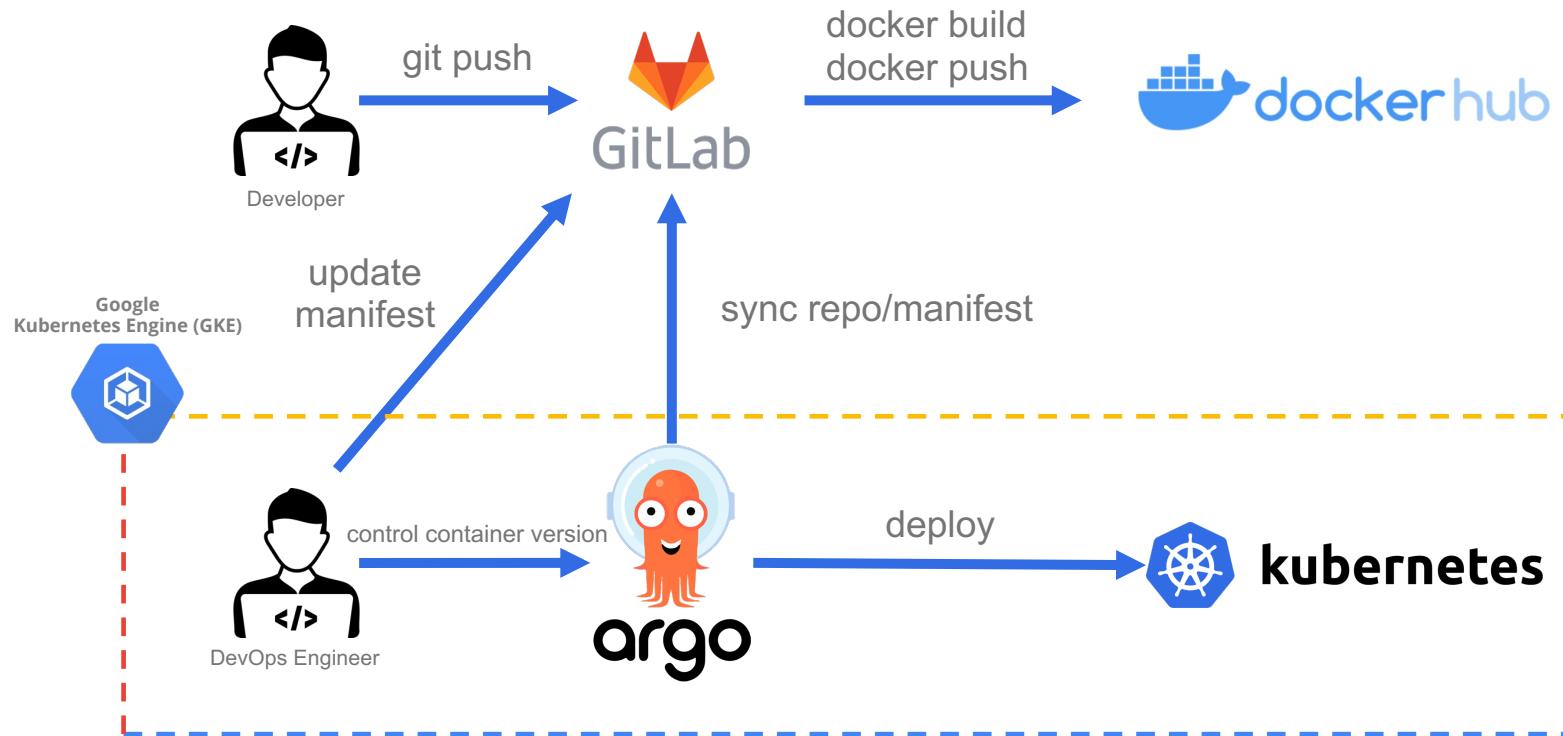
지속적인 제공이란 개발자들이 애플리케이션에 적용한 변경 사항이 버그 테스트를 거쳐 레파지토리에 자동으로 업로드되는 것을 뜻하며, 운영팀은 이 레파지토리에서 애플리케이션을 실시간으로 배포할 수 있습니다.

지속적인 배포란 개발자의 변경 사항을 레파지토리에서 실제 서비스까지 자동으로 릴리스하는 것을 의미합니다. 이는 애플리케이션 제공 속도를 저해하는 수동 프로세스로 인한 운영팀의 프로세스 과부화 문제를 해결할 수 있습니다.



# Diagram

# Diagram





# Used Tech



# Used Tech

Google  
Kubernetes Engine (GKE)



GitLab



argo



docker



FastAPI



kubernetes

# AKS vs EKS vs GKE

## KUBERNETES

	Azure (AKS)	AWS (EKS)	GCP (GKE)
Released	2018	2018	2015
Gov Cloud	Yes	Yes	No
Upgrades	Automated	Some manual steps	Automated
Cluster Nodes	VMs and GPUs	VMs, GPUs, Bare Metal	VMs and GPUs
Command Line	Supported	Partial support	Supported
Service Mesh	Not yet	App Mesh	Istio
Nodes	500	100	5000

구글 GCP가 쿠버네티스를 가장 오래 서비스 했으며,  
쿠버네티스가 안정적으로 지원할 것으로 생각되었으며,  
결정적으로 타 클라우드 플랫폼보다 UI가 상당히 보기 편했습니다.

# Jenkins vs Gitlab



## Jenkins

Java 기반 오픈소스로 구성되어있고, **가장 대중적으로 사용되고** 있으며, 다양한 플러그인을 지원하고 문서화가 잘 되어있습니다. 호스팅을 직접 하기 때문에 CI/CD 프로세스를 직접 관리할 수 있지만, **호스팅 비용이** 발생하게 됩니다. 그리고 플러그인이 많기도 하고, 간혹 플러그인 간에 의존성 문제도 발생하게 됩니다.



## GitLab

Gitlab에는 CI/CD 파이프라인을 빠르게 설정하는 데 도움이 되는 CI/CD 기능이 내장되어 있습니다. Auto DevOps는 소프트웨어 제공을 쉽고 효율적이며 자동화하는 Gitlab의 또 다른 강력한 기능입니다. **코드 언어를 스캔하고 작업을 자동으로 빌드, 테스트 및 배포하는** 사전 정의된 기본 CI/CD 템플릿을 사용합니다. Gitlab은 **400분/월** 무료로 이용 가능합니다

# What is ArgoCD?

*Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.*



쿠버네티스를 위한 CD(Continuous Delivery)툴입니다.  
GitOps방식으로 관리되는 **Manifest** 파일의 변경사항을  
감시하며, 현재 배포된 환경의 상태와 Git에 정의된  
**Manifest** 상태를 동일하게 유지하는 역할을 수행 합니다.

# What is FastAPI?

*FastAPI framework, high performance, easy to learn, fast to code, ready for production*



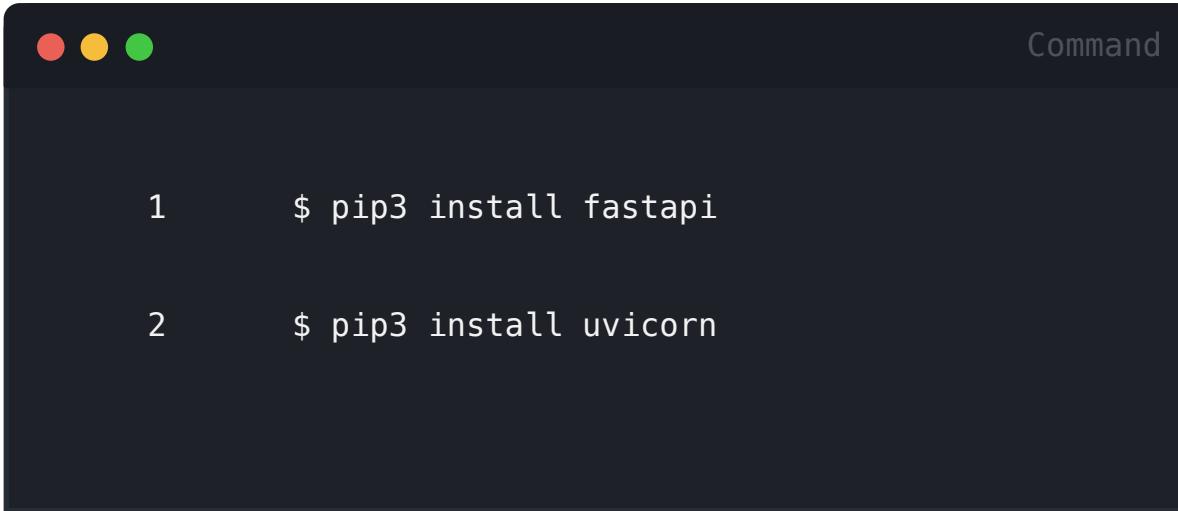
파이썬 3.6 부터 제공되는 파이썬 프레임워크입니다.  
요즘 트렌드는 마이크로서비스가 트렌드입니다. 그러기  
위해선 많은 데이터를 처리해야합니다. FastAPI는  
기본적으로 비동기를 지원하며, 결론적으로 빠르게 만들  
수 있기 때문입니다.



# Create FastAPI File



# Install FastAPI



```
1 $ pip3 install fastapi
2 $ pip3 install uvicorn
```



# Code



```
main.py

from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"Hello": "CI/CD"} # 출력 화면
```



# Start Server



```
main.py
$ uvicorn --host=0.0.0.0 --port 8000 main:app
```



{"HELLO": "CI/CD"}



# Gitlab Pipeline

# Create Repository

The screenshot shows the GitLab interface for creating a new project. On the left, a sidebar menu is visible with options like 'Your work', 'Projects' (which is selected and highlighted in grey), 'Groups', 'Issues', 'Merge requests', 'To-Do List', 'Milestones', 'Snippets', and 'Activity'. The main content area is titled 'Create new project' and contains the following sections:

- Create blank project**:  
GitLab에서 새로운 레파지토리를 생성합니다.  
본인의 소스코드를 PUSH 합니다.  
Create a blank project to store your files, plan your work, and collaborate on code, among other things.
- Create from template**:  
Create a project pre-populated with the necessary files to get you started quickly.
- Import project**:  
Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.
- Run CI/CD for external repository**:  
Connect your external repository to GitLab CI/CD.

At the bottom, a note says: You can also create a project from the command line. [Show command](#)

# Create Repository

## CI/CD templates

Get started with GitLab CI/CD and your favorite programming language or framework by using a `.gitlab-ci.yml` template.

When you create a `.gitlab-ci.yml` file in the UI, you can choose one of these templates:

- [Android \(Android.gitlab-ci.yml\)](#)
- [Android with fastlane \(Android-Fastlane.gitlab-ci.yml\)](#)
- [Bash \(Bash.gitlab-ci.yml\)](#)
- [C++ \(C++.gitlab-ci.yml\)](#)
- [Chef \(Chef.gitlab-ci.yml\)](#)
- [Clojure \(Clojure.gitlab-ci.yml\)](#)
- [Composer Composer.gitlab-ci.yml](#)
- [Crystal \(Crystal.gitlab-ci.yml\)](#)
- [Dart \(Dart.gitlab-ci.yml\)](#)
- [Django \(Django.gitlab-ci.yml\)](#)
- [Docker \(Docker.gitlab-ci.yml\)](#)
- [dotNET \(dotNET.gitlab-ci.yml\)](#)
- [dotNET Core \(dotNET-Core.gitlab-ci.yml\)](#)
- [Elixir \(Elixir.gitlab-ci.yml\)](#)
- [Flutter \(Flutter.gitlab-ci.yml\)](#)
- [Golang \(Go.gitlab-ci.yml\)](#)
- [Gradle \(Gradle.gitlab-ci.yml\)](#)
- [Grails \(Grails.gitlab-ci.yml\)](#)
- [iOS with fastlane \(iOS-Fastlane.gitlab-ci.yml\)](#)
- [Julia \(Julia.gitlab-ci.yml\)](#)
- [Laravel \(Laravel.gitlab-ci.yml\)](#)
- [LaTeX \(LaTeX.gitlab-ci.yml\)](#)
- [Maven \(Maven.gitlab-ci.yml\)](#)
- [Mono \(Mono.gitlab-ci.yml\)](#)
- [npm \(npm.gitlab-ci.yml\)](#)
- [Node.js \(Nodejs.gitlab-ci.yml\)](#)
- [OpenShift \(OpenShift.gitlab-ci.yml\)](#)
- [Packer \(Packer.gitlab-ci.yml\)](#)
- [PHP \(PHP.gitlab-ci.yml\)](#)
- [Python \(Python.gitlab-ci.yml\)](#)
- [Ruby \(Ruby.gitlab-ci.yml\)](#)
- [Rust \(Rust.gitlab-ci.yml\)](#)
- [Scala \(Scala.gitlab-ci.yml\)](#)
- [Swift \(Swift.gitlab-ci.yml\)](#)
- [Terraform \(Terraform.gitlab-ci.yml\)](#)
- [Terraform \(Terraform.latest.gitlab-ci.yml\)](#)

Gitlab에서 CI/CD의 기능을 사용하기 위해서  
“`.gitlab-ci.yml`”이라는 파일에서 JOB을 코딩하여  
CI/CD기능을 이용할 수 있습니다.

# Create .gitlab-ci.yml

```
docker-build:  
  # Use the official docker image.  
  image: docker:latest  
  stage: build  
  services:  
    - docker:dind  
  before_script:  
    - echo "$CI_REGISTRY_PASSWORD" | docker login --username "$CI_REGISTRY_USER" --password-stdin  
  script:  
    - |  
        if [[ "$CI_COMMIT_BRANCH" == "$CI_DEFAULT_BRANCH" ]]; then  
          tag=""  
          echo "Running on default branch '$CI_DEFAULT_BRANCH': tag = 'latest'"  
        else  
          tag=":$CI_COMMIT_REF_SLUG"  
          echo "Running on branch '$CI_COMMIT_BRANCH': tag = $tag"  
        fi  
    - docker build --pull -t "$CI_REGISTRY_IMAGE${tag}" .  
    - docker push "$CI_REGISTRY_IMAGE${tag}"  
  # Run this job in a branch where a Dockerfile exists  
  rules:  
    - if: $CI_COMMIT_BRANCH exists:  
      - Dockerfile
```

기본 docker 템플릿입니다.  
미리 정의된 변수들을 활용하여 코딩을 할 수 있습니다.  
다른 변수들은 [여기서](#) 추가로 볼 수 있습니다

그냥 쓰면 docker 이미지버전이 latest가 형상관리하기 어렵기 때문에 버전을 바꿔줄 필요가 있다고 생각했습니다.

# Edit .gitlab-ci.yml



.gitlab-ci.yml

```
docker-build:  
  # Use the official docker image.  
  image: docker:latest  
  stage: build  
  services:  
    - docker:dind  
  before_script:  
    - echo "$CI_REGISTRY_PASSWORD" | docker login --username "$CI_REGISTRY_USER" --  
      password-stdin  
  script:  
    - |  
        if [[ "$CI_COMMIT_BRANCH" == "$CI_DEFAULT_BRANCH" ]]; then  
          tag=""  
          echo "Running on default branch '$CI_DEFAULT_BRANCH': tag = 'latest'"  
        else  
          tag=":$CI_COMMIT_REF_SLUG"  
          echo "Running on branch '$CI_COMMIT_BRANCH': tag = $tag"  
        fi  
    - docker build --pull -t "$CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA" .  
    - docker push "$CI_REGISTRY_IMAGE:$CI_COMMIT_SHORT_SHA"
```

# Run this job in a branch where **CI\_COMMIT\_SHORT\_SHA**는 commit했을 때 commit hash의 앞 8자리로 이미지를 만들게 코딩했습니다.  
rules:  
- if: \$CI\_COMMIT\_BRANCH exists:  
- Dockerfile

Docker 이미지버전을 commit hash로 하면 어떤 업데이트가 이루어 질는지  
형상관리하기 편할 것이라고 생각했습니다.

# Add Predefined Variables Reference

## Variables

Collapse

Variables store information, like passwords and secret keys, that you can use in job scripts. Each project can define a maximum of 8000 variables. [Learn more.](#)

Variables can have several attributes. [Learn more.](#)

- Protected: Only exposed to protected branches or protected tags.
- Masked: Hidden in job logs. Must match masking requirements.
- Expanded: Variables with \$ will be treated as the start of a reference to another variable.

Environment variables are configured by your administrator to be [protected](#) by default.

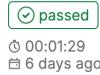
Type	↑ Key	Value	Options	Environments		
Docker hub 프로필 주소	Variable	CI_REGISTRY 	***** 	Expanded	All (default) 	
Docker 이미지 이름	Variable	CI_REGISTRY_IMAGE 	***** 	Expanded	All (default) 	
Docker hub pwd	Variable	CI_REGISTRY_PASSWORD 	***** 	Expanded	All (default) 	
Docker hub id	Variable	CI_REGISTRY_USER 	***** 	Expanded	All (default) 	

Add variable

Reveal values

Repo → Settings → CI/CD

# Check CI/CD



fixed  
[#764942836](#) ⚡ main -o [58a72cdb](#) 🎉  
latest



failed  
⌚ 00:01:20  
⌚ 6 days ago



fixed  
[#763763735](#) ⚡ main -o [1a4133e1](#) 🎉



Commit을 했을 때 파이프라인 내부 빌드 과정을 확인할 수 있으며  
오류가 생겼을 때 어디서 오류가 났는지 확인할 수 있습니다.

```
174 9f4f964da727: Preparing
175 49b333f7bad4: Preparing
176 a463dbda4d44: Preparing
177 a99993c159f5: Preparing
178 91a3056ac3de: Waiting
179 3306a676f708: Waiting
180 dc6442f7bb8b: Waiting
181 a4db1a405763: Waiting
182 9f4f964da727: Waiting
183 49b333f7bad4: Waiting
184 a463dbda4d44: Waiting
185 a99993c159f5: Waiting
186 7673d2988975: Layer already exists
187 91a3056ac3de: Layer already exists
188 3306a676f708: Layer already exists
189 dc6442f7bb8b: Layer already exists
190 d9f6f5eaee8b: Pushed
191 a4db1a405763: Layer already exists
192 9f4f964da727: Layer already exists
193 49b333f7bad4: Layer already exists
194 8236387c974: Pushed
195 a463dbda4d44: Layer already exists
196 a99993c159f5: Layer already exists
197 98fd34adafb: Pushed
198 ffe15fd6dc4c: Pushed
199 58a72cdb: digest: sha256:5bd0cf28679ff157b796e9843482842bb4efa859ea1294339e62d124f87d2c74 size: 3061
200 Cleaning up project directory and file based variables
201 Job succeeded
```

```
1 Running with gitlab-runner 15.6.0-beta.186.gae889181a (a889181a)
2 on blue-4-shared.runners-manager.gitlab.com/default J2nyww-s
3 Preparing the "docker-machine" executor
4 Using Docker executor with image docker:latest ...
5 Starting service docker:dind ...
6 Putting docker image sha256:13e76dab1a191647d74eefad689b7a05d564f06cc8ca108bff31a7d439511f0f02 for docker:dind with digest docker@sha256:1ddc6956bddc29c4eb3f3ff9e9a82843093a8227e353a60bf72e10c485f1fe01a89e ...
7 Using docker image sha256:a028932855457fd6942dadbd316a5303233c88ffd39ff19f960b9d38110 for docker:latest with digest docker@sha256:919b061b7f4ec6cc50f480a7ee93c1f902012fa8ae0e1d1601ab6 ...
8 Waiting for services to be up and running (timeout 30 seconds)...
9 Pulling docker image docker:latest ...
10 Using docker image sha256:a028932855457fd6942dadbd316a5303233c88ffd39ff19f960b9d38110 for docker:latest with digest docker@sha256:919b061b7f4ec6cc50f480a7ee93c1f902012fa8ae0e1d1601ab6 ...
11 Preparing environment
12 Running runner J2nyww-s-project-42945634-concurrent-0 via runner-j2nyww-s-shared-1675065854-56bfad77...
13 Getting source from Git repository
14 $ eval "$CI_PRE_CLONE_SCRIPT"
15 Fetching changes with git depth set to 20...
16 Initialized empty Git repository in /builds/handgym2/gke-cicd/.git/
17 Created fresh repository.
18 Checking out ecd05e79 as main...
19 Skipping Git submodules setup
20 Executing "step_script" stage of the job script
21 Using docker image sha256:a028932855457fd6942dadbd316a5303233c88ffd39ff19f960b9d38110 for docker:latest with digest docker@sha256:919b061b7f4ec6cc50f480a7ee93c1f902012fa8ae0e1d1601ab6 ...
22 $ echo "$CI_REGISTRY_PASSWORD" | docker login --username "$CI_REGISTRY_USER" --password-stdin
23 Error response from daemon: Get "https://registry-1.docker.io/v2/": unauthorized: incorrect username or password
24 Cleaning up project directory and file based variables
25 ERROR: Job failed: exit code 1
```

Repo → CI/CD → Pipelines

# Check CI/CD

 fixed  
jimin son authored 6 days ago  58a72cdb  

Commit을 했을 때 commit hash 값과  
이미지버전이 같은 모습을 확인 할 수 있습니다.

 jeff125 / kube-fastapi

## Description

This repository does not have a description 

 Last pushed: 6 days ago

## Tags

 VULNERABILITY SCANNING - DISABLED  
[Enable](#)

This repository contains 9 tag(s).

Tag	OS	Type	Pulled	Pushed
 58a72cdb		Image	---	6 days ago



# Create Cluster

# Create Cluster

The screenshot shows the Google Cloud Platform interface for creating a Kubernetes cluster. The top navigation bar includes the Google Cloud logo, a project dropdown set to "fastapi CICD", a search bar, and a title "Standard로 클러스터를 제작합니다." (Create cluster using Standard). The main content area displays a message: "Autopilot모드는 클러스터의 리소스가 많아지면 자동으로 늘어나며, 곧 비용문제로 직결되기 때문입니다." (Autopilot mode increases automatically as resources grow, leading to cost issues). A modal window titled "클러스터 만들기" (Create cluster) is open, showing two options: "Autopilot: Google manages your cluster (Recommended)" and "Standard: 사용자가 클러스터 관리". The "Standard" option is highlighted with a red border. Below the modal, there is a note: "클러스터 모드를 비교하여 차이점에 대해 자세히 알아보세요." (Compare cluster modes to learn about differences). The left sidebar lists various Kubernetes Engine features: Clusters, Workload, Services & Mesh, Applications, Secret & ConfigMap, Storage, Container Registry, Migrate to Containers, Backup for GKE, and Monitoring. The bottom navigation bar includes Marketplace, Release Notes, and Help.

# Create Cluster

The screenshot shows the Google Cloud Platform interface for creating a new Kubernetes cluster. The top navigation bar includes the Google Cloud logo, a project selector for 'fastapi CICD', and a search bar with placeholder text '리소스, 문서, 제품 등 검색(/)' and a search button.

The main page title is 'Kubernetes 클러스터 만들기' (Create Cluster). Below it, there are tabs for '노드 풀 추가' (Add Node Pool), '노드 풀 삭제' (Delete Node Pool), and '설정 가이드 사용' (Use Setup Guide).

The left sidebar contains a tree view of cluster components:

- 클러스터 기본사항
  - 노드 풀
    - default-pool
      - 노드
      - 네트워킹
      - 보안
      - 메타데이터
  - 클러스터
    - 자동화
    - 네트워킹
    - 보안
    - 메타데이터
    - 기능

The '클러스터 기본사항' section is expanded, showing configuration details:

- 클러스터 기본사항**: Describes the creation of a new cluster with the name and location.
- 예상 월 비용**: [미리보기](#) (Preview) - US\$158.38, 1시간당 약 US\$0.22.
- 가격은 사용한 리소스, 관리 수수료, 할인, 크레딧을 기준으로 책정됩니다.** ([자세히 알아보기](#))
- 비용 분석 표시**: Shows a preview of monthly costs.

**이름, 영역 외에 견들이지 않았습니다.**

**그 이유는 성능, 노드 개수 증가 등 있어 비용으로 청구됩니다.**

At the bottom, there are buttons for '만들기' (Create), '취소' (Cancel), and '동등한 REST 또는 명령줄' (Equivalent REST or Command Line).



# Settings Kubernetes

# Settings Kubernetes

The screenshot shows the Google Cloud Kubernetes Engine settings interface for a cluster named 'cluster-1'. The left sidebar lists various management options like '작업 부하', '서비스 및 수신', and '보안 베일 및 ConfigMap'. The main panel displays basic cluster information, including its name ('cluster-1'), location ('영역'), and node configuration ('asia-east1-a'). It also shows network details like external IP addresses and subnet ranges. A red box highlights the '연결' (Connect) button at the top right of the main panel.

리소스, 문서, 제품 등 검색(/)

검색

Kubernetes Engine

클러스터

수정

삭제

노드 풀 추가

배포

연결

복제

cluster-1

세부정보

노드

스토리지

관측 가능성

로그

클러스터 기본사항

이름	cluster-1	🔒
위치 유형	영역	🔒
제어 영역의 영역	asia-east1-a	🔒
기본 노드 영역	asia-east1-a	✎
출시 채널	일반 채널	✎ 업그레이드 사용 가능
버전	1.24.8-gke.2000	
전체 크기	3	ⓘ
외부 엔드포인트	35.201.197.55 <a href="#">클러스터 인증서 표시</a>	✎
내부 엔드포인트	10.140.0.2 <a href="#">클러스터 인증서 표시</a>	🔒

자동화

유지보수 기간	시간 무관	✎
유지보수 제외	없음	
알림	사용 중지됨	✎
수직형 pod 자동 확장	사용 중지됨	✎
노드 자동 프로비저닝	사용 중지됨	✎
자동 프로비저닝 네트워크 태그		✎
자동 확장 프로필	균형	✎

네트워킹

내부 네트워크 설정

내부 주소

△

# Settings Kubernetes

The screenshot shows the Google Cloud Platform interface for managing a Kubernetes cluster named 'cluster-1'. The left sidebar lists various management options like '작업 부하' (Workload), '서비스 및 수신' (Services & Receiving), and '보안 비밀 및 ConfigMap' (Security Secrets & ConfigMap). The main panel displays a modal window titled '클러스터에 연결' (Connect to Cluster) with the following content:

**클러스터에 연결**

명령줄을 통하거나 대시보드를 사용하여 클러스터에 연결할 수 있습니다.

**명령줄 액세스**

다음 명령어를 실행하여 [kubectl](#) 명령줄 액세스를 구성합니다.

```
$ gcloud container clusters get-credentials cluster-1 --zone asia-east1-a --project level-slate-375807
```

**CLOUD SHELL에서 실행** (This section is highlighted with a red box.)

**Cloud Console 대시보드**

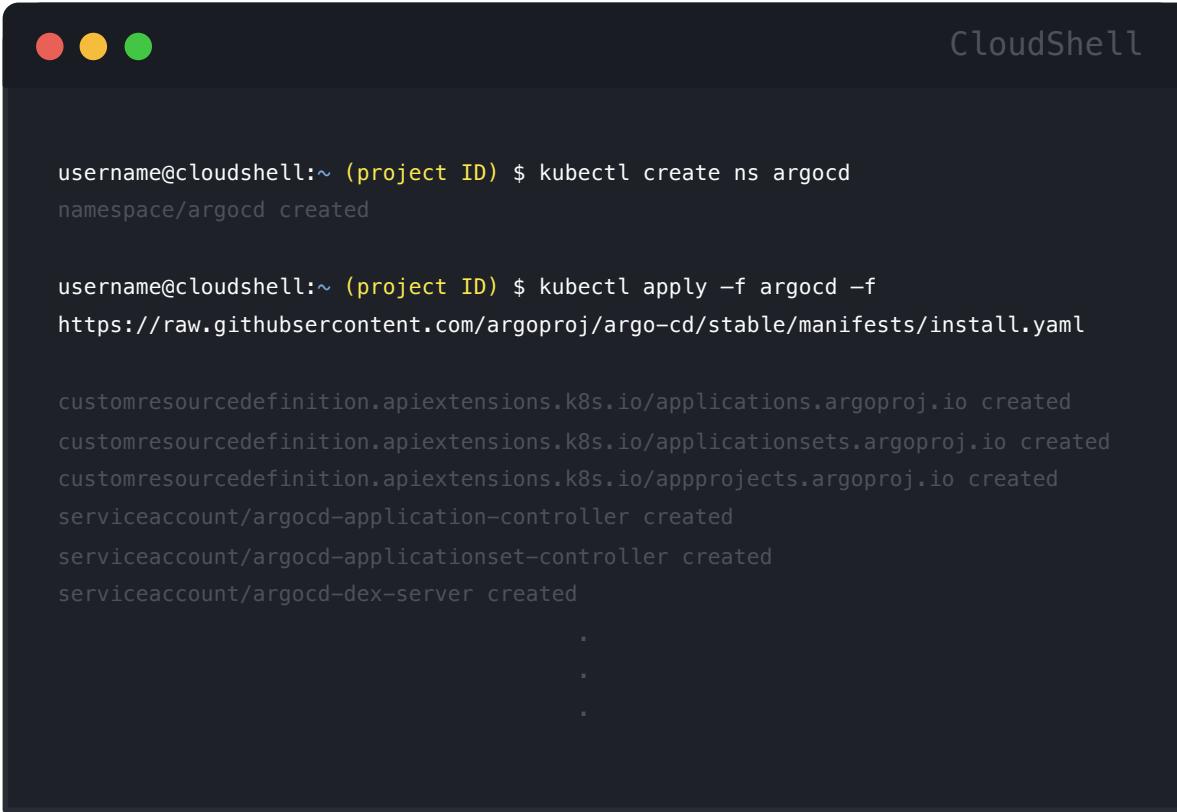
Cloud Console [워크로드 대시보드](#)에서 클러스터에서 실행 중인 워크로드를 볼 수 있습니다.

[워크로드 대시보드 열기](#)

At the bottom right of the modal, there is a '확인' (Confirm) button.



# Install ArgoCD



The screenshot shows a terminal window titled "CloudShell" with a dark theme. It displays the command-line steps to install ArgoCD into a Kubernetes namespace:

```
username@cloudshell:~ (project ID) $ kubectl create ns argocd
namespace/argocd created

username@cloudshell:~ (project ID) $ kubectl apply -f argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created

.
```

# Patch yaml ArgoCD Service



CloudShell

```
username@cloudshell:~ (project ID) $ kubectl get -n argocd
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
argo-controller	ClusterIP	10.76.0.137	<none>	9090/TCP	60s
argo-server	ClusterIP	10.76.8.11	<none>	2746/TCP	60s
argo-workflows-postgresql-exporter-svc	ClusterIP	10.76.5.1	<none>	9187/TCP	60s
argo-workflows-1-postgresql-svc	ClusterIP	10.76.13.151	<none>	5432/TCP	60s

```
username@cloudshell:~ (project ID) $ kubectl patch svc argo-server -n argocd -p '{"spec" : {"type" : "LoadBalancer"} }'
```

#argo-server의 yaml의 spec/type/ClusterIP를 LoadBalancer 또는 NodePort로 수정합니다. 그 이유는 외부에서 접속을 해야하기 때문입니다.  
service/argo-server patched

```
username@cloudshell:~ (project ID) $ kubectl get -n argocd
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
argo-server	LoadBalancer	10.76.8.11	<Your External IP>	2746:31922/TCP	10m



# Get ArgoCD Admin PWD

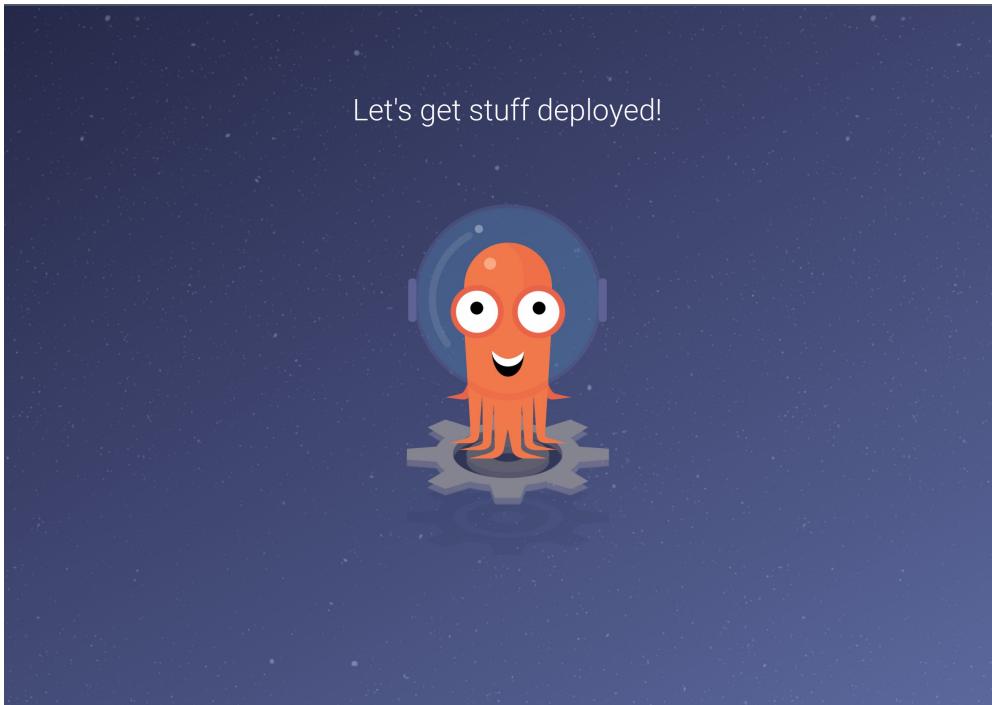


CloudShell

```
username@cloudshell:~ (project ID) $ kubectl get secrets argocd-initial-admin-secret -n argocd  
--template={{.data.password}} | base64 -d; echo  
<decode your argocd password>
```



# Login ArgoCD



argo

Username

Password

SIGN IN

기본 ID는 admin입니다.

argo

Your argo-server External IP



# Settings ArgoCD



# Create Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: fastapi
  name: fastapi
spec:
  selector:
    matchLabels:
      app: fastapi
template:
  metadata:
    labels:
      app: fastapi
  spec:
    containers:
      - name: fastapi
        image: jeff125/kube-fastapi:58a72cdb
        resources:
          limits:
            memory: "128Mi"
            cpu: "500m"
        ports:
          - containerPort: 8000
---
apiVersion: v1
kind: Namespace
metadata:
  name: fastapi
---
apiVersion: v1
kind: Service
metadata:
  name: fastapi-service
spec:
  type: LoadBalancer
  selector:
    app: fastapi
  ports:
    - port: 8000
      targetPort: 8000
```

# Create APP

CREATE CANCEL X

GENERAL EDIT AS YAML

Application Name ArgoCD의 APP 이름

Project Name default

SYNC POLICY Manual ▾

SET DELETION FINALIZER ⓘ

SYNC OPTIONS

SKIP SCHEMA VALIDATION  AUTO-CREATE NAMESPACE

PRUNE LAST  APPLY OUT OF SYNC ONLY

RESPECT IGNORE DIFFERENCES  SERVER-SIDE APPLY

PRUNE PROPAGATION POLICY: foreground ▾

REPLACE ⚠  RETRY

# Create APP

DESTINATION

Cluster URL  
<https://kubernetes.default.svc> URL ▾

Namespace

Directory ▾

DIRECTORY

DIRECTORY RECURSE

TOP-LEVEL ARGUMENTS

No items +

EXTERNAL VARIABLES

No items +

# Check Sync

마지막 commit hash를 확인 할 수 있다.

APP HEALTH ⓘ Healthy

CURRENT SYNC STATUS ⓘ Synced MORE To HEAD (58a72cd)

Author: handgym2 <k36470125@gmail.com> - Comment: fixed

LAST SYNC RESULT ⓘ Sync OK MORE To 58a72cd

Succeeded 7 days ago (Thu Feb 02 2023 13:52:43 GMT+0900)  
Author: handgym2 <k36470125@gmail.com> - Comment: fixed

The diagram illustrates the deployment flow from the application layer down to the pod layer. It shows the following components and their sync status:

- fastapi** (application layer): Synced, 8 days old.
- fastapi** (ns): Synced, 8 days old.
- fastapi-service** (SVC): Synced, 8 days old.
- fastapi-service** (ep): Synced, 8 days old.
- fastapi-service-9l9mj** (endpointslice): Synced, 8 days old.
- fastapi-5f6844d94d** (rs): Synced, 8 days old, rev. 5.
- fastapi-fd4c94dff** (rs): Synced, 7 days old, rev. 6.
- fastapi-fd4c94dff-s9hxk** (pod): Synced, 7 days old, running, 1/1.

A callout box highlights the pods: **실제 서비스하고 있는 POD 이름** (Actual pods in service).

Below the diagram, a table lists the managed pods:

수정 버전	이름	상태	재시작	생성일 ↑
6	<a href="#">fastapi-fd4c94dff-s9hxk</a>	<span>Running</span>	0	2023. 2. 2. PM 1:52:43



# Monitoring GKE

# Default Dashboard

대시보드 목록 샘플 라이브러리

카테고리	전체 대시보드	라벨	?
카테고리별 필터링	<input type="checkbox"/> 대시보드 필터링		
 전체	16	<input type="checkbox"/> ↓ 이름	Type ⓘ 라벨 작업
 최근에 본 항목	7	 <a href="#">인프라 요약</a>	Google Cloud Platform
 즐겨찾기	0	 <a href="#">Autoscaler Monitoring</a>	Google Cloud Platform
 커스텀	1	 <a href="#">Disks</a>	Google Cloud Platform
 GCP	15	 <a href="#">Firewalls</a>	Google Cloud Platform
 통합	0	 <a href="#">GCE VM Instance Monitoring</a>	Google Cloud Platform
 기타	0	 <a href="#">GKE</a>	Google Cloud Platform
		 <a href="#">GKE Active/Idle clusters</a>	Google Cloud Platform
		 <a href="#">GKE Cluster Monitoring</a>	Google Cloud Platform
		 <a href="#">GKE Clusters</a>	Google Cloud Platform
라벨 지정됨		 <a href="#">GKE Compute Resources - Cluster View</a>	Google Cloud Platform
		 <a href="#">GKE Compute Resources - Node View</a>	Google Cloud Platform
		 <a href="#">GKE Compute Resources - Workload View</a>	Google Cloud Platform
		 <a href="#">GKE Nodes and Pods - Cluster View</a>	Google Cloud Platform
	<input type="checkbox"/>	 <a href="#">GKE Nodes and Pods - Cluster View - 2월 3, 2023 3:49 오후</a>	Custom
		 <a href="#">Google Cloud Load Balancers</a>	Google Cloud Platform

모니터링 → 대시보드

# Custom Dashboard

대시보드 목록 [샘플 라이브러리](#)

카테고리	All 샘플	가져오기	설명	개요	가격 책정	문서	지원
카테고리별 필터링	<a href="#">필터 대시보드 필터링</a>						
All	162	<input type="checkbox"/> 이름	This dashboard has charts displaying 'Operations p...	<a href="#">미리보기</a>			
Active Directory DS	2	<input type="checkbox"/> Active Directory Domain Services GCE Overview	This dashboard has charts displaying 'Total Inboun...	<a href="#">미리보기</a>			
		<input type="checkbox"/> Active Directory Replication Agent GCE Overview	This dashboard has 14 charts for viewing ActiveMQ...	<a href="#">미리보기</a>			
ActiveMQ	2	<input type="checkbox"/> ActiveMQ GCE Overview	This dashboard has 14 charts for viewing ActiveMQ...	<a href="#">미리보기</a>			
Aerospike	2	<input type="checkbox"/> Aerospike Prometheus Overview	This dashboard is based on prometheus metrics expo...	<a href="#">미리보기</a>			
		<input type="checkbox"/> Aerospike GCE Overview	This dashboard has 8 charts for viewing Aerospike...	<a href="#">미리보기</a>			
Anthos	8	<input type="checkbox"/> Aerospike Prometheus Overview	This dashboard is based on prometheus metrics expo...	<a href="#">미리보기</a>			
Anthos Config Management	2	<input type="checkbox"/> Airflow Scheduler Prometheus Overview	This dashboard has charts displaying 'Tasks: Runni...	<a href="#">미리보기</a>			
		<input type="checkbox"/> Anthos cluster control plane uptime	This dashboard has 4 charts to indicate the uptime...	<a href="#">미리보기</a>			
Apache	2	<input type="checkbox"/> Anthos cluster node status	This dashboard has 6 charts to indicate the status...	<a href="#">미리보기</a>			
Apache Airflow	1	<input type="checkbox"/> Anthos cluster pod status	This dashboard has 6 charts to indicate the statu...	<a href="#">미리보기</a>			
Apigee	2	<input type="checkbox"/> Anthos Utilization Metering	This dashboard has 6 charts to indicate the reque...	<a href="#">미리보기</a>			
Argo Workflows	1	<input type="checkbox"/> Apache Flink Job Manager Prometheus Overview	This dashboard is based on prometheus metrics expo...	<a href="#">미리보기</a>			
		<input type="checkbox"/> Apache Flink Task Manager Prometheus Overview	This dashboard is based on prometheus metrics expo...	<a href="#">미리보기</a>			
Big Data	3	<input type="checkbox"/> Apache GCE Overview	This dashboard has charts displaying 'Request Rate...	<a href="#">미리보기</a>			
Bindplane	1	<input type="checkbox"/> Apache Prometheus Overview	This dashboard is based on Prometheus metrics expo...	<a href="#">미리보기</a>			

[◀](#) 제품 세부정보



Prometheus & Grafana

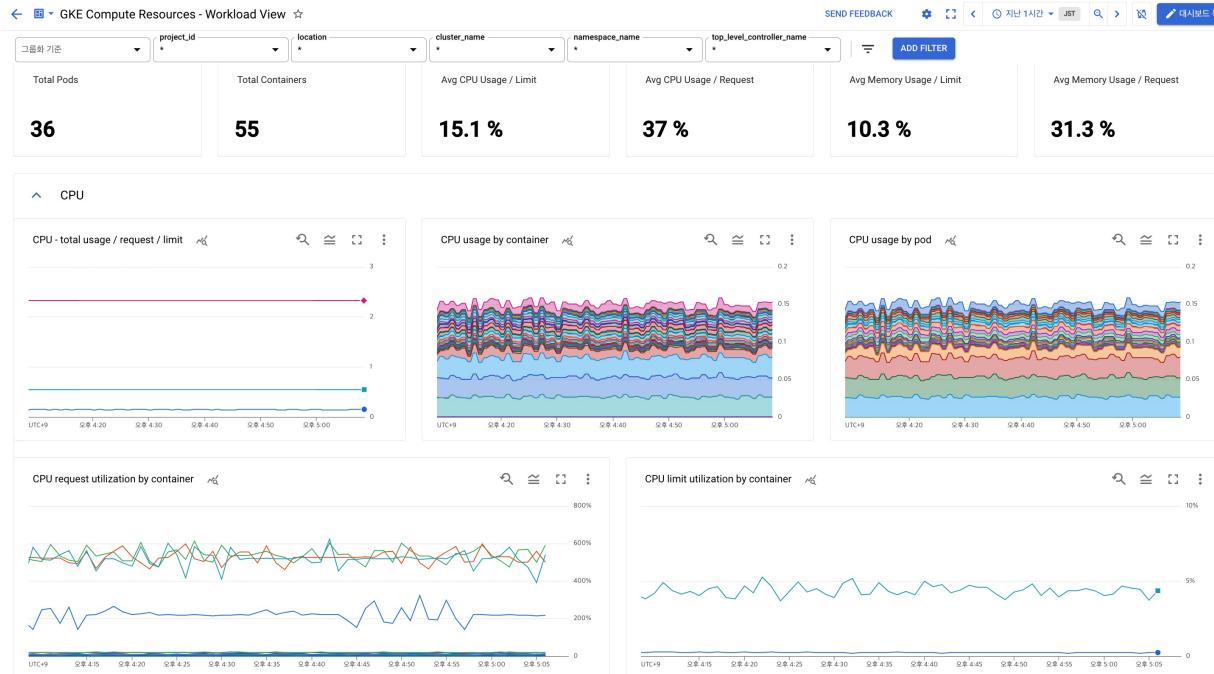
버전: 2.41

[Google Click to Deploy containers](#)

Fully functional GKE monitoring platform

구성

# Metrics Dashboard



이 대시보드는 GKE Cluster의 Metrics를 확인 할 수 있습니다.

또한 Node, POD 등의 Metrics를 확인 할 수 있는 템플릿도 따로 있습니다



GCP의 UI/UX가 상당히 잘 만들어졌으며, 검색을 많이 안하고도 GKE를 사용했습니다. 또한 따로 SSH 접속 할 필요 없이 Web SSH로도 설정이 가능했던 점이 굉장히 편했습니다.

Gitlab의 CI/CD를 구축 했을 때 Gitlab의 기능들을 눌러보며 연구해봤으며 그 중에서도 직접 쿠버네티스, GCP에 연동하기 쉽게 한 것도 엄청난 장점이라고 생각했습니다.

ArgoCD를 설정했을 때 “진짜 쿠버네티스를 위해서 만든건가 보네”라는 생각을 했습니다. Manifest를 Update만 하면 GKE의 POD들이 교체가 되는 것을 보고 정말 짜릿했습니다.