

Ch 5. 도커 레지스트리

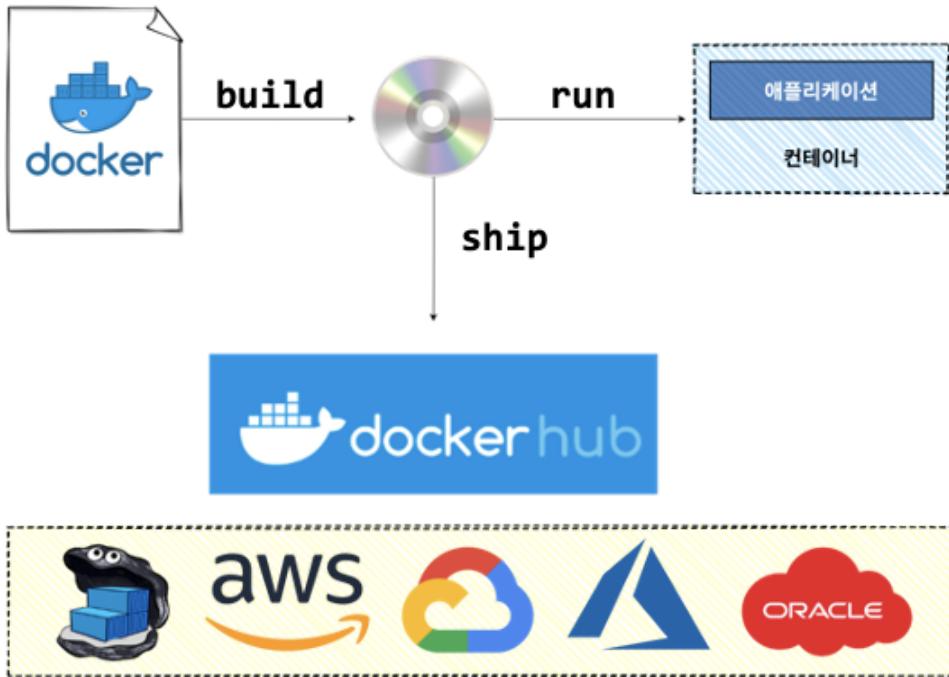
5. 도커 레지스트리

도커 레지스트리는 도커 이미지들을 모아 놓고 필요할 때 사용할 수 있도록 구축한 저장소입니다. 도커 이미지를 빌드하거나 컨테이너를 실행할 수 있었던 것도 레지스트리의 이미지를 사용하기 때문이죠.

도커를 처음으로 소개할 때, 제가 도커라는 플랫폼을 정의했던 문장이 다음과 같습니다.

 도커는 어플리케이션 및 실행 환경을 정의한 이미지를 생성/공유함과 동시에, 이를 이용하여 컨테이너를 작동할 수 있도록 하는 플랫폼이다.

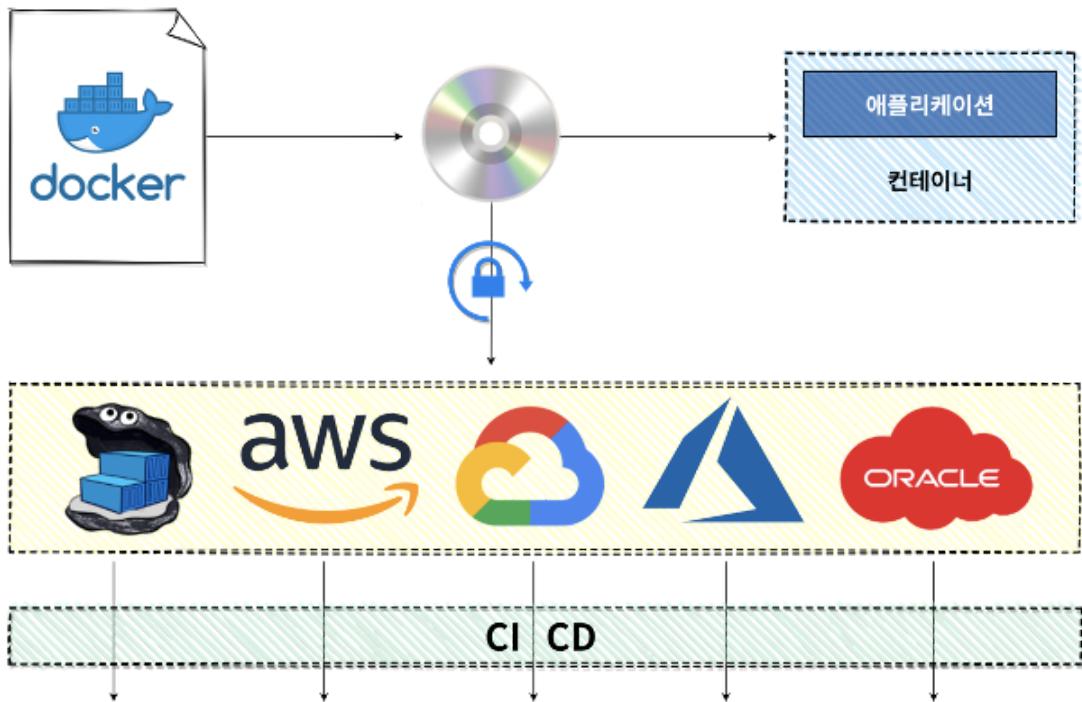
컨테이너 실행, 이미지 빌드를 앞 부분에서 다루었다면 이번 시간에는 레지스트리를 통하여 이미지를 공유하는 여러 가지 방법과 이용 가능한 플랫폼 등을 확인해보겠습니다.



도커의 기능을 한 눈에 정리할 수 있도록 그림으로 나타내 보았습니다. Dockerfile을 통해 빌드된 이미지가 컨테이너를 실행하는 데에 이용되기도 하고, 혹은 레지스트리에 저장되어 여러 사람이 사용할 수 있도록 공개되기도 하죠. 이를 각각 Building, Running, Shipping(Sharing) 이라는 단어로 표현합니다.

도커 이미지가 공유되는 레지스트리를 크게 3가지로 구분해보면, 디풀트로 이미지를 받아오는 도커 社의 도커 허브, 사용자가 직접 컨테이너에 구축한 로컬 레지스트리, 클라우드 사업자에서 제공하는 컨테이너 레지스트리가 있습니다. 뒤에 이어지는 내용에서 이 3가지를 모두 다룰 예정입니다.

본격적으로 레지스트리 구축을 실습하기 전에 레지스트리를 사용하는 목적에 대해 짚고 넘어가도록 하겠습니다. 이미지를 파일 형식으로도 저장 가능한데 왜 굳이 비용을 지불하면서 레지스트리를 사용하는 걸까요? 여기에는 여러 이유가 있습니다.



첫 번째는 보안 문제입니다. 이미지 내부에 보안상 중요한 파일이나 내용이 필요할 경우에는 더욱 엄격히 관리 할 필요가 있습니다. 이미지 자체도 암호화 되어야 하고 레지스트리에 접근할 수 있는 권한도 통제해야 하죠. 이런 일련의 것들을 클라우드 사업자에서는 IAM(Identity and Access Management) 라는 형태로 제공하고 있습니다.

두 번째는 배포 파이프라인 효율화입니다. 위 그림에서 보다시피 레지스트리에 저장된 이미지는 단순히 저장에서 끝나는 것이 아니라 CI/CD라는 과정까지 연속적으로 활용됩니다. CI/CD 용어에 대해서는 하단에 부연하였습니다.



CI(Continuous Integration) : 지속적 통합

빌드/테스트가 자동으로 수행된 이후 개발자의 수정 사항이 중앙 저장소에 머지되는 것 까지의 과정
일련의 자동화 과정 및 개발 문화가 포함



CD(Continuous Delivery) : 지속적 전달

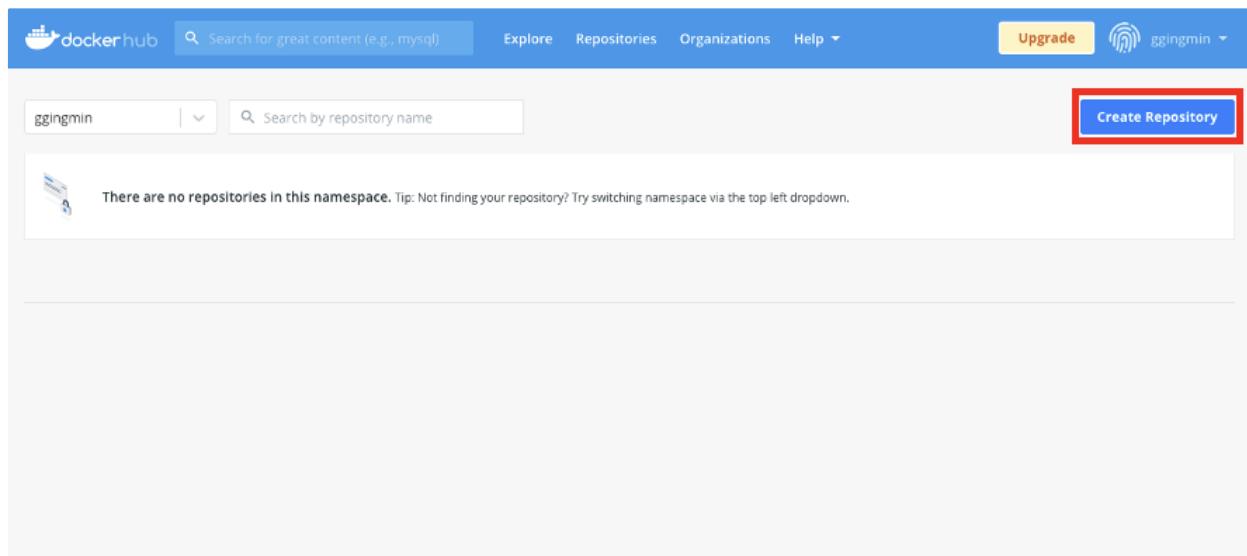
운영 환경에 배포할 소스코드가 자동으로 세팅
빌드 이후의 변경 사항을 테스트 및 운영 서버에 자동으로 배포
필요에 따라 테스트 서버를 동적으로 생성할 수 있음

5.1 도커 허브

- [Docker Hub 페이지](#)

도커 허브는 도커에서 디풀트로 참조하는 레지스트리로서, 각종 공식 이미지를 손쉽게 사용할 수 있도록 구성되어 있습니다. 또한 무료 계정을 생성하여 개인이 빌드한 이미지도 공개할 수 있죠. 실습을 진행하기 전에 계정 생성을 먼저 진행해야 합니다.

1) 저장소 만들기



계정 생성이 완료되면 'Create Repository' 버튼을 눌러 저장소를 만들어 줍니다.

A screenshot of the 'Create Repository' form. The top navigation bar is identical to the previous one. The main form has a title 'Create Repository' and a namespace dropdown set to 'gggingmin' with a sub-menu 'portfolio'. There's a 'Description' input field which is empty. On the right, a 'Pro tip' section contains a code snippet: 'docker tag local-image:tagname new-repo:tagname' and 'docker push new-repo:tagname'. Below this, a note says 'Make sure to change tagname with your desired image repository tag.' At the bottom, there are 'Cancel' and 'Create' buttons. The 'Create' button is highlighted with a red box.

저장소명은 용도에 맞게 자유롭게 입력해주면 됩니다. **Visibility** 항목에서는 도커 허브에서 내 저장소가 검색 결과에 조회되는 가에 대해 **Public**과 **Private** 설정을 할 수 있는데, 무료계정의 경우 하나의 저장소만 Private 으로 설정할 수 있습니다.

The screenshot shows a Docker Hub repository page for 'ggingmin/portfolio'. At the top, there's a search bar, navigation links for 'Explore', 'Repositories', 'Organizations', and 'Help', and a yellow 'Upgrade' button. The user 'ggingmin' is logged in. Below the header, the repository path 'ggingmin > Repositories > portfolio' is shown, along with a message about unused content and a link to 'View preview'. The main content area includes a section for 'Advanced Image Management', a repository summary ('ggingmin / portfolio'), and a 'Docker commands' section with a 'Public View' button. There are also sections for 'Tags and Scans' (disabled) and 'Automated Builds'.

저장소가 성공적으로 생성되었다면 위와 같은 화면이 나오게 됩니다. 아직은 이미지가 올라가거나 별도 작업을 하지 않았기 때문에 특별한 내용은 없습니다.

오른쪽 하단에 '**Automated Builds**'라는 항목이 있는데 이는 Github나 Bitbucket에 push한 소스를 기반으로 자동으로 이미지를 빌드해주는 기능입니다. 올해 초까지만 하더라도 무료계정을 이용하는 유저 역시 사용이 가능했지만 6월 7일, 무료 계정에 제공되는 기능에서 제외되었습니다. 관련하여 도커 공식 블로그 링크를 다음과 같이 공유드립니다.

- [Docker blog - Changes to Docker Hub Autobuilds](#)

자동 빌드와 관련해서는 GCP에서 제공하는 Cloud Build API를 통해 실습할 예정입니다.

2) 예제코드 클론

```
mkdir <프로젝트명> cd <프로젝트명> git clone https://github.com/ggingmin/portfolio.git
```

위와 같이 예제 소스를 Ubuntu 가상머신에 받습니다.

3) 이미지 빌드

```
sudo docker build -t <계정명>/<저장소명>: [태그명] . sudo docker build -t ggingmin/portfolio:1.0 .
```

```
FROM nginx:alpine # alpine은 경량화 리눅스 배포판입니다. # 도커 베이스 이미지로 alpine 리눅스를 활용하는 이유는 이미지 용량을 적게 차지할 뿐만 아니라 처리속도가 빠르다는 이점이 있기 때문입니다. WORKDIR /usr/share/nginx/html # 이 경로는 nginx 웹서버의 index.html 파일이 위치한 곳입니다. # 브라우저를 통해 웹서버로 접근했을때 로드되는 페이지가 바로 이 곳을 참조하여 렌더링 되는 것입니다. RUN rm -rf /* # 클론한 소스를 이미지에 복사하기 위해 기존 이미지 내의 파일을 전부 삭제합니다. COPY ./ ./ # Dockerfile이 위치한 경로의 html, css, js 등의 파일을 이미지 내로 복사합니다. ENTRYPOINT ["nginx", "-g", "daemon off;"] # nginx 웹서버를 기동합니다.
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker build -t ggingmin/portfolio:1.0 .
Sending build context to Docker daemon 2.068MB
Step 1/5 : FROM nginx:alpine
--> 7ce0143dee37
Step 2/5 : WORKDIR /usr/share/nginx/html
--> Running in b2fc7d7ae15e
Removing intermediate container b2fc7d7ae15e
--> ecd3d2d40a04
Step 3/5 : RUN rm -rf /*
--> Running in 61c3d8ab4271
Removing intermediate container 61c3d8ab4271
--> 2839813e9880
Step 4/5 : COPY ./ .
--> 3e0a7fa97914
Step 5/5 : ENTRYPOINT ["nginx", "-g", "daemon off;"]
--> Running in a09acea94aa3
Removing intermediate container a09acea94aa3
--> 96d10654d93b
Successfully built 96d10654d93b
Successfully tagged ggingmin/portfolio:1.0
```

- nginx:alpine Github repository

도커 허브에 이미지를 공유하기 위해서는 빌드할 때 반드시 명명규칙을 따라야합니다. <계정명>:<저장소명> 의 형태가 맞는지 꼭 확인해주세요. 성공적으로 빌드가 됐다면 위와 같이 메세지가 출력됩니다.

4) 도커 허브 계정 로그인

```
sudo docker login
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID
, head over to https://hub.docker.com to create one.
Username: ggingmin
Password:
WARNING! Your password will be stored unencrypted in /home/ggingmin/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

명령어를 입력하면 username과 password를 입력하는 프롬프트가 출력되고 계정정보가 맞다면 위와 같이 로그인 됩니다.

5) 도커 허브 이미지 공유

```
sudo docker <계정명>/<저장소명>: [태그명] sudo docker push ggingmin/portfolio:1.0
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker push ggingmin/portfolio:1.0
The push refers to repository [docker.io/ggingmin/portfolio]
c0b62fc57690: Pushed
f45e1958ca6c: Pushed
cb460864147e: Mounted from library/nginx
c126a1b4f317: Mounted from library/nginx
a33cbf3e5439: Mounted from library/nginx
b9239390c1b8: Mounted from library/nginx
82c3b921d80c: Mounted from library/nginx
bc276c40b172: Mounted from library/nginx
1.0: digest: sha256:1d441e3d3b2236f6d6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9 size: 1986
```

이미지 명명 규칙 및 계정정보에 이상이 없다면 정상적으로 이미지가 공유됩니다.

The screenshot shows the GitHub Docker Hub repository page for 'ggingmin/portfolio'. At the top, there's a navigation bar with 'ggingmin' and 'Repositories' followed by 'portfolio'. Below the navigation, there are tabs for 'General', 'Tags', 'Builds', 'Collaborators', 'Webhooks', and 'Settings'. A message indicates 'Using 0 of 1 private repositories. Get more'. Under the 'General' tab, there's a section titled 'Advanced Image Management' with a link to 'View preview'. Below this, the repository name 'ggingmin / portfolio' is displayed, along with a note that it 'This repository does not have a description'. It shows the last push was 3 minutes ago. To the right, there's a 'Docker commands' section with a button to 'Push a new tag to this repository' and a command box containing 'docker push ggingmin/portfolio:tagname'. The 'Tags and Scans' section shows one tag, '1.0', which was pushed 3 minutes ago. There's also a 'VULNERABILITY SCANNING - DISABLED' link with an 'Enable' button. The 'Automated Builds' section is shown as available on Pro and Team plans, with a 'Upgrade to Pro' button and a 'Learn more' link.

The screenshot shows the Docker Hub image details page for 'ggingmin/portfolio:1.0'. At the top, there's a large blue hexagonal icon, the image name 'ggingmin/portfolio:1.0', its digest 'sha256:1d441e3d3b2236f6d6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9', and a 'Delete Tag' button. Below this, there are sections for 'OS/ARCH' (linux/amd64), 'COMPRESSED SIZE' (10.68 MB), and 'LAST PUSHED' (4 minutes ago by ggingmin). Underneath, there are tabs for 'Image Layers' and 'Vulnerabilities', with 'Image Layers' currently selected. The 'IMAGE LAYERS' section lists seven layers with their corresponding commands and sizes:

Layer	Command	Size
1	ADD file ... in /	2.68 MB
2	CMD ["/bin/sh"]	9.5 kB
3	LABEL maintainer=NGINX Docker Maintainers	9.5 kB
4	ENV NGINX_VERSION=1.21.1	9.5 kB
5	ENV NJS_VERSION=0.6.1	9.5 kB
6	ENV PKG_RELEASE=1	9.5 kB
7	/bin/sh -c set -x	6.78 MB

도커 허브 계정에서도 역시 공유한 이미지 정보 및 태그 등을 확인할 수 있으며 Public으로 저장소를 설정한 경우 검색 결과에도 조회됩니다.

5.2 로컬 레지스트리 구축

도커를 통해 실행되는 어플리케이션은 컨테이너 단위로 실행됩니다. 이미지가 저장되는 레지스트리 역시 컨테이너 상에 구축할 수 있는데요, 이는 도커 社에서 Registry라는 이름으로 제공하고 있습니다.

1) Registry 컨테이너 실행

```
sudo docker run -d -p 5000:5000 --restart always --name registry registry:2 #  
--restart always의 경우 도커 엔진이 재시작되는 경우 자동으로 컨테이너를 재시작하도록 하는 옵션입니다.
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker ps  
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS  
NAMES  
cb374590619a registry:2 "/entrypoint.sh /etc..." 51 seconds ago Up 49 seconds 0.0.0.0:5000->5000/tcp, :::5000->5000/tcp registry
```

위와 같이 정상적으로 컨테이너가 잘 실행되고 있는지 확인해보았습니다.

2) 이미지 태깅

```
sudo docker tag <기존 이미지명>:[태그명] <레지스트리 컨테이너 IP>/<이미지명>:[태그명] sudo  
docker tag ggingmin/portfolio:1.0 localhost:5000/portfolio:1.0
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker image ls  
REPOSITORY TAG IMAGE ID CREATED SIZE  
ggingmin/portfolio 1.0 96d10654d93b About an hour ago 24.9MB  
portfolio 1.0 96d10654d93b About an hour ago 24.9MB  
localhost:5000/portfolio 1.0 96d10654d93b About an hour ago 24.9MB  
nginx alpine 7ce0143dee37 4 days ago 22.8MB  
registry 2 1fd8e1b0bb7e 3 months ago 26.2MB  
hello-world latest d1165f221234 5 months ago 13.3kB
```

컨테이너에 구축한 레지스트리에 이미지를 공유하는 경우 <컨테이너 IP>:<포트>/<이미지명>의 양식을 반드시 지켜야 합니다. 포트의 경우 컨테이너를 실행할 당시 5000으로 지정이 되어 있었으나 구성 상 변경이 필요한 경우 태그를 설정할 때도 반영해주어야 합니다.

3) 이미지 공유

```
sudo docker push localhost:5000/portfolio:1.0
```

```
ggingmin@ubuntu-server:~/project/portfolio$ sudo docker push localhost:5000/portfolio:1.0  
The push refers to repository [localhost:5000/portfolio]  
c0b62fc57690: Pushed  
f45e1958ca6c: Pushed  
cb460864147e: Pushed  
c126a1b4f317: Pushed  
a33cbf3e5439: Pushed  
b9239390c1b8: Pushed  
82c3b921d80c: Pushed  
bc276c40b172: Pushed  
1.0: digest: sha256:1d441e3d3b2236f6d6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9 size: 1986
```

명령을 실행하면 도커 하브와는 다른 경로로 이미지가 공유된 것을 확인하실 수 있습니다.

5.3 GCP Artifact Registry

AWS, MS Azure, GCP 등... 많은 클라우드 사업자가 현재 기업 및 개인에게 서비스를 제공하고 있습니다. IT 업계에서의 영향력도 점점 커지고 있죠. 하지만 이를 활용해 볼 수 있는 경험을 얻기 위해서는 다소 수고스러운 과정을 거쳐야 하고 매뉴얼도 다소 불친절하게 느껴져 포기하는 경우가 많습니다. 하지만 구축 과정을 한 사이클 진행해봄으로써 클라우드 운용의 다양한 요소들을 습득할 수 있기 때문에 포기하지 마시고 꼭 실습을 진행해보셨으면 좋겠습니다.

본 실습은 GCP에서 제공하는 무료 API를 이용하여 진행하겠습니다.

1) GCP Console 접근 및 프로젝트 생성

- GCP Console 페이지



Google Cloud로 앞당기는 혁신

앱을 더 빠르게 빌드하고 보다 협업한 비즈니스 의사결정을 내리며 세계 각지의 사람들과 소통할 수 있습니다.

Console로 이동

데이터 기반 문화 구축을 위한 Google의 가이드

데이터 기반 문화 구축을 위한 가이드를 다운로드하세요

☰ Google Cloud Platform

새 프로젝트

projects 할당량이 21개 남았습니다. 할당량 증가를 요청하거나 프로젝트를 삭제하세요. [자세히 알아보기](#)

[MANAGE QUOTAS](#)

프로젝트 이름 *
docker-registry

프로젝트 ID *
ggingmin-docker-registry

프로젝트 ID에는 소문자, 숫자, 하이픈을 사용할 수 있으며 소문자로 시작하고 문자나 숫자로 끝나야 합니다.

위치 *
 조직 없음

상위 조직 또는 풀더

[만들기](#) [취소](#)

GCP에서는 프로젝트 단위로 어플리케이션을 관리합니다. Firebase 등의 서비스도 GCP에서 관리되는 요소 중 하나입니다. 위와 같이 프로젝트 이름과 ID를 설정해주시면 되고 ID의 경우에는 gcloud(GCP CLI)에서 활용되므로 잘 관리해야 합니다.

2) 결제 계정 등록

The screenshot shows the Google Cloud Platform dashboard for a project named 'docker-registry'. The '결제' (Billing) section is highlighted with an orange box. The sidebar on the left lists various services: Home, 최근 (Recent), Artifact Registry, Cloud Build, API 및 서비스 (API & Services), 결제 (Billing), Marketplace, API 및 서비스 (API & Services) again, 지원 (Support), and IAM 및 관리자 (IAM & Admin). The main content area displays '결제 정보' (Billing Information) with fields for 프로젝트 이름 (Project Name: docker-registry), 프로젝트 ID (Project ID: gcr-docker-registry), and 프로젝트 번호 (Project Number: 751016727). A button labeled '프로젝트에 사용자 추가' (Add user to project) is visible. Below this, the 'Marketplace' section is shown with a message: '프로젝트에는 리소스가 없습니다.' (No resources in the project). A tooltip '프로젝트 설정으로 이동' (Move to project settings) is visible over the Marketplace link.

GCP에서 제공하는 무료 기능을 이용하기 위해서는 결제 계정을 등록해야 합니다. 해외 결제가 가능한 신용/체크카드를 등록하면 됩니다.

3) 서비스 사용 등록

This screenshot is similar to the previous one, but the '결제' (Billing) section has been expanded to show its sub-menu. The sub-menu includes '대시보드' (Dashboard), which is highlighted with an orange box, and '라이브러리' (Library). Other options in the sub-menu are '사용자 인증 정보' (User authentication information), 'OAuth 등의 허면' (OAuth and other consent), '도메인 확인' (Domain verification), and '페이지 사용 통계' (Page usage statistics). The rest of the dashboard and sidebar are identical to the first screenshot.



Artifact Registry API
Google Enterprise API

사용 API 사용해 보기

개요 가격 책정 문서

개요

With Artifact Registry you can store and manage your build artifacts (e.g. Docker images, Maven packages, npm packages). In a scalable and integrated repository service built on Google infrastructure. You can manage repository access with IAM and interact with repositories via gcloud, Cloud Console, and native package format tools. The service can also be integrated with Cloud Build and other CI/CD systems. Artifact Registry abstracts away infrastructure management, so you can focus on what matters most – delivering value to the users of your services and applications. Note: Enabling the Artifact Registry API will not affect your use of Container Registry in the same project.

[자세히 알아보기](#)

가격 책정

Artifact Registry Storage	무료	USD 0.10
Artifact Registry GOOGLE-API Egress	Starting from 0 gibibyte/month	/gibibyte month
Artifact Registry Network HTTP/Load Balancing Egress to Load Balancer	Starting from 0.5 gibibyte/month/B	/gibibyte month/B



Cloud Build API
Google Enterprise API

Continuously build, test, and deploy.

사용 API 사용해 보기

개요 가격 책정 문서

개요

Cloud Build, Google Cloud's continuous integration (CI) and continuous delivery (CD) platform, lets you build software quickly across all languages. Get complete control over defining custom workflows for building, testing, and deploying across multiple environments such as VMs, serverless, Kubernetes, or Firebase.

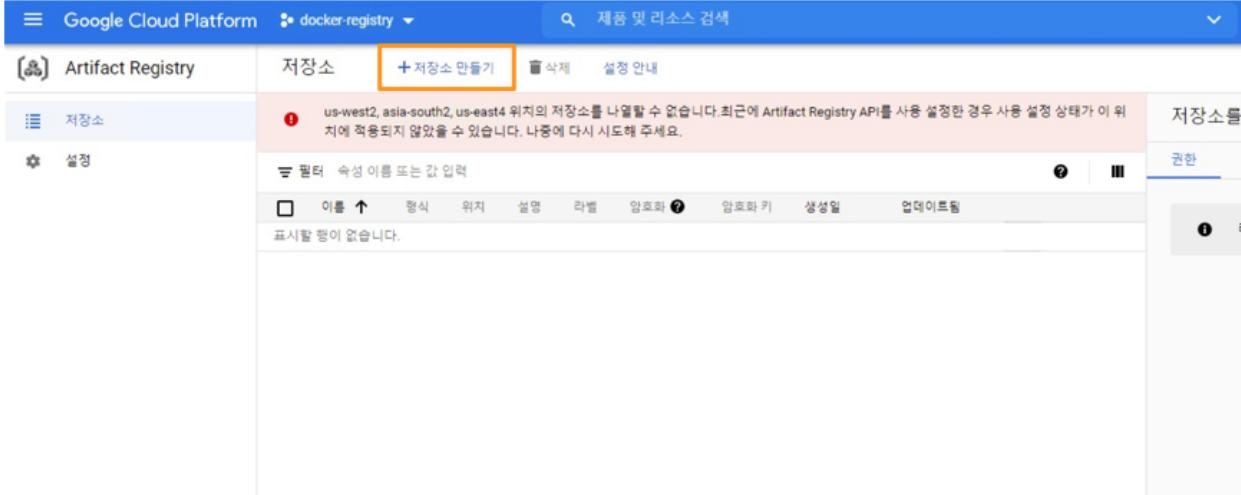
[자세히 알아보기](#)

가격 책정

Build time	무료	USD 0.00
Build time n1-highcpu-32	Starting from 0 minutes of build time/B	/minutes of build time
Build time n1-highcpu-8	Starting from 0 minutes of build time/B	/minutes of build time
SSD disk for build time	Starting from 120 minutes of build time/B	/minutes of build time

'API 및 서비스 - 라이브러리' 메뉴를 들어가면 GCP에서 제공하는 여러 API를 한 눈에 볼 수 있습니다. 이 중에 'Artifact Registry'와 'Cloud Build'를 사용하기 위해 각 서비스를 검색하여 '사용' 버튼을 누릅니다. 이 두 서비스는 일정 한도 내에서는 무료로 제공되므로 실습에 별도의 비용을 지불할 필요가 없습니다.

4) GCP 저장소 만들기



The screenshot shows the Google Cloud Platform interface for creating a new storage location in the Artifact Registry. The top navigation bar includes the Google Cloud Platform logo, a search bar, and a dropdown menu for 'docker-registry'. The main area has a sidebar with 'Artifact Registry' selected under '저장소' (Storage). A red box highlights the '+ 저장소 만들기' (Create new storage) button. Below it, a message states: 'us-west2, asia-south2, us-east4 위치의 저장소를 나열할 수 없습니다. 최근에 Artifact Registry API를 사용 설정한 경우 사용 설정 상태가 이 위치에 적용되지 않았을 수 있습니다. 나중에 다시 시도해 주세요.' (No storage locations are listed for us-west2, asia-south2, us-east4 locations. If you recently used the Artifact Registry API, the usage status for this location may not have been applied. Try again later.)

The screenshot shows the Google Cloud Platform Artifact Registry interface. On the left, there's a sidebar with 'Artifact Registry' selected under 'Storage'. The main area has a title '저장소 만들기' (Create Repository) with a back arrow. A search bar at the top right says '제품 및 리소스 검색'. The form fields include:

- 이름:** portfolio
- 형식:** Docker (radio button selected)
- 위치:** asia-northeast3(서울) (dropdown menu)
- 암호화:** Google 관리 암호화 키 (radio button selected)
- 생성일:** 선택 (dropdown menu)
- 업데이트일:** 선택 (dropdown menu)
- 리포지토리 타입:** Docker (radio button selected)
- 설정:** 설정 (dropdown menu)
- 리포지토리 목록:** portfolio (list item)
- 설정:** 설정 (button)

At the bottom, there are two buttons: '만들기' (Create) and '취소' (Cancel).

The screenshot shows the same Google Cloud Platform Artifact Registry interface, but now it lists the newly created repository 'portfolio' in the main table. The table columns are: 이름 (Name), 형식 (Format), 위치 (Location), 설명 (Description), 리포지토리 타입 (Repository Type), 암호화 키 (Encryption Key), 암호화 일 (Encryption Date), 생성 일 (Creation Date), and 업데이트 일 (Update Date). The 'portfolio' entry shows 'Docker' as the format, 'asia-northeast3(서울)' as the location, and 'Google 관리 키' as the encryption key.

Artifact Registry 콘솔에서 '저장소 만들기' 버튼을 누릅니다. 도커 허브에서 저장소를 만드는 것과 같은 과정으로 보시면 됩니다. 이어서 저장소 설정이 나오게 되는데 형식은 'Docker'로, 리전은 'asia-northeast3(서울)'로 설정해줍니다. 암호화는 'Google 관리 암호화 키'를 선택합니다.

마지막 이미지와 같이 저장소가 생성되었다면 다음 단계로 넘어가도록 하겠습니다.

5) 리눅스 도커 보안 그룹 설정

```
sudo usermod -a -G docker [계정명]
```

GCP CLI를 사용하기 전에 보안 그룹에 현재 사용 중인 계정명을 추가합니다. 이후에는 `sudo` 명령어를 함께 입력하지 않아도 docker 명령어를 사용할 수 있습니다.

6) google cloud SDK 패키지 경로 추가

```
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" | sudo tee -a
/etc/apt/sources.list.d/google-cloud-sdk.list
```

패키지 소스 URI를 추가해줍니다. 우리가 사용하려는 gcloud는 이 경로를 통해 내려받게 됩니다.

7) google cloud SDK 공개키 내려받기

```
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key --keyring /usr/share/keyrings/cloud.google.gpg add -
```

8) google cloud SDK 설치

```
sudo apt-get update && sudo apt-get install google-cloud-sdk
```

google cloud SDK 설치를 진행합니다. 기타 패키지에 비해 용량이 큰 관계로 다소 시간이 소요됩니다.

9) google cloud SDK 초기화

```
gcloud init
```

```
ggingmin@ubuntu-server:~/project/portfolio$ gcloud init
Welcome! This command will take you through the configuration of gcloud.

Your current configuration has been set to: [default]

You can skip diagnostics next time by using the following flag:
  gcloud init --skip-diagnostics

Network diagnostic detects and fixes local network connection issues.
Checking network connection...done.
Reachability Check passed.
Network diagnostic passed (1/1 checks passed).

You must log in to continue. Would you like to log in (Y/n)?  y

Go to the following link in your browser:

  https://accounts.google.com/o/oauth2/auth?response_type=code&client_id=32555940559.apps.goog
[REDACTED]

Enter verification code: |
```

```
Enter verification code: [REDACTED]
You are logged in as: [ggingmin@gmail.com].  
  
Pick cloud project to use:  
[REDACTED]  
[2] ggingmin-docker-registry  
[REDACTED]  
[4] Create a new project  
Please enter numeric choice or text value (must exactly match list  
item): |  
  
Your current project has been set to: [ggingmin-docker-registry].  
  
Not setting default zone/region (this feature makes it easier to use  
[gcloud compute] by setting an appropriate default value for the  
--zone and --region flag).  
See https://cloud.google.com/compute/docs/gcloud-compute section on how to set  
default compute region and zone manually. If you would like [gcloud init] to be  
able to do this for you the next time you run it, make sure the  
Compute Engine API is enabled for your project on the  
https://console.developers.google.com/apis page.  
  
Created a default .boto configuration file at [/home/ggingmin/.boto]. See this file and  
[https://cloud.google.com/storage/docs/gsutil/commands/config] for more  
information about configuring Google Cloud Storage.  
Your Google Cloud SDK is configured and ready to use!  
  
* Commands that require authentication will use ggingmin@gmail.com by default  
* Commands will reference project 'ggingmin-docker-registry' by default  
Run 'gcloud help config' to learn how to change individual settings  
  
This gcloud configuration is called [default]. You can create additional configurations if you w  
ork with multiple accounts and/or projects.  
Run 'gcloud topic configurations' to learn more.
```

google cloud SDK 초기화를 위해 `gcloud init` 명령을 실행하였습니다. 이후 google 계정에 로그인을 할 수 있는 링크가 나오게 되는데 링크에 접속하여 인증코드를 복사한 후 프롬프트에 붙여넣기 하시면 됩니다. 도커 허브 계정으로 로그인 했던 것에 비해서 다소 복잡하다고 느끼실 수 있으나 차근차근 진행해주시면 됩니다.

성공적으로 로그인이 되었다면 프로젝트를 선택하는 프롬프트가 뜨는데, 위에서 생성했던 프로젝트를 선택해주시면 됩니다.

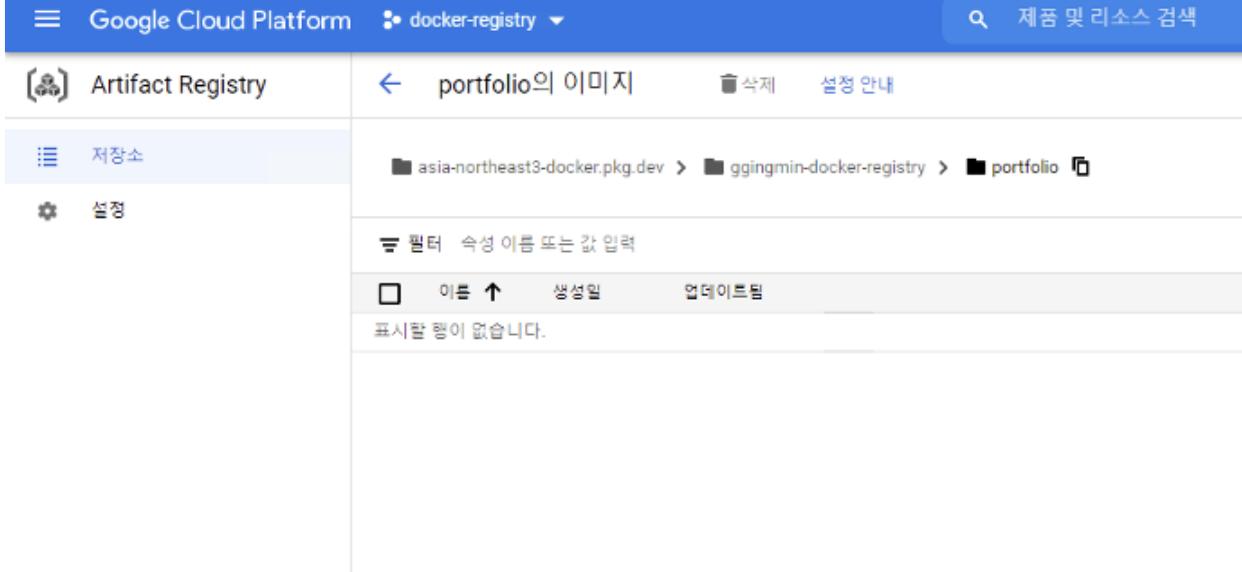
10) GCP Registry 저장소 인증

```
sudo gcloud auth configure-docker asia-northeast3-docker.pkg.dev  
  
ggingmin@ubuntu-server:~/project/portfolio$ sudo gcloud auth configure-docker asia-northeast3-d  
cker.pkg.dev  
[sudo] password for ggingmin:  
Adding credentials for: asia-northeast3-docker.pkg.dev  
After update, the following will be written to your Docker config file  
located at [/home/ggingmin/.docker/config.json]:  
{  
  "credHelpers": {  
    "asia-northeast3-docker.pkg.dev": "gcloud"  
  }  
}  
  
Do you want to continue (Y/n)? y  
  
Docker configuration file updated.
```

이 명령은 루트 권한이 필요하므로 반드시 `sudo` 명령을 붙여주셔야 합니다. 정상적으로 실행이 되면 `/home/<username>/docker/config.json` 이라는 파일과 함께 저장소 경로가 등록됩니다.

11) 이미지 태깅

```
sudo docker tag ggingmin/portfolio:1.0 asia-northeast3-docker.pkg.dev/[프로젝트ID]/[저장소명]/portfolio sudo docker tag ggingmin/portfolio:1.0 asia-northeast3-docker.pkg.dev/ggingmin-docker-registry/portfolio/portfolio
```

A screenshot of the Google Cloud Platform Artifact Registry interface. The top navigation bar shows 'Google Cloud Platform' and 'docker-registry'. The main area is titled 'Artifact Registry' and shows a breadcrumb path: 'asia-northeast3-docker.pkg.dev > ggingmin-docker-registry > portfolio'. There is a search bar and a delete button. Below the path, there is a filter input field and sorting options ('이름 ↑', '생성일', '업데이트됨'). A message at the bottom says '표시할 행이 없습니다.'.

GCP의 레지스트리에 이미지를 업로드 하려면 이미지명 양식을 맞춰 주어야 합니다. 이미 도커 허브와 로컬 레지스트리 구축 때에도 한 번 실습을 진행했었죠. 이미지명이 너무 길어서 한숨이 나오실 수도 있지만 GCP 콘솔의 저장소로 들어가시면 저렇게 클립보드로 복사해주는 버튼이 있습니다.

12) 이미지 공유

```
sudo docker push asia-northeast3-docker.pkg.dev/[프로젝트ID]/[저장소명]/portfolio  
sudo docker push asia-northeast3-docker.pkg.dev/ggingmin-docker-registry/portfolio/portfolio
```

```
ggingmin@ubuntu-server:~$ sudo docker push asia-northeast3-docker.pkg.dev/ggingmin-docker-registry/portfolio/portfolio  
Using default tag: latest  
The push refers to repository [asia-northeast3-docker.pkg.dev/ggingmin-docker-registry/portfolio/portfolio]  
c0b62fc57690: Pushed  
f45e1958ca6c: Pushed  
cb460864147e: Pushed  
c126a1b4f317: Pushed  
a33cbf3e5439: Pushed  
b9239390c1b8: Pushed  
82c3b921d80c: Pushed  
bc276c40b172: Pushed  
latest: digest: sha256:1d441e3d3b2236f6d6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9 size: 1986
```

[◀ 1d441e3d3b223](#) [삭제](#) [설정 안내](#) [배포](#) [새로고침](#)

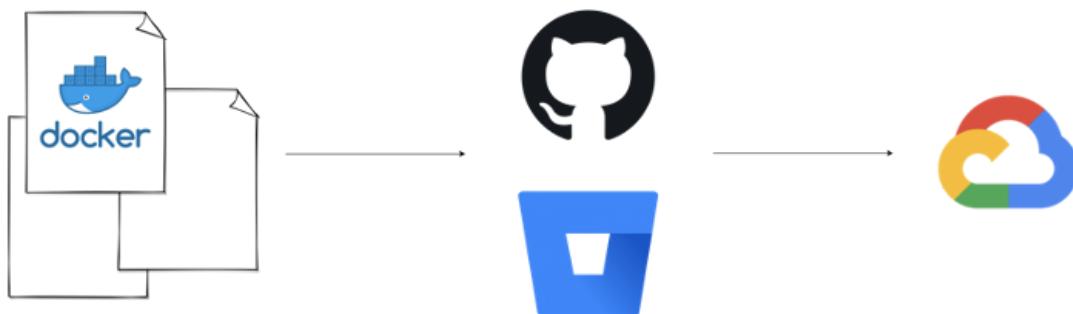
▶ asia-northeast3-docker.pkg.dev > ▶ ggingmin-docker-registry > ▶ portfolio > ▶ portfolio > sha256:1d441e3d3b2236fd6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9 ▶

개요	가져오기	매니페스트
형식	Docker	
미디어 유형	application/vnd.docker.distribution.manifest.v2+json	
프로젝트	ggingmin-docker-registry	
위치	asia-northeast3(서울)	
저장소	portfolio	
이미지	portfolio	
다이제스트	sha256:1d441e3d3b2236fd6a8b9ba0e123005d1c9add964ff7eb547f5f59d1fa843b9	
가상 크기	10.7MB	
빌드됨	2021. 8. 11. PM 3:38:59	
생성일	2021. 8. 11. PM 9:26:21	
업데이트됨	2021. 8. 11. PM 9:26:21	
태그	latest	

드디어 이미지를 공유하는 순간입니다. 명령이 성공적으로 실행되면 이미지와 같이 GCP에 생성한 저장소 경로가 나옵니다. GCP 콘솔에서도 정보들을 확인할 수 있죠. 굉장히 길고 복잡하다고 느끼실 수 있는 과정인데 진행하시느라 고생 많으셨습니다👍👍👍

5.4 GCP Cloud Build

도커 허브에서 잠깐 언급했던 'Automated Build' 기능 기억나시나요? 소스 저장소의 변동사항을 자동으로 이미지로 빌드하는 기능이었죠.



처리되는 과정은 이미지와 같습니다. Github이나 Bitbucket의 Branch에 push 된 소스를 기반으로 도커 이미지를 자동으로 빌드하는 것이죠. 아래의 단계대로 진행해봅시다.

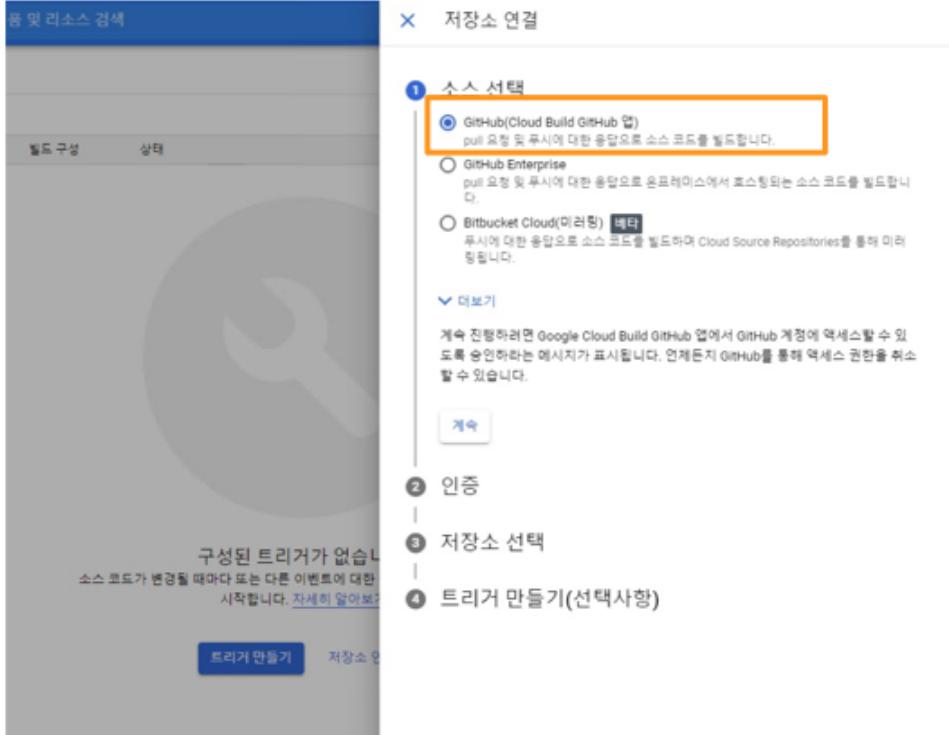
1) 트리거 생성 및 저장소 연결

The screenshot shows the GCP dashboard with the 'Cloud Build' service selected. A dropdown menu is open over the 'Triggers' option, with the 'Triggers' item highlighted by an orange box. The main pane displays a search interface for builds, with a large magnifying glass icon and placeholder text '검색어 또는 값 입력'. Below the search bar, there are columns for 'Build', 'Source', 'Timestamp', 'Commit', 'Trigger Name', 'Creation Time', and 'Duration'. At the bottom of the triggers section, there are two buttons: 'Trigger 만들기' (Create Trigger) and '저장소 연결' (Connect Repository), with '저장소 연결' also highlighted by an orange box.

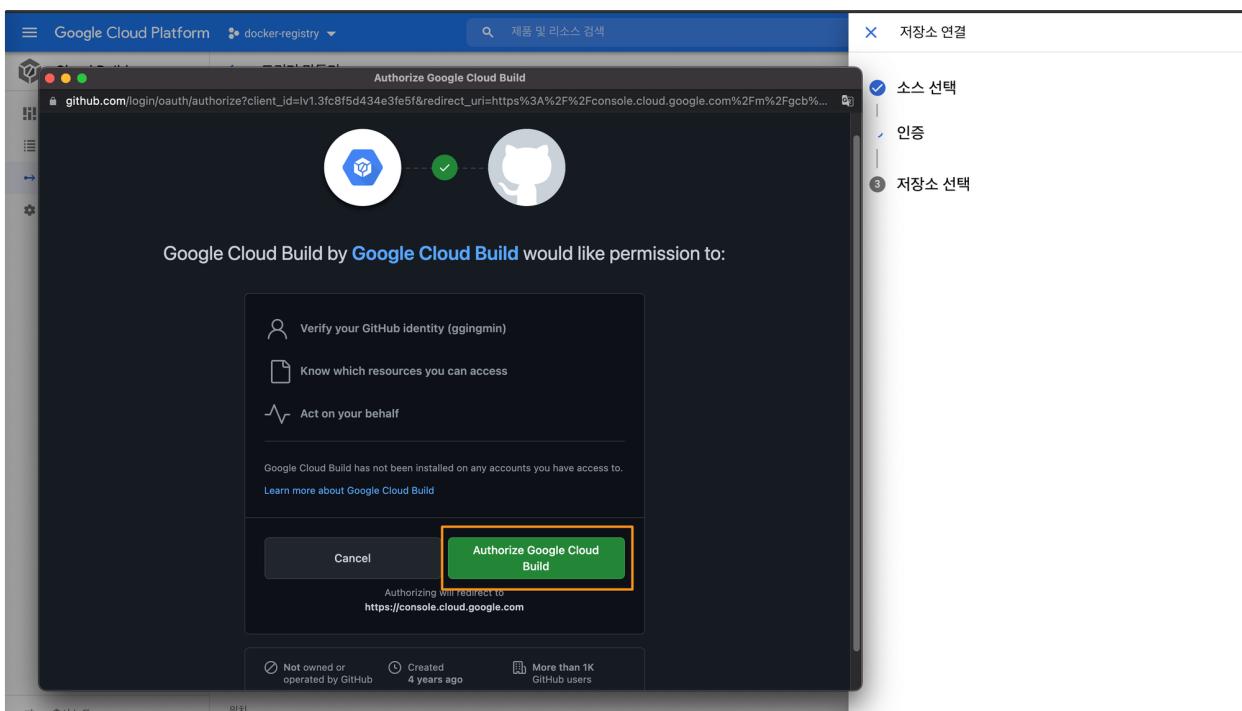
구성된 트리거가 없습니다.
소스 코드가 변경될 때마다 또는 다른 이벤트에 대한 응답으로 빌드를 자동으로 시작합니다. [자세히 알아보기](#)

트리거 만들기 저장소 연결

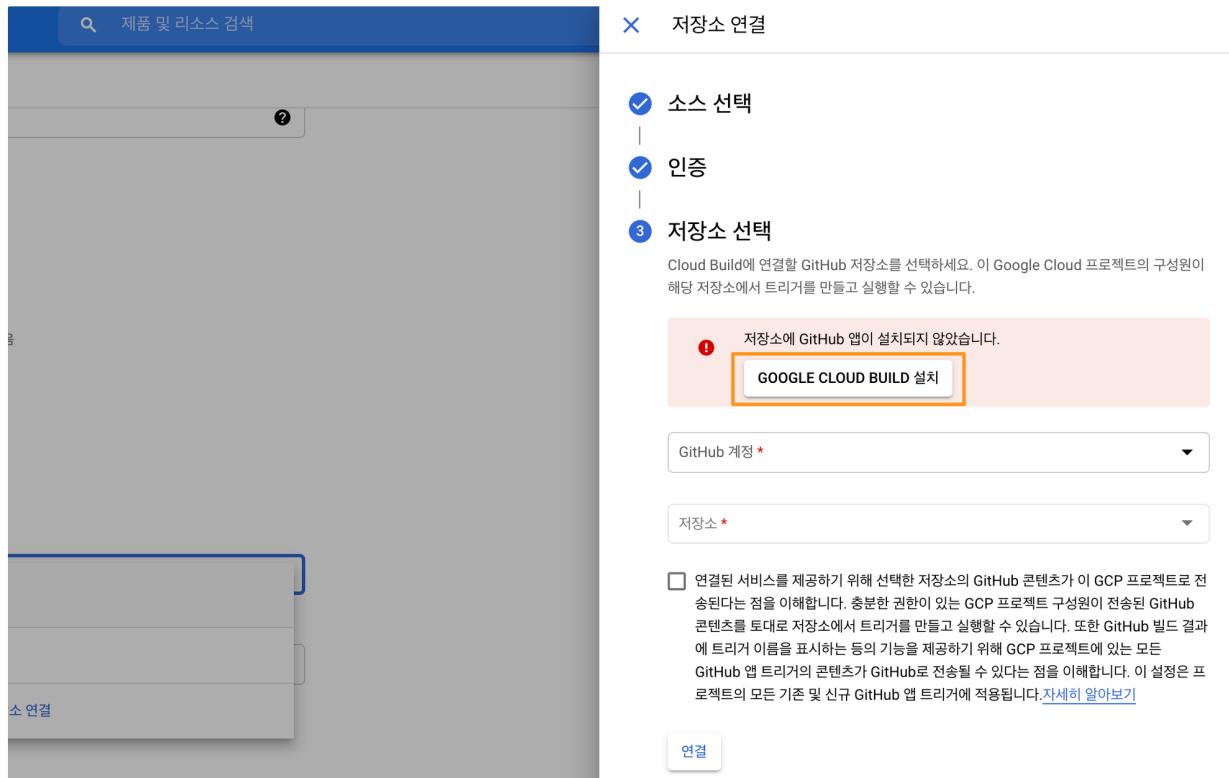
'Cloud Build' 메뉴에서 '트리거'를 누른 후 저장소 연결 버튼을 이어 누릅니다.



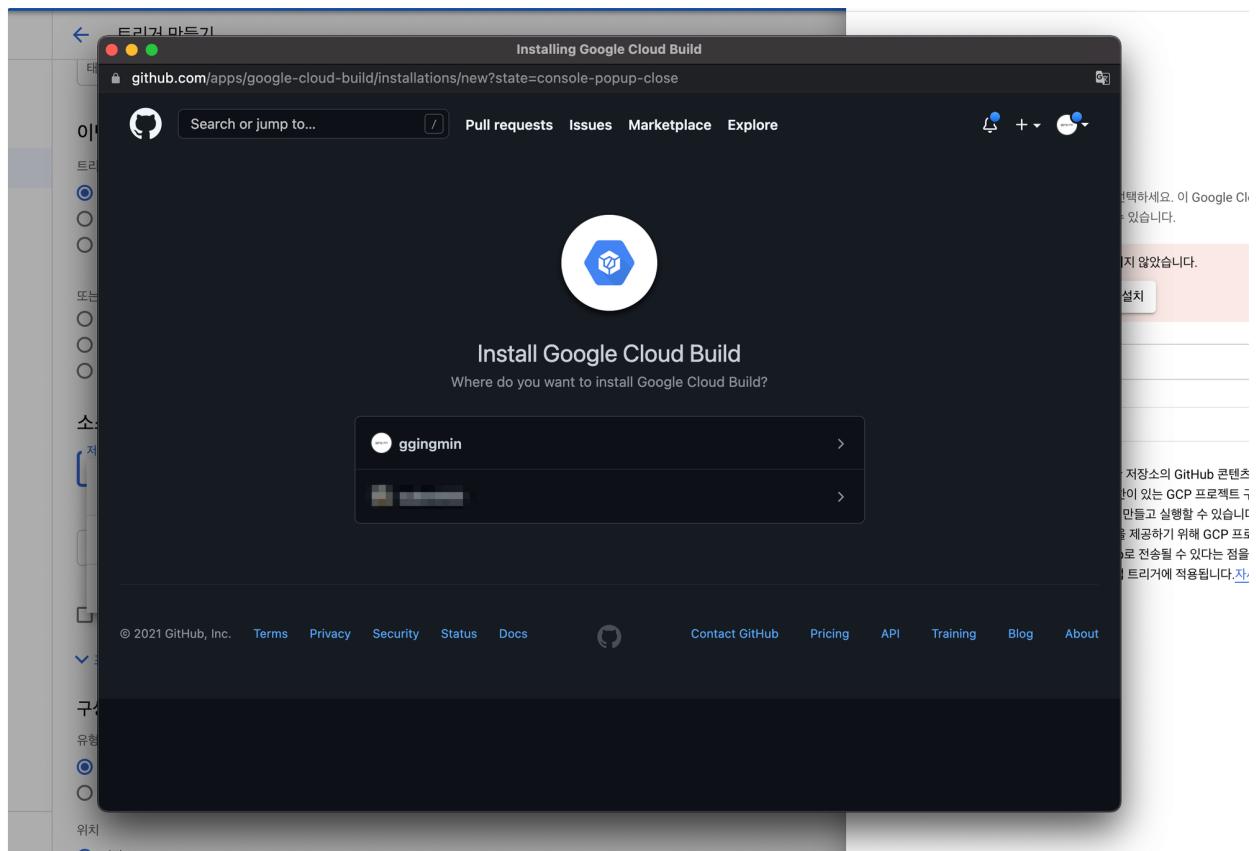
현재 Github, Github Enterprise, Bitbucket 등을 지원하고 있는데, Github를 선택하고 로그인을 진행하겠습니다.



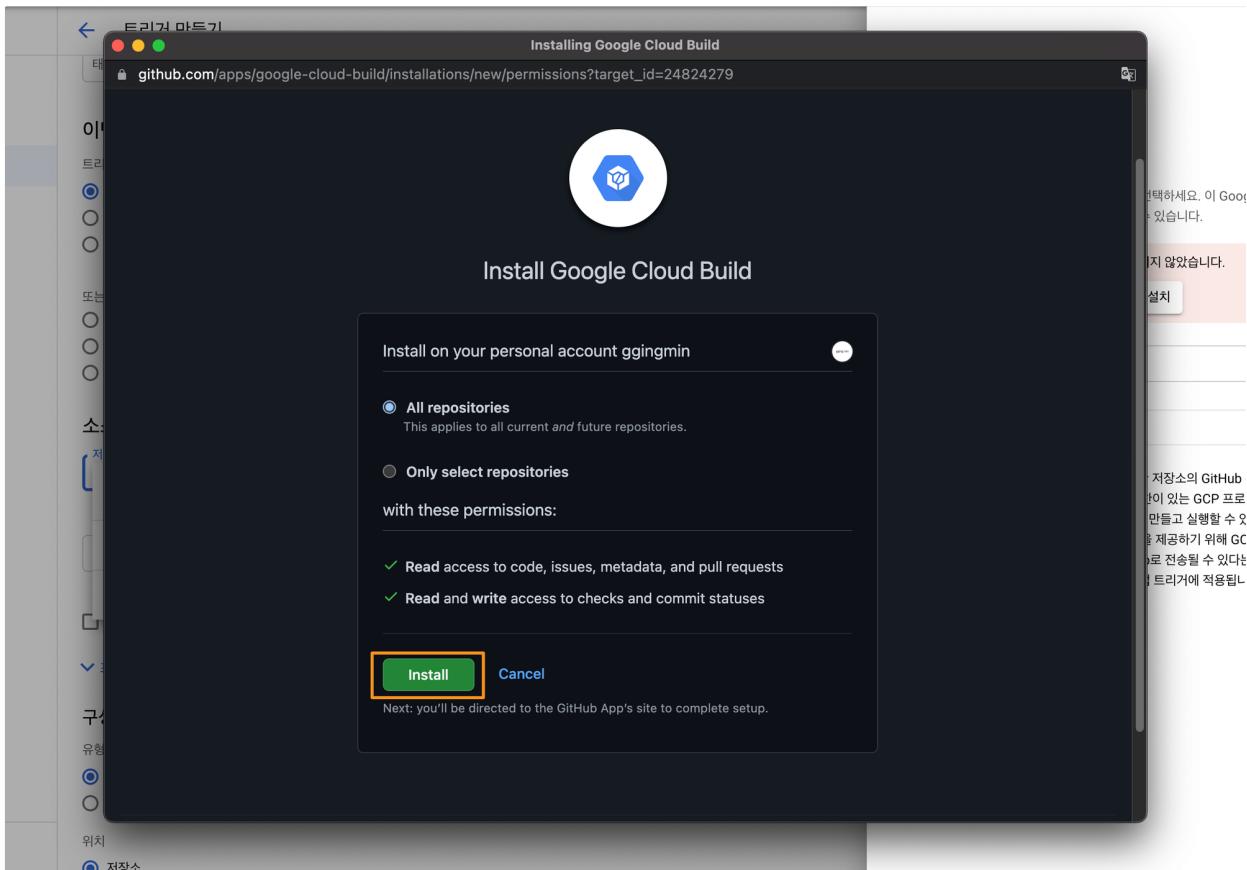
로그인 후 Google Cloud Build와 github 계정을 연동하는 여부를 묻는 창이 나오게 됩니다. **Authorize Google Cloud Build** 를 눌러 줍니다.



github의 push 내역을 가져오기 위해 **Google Cloud Build 설치**를 누릅니다.



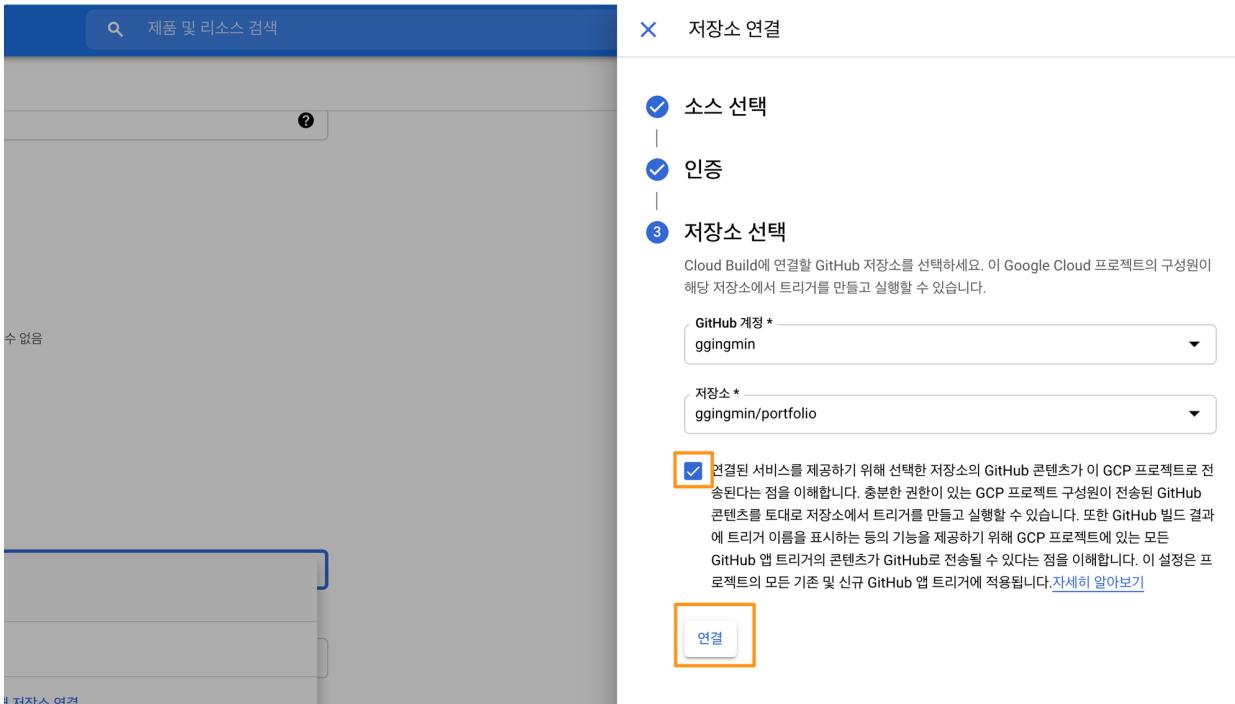
설치 대상 계정을 선택합니다.



계정에 생성된 저장소 전체에 Google Cloud Build 를 적용하려면 **All repositories**, 특정 저장소만 선택하는 경우 **Only Select repositories** 를 선택한후 **Install**을 클릭합니다.

The screenshot shows the "Repository Selection" step in the Google Cloud Platform interface. It displays a list of GitHub repositories under "Selected repositories", with "ggingmin/portfolio" checked. The "Next Step" button is visible at the bottom.

GCP의 Cloud Build 기능을 적용할 저장소를 선택하고 확인 버튼을 누릅니다.



체크박스에 체크해주시고 연결을 누릅니다.



성공적으로 로그인이 되면 여러 Github Repository 중에 이미지를 빌드할 대상을 선택하면 이미지와 같이 저장소가 연결됩니다.

2) 트리거 설정

[←](#) 트리거 만들기

이름 *
portfolio-trigger

프로젝트 내에서 고유해야 합니다.

설명

태그



이벤트

트리거를 호출하는 저장소 이벤트

- 브랜치로 푸시
- 새 태그 푸시
- pull 요청
Cloud Source Repositories에는 사용할 수 없음

또는 다음 이벤트에 응답:

- 수동 호출
- Pub/Sub 메시지
- 월록 이벤트

트리거

+ 트리거 만들기

→ 저장소 연결

저장소 관리

필터 속성 이름 또는 값 입력

이름 ↑	설명	저장소	이벤트	빌드 구성	상태	사용 설정됨	실행	⋮
portfolio-trigger	-	gggingmin/portfolio	분기에 푸시	자동 감지	?	사용 설정됨	실행	⋮

트리거의 이름과 이벤트를 설정합니다. 이벤트를 기준으로 빌드 트리거가 작동하기 때문에 상황에 알맞게 세팅 해주어야 합니다. 본 실습에서는 '브랜치로 푸시'를 선택하겠습니다. 정상적으로 트리거가 생성되었는지 콘솔에서 확인이 가능한데 이벤트에 '분기에 푸시'라고 표기되어 있습니다. 이는 'branch'가 '분기'로 번역되어 표시된 것으로 추정되며 콘솔 곳곳에 번역이 불일치 하는 부분이 눈에 띕니다.

3) 서비스 계정 권한 사용 설정

Cloud Build

대시보드

기록

트리거

설정

설정

서비스 계정

Cloud Build는 프로젝트에 연결된 Cloud Build 서비스 계정에 부여된 권한으로 빌드를 실행합니다. 서비스 계정에 추가 역할을 부여하면 Cloud Build에서 다른 GCP 서비스를 이용할 수 있습니다.

서비스 계정 이메일: 1075751016727@cloudbuild.gserviceaccount.com

GCP 서비스	역할	상태
Cloud Functions	Cloud Functions 개발자	사용 중지됨
Cloud Run	Cloud Run 관리자	사용 중지됨
App Engine	App Engine 관리자	사용 중지됨
Kubernetes Engine	Kubernetes Engine 개발자	사용 중지됨
Compute Engine	Compute 인스턴스 관리자(v1)	사용 중지됨
Firebase	Firebase 관리자	사용 중지됨
Cloud KMS	클라우드 KMS CryptoKey 복호화	사용 중지됨
보안 비밀 관리자	보안 비밀 관리자 보안 비밀 접근자	사용 중지됨
서비스 계정	서비스 계정 사용자	사용 설정됨
Cloud Build	Cloud Build WorkerPool 사용자	사용 설정됨

여기에서 나열되지 않은 역할은 IAM 섹션에서 관리할 수 있습니다.

Cloud Build 설정의 서비스 계정 탭에서 표시된 두 항목을 '사용 설정됨' 상태로 변경해줍니다.

4) Github에 소스 푸시 후 빌드 확인



The screenshot shows the Google Cloud Build dashboard. At the top, there's a search bar and a filter dropdown. Below that, a table displays a single build log:

성공: ggingmin/portfolio - portfolio-trigger	
최근 빌드 21. 8. 11. 오후 10:29	지속 시간 00:00:20
트리거 설정 -	소스 ggingmin/portfolio
커밋 d3ab74d	
빌드 기록 최신 모두 보기	평균 지속 시간 00:00:18
	통과 - 실패 비율 100% - 0%

Github에 소스를 push 하면 자동으로 트리거가 실행되고 이미지가 빌드됩니다. 빌드와 관련한 성공/실패 로그는 GCP 대시보드에서 확인하실 수 있습니다.