

Registry

Image Registry

❖ Docker Image Registry

✓ 개요

- Docker 이미지 레지스트리(Docker Image Registry)는 Docker 이미지를 저장하고 배포하는 중앙 저장소
- 개발자가 만든 컨테이너 이미지를 올려두고(Push) 필요할 때 어디서든 내려받을 수(Pull) 있는 이미지 전용 클라우드/서버

✓ 레지스트리는 크게 공개 여부에 따라 두 가지로 나뉨

- Public Registry: 누구나 이미지를 올리고 내려받을 수 있는 것으로 오픈 소스 프로젝트나 공통 라이브러리 배포에 적합한데 Docker Hub가 가장 대표적인 기본 레지스트리
- Private Registry: 특정 조직이나 개인만 접근할 수 있는데 기업의 내부 소스 코드나 보안이 중요한 이미지를 관리할 때 사용

✓ 대표적인 레지스트리

- Docker Hub: 가장 방대한 생태계, 기본 설정 없이 바로 사용 가능
- Amazon ECR: AWS 환경과 강력한 통합, 보안 및 IAM 권한 제어 우수
- Google Artifact Registry: GCP 환경 최적화, 취약점 스캔 기능 제공
- GitHub Packages: GitHub Action과 연동하여 CI/CD 파이프라인 구축 용이
- Quay.io: Red Hat 제공, 이미지 빌드 및 보안 스캔 기능 강화

Image Registry

❖ Docker Image Registry

- ✓ 직접 구축(Self-hosted): 보안이나 비용 문제로 외부 서비스를 쓰기 어렵다면 직접 레지스트리를 운영할 수도 있음
 - Docker Registry: Docker에서 공식 제공하는 오픈소스 레지스트리 서버(매우 가벼움)
 - Harbor: 기업용으로 설계된 오픈소스 레지스트리로 권한 관리, 이미지 복제, 보안 스캔 등 강력한 관리 UI를 제공
- ✓ 동작 과정
 - Build: 내 컴퓨터(Local)에서 Docker 이미지를 생성
 - Tag: 이미지에 레지스트리 주소 정보를 포함한 이름을 붙임 - `docker tag my-app:v1 username/my-app:v1`
 - Push: 레지스트리로 이미지를 업로드 - `docker push username/my-app:v1`
 - Pull: 다른 서버나 환경에서 이미지를 다운로드 - `docker pull username/my-app:v1`



Image Registry

❖ Docker Hub 활용

✓ Docker Hub 로그인 관련 명령

- `docker login(echo "내_비밀번호" | docker login -u "내_아이디" --password-stdin)`
- `docker logout`

✓ 로그인 흐름

- Client: 사용자가 ID/PW를 입력
- Registry: 레지스트리 서버가 정보를 확인하고 Auth Token을 발급
- Local Storage: 발급된 토큰(또는 인코딩된 정보)이 로컬 설정 파일에 저장되어, 이후 push나 pull 시 자동으로 사용

✓ Image 이름 과 태그

- docker의 태그는 Registry에 업로드를 상정한 Image 이름
- 태그는 Registry의 주소와 버전 표기를 추가해 정식 명칭으로 생성
adamsoft.com(Registry 주소)/adam(레포지토리 이름):13(버전)
- 태그 이름 부여
`docker tag 원본_Image_이름[:버전] 참조_Image_이름[:버전]`
- docker Hub에 업로드
`docker push 이름`

Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 public repository에 이미지 업로드

● Docker 이미지 생성

◆ 패키지 설치: `pip3 install flask --break-system-packages`

◆ 소스 코드 작성: `app.py`

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
@app.route('/')
```

```
def hello():
```

```
    return "This is my Python Web Server."
```

```
if __name__ == '__main__':
```

```
    app.run(host='0.0.0.0', port=5000)
```

◆ 웹 서버 실행: `python3 app.py`

Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 public repository에 이미지 업로드

- Docker 이미지 생성

- ◆ Dockerfile 생성

- # 1. 베이스 이미지 설정 (파이썬 3.9 슬림 버전)

- FROM python:3.9-slim

- # 2. 작업 디렉토리 설정

- WORKDIR /app

- # 3. 필요한 패키지 설치 (Flask)

- RUN pip install flask

- # 4. 현재 폴더의 소스코드를 컨테이너 내부로 복사

- COPY . .

- # 5. 서버 실행 (5000번 포트)

- CMD ["python", "app.py"]

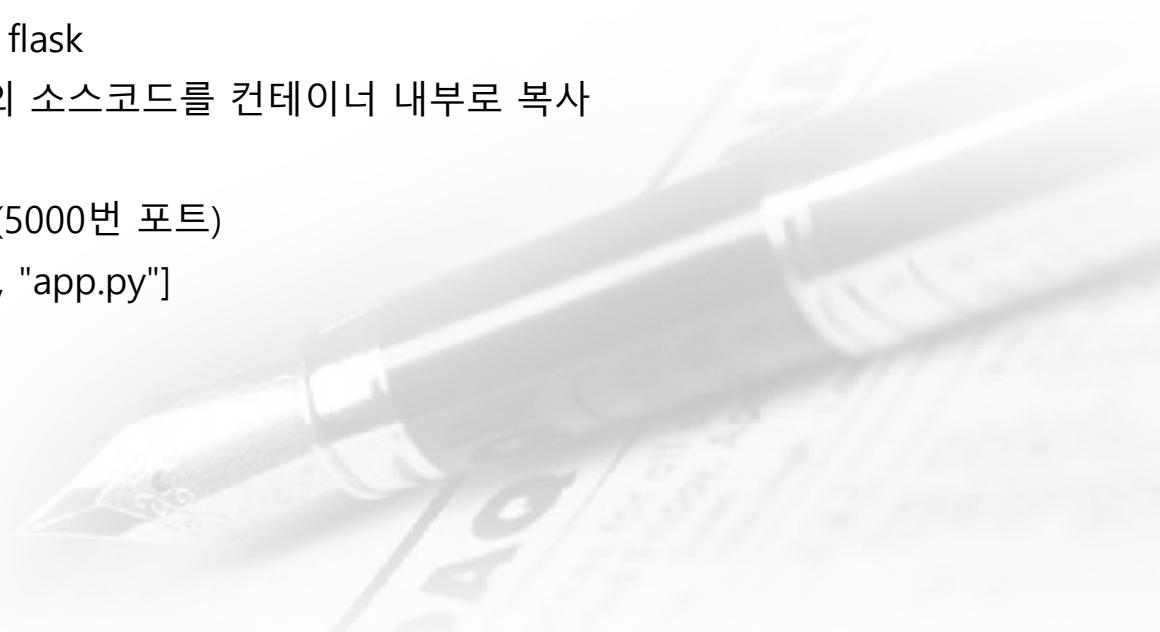


Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 public repository에 이미지 업로드

- Docker 이미지 생성

- ◆ 이미지 빌드

- ```
docker build -t my-python-app .
```

- ◆ 컨테이너 실행 테스트

- ```
docker run -d -p 5000:5000 my-python-app
```



Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 public repository에 이미지 업로드

- Docker Hub에 레포지토리 생성: ggnagpae1/my-python-app
- 이미지 태그 설정: `docker tag my-python-app ggnagpae1/my-python-app:v1.0`
- Push: `docker push ggnagpae1/my-python-app:v1.0`
- Test: `docker run -d -p 8080:5000 ggnagpae1/my-python-app:v1.0`

The screenshot shows the Docker Hub interface for a repository named 'ggnagpae1 / myhttpd'. The page includes a search bar, navigation links (Explore, Repositories, Organizations, Help), and an 'Upgrade' button. The repository page has tabs for General, Tags, Builds, Collaborators, Webhooks, and Settings. A description box at the top prompts the user to add a short description. Below this, the repository name 'ggnagpae1 / myhttpd' is displayed, along with a description field that currently says 'This repository does not have a description'. A 'Last pushed' timestamp of '5 minutes ago' is shown. On the right, 'Docker commands' are provided, including a 'Public View' button and a code block for pushing a new tag: `docker push ggnagpae1/myhttpd:tagname`. At the bottom, there is a 'Tags' section showing a table with one tag, '3.0', of type 'Image', pushed 5 minutes ago. An 'Automated Builds' section is also visible, explaining how to connect to GitHub or Bitbucket for automatic builds.

docker hub Search Docker Hub Explore Repositories Organizations Help Upgrade ggnagpae1

ggnagpae1 Repositories myhttpd General Using 0 of 1 private repositories. Get more

General Tags Builds Collaborators Webhooks Settings

Add a short description for this repository. Update
The short description is used to make your content on Docker Hub and to search engines. It's visible to users in search results.

ggnagpae1 / myhttpd

Description
This repository does not have a description

Last pushed: 5 minutes ago

Docker commands
To push a new tag to this repository:
`docker push ggnagpae1/myhttpd:tagname` Public View

Tags
This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
3.0		Image	4 minutes ago	5 minutes ago

See all Go to Advanced Image Management

Automated Builds
Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.
Available with Pro, Team and Business subscriptions. [Read more about automated builds](#)

Upgrade

Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 public repository의 이미지를 kubernetes에서 배포

- deployment.yaml

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: flask-deployment
```

```
  labels:
```

```
    app: flask-web
```

```
spec:
```

```
  replicas: 3 # 3개의 파드(Pod)를 띄워 부하 분산 및 가용성 확보
```

```
  selector:
```

```
    matchLabels:
```

```
      app: flask-web
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: flask-web
```

```
    spec:
```

```
      containers:
```

```
        - name: flask-container
```

```
          image: ggnagpae1/my-python-app:v1.0 # Docker 허브에 올린 이미지 주소
```

```
          ports:
```

```
            - containerPort: 5000
```

Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 public repository의 이미지를 kubernetes에서 배포

- service.yaml

apiVersion: v1

kind: Service

metadata:

name: flask-service

spec:

selector:

app: flask-web

ports:

- protocol: TCP

port: 80 # 서비스가 외부(또는 Ingress)로부터 받는 포트

targetPort: 5000 # 컨테이너 내부에서 실행 중인 포트

type: ClusterIP # 클러스터 내부용 IP만 할당 (Ingress를 통해 노출할 예정)

Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 public repository의 이미지를 kubernetes에서 배포

- ingress.yaml

```
apiVersion: networking.k8s.io/v1
```

```
kind: Ingress
```

```
metadata:
```

```
  name: ingress-nginx
```

```
  annotations:
```

```
    nginx.ingress.kubernetes.io/rewrite-target: /
```

```
    kubernetes.io/ingress.class: "nginx"
```

```
spec:
```

```
  rules:
```

```
  - host: "my-flask-app.com"
```

```
    http:
```

```
      paths:
```

```
      - pathType: Prefix
```

```
        path: /
```

```
        backend:
```

```
          service:
```

```
            name: flask-service
```

```
            port:
```

```
              number: 80
```

Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 public repository의 이미지를 kubernetes에서 배포

- 리소스 배포

- ◆ `kubectl apply -f deployment.yaml`

- ◆ `kubectl apply -f service.yaml`

- ◆ `kubectl apply -f ingress.yaml`

- 리소스 확인

- ◆ `kubectl get pods,services`

NAME	READY	STATUS	RESTARTS	AGE
pod/flask-deployment-6ffbbbf6f7-7pzh4	1/1	Running	0	52m
pod/flask-deployment-6ffbbbf6f7-cx5np	1/1	Running	0	52m
pod/flask-deployment-6ffbbbf6f7-m4v92	1/1	Running	0	52m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
service/flask-service	ClusterIP	10.109.115.43	<none>	80/TCP
AGE				
50m				
service/kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
3d7h				

Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 public repository의 이미지를 kubernetes에서 배포

- End Points 확인

- ◆ kubectl get endpoints

NAME	ENDPOINTS	AGE
flask-service	10.244.1.76:5000,10.244.2.75:5000,10.244.3.38:5000	12m
kubernetes	10.0.2.15:6443	3d7h

- 인그레스 확인

- ◆ kubectl get ingress

NAME	CLASS	HOSTS	ADDRESS	PORTS	AGE
ingress-nginx	<none>	my-flask-app.com	10.100.254.224	80	38m



Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 private repository에 이미지 업로드

- Docker Hub에 레포지토리 생성: ggnagpae1/private-python-app
- 이미지 태그 설정: `docker tag my-python-app ggnagpae1/private-python-app:v1.0`
- Push: `docker push ggnagpae1/private-python-app:v1.0`
- Test: `docker run -d -p 8080:5000 ggnagpae1/private-python-app:v1.0`

[Repositories](#) / [Create](#)

Create repository

Repository Name *

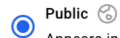


Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

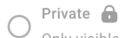
Visibility

Using 1 of 1 private repositories. [Get more](#)



Public

Appears in Docker Hub search results



Private

Only visible to you

Cancel

Create

Using 1 of 1 private repositories. [Get more](#)

Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 private repository의 이미지를 kubernetes에서 배포

- deployment.yaml

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: flask-deployment
```

```
  labels:
```

```
    app: flask-web
```

```
spec:
```

```
  replicas: 3 # 3개의 파드(Pod)를 띄워 부하 분산 및 가용성 확보
```

```
  selector:
```

```
    matchLabels:
```

```
      app: flask-web
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: flask-web
```

```
    spec:
```

```
      containers:
```

```
        - name: flask-container
```

```
          image: ggnagpae1/private-python-app:v1.0 # Docker 허브에 올린 이미지 주소
```

```
          ports:
```

```
            - containerPort: 5000
```

Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 private repository의 이미지를 kubernetes에서 배포

- 배포: `kubectl apply -f deployment.yaml`
- 파드 결과 확인: `kubectl get pods`

NAME	READY	STATUS	RESTARTS	AGE
flask-deployment-96578cc8b-6vmfb	0/1	ErrImagePull	0	4s
flask-deployment-96578cc8b-78vvc	0/1	ErrImagePull	0	4s
flask-deployment-96578cc8b-crdb4	0/1	ErrImagePull	0	4s



Image Registry

❖ Docker Hub 활용

- ✓ Docker Hub의 private repository의 이미지를 kubernetes에서 배포
 - Docker 레지스트리 인증용 시크릿 생성: `kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-username=<내-아이디> --docker-password=<내-액세스-토큰> --docker-email=<내-이메일>`
 - deployment.yaml 파일에 추가

spec:

replicas: 3 # 3개의 파드(Pod)를 띄워 부하 분산 및 가용성 확보

selector:

matchLabels:

app: flask-web

template:

metadata:

labels:

app: flask-web

spec:

imagePullSecrets:

- name: regcred # 위에서 생성한 Secret 이름

containers:

Image Registry

❖ Docker Hub 활용

✓ Docker Hub의 private repository의 이미지를 kubernetes에서 배포

- 배포: `kubectl apply -f deployment.yaml`
- 파드 결과 확인: `kubectl get pods`

NAME	READY	STATUS	RESTARTS	AGE
flask-deployment-6b9c866f58-5r4p8	1/1	Running	0	3m34s
flask-deployment-6b9c866f58-dmqn9	1/1	Running	0	3m39s
flask-deployment-6b9c866f58-zjv7s	1/1	Running	0	3m36s



Image Registry

❖ Docker Registry

✓ 개요

- Docker 허브에서 프라이빗 레지스트리 구성을 위한 registry 이미지를 제공하는데 이 이미지를 컨테이너로 실행하고 그 안에 이미지를 로컬에 저장하는 방식
- 단순 텍스트 방식만 지원하기 때문에 웹 기반의 검색을 하려면 GUI 인터페이스를 제공하는 다른 컨테이너와 결합해서 사용할 수 있음



Image Registry

❖ Docker Registry

✓ Registry 생성

- 생성: `docker run -d --name registry -p 5000:5000 registry:2`
- 외부에서 확인: `curl http://3.36.69.129:5000/v2/`
현재는 이미지가 없어서 { }



Image Registry

❖ Docker Registry

✓ GUI 연결

```
docker run -it -d -p 접속할포트번호:8080 --name registry-web --link registry -e  
REGISTRY_URL=http://IP:5000/v2 -e REGISTRY_NAME=IP:5000 --restart=always  
hyper/docker-registry-web
```

Web Registry

Home

Repositories

Registry

3.36.69.129:5000

Repository	Tags
------------	------

Image Registry

❖ registry:2를 이용한 Private Registry

✓ Push – Error 발생

- `docker image tag ggnagpae1/private-python-app:v1.0 10.0.2.200:5000/pythonapp:v1`
- `docker push 10.0.2.200:5000/pythonapp:v1`

failed to do request: Head

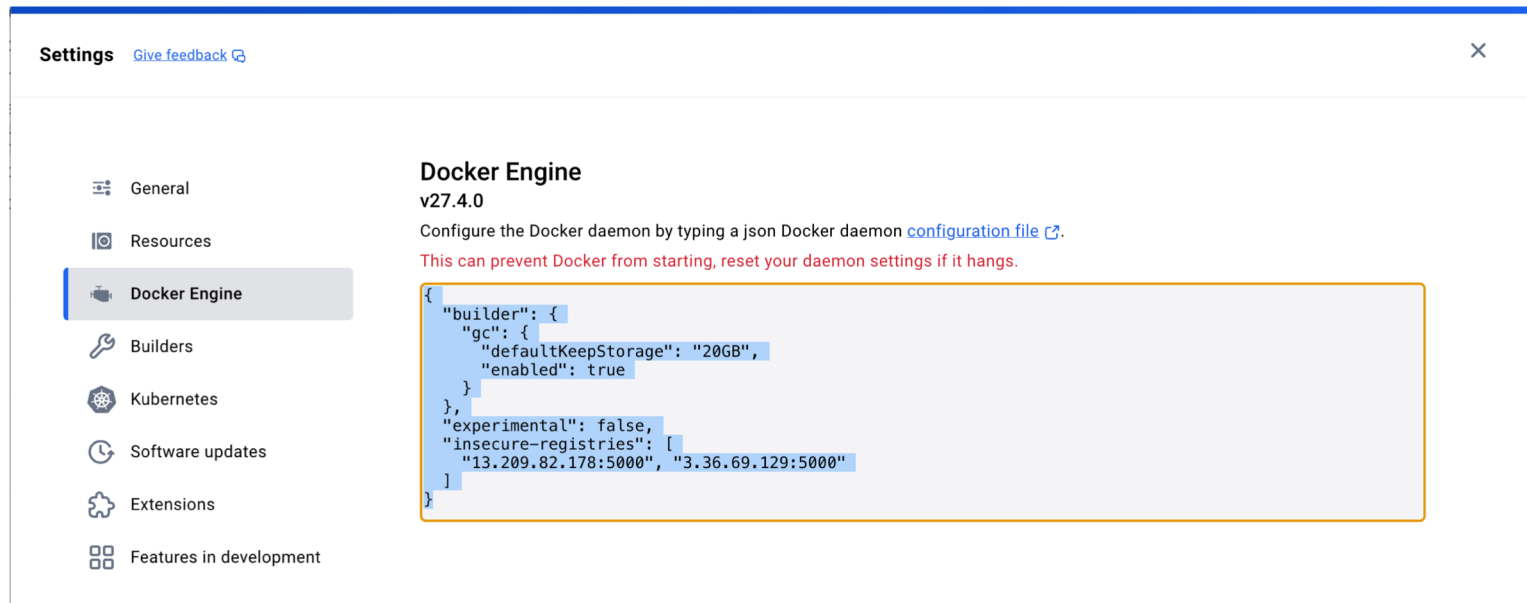
"https://10.0.2.200:5000/v2/pythonapp/blobs/sha256:0d85b9734e5dfa5b83c3623292b4159da90e9e4d16c8d7bdb728c67ded5cf8a5": http: server gave HTTP response to HTTPS client

- ◆ 이 레지스트리 컨테이너는 이미지를 푸시하고 내려받기 위해 보안 프로토콜인 HTTPS 대신 비보안 프로토콜인 HTTP를 사용
- ◆ Docker의 기본 설정에서는 비보안 프로토콜이 적용된 레지스트리를 사용할 수 없게 돼 있음
- ◆ 비보안 레지스트리를 사용하려면 로컬 컴퓨터의 레지스트리를 비보안 레지스트리 허용 목록에 추가해야 함
- ◆ 이미지 레이어의 저장 경로, Docker API가 주시하는 포트 번호, 허용된 비보안 레지스트리 목록 등 Docker 엔진의 모든 설정은 `daemon.json`이라는 이름의 JSON 포맷으로 된 설정 파일에 들어 있음
- ◆ 이 파일은 윈도우에서는 `C:\Program Data\docker\config`. 리눅스에서는 `/etc/docker`에 위치

Image Registry

❖ registry:2를 이용한 Private Registry

- ✓ 보안이 적용되지 않은 레지스트리(http) 추가 후 Docker 재시작(sudo systemctl restart docker)



- docker push 10.0.2.200:5000/pythonapp:v1
- curl -X GET http://10.0.2.200:5000/v2/_catalog

Image Registry

❖ registry:2를 이용한 Private Registry

✓ Kubernetes 에서 사용

● 모든 노드 컴퓨터에서 작업

◆ /etc/containerd/config.toml 파일에 내용 추가

```
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."10.0.2.200:5000"]  
  endpoint = ["http://10.0.2.200:5000"]
```

◆ containerd 재시작

```
sudo systemctl restart containerd
```



Image Registry

❖ registry:2를 이용한 Private Registry

✓ Kubernetes 에서 사용

● 리소스 배포 파일: deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: internal-registry-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: internal-app
  template:
    metadata:
      labels:
        app: internal-app
    spec:
      containers:
        - name: my-app-container
          # 레지스트리 주소와 포트를 이미지 이름 앞에 붙여줍니다.
          image: 10.0.2.200:5000/pythonapp:v1
          ports:
            - containerPort: 5000
```