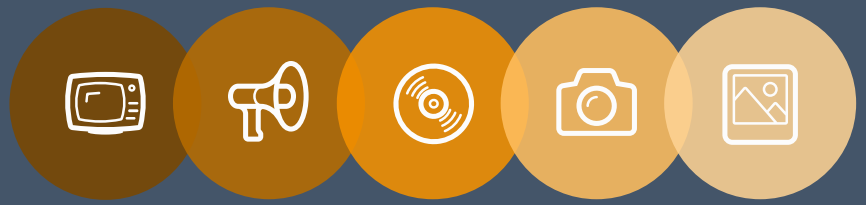
카카오 클라우드 엔지니어 양성과정 1기



도커 토이 프로젝트

(도커 스웜 모드 기반의 인프라 구성)

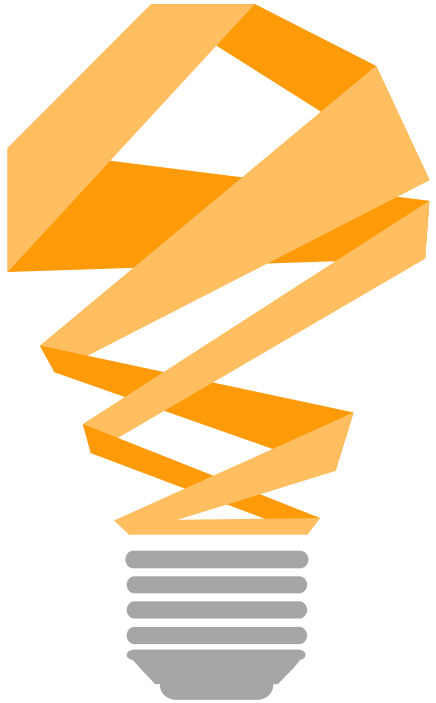
연수연 (개인프로젝트)

고용노동부 KOREATECH 직업능력심사평가원 kakao RAPA 한국전파진흥협회 Korea Radio Promotion Association

목차

프로젝트 배경

프로젝트 주제 및 선정 배경



01 도커 스웜 모드를 공부하며 클라우드 환경을 제공하기 위해서는 전체적인 **인프라를 이해하고 구상**하는 능력이 필요하다는 것을 느끼게 되었고, 도커를 이용하여 인프라 구성에 대하여 스스로 생각해보고 구현을 해야겠다는 생각이 들게 됨

02 또한, 도커를 학습하며 배운 내용을 **복습**하고, 그에서 더 나아가 기존 인프라 환경에서 **추가적인 기능**을 추가하여 안정적이고 편리한 환경을 만들고자 함

03 따라서 L4, L7 로드밸런서를 사용해 사용자가 웹 사이트를 접속했을 때 적절히 **로드밸런싱** 되는 환경을 구상하게 되었고, 이에 추가적으로 **모니터링 기능 및 자동화 기능**을 추가하여 해당 프로젝트를 구성하려 함

프로젝트 배경 (Cont.)

프로젝트 목적

01 해당 프로젝트에서는 L7 로드 밸런서와 L4 로드 밸런서를 이용해 Manager노드에서 Worker 노드에 웹 컨테이너를 배포하는 **프로세스**와 **로드 밸런서**의 역할을 이해한다.

02 클라우드에서 이용하는 사설 저장소 Harbor를 통해 이미지를 관리하는 방법을 습득한다.

03 **모니터링** 도구를 사용하여 컨테이너 및 노드의 상태 정보를 시각화 하여 보는 방법을 터득한다.

04 Github에서 파일을 clone하여 매일 같은 시간에 파일 및 도커 이미지가 업데이트 되는 과정을 통해 클라우드에서 중요한 개념 중 하나 인 CI/CD의 과정에 대하여 이해한다.

05 통합적으로 이 모든 프로세스 및 전체 인프라 환경 구성에 대해 이해하고 구현할 수 있는 능력을 기른다.

프로젝트 배경 (Cont.)

프로젝트 개요

컨셉

클라우드 인프라 환경 구성을 학습한다.
환경을 구상하고, 사용자의 입장에서 구성한 내용들을 살펴 볼 예정이다.

훈련 관련성

훈련에서 학습한 **도커** 및 **도커 스웜 모드**를 이용하여 도커 이미지를 생성한다.
리눅스에서 학습한 CRON을 이용하여 인덱스를 Github에서 특정 시간마다 불러와 도커 이미지를 자동으로 배포할 수 있도록 한다.
학습에서 배운 저장소가 아닌 Harbor를 이용해 이미지를 관리할 수 있도록 한다.
훈련에서 학습한 **모니터링**을 더 자세히 학습 해보기 위해 컨테이너를 모니터링 할 수 있는 cAdvisor, 노드를 모니터링 하기 위한 Node-exporter을 워커 노드에 설치하고, 매니저 노드에서 Prometheus와 Grafana를 통해 모니터링 결과를 시각화하여 볼 수 있다.

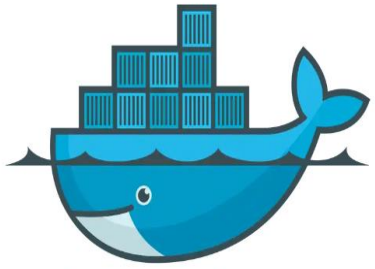
프로젝트 배경 (Cont.)

개발 환경

Linux

Grafana

Prometheus

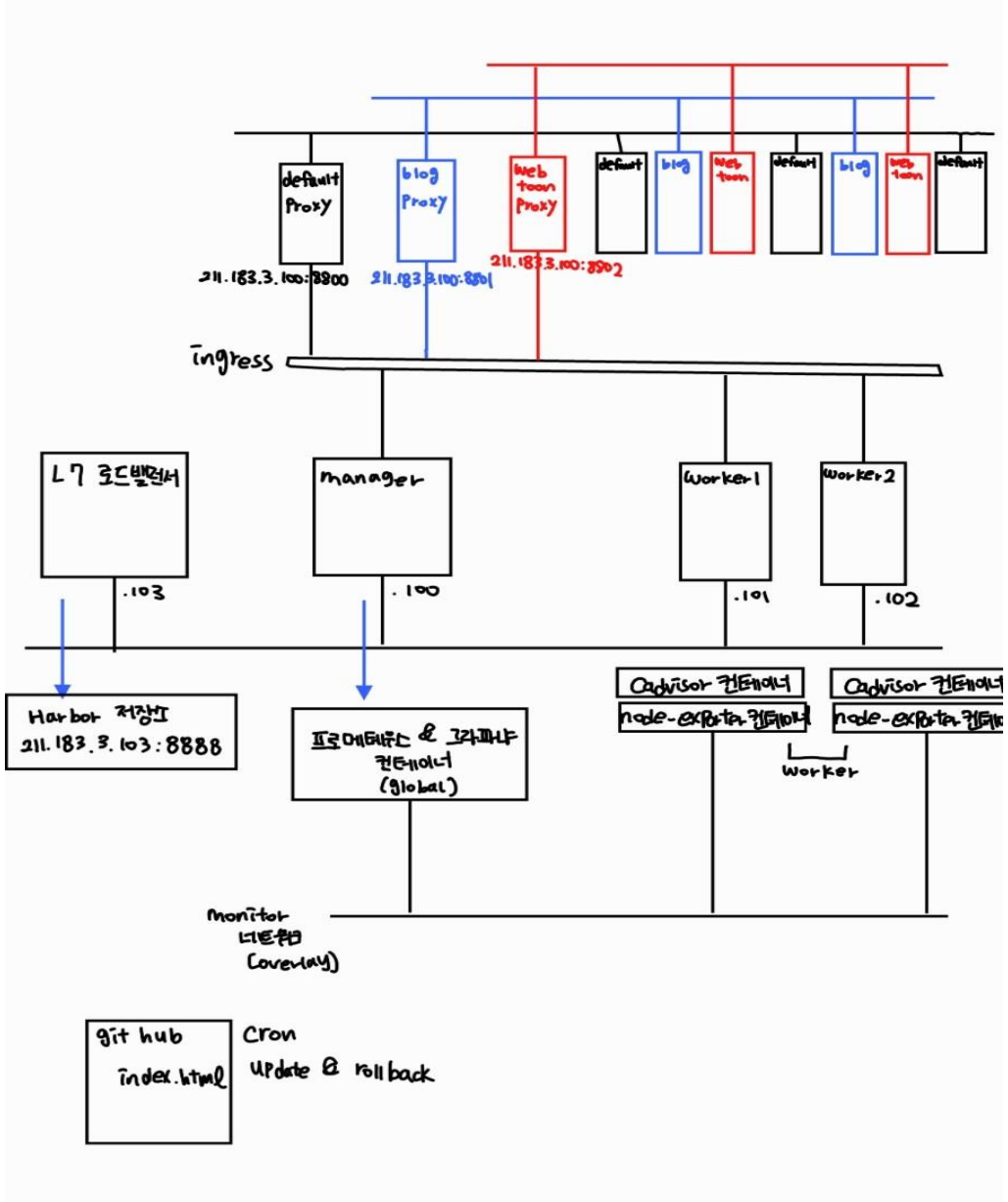


docker

Node exporter

cAdvisor

프로젝트 배경 (Cont.)



프로젝트 구조

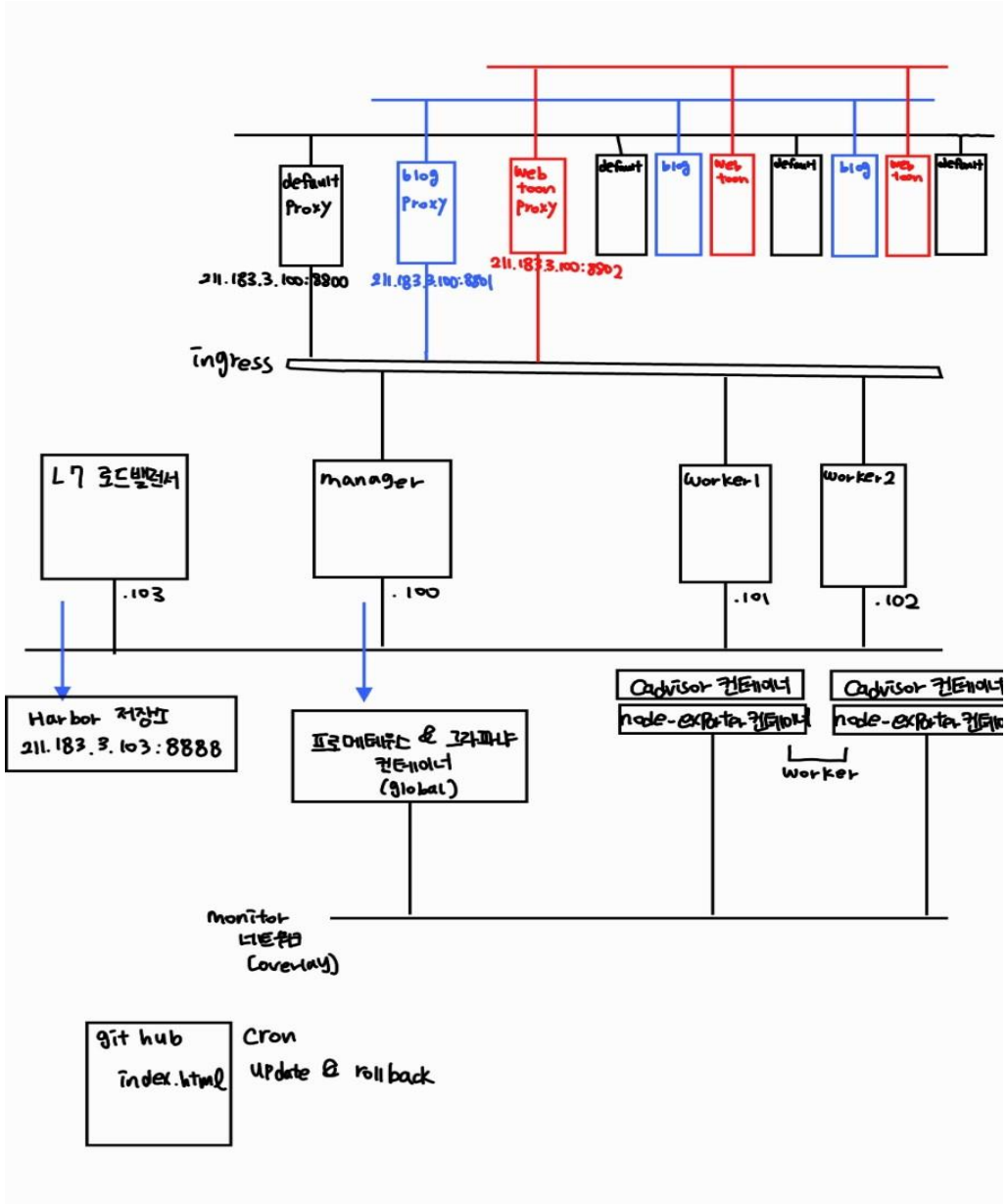
L7 로드밸런서

- 사용자가 /, /blog, /webtoon 으로 들어왔을 때 각각의 Proxy(L4 **로드밸런서**)로 보내주는 역할을 수행한다.
- 또한, 해당 노드의 8888**번 포트**로 들어왔을 경우 Harbor **저장소** 페이지에 접속할 수 있도록 구성했다.

L4 로드밸런서

- Proxy에서 각각의 웹 컨테이너에게 로드밸런서 역할을 수행하여 컨테이너로 보내 웹페이지가 보이도록 한다.

프로젝트 배경 (Cont.)



프로젝트 구조

매니저 노드

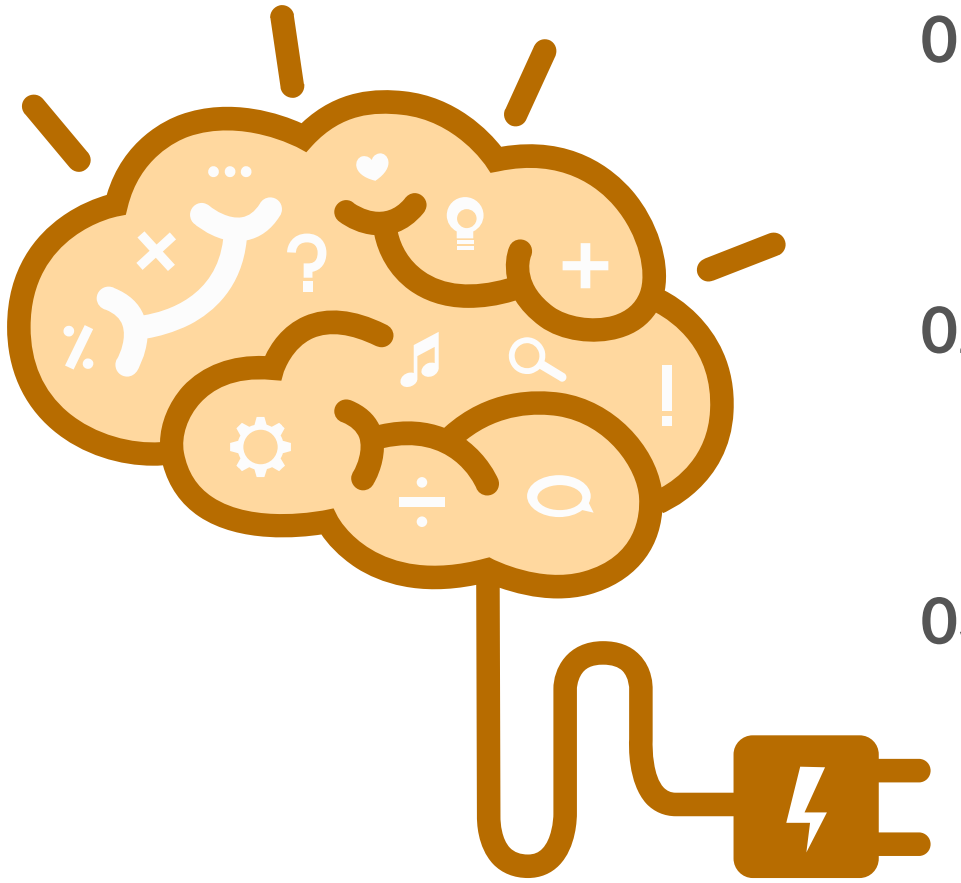
- 워커 노드에게 명령을 내리는 역할을 해 워커 노드에 컨테이너를 배포할 수 있도록 한다.
- 프로메테우스와 그라파냐와 같은 모니터링 도구를 이용해 워커 노드의 상태와 컨테이너 상태를 **모니터링** 할 수 있다.

워커 노드

- 매니저 노드에서 내린 명령을 받아 컨테이너를 배포한다. 이때, 이미지는 Github에서 받아오며 **주기적으로 자동 업데이트**가 가능하며, 이전 상태로 **롤백**이 가능하다.

프로젝트 배경 (Cont.)

기대효과



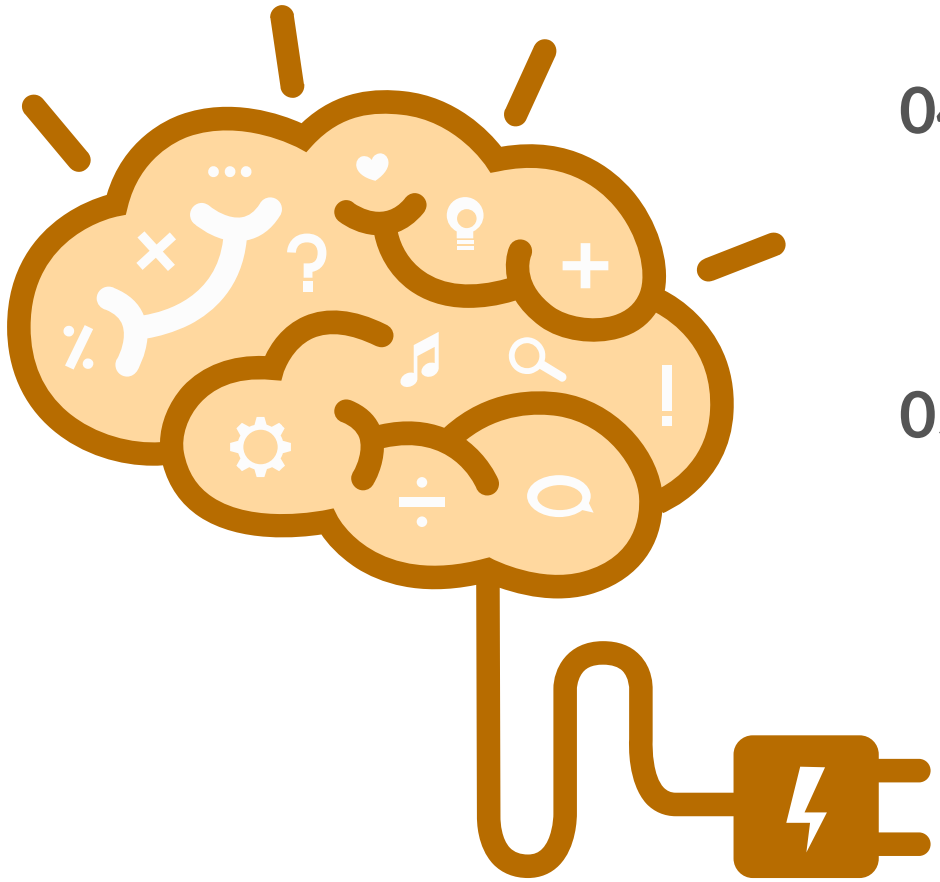
01 해당 프로젝트를 통해 도커 스웜 모드에 대한 이해 및 클라우드 인프라에 대한 생각의 폭을 넓히고, 구상한 환경을 직접 구축해 제공할 수 있는 능력을 기를 수 있다.

02 도커 스웜 모드 환경에 대해 이해하며 이후에 배우게 될 쿠버네티스를 학습하는 데 더욱 깊게 이해 할 수 있다.

03 프로젝트를 시연하며 구축한 환경이 사용자의 입장에서 어떻게 제공되는 지 경험할 수 있다.

프로젝트 배경 (Cont.)

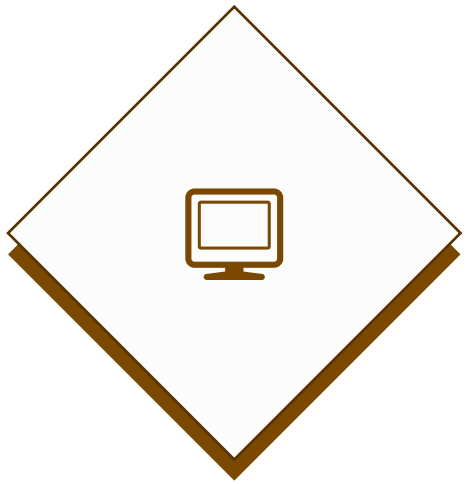
기대효과



04 다양한 모니터링 도구를 사용하며 컨테이너와 노드를 모니터링 해 관리할 수 있다.

05 자동화를 통해 CI/CD 환경을 이해할 수 있고, 추후 Jenkins를 배우고 이를 업데이트 할 예정으로 프로젝트를 이후 유지보수를 통해 좋은 서비스를 구축 할 수 있다.

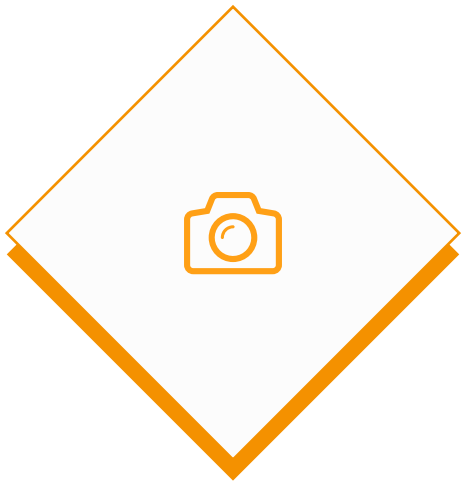
프로젝트 팀 구성 및 역할
(개인 : 연수연)



개발 환경 설치 및 구성

기능 구현

- L7 로드 밸런서
- 기능 별 Proxy 구성
- Harbor 사설 저장소
- 모니터링 도구
- CRON 자동화



시연영상 촬영

슬라이드 작성

프로젝트 수행 절차 및 방법

구분	기간	활동	비고
프로젝트 주제 선정 및 기획	08/29(월) 오후	- 프로젝트 주제 선정 - 프로젝트 환경 기획	- 프로젝트 주제 결정
프로젝트 구현	08/30(화)	- 프로젝트 환경 구성 - 프로젝트 구현	- 프록시 및 스웜 모드 구성 - 모니터링 도구 구축 - 사설 저장소 구축
프로젝트 오류 수정 및 유지 보수	08/31(수)	- 프로젝트 오류 수정 - 프로젝트 유지 보수	- CRON 자동화 구축
프로젝트 데모 영상 촬영 및 슬라이드 작성	09/01(목) 오전	- 슬라이드 작성 - 데모 영상 촬영	- 슬라이드 자료 완성

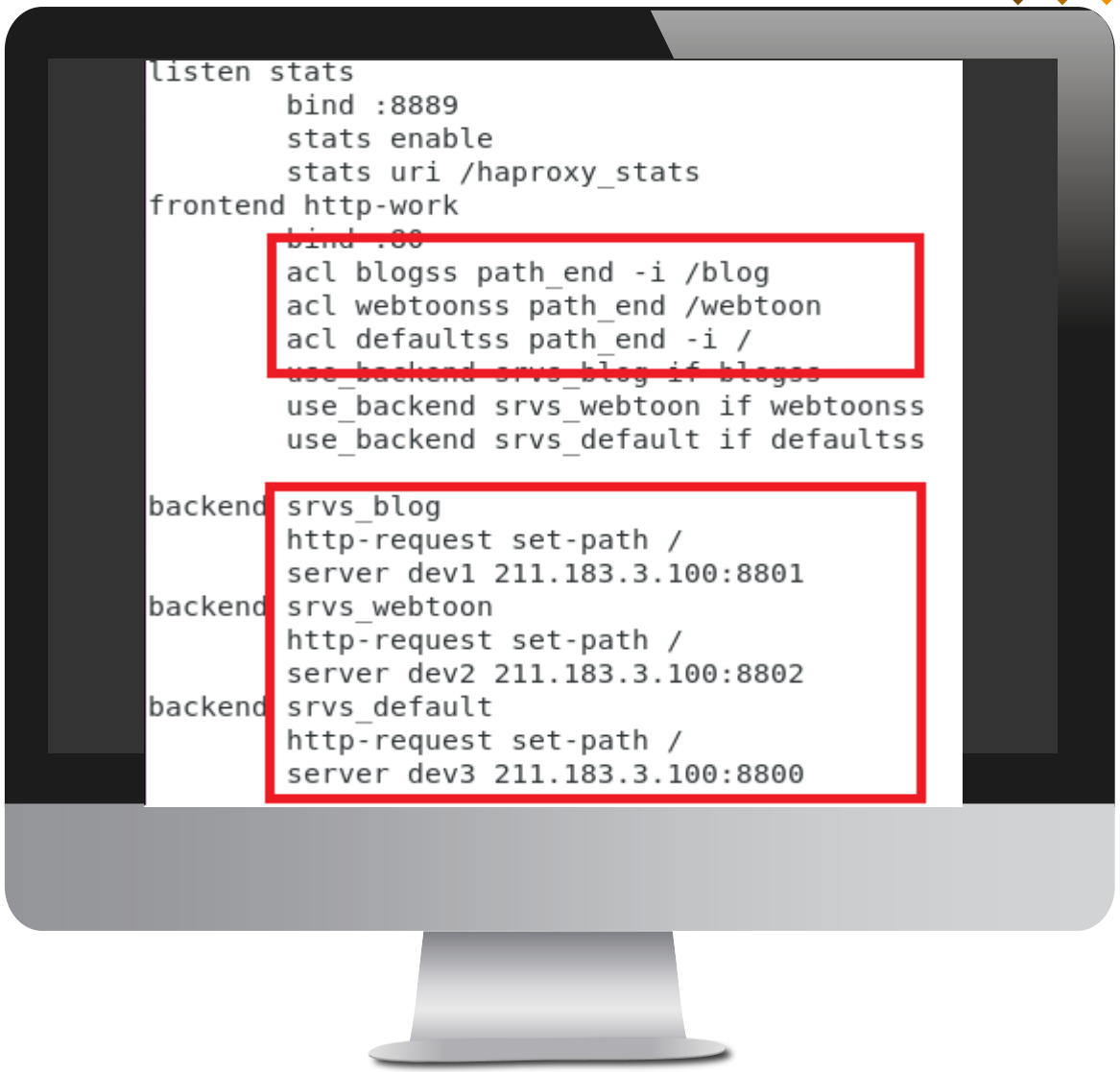
프로젝트 수행 결과

차례

1. **로드밸런서 Proxy 환경 구성**
2. Harbor **저장소 구축**
3. Manager-Worker **노드**
4. **모니터링 구축**
5. **자동화 업데이트**
6. **시연 동영상**

프로젝트 수행 결과 (Cont.)

로드밸런서 Proxy 환경 구성



L7 로드밸런서에서 /로 가면 default(main)으로, /blog 가면 blog로, /webtoon으로 가면 webtoon으로 가도록 로드밸런싱 한다.

프로젝트 수행 결과 (Cont.)

Harbor 저장소 구축

HARBOR을 선택한 이유

Docker를 사용할 때 반드시 Registry가 필요하다.

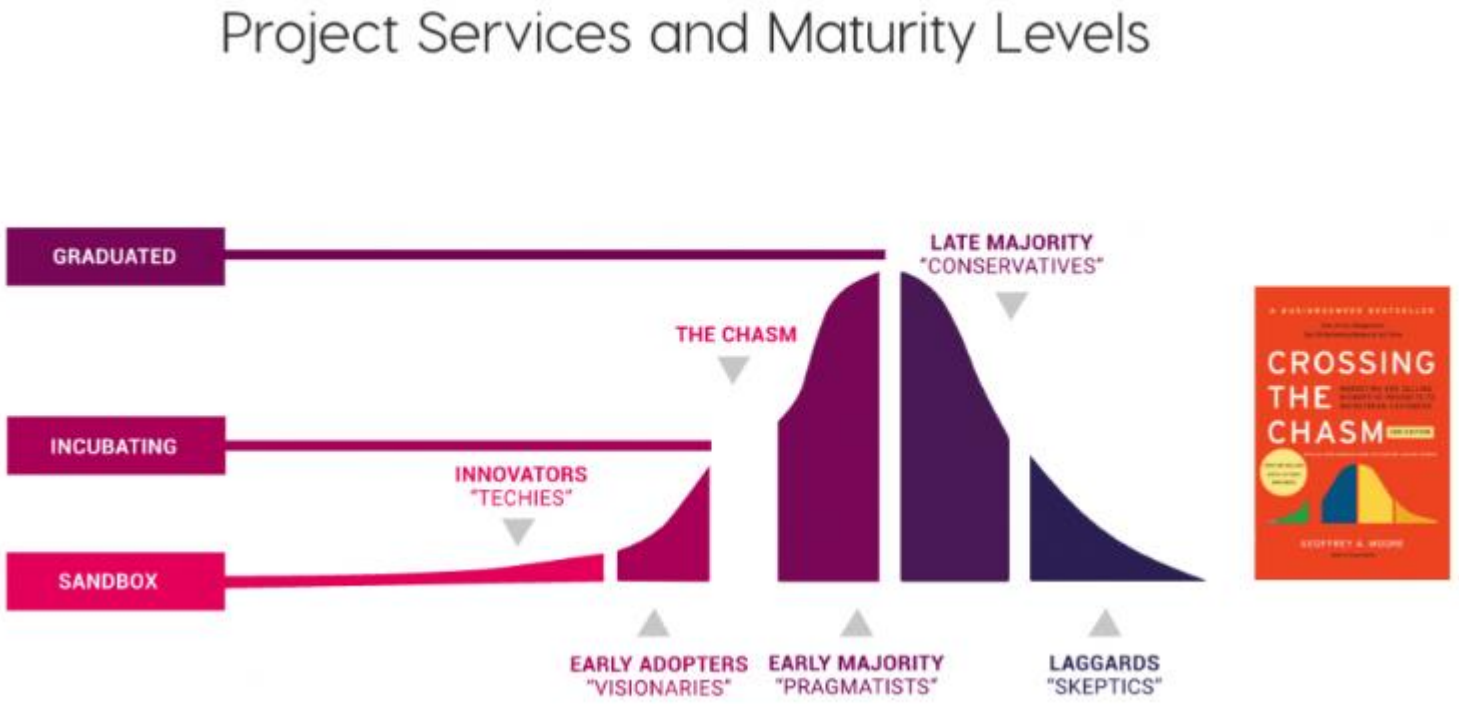
Private Docker Registry가 있지만, 이는 기초적이기 때문에 관리나 보안 측면에서 개선할 필요가 있다고 보여졌다.

이에 검증되고, UI 등 운영이 쉬운 소프트웨어를 고려하게 되었다.

프로젝트 수행 결과 (Cont.)

Harbor 저장소 구축

CNCF 프로젝트 성숙도



CNCF(Cloud Native Computing Foundation) 재단에서 Harbor의 수준을 보았을 때, Incubating 수준으로 충분한 성속도를 가졌다고 생각해 Harbor을 사용해보기로 결정했다.

프로젝트 수행 결과 (Cont.)

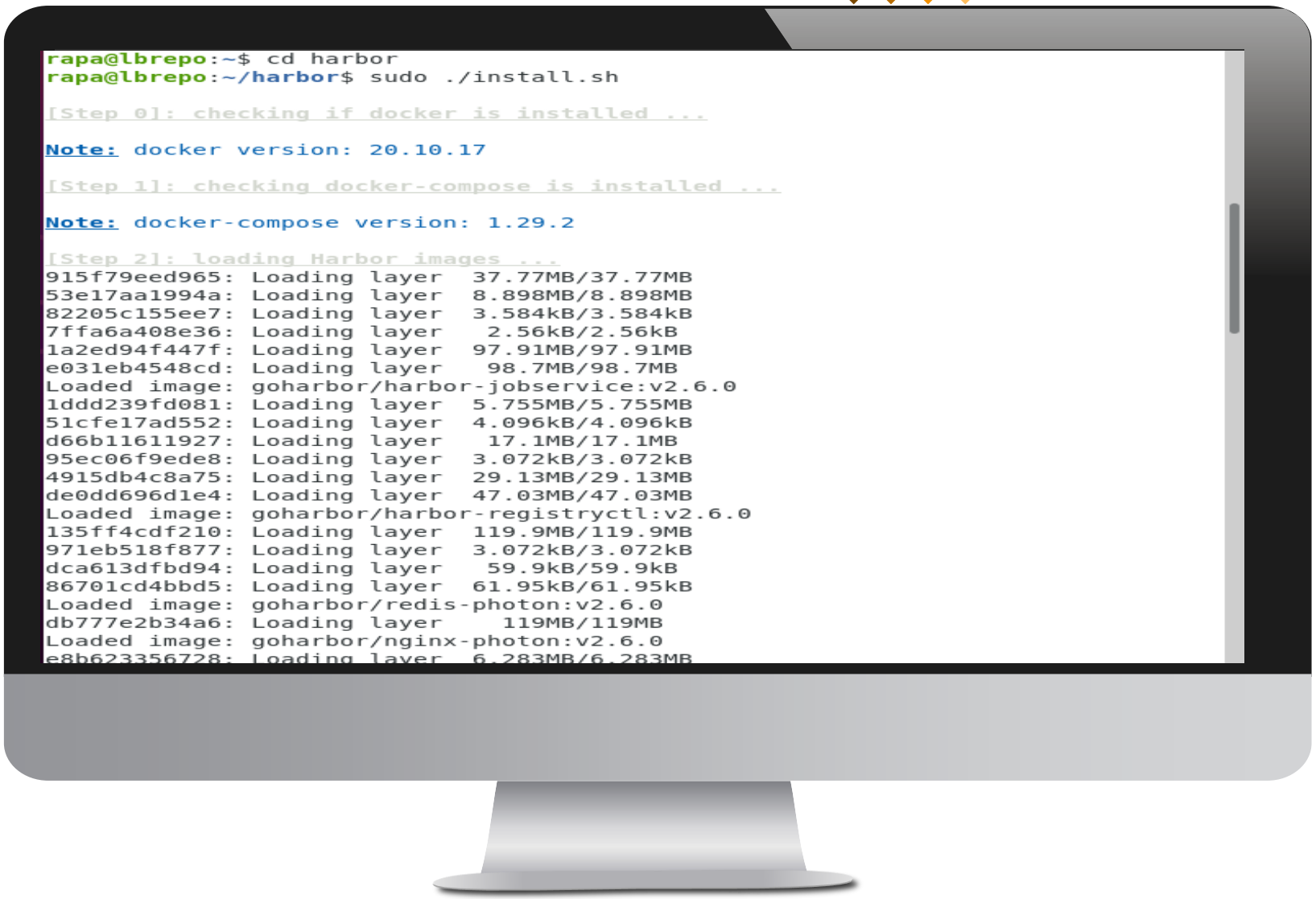
Harbor 핵심 기능

Harbor 저장소 구축

Harbor에서는 또한 여러가지 보안적인 기술과 편리한 기능을 제공하고 있기 때문에 충분히 사용하기 적합하다고 생각해 Harbor 저장소로 프로젝트를 구현해보기로 했다.

프로젝트 수행 결과 (Cont.)

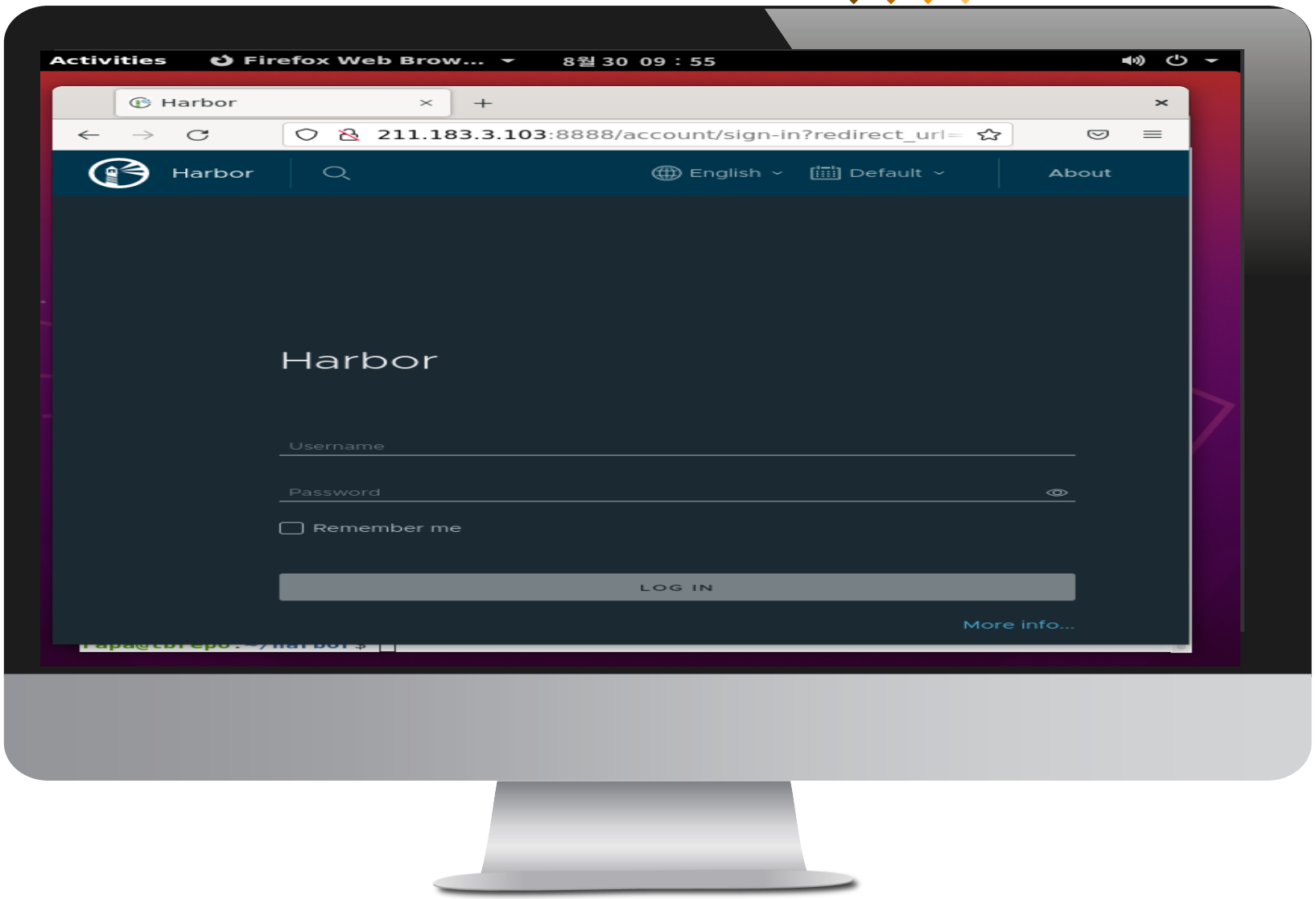
Harbor 저장소 구축



Harbor installer를
다운받고, 설정을 바꾼 후
설치를 시작한다.

프로젝트 수행 결과 (Cont.)

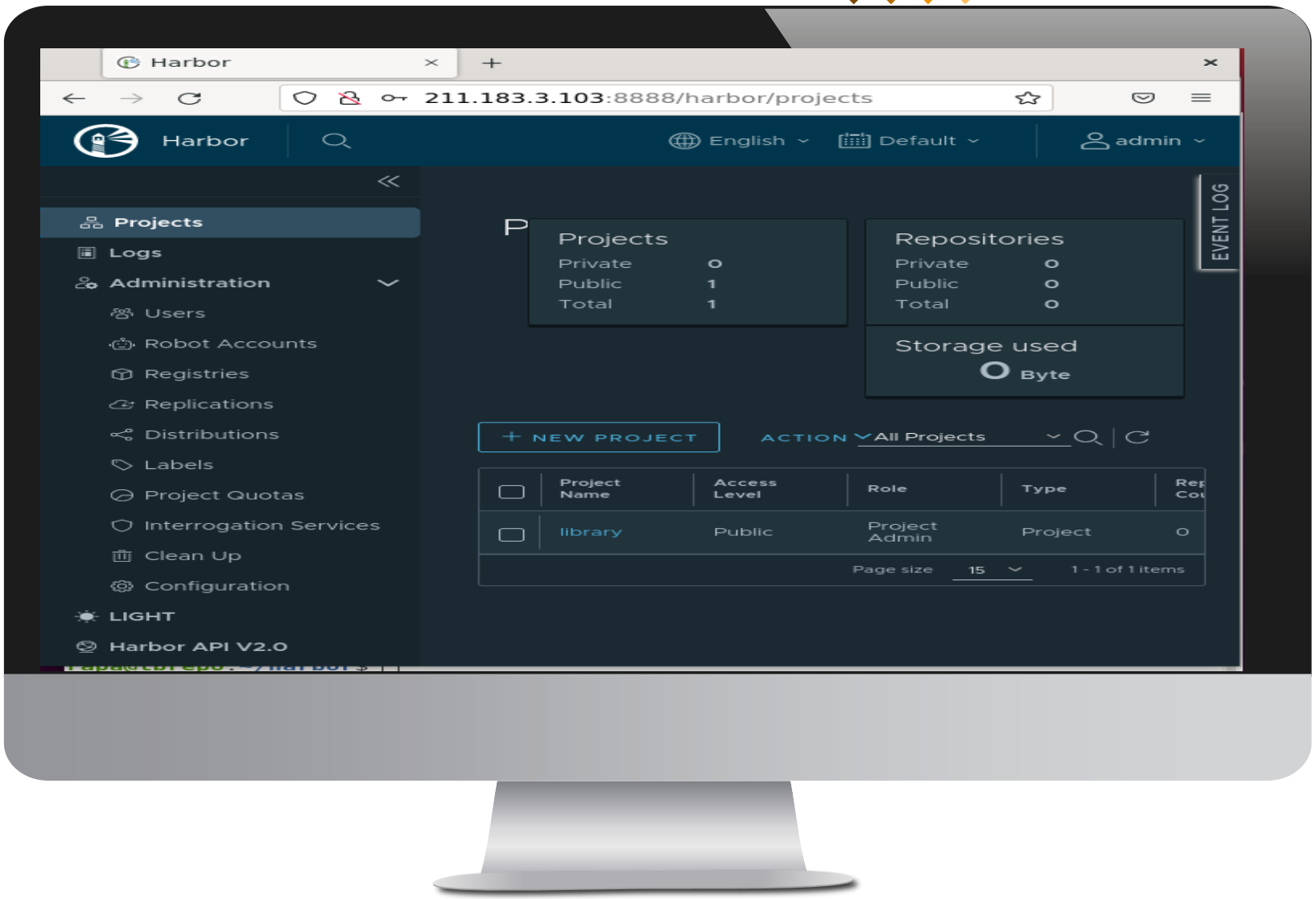
Harbor 저장소 구축



Harbor installer 파일에서 설정한 username과 password로 Harbor에 접속할 수 있다.

프로젝트 수행 결과 (Cont.)

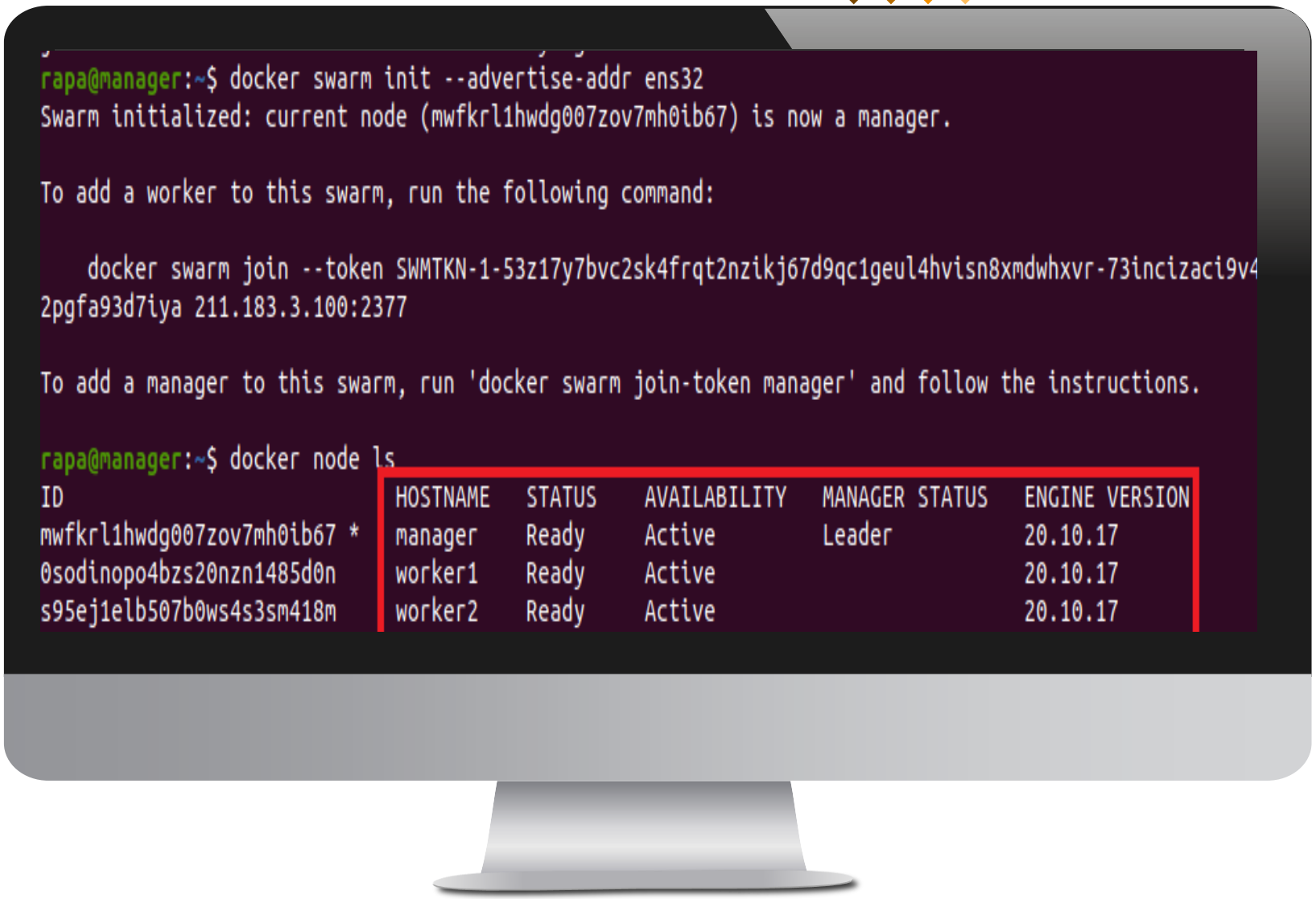
Harbor 저장소 구축



다음과 같이 Harbor 저장소에서 Registry를 통해 이미지를 효과적으로 관리할 수 있다.

프로젝트 수행 결과 (Cont.)

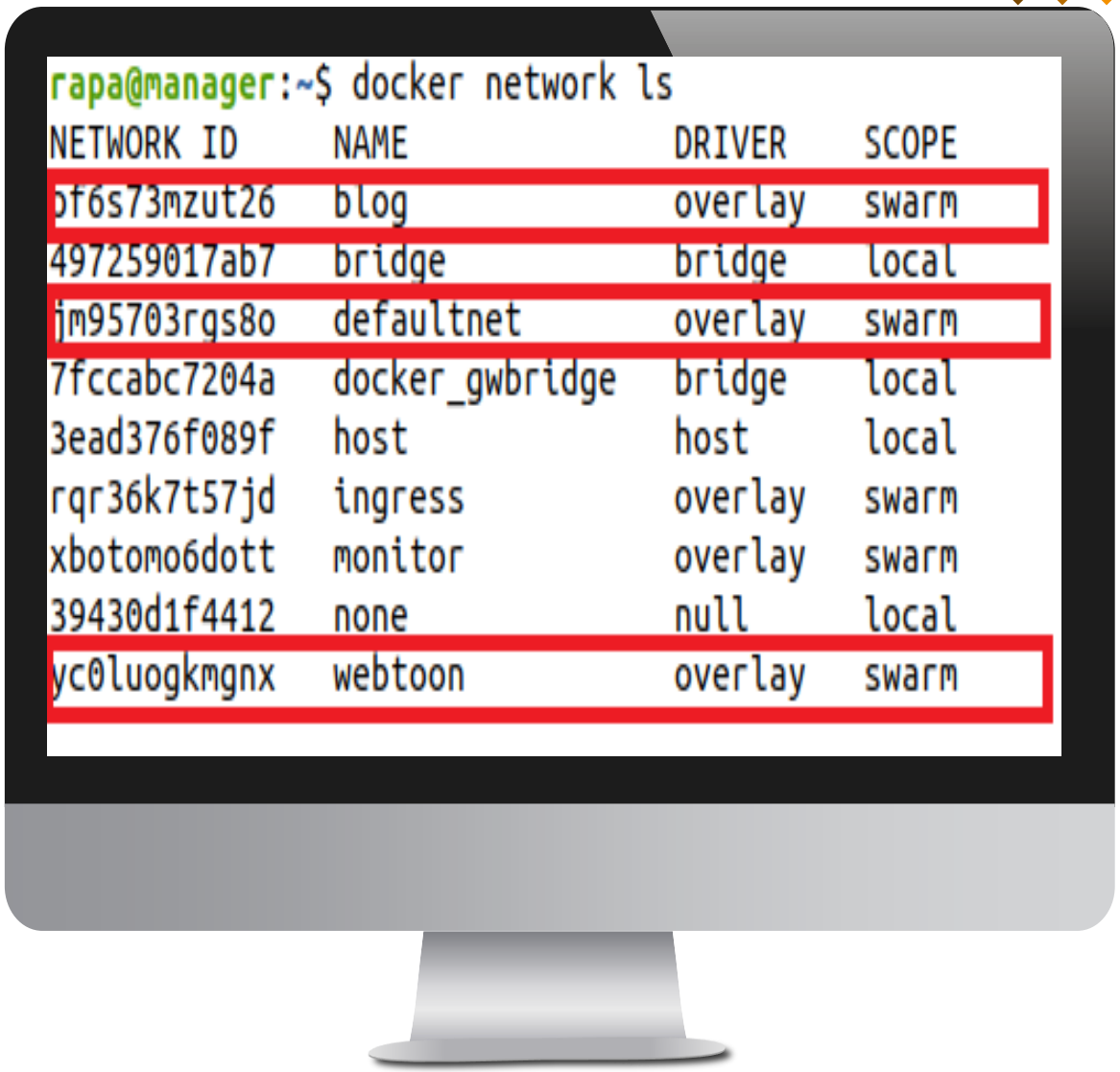
Manager-Worker 노드



도커 스웜 모드를 이용해 Manager노드와 Worker 노드를 구성한다.

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드



환경에 맞추어 각각(main, blog, webtoon)의 오버레이 네트워크를 생성한다.

프로젝트 수행 결과 (Cont.)

```yaml
version: '3.7'

services:
  main:
    image: 211.183.3.103:8888/proweb/web:main
    deploy:
      replicas: 3
      placement:
        constraints: [node.role!=manager]
      restart_policy:
        condition: on-failure
        max_attempts: 2
    environment:
      SERVICE_PORTS: 80
    networks:
     - defaultnet

  proxy:
    image: dockercloud/haproxy
    depends_on:
      - main
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - "8800:80"
    networks:
      - defaultnet
    deploy:
      mode: global
      placement:
        constraints: [node.role==manager]

networks:
  defaultnet:
    external: true
~
```

Main

web.yaml

Main : 워커에 배치, 사설저장소 이미지 받아오기

Proxy : 8800 포트, 매니저에 배치

```yaml
version: '3.7'

services:
  blog:
    image: 211.183.3.103:8888/proweb/web:blog
    deploy:
      replicas: 2
      placement:
        constraints: [node.role!=manager]
      restart_policy:
        condition: on-failure
        max_attempts: 2
    environment:
      SERVICE_PORTS: 80
    networks:
     - blog

  proxy:
    image: dockercloud/haproxy
    depends_on:
      - blog
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - "8801:80"
    networks:
      - blog
    deploy:
      mode: global
      placement:
        constraints: [node.role==manager]

networks:
  blog:
    external: true
```

Blog

web.yaml

Blog : 워커에 배치, 사설저장소 이미지 받아오기

Proxy : 8801 포트, 매니저에 배치

```yaml
version: '3.7'

services:
  webtoon:
    image: 211.183.3.103:8888/proweb/web:webtoon
    deploy:
      replicas: 2
      placement:
        constraints: [node.role!=manager]
      restart_policy:
        condition: on-failure
        max_attempts: 2
    environment:
      SERVICE_PORTS: 80
    networks:
     - webtoon

  proxy:
    image: dockercloud/haproxy
    depends_on:
      - webtoon
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - "8802:80"
    networks:
      - webtoon
    deploy:
      mode: global
      placement:
        constraints: [node.role==manager]

networks:
  webtoon:
    external: true
```

Webtoon

web.yaml

Main : 워커에 배치, 사설저장소 이미지 받아오기

Proxy : 8802 포트, 매니저에 배치

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드

main, blog, webtoon stack을 배포한다.

Main proxy는 8800포트,

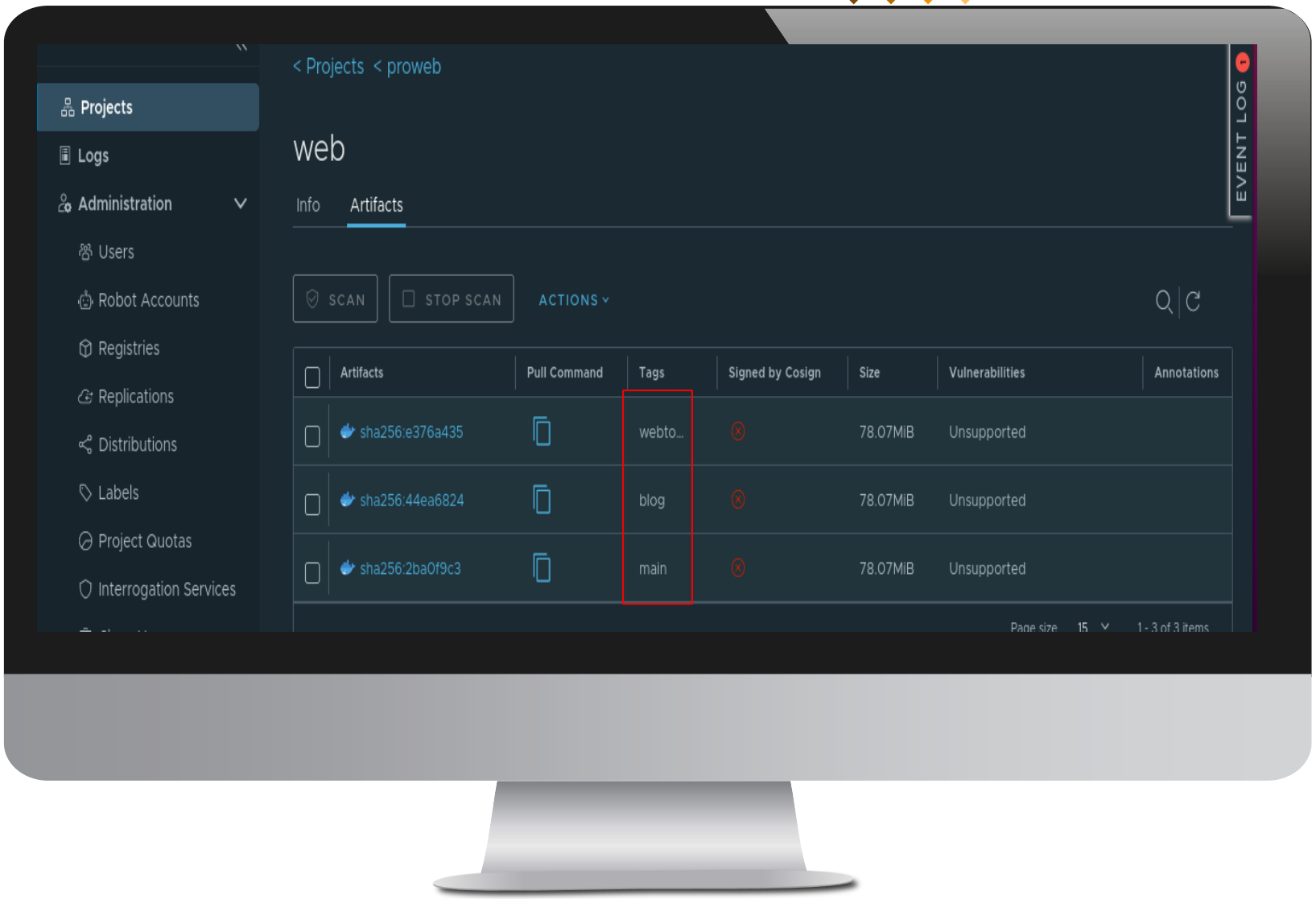
blog proxy는 8801포트,

webtoon proxy는 8882번 포트로 지정한다.

각각의 Proxy로 올 경우 배포된 웹 컨테이너로 로드밸런싱 한다.

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드



Harbor에서 정상적으로 main, blog, webtoon 이미지가 올라와 있는 것을 확인 할 수 있다.

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드

211.183.3.103에 접속할 경우 L7 로드밸런서에 의해 / 인 Main Page가 보이게 된다.

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드

Main Page에서 Blog URL을 클릭할 경우 로드밸런서에 의해 /blog로 이동되어 Blog Page가 보이게 된다.

프로젝트 수행 결과 (Cont.)

Manager-Worker 노드

Main Page에서 Webtoon URL을 클릭할 경우 로드밸런서에 의해 /webtoon으로 이동되어 Webtoon Page가 보이게 된다.

프로젝트 수행 결과 (Cont.)

Harbor 저장소 구축

Prometheus & Grafana **도입 이유**

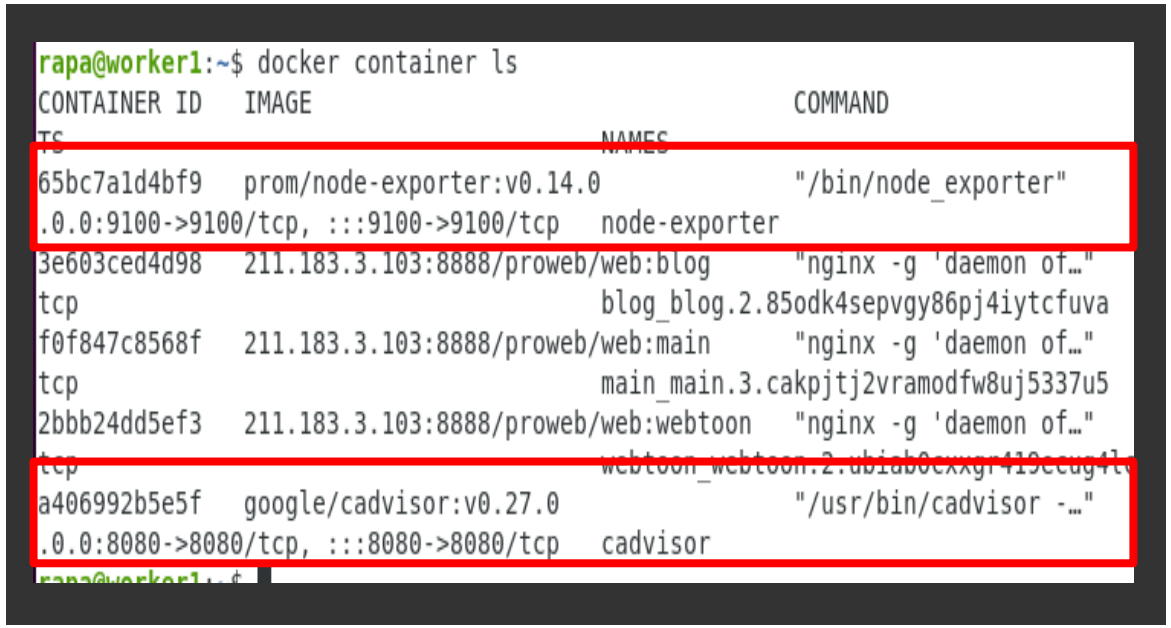
Prometheus 외에 Datadog, New Relic 등과 같은 모니터링 도구가 있다.

그 중 Prometheus는 널리 쓰는 모니터링 도구 중 하나이며 **오픈 소스**이기 때문에 사용자 층이 넓어 프로젝트를 하며 필요한 자료를 참고하기 비교적 쉬울 것이라 생각했다.

또한, Prometheus는 Grafana와의 호환성이 좋아 모니터링 데이터들을 간단하게 **시각화** 해 웹 브라우저로 분석 할 수 있어 해당 도구를 프로젝트에서 도입했다.

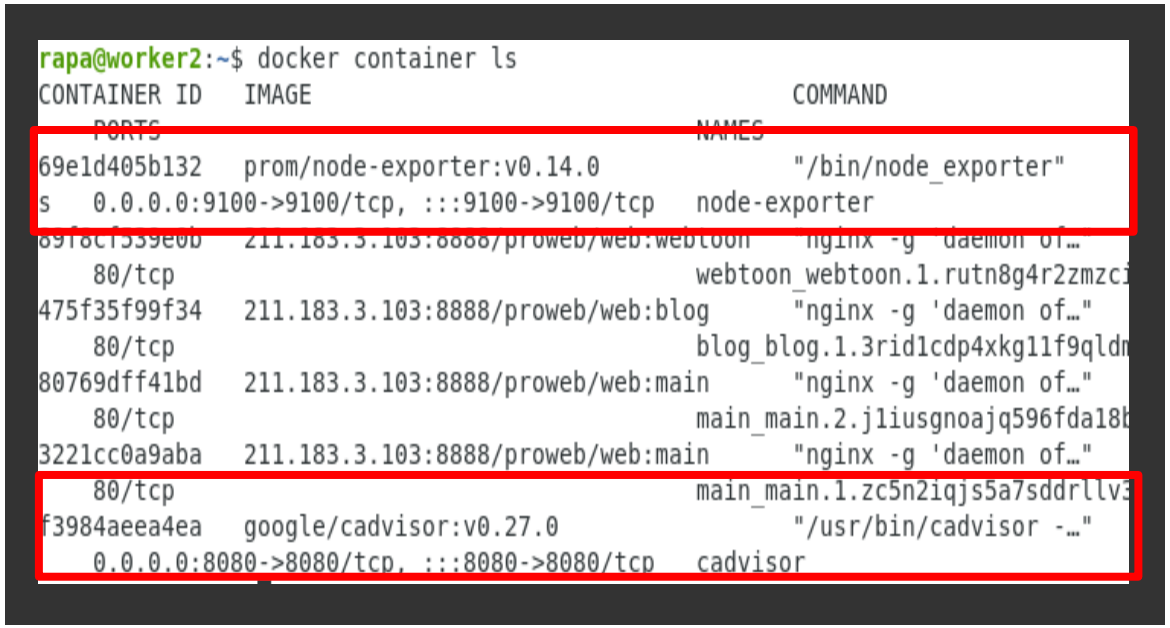
프로젝트 수행 결과 (Cont.)

모니터링 구축 – Worker 노드



Worker 1

Worker1에 cadvisor, node-exporter 컨테이너를 구축



Worker 2

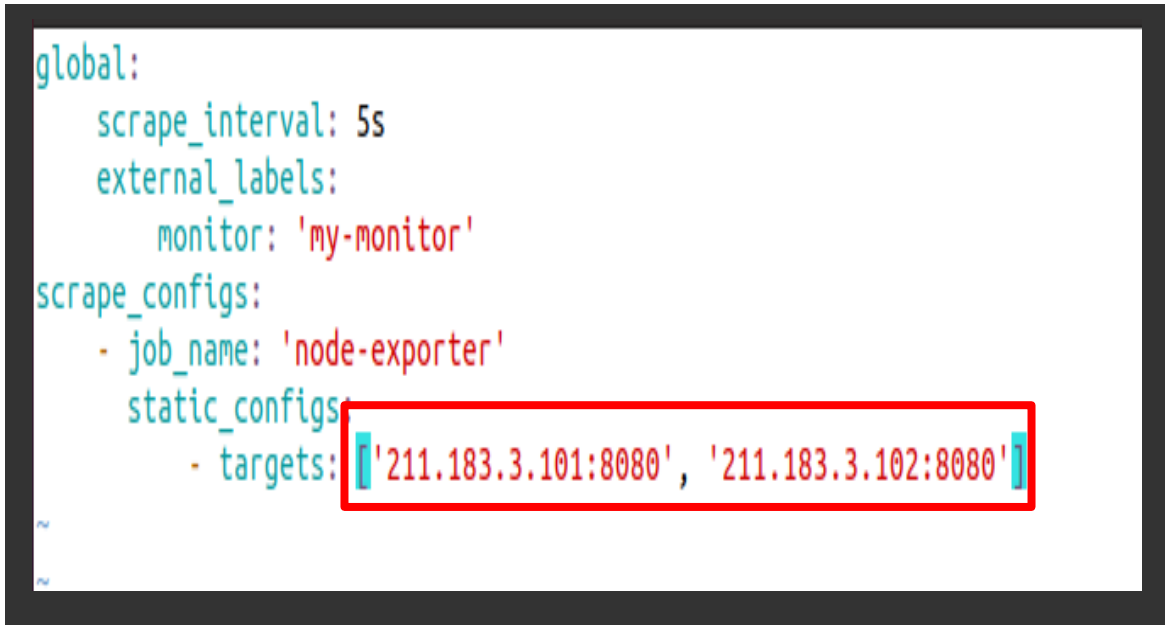
Worker2에 cadvisor, node-exporter 컨테이너를 구축

프로젝트 수행 결과 (Cont.)

모니터링 구축 – Manager 노드

prometheus.yaml

Node-exporter을 위한 Prometheus 설정 : 9100포트

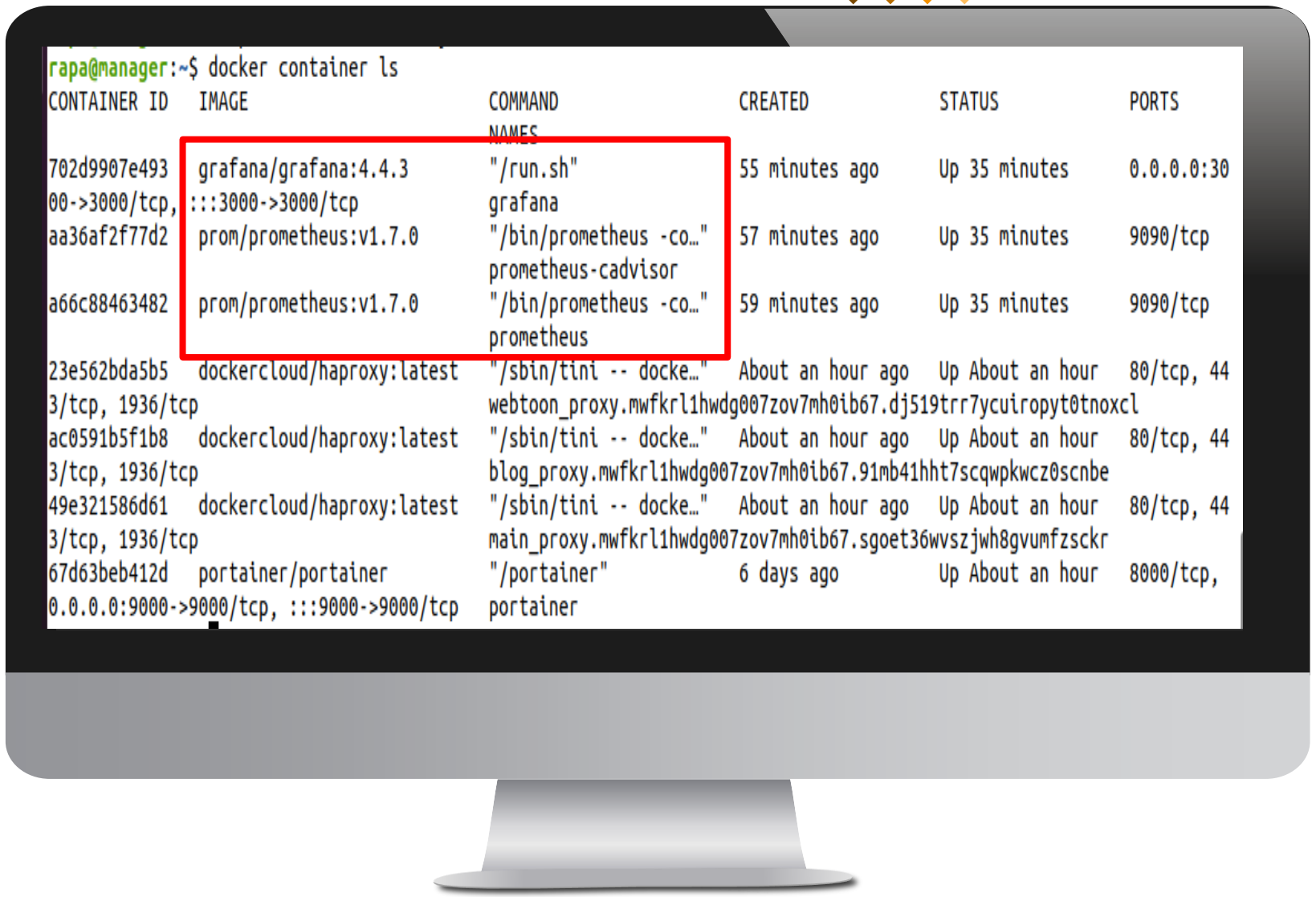


prometheus-cadvisor.yaml

cAdvisor을 위한 Prometheus 설정 : 8080포트

프로젝트 수행 결과 (Cont.)

모니터링 구축 – Manager 노드



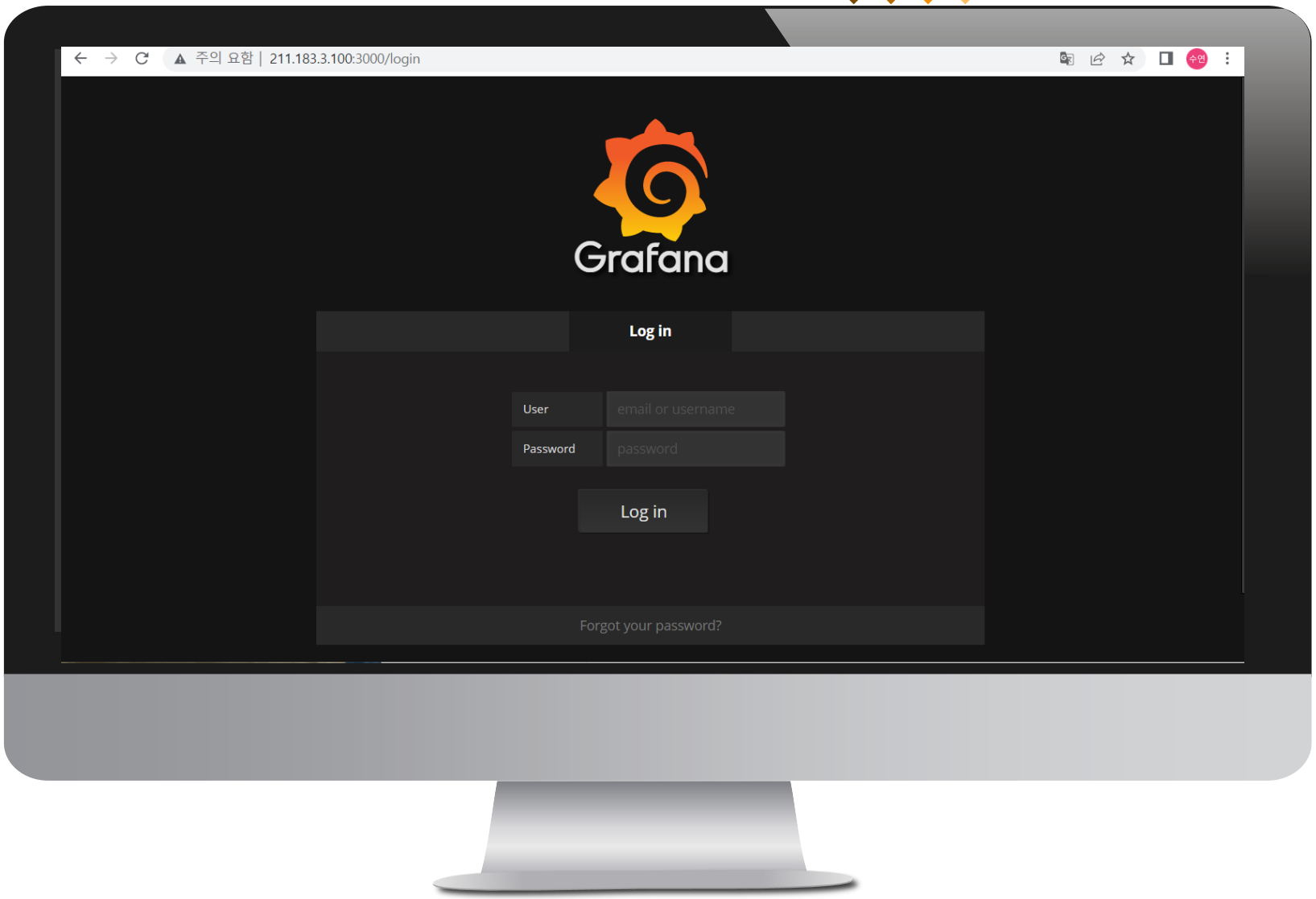
```
rapa@manager:~$ docker container ls
CONTAINER ID   IMAGE                                COMMAND                  CREATED          STATUS          PORTS
                                                    NAMES
702d9907e493   grafana/grafana:4.4.3                "/run.sh"                55 minutes ago   Up 35 minutes   0.0.0.0:30
00->3000/tcp, :::3000->3000/tcp                     grafana
aa36af2f77d2   prom/prometheus:v1.7.0               "/bin/prometheus -co…"   57 minutes ago   Up 35 minutes   9090/tcp
                                                    prometheus-cadvisor
a66c88463482   prom/prometheus:v1.7.0               "/bin/prometheus -co…"   59 minutes ago   Up 35 minutes   9090/tcp
                                                    prometheus
23e562bda5b5   dockercloud/haproxy:latest           "/sbin/tini -- docke…"   About an hour ago   Up About an hour   80/tcp, 44
3/tcp, 1936/tcp                                    webtoon_proxy.mwfkrl1hwdg007zov7mh0ib67.dj519trr7ycuiropyt0tnoxcl
ac0591b5f1b8   dockercloud/haproxy:latest           "/sbin/tini -- docke…"   About an hour ago   Up About an hour   80/tcp, 44
3/tcp, 1936/tcp                                    blog_proxy.mwfkrl1hwdg007zov7mh0ib67.91mb41hht7scqwpkwcz0scnbe
49e321586d61   dockercloud/haproxy:latest           "/sbin/tini -- docke…"   About an hour ago   Up About an hour   80/tcp, 44
3/tcp, 1936/tcp                                    main_proxy.mwfkrl1hwdg007zov7mh0ib67.sgoet36wvszjwh8gvumfzsckr
67d63beb412d   portainer/portainer                  "/portainer"             6 days ago       Up About an hour   8000/tcp,
0.0.0.0:9000->9000/tcp, :::9000->9000/tcp           portainer
```

Cadvisor**과** node-exporter을 모니터링 하기 위한 Prometheus **컨테이너**가 생성되었다.

이를 시각화하기위해 Grafana **컨테이너**로 모니터링 결과를 시각화 해 볼 수 있다.

프로젝트 수행 결과 (Cont.)

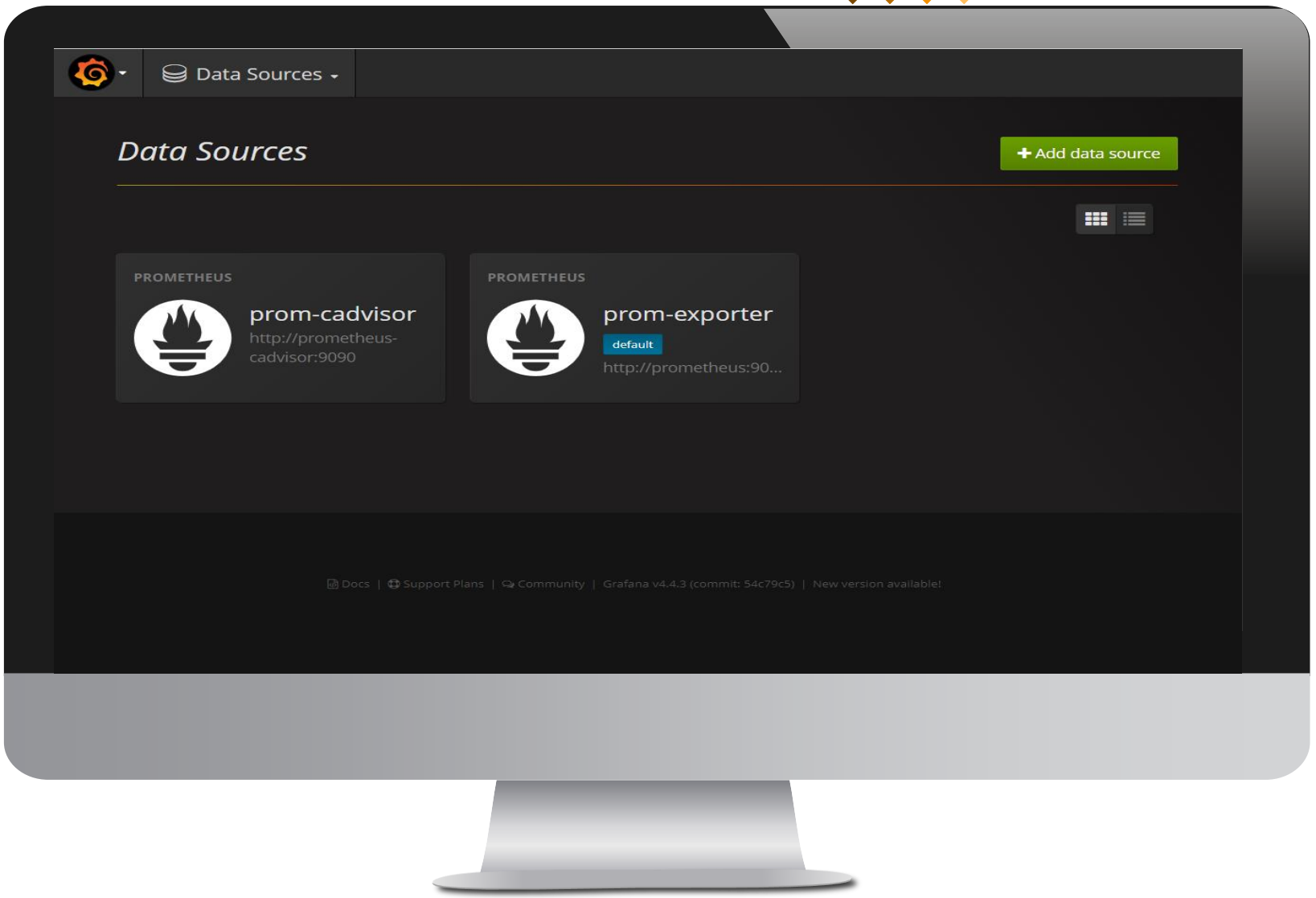
모니터링 구축 – Manager 노드



매니저 노드의 3000번 포트로 Grafana에 접속 할 수 있다.

프로젝트 수행 결과 (Cont.)

모니터링 구축 – Manager 노드



Grafana 대시보드를 이용해 prom-cadvisor과 prom-exporter 를 모니터링 할 수 있다.

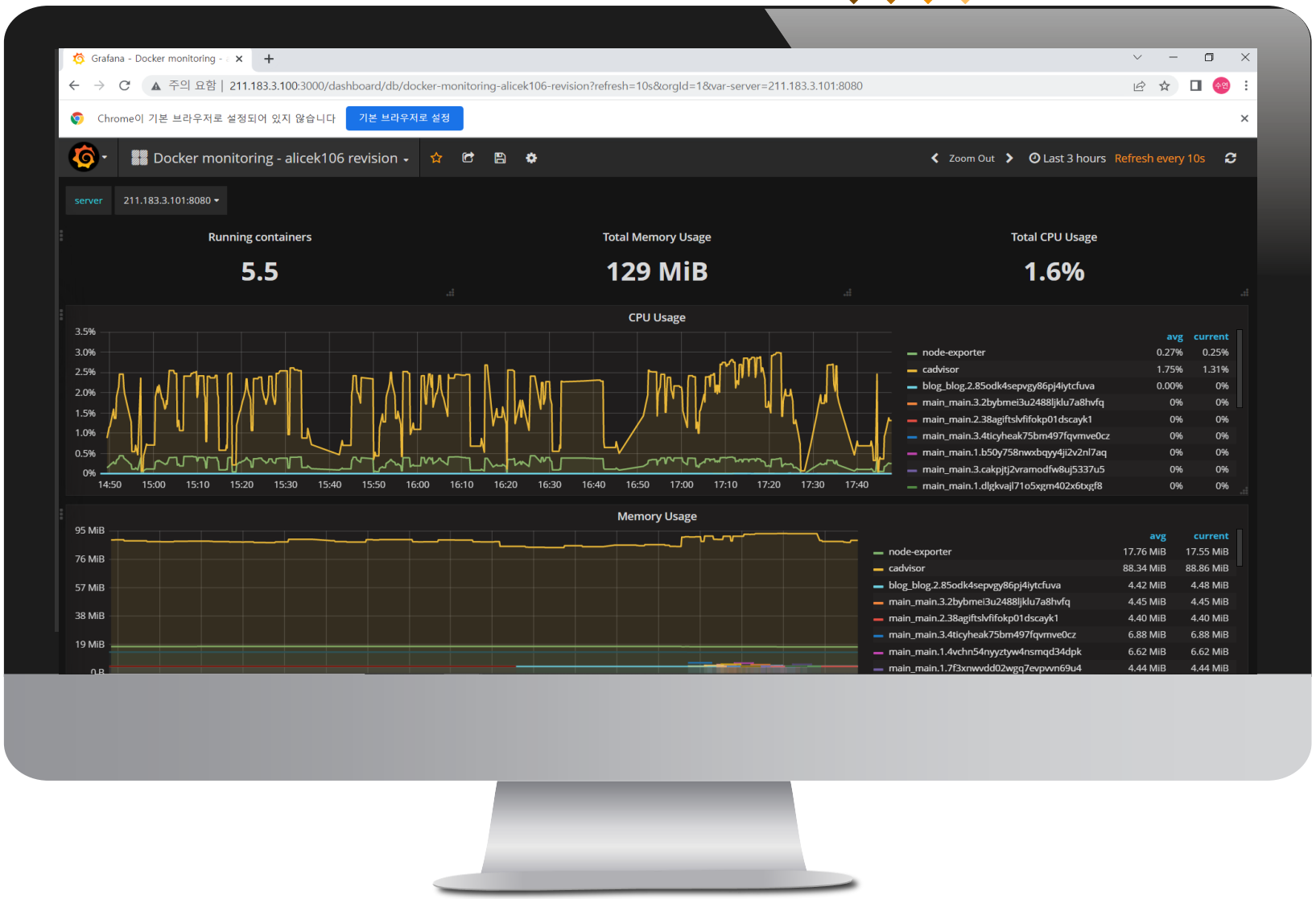
프로젝트 수행 결과 (Cont.)

모니터링 구축 – Manager 노드

Worker 노드 모니터링

프로젝트 수행 결과 (Cont.)

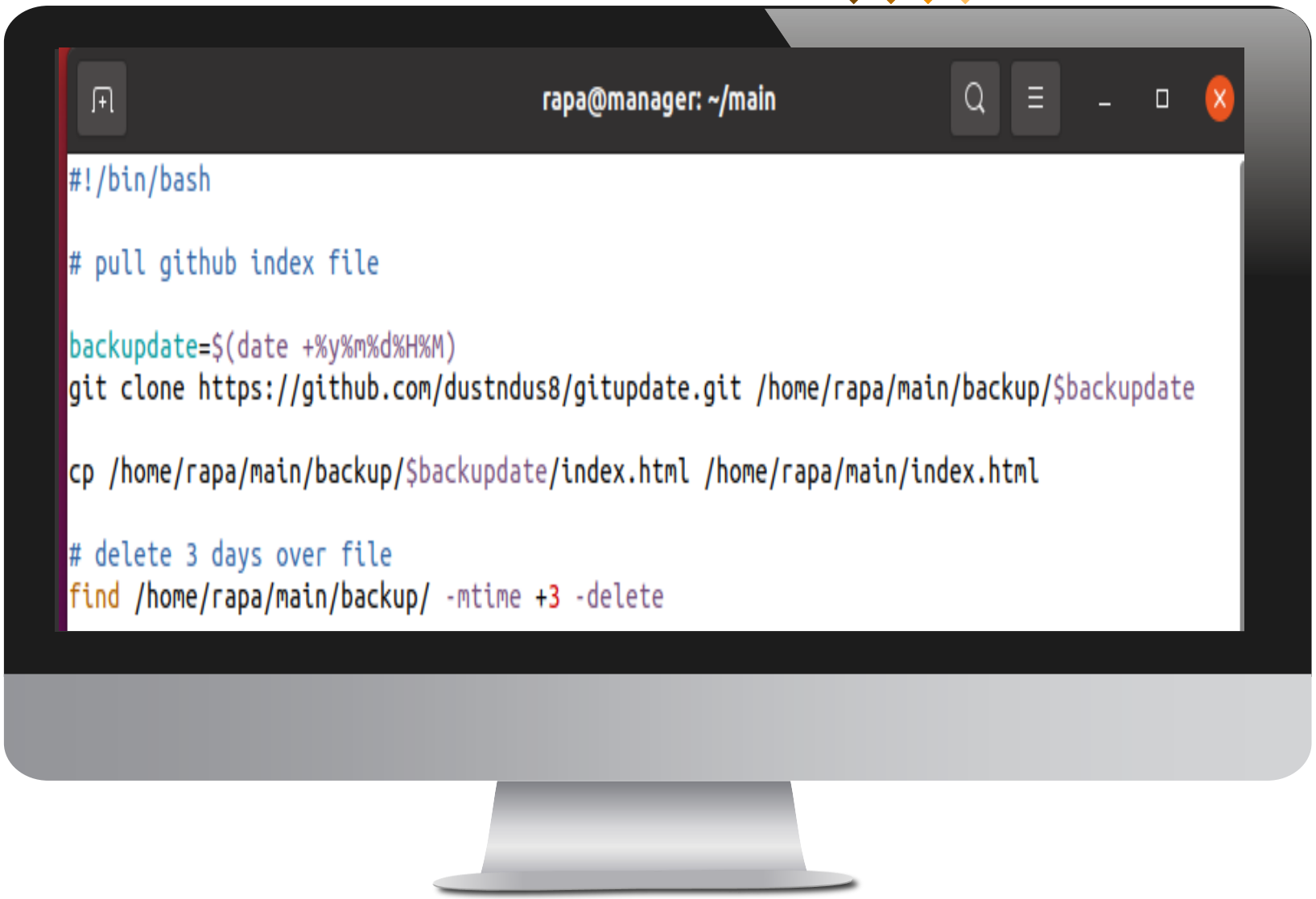
모니터링 구축 – Manager 노드



컨테이너 모니터링

프로젝트 수행 결과 (Cont.)

자동화 업데이트



backup.sh

Github 저장소에서 파일을 Clone해서 main의 index.html로 업데이트 한다.

또한, 3일 이상 지난 파일(Clone된 파일)은 삭제하도록 한다.

프로젝트 수행 결과 (Cont.)

자동화 업데이트

```
#!/bin/bash

backupdate=$(date +%y%m%d%H%M)
docker build -t proweb:main${backupdate} /home/rapa/main/
docker image tag proweb:main${backupdate} 211.183.3.103:8888/proweb/web:main${backupdate}
docker push 211.183.3.103:8888/proweb/web:main${backupdate}

sed -i '5s/.*/    image:211.183.3.103:8888\/proweb\/web:main${backupdate}/g' /home/rapa/main/web.yml
docker service update --image 211.183.3.103:8888/proweb/web:main${backupdate} main_main
```

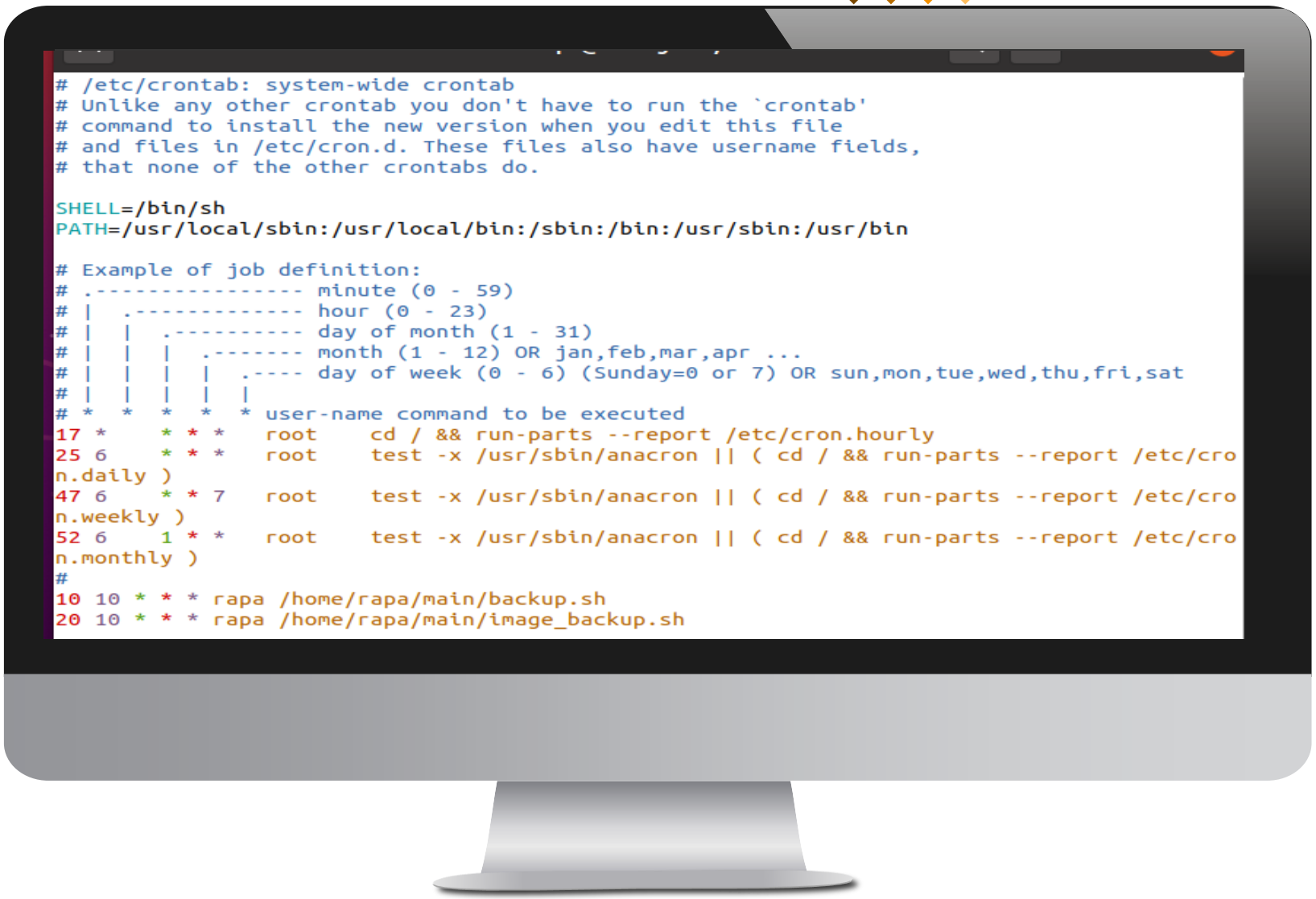
Image_backup.sh

Main의 도커 이미지를 사설저장소에 Push한다.

또한, main의 스택 파일을 업데이트한 이미지로 바꾸어 업데이트를 진행한다.

프로젝트 수행 결과 (Cont.)

자동화 업데이트

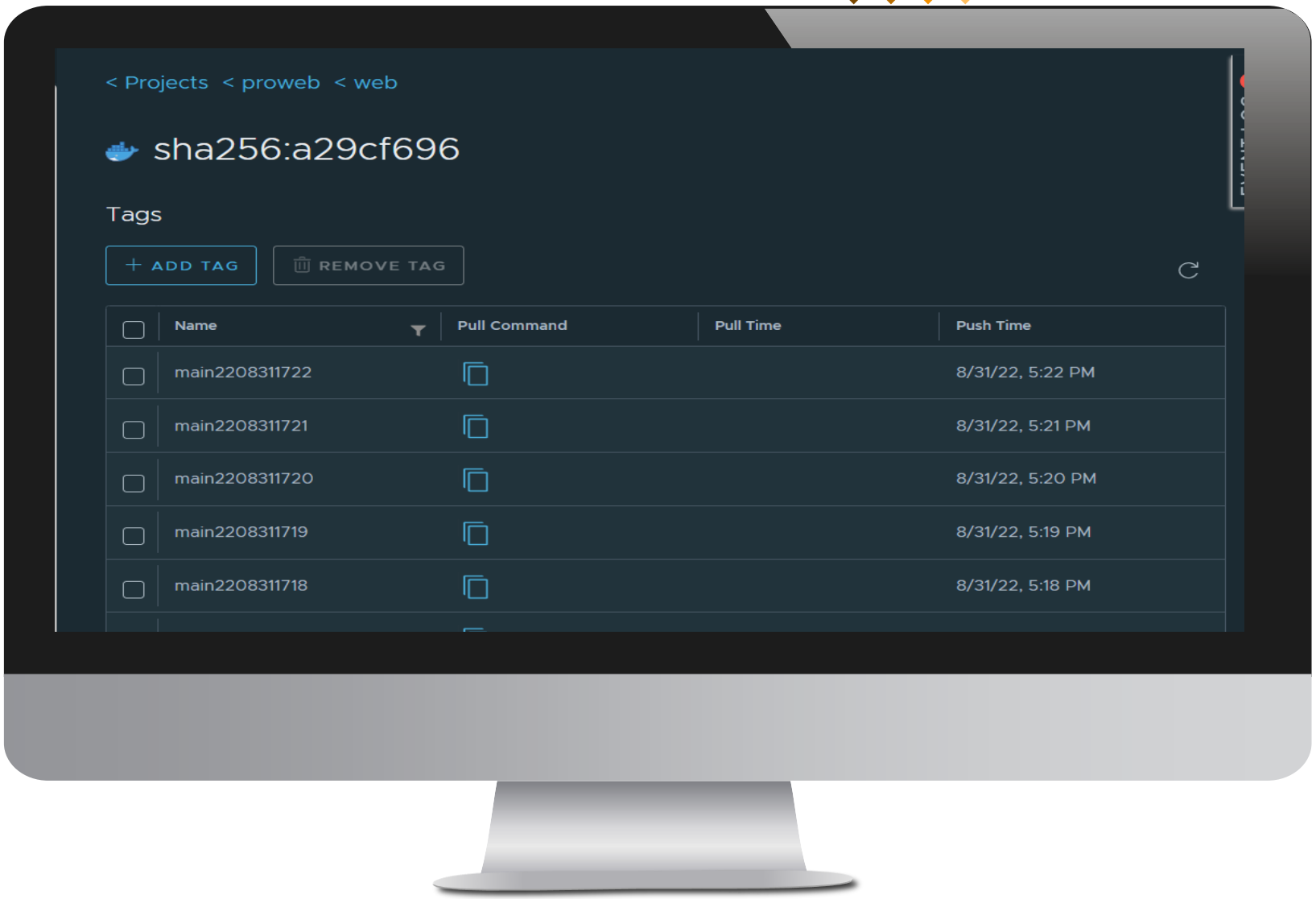


CRON **설정**

CRON을 이용해 backup.sh와 image_backup.sh를 매일 10시 10분 / 20분에 업데이트 하도록 설정한다.

프로젝트 수행 결과 (Cont.)

자동화 업데이트

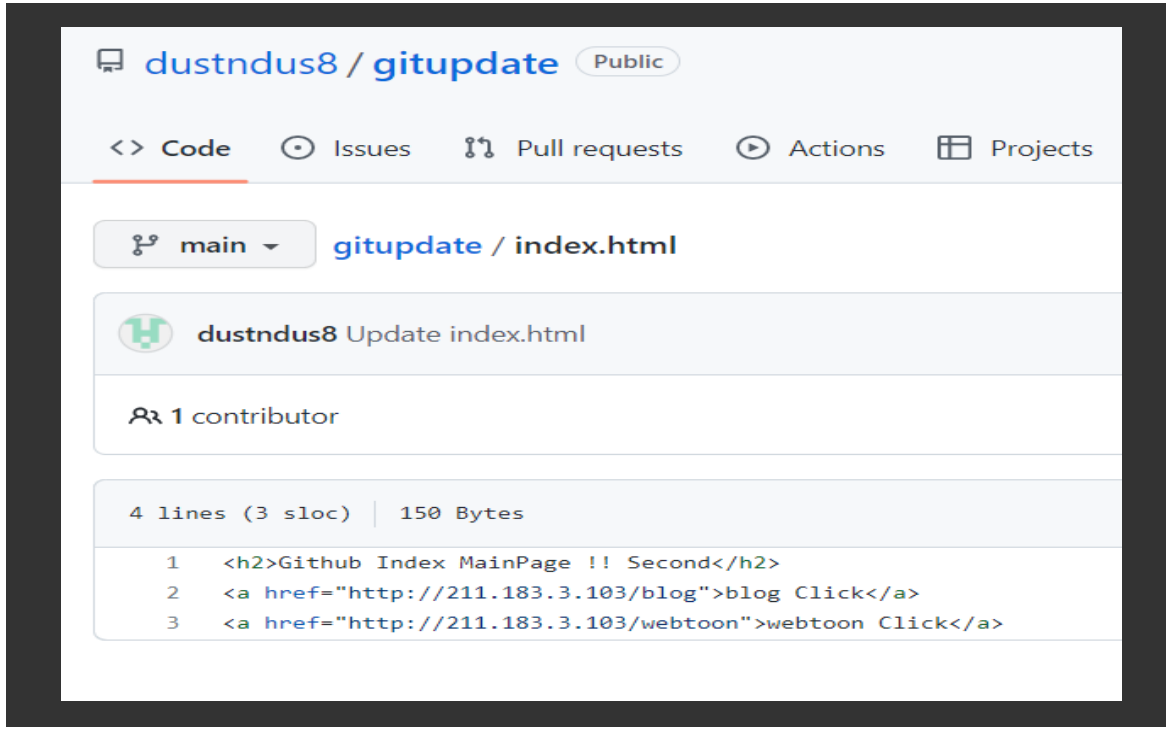


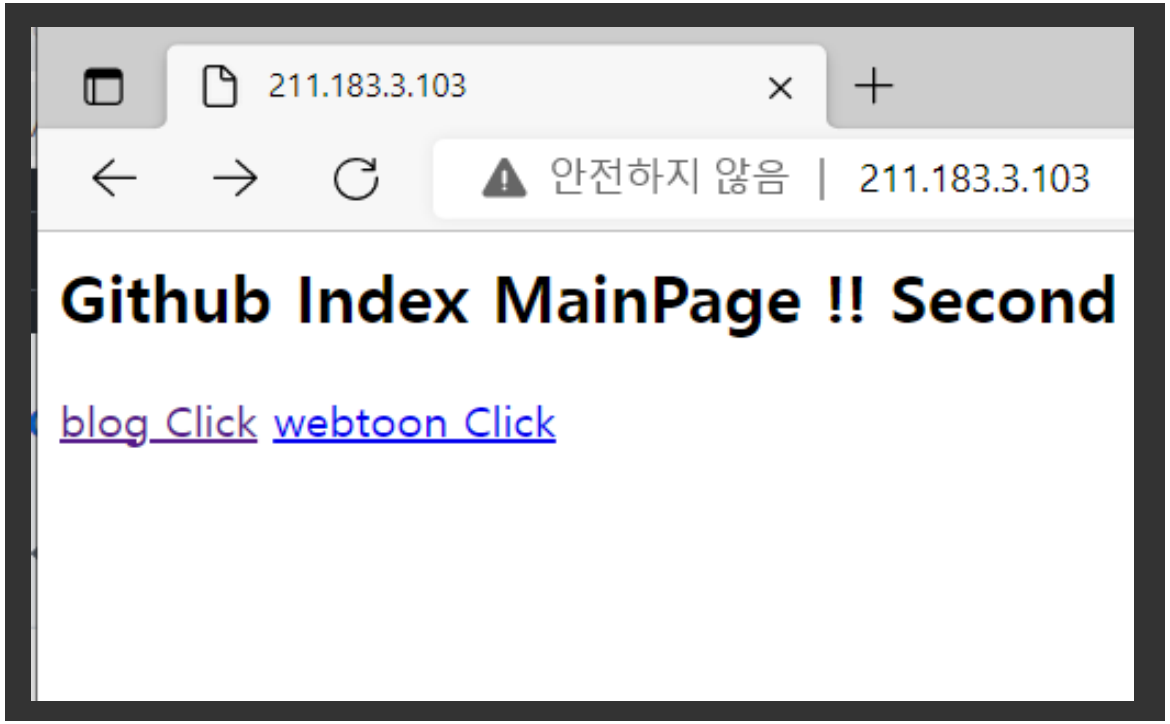
Harbor 저장소에 업데이트 된 이미지가 올라 간 것을 확인할 수 있다.

해당 이미지는 테스트용으로 1분마다 생성하게 했다.

프로젝트 수행 결과 (Cont.)

자동화 업데이트





Github에 업데이트 한 index 파일대로 Main Page가 업데이트 된 것을 확인할 수 있다.

프로젝트 수행 결과 (Cont.)

시연 영상

https://youtu.be/VCmE9YuIp6M

느낀점

어려움 극복

- 인프라 환경을 직접 구현하는 것만큼 구조를 이해하고 구상하는 것이 어려웠는데, 그림으로 직접 그려가며 논리적이고 적절한 인프라 환경을 찾아 구상하였다.
- 또한, 구현하는 과정에서 CRON 기능이 수행되지 않는 문제가 발생해 시간을 소요했는데, 먼저 CRON 자체가 실행이 되지 않는 지 로그를 통해 확인했고, 다음으로 스크립트를 한 줄 씩 실행 해보면서 수행 되지 않는 부분을 찾아가며 작은 부분부터 천천히 문제를 해결했다.

느낀점

잘한 부분

- 처음 구상한 인프라 환경에서 개선할 부분을 찾아 자동화에 대한 기능을 추가했다.
- 또한, 단기간에 수행 할 수 있도록 적절한 계획을 세워 일정에 맞게 수행했으며, 오류가 발생했을 때 포기하지 않고 해결해나갔다.

아쉬운 부분

- CRON을 통해 주기적으로 인덱스 파일 및 도커 이미지를 업데이트 할 때, 인덱스 파일은 일정 시간이 지나면 삭제되도록 스크립트를 작성할 수 있었지만, Harbor에 저장된 도커 이미지는 직접 삭제해주어야 한 점이 아쉬웠다. 이후 Jenkins를 이용하여 더 효율적으로 관리할 수 있는 CI/CD 환경을 구상하면 좋을 것 같다.

느낀점 (Cont.)

진로 설계

- 해당 훈련에서 클라우드에 대해 처음 공부해보았는데, 직접 클라우드 인프라 환경을 구상하고, 구현해보며 훈련에서 배운 클라우드에 대해 더 깊이 이해할 수 있었다.
- 클라우드에 대해 더 배우고 프로젝트를 발전시켜 직접 클라우드 서비스를 제공할 수 있도록 하고 싶다. 또한, 이후 취업을 준비 할 때에도 클라우드 관련 직무를 가지고 일을 하고 싶다.

기타

- 프로젝트를 여기서 끝내는 것이 아니라, 이후 프로젝트에서 해당 프로젝트에서 아쉬운 부분 등을 보완하여 더 발전된 프로젝트를 구상하고 싶다.

카카오 클라우드 엔지니어 양성과정 1기

Finish

연수연 (개인프로젝트)