

Docker 9,13

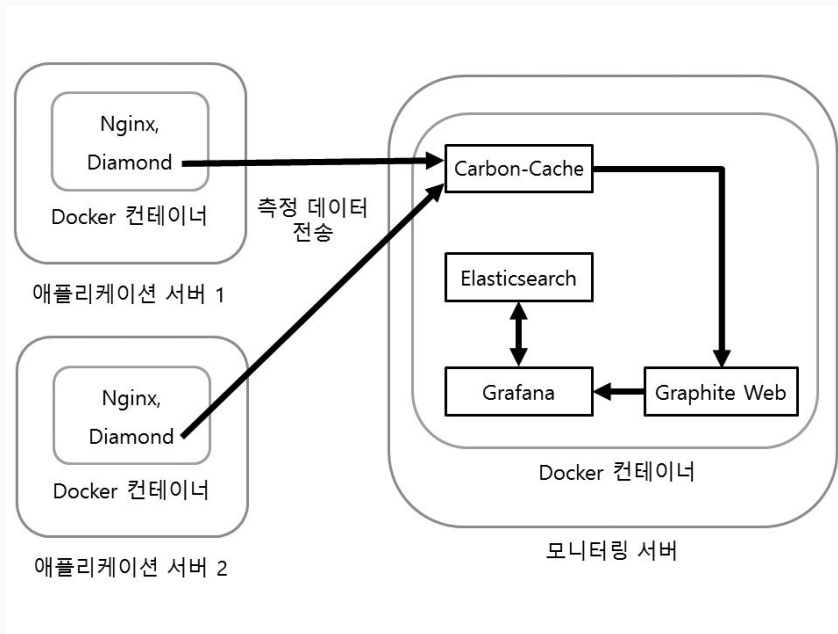
9.Docker 모니터링하기

Docker로 서비스를 구축했을 때 서버의 CPU나 메모리 사용량이 얼마나 되는지 모니터링할 필요가 있다.

다양한 도구 중에서 **Graphite**를 사용하여 서버 모니터링 시스템을 구축

Graphite는 서버의 자원을 모니터링하고 그래프를 출력해주는 오픈 소스 도구이며 다양한 도구를 조합하여 사용합니다.

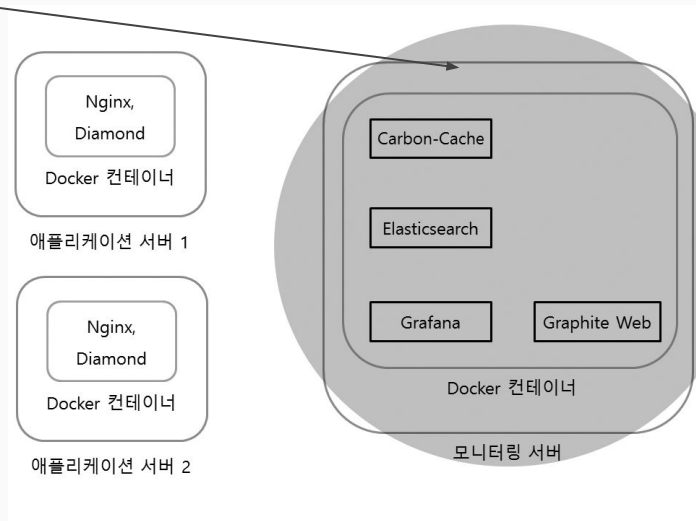
(Carbon, Graphite Web, Grafana, Elasticsearch, Diamond)



- **Graphite**: 시간 기반의 측정치(Metric) 데이터를 저장하는 저장소입니다. Python으로 작성되어 있습니다.
 - **Graphite Web**: 저장된 측정치 데이터와 시간을 그래프로 표시해주는 웹 애플리케이션입니다.
 - **Carbon-Cache**: 네트워크상에서 측정치 데이터를 수집하고 디스크에 저장하는 데몬입니다.
- **Grafana**: Graphite 대시보드입니다. 웹 애플리케이션이며 Graphite Web보다 UI나 화면 구성이 좀더 깔끔합니다. Graphite Web에서 데이터를 가져오기 때문에 Graphite Web이 필요합니다.
 - **Elasticsearch**: Apache Lucene을 기반으로하는 검색엔진입니다. 여기서는 Grafana 대시보드상의 그래프 설정값을 저장하는 용도로만 사용합니다.
- **Diamond**: 서버 자원의 측정치 데이터를 수집하여 Carbon-Cache로 전송하는 도구입니다. CPU, 메모리, 네트워크 사용량과 디스크 I/O 등의 데이터를 수집합니다.

9-1.모니터링 서버 Dockerfile 작성하기

모니터링 서버에서 사용할 **Docker** 이미지를 만듭니다



다음 파일들은 저의 **GitHub** 저장소에 있는 예제 파일을 받아서 사용합니다.
모니터링 서버를 구축할 것이므로 모니터링 서버로 사용할 컴퓨터에 받습니다.

```
$ git clone https://github.com/pyrasis/dockerbook.git
```

9-1.모니터링 서버 Dockerfile 작성하기

Dockerfile

```
FROM ubuntu:14.04
```

```
RUN apt-get update
```

```
RUN apt-get -y install curl
```

```
RUN curl -s http://packages.elasticsearch.org/GPG-KEY-elasticsearch | apt-key add -
```

```
RUN echo "deb http://packages.elasticsearch.org/elasticsearch/1.0/debian stable main" > /etc/apt/sources.list.d/elasticsearch.list
```

```
RUN apt-get update
```

```
RUN apt-get install -y graphite-carbon
```

```
RUN echo "CARBON_CACHE_ENABLED=true" > /etc/default/graphite-carbon
```

```
RUN apt-get install -y graphite-web apache2 apache2-mpm-worker libapache2-mod-wsgi
```

```
RUN sudo -u _graphite graphite-manage syncdb --noinput
```

```
RUN rm -f /etc/apache2/sites-enabled/000-default.conf
```

```
RUN cp /usr/share/graphite-web/apache2-graphite.conf /etc/apache2/sites-enabled/graphite.conf
```

```
RUN apt-get install -y elasticsearch openjdk-7-jre-headless
```

```
RUN update-rc.d elasticsearch defaults
```

```
RUN apt-get install -y nodejs npm
```

```
RUN ln -s /usr/bin/nodejs /usr/local/bin/node
```

```
RUN curl https://code.load.github.com/grafana/tar.gz/v1.7.0 | tar -xz
```

```
RUN mv grafana-1.7.0 /usr/share/grafana
```

```
WORKDIR /usr/share/grafana
```

```
RUN npm install
```

```
RUN node_modules/grunt-cli/bin/grunt
```

```
RUN echo "alias /grafana /usr/share/grafana/src" > /etc/apache2/sites-enabled/grafana.conf
```

```
ADD config.js /usr/share/grafana/src/config.js
```

```
ADD entrypoint.sh /entrypoint.sh
```

```
RUN chmod +x /entrypoint.sh
```

```
ENTRYPOINT ["/entrypoint.sh"]
```

Update 를 왜 두 번 사용할까?

graphite-carbon 패키지를 설치하고 Carbon-Cache를 사용할 수 있도록 설정

graphite-web과 Apache 웹 서버 패키지를 설치합니다.

graphite-manage로 SQLite DB 파일을 생성하고, 기본 Apache 설정을 삭제한 뒤 Graphite Web용 설정 파일을 복사합니다.

OpenJDK와 **Elasticsearch**를 설치하고, **update-rc.d**를 이용하여 서비스로 실행할 수 있도록 설정

nodejs, **npm** 패키지를 설치하고, **/usr/bin/nodejs** 실행

파일을 **/usr/local/bin/node**로 링크합니다. **node**로

링크하지 않으면 **npm install**을 실행할 때 중간에 에러가 발생합니다.

9-1.모니터링 서버 Dockerfile 작성하기

config.js

```
define(['settings'],
function (Settings) {
    "use strict";

    return new Settings({
        datasources: {
            graphite: {
                type: 'graphite',
                url: "http://192.168.0.40",
            },
            elasticsearch: {
                type: 'elasticsearch',
                url: "http://192.168.0.40:9200",
                index: 'grafana-dash',
                grafanaDB: true,
            }
        },
        search: {
            max_results: 20
        },
        default_route: '/dashboard/file/default.json',
        unsaved_changes_warning: true,
        playlist_timespan: "1m",
        admin: {
            password: ''
        },
        plugins: {
            panels: []
        }
    });
});
```

- **graphite**의 **url** 부분에 모니터링 서버의 도메인 또는 IP 주소를 설정합니다. 192.168.0.40은 저의 모니터링 서버 IP 주소이므로 여러분의 IP 주소 또는 도메인으로 설정합니다.
- **elasticsearch**의 **url** 부분에 모니터링 서버의 도메인 또는 IP 주소를 설정하고, 포트 번호는 9200번으로 설정합니다. 192.168.0.40은 저의 모니터링 서버 IP 주소이므로 여러분의 IP 주소 또는 도메인으로 설정합니다.
- **config.js** 파일은 웹 브라우저에서 사용되는 파일이므로 127.0.0.1과 같은 루프백 주소는 설정하면 안됩니다.

Loopback Address

네트워크에서 입출력을 테스트하기 위한 가상 주소이다.

해당 주소로 데이터를 입력하면 발송된 데이터가 중간 계층을 거쳐 되돌아와서 수신된 것처럼 보이지만, 실제로는 네트워크가 연결되어 있지 않기 때문에 네트워크 외부로 데이터가 전달되지 않고 다시 가상의 주소로 전달되어 온 것이다.

웹 서버나 인터넷 소프트웨어의 네트워크 동작 기능을 테스트하는 데에 사용되며, 자신에게 다시 데이터가 돌아오게 하는 루프백 주소는 127.0.0.1이며, 이 주소에 대응되는 호스트 이름을 localhost라고 한다.

9-1.모니터링 서버 Dockerfile 작성하기

entrypoint.sh

컨테이너가 시작될 때 실행될 entrypoint.sh 파일

```
#!/bin/bash
```

```
service carbon-cache start  
service elasticsearch start
```

carbon-cache, elasticsearch, apache2 서비스를 실행

```
apachectl -DFOREGROUND
```

컨테이너를 데몬 모드로 실행하려면 마지막에 실행되는 프로세스가 항상 foreground에 대기 상태가 되어야 합니다. 따라서 `-DFOREGROUND` 옵션을 사용하여 Apache 웹 서버를 foreground로 실행

Dockerfile이 있는 디렉터리로 이동한 뒤 `docker build` 명령으로 이미지를 생성

```
~$ cd dockerbook/Chapter09/Graphite
```

```
~/dockerbook/Chapter09/Graphite$ sudo docker build --tag graphite .
```

9-1.모니터링 서버 Dockerfile 작성하기

Graphite 데이터는 시계열 데이터이기 때문에 시간이 정확해야 합니다. 따라서 `ntpdate` 명령으로 정확한 시간을 설정해줍니다. 호스트에서 시간을 설정하면 컨테이너에는 자동으로 적용됩니다.

```
$ sudo ntpdate time2.kriss.re.kr
```

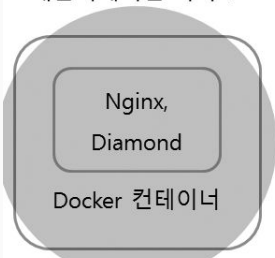
```
$ sudo docker run -d --name graphite -p 2003:2003 -p 9200:9200 -p 80:80 graphite
```

graphite 이미지를 컨테이너로 생성합니다. `-d` 옵션을 사용하여 데몬 모드로 실행하고, 2003, 9200, 80번 포트를 연결한 뒤 외부에 노출합니다.

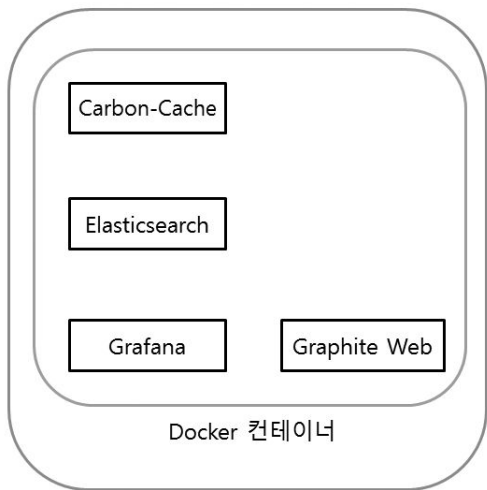
9-2. 애플리케이션 서버 Dockerfile 작성



애플리케이션 서버 1



애플리케이션 서버 2



모니터링 서버

다음 파일들은 저의 **GitHub** 저장소에 있는 예제 파일을 받아서 사용합니다. 애플리케이션 서버를 구축할 것이므로 애플리케이션 서버로 사용할 각 컴퓨터에 받습니다.

```
$ git clone https://github.com/pyrasis/dockerbook.git
```

9-2.애플리케이션 서버 Dockerfile작성

Dockerfile

```
FROM ubuntu:14.04

RUN apt-get update
RUN apt-get install -y nginx \
    git make pbuilder python-mock python-configobj \
    python-support cdb

RUN echo "ndaemon off;" >> /etc/nginx/nginx.conf
RUN chown -R www-data:www-data /var/lib/nginx

WORKDIR /tmp
RUN git clone https://github.com/BrightcoveOS/Diamond.git
RUN cd Diamond && git checkout v3.4 && make deb
RUN dpkg -i Diamond/build/diamond_3.4.0_all.deb
RUN cp /etc/diamond/diamond.conf.example /etc/diamond/diamond.conf

ADD entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]

EXPOSE 80
EXPOSE 443
```

nginx를 데몬이 아닌 **foreground**로 실행하도록 설정합니다.
그리고 **/var/lib/nginx** 디렉터리의 소유자와 그룹을 **www-data**로 설정합니다.

/tmp 디렉터리 아래에 **git**을 사용하여 **Diamond**를 받고, **v3.4** 태그를 체크아웃합니다. 그리고 **make deb** 명령으로 **deb** 패키지 파일을 생성합니다. **dpkg**로 **deb** 파일을 설치할 때 파일명에 버전이 있으므로, 소스도 버전을 맞추어주는 것입니다.

dpkg로 빌드된 **deb** 파일을 설치합니다.

Diamond 예제 설정 파일을 기본 설정 파일(**diamond.conf**)로 복사합니다.

이미지에 파일 추가하고 **/entrypoint.sh** 파일에 실행 권한을 줍니다.

ENTRYPOINT로 컨테이너가 생성될 때 **entrypoint.sh**가 실행되도록 설정

9-2.애플리케이션 서버 Dockerfile작성

entrypoint.sh

```
#!/bin/bash

sed -i "s/host = graphite/host = $GRAPHITE_HOST/g" /etc/diamond/diamond.conf
diamond

cd /etc/nginx
nginx
```

- **/etc/diamond/diamond.conf** 파일의 **host**에 **\$GRAPHITE_HOST** 변수 값을 설정합니다. 이 변수는 **docker run** 명령에서 **-e** 옵션으로 설정할 수 있습니다.
- **diamond**를 실행합니다. 실행하는 즉시 데몬으로 동작됩니다.
- 저는 예제로 **nginx**를 실행했습니다. 이 부분은 나중에 여러분이 작성한 애플리케이션을 실행하면 됩니다.

9-2.애플리케이션 서버 Dockerfile작성

Dockerfile이 있는 디렉터리로 이동한 뒤 `docker build` 명령으로 이미지를 생성합니다.

```
~$ cd dockerbook/Chapter09/Diamond
~/dockerbook/Chapter09/Diamond$ sudo docker build --tag diamond .
```

모니터링 서버와 마찬가지로 애플리케이션 서버도 시간을 정확히 맞추어줍니다

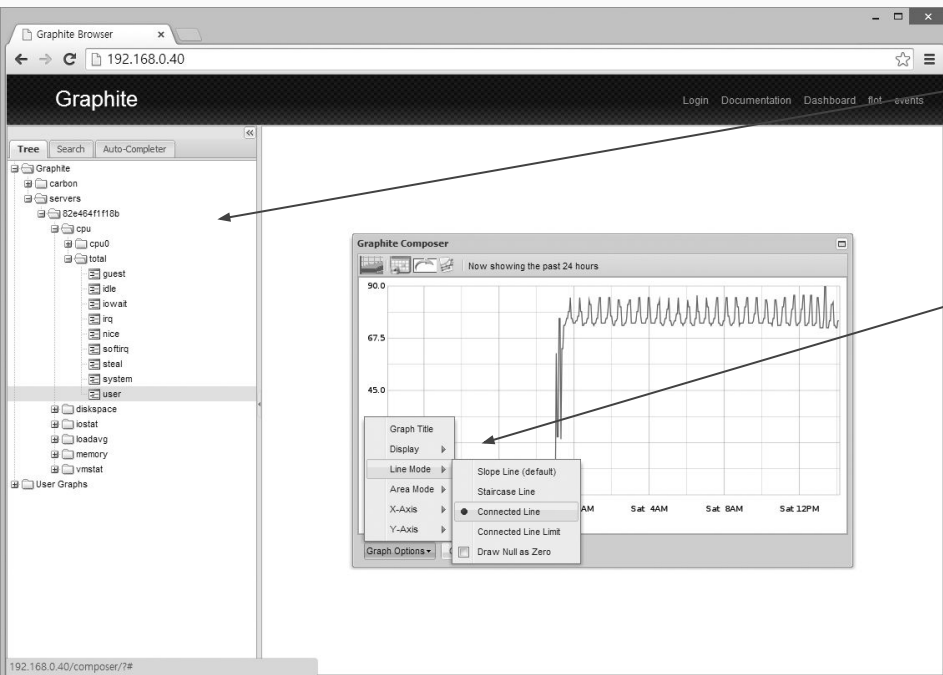
```
$ sudo ntpdate time2.kriss.re.kr
```

diamond 이미지를 컨테이너로 생성합니다. `-d` 옵션을 사용하여 데몬 모드로 실행하고, 80번 포트를 연결한 뒤 외부에 노출합니다. `-e` 옵션으로 `GRAPHITE_HOST` 변수에 모니터링 서버의 IP 주소 또는 도메인을 설정합니다. `192.168.0.40`은 저의 모니터링 서버 IP 주소이므로 여러분의 IP 주소 또는 도메인으로 설정합니다.

```
$ sudo docker run -d --name app1 -p 80:80 -e GRAPHITE_HOST=192.168.0.40 diamond
```

9-3.웹 브라우저에서 그래프 확인

웹 브라우저를 실행하고 모니터링 서버의 IP 주소나 도메인으로 접속합니다.



왼쪽 tree에서 Graphite → servers로 이동하면 방금 애플리케이션 서버에서 생성한 컨테이너의 호스트 이름이 보일 것입니다. 하위 항목을 클릭하면 오른쪽에 해당 항목의 그래프가 표시됩니다.

Graph Options 버튼을 클릭하고 Line Mode에서 Connected Line을 클릭하면 그림 같이 선이 연결된 그래프가 표시됩니다

방금 Diamond를 실행했기 때문에 데이터가 쌓이지 않아서 그래프가 제대로 표시되지 않습니다. 그림은 필자가 14시간 이상 데이터를 수집한 뒤 출력한 그래프입니다.

9-3. 웹 브라우저에서 그래프 확인

CPU, 메모리 사용량 테스트하기

그래프를 출력하기 위해 데이터를 쌓아야 한다면 다음과 같이 `entrypoint.sh`를 수정합니다.

entrypoint.sh

```
#!/bin/bash

sed -i "s/host = graphite/host = $GRAPHITE_HOST/g" /etc/diamond/diamond.conf
diamond

cd /etc/nginx
nginx
```

entrypoint.sh

```
#!/bin/bash

sed -i "s/host = graphite/host = $GRAPHITE_HOST/g" /etc/diamond/diamond.conf
diamond

apt-get install -y wget make gcc
wget http://nodejs.org/dist/v0.10.31/node-v0.10.31.tar.gz
tar vxzf node-v0.10.31.tar.gz
cd node-v0.10.31

while :
do
    make
    make test
    make clean
done
```

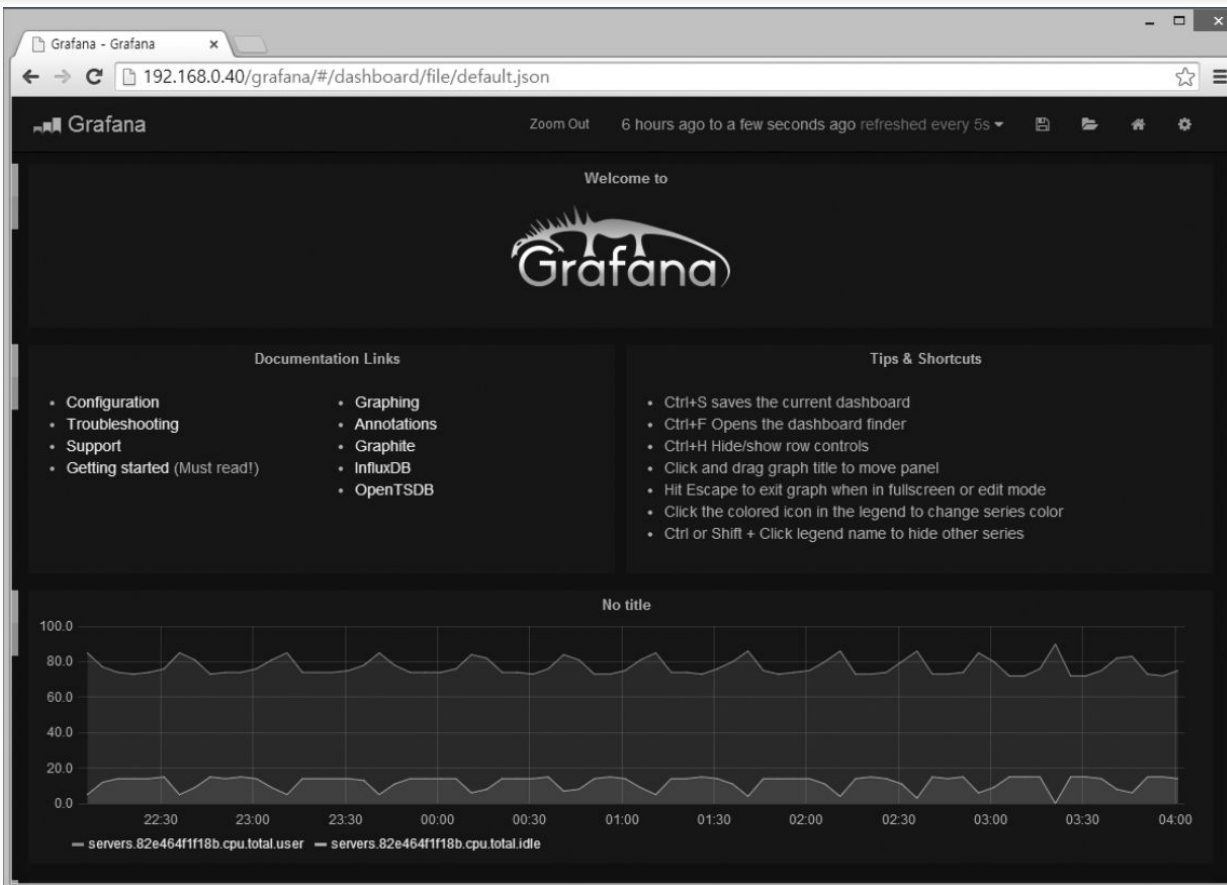
Node.js 소스를 받은 뒤 컴파일하는 예제입니다. 이미지를 다시 생성한 뒤 컨테이너로 생성합니다. 몇 시간 동안 컨테이너를 실행하면 CPU, 메모리 사용량 데이터를 수집할 수 있습니다.

```
$ sudo docker build --tag compile .
$ sudo docker run -i -t --rm -e GRAPHITE_HOST=192.168.0.40 compile
```

9-3. 웹 브라우저에서 그래프 확인

웹 브라우저에서 <모니터링 서버의 IP 주소 또는 도메인>/grafana로 접속합니다.

Grafana에서도 수집한 데이터의 그래프를 볼 수 있습니다. Graphite Web과 마찬가지로 필자가 14시간 이상 데이터를 수집한 뒤 출력한 그래프입니다.



13.Docker Hub 사용하기

Docker Hub는 Docker 공식 이미지를 제공하고, 사용자들끼리 이미지를 공유하는 사이트입니다.

- 공개 저장소(Public Repository): 저장된 이미지를 다른 사람들도 사용할 수 있습니다. 또한, 개수 제한 없이 무료로 생성할 수 있습니다.
- 개인 저장소(Private Repository): 저장된 이미지를 혼자만 사용할 수 있습니다. 1개까지 무료로 생성할 수 있고 그 이상부터는 유료입니다.
- 즐겨 찾기(Starred): 다른 사람의 유용한 저장소를 다음에 사용할 수 있도록 즐겨 찾기에 추가할 수 있습니다.
- Automated Build: GitHub 또는 BitBucket 저장소와 연동하여 Dockerfile을 Push했을 때 이미지를 자동으로 빌드하는 기능입니다.

가입과정 생략

<https://hub.docker.com/>

13-2.Push 명령으로 이미지 올리기

먼저 **example-nginx** 디렉토리를 생성하고, **Dockerfile**을 작성합니다.
지금까지 계속 사용해왔던 Nginx 예제입니다.

```
~$ mkdir example-nginx
~$ cd example-nginx
```

~/example-nginx/Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Foo Bar <exampleuser@example.com>

RUN apt-get update
RUN apt-get install -y nginx
RUN echo "\ndaemon off;" >> /etc/nginx/nginx.conf
RUN chown -R www-data:www-data /var/lib/nginx

VOLUME ["/data", "/etc/nginx/site-enabled", "/var/log/nginx"]

WORKDIR /etc/nginx

CMD ["nginx"]

EXPOSE 80
EXPOSE 443
```

docker build 명령으로 이미지를 생성합니다.

```
~/example-nginx$ sudo docker build --tag exampleuser/example-nginx:0.1 .
```

13-2.Push 명령으로 이미지 올리기

- Docker Hub에 이미지를 올리려면 이미지 이름을 <Docker Hub 사용자 계정>/<이미지 이름>:<태그> 형식으로 생성해야 합니다.
- 아무 사용자 이름이나 사용할 수 있지만 내 계정 이름과 일치해야 이미지를 올릴 수 있습니다.
- 태그를 지정하지 않으면 **latest**가 됩니다.

Docker Hub에 이미지를 올리기 전에 로그인부터 합니다. 사용자 계정, 비밀번호, 가입할 때 사용한 이메일 주소를 입력합니다.

```
~/example-nginx$ sudo docker login
Username: exampleuser
Password:
Email: exampleuser@example.com
Login Succeeded
```

이제 `docker push` 명령으로 이미지를 올립니다.

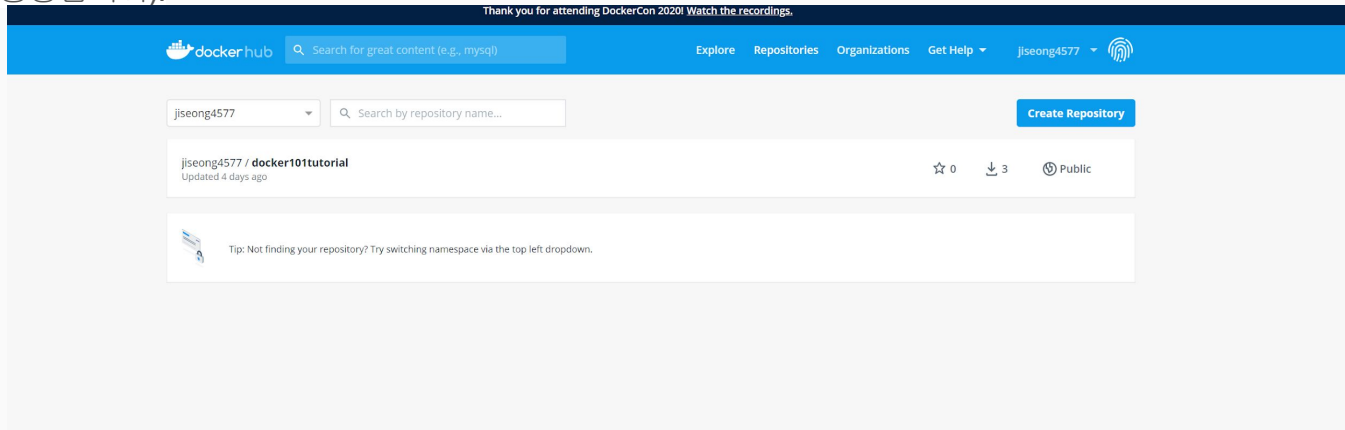
```
~/example-nginx$ sudo docker push exampleuser/example-nginx:0.1
```

`docker push <Docker Hub 사용자 계정>/<이미지 이름>:<태그>` 형식입니다.

13-2.Push 명령으로 이미지 올리기

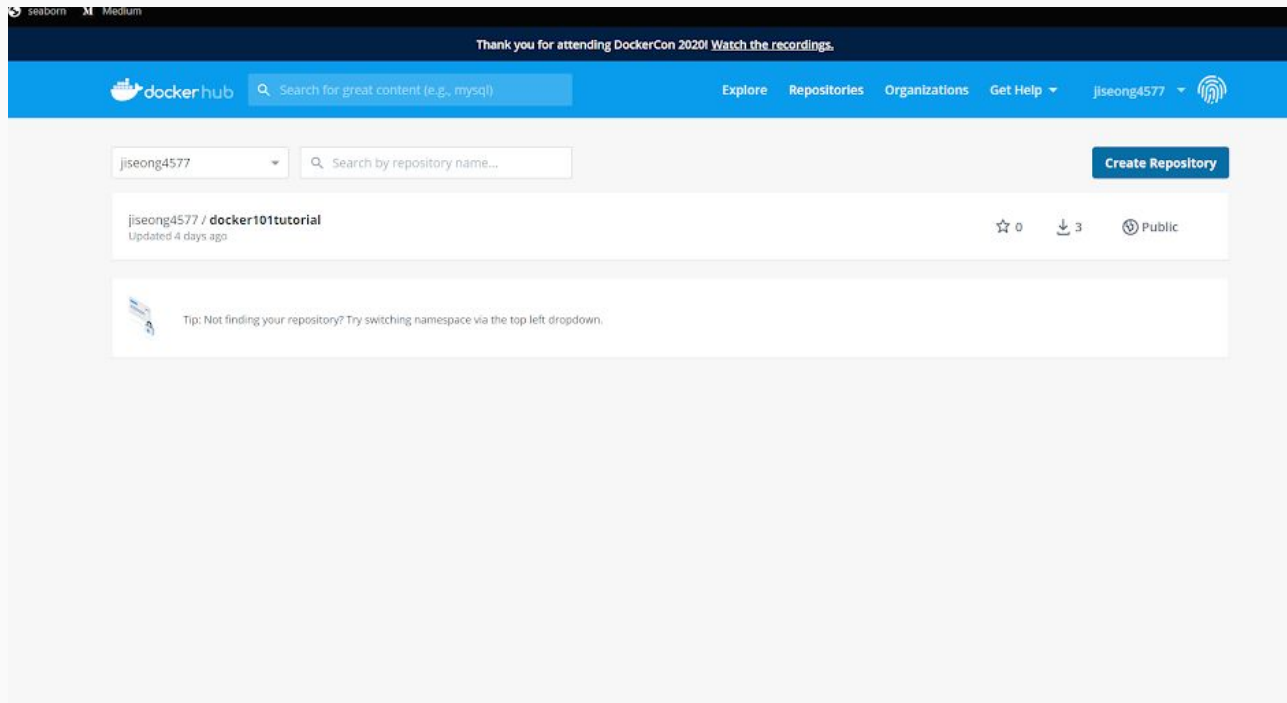
잠시 기다리면 이미지가 Docker Hub의 공개 저장소에 완전히 올라갑니다. Docker Hub 페이지에서 Repositories 메뉴를 클릭하면 방금 올렸던

<Docker Hub 사용자 계정>/example-nginx 이미지가 표시됩니다(Docker Hub에 생성된 공개 저장소가 없다면 이미지를 올렸을 때 공개 저장소가 자동으로 생성됩니다).



그림에서 Create Repository 버튼을 통해 공개 저장소를 미리 만든 뒤 `docker push` 명령으로 이미지를 올려도 됩니다. 이제 다른 사람들이 `docker pull <Docker Hub 사용자 계정>/example-nginx` 명령으로 **example-nginx** 이미지를 사용할 수 있습니다

13-3.개인 저장소 생성하기



Create Repository 누르기

13-3.개인 저장소 생성하기

Repositories

Create

Create Repository

jiseong4577

Name

Description

Visibility

Using 0 of 1 private repositories. [Get more](#)

☐ Public


Public repositories appear in Docker Hub search results

☒ Private


Only you can view private repositories


Build Settings (optional)

Autobuild triggers a new build with every **git push** to your source code repository. [Learn More](#)

 Please re-link a GitHub or Bitbucket account

We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. [Learn More](#)

 Disconnected

 Disconnected

Cancel

Create

Create & Build

Private 로 생성하고

```
~/example-private$ sudo docker push exampleuser/example-private:0.1
```

공개 저장소에 **pull**하는 것과 동일하게
실행하면 됩니다!

13-4.Docker Hub Automated Build 활용하기

Docker Hub는 GitHub과 BitBucket을 연동하여 이미지를 자동 빌드하는 기능을 제공합니다.

GitHub을 기준으로 설명하겠습니다.


먼저 GitHub에서 Dockerfile을 저장할 저장소부터 생성합니다.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
[Import a repository.](#)

Owner *


Repository name *

 zi-seong ▼


 /

Great repository names are short and memorable. Need inspiration? How about [vigilant-octo-dollop](#)?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.


Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▼

Add a license: **None** ▼



Create repository

13-4.Docker Hub Automated Build 활용하기

DockerHub에서 create repository하면 Build Setting이 있는데 거기서 **disconnected**되어있는 아이콘을 클릭하면(**connected**상태라면 아이콘 클릭해서 바로 설정가능)페이지가 전환되고 **connect** 누르면 로그인해서 연결가능

The image shows two parts of the Docker Hub interface. On the left is the 'Create Repository' page, and on the right is the user's account settings page.

Create Repository Page:

- Header: docker hub, Search for great content (e.g., mysql), Explore, Repositories
- Sub-headers: Repositories, Create
- Form: Create Repository
- Fields: Repository name (jseong4577), Description, Visibility (Public/Private), Build Settings (optional).
- Buttons: Cancel, Create, Create & Build.
- Icons: Connected (blue), Disconnected (grey).

Account Settings Page:

- Header: jseong4577, User, joined July 13, 2020
- Menu: General, Linked Accounts, Security, Default Privacy, Notifications, Convert Account, Deactivate Account.
- Section: Linked Accounts
- Text: These account links are used for Automated Builds, so that Docker Hub can access your project lists and help you configure your Automated Builds. **Please note: A Github/Bitbucket account can only be connected to one Docker Hub account at a time.**
- Table:

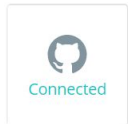
Provider	Account	Refresh	Disconnect	Connect
GitHub	zi-seong			
Bitbucket	No account linked			

A red box highlights the 'Disconnected' icon in the 'Create Repository' page, and an arrow points from it to the 'Connect' button in the 'Account Settings' page.

13-4.Docker Hub Automated Build 활용하기

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository. [Learn More.](#)



Select organization ▼

Select repository ▼

BUILD RULES +

The build rules below specify how to build your source into Docker images.

▶ View example build rules

Cancel

Create

Create & Build

연결하고 **Connected** 되어있는 아이콘 클릭하면
아이디 선택하고 **repository** 선택 가능합니다

13-4.Docker Hub Automated Build 활용하기

앞에서 만든 GitHub 저장소를 받은 뒤 저장소 디렉터리로 이동합니다.

```
~$ git clone git@github.com:<GitHub 사용자 계정>/example-nginx2.git
~$ cd example-nginx2
```

Git 저장소 디렉터리에 Dockerfile로 저장합니다.

~/example-nginx2/Dockerfile

```
FROM ubuntu:14.04
MAINTAINER Foo Bar <exampleuser@example.com>

RUN apt-get update
RUN apt-get install -y nginx
RUN echo "\ndaemon off;" >> /etc/nginx/nginx.conf
RUN chown -R www-data:www-data /var/lib/nginx

VOLUME ["/data", "/etc/nginx/site-enabled", "/var/log/nginx"]

WORKDIR /etc/nginx

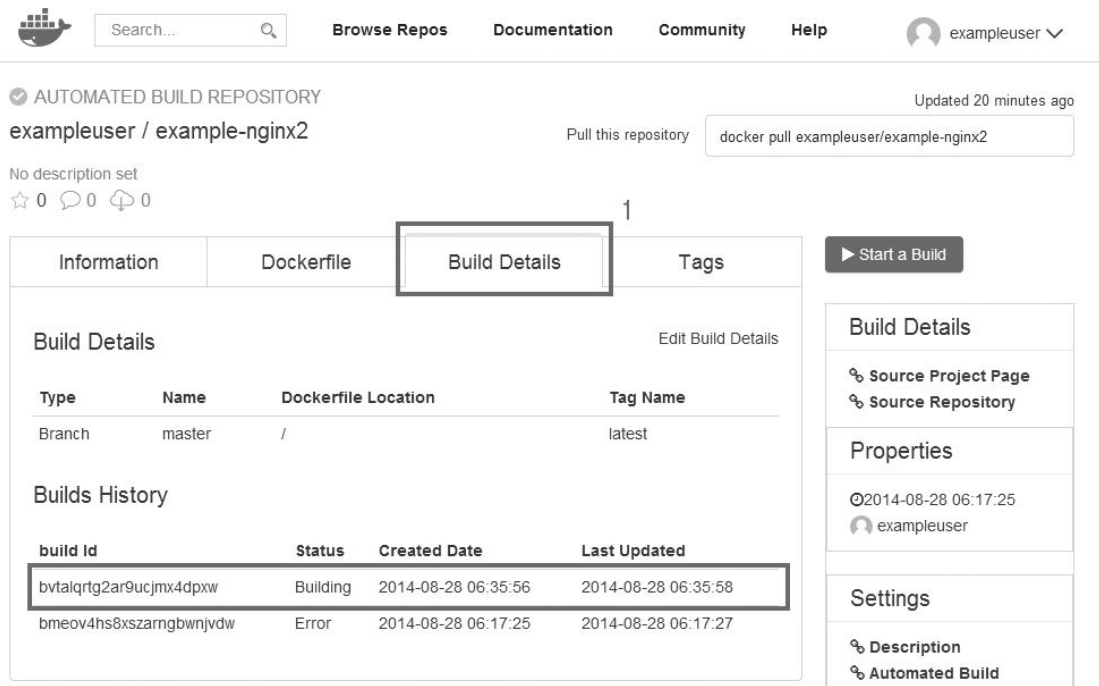
CMD ["nginx"]

EXPOSE 80
EXPOSE 443
```

Git 저장소에 커밋한 뒤 GitHub에 Push합니다.

```
~/example-nginx2$ git add Dockerfile
~/example-nginx2$ git commit -m "add Dockerfile"
~/example-nginx2$ git push origin master
```

13-4.Docker Hub Automated Build 활용하기



Automated Build Repository

exampleuser / example-nginx2

Updated 20 minutes ago

Pull this repository: `docker pull exampleuser/example-nginx2`

No description set

☆ 0 💬 0 🔗 0

1

Information	Dockerfile	Build Details	Tags
Build Details Edit Build Details			
Type	Name	Dockerfile Location	Tag Name
Branch	master	/	latest
Builds History			
build Id	Status	Created Date	Last Updated
bvtalqrg2ar9ucjmx4dpxw	Building	2014-08-28 06:35:56	2014-08-28 06:35:58
bmeov4hs8xsarngbwnjvdw	Error	2014-08-28 06:17:25	2014-08-28 06:17:27

[Start a Build](#)

Build Details

- [Source Project Page](#)
- [Source Repository](#)

Properties

🕒 2014-08-28 06:17:25

👤 exampleuser

Settings

- [Description](#)
- [Automated Build](#)

웹 브라우저에서 Docker Hub **example-nginx2** 저장소 페이지로 이동합니다. **Build Details** 탭을 클릭하면 방금 push한 Dockerfile이 빌드 중으로 표시됩니다.

잠시 기다리면 Status가 Finished로 바뀝니다. 이제 GitHub에 커밋할 때마다 Docker Hub에 새 이미지가 생성됩니다.

13-4.Docker Hub Automated Build 활용하기

이번에는 Git 태그와 Docker 이미지 태그를 설정해보겠습니다. 먼저 Git 저장소에서 태그를 설정한 뒤 태그를 Push합니다.

```
~/example-nginx2$ git tag -a 0.1 -m "Version 0.1"
~/example-nginx2$ git push origin 0.1
```

The screenshot shows the Docker Hub interface for the repository 'example-nginx2'. At the top, there's a navigation bar with 'Browse Repos', 'Documentation', 'Community', and 'Help'. Below the repository name, it says 'Automated Build Repository' and 'Updated 20 minutes ago'. There's a 'Pull this repository' button with the command 'docker pull exampleuser/example-nginx2'. The main content area has tabs for 'Information', 'Dockerfile', 'Build Details', and 'Tags'. The 'Build Details' tab is active, showing a table with columns 'Type', 'Name', 'Dockerfile Location', and 'Tag Name'. The table has one row: 'Branch', 'master', '/', 'latest'. Below the table is a 'Builds History' section with a table showing build details. On the right side, there's a sidebar with 'Build Details', 'Properties', and 'Settings' sections.

Type	Name	Dockerfile Location	Tag Name
Branch	master	/	latest

build Id	Status	Created Date	Last Updated
bvtalqrtg2ar9ucjmx4dpxw	Building	2014-08-28 06:35:56	2014-08-28 06:35:58
bmeov4hs8xszarngbnjvjd	Error	2014-08-28 06:17:25	2014-08-28 06:17:27

- 버튼을 클릭하면 설정이 추가됩니다.
- **Type:** Git 브랜치 형식입니다. **Branch**와 **Tag**를 선택할 수 있습니다. 여기서는 **Tag**를 선택합니다.
- **Name:** Git 브랜치, 태그 이름입니다. **0.1**을 입력합니다.
- **Dockerfile Location:** 저장소에서 **Dockerfile**이 있는 경로입니다. 기본값 그대로 사용합니다.
- **Docker Tag Name:** Docker 태그 이름입니다. **0.1**을 입력합니다.

13-4.Docker Hub Automated Build 활용하기

Edit Automated Build settings for exampleuser01/example-nginx2

Repo Name:
exampleuser/example-nginx2

You can't change the Repo Name after the build has been created


Active:
☒ When active we will build when new pushes occur



Type	Name	Dockerfile Location	Docker Tag Name
Branch	master	/	latest
Tag	0.1	/	0.1


Save and trigger build

Build Details 탭에 태그 **0.1**이 추가되었습니다. 빌드 설정을 수정하면 모든 태그를 다시 빌드하기 때문에 그림처럼 **latest**와 **0.1** 두 개를 빌드합니다.

13-4.Docker Hub Automated Build 활용하기



[Browse Repos](#) [Documentation](#) [Community](#) [Help](#)  [exampleuser](#) 

 AUTOMATED BUILD REPOSITORY

exampleuser / example-nginx2

Updated 13 minutes ago

Pull this repository

docker pull exampleuser/example-nginx2

No description set
☆ 0 💬 0 📄 0

Configuration saved, and a build was triggered for exampleuser01/example-nginx2. Check back in a few minutes for the results!

▶ Start a Build

Information

Dockerfile

Build Details

Tags

Build Details

Edit Build Details

Type	Name	Dockerfile Location	Tag Name
Tag	0.1	/	0.1
Branch	master	/	latest


Builds History

build Id	Status	Created Date	Last Updated
bdhngxnu5snxjlsbcmvgmvpe	Building	2014-08-28 06:51:41	2014-08-28 06:54:10
bmyvsrrfpisb8nmsppwppwd	Building	2014-08-28 06:51:40	2014-08-28 06:54:30
bvtalqrg2ar9ucjmx4dpxw	Finished	2014-08-28 06:35:56	2014-08-28 06:38:47
bmeov4hs8xszarngbnjvdw	Error	2014-08-28 06:17:25	2014-08-28 06:17:27

Build Details

[Source Project Page](#)
[Source Repository](#)

Properties

🕒 2014-08-28 06:17:25
 exampleuser

Settings

[Description](#)
[Automated Build](#)
[Webhooks](#)
[Collaborators](#)
[Build Triggers](#)
[Repository Links](#)
[Mark as unlisted](#)

[Make Private](#)

잠시 기다리면 Status가 Finished로 바뀌고 **example-nginx2:0.1** 이미지도 사용할 수 있게 됩니다.

앞으로 Dockerfile을 수정한 뒤 Git으로 브랜치, 태그를 Push하면 해당 이미지는 새로 생성됩니다. 단 Build Details에서 설정하지 않은 Git 브랜치, 태그는 Push해도 이미지가 생성되지 않습니다.