

Docker로 애플리케이션 배포하기

목차

1. 서버 한 대에 애플리케이션 배포하기

1. 개발자 PC에서 Git 설치 및 저장소 생성하기
2. 개발자 PC에서 Node.js로 웹 서버 작성하기
3. 개발자 PC에서 Dockerfile 작성하기
4. 개발자 PC에서 SSH키 생성하기
5. 서버에 Git 설치 및 저장소 생성하기
6. 서버에 Docker 설치하기
8. 서버에 Git Hook 설정하기
9. 개발자 PC에서 소스 Push하기

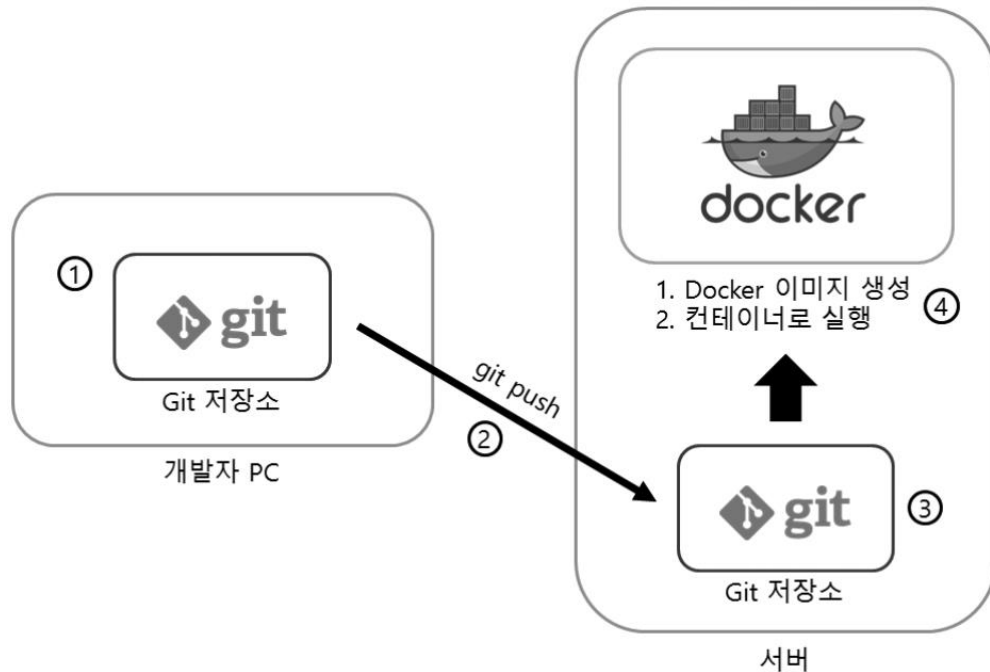
2. 서버 여러 대에 애플리케이션 배포하기

1. 개발자 PC에서 Git 설치 및 저장소 생성하기
2. 개발자 PC에서 Node.js로 웹 서버 작성하기
3. 개발자 PC에서 Dockerfile 작성하기
4. 개발자 PC에서 SSH키 생성하기
5. 배포 서버에 Git 설치 및 저장소 생성하기
6. 배포 서버에 SSH 키 생성하기
7. 배포 서버에 Docker 설치하기
8. 배포 서버에 Docker 레지스트리 서버 설정하기
9. 배포 서버에 SSH 키 설정하기
10. 배포 서버에 Git Hook 설정하기
11. 애플리케이션 서버에 Docker 설치하기
12. 애플리케이션 서버에 SSH 키 설정하기
13. 개발자 PC에서 소스 Push하기

1. 서버 여러 대에 애플리케이션 배포하기

먼저 서버 한 대에 애플리케이션을 배포하는 방법입니다.

1. 개발자의 PC에서 애플리케이션을 개발한다.
2. `git push` 명령으로 소스를 서버에 올린다.
3. 서버에서는 저장소에 `git push` 명령이 발생하면 `git hook`을 실행시킨다.
4. `git hook`에서 Docker 이미지를 생성하고, 이미지를 컨테이너로 실행한다.



1. 서버 한 대에 애플리케이션 배포하기

1. 개발자 PC에서 Git 설치 및 저장소 생성하기

우분투

```
$ sudo apt-get install git
```

CentOS

```
$ sudo yum install git
```

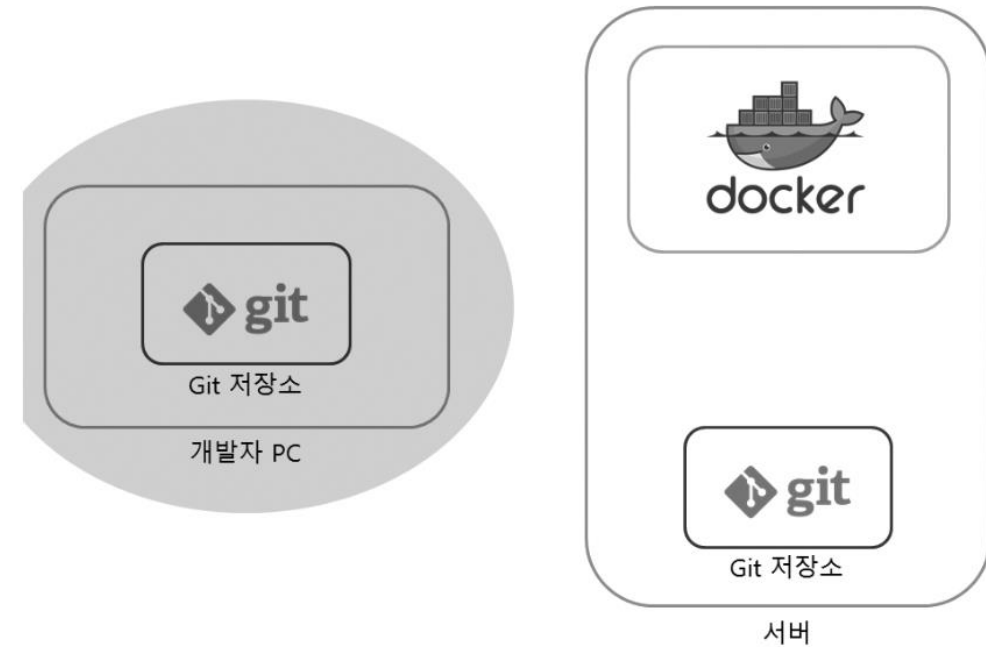
이제 Windows에서는 Git Bash를 실행합니다(Git Bash에서는 유닉스/리눅스 방식의 명령을 실행할 수 있으므로 지금부터 나오는 명령은 Mac OS X, 리눅스, Windows에서 모두 동일합니다). Mac OS X는 터미널을 실행하고, 리눅스에서는 터미널에서 그대로 진행합니다.

Git 저장소를 생성하고, 저장소 디렉터리로 이동합니다.

```
~$ git init exampleapp  
~$ cd exampleapp
```

`git config` 명령으로 자신의 이메일과 이름을 설정합니다.

```
~/exampleapp$ git config --global user.email gd@yuldo.com  
~/exampleapp$ git config --global user.name "Hong, Gildong"
```



1. 서버 한 대에 애플리케이션 배포하기

2. 개발자 PC에서 Node.js로 웹 서버 작성하기

~/exampleapp/app.js

```
var express = require('express');
var app = express();

app.get(['/', '/index.html'], function (req, res) {
  res.send('Hello Docker');
});

app.listen(80);
```

Node.js npm 패키지 사용을 위해 다음과 같이 작성한 뒤 package.json로 저장합니다.

~/exampleapp/package.json

```
{
  "name": "exampleapp",
  "description": "Hello Docker",
  "version": "0.0.1",
  "dependencies": {
    "express": "4.4.x"
  }
}
```

git add, git commit 명령으로 개발자 PC의 exampleapp 저장소에 파일을 커밋합니다.

```
~/exampleapp$ git add app.js package.json
~/exampleapp$ git commit -m "add source"
```

1. 서버 한 대에 애플리케이션 배포하기

3. 개발자 PC에서 Dockerfile 작성하기

서버에서 Docker 이미지를 생성할 수 있도록 개발자 PC에서 Dockerfile을 작성합니다. 다음 내용을 Dockerfile로 저장합니다.

~/exampleapp/Dockerfile

```
FROM ubuntu:14.04

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

ADD app.js /var/www/app.js
ADD package.json /var/www/package.json

WORKDIR /var/www
RUN npm install

CMD nodejs app.js
```

- FROM으로 ubuntu:14.04를 기반으로 이미지를 생성하도록 설정합니다.
- RUN으로 nodejs, npm 패키지를 설치합니다.
- ADD로 app.js와 package.json을 이미지의 /var/www 디렉터리에 복사합니다.
- WORKDIR로 실행 디렉터를 /var/www로 변경합니다. 그리고 RUN으로 `npm install` 명령을 실행하여 package.json에 설정된 Node.js 모듈을 설치합니다.
- CMD로 컨테이너가 시작되면 nodejs를 이용하여 app.js를 실행하도록 설정합니다(우분투에서 Node.js를 패키지로 설치하면 실행 파일은 node가 아닌 nodejs입니다).

Dockerfile도 개발자 PC의 exampleapp 저장소에 커밋합니다.

```
~/exampleapp$ git add Dockerfile
~/exampleapp$ git commit -m "add Dockerfile"
```

1. 서버 한 대에 애플리케이션 배포하기

4. 개발자 PC에서 SSH키 생성하기

개발자 PC에서 ssh-keygen 명령을 실행하여 SSH 키를 생성합니다(Windows는 Git Bash에서 다음 명령을 실행합니다).

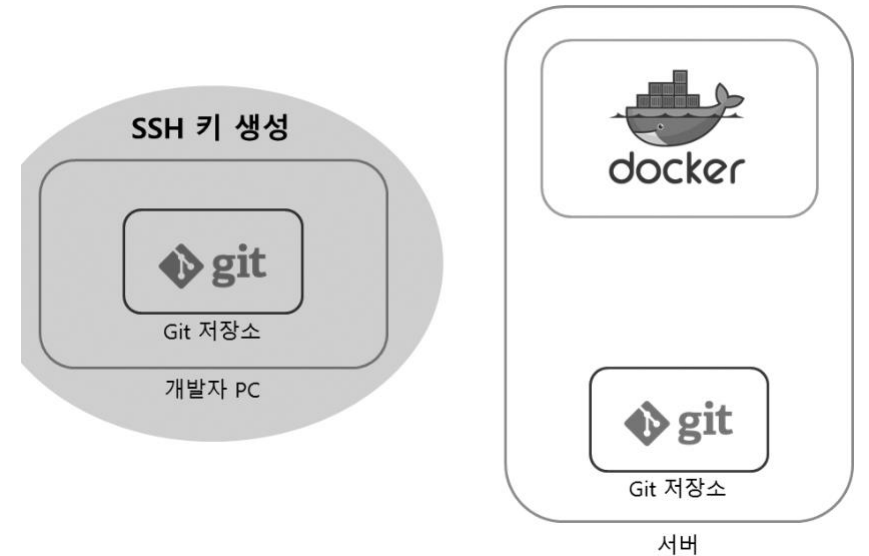
- Enter passpEnter file in which to save the key: 기본값 그대로 사용합니다.
 - hrse, Enter same passphrase again: 엔터를 입력하여 비밀번호를 설정하지 않습니다.
- 이미 /home/<사용자 계정>/.ssh 디렉터리에 id_rsa, id_rsa.pub 파일이 있다면 이 부분은 건너뜁니다.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pyrasis/.ssh/id_rsa):
Created directory '/home/pyrasis/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pyrasis/.ssh/id_rsa.
Your public key has been saved in /home/pyrasis/.ssh/id_rsa.pub.
The key fingerprint is:
64:9e:5a:8a:0b:93:f0:c5:fb:89:41:31:c4:bb:a2:ab pyrasis@ubuntu
The key's randomart image is:
+--[ RSA 2048 ]-----+
|  ..                    |
|  ..                    |
|   o. o                 |
|  ..o+.                 |
| .+. S                  |
| o.+o+                  |
| .+.o                   |
| . o =                  |
| E. oo                  |
+-----+

```

이제 /home/<사용자 계정>/.ssh 디렉터리에 id_rsa, id_rsa.pub 파일이 생성되었습니다.

```
heechul@Ubuntu-20:~/ssh$ ls
id_rsa id_rsa.pub
```



1. 서버 한 대에 애플리케이션 배포하기

5. 서버에 Git 설치 및 저장소 생성하기

이제 서버를 설정할 차례입니다. Docker가 리눅스 전용이므로 리눅스 서버에서 작업하도록 합니다.

서버에도 Git을 설치합니다.

우분투

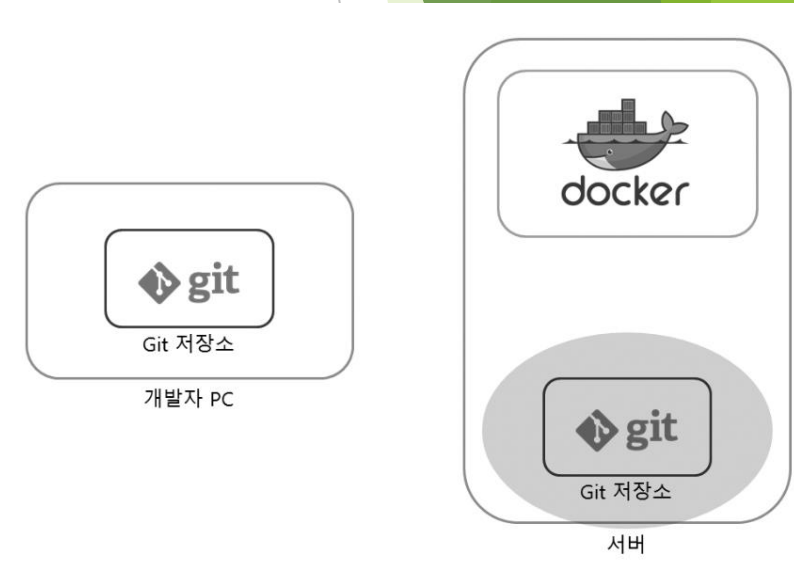
```
$ sudo apt-get install git
```

CentOS

```
$ sudo yum install git
```

현재 리눅스 계정의 홈 디렉터리(/home/<사용자 계정>)에 exampleapp 저장소를 생성합니다. 그리고 개발자 PC에서 push한 소스를 받을 수 있도록 receive.denycurrentbranch를 ignore로 설정합니다.

```
~$ git init exampleapp
~$ cd exampleapp
~/exampleapp$ git config receive.denycurrentbranch ignore
```



1. 서버 한 대에 애플리케이션 배포하기

6. 서버에 Docker 설치하기

서버에서 사용할 Docker를 설치합니다. CentOS에서 EPEL 설치 방법은 [‘2.1.3 RedHat Enterprise Linux, CentOS’](#)를 참조하기 바랍니다.

우분투

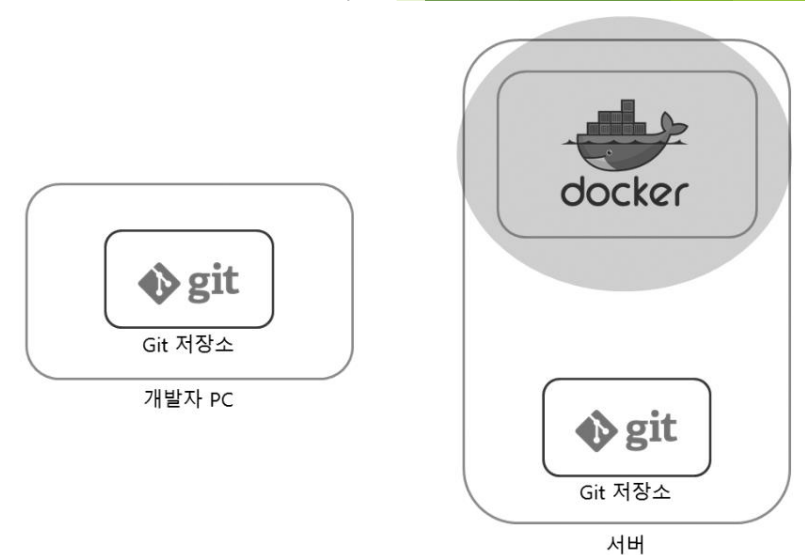
```
$ sudo apt-get install docker.io  
$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
```

CentOS

```
$ sudo yum install docker-io  
$ sudo service docker start
```

docker 명령을 `sudo` 없이 root 권한으로 사용할 수 있도록 현재 리눅스 계정을 `docker` 그룹에 추가합니다.

```
$ sudo usermod -aG docker ${USER}  
$ sudo service docker restart
```



1. 서버 한 대에 애플리케이션 배포하기

7. 서버에 SSH 키 설정하기

개발자 PC에서 비밀번호 없이 서버에 접속할 수 있도록 앞에서 생성한 SSH 키를 설정합니다.

서버의 `/home/<서버 사용자 계정>` 디렉터리 아래에 `.ssh` 디렉터리를 생성하고, 권한을 설정합니다.

```
~$ mkdir .ssh
~$ chmod 700 .ssh
```

방금 생성한 `.ssh` 디렉터리에 `authorized_keys` 파일을 생성합니다. 그리고 개발자 PC에서 생성한 `id_rsa.pub` 파일의 내용을 복사해서 `authorized_keys` 파일 붙여넣습니다.

```
~/ssh/authorized_keys
```

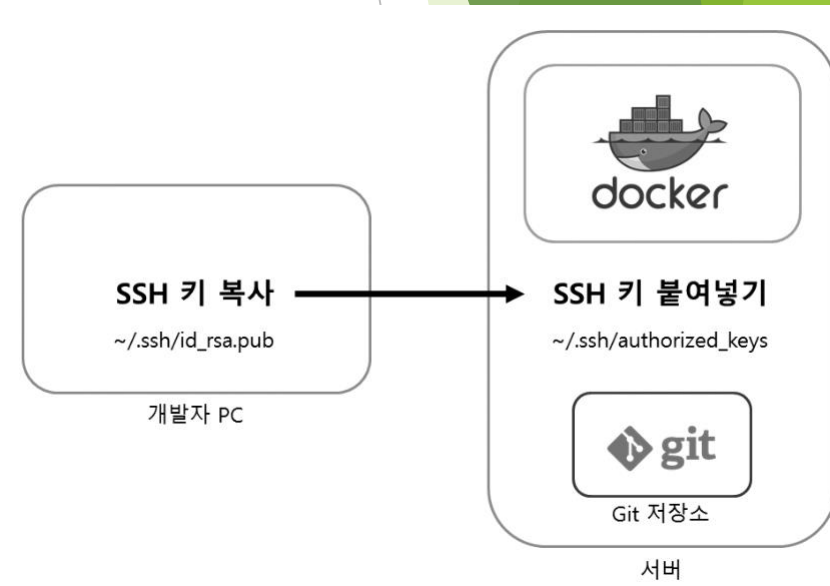
```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCA4nkKAHdB8gU9DT9te9HDNWA8qTY/9YjgxVr493YxqS3vu+Y5/UyXgRPMEgRZGKWZEDtyssYi/X
xQ5Rlk0xXF8NE/T/x8DULVc32e3mtA9vznOMqsHjVLqP0ZZXdhkipEw6s2uKJVtmXQdN+Pz3Dupyj+1a0jww/n3y62goEZ2f9jr7ZnGtL2haYSZJE
Mh57RVAYwLW3n1Ax53dhKE9ha9xdBC+BRTjkqE8oEq+Vg67H01604kxnRvubiVDIXfm1/mvXqz+GdgzSta8Uspz59Tth01J0cJbAZydL5VEBYV5rz
iHvRmqwDewV0Mn7hZjAuVYDlGPMbW26/hXsdxyCMJ pyrasis@ubuntu
```

위의 내용은 저의 공개 키입니다. 여러분의 공개 키(`id_rsa.pub`)를 설정하기 바랍니다.

`authorized_keys` 파일도 권한을 설정합니다.

```
$ chmod 600 authorized_keys
```

이제 개발자 PC에서 비밀번호 없이 `git push` 명령을 사용할 수 있습니다.



1. 서버 한 대에 애플리케이션 배포하기

8. 서버에 Git Hook 설정하기

개발자 PC에서 **git push** 명령으로 소스를 올렸을 때 Docker 이미지와 컨테이너를 생성하도록 Git Hook을 설정합니다.

다음 내용을 서버의 `/home/<서버 사용자 계정>/exampleapp/.git/hooks` 디렉터리에 `post-receive`로 저장합니다.

- [dockerbook/Chapter08/SingleServerDeployment/hooks/post-receive](#)

~/exampleapp/.git/hooks/post-receive

```
#!/bin/bash

APP_NAME=exampleapp
APP_DIR=$HOME/$APP_NAME
REVISION=$(expr substr $(git rev-parse --verify HEAD) 1 7)

GIT_WORK_TREE=$APP_DIR git checkout -f

cd $APP_DIR
docker build --tag $APP_NAME:$REVISION .
docker stop $APP_NAME
docker rm $APP_NAME
docker run -d --name $APP_NAME -p 80:80 $APP_NAME:$REVISION
```

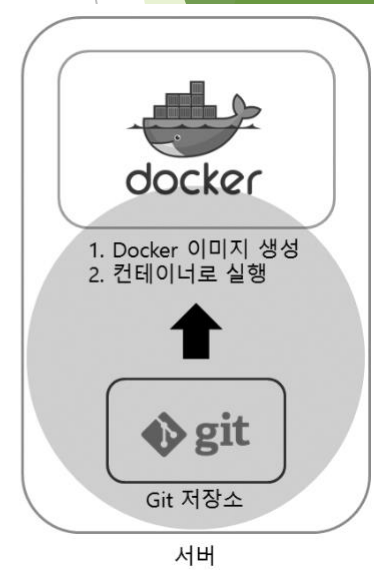
- `APP_NAME` 변수에 현재 애플리케이션의 이름을 설정합니다. 이 이름은 저장소 이름과 같아야 합니다.
- `APP_DIR` 변수에 저장소 디렉터리 경로를 설정합니다. 셸에서 `HOME` 변수에는 사용자 홈 디렉터리 경로 (`/home/<사용자 계정>`)가 들어있습니다.
- `git rev-parse` 명령으로 Push된 최신 리비전을 구한 뒤 앞의 7자리만 `REVISION` 변수에 저장합니다.
- `git checkout -f` 명령으로 Push된 소스를 저장소 디렉터리에 받습니다. 반드시 `GIT_WORK_TREE` 변수를 설정해 주어야 합니다.
- `exampleapp` 저장소 디렉터리로 이동합니다.
- `docker build` 명령으로 이미지를 생성합니다. 이미지 이름은 `exampleapp`이고 태그에는 방금 구한 Git 리비전을 설정합니다.
- `docker stop`, `rm` 명령으로 현재 실행되고 있는 Docker 컨테이너를 정지한 뒤 삭제합니다.
- `docker run` 명령으로 새로 만든 이미지를 컨테이너로 생성하고, `-p` 옵션으로 80번 포트를 연결하고 외부에 노출합니다. 이미지 이름은 `exampleapp`이고 태그에는 앞에서 구한 Git 리비전을 설정합니다.

`post-receive` 파일에 실행 권한을 부여합니다.

```
~/exampleapp/.git/hooks$ chmod +x post-receive
```



개발자 PC



서버

1. 서버 한 대에 애플리케이션 배포하기

9. 개발자 PC에서 소스 Push하기

이제 개발자 PC로 돌아옵니다.

exampleapp 저장소 디렉터리로 이동한 뒤 `git remote add` 명령으로 origin 주소를 설정합니다.

```
~/exampleapp$ git remote add origin <서버 사용자 계정>@<서버 IP 주소 또는 도메인>:exampleapp
```

예를 들면 서버 사용자 계정이 pyrasis이고, 서버 IP 주소가 192.168.0.40이면 origin 주소는 pyrasis@192.168.0.40:exampleapp이 됩니다.

`git push` 명령으로 소스를 서버에 올립니다.

```
~/exampleapp$ git push origin master
```

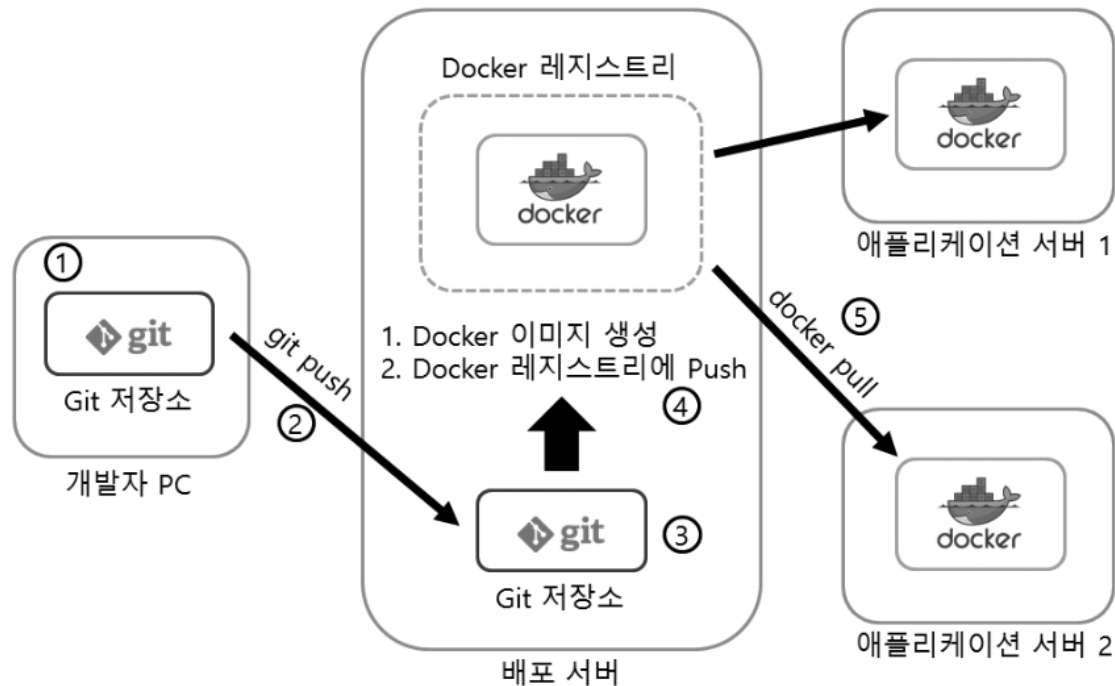
`git push` 명령 출력 결과에서 Docker 이미지와 컨테이너가 생성되는 모습을 볼 수 있습니다. `git push` 명령이 완전히 끝나면 웹 브라우저를 실행하고 서버 IP 주소로 접속합니다.

웹 브라우저에서 app.js에서 출력한 Hello Docker가 표시됩니다. 이제 소스를 수정한 뒤 서버에 Push하면 새 Docker 컨테이너가 생성됩니다. Dockerfile과 post-receive 파일을 각자 상황에 맞게 수정하여 사용하면 됩니다.

2. 서버 여러 대에 애플리케이션 배포하기

이번에는 서버 여러 대에 애플리케이션을 배포하는 방법입니다. 서버 한 대에 Docker 이미지를 생성할 때와는 달리 Docker 이미지를 여러 서버에 전달해야 하기 때문에 위해 Docker 레지스트리 서버를 구축해야 합니다.

1. 개발자의 PC에서 애플리케이션을 개발한다.
2. `git push` 명령으로 소스를 배포 서버에 올린다.
3. 배포 서버에서는 저장소에 `git push` 명령이 발생하면 `git hook`을 실행시킨다.
4. `git hook`에서 Docker 이미지를 생성한 뒤 Docker 레지스트리에 올린다.
5. 배포 서버는 SSH로 애플리케이션 서버에서 `docker pull` 명령을 실행시키고, `docker run` 명령으로 컨테이너를 생성한다.



2. 서버 여러 대에 애플리케이션 배포하기

1. 개발자 PC에서 Git 설치 및 저장소 생성하기

개발자 PC에 Git을 설치하지 않았다면 Git(<http://git-scm.com>)을 설치합니다.

Windows와 Mac OS X에서는 다음 주소에서 설치 파일을 다운로드하여 설치하면 됩니다. 설치 방법은 특별한 것이 없으므로 따로 설명하지 않겠습니다.

- Windows: <http://msysgit.github.com>
- Mac OS X: <http://sourceforge.net/projects/git-osx-installer>

우분투

```
$ sudo apt-get install git
```

CentOS

```
$ sudo yum install git
```

이제 Windows에서는 Git Bash를 실행합니다(Git Bash에서는 유닉스/리눅스 방식의 명령을 실행할 수 있으므로 지금부터 나오는 명령은 Mac OS X, 리눅스, Windows에서 모두 동일합니다). Mac OS X는 터미널을 실행하고, 리눅스에서는 터미널에서 그대로 진행합니다.

Git 저장소를 생성하고, 저장소 디렉터리로 이동합니다.

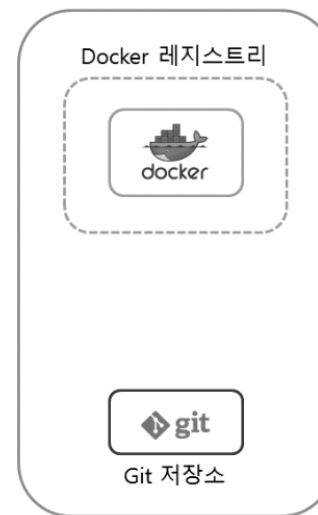
```
~$ git init exampleapp
~$ cd exampleapp
```

`git config` 명령으로 자신의 이메일과 이름을 설정합니다.

```
~/exampleapp$ git config --global user.email gd@yuldo.com
~/exampleapp$ git config --global user.name "Hong, Gildong"
```



개발자 PC



배포 서버



애플리케이션 서버 1



애플리케이션 서버 2

2. 서버 여러 대에 애플리케이션 배포하기

2. 개발자 PC에서 Node.js로 웹 서버 작성하기

개발자 PC에서 간단하게 Node.js로 웹 서버를 작성하겠습니다. 다음 내용을 app.js로 저장합니다.

~/exampleapp/app.js

```
var express = require('express');
var app = express();

app.get(['/', '/index.html'], function (req, res) {
  res.send('Hello Docker');
});

app.listen(80);
```

Node.js npm 패키지 사용을 위해 다음과 같이 작성한 뒤 package.json로 저장합니다.

~/exampleapp/package.json

```
{
  "name": "exampleapp",
  "description": "Hello Docker",
  "version": "0.0.1",
  "dependencies": {
    "express": "4.4.x"
  }
}
```

git add, git commit 명령으로 개발자 PC의 exampleapp 저장소에 파일을 커밋합니다.

```
~/exampleapp$ git add app.js package.json
~/exampleapp$ git commit -m "add source"
```

2. 서버 여러 대에 애플리케이션 배포하기

3. 개발자 PC에서 Dockerfile 작성하기

배포 서버에서 Docker 이미지를 생성하고 배포할 수 있도록 개발자 PC에서 Dockerfile을 작성합니다. 다음 내용을 Dockerfile로 저장합니다

~/exampleapp/Dockerfile

```
FROM ubuntu:14.04

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

ADD app.js /var/www/app.js
ADD package.json /var/www/package.json

WORKDIR /var/www
RUN npm install

CMD nodejs app.js
```

- FROM으로 ubuntu:14.04를 기반으로 이미지를 생성하도록 설정합니다.
- RUN으로 nodejs, npm 패키지를 설치합니다.
- ADD로 app.js와 package.json을 이미지의 **/var/www **디렉터리에 복사합니다.
- WORKDIR로 실행 디렉터를 /var/www로 변경합니다. 그리고 RUN으로 `npm install` 명령을 실행하여 package.json에 설정된 Node.js 모듈을 설치합니다.
- CMD로 컨테이너가 시작되면 nodejs를 이용하여 app.js를 실행하도록 설정합니다(우분투에서 Node.js를 패키지로 설치하면 실행 파일은 node가 아닌 nodejs입니다).

Dockerfile도 개발자 PC의 exampleapp 저장소에 커밋합니다.

```
~/exampleapp$ git add Dockerfile
~/exampleapp$ git commit -m "add Dockerfile"
```


2. 서버 여러 대에 애플리케이션 배포하기

4. 개발자 PC에서 SSH키 생성하기

개발자 PC에서 ssh-keygen 명령을 실행하여 SSH 키를 생성합니다(Windows는 Git Bash에서 다음 명령을 실행합니다).

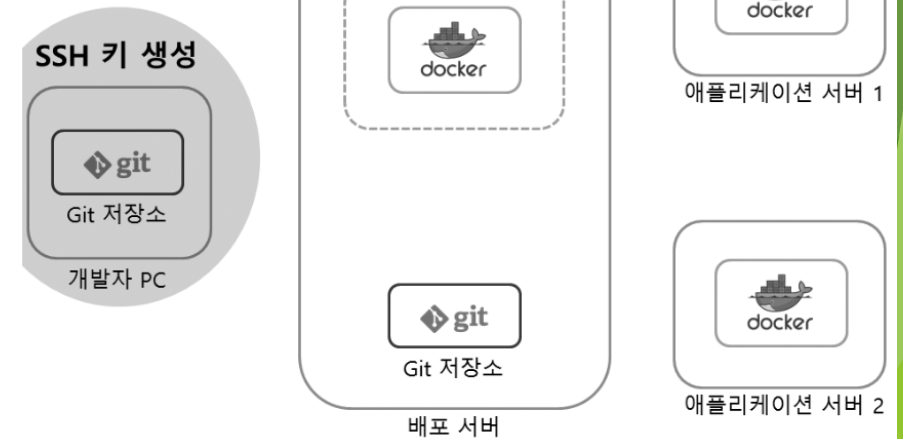
- Enter passpEnter file in which to save the key: 기본값 그대로 사용합니다.
 - hrase, Enter same passphrase again: 엔터를 입력하여 비밀번호를 설정하지 않습니다.
- 이미 /home/<사용자 계정>/.ssh 디렉터리에 id_rsa, id_rsa.pub 파일이 있다면 이 부분은 건너뜁니다.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pyrasis/.ssh/id_rsa):
Created directory '/home/pyrasis/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pyrasis/.ssh/id_rsa.
Your public key has been saved in /home/pyrasis/.ssh/id_rsa.pub.
The key fingerprint is:
64:9e:5a:8a:0b:93:f0:c5:fb:89:41:31:c4:bb:a2:ab pyrasis@ubuntu
The key's randomart image is:
+--[ RSA 2048 ]-----+
|  ..                    |
|  ..                    |
|  o. o                  |
|  ..o + .               |
|  . +. S                |
|  o.+o +                |
|  .+. + o               |
|  . o = .               |
| E. o o                |
+-----+

```

이제 /home/<사용자 계정>/.ssh 디렉터리에 id_rsa, id_rsa.pub 파일이 생성되었습니다.

```
heechul@Ubuntu-20:~/ .ssh$ ls
id_rsa id_rsa.pub
```



2. 서버 여러 대에 애플리케이션 배포하기

5. 배포 서버에 Git 설치 및 저장소 생성하기

이제 배포 서버를 설정할 차례입니다. Docker가 리눅스 전용이므로 리눅스 서버에서 작업하도록 합니다.

배포 서버에도 Git을 설치합니다.

우분투

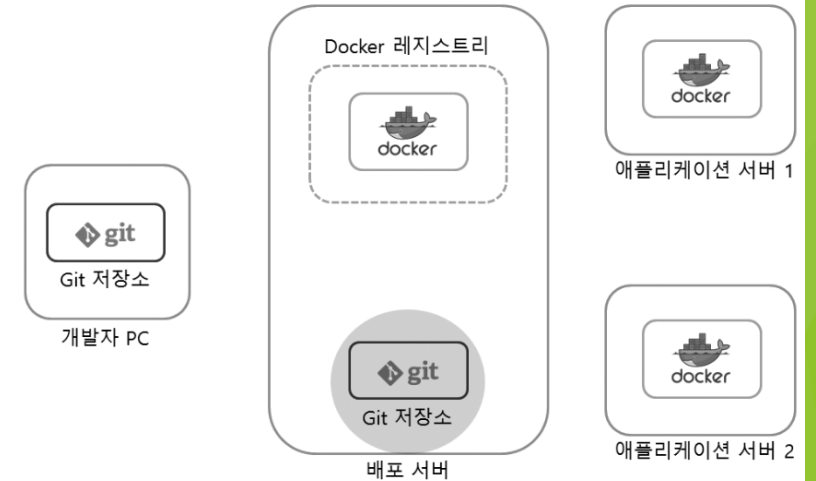
```
$ sudo apt-get install git
```

CentOS

```
$ sudo yum install git
```

현재 리눅스 계정의 홈 디렉터리(/home/⟨사용자 계정⟩)에 exampleapp 저장소를 생성합니다. 그리고 개발자 PC에서 push한 소스를 받을 수 있도록 receive.denycurrentbranch를 ignore로 설정합니다.

```
~$ git init exampleapp
~$ cd exampleapp
~/exampleapp$ git config receive.denycurrentbranch ignore
```



2. 서버 여러 대에 애플리케이션 배포하기

6. 개발자 PC에서 Git 설치 및 저장소 생성하기

배포 서버가 애플리케이션 서버에서 비밀번호 없이 SSH 명령을 실행하려면 먼저 SSH 키를 생성해야 합니다.

배포 서버에서 **ssh-keygen** 명령을 실행하여 SSH 키를 생성합니다.

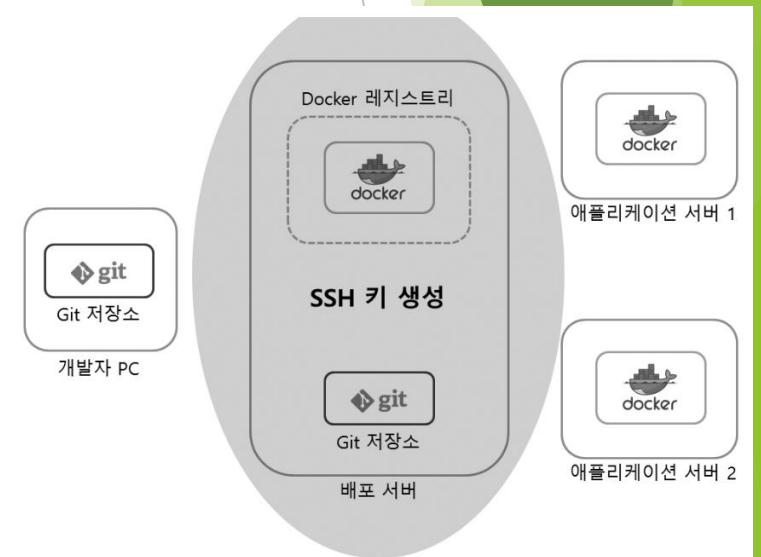
- Enter file in which to save the key: 기본값 그대로 사용합니다.

- Enter passphrase, Enter same passphrase again: 엔터를 입력하여 비밀번호를 설정하지 않습니다.

이미 /home/<배포 서버 사용자 계정>/.ssh` 디렉터리에 ****id_rsa, id_rsa.pub** 파일이 있다면 이 부분은 건너뜁니다.

```
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/pyrasis/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pyrasis/.ssh/id_rsa.
Your public key has been saved in /home/pyrasis/.ssh/id_rsa.pub.
The key fingerprint is:
59:cc:4c:1f:5b:f9:d7:3d:96:36:95:7b:91:09:b8:22 pyrasis@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048 ]-----+
|      . o.o.+         |
|     = o +.=.         |
|    = + .B           |
|   E + .  B*          |
|  S .   o =          |
|-----+-----+
+-----+-----+
```

이제 /home/<배포 서버 사용자 계정>/.ssh 디렉터리에 **id_rsa, id_rsa.pub** 파일이 생성되었습니다.



2. 서버 여러 대에 애플리케이션 배포하기

7. 배포 서버에 Docker 설치하기

배포 서버에서 사용할 Docker를 설치합니다. CentOS에서 EPEL 설치 방법은 '2.1.3 RedHat Enterprise Linux, CentOS'를 참조하기 바랍니다.

우분투

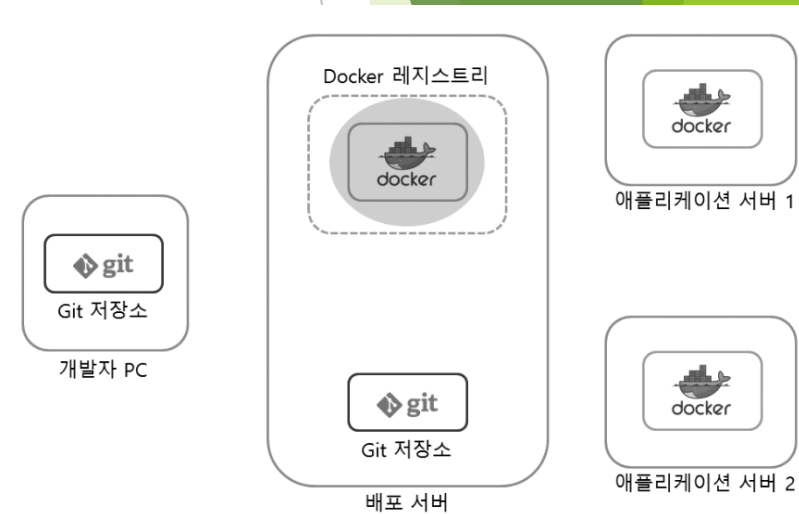
```
$ sudo apt-get install docker.io  
$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
```

CentOS

```
$ sudo yum install docker-io  
$ sudo service docker start
```

docker 명령을 **sudo** 없이 **root** 권한으로 사용할 수 있도록 현재 리눅스 계정을 **docker** 그룹에 추가합니다.

```
$ sudo usermod -aG docker ${USER}  
$ sudo service docker restart
```



2. 서버 여러 대에 애플리케이션 배포하기

8. 배포 서버에 Docker 레지스트리 서버 설정하기

Docker 이미지를 여러 애플리케이션 서버에 전달하기 위해 Docker 레지스트리(개인 저장소) 서버를 설정합니다.

먼저 Docker 레지스트리 이미지를 받습니다.

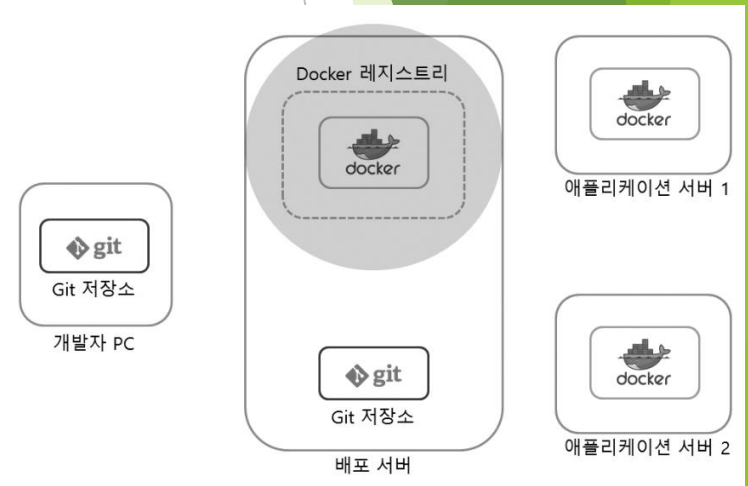
```
$ sudo docker pull registry:latest
```

registry:latest 이미지를 컨테이너로 실행합니다.

```
$ sudo docker run -d -p 5000:5000 --name example-registry \  
-v /tmp/registry:/tmp/registry \  
registry
```

이제 배포 서버의 5000번 포트로 Docker 레지스트리 서버를 사용할 수 있게 되었습니다.

이미지 데이터를 Amazon S3에 저장하려면 ['6.1.3 Amazon S3에 이미지 데이터 저장'](#)을 참조하기 바랍니다.



2. 서버 여러 대에 애플리케이션 배포하기

9. 배포 서버에 SSH 키 설정하기

개발자 PC에서 비밀번호 없이 배포 서버에 접속할 수 있도록 앞에서 생성한 SSH 키를 설정합니다.

배포 서버의 `/home/〈배포 서버 사용자 계정〉` 디렉터리 아래에 `.ssh` 디렉터리를 생성하고, 권한을 설정합니다.

```
~$ mkdir .ssh
~$ chmod 700 .ssh
```

방금 생성한 `.ssh` 디렉터리에 `authorized_keys` 파일을 생성합니다. 그리고 개발자 PC에서 생성한 `id_rsa.pub` 파일의 내용을 복사해서 `authorized_keys` 파일 붙여넣습니다.

```
~/ssh/authorized_keys
```

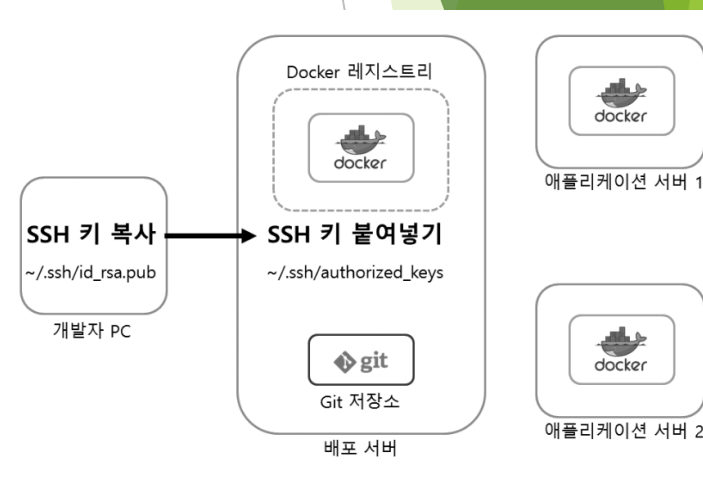
```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC4nKkAHdB8gU9DT9te9HDMWA8qTY/9YjgxVr493YxqS3vu+Y5/UyXgRPMegRZGKWZEDtyssYi/X
xQ5R1k0xXF8NE/T/x8DULVc32e3mtA9vnzOMqshjVLqP0ZZXdhkipEw6s2uKJVtmXQdN+Pz3DupyJ+1a0jww/n3y62goEZ2f9jr7ZnGtL2haYSZJE
Mh57RVAYwLW3n1Ax53dhKE9ha9xdBC+BRTjkqE8oEq+Vg67H01604kxnRvubiVDIXfm1/mvXqz+GdgzSta8Uspz59Tth01J0cJbAZydL5VEBYV5rz
iHvRmqwDewV0Mn7hZjAuVYD1gPMbW26/hXsdxycMJ pyrasis@ubuntu
```

위의 내용은 저의 공개 키입니다. 여러분의 공개 키(`id_rsa.pub`)를 설정하기 바랍니다.

`authorized_keys` 파일도 권한을 설정합니다.

```
$ chmod 600 authorized_keys
```

이제 개발자 PC에서 비밀번호 없이 `git push` 명령을 사용할 수 있습니다.



2. 서버 여러 대에 애플리케이션 배포하기

10. 배포 서버에 Git Hook 설정하기

개발자 PC에서 git push 명령으로 소스를 올렸을 때 Docker 이미지를 생성하고,
Docker 레지스트리 서버에 올린 뒤 각 애플리케이션 서버에 배포하도록 Git Hook을 설정합니다.

다음 내용을 배포 서버의 `/home/<배포 서버 사용자 계정>/exampleapp/.git/hooks` 디렉터리에 `post-receive`로 저장합니다.

- [dockerbook/Chapter08/MultipleServerDeployment/hooks/post-receive](#)

```
#!/bin/bash

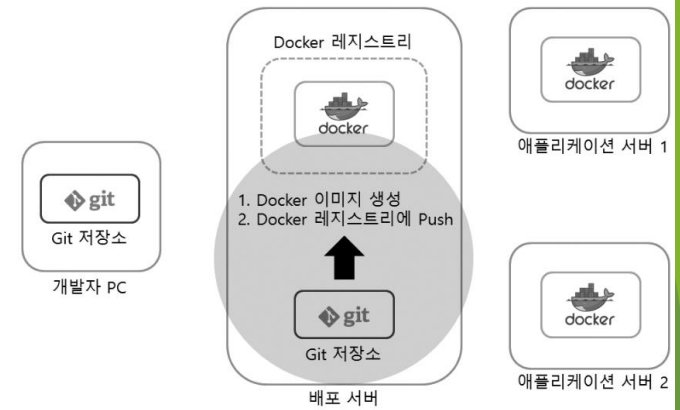
APP_NAME=exampleapp
APP_DIR=$HOME/$APP_NAME
REVISION=$(expr substr $(git rev-parse --verify HEAD) 1 7)

REGISTRY=192.168.0.40:5000
APP_SERVERS=(
    "pyrasis@192.168.0.101"
    "pyrasis@192.168.0.102"
)

GIT_WORK_TREE=$APP_DIR git checkout -f

cd $APP_DIR
docker build --tag $APP_NAME:$REVISION .
docker tag $APP_NAME:$REVISION $REGISTRY/$APP_NAME:$REVISION
docker push $REGISTRY/$APP_NAME:$REVISION

SSH="ssh -o StrictHostKeyChecking=no"
for SERVER in ${APP_SERVERS[@]}
do
    $SSH $SERVER docker pull $REGISTRY/$APP_NAME:$REVISION
    $SSH $SERVER docker stop $APP_NAME
    $SSH $SERVER docker rm $APP_NAME
    $SSH $SERVER docker run -d --name $APP_NAME \
        -p 80:80 $REGISTRY/$APP_NAME:$REVISION
done
```



- `APP_NAME` 변수에 현재 애플리케이션의 이름을 설정합니다. 이 이름은 저장소 이름과 같아야 합니다.
- `APP_DIR` 변수에 저장소 디렉터리 경로를 설정합니다. 셸에서 `HOME` 변수에는 사용자 홈 디렉터리 경로(`/home/<사용자 계정>`)가 들어있습니다.
- `git rev-parse` 명령으로 Push된 최신 리비전을 구한 뒤 앞의 7자리만 `REVISION` 변수에 저장합니다.
- `REGISTRY` 변수에 Docker 레지스트리 서버의 IP 주소와 포트 번호를 설정합니다. `192.168.0.40:5000`은 저의 레지스트리 서버 IP 주소이므로 이 부분은 여러분의 IP 주소를 설정합니다.
- `APP_SERVERS` 변수에 애플리케이션 서버의 사용자 계정과 IP 주소를 배열 형태로 설정합니다. `pyrasis@192.168.0.101`, `pyrasis@192.168.0.102`는 저의 애플리케이션 서버 계정과 IP 주소이므로 이 부분은 여러분의 계정과 IP 주소를 설정합니다.
- `git checkout -f` 명령으로 Push된 소스를 저장소 디렉터리에 받습니다. 반드시 `GIT_WORK_TREE` 변수를 설정해 주어야 합니다.
- `exampleapp` 저장소 디렉터리로 이동합니다.
- `docker build` 명령으로 이미지를 생성합니다. 이미지 이름은 `exampleapp`이고 태그에는 방금 구한 Git 리비전을 설정합니다.
- 레지스트리 서버에 올릴 수 있도록 `docker tag` 명령으로 방금 생성한 이미지를 `<레지스트리 서버 주소>/exampleapp:<리비전>`으로 태그를 생성합니다.
- `docker push` 명령으로 이미지를 레지스트리 서버에 올립니다.
- SSH 변수에 `ssh` 명령과 옵션을 설정합니다. `StrictHostKeyChecking=no` 옵션을 설정해야 호스트 키 경고를 무시하고 바로 명령을 실행할 수 있습니다. 그렇지 않으면 사람이 일일이 애플리케이션 서버에 접속하여 호스트 키 경고에서 `yes`를 입력해주어야 합니다.
- 반복문을 이용하여 `APP_SERVERS` 배열에 담긴 애플리케이션 서버 주소마다 SSH로 명령을 실행합니다.
 - `docker pull` 명령으로 레지스트리 서버에 저장된 이미지를 받습니다.
 - `docker stop`, `rm` 명령으로 현재 실행되고 있는 Docker 컨테이너를 정리한 뒤 삭제합니다.
 - `docker run` 명령으로 방금 레지스트리 서버에서 받은 이미지를 컨테이너로 생성하고, `-p` 옵션으로 80번 포트를 연결하고 외부에 노출합니다. 이미지 이름은 `<레지스트리 서버 주소>/exampleapp` 이고 태그에는 앞에서 구한 Git 리비전을 설정합니다.

`post-receive` 파일에 실행 권한을 부여합니다.

```
~/exampleapp/.git/hooks$ chmod +x post-receive
```

2. 서버 여러 대에 애플리케이션 배포하기

11. 애플리케이션 서버에 Docker 설치하기

이제 Docker 컨테이너를 생성할 각 애플리케이션 서버에 Docker를 설치합니다. CentOS에서 EPEL 설치 방법은 '2.1.3 RedHat Enterprise Linux, CentOS'를 참조하기 바랍니다.

우분투

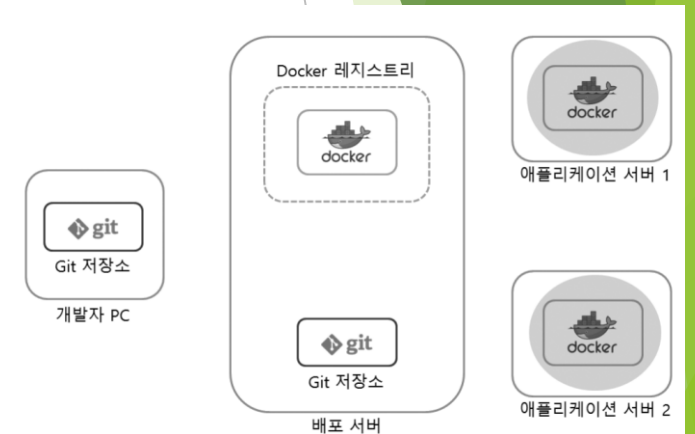
```
$ sudo apt-get install docker.io  
$ sudo ln -sf /usr/bin/docker.io /usr/local/bin/docker
```

CentOS

```
$ sudo yum install docker-io  
$ sudo service docker start
```

docker 명령을 **sudo** 없이 root 권한으로 사용할 수 있도록 현재 리눅스 계정을 **docker** 그룹에 추가합니다.

```
$ sudo usermod -aG docker ${USER}  
$ sudo service docker restart
```



2. 서버 여러 대에 애플리케이션 배포하기

12. 애플리케이션 서버에 SSH 키 설정하기

배포 서버에서 애플리케이션 서버에 비밀번호 없이 접속할 수 있도록 앞에서 생성한 SSH 키를 설정합니다. 이 부분은 반드시 애플리케이션 서버마다 설정해주어야 합니다.

애플리케이션 서버의 `/home/〈애플리케이션 서버 사용자 계정〉` 디렉터리 아래에 `.ssh` 디렉터리를 생성하고, 권한을 설정합니다.

```
~$ mkdir .ssh
~$ chmod 700 .ssh
```

방금 생성한 `.ssh` 디렉터리에 `authorized_keys` 파일을 생성합니다. 그리고 배포 서버에서 생성한 `id_rsa.pub` 파일의 내용을 복사해서 `authorized_keys` 파일 붙여넣습니다.

```
~/ssh/authorized_keys
```

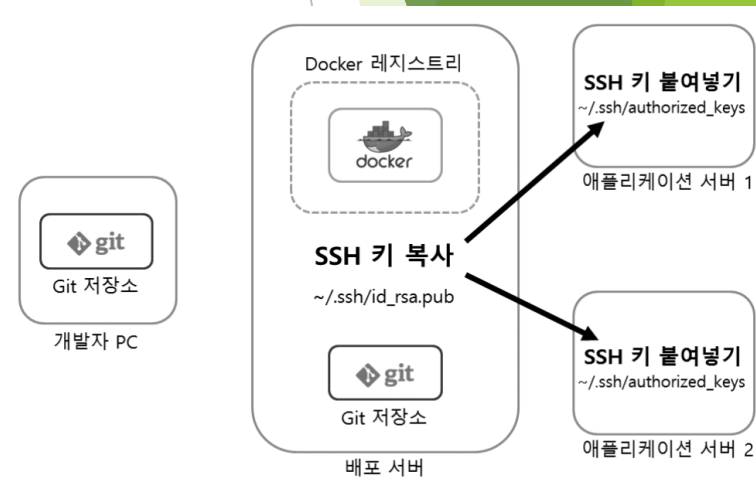
```
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAQEAtahmHnklgmvyntVT1FEfSn1HhA1LcIpuz/1jzRuVzQj89nGx+hBIM7AHSKp6AzjicR6/WmvY2hCny
20dmbjkJ9k6Ji8rpOTwlr1kLk1kRauonVuZrm0kHCez1gPpYjaYGQ55eQrFF6FJwD4Dyr1cPRw4HFk/RwdSP+qKdu2QZn94n0rj851MMYPxzFHdnd
o/F90pTnXF2YwApNhBh0Njkh/fRBX8qt9qCLvXrGK/ZEsd6d8kBJC8HX1zRuweG+op7QkWR7s9GY1TqOdFYG4MQfLr7K+2RK2qJnjpP3f114t+jJoc
2iQ7Hb3q+EbNyDaFwb6Ye2sZJSFX00v3XDHSXNQ== pyrasis@localhost.localdomain
```

위의 내용은 저의 공개 키입니다. 여러분의 공개 키(`id_rsa.pub`)를 설정하기 바랍니다.

`authorized_keys` 파일도 권한을 설정합니다.

```
$ chmod 600 authorized_keys
```

이제 배포 서버에서 비밀번호 없이 SSH로 명령을 사용할 수 있습니다.



2. 서버 여러 대에 애플리케이션 배포하기

13. 개발자 PC에서 소스 Push하기

이제 개발자 PC로 돌아옵니다.

exampleapp 저장소 디렉터리로 이동한 뒤 `git remote add` 명령으로 `origin` 주소를 설정합니다.

```
~/exampleapp$ git remote add origin <서버 사용자 계정>@<배포 서버 IP 주소 또는 도메인>:exampleapp
```

예를 들면 배포 서버 사용자 계정이 `pyrasis`이고, 배포 서버 IP 주소가 `192.168.0.40`이면 `origin` 주소는 `pyrasis@192.168.0.40:exampleapp`이 됩니다.

`git push` 명령으로 소스를 서버에 올립니다.

```
~/exampleapp$ git push origin master
```

`git push` 명령 출력 결과에서 Docker 이미지가 생성되고 배포되는 모습을 볼 수 있습니다. `git push` 명령이 완전히 끝나면 웹 브라우저를 실행하고 애플리케이션 서버의 IP 주소로 각각 접속합니다.

웹 브라우저에서 `app.js`에서 출력한 Hello Docker가 표시됩니다. 이제 소스를 수정한 뒤 서버에 Push하면 여러 서버에 Docker 이미지가 배포됩니다. Dockerfile과 `post-receive` 파일을 각자 상황에 맞게 수정하여 사용하면 됩니다.