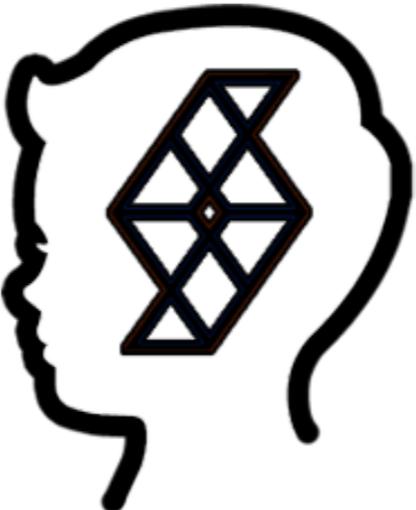


Container

가상화 입문



김대권 (nacyot)

Software & System Engineer

TOC

- Docker 입문
- Immutable Infrastructure



시대는 바야흐로 가상화

하드웨어 가상화

소프트웨어로 구현된 가상의 하드웨어

물리적 하드웨어 위의 소프트웨어로 구현된 하드웨어

VirtualBox¹

Parallels²

VMWare³

¹ <https://www.virtualbox.org/>

² <http://www.parallels.com/products/desktop/>

³ <http://www.vmware.com/>

이미 일상적인 기술

인프라스트럭처

하드웨어 파편화 최소화
서버 없는 사무실
거의 무한한 확장성

그리고 새로이 등장한 재발견된

Container?

Container 가상화의 대표주자

Docker⁴

⁴ <https://www.docker.com/>

Hello, Docker

간단한 예제로 시작하기

Ghost 블로그⁵

ex) 리모티 블로그

⁵ <http://ghost.org/>

고스트 설치하기

1. node 설치하기
2. ghost 소스코드 다운로드
3. 데이터베이스 셋업
4. 관련 환경설정
5. 서버 설정하기
6. 프로덕션 서버에 배포

Docker로 실행하기

```
$ docker pull ghost
```

```
$ docker images | grep ghost
ghost latest 85c3e5ecdf68 2 days ago 231.3 MB
```

```
$ docker run --name ghost -p 8080:2368 -d ghost
b81b2a20c09f0e973e6
```



Welcome to Ghost

You're all set! Now We've put together a little post to introduce you to the Ghost editor and get started with your own content. Ghost is now ready to go. Getting Started | 28 July 2015

Ghost Blog

프로세스가 실행되는 두 가지 환경

1. Docker Server가 실행되는 컴퓨터의 환경
2. ghost가 설치된 Docker 이미지의 환경

Docker가 실행되는 환경

```
$ docker version  
Client version: 1.7.1  
Client API version: 1.19  
Go version (client): go1.4.2  
Git commit (client): 786b29d  
OS/Arch (client): linux/amd64  
...
```

```
$ which node  
-sh: node: not found
```

xaos가 설치된 Docker 이미지의 환경

```
$ # docker server 환경  
$ docker run --rm -it ghost bash
```

```
ShellInDockerContainer$ # 컨테이너 환경  
ShellInDockerContainer$ which node  
/usr/local/bin/node
```

```
ShellInDockerContainer$ npm list | grep ghost  
ghost@0.6.4 /usr/src/ghost
```

**소프트웨어 패키지가 아니라
설치된 환경 자체**

하드웨어 가상화 없는 격리된 환경의 프로세스

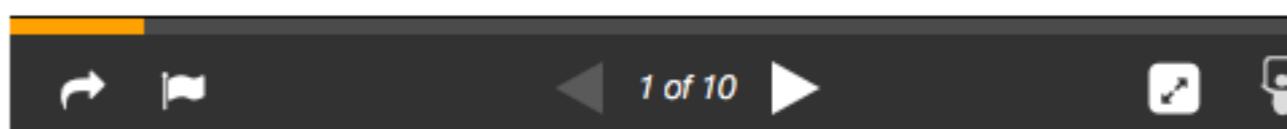
Container!

그래서 대단한 거야?

The Future of Everything

The Future of Everything

Link



The Future of Everything

The Future of Everything

웹에서 보기

5

5
리트윗
13
관심



오후 11:44 - 2014년 10월 7일 · 자세히

docker

docker

docker

docker



실전으로 배우는

Docker

이미지

**프로세스를 실행하기 위해
미리 준비된 환경**

컨테이너

이미지 환경에서 실행된 프로세스

원시적인 컨테이너

chroot

프로세스의 root를 속이자

Container = chroot + @

Docker

**이미지 관리 인터페이스
컨테이너 관리 인터페이스
계층화된 파일 시스템**

Docker 설치

Docker 설치

- 리눅스

```
$ curl -sSL https://get.docker.com/ | sh
```

- 윈도/OSX : boot2docker로 설치

boot2docker

<http://boot2docker.io/>

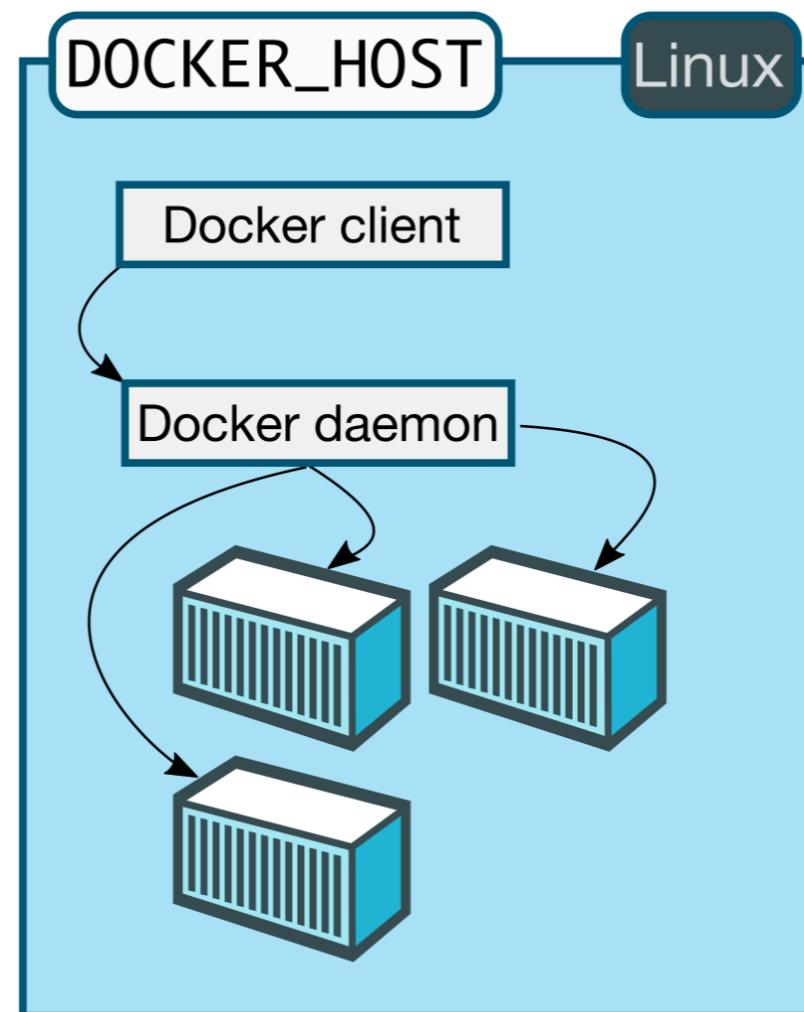
boot2docker (1)

- Docker는 리눅스 커널만 지원
- 윈도/OSX에서는 Docker를 지원하지 않음
- 따라서 리눅스 가상 머신이 필요
- boot2dokcer = CoreLinux on VirtualBox

boot2docker (2)

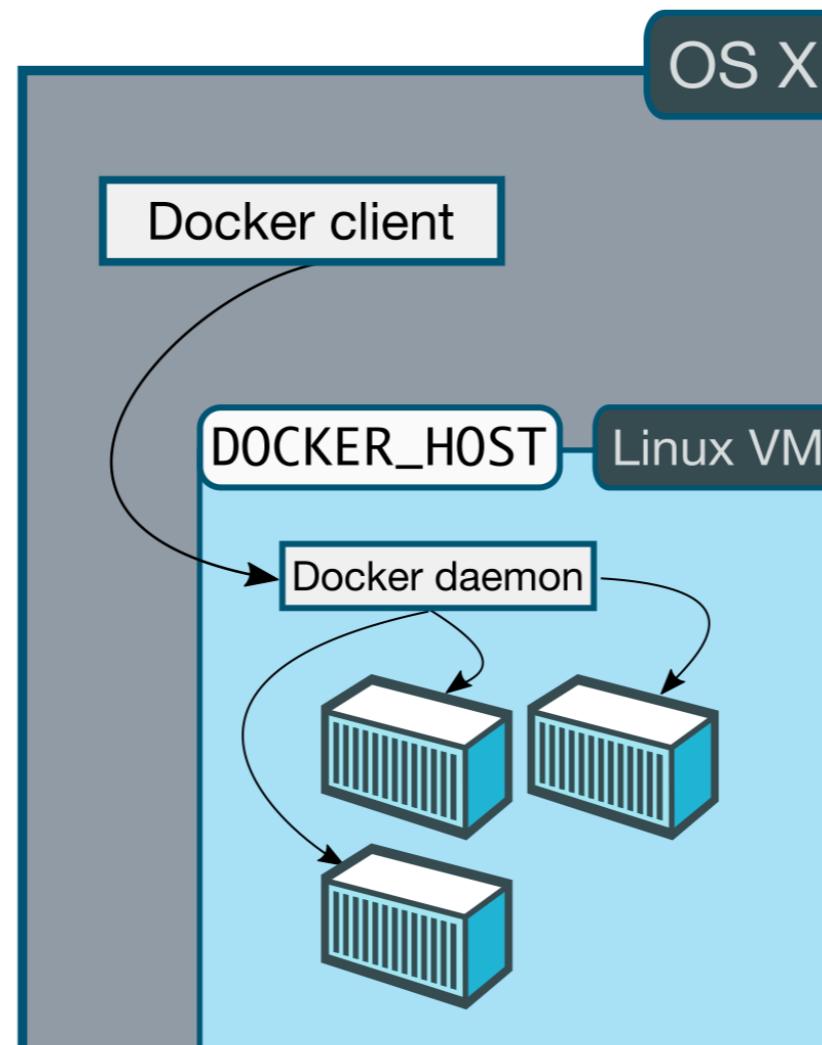
- 가상머신을 쓰면서까지 Docker를 써야하나?
 - 성능 면에서는 분명히 불리함
 - 이미지/컨테이너 개념은 여전히 강력함
 - 실배포에서는 클라우드에서 리눅스 머신을 사용

Docker 원리 이해하기



from

boot2dokcer 원리 이해하기



from

boot2docker 설치

1. 먼저 boot2docker에서 OS별 인스톨러 다운로드
2. 인스톨러로 설치
 - docker와 boot2docker 명령어가 설치
3. 터미널에서 boot2docker init 실행
 - CoreLinux 가상머신 생성
4. boot2docker up 실행
5. eval \$(boot2docker shellinit) 실행

```
eval $(boot2docker shellinit)
```

- docker 명령어로 가상머신의 데몬을 사용하도록 지정
 - docker = Docker Client
 - 가상머신의 Docker 데몬 = Docker Server
 - 명령을 내리면 실제로 가상머신에서 실행
- 터미널을 새로 열 때마다 실행해야함
 - ~/.bashrc 파일을 열어 맨 뒤에 추가
 - eval \$(boot2docker shellinit)



Select a Docker image to create a new container.

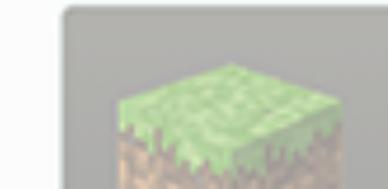
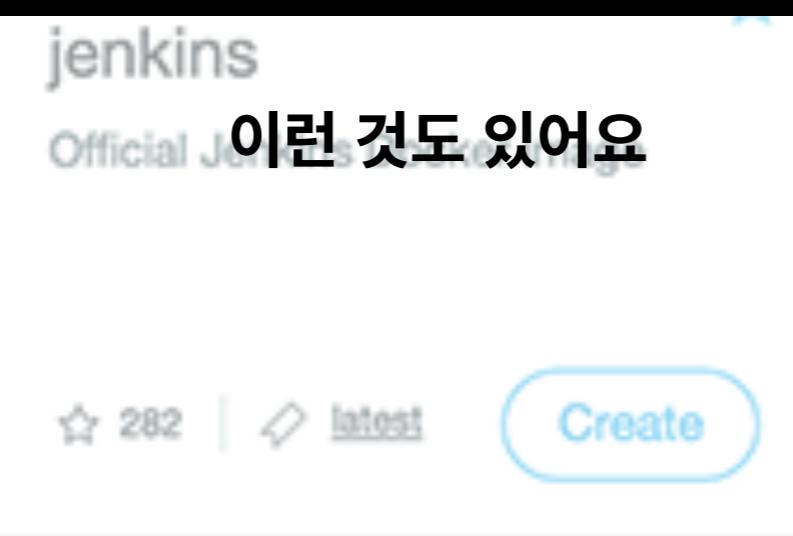
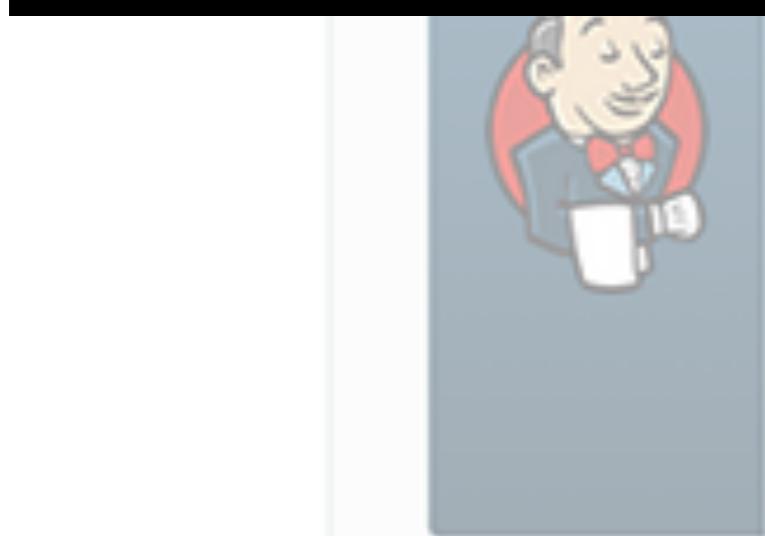


Search Docker Hub for

ner

Recommended

Docker GUI 인터페이스 Kitematic



주제

온라인!

이미지를 만들어보자!

새로운 환경 정의하기

wget이 설치된 Ubuntu 이미지

docker pull

이미지를 받아오는 명령어

```
$ docker pull <IMAGE_NAME>
```

```
$ docker pull ubuntu:14.04
```

docker images

도커에서 사용가능한 이미지 목록

```
$ docker images
```

REPOSITORY	TAG	IMAGE ID	...
ubuntu	14.04	d2a0ecffe6fa	...

docker run

이미지로부터 컨테이너 실행하기

```
$ docker run -it <IMAGE_NAME> <COMMAND>
```

ubuntu:14.04 이미지에서 bash 명령어를 실행

\$ # <- 여기는 호스트의 셸

```
$ docker run -it ubuntu:14.04 bash
```

root@8b7290edaa5c:/\$ # <- 새로운 환경 안의 셸

Container ID
8b7290edaa5c

docker ps

실행중인 컨테이너 목록

호스트의 다른 셸에서 실행

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	...
8b7290edaa5c	ubuntu:14.04	"bash"	...

컨테이너에서 wget 설치하기

```
root@8b7290edaa5c:/# wget  
bash: wget: command not found
```

```
root@8b7290edaa5c:/# apt-get update  
root@8b7290edaa5c:/# apt-get install -y wget
```

```
root@8b7290edaa5c:/# wget  
wget: missing URL  
Usage: wget [OPTION]... [URL]...
```

Try `wget --help' for more options.

docker diff

실행한 이미지로부터 현재 컨테이너의 변경사항 출력

```
$ docker diff <CONTAINER_ID>
```

호스트의 다른 셸에서 실행

```
$ docker diff 8b7290edaa5c
A /.wh..wh.plnk/101.138481 # 파일 추가
A /.wh..wh.plnk/361.138462 # 파일 추가
C /etc # 변경 사항
A /etc/ca-certificates # 파일 추가
...
.
```

docker commit

실행한 이미지로부터 현재 컨테이너의 변경사항 저장

```
$ docker commit <CONTAINER_ID> <IMAGE_NAME>
```

```
$ docker commit 8b7290 nacyot/wget:latest  
9ea5dab42924a2a7ccb4a...
```

새로 생성된 이미지 ID

```
$ docker images  
REPOSITORY      TAG      IMAGE ID      ...  
ubuntu          14.04    d2a0ecffe6fa    ...  
nacyot/wget     latest   9ea5dab42924    ...
```

nacyot/wget 실행해보기

```
$ docker run -it ubuntu:14.04 bash  
root@c30e6fa29017:/# wget  
bash: wget: command not found
```

```
$ docker run -it nacyot/wget bash  
root@f87cd323f346:/# wget  
wget: missing URL  
Usage: wget [OPTION]... [URL]...
```

Try `wget --help' for more options.

Dockerfile

이미지 빌드 과정을 파일로 기술

```
FROM ubuntu:14.04
```

```
RUN apt-get update
```

```
RUN apt-get install -y wget
```

docker build

Dockerfile로 이미지 빌드하기

```
$ docker build -t <IMAGE_NAME> <TARGET_DIR>
```

```
$ ls
```

Dockerfile

```
$ dokcer build -t nacyot/wget:latest .
```

Demo

더 공부하기

도커(Docker) 튜토리얼 : 깐 김에 배포까지

컨테이너가 필요한 이유

보편적 물리법칙

언제 어디서나

**컴퓨터의 환경은
보편적이지 않다**

특수한 환경

특정 하드웨어

특정 OS

특정 시점의 시스템 설정

설치된 소프트웨어들

컴퓨터를 수리하는 가장 일반적인 알고리즘

재부팅

그래도 안 되면...

재설치

그래도 안 되면...

OS 재설치

윈도우 다시 깔아.

**상태 관리는 원래 어렵다
서버도 어렵다
데스크탑도 똑같이 어렵다**

깨끗한 환경

Dockerfile이란
깨끗한 환경으로부터
애플리케이션 실행 환경까지 최단경로

이미지 = 작동되는 상태

10명의 맥북

10개의 서로 다른 환경

하나의 이미지

항상 같은 환경

Docker is

Docker is
뽕 맞은 chroot

Docker is
초강력한 포터블 앱

재현성

이미지로 만들면 공유 가능
여기서 되면, 저기서도 됨

Docker hub

- 도커 공식 이미지 공유를 위한 서비스
- 다양한 이미지가 미리 준비되어 있음
 - 기본 운영체제 : Ubuntu, CentOS, ...
 - CMS: Wordpress, Ghost, drupal, ...
 - Ipython, Jira, Gitlab, Deepdream, ...
- docker run 명령어 하나면 실행 가능

Wordpress

난이도: 중

Wordpress on Docker

```
$ docker run --name wp-mysql -e MYSQL_ROOT_PASSWORD=password -d mysql  
$ docker run --link wp-mysql:mysql -p 8000:80 -d wordpress
```



Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

Information needed

Please provide the following information. Don't worry, you can always change these settings later.

http://192.168.59.103:8000

Username

Usernames can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

Password, twice

A password will be automatically generated for you if you leave this blank.

Strength indicator

Hint: The password should be at least seven characters long. To make it stronger, use upper and lower case letters, numbers, and symbols like ! " ? \$ % ^ &).

Your E-mail

Double-check your email address before continuing.

Privacy

Allow search engines to index this site.

[Install WordPress](#)

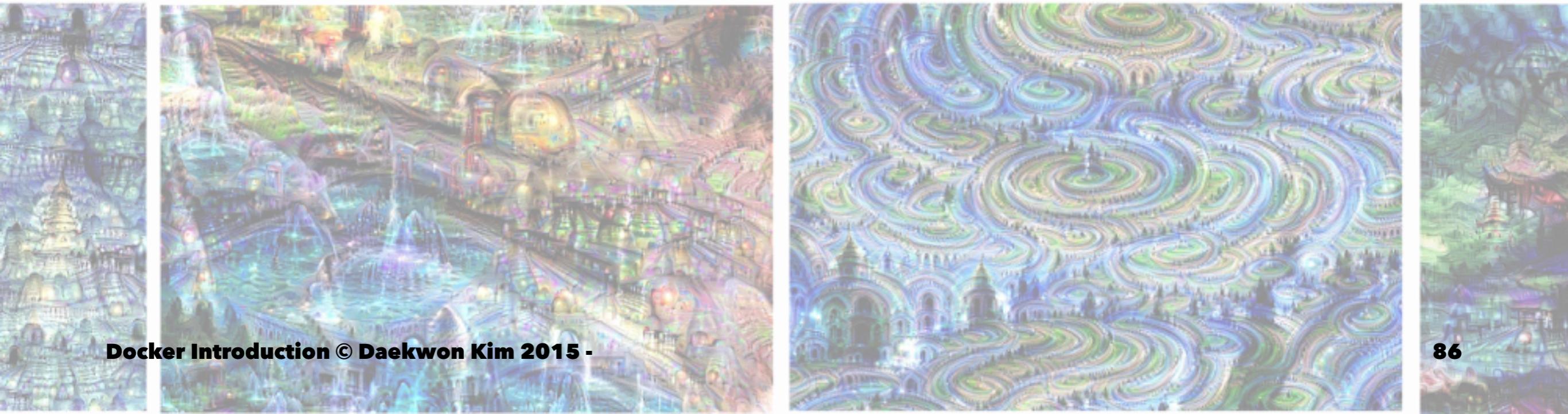


Deepdream



from

난이도: 상



Deepdream

- Deep Learning을 통해 컴퓨터의 눈으로 보는 세계
- Deep Learning 프레임워크 caffe를 사용

나도 한번 해보자

Deepdream

그럼 먼저 `caffe`를 설치해 봅.....

```
/usr/include/secure/_common.h:30:32: note: expanded from macro '_USE_FORTIFY_LEVEL'
# define _USE_FORTIFY_LEVEL 2
^
```

OSX Installation

```
/usr/include/secure/_common.h:39:62: note: expanded from macro '__darwin_obsz'
#define __darwin_obsz(object) __builtin_object_size (object, _USE_FORTIFY_LEVEL > 1 ? 1 : 0)
```

```
We highly recommend using the Homebrew package manager. Ideally you could start from a clean /usr/local to avoid conflicts. In the following, we assume that you're using Anaconda Python and Homebrew.
```

```
CUDA: Install via the NVIDIA package that includes both CUDA and the bundled drivers. CUDA 7 is strongly suggested. Other CUDA require libstdc++ while clang++ is the default compiler and libstdc++ the default standard library on OS X 10.9+. This disagreement makes it necessary to change the compilation settings for each of the dependencies. This is prone to error.
```

```
Library Path: We find that everything compiles successfully if $LD_LIBRARY_PATH is not set at all, and $DYLD_FALLBACK_LIBRARY_PATH is set to provide CUDA, Python, and other relevant libraries (e.g. /usr/local/cuda/lib:$HOME/anaconda/lib:/usr/local/lib:/usr/lib). In other ENV settings, things may not work as expected.
```

```
# define _USE_FORTIFY_LEVEL 2
```

```
General dependencies
```

```
brew install vd snappy leveldb gflags glog szip lmdb
```

```
* need the homebrew science source for OpenCV and hdf5
```

```
brew tap homebrew/science
```

```
brew install hdf5 openpy
```

```
If using Anaconda Python, a modification to the OpenCV formula might be needed. Do brew edit openpy and change the lines that look like the two lines below.
```

```
-DOPENCV_LIBRARY=$(py_prefix)/lib/libpython2.7.dylib
```

```
-DOPENCV_INCLUDE_DIR=$(py_prefix)/include/python2.7
```

```
If using Anaconda Python, hdf5 is backed and the hdf5 formula can be skipped.
```

```
Remaining dependencies, with / without Python:
```

```
* with Python pyopencv needs dependencies built from source
```

```
brew install --build-from-source --with-python -vd protobuf
```

```
brew install --build-from-source -vd boost boost-python
```

```
* without Python the usual installation suffices
```

```
brew install protobuf boost
```

```
BLAS: already installed as the Accelerate / vecLib Framework. OpenBLAS and MKL are alternatives for faster CPU computation.
```

```
Python: (optional), Anaconda is the preferred Python. If you decide against it, please use Homebrew. Check that Caffe and dependencies are linking against the same, desired Python.
```

```
Continue with compilation.
```

```
libstdc++: installation
```

```
This route is not for the faint of heart. For OS X 10.10 and 10.9 you should install CUDA 7 and follow the instructions above. If this is not an option, take a deep breath and carry on.
```

```
In OS X 10.9+, clang++ is the default C++ compiler and uses libc++ as the standard library. However, NVIDIA CUDA (even version 6.0) currently links only with libstdc++. This makes it necessary to change the compilation settings for each of the dependencies.
```

```
We do this by modifying the Homebrew formulas before installing any packages. Make sure that Homebrew doesn't install any software dependencies in the background; all packages must be linked to libstdc++.
```

```
The prerequisite Homebrew formulae are
```

```
boost snappy leveldb protobuf gflags glog szip lmdb homebrew/science/openpy
```

```
For each of these formulas, brew edit FORMULA, and add the ENV definitions as shown.
```

```
def install
    #include <FOLIO.h>
    #include <CUDALIB.h> // stdlib/libstdc++
    ENV['CXXFLAGS'] = "-stdlib=libstdc++"
    ENV['LDFLAGS'] = "-stdlib=libstdc++ -lstdc++"
    ENV.append("LDLIBRARY", "-stdlib=libstdc++ -lstdc++")
    # The following is necessary because libtool likes to strip LDFLAGS:
    ENV['CXX'] = "/usr/bin/clang++ -stdlib=lmdb"
    ENV['LIBRARY'] = "/usr/lib/libstdc++.6.dylib"
```

```
~
```

```
To edit the formulae in turn, run
```

```
for x in snappy leveldb protobuf gflags glog szip boost boost-python lmdb homebrew/science/openpy; do brew edit $x done
```

```
After this, run
```

```
brew install --build-from-source -vd std::string::size_t strlcat_chk (dest, src, len, __darwin_obsz (dest))
```

```
~
```

```
After this, run
```

```
for x in snappy leveldb protobuf gflags glog szip lmdb homebrew/science/openpy; do brew uninstall $x; brew install --build-from-source -vd $x done
```

```
brew uninstall protobuf; brew install --build-from-source --with-python -vd protobuf
```

```
brew install --build-from-source -vd boost boost-python
```

```
If this is not done exactly right then linking errors will trouble you.
```

```
Homebrew versioning that Homebrew maintains itself as a separate git repository and making the above brew edit FORMULA changes will change files in your local copy of homebrew's master branch. By default, this will prevent you from updating Homebrew using brew update, as you will get an error message like the following:
```

```
$ brew update
error: Your local changes to the following files would be overwritten by merge:
  LibraryFormula/leveldb.rb
Please, commit your changes or stash them before you can merge.
```

```
Aborting.
Error: Failure while executing: git pull -q origin refs/heads/master:refs/remotes/origin/master
```

```
One solution is to commit your changes to a separate Homebrew branch, run brew update, and rebase your changes onto the updated master. You'll have to do this both for the main Homebrew repository in /usr/local/ and the Homebrew science repository that contains OpenCV in /usr/local/Library/Taps/homebrew/homebrew-science, as follows:
```

```
cd /usr/local
```

```
git checkout -b coffee
```

```
git add -A
```

```
git commit -m "Update Caffe dependencies to use libstdc++"
```

```
cd /usr/local/Library/Taps/homebrew/homebrew-science
```

```
git checkout -b coffee
```

```
git add -A
```

```
git commit -m "Update Caffe dependencies"
```

```
Then, whenever you want to update homebrew, switch back to the master branches, do the update, rebase the caffe branches onto master and fix any conflicts:
```

```
* Switch back to homebrew master branches
```

```
cd /usr/local
```

```
git checkout master
```

```
cd /usr/local/Library/Taps/homebrew/homebrew-science
```

```
git checkout master
```

```
git checkout master
```

```
git merge coffee
```

```
* Update homebrew; hopefully this works without errors!
```

```
brew update
```

```
* Switch back to the caffe branches with the formulae that you modified earlier
```

```
cd /usr/local
```

```
git rebase master coffee
```

```
* Fix any merge conflicts and commit to coffee branch
```

```
cd /usr/local/Library/Taps/homebrew/homebrew-science
```

```
git rebase master coffee
```

```
* Fix any merge conflicts and commit to coffee branch
```

```
At this point, you should be running the latest Homebrew packages and your Caffe-related modifications will remain in place.
```

```
~
```

```
In file included from spo_alert_cef.c:67:
```

```
../strlcp.h:24:8: error: expected parameter declarator
```

```
size_t strlcpy (char *, const char *, size_t);
```

```
^
```

```
/usr/include/secure/_string.h:105:44: note: expanded from macro 'strlcpy'
```

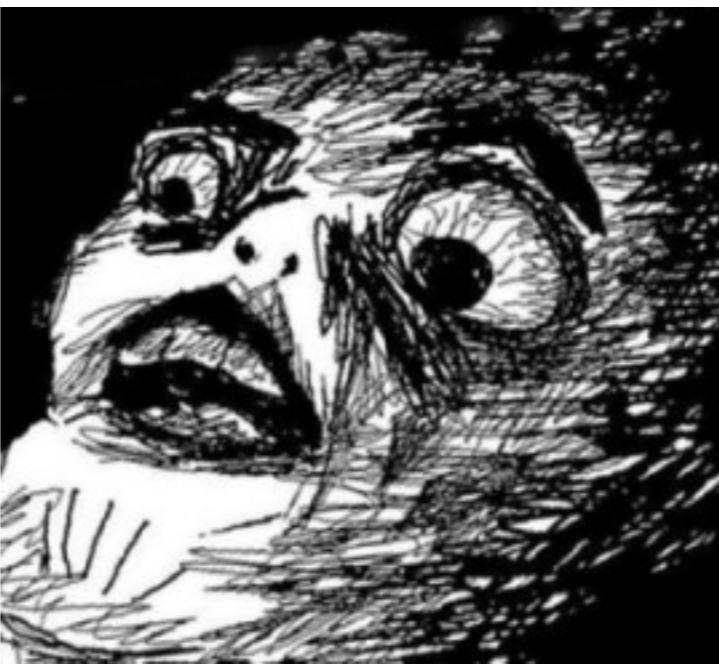
```
__builtin__strlcpy_chk (dest, src, len, __darwin_obsz (dest))
```

```
^
```

```
/usr/include/secure/_common.h:39:62: note: expanded from macro '__darwin_obsz'
```

```
#define __darwin_obsz(object) __builtin_object_size (object, _USE_FORTIFY_LEVEL > 1 ? 1 : 0)
```

Done!



Docker 등판

ryankennedyio/deepdream

```
$ git clone https://github.com/ryankennedyio/deep-dream-generator.git  
$ cd deep-dream-generator  
$ docker run -d \  
  -p 443:8888 \  
  -e "PASSWORD=password" \  
  -v $(pwd):/src \  
  ryankennedyio/deepdream  
...  
a4ea7d082a79e6251a4
```

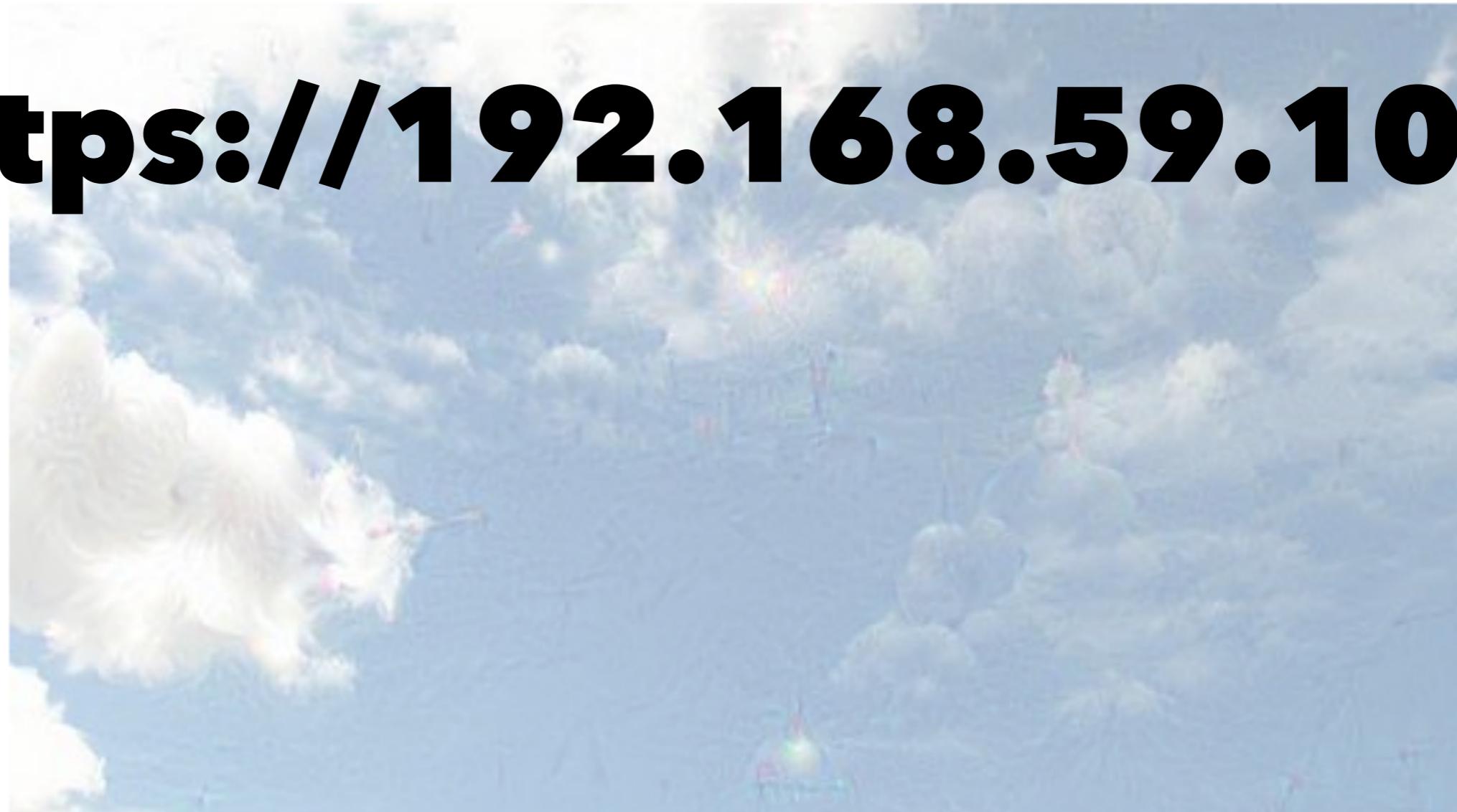
File Edit View Insert Cell Kernel Help



Running the next code cell starts the detail generation process. You may see how new patterns s

In [*]: `_=deepdream(net, img)`

https://192.168.59.103



더 자세히 이해하기

Running deep dream on Windows and OSX

Gitlab

난이도: 최상

Gitlab?

- Github와 비슷한 설치형 오픈소스 Git 호스팅 서비스
- 서비스도 있고, 직접 설치해서 사용하는 것도 가능
- 개발자에게도 어려운 설치
 - 한때 설치하기 어렵기로 정평이 나있었음
- Git, ssh, Database, Redis, Ruby, Rails...

sameersbn/docker-gitlab

```
# Run Postgres Container
$ docker run --name gitlab-postgresql -d \
  --env 'DB_NAME=gitlabhq_production' \
  --env 'DB_USER=gitlab' --env 'DB_PASS=password' \
  --volume /srv/docker/gitlab/postgresql:/var/lib/postgresql \
  sameersbn/postgresql:9.4-2

# Run Redis Container
$ docker run --name gitlab-redis -d \
  --volume /srv/docker/gitlab/redis:/var/lib/redis \
  sameersbn/redis:latest # Redis Container

# Run Gitlab Container
$ docker run --name gitlab -d \
  --link gitlab-postgresql:postgresql --link gitlab-redis:redisio \
  --publish 10022:22 --publish 10080:80 \
  --env 'GITLAB_PORT=10080' --env 'GITLAB_SSH_PORT=10022' \
  --volume /srv/docker/gitlab/gitlab:/home/git/data \
  sameersbn/gitlab:7.13.3
```

nf: falling back to frontend: Teletype
ing SSH2 RSA key; this may take some time ...
ing SSH2 DSA key; this may take some time ...
ing SSH2 ECDSA key; this may take some time ...
ing SSH2 ED25519 key; this may take some time ...
e-rc.d: policy-rc.d denied execution of restart.
ng for database server to accept connections
ng up GitLab for firstrun. Please be patient, this could take a while...
bhq_production already exists
ting database...
ling assets. Please be patient, this could take a while...
ng for database server to accept connections
ng up GitLab for firstrun. Please be patient, this could take a while...
ting database...
ling assets. Please be patient, this could take a while...
ing supervisord.
08-07 05:47:17,790 INFO supervisor running as root (no user specified)
08-07 05:47:17,791 CRIT No configuration file "/etc/supervisor/supervisord.conf" found, using
08-07 05:47:17,791 WARN Included extra file "/etc/supervisor/conf.d/unicorn.conf" during parsing
08-07 05:47:17,791 WARN Included extra file "/etc/supervisor/conf.d/nginx.conf" during parsing
08-07 05:47:17,791 WARN Included extra file "/etc/supervisor/conf.d/cron.conf" during parsing
08-07 05:47:17,791 WARN Included extra file "/etc/supervisor/conf.d/sshd.conf" during parsing
08-07 05:47:17,842 INFO RPC interface 'supervisor' initialized
08-07 05:47:17,843 CRIT Server 'unix_http_server' running without any HTTP authentication checking
08-07 05:47:17,843 INFO supervisord started with pid 1
08-07 05:47:18,849 INFO spawned: 'sidekiq' with pid 563
08-07 05:47:18,870 INFO spawned: 'unicorn' with pid 564
08-07 05:47:18,879 INFO spawned: 'cron' with pid 565
08-07 05:47:18,884 INFO spawned: 'nginx' with pid 566
08-07 05:47:18,935 INFO spawned: 'sshd' with pid 568
08-07 05:47:20,130 INFO success: sidekiq entered RUNNING state, process has stayed up for > than 1 seconds (starts
08-07 05:47:20,131 INFO success: unicorn entered RUNNING state, process has stayed up for > than 1 seconds (starts
08-07 05:47:20,131 INFO success: cron entered RUNNING state, process has stayed up for > than 1 seconds (starts
08-07 05:47:20,131 INFO success: nginx entered RUNNING state, process has stayed up for > than 1 seconds (starts
08-07 05:47:20,131 INFO success: sshd entered RUNNING state, process has stayed up for > than 1 seconds (starts

Wait 5 minute



Welcome to GitLab!

Self hosted Git management application.

You don't have access to any projects right now.

http://192.168.59.103:10080
root / 5iveL!fe

+ New Project



You can create a group for several dependent projects.

Groups are the best way to manage projects and members.

+ New Group

Build once, Run anywhere

감사합니다 :)

propellerheaven@gmail.com