**### Docker Package Installation ###**

**01. Host 장비 인터넷 연결 상태 확인**

# ping www.google.com

>>> [Ctrl + C]

**02. Host 운영체제 커널 버전 및 배포판 리눅스 버전 확인**

# uname -r

# cat /etc/os-release

**03. 레파지토리 정보 갱신**

# apt-get update

**04. https 기반 레파지토리 사용 지원 패키지 설치**

# apt-get install -y apt-transport-https ca-certificates curl software-properties-common

**05. Docker 공식 레파지토리에서 패키지를 다운 시 위변조 확인을 위한 GPG 키를 추가**

# curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

**06. docker.com GPG키 등록 확인**

# apt-key fingerprint

**07. Docker 공식 저장소를 레파지토리로 등록**

# add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

**08. 저장소 등록정보 확인**

# grep docker /etc/apt/sources.list

**09. 레파지토리 정보 갱신**

# apt-get update

**10. Docker Container Engine 설치**

# apt-get install -y docker-ce

**11. 설치 확인**

# docker --version

# docker version

**12. 상세 정보 확인**

# docker info

**13. 부팅 시 Docker Daemon 자동 시작 설정**

# systemctl status docker

>>> [Ctrl + C]

# systemctl enable docker

**### Docker Container ###**

**I. 컨테이너 생성**

**01. Container 및 Image 확인**

# docker container ps -a

# docker image ls

**02. Container 생성** (**이름 지정**)

# docker container create -it --name testos centos

**03. Container 및 Image 확인**

# docker container ps

# docker container ps -a

# docker image ls

**II. 컨테이너 시작**

**01. yumi00 Container 생성**

# docker container create -it --name yumi00 centos

**02. yumi00 Container 생성 확인**

# docker container ps -a

**03. yumi00 Container 시작**

# docker container start -ai yumi00

>>> **Prompt 모양 변경** (컨테이너 진입 상태)

**04. terminal2 오픈 후 두 창에서 명령어 실행 결과 비교**

terminal1, terminal2 # ls /

terminal1, terminal2 # ps -ef

terminal1, terminal2 # hostname

terminal1, terminal2 # ip a

**05. terminal2에서 watch로 컨테이너 상태 1초 주기 모니터링**

terminal2 # watch -n 1 docker container ps -a

**III. 컨테이너 종료**

**01. yumi00 Container 종료**

terminal1 (**Container**) #

>>>Container to Host: [Ctrl + P + Q]

>>> Container에서 Host로 이동

# docker container stop yumi00

**IV. 컨테이너 실행 (일반적)**

**01. yumi01 Container 실행** (**생성 + 시작**)

# docker container run -it --name yumi01 centos

>>> Prompt 모양 변경

>>> [Ctrl + P + Q]

**02. Container 실행 시 시작 프로그램 변경** (Override)

# docker container run -it --name yumi02 centos /bin/date

>>> ‘date’ 실행 후 컨테이너 종료

**V. 컨테이너 detach 모드**

**01. attach mode (기본)**

# docker container run -it --name ping00 centos /bin/ping localhost

>>> [Ctrl + C]

**02. detach mode**

# docker container run -d --name yumi03 centos /bin/ping localhost

>>> Host 상태 (컨테이너로 진입 안함)

**03. yumi03 Container 내부 PID 1번 프로세스 표준출력 확인**

# docker container logs yumi03

# docker container logs --since 20s --until 10s yumi03

# docker container logs -tf yumi03

>>> [Ctrl + C]

**04. yumi03 Container 내부에서 “ls -l /hello.out” 실행**

# docker container exec yumi03 ls -l /hello.out

**05. yumi03 Container 내부에서 “/bin/bash” 실행**

# docker container exec -it yumi03 /bin/bash

>>> (**Container**) # ps -ef

>>> (**Container**) # exit

**06. attach mode로 yumi04 컨테이너 생성**

# docker container run -it --name yumi04 centos

>>> (**Container**) # ps -ef

>>> (**Container**) # [Ctrl + P + Q] (detach mode)

**07. yumi04 Container로 attach (재진입)**

# docker container attach yumi04

>>> (**Container**) # exit

**VI. 컨테이너 삭제**

**01. Container 상태 확인**

# docker container ps -a

**02. Container 삭제**

# docker container rm yumi00

# docker container rm yumi05

# docker container rm -f yumi05

**03. Container 일괄 삭제**

# docker container prune

>>> Up 상태가 아닌 컨테이너 일괄 삭제

# docker container rm -f $(docker container ps -aq)

>>> 상태 상관없이 모두 일괄 삭제

**### Docker Registry ###**

**Docker Hub**

**01. Docker Hub Site 계정 생성**

>>> http://hub.docker.com

**02. 가입 시 실제 수신 가능한 이메일 사용**

>>> 수신된 메일 확인 후 계정 생성 완료

**### Docker Image ###**

**01. Image 정보 확인**

# docker info | grep Registry

# docker image ls

# docker image inspect centos

# docker system df

>>> Image 사용량 확인

# docker image rm centos

>>> Image 삭제 후 사용량 변화 확인

# docker image ls

# docker system df

**02. Image Naming Convention**

**docker.io / library / busybox:latest**

**registry server / repository / image : tag**

**03. Image pull**

# docker image ls

# docker image pull busybox

# docker image ls

# docker image pull **docker.io/library/alpine:3.5**

# docker image ls

**04. Docker Hub Login**

# docker login

# cat /root/.docker/config.json

>>> 로그인 시 사용자 인증정보 로컬 파일로 저장하여 보관

# docker logout

>>> 로그아웃 시 사용자 인증정보 삭제

# cat /root/.docker/config.json

**05. Image tag**

# docker image ls

# docker image tag busybox rusita/box:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

# docker image ls

# docker image tag nginx rusita/web-server:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

# docker image ls

**06. Image push**

# docker image push rusita/box:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

>>> **에러 발생**

# docker login

# docker image push rusita/box:1.0

# docker image push rusita/web-server:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

>>> Docker Hub에서 Image 업로드 확인

**07. Image rm**

# docker image ls

# docker image rm rusita/web-server:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

# docker image rm nginx

>>> Image 삭제

# docker image pull rusita/web-server:1.0

>>> repository 이름을 **본인 Docker Hub ID로 수정**하고 진행

# docker image pull nginx

>>> 다시 다운로드

# docker image ls

**08. 모든 Image 삭제 (주의)**

# docker rmi $(docker images -q)

# docker rmi -f $(docker images -q)

**### WordPress Container 실습 ###**

**01. 작업 디렉토리 경로 확인**

# cd

# pwd

**02. MariaDB 설정 파일 생성**

# vi db-env

===================================================================================

MYSQL\_ROOT\_PASSWORD=somewordpress

MYSQL\_DATABASE=wordpress

MYSQL\_USER=wordpress

MYSQL\_PASSWORD=wordpress

===================================================================================

**03. MariaDB Container 실행**

# docker container run -d --name db --env-file db-env -v /db\_data:/var/lib/mysql mariadb

# docker container ps

**04. WordPress 설정 파일 생성**

# vi wp-env

===================================================================================

WORDPRESS\_DB\_HOST=**db:3306**

WORDPRESS\_DB\_USER=wordpress

WORDPRESS\_DB\_PASSWORD=wordpress

WORDPRESS\_DB\_NAME=wordpress

===================================================================================

**05. WordPress Container 실행**

# docker container run -d --name wordpress --env-file wp-env -p 8080:80 --link db wordpress

# docker container ps

**06. IP Address 확인 후 WordPress 접속 확인**

# docker container inspect wordpress | grep IPAddr

>>> Chrome >>> http://Container\_IP\_Address

* **수고하셨습니다 -**