클라우드컴퓨팅과 AI서비스
(클라우드, 리눅스)

융합학과 권오영

oykwon@koreatech.ac.kr

학습내용

- ❖ 클라우드 컴퓨팅 소개
- ❖ 인공지능 소개
- ❖ 리눅스와 도커에 대한 이해

클라우드 컴퓨팅

고성능에서 고가용성으로

❖ 고성능 시스템보다 필요성이 증가한 시스템
- 웹 서버
- 이메일 서버
- 인트라넷 서버 등

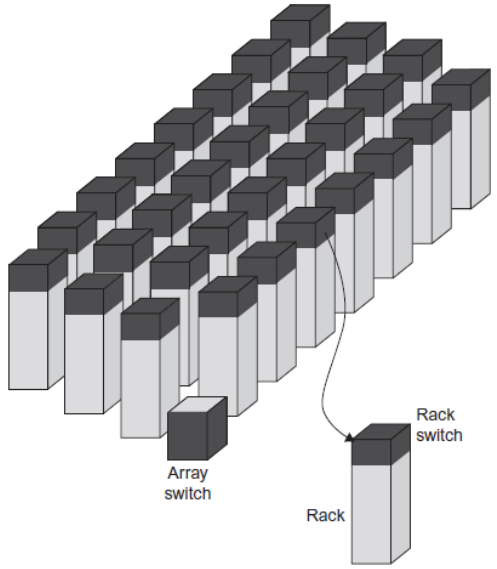
❖ 대부분이 낮은 수준의 병렬 시스템 (Lowly parallel systems)
- 단일 문제의 빠른 처리보다는 많은 사용자들의 요구를 중단없이 지원
- 성능 감소에도 불구하고 지속적으로 서비스로 제공 (가용성)

❖ 가용성이 중요해짐
- 확장성
- 관리성
- 기술 적응성

WSC (Warehouse-Scale Computers)

❖ Warehouse-scale computer (WSC)

- 인터넷 서비스 제공
  - ✓ 검색, 소셜 네트워킹, 온라인지도, 비디오 공유, 온라인 쇼핑, 이메일, 클라우드 컴퓨팅 등
- 고성능 클러스터와 다른점
  - ✓ 고성능 프로세서 및 네트워크로 구성
  - ✓ 클러스터는 스레드(thread) 수준 병렬 처리를 강조하고 WSC는 요청 수준(request-level) 병렬 처리를 강조
- 데이터 센터와 차이점
  - ✓ 데이터 센터는 다른 머신과 소프트웨어를 한 위치에 통합
  - ✓ 데이터 센터는 다양한 고객에게 서비스를 제공하기 위해 가상 머신 및 하드웨어 이종성(heterogeneity)을 강조



❖ WSC의 목표는 warehouse의 하드웨어/소프트웨어가 다양한 응용 프로그램을 실행하는 단일 컴퓨터처럼 작동하도록 하는 것임

클라우드 컴퓨팅

- 정의: 서버나 네트워크, 스토리지와 같은 컴퓨팅 자원을 언제 어디서든 원격의 공유된 풀에서 필요한 경우 요청하여 필요한 만큼 사용하는 모델

- Utility computing; pay-as-you-go service

- 클라우드(원격지)에 중요한 데이터를 모아 놓으면, 해킹에 취약하다는 부정적인 의견이 도입 초기에 많았음
  (In 2011)
  - WSC는 오늘날 대부분의 로컬 데이터 센터보다 훨씬 안전함
  - WSC는 지속적으로 공격을 받고 있으며, 운영자는 보다 신속하게 대응하여 공격을 중단하고 더 나은 방어를 구축

클라우드의 장단점

❖ 장점

- 웹을 통해 서비스를 설정하고 배포가 가능하고 이용이 쉽고 빠름
- 초기 투자 비용이 저렴함
- 무제한적인 확장성 및 탄력성

❖ 단점

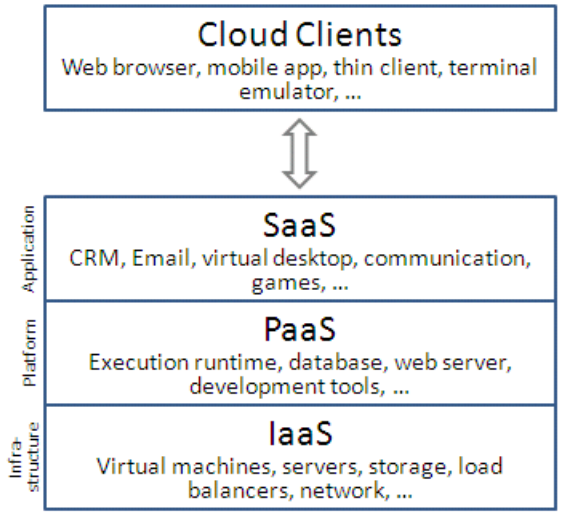
- 서버 비용 외 네트워크, 디스크 등에 대한 비용이 추가
- 다른 사용자와 물리적인 하드웨어를 공유하기 때문에 절대적인 성능을 예측할 수 없음 (성능저하)
- 가상머신이 사라지거나, 서비스 가용 시간이 100%가 아니기 때문에 불안정함

퍼블릭 클라우드 컴퓨팅 대표적인 예

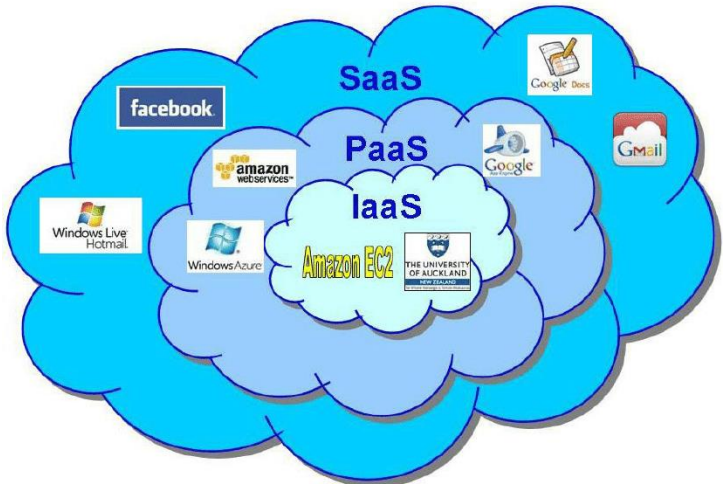
- Amazon Web Services(AWS)
  - 가상머쉰: 리눅스/Xen
  - 저비용
  - (초기) 오픈 소스 소프트웨어에 의존
  - 서비스 (초기) 보증 없음
    - ✓ (원래) 최선의 노력(best-effort)
    - ✓ (최근) 최대 99.95 %의 가용성
  - 계약 불필요 (신용 카드 사용)
  - AWS의 마진은 25 %, 아마존은 3 %
  - AWS는 아마존 수익의 ¾ 을 담당

- 클라우드는 오버 프로비저닝 또는 언더 프로비저닝의 위험을 회피
  - 스타트 업 위험 회피 (Netflix)
  - Zynga : AWS-> 자체 데이터 센터-> AWS

서비스 단계에 따른 분류



(출처:https://ko.wikipedia.org/wiki/클라우드_컴퓨팅)



https://www.researchgate.net/figure/Forms-of-Cloud-computing-SaaS-Software-as-a-Service-PaaS-Platform-as-a-service-IaaS_fig2_48333206

KOREATECH 한국기술교육대학교

인공지능

철학적 관점의 인공지능

❖ 약 인공지능Weak AI

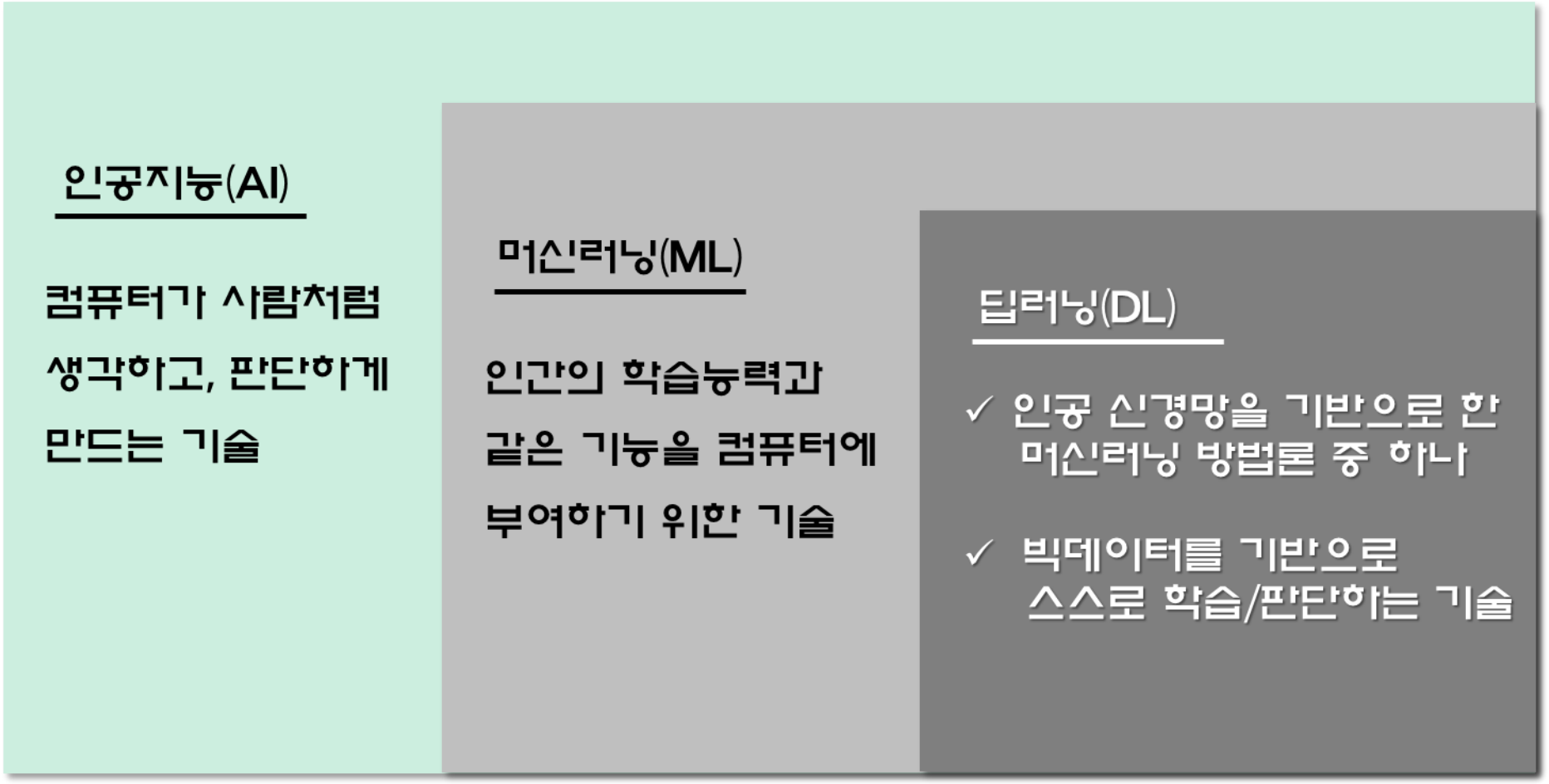
- 유사표현 : 협소적Narrow 인공지능
- 인간의 구현을 목적으로 하는 것이 아니며 기존의 인간 지능으로만 가능하던 작업의 일부를 컴퓨터로 수행하게 만드는 것
  - ✓ 정해진 틀에서 특정 업무만 수행
  - ✓ 예측과 관리가 수월
- 현재까지 인간이 만들어낸 거의 대부분의 AI는 약 인공지능

❖ 강 인공지능Strong AI

- 유사표현 : 범용 인공지능
- 인간의 지성 전체를 컴퓨터의 정보처리능력으로 구현했거나 구현하는 것을 목적으로 한 시스템
- 인간처럼 이성적/감성적으로 사고하고 판단하는 시스템
  - ✓ 규칙성을 벗어나 능동적으로 학습 가능

출처 : 나무위키, 위키피디아

인공지능-머신러닝-딥러닝



리눅스

운영체제

❖ 운영체제: 리눅스 (안드로이드)

- 일반 리눅스
  - ✓ 일반 테스크 탑 환경인 고성능 프로세서와 대용량 메모리 환경에서 동작 하는 범용 컴퓨터용 리눅스
- 임베디드 리눅스
  - ✓ 저성능의 마이크로 프로세서와 제한된 메모리 환경에서 동작하는 임베디드 시스템용 리눅스

리눅스 역사(history)

❖ Linux 역사
- 핀란드 대학원생 Linus B. Torvalds
- 1991.5 version 0.01 발표
  - ✓ Intel 80386, Minix 파일 시스템, 네트워킹 없음
- 1994.3 리눅스 커널 버전 1.0
  - ✓ 확장된 파일 시스템, 네트워킹 지원
- 1996.6 버전 2.0
  - ✓ Alpha와 x86 이외에 Sparc도 지원, IDE 장치, SCSI 장치, 네트워크 카드 지원, 커널 모듈 지원, 전원 절약 장치, SMP 지원, 여러 파일 시스템, 쿼터, 커널 스레드 지원
- 1999년 2.2 버전의 발표로, 엔터프라이즈 환경에 진입할 수 있는 초석 마련

- A. Tanenbaum 교수의 Minix 기반 (1987) (http://www.cs.vu.nl/~ast/minix.html)
- Philosophy of COPYLEFT(open source)
- GNU support
- Various Distributions : Redhat, Debian, Slackware, Alzza, MontaVista, Lineo, Gmate, Zaurus, Samsung, IBM, ..

리눅스와 GNU

- GNU (GNU's not Unix)
  - 80년대 초반 리차드 스톨만(Richard Stallman)에 의하여 시작
  - GPL (GNU Public License)
    - ✓ GPL에 의거한 모든 소프트웨어는 무료
    - ✓ 변경 사항을 포함해서 재판매 하는 것은 허용하나 소스는 공개해야 함
    - ✓ 프로그래머는 자신의 소프트웨어로 발생하는 어떤 위험이나 손해에 대한 법률적 책임이 없음
  - Linux에 gcc, emacs 등을 이식
  - BSD의 많은 유용한 유틸리티를 포함하게 하는 계기가 됨
- 리눅스는 GPL에 의거하여 배포

리눅스 전체 디렉토리 구조

- /etc - 시스템 설정 파일 디렉토리 (ex: /etc/lilo.conf, /etc/fstab ..)
- /bin - 시스템의 기초 명령 (ex: gzip, su, tar, rpm, vi, mount..)
- /dev - 장치파일 (ex: /dev/hda, /dev/cdrom..)
- /lib - 시스템 공유 라이브러리 디렉토리
- /home - 사용자 홈 디렉토리 (ex: /home/ssyoo, /home/pirami..)
- /root - 루트사용자의 홈 디렉토리
- /proc - 시스템 정보 디렉토리 (ex: /proc/cpuinfo, proc/meminfo..)
- /sbin - 시스템 관리 명령 디렉토리
- /tmp - 임시파일 생성 디렉토리
- /var - 시스템 가동 중 가변 자료 저장 디렉토리
- /usr - 애플리케이션이 설치되는 디렉토리

  - 커널소스는 보통 /usr/src 에 설치

리눅스 사용을 위한 준비

- SHELL 이해
- 리눅스 명령어 이해
  - 시스템 관리 명령어
  - 네트웍 관리 명령어
  - 파일 관리를 포함한 다양한 리눅스 명령어
  - 에디터, 컴파일러 이해

- 참고

  https://www.edx.org/course/introduction-linux-linuxfoundationx-lfs101x-0

Shell

❖ Shell

- Command interpreter로서 user와 kernel을 연결
- 사용자가 명령어를 입력하면 shell은 특정한 작업을 수행하기 위하여 요구된 기계어의 집합(System Call)으로 명령어를 변환하여 실행
- bourne shell(sh), C shell(csh), Bash, Ksh 등
- 리눅스의 기본 shell : Bash(Bourne Again Shell)
- 셸은 인식할 수 있는 "스크립트(Script)" 파일을 만들어서 리눅스의 일반적인 명령어뿐만 아니라 shell script 언어도 포함시킬 수 있음
- Shell 스크립트는 "ed", "vi" 편집기를 사용하여 작성 가능

Shell 사용

❖ Bash 설정 방법

- Bash와 관련된 파일들 - .bashrc, .bash_logout, .bash_profile
- .bashrc
  - ✓ 자주 사용하는 명령어의 alias 설정
- .bash_profile
  - ✓ program이나 library에 대한 path 설정

```
# .bashrc
# User specific aliases and functions

alias ifc='ifconfig'

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
export
```

❖ ifconfig 에 대한 alias 설정

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    ~/.bashrc
fi
# User specific environment and startup programs
PATH=$PATH:$HOME/bin:/usr/local/arm-linux/bin:
BASH_ENV=$HOME/.bashrc
USERNAME=""
      export USERNAME BASH_ENV PATH
```

❖ arm-linux 관련 명령어 path 설정
(/usr/local/arm-linux/bin)

리눅스 기본 명령어

- 간단한 리눅스 명령어들
  - ls, more, cat, grep, who, pwd
  - ps(process status), su(super user),
  - chown(change ownership), chmod(change mode)
  - useradd, userdel, passwd

- 시스템 관리 명령어들
  - mount - 디바이스 파일 시스템을 리눅스 메인 파일 시스템에 설치
  - unmount - 디바이스 파일 시스템 해제
  - e2fsck - 시스템 파일 검사
  - mke2fs - 새로운 시스템 파일 만들기
  - mknod - FIFO 파이프나 캐릭터나 블록 모드 디바이스 파일 생성

- 네트웍 관리 명령어들
  - ping - 도착지로 요청을 보내고 회신을 받는 일(네트웍 연결 상태 확인)
  - ifconfig - 네트웍 인터페이스 구성정보
  - netstat - 모든 내트워크 연결상태에 대한 요약정보

gcc 사용법

- GNU C Compiler , GNU Compiler Collection 의미
- FSF (Free Software Foundation)의 C, C++ 컴파일러
- 컴파일러, 어셈블러, 로더 역할을 함
- 한 개의 파일(hello.c) 를 컴파일 하기
  - $gcc –o hello hello.c
    - ✓ -o 옵션 : 실행파일명을 지정하는 옵션
    - ✓ -S 옵션 : c언어 컴파일 과정 까지만 처리 -> 어셈블리 코드 출력
- 모든 컴파일 과정을 한번에 처리하기
  - $ gcc –o hello hello.c hello_world.c
    - ✓ -o 옵션이 없으면 a.out이 실행 파일임
- 컴파일러 매크로
  - 컴파일러가 파일을 제어할 때 사용하는 정보
  - -D 옵션을 사용한 매크로 정의
    - ✓ $ gcc –c –DEM_FILE=“embedded” helloworld.c
    - ✓ -DEM_FILE은 소스에서 #define EM_FILE embedded와 동일한 결과
  - -U 옵션은 매크로 정의를 해제함 - #undef와 동일한 효과임

Linux Commands Cheat Sheet

Easy to use Linux shortcuts
for developers.

RED HAT DEVELOPERS

ssh [ip or hostname] **"vagrant ssh" in the same directory as the Vagrantfile to shell into the box/machine (assumes you have successfully "vagrant up")**	Secure shell, an encrypted network protocol allowing for remote login and command execution On Windows: PuTTY and WinSCP An "ssh.exe" is also available via Cygwin as well as with a Git installation.
pwd	Print Working Directory Displays the full path name
whoami	Displays your logged in user id
cd / **cd target** **cd ~**	Change directory to the root of the filesystem Change directory to "target" directory Change directory to your home directory
ls **ls -l** **ls -la**	Directory listing Long listing, displays file ownership Displays hidden files/directories

```
[vagrant@rhel-cdk /]$ ls
bin  boot  dev  etc  home  lib  lib64  lost+found  media  mnt  opt  pro
[vagrant@rhel-cdk /]$ ls -l
total 62
lrwxrwxrwx.   1 root    root        7 Mar  8 20:36 bin -> usr/bin
dr-xr-xr-x.   4 root    root     1024 Mar 12 19:26 boot
drwxr-xr-x.  18 root    root     3100 Mar 12 19:49 dev
drwxr-xr-x.  85 root    root     4096 Mar 12 19:31 etc
drwxr-xr-x.   3 root    root     4096 Mar  8 20:54 home
lrwxrwxrwx.   1 root    root        7 Mar  8 20:36 lib -> usr/lib
lrwxrwxrwx.   1 root    root        9 Mar  8 20:36 lib64 -> usr/lib64
drwx------.   2 root    root    16384 Mar  8 20:34 lost+found
drwxr-xr-x.   2 root    root     4096 May 25  2015 media
drwxr-xr-x.   2 root    root     4096 May 25  2015 mnt
drwxr-xr-x.   3 root    root     4096 Mar  8 20:50 opt
dr-xr-xr-x. 166 root    root        0 Mar 12 19:26 proc
dr-xr-x---.   3 root    root     4096 Mar 12 19:38 root
drwxr-xr-x.  31 root    root     1000 Mar 12 19:31 run
lrwxrwxrwx.   1 root    root        8 Mar  8 20:36 sbin -> usr/sbin
drwxr-xr-x.   2 root    root     4096 May 25  2015 srv
dr-xr-xr-x.  13 root    root        0 Mar 12 19:26 sys
drwxrwxrwt.   7 root    root     4096 Mar 12 20:31 tmp
drwxr-xr-x.  13 root    root     4096 Mar  8 20:36 usr
drwxr-xr-x.   3 vagrant vagrant  4096 Mar 12 19:25 vagrant
drwxr-xr-x.  19 root    root     4096 Mar 12 19:26 var
[vagrant@rhel-cdk /]$
```

clear	Clear the terminal screen

1

Opensource.com: Linux Common Commands Cheat Sheet

BY SETH KENLON

Use this handy Linux command cheat sheet for executing common tasks such as navigating files, installing software, and starting services.

NAVIGATE FILES

LIST DIRECTORIES (WITH TYPE INDICATOR)
```
$ ls --file-type
```

CHANGE DIRECTORY TO "EXAMPLE"
```
$ cd example
```

MOVE UP ONE DIRECTORY
```
$ cd ..
```

MOVE UP TWO DIRECTORIES
```
$ cd ../..
```

CHANGE TO HOME DIRECTORY
```
$ cd ~
```

GET CURRENT DIRECTORY
```
$ pwd
```

GET ABSOLUTE PATH TO A FILE OR FOLDER
```
$ readlink -f example
```

GET FILE TYPE OF "EXAMPLE.EXT"
```
$ file example.ext
```

INSTALLING SOFTWARE

- On Fedora and CentOS, [COMMAND] is dnf
- On Ubuntu and Debian, [COMMAND] is apt
- On OpenSUSE, [COMMAND] is zypper
- Other distributions may use different commands

SEARCH FOR AN APPLICATION CALLED EXAMPLE
```
$ sudo [COMMAND] search example
```

INSTALL AN APPLICATION CALLED EXAMPLE
```
$ sudo [COMMAND] install example
```

UNINSTALL AN APPLICATION CALLED EXAMPLE
```
$ sudo [COMMAND] remove example
```

SERVICES

START SERVICES
```
$ sudo systemctl start example
```

STOP SERVICES
```
$ sudo systemctl stop example
```

GET STATUS OF SERVICES
```
$ sudo systemctl status example
```

FILE MANAGEMENT

COPY A FILE IN PLACE
```
$ cp example.txt example-1.txt
```

COPY A FILE TO DOCUMENTS
```
$ cp example.txt ~/Documents/example-1.txt
```

MOVE A FILE TO DOCUMENTS
```
$ mv example.txt ~/Documents
```

CREATE A DIRECTORY (FOLDER)
```
$ mkdir example
```

REMOVE AN EMPTY DIRECTORY
```
$ rmdir example
```

SAFELY REMOVE A FILE
```
$ trash example.txt
```

REMOVE A FILE (WITHOUT TRASH COMMAND)
```
$ mv example.txt ~/.local/share/Trash/files
```

PERMANENTLY DELETE A FILE
```
$ shred example.txt
```

DOWNLOAD A FILE FROM AN NETWORK LOCATION
```
$ wget http://example.com/file
```

opensource.com Twitter @opensourceway | facebook.com/opensourceway | CC BY-SA 4.0

도커(DOCKER)

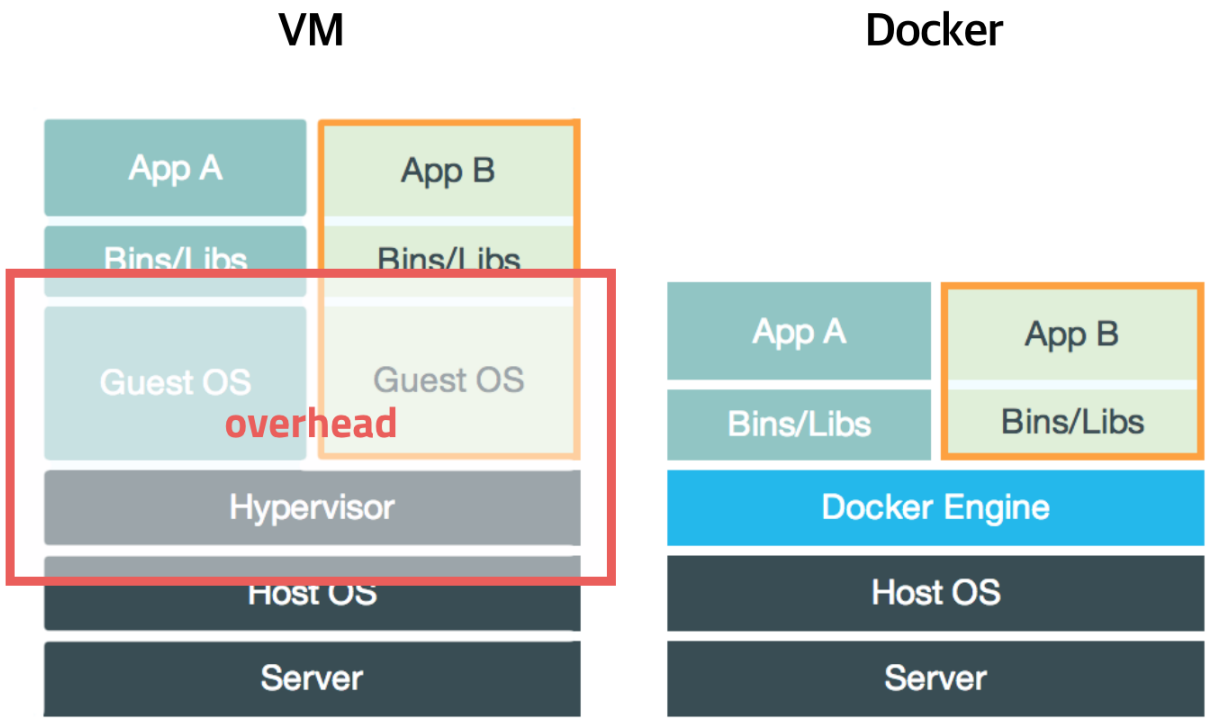
도커

❖ 서버관리(bare metal)의 어려움

- 리눅스 서버의 셋팅은 버전이 바뀌거나 환경이 달라지면 문제가 발생할 가능성이 있음
- 서버 환경이 바뀌면서 지속적으로 서버 관리에 필요한 새로운 도구들이 소개되고, 이들 도구들을 습득해야함
- 하나의 서버에 여러 개의 프로그램을 설치할 경우 각각의 프로그램이 필요로 하는 라이브러리의 버전이 상이하고, 동일한 포트를 요청하는 등 여러 문제를 야기할 수 있으며, 하나의 서버에 하나의 프로그램을 설치해야만 하는 상황이 발생하여 자원 낭비를 초래함
- DevOps의 등장으로 개발과 배포의 주기가 단축되고, 클라우드의 발전으로 수백에서 수천대에 이르는 서버에 프로그램을 신속히 설치하고 운영, 관리하는 문제가 발생

도커

❖ 도커는 2013년 3월 산타클라라에서 열린 Pycon Conference에서 dotCloud의 창업자인 Solomon Hykes가 처음 발표



❖ 도커는 컨테이너 기반의 오픈소스 가상화 플랫폼

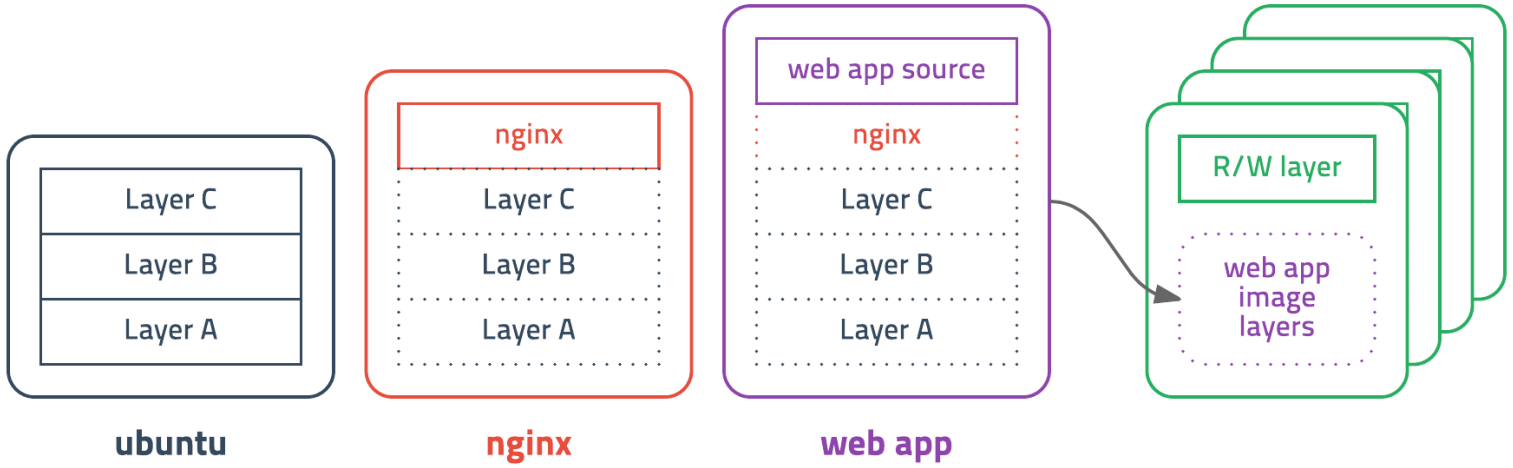
- 컨테이너는 격리된 공간에서 프로세스가 동작하는 기술
- 기존의 가상화 방식은 주로 OS를 가상화한 것으로, VMware나VirtualBox같은 가상머신은 호스트 OS위에 게스트 OS 전체를 가상화하여 사용하는 방식
- 게스트 OS가 필요하긴 하지만 전체OS를 가상화하는 방식이 아니였기 때문에 호스트형 가상화 방식에 비해 성능이 향상

도커

- 하나의 서버에 여러 개의 컨테이너를 실행하면 서로 영향을 미치지 않고 독립적으로 실행되어 마치 가벼운 Virtual Machine을 사용하는 것과 같음
  - 실행중인 컨테이너에 접속하여 명령어를 입력할 수 있고 apt-get이나 yum으로 패키지를 설치할 수 있으며 사용자도 추가하고 여러 개의 프로세스를 백그라운드로 실행할 수도 있음
  - CPU나 메모리 사용량을 제한할 수 있고 호스트의 특정 포트와 연결하거나 호스트의 특정 디렉토리를 내부 디렉토리인 것처럼 사용할 수도 있음
- 도커이미지
  - 이미지는 컨테이너 실행에 필요한 파일과 설정값 등을 포함하고 있는 것으로 상태값을 가지지 않고 변하지 않음(Immutable)
  - 컨테이너는 이미지를 실행한 상태라고 볼 수 있고 추가되거나 변하는 값은 컨테이너에 저장됨
  - 하니의 이미지에서 여러 개의 컨테이너를 생성할 수 있고 컨테이너의 상태가 바뀌거나 컨테이너가 삭제되더라도 이미지는 변하지 않고 그대로 유지됨

도커이미지

- 이미지는 컨테이너를 실행하기 위한 모든 정보를 가지고 있음
- 도커 이미지는 Docker hub(https://hub.docker.com)에 등록하거나 Docker Registry를 직접 만들어 관리할 수 있음
- 도커는 레이어라는 개념을 사용하고 유니온 파일 시스템을 이용하여 여러 개의 레이어를 하나의 파일시스템으로 사용할 수 있게 해줌
  - 이미지는 여러 개의 읽기 전용(read only) 레이어로 구성되고 파일이 추가되거나 수정되면 새로운 레이어가 생성됨



도커이미지

❖ 도커는 이미지를 만들기 위해 Dockerfile 이라는 파일에 자체 DSL (Domain-specific language)언어를 이용하여 이미지 생성 과정을 기술

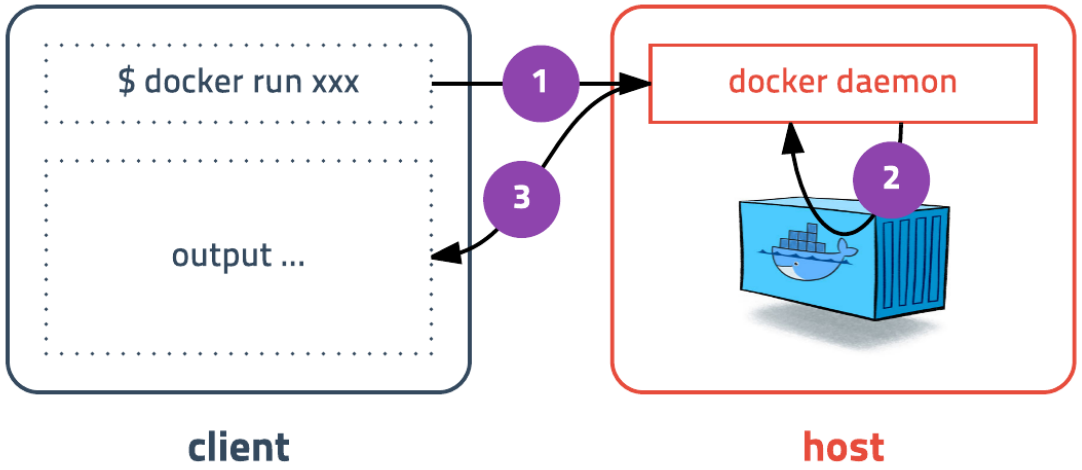
- Dockerfile을 작성하는 법은 docker 사이트의 Dockerfile reference(https://docs.docker.com/engine/reference/builder/)를 참고

```
# vertx/vertx3 debian version
FROM subicura/vertx3:3.3.1
MAINTAINER chungsub.kim@purpleworks.co.kr

ADD build/distributions/app-3.3.1.tar /
ADD config.template.json /app-3.3.1/bin/config.json
ADD docker/script/start.sh /usr/local/bin/
RUN ln -s /usr/local/bin/start.sh /start.sh

EXPOSE 8080
EXPOSE 7000
CMD ["start.sh"]
```

도커설치

- docker는 기본적으로 root권한이 필요
- 설치가 완료되었다면 정상적으로 설치되었는지 도커 명령어를 입력해 확인(정상적으로 설치되면 Client와 Server 정보가 출력됨)

  $ docker version
- 도커는 하나의 실행파일이지만 실제로 클라이언트와 서버역할을 각각 수행하며, 도커 커맨드를 입력하면 도커 클라이언트가 도커 서버로 명령을 전송하고 결과를 받아 터미널에 출력



설치된 Linux위에 Docker 설치

❖ Docker 설치

- sudo apt update
- sudo apt install docker.io
- sudo docker version (검증)
- sudo docker run --rm -it ubuntu /bin/bash

❖ sudo를 없애기 위해 $USER를 docker 그룹에 추가

- sudo usermod -aG docker $USER
- 재로그인해야만 sudo 없이 docker 수행가능

❖ sudo apt install docker-compose

도커 실행 명령어

$ docker run [OPTIONS] IMAGE[:TAG|@DIGEST] [COMMAND] [ARG...]

옵션	설명
-d	detached mode 흔히 말하는 백그라운드 모드
-p	호스트와 컨테이너의 포트를 연결 (포워딩)
-v	호스트와 컨테이너의 디렉토리를 연결 (마운트)
-e	컨테이너 내에서 사용할 환경변수 설정
-name	컨테이너 이름 설정
-rm	프로세스 종료시 컨테이너 자동 제거
-it	-i와 -t를 동시에 사용한 것으로 터미널 입력을 위한 옵션
-link	컨테이너 연결 [컨테이너명:별칭]

❖ ubuntu 실행

$ docker run --rm -it ubuntu:16.04 /bin/bash

기본명령어

- 컨테이너 목록 확인 (ps)

  $ docker ps [OPTIONS] # 옵션: -a, --all

- 컨테이너 중지 (stop)

  $ docker stop [OPTIONS] CONTAINER [CONTAINER...]

- 컨테이너 제거 (rm)

  $ docker rm [OPTIONS] CONTAINER [CONTAINER...]

- 이미지 목록 확인 (images)

  $ docker images [OPTIONS] [REPOSITORY[:TAG]]

  - 도커가 다운로드하여 로컬 저장소에 있는 이미지들의 목록을 보여줌

- 이미지 다운로드 (pull)

  $ docker pull [OPTIONS] NAME[:TAG|@DIGEST]

  (예) $ docker pull ubuntu:16.04)

  - run 명령어를 입력하면 이미지가 없을 때 자동으로 다운로드 되며, pull 명령어는 최신 버전으로 다시 다운로드 받을 때 사용

기본명령어

- 이미지 삭제하기 (rmi)

  $ docker rmi [OPTIONS] IMAGE [IMAGE...]

- 컨테이너 명령어 실행 (exec)

  $ docker exec [OPTIONS] CONTAINER COMMAND [ARG...]

  - run 명령은 새로 컨테이너를 만들어서 실행하고, exec는 실행중인 컨테이너에 명령을 보낼 때 사용

- 호스트 디렉토리를 컨테이너에 마운트해서 사용하기 위해 -v 명령어를 사용함 (-v hostDir:containerDir)

  $ docker run -d -p 3306:3306 -v /my/own/datadir:/var/lib/mysql ...

  - 호스트의 /my/own/datadir를 컨테이너의 /var/lib/mysql 디렉토리에 연결하여 컨테이너에서 저장한 데이터가 호스트에 저장될 수 있도록 함

- docker의 커맨드라인 명령어작업이 복잡해지면, docker의 실행 명령을 yaml 방식으로 설정한 파일(docker-compose.yml)을 만들어서 docker를 실행하는 docker-compose 도구도 있음

가상머쉰에 리눅스 설치하기

Virtual Box

- 위키의 정의(https://ko.wikipedia.org/wiki/버추얼박스)
- **버추얼박스**(VirtualBox)는 본래 이노테크(InnoTek)가 개발한 뒤, 현재는 오라클이 개발 중인 상용, 사유 소프트웨어(제한된 GPL 버전)로, 리눅스, macOS, 솔라리스, 윈도우를 게스트 운영 체제로 가상화하는 x86 가상화 소프트웨어이다. 개발된 지 몇 해가 지나, 제한된 오픈 소스 버전인 버추얼박스 OSE가 GPL 하에 2007년 1월에 공개되었다.
- 버추얼박스는 인텔의 하드웨어 가상화 VT-x와 AMD의 AMD-V를 지원한다.
- 2008년 2월 12일, 썬 마이크로시스템즈는 버추얼박스를 개발한 회사인 이노텍(Innotek)을 인수하였다. 2009년 4월 20일 오라클이 썬 마이크로시스템즈를 인수하면서 현재 Oracle VM VirtualBox로 배포되고 있다.

WSL 추천

- Windows Subsystem for Linux 사용 추천
- VirtualBox나 VMWare사용하지 않고 windows 10 이상에서 설치가능
- https://learn.Microsoft.com/ko-kr/windows/wsl/install

Summary

- 클라우드 컴퓨팅, 인공지능에 대한 간략한 소개
- 리눅스와 도커 이해
- VirtualBox 설치하고 Ubuntu (Linux OS)와 도커를 설치해보자.
- WSL2를 설치할 수 있으면 설치해볼 것을 권장
- 교내클라우드 사용을 위해 교내이메일 주소를 id로 활용할 계획임