용어정리

1. Virtual Box

리눅스, macOS, 윌리암스, 윈도를 게스트 운영체제로 가상화 하는 x86 가상화 오픈소스 하이퍼바이저

현재 수업에서 Virtual Box를 통해 Linux - Ubuntu의 Virtual Machine을 만들어 실습 진행 중이다.

2. AWS, GCP

1) Amazon Web Services : 클라우드 컴퓨팅 서비스 사이트로, 다른 웹사이트나 클라이언트 측 응용프로그램이 대화 온라인 서비스를 제공하는 API다. 아마존 웹 서비스의 자동 서비스는 REST 프로토콜 및 SOAP 프로토콜을 통해 접근, 이용 및 관리가 가능하다. (유료)

2) Google Cloud Platform : 구글 검색과 유튜브 같은 최종 사용자 제품을 위해 구글이 사용하는 동일한 라우팅 인프라스트럭처 위에서 호스팅을 제공하는 구글의 클라우드 컴퓨팅 서비스.
단순 웹 서비스 ~ 복잡한 어플리케이션에 이르는 일련의 프로그램 빌드를 개발과 제품을 제공한다.

정리하기

1. VM 복제하여 새로운 서버 구성

1) 복제 원본이 될 VM의 시스템을 종료한 후에 마우스 우클릭하여 '복사' 선택

2) 완전한 복제 선택 후 복제 완료시 "hostnamectl set-hostname 원하는 hostname"으로 hostname 변경

3) 네트워크 설정
① sudo vi /etc/netplan/00-installer-config.yaml 로 들어가서 IP주소 변경
② sudo netplan apply 후 PuTTY 통해 접속해 본 확인
③ Virtual Box 내 Virtual Machine에서 ifconfig 명령으로 IP 주소 정상 변경 확인
④ Virtual Box 전체 환경설정 > 네트워크 > 네트워크 추가 포트포워딩
⑤ host IP 공란으로 둠 host port number는 서로 겹치지 않게 ssh, apache2, mysql 등 원본과 복제본의 port 설정
⑥ Virtual Box 내 원본 Virtual machine 과 복제 Virtual machine 각각의 "환경설정 > 네트워크"에서 NAT → NAT 네트워크로 변경
위에서 생성한 네트워크 이름 선택 > 저장
⑦ Virtual machine에서 원본 ↔ 복제 ping 보내고받고 확인
⑧ 각 포트포워딩한 번호로 PuTTY 접속 확인

2. AWS / GCP 설정

1) AWS

① 서비스 가입 > 서비스 > Compute > EC2 로 이동
② 인스턴스 카테고리에서 인스턴스 만들기 : Ubuntu 20.04 / 프리티어 사용가능유형
③ 인스턴스 생성하면서 새 키페어 생성 이름 PC 저장
④ 인스턴스 ID 클릭 > 연결 클릭하여 브라우저에서 리눅스 연결 테스트 확인
⑤ PuTTY KEYGEN에서 Private key 생성
⑥ PuTTY 접속하여 SSH > Auth > Browse keygen에서 만든 Private key 파일 등록 및 PuTTY 연결하기 (HostIP=AWS IP)
⑦ Conversions > Export OpenSSH key 생성
⑧ 노트패드에서 key file 열고 FTP로 kopo 서버로 접속하여 .ssh 파일에 key file 저장
⑨ PuTTY 접속 : ssh -i ./키파일명 ubuntu (AWS hostname)@AWS IP주소 로 AWS PuTTY에서 접속확인

2) GCP

① GCP 접속하여 VM 인스턴스 생성 - 시리즈 : N2 / 머신유형 : f1-micro (무료버전) / 운영체제 : Ubuntu / 버전 : 20.04
② VM 인스턴스 생성중 웹상 잘 작동하는지 SSH에 접속해서 확인
③ PuTTY에서 "ssh-keygen -t rsa -f ~/.ssh/임의디렉토리명 -C GCP상 hostname" 명령으로 key 생성
④ 생성한 key 중 ".pub"으로 되어있는 key Cat으로 읽어서 내용 복사
⑤ GCP 메타데이터에 key 입력 및 등록
⑥ PuTTY에서 "ssh -i ~/.ssh/임의디렉토리명 GCP hostname@GCP에서 부여받은 IP 주소"로 GCP 접속

3. Docker 에서 Apache2, tomcat 컨테이너 구성

1) Docker 설치

① sudo apt update
sudo apt install apt-transport-https
" ca-certificates
" curl
" gnupg
" lsb-release
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
echo \
"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io

↳ 순서대로 설치 한 후 docker 입력하여 docker 관련내용 뜨는지 확인

2) Apache2

① sudo docker run -dit -p 81:80 Ubuntu:20.04
② sudo docker ps -a 로 ID와 상태가 Up 인지 (Run 중인지) 확인
③ sudo docker exec -it ID이름 혹은자 /bin/bash 로 container 접속
④ apt update / install apache2 로 apache 설치하기 > service apache2 restart로 아파치 재시작
⑤ 전체 Virtual Box 환경설정 > 네트워크 > 네트워크 선택 후 포트포워딩
⑥ 웹브라우저에서 내 IP : 포트포워딩한 포트넘버 (ex. 192.168.23.34:82) 입력하여 아파치 연결여부 확인

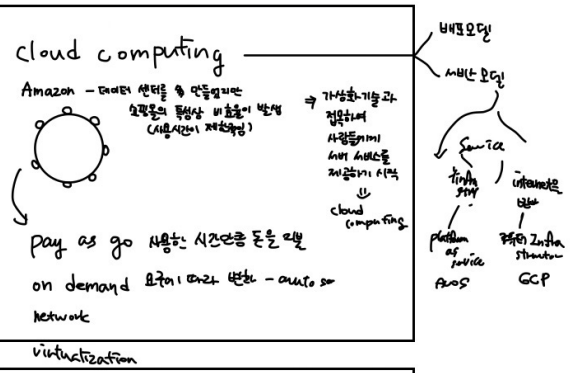
3) Tomcat

① mkdir -p datadir/ROOT 로 해당 경로 한번에 만들기
② sudo docker run -dit -P 8081:8080 -v /home/kopo08/datadir:/usr/local/tomcat/webapps tomcat:9
디렉토리 공유
③ sudo docker ps -a 로 docker 정상 생성 및 작동여부 확인
④ 만들어둔 공유 디렉토리 datadir/ROOT로 들어가서 index.jsp 파일 만들고 저장
⑤ 전체 Virtual Box 환경설정 > 네트워크 > 네트워크 선택 후 포트포워딩
웹브라우저에서 내 IP : 포트포워딩한 포트넘버 (ex. 192.168.23.34:8082) 입력하여 톰캣 연결여부 확인

2. 생각해볼 문제 및 용어

학습전 생각해볼 문제

- VirtualBox 복제등을 찾아보고 학습해 봅니다.
- AWS,GCP에 대하여 알아보고 설정방법을 미리 학습해 봅니다.

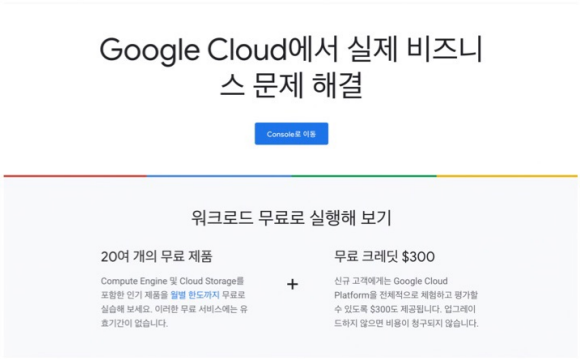


용어 (강의 정리 시 필기 할 것)

- **Virtual Box**
- **AWS,GCP**

3. 이해하기

- **GCP가입** (Google — AWS Amazon)
  - **Google Cloud Platform(GCP)**
    - **Compute**
    - **Storage**
    - **Database**
    - **Networking**
    - **…**
  - **가입 URL : https://console.cloud.google.com/**
  - **무료 크레딧 $300 미치 일정 수준까지 무료**
    - **유료 결제되지 않도록 사용량에 주의**



3. 이해하기

$ ssh-keygen -t rsa -f ~/.ssh/comics77 -C comics77
=> ssh-keygen -t rsa -f ~/.ssh/{파일이름 아무거나} –C {GCP에서 username 확인 후 기입}
$ cat ~/.ssh/comics77.pub
=> 복사 후 gcp – 메타데이터 – ssh에 등록
$ssh -i ~/.ssh/comics77 comics77@35.184.167.16

keygen 명령어
↓
private key → x .
public key → .pub

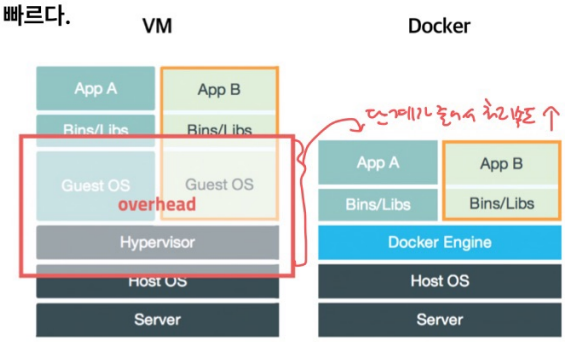
3. 이해하기

- **Docker란**
  - LXC(리눅스 컨테이너)라는 커널 컨테이너 기술을 이용하여 만든 컨테이너 기술 중 하나. 만들어지는데 빠름
  - 컨테이너 기반의 오픈 소스 가상화 플랫폼이며 현재 리눅스 컨테이너 기술 부분에서 사실상 업계 표준이 됨
  - 일반적으로 서버를 관리하는 것은 굉장히 복잡하고 어려워 고급 개발자들의 섬세한 작업이 필요한 영역
  - 도커는 기존의 서버 관리 방식을 완전히 뒤바꿈
  - 다양한 프로그램 실행 환경을 컨테이너로 추상화하고 동일한 인터페이스를 제공하여 프로그램의 배포 및 관리를 단순화 서버가 여러개면 호환, 상호영향 등의 문제로 어려웠는데 Docker 생긴 후로 좋아짐
  - 백엔드 프로그램, 데이터베이스 서버, 메시지 큐 등 어떤 프로그램도 컨테이너로 추상화 가능하며 조립PC, AWS, Azure, GCP 등 어떤 환경에서나 실행 가능

3. 이해하기

- Container
  - 기존의 가상화 (VMware, VirtualBox 등) 방식은 OS를 가상화
  - 여러가지 OS를 가상화 할 수 있고 비교적 사용법이 간단하지만 무겁고 느리다.

- 성능 문제를 개선하기 위해 격리된 공간에서 process 가 동작하는 방식의 가상화를 Linux container 라고 한다.
  - 가볍고 빠르며 CPU나 메모리는 process 가 필요한 만큼만 추가로 사용
  - 하나의 서버에 여러 개의 container 를 실행하면 서로 독립적으로 실행
  - 실행 중인 container 에 접속해 명령어 입력 가능하며 apt-get, yum으로 패키지 설치, 사용자 추가 등이 가능
  - 새로운 container 생성 시간 1 ~ 2초 정도로 굉장히 빠르다.



3. 이해하기

- Image
  - Docker에서 가장 중요한 개념은 Container와 더불어 Image라는 개념이다.
  - Image : Container 실행에 필요한 모든 파일과 설정 값 등을 포함하고 있는 것
  - 의존성 파일을 컴파일하고 설치할 필요가 없다.
  - Container는 image를 실행한 상태로, 추가되거나 변하는 값은 container에 저장되며 같은 image에서 다수의 container 생성 가능
  - Container의 상태와 무관하게 image는 불변(immutable)
  - 새로운 서버가 추가되면 미리 만들어 놓은 image를 다운 받고 container를 생성하기만 하면 됨.
  - 한 서버에서 다수의 container 실행 가능하며 수천 대의 서버에도 문제 없음
  - Docker image는 docker hub에 등록하거나 docker registry 저장소를 직접 만들어 관리 가능
    - 현재 공개된 docker image는 50만 개 이상, docker hub의 이미지 다운로드 수는 80 억 회 이상

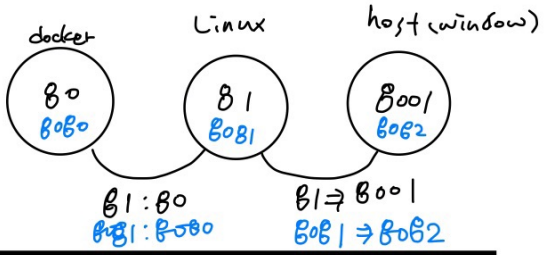
- Docker 는 완전히 새로운 기술이 아니며 이미 존재하는 기술을 잘 포장했다고 볼 수 있다.
- 이미 존재하는 기술을 docker 가 잘 조합하고 사용하기 쉽게 만들었으며 사용자들이 원하는 기능을 간단하지만 획기적인 아이디어로 구현

3. 이해하기

Docker 시작하기

- docker 실행
  - -d : 백그라운드 모드
  - -p : 포트 맵핑
  - ubuntu:20.04 이미지

```
$ docker run –d –p 80:80 ubuntu:20.04
```



- tomcat 9 설치 및 시작

```
kopo26@kopo26copy:~$ docker run -d -p 8081:8080 \
> --name tomcat_docker \
> -v /home/kopo26/datadir:/usr/local/tomcat/webapps \
> tomcat:9
Unable to find image 'tomcat:9' locally
9: Pulling from library/tomcat
57df1a1f1ad8: Already exists
71e126169501: Already exists
1af28a55c3f3: Already exists
03f1c9932170: Already exists
881ad7aafb13: Already exists
9c0ffd4062f3: Already exists
bd62e479351a: Already exists
48ee8bc64dbc: Already exists
07cb85cca4f0: Already exists
6a78fac8d191: Already exists
Digest: sha256:1bab37d5d97bd8c74a474b2c1a62bbf1f1b4b62f151c8dcc472c7d577eb3479d
Status: Downloaded newer image for tomcat:9
ea4f1cc2d5fa23e14e38ce296bdfb4eb379cb21dc1eca6a0052cb3c497bfd011
```