[책]컨테이너 인프라 환경 구축을 위한 쿠버네티스/도커

깃허브 : https://github.com/sysnet4admin/_Book_k8sInfra

▼ 1장

1.1 컨테이너 인프라 환경이란?

- 컨테이너를 중심으로 구성된 인프라 환경
  - 컨테이너: 하나의 운영체제 커널에서 다른 프로세스에 영향을 받지 않고 독립적으로 실행되는 프로세스 상태
  - 컨테이너는 가상화 상태에서 동작하는 프로세스보다 가볍고 빠르게 동작

- 모놀로식 아키텍쳐: 하나의 큰 서비스에 여러기능 통합된 구조(포함)
- **마이크로서비스 아키텍쳐(MSA):** 소규모, 개별 기능의 독립된 서비스를 각각 개발하여 연결
  - 변경→ 다른 서비스에 영향 미칠 ㄱ능성 줄어듬, 사용량 변화에 따라 특정 서비스만 확장 가능
  - 복잡도 높고 각 서비스가 유기적으로 통신하기 때문에 네트워크를 통한 호출 횟수 증가 → 성능 영향 줄수도
  - 우리가 공부할 컨테이너 인프라 환경은 MSA로 구현하기 적합함.
  - 컨테이너 인프라 환경애서는 컨테이너를 서비스단위로 포장해 손쉽게 배포 및 확장 가능

1.2.1 도커 (4장 자세히)

- 컨테이너 인프라 환경 구성:
  - 컨테이너
  - 컨테이너 관리
  - 개발 환경 구성 및 배포 자동화
  - 모니터링
- **도커** : 컨테이너 환경에서 독립적으로 어플리케이션을 실행 할 수있도록 컨테이너 만들고 관리하는 것을 도와주는 컨테이너 도구
  - 도커로 애플리케이션 실행하면 OS에 관계없이 독립적인 환경에서 일관된 결과 보장

1.2.2 쿠버네티스(3장 자세히)

- **쿠버네티스**: 다수의 컨테이너를 관리하는데 사용함(통합 관리 솔루션)
    - 컨테이너의 자동배포, 배포된 컨테이너에 대한 동작 보증, 부하에 따른 동적 확장 등의 기능 제공

1.2.3 젠킨스

- 지속적 통합(CI), 지속적 배포(CD) 지원
- CI, CD는 개발한 프로그램의 빌드, 테스트, 패키지화, 배포 단계를 모두 **자동화**해 개발 단계를 표준화
- 개발된 코드의 빠른 적용, 효과적인 관리 → 개발 생산성을 높임
- 컨테이너 인프라 환경처럼 단일 기능을 빠르게 개발해 적용해야 하는 환경에 매우 적합한 도구
- 비슷한 도구들 : Bamboo, 깃허브 액션, 팀시티 등등

1.2.4 프로메테우스와 그라파나(6장 자세히)

- 모니터링 도구
- 프로메테우스 : 상태 데이터 수집
- 그라파나: 수집한 데이터를 시각화
- 컨테이너 인프라 환경 → 중앙모니터링 필요 → 프로메테우스+그라파나 조합으로 모니터링
    - 프/그는 컨테이너로 패키징돼 동작, 최소한의 자원으로 쿠버네티스 클러스터의 상태를 시각적으로 표현

▼ 2장 테스트 환경 구성하기

2.1 테스트 환경 자동으로 구성하는 도구

- 베이그런트: 코드로 인프라 생성할 수 있게 지원하는 sw
    - 프로비저닝 지원
        - 프로비저닝: 사용자의 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요할 때 시스템을 사용할 수 있는 상태로 만들어 줌
- 버추얼박스: 이노테크에서 개발한 가상화 sw

📌베이그런트 구성하고 테스트 p.40~

- vagrantfile에 기존에 있던 이미지 파일을 추가해 가상머신 생성/삭제 실습
- 프로비저닝 코드 : c:/Hashicorp

```
C:\Users\USER>cd C:\HashiCorp

C:\HashiCorp>vagrant init
A `Vagrantfile` has been placed in this directory. You are now
ready to `vagrant up` your first virtual environment! Please read
the comments in the Vagrantfile as well as documentation on
`vagrantup.com` for more information on using Vagrant.
```

2.2 베이그런트로 테스트 환경 구축하기

1. vagrantfile을 수정해 원하는 구성이 자동으로 CentOS에 입력되도록 해보기(원하는 구성 자동으로 생성하도록)
   - Ruby 언어로 작성
2. 이번엔 CentOS(가상머신)에 추가 패키지 설치하는 법 알아보기
   a. vagrantfile에 셸 프로비전 추가, install_pkg.sh를 CentOS 내부에서 호출해 실행하게 함
   b. install_pkg.sh에는 설치하고자 하는 패키지 다운로드 명령어들이 작성되어있음
3. 가상머신 추가로 구성하기 → 핑테스트로 기존의 vm과 추가된 vm간의 네트워크 통신 작동 잘 하는지 확인해보자!

```
## at vagrantfile
# 게스트 OS가 호출 및 실행
cfg.vm.provision "shell", path: "install_pkg.sh"  #추가패키지설치

# 게스트 OS에 파일 전달(source를 게스트OS의 destination 디렉토리에 전달)
cfg.vm.provision "file", source: "ping_2_nds.sh", destination: "ping_2_nds.sh"  #핑테스트 파일

cfg.vm.provision "shell", path: "config.sh"  #
```

```
# "ping_2_nds.sh"
# ping 3 times per nodes
ping 192.168.1.101 -c 3
ping 192.168.1.102 -c 3
ping 192.168.1.103 -c 3
```

- 핑테스트 파일 → 추가로 설치한 CentOS 3대로 ping을 보내 네트워크가 ㅈ제대로 작동하는지 확인하는 명령어! 0ㅊ 옵션의 몇번의 ping을 보낼것인지 지

```
## "config.sh"
#!/usr/bin/env bash
# modify permission
chmod 744 ./ping_2_nds.sh
```

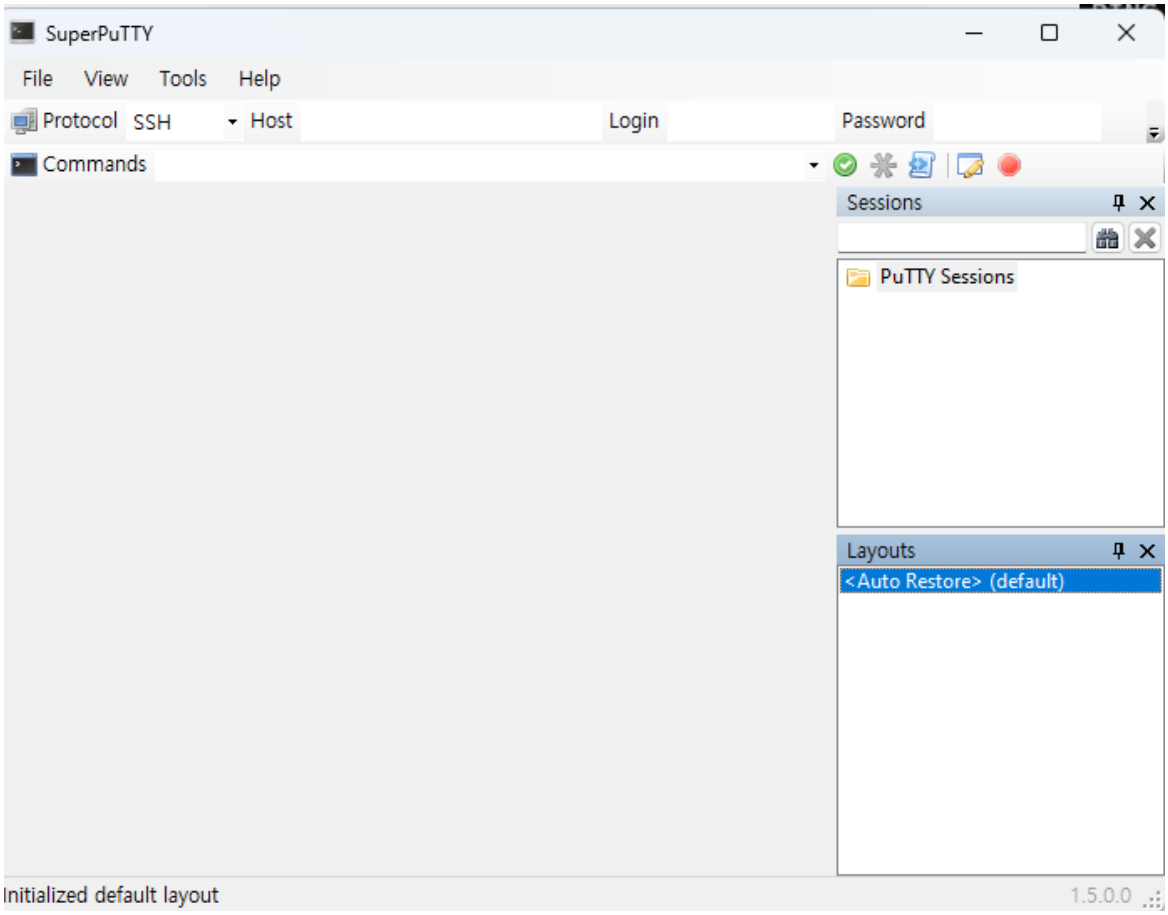
- ping 테스트 파일 업로드 되고 난 후 실행할 수 있도록 권한을 744로
- 가상머신 여러대 있을 때는 `vagrant ssh <가상머신이름>` 으로 접속

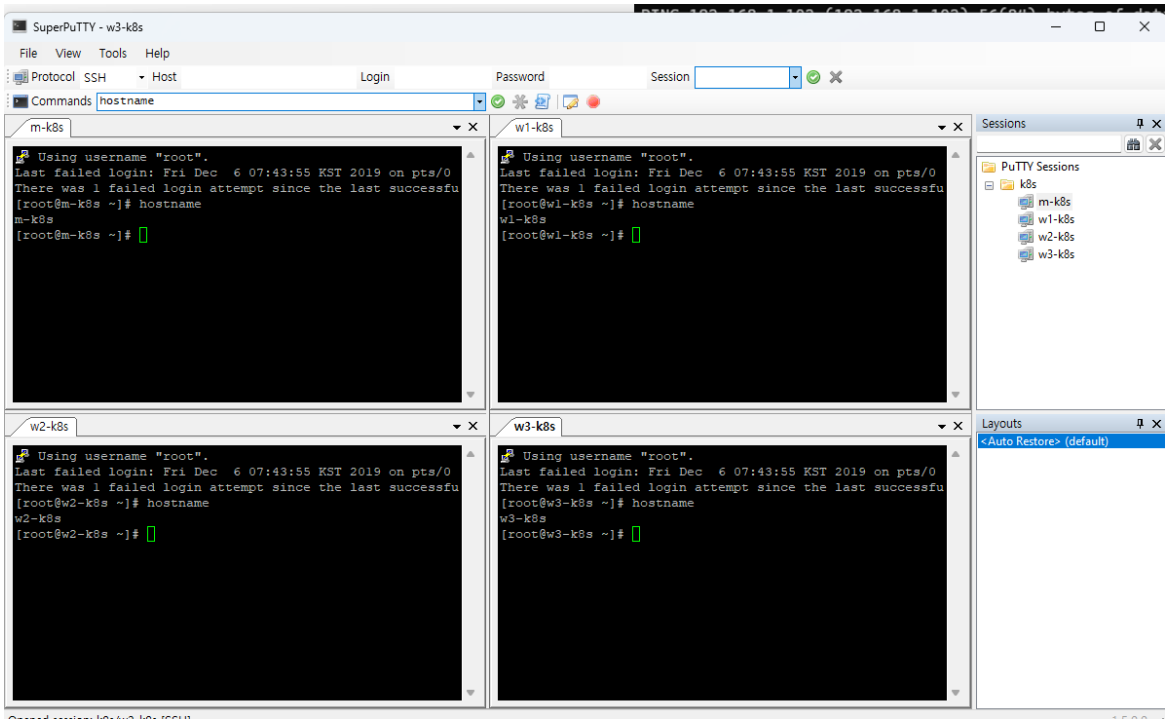
```
C:\HashiCorp>vagrant ssh m-k8s
[vagrant@m-k8s ~]$ ./ping_2_nds.sh
PING 192.168.1.101 (192.168.1.101) 56(84) bytes of data.
64 bytes from 192.168.1.101: icmp_seq=1 ttl=64 time=0.480 ms
64 bytes from 192.168.1.101: icmp_seq=2 ttl=64 time=0.556 ms
64 bytes from 192.168.1.101: icmp_seq=3 ttl=64 time=0.605 ms

--- 192.168.1.101 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2010ms
rtt min/avg/max/mdev = 0.480/0.547/0.605/0.051 ms
PING 192.168.1.102 (192.168.1.102) 56(84) bytes of data.
64 bytes from 192.168.1.102: icmp_seq=1 ttl=64 time=0.261 ms
64 bytes from 192.168.1.102: icmp_seq=2 ttl=64 time=0.611 ms
64 bytes from 192.168.1.102: icmp_seq=3 ttl=64 time=0.457 ms

--- 192.168.1.102 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.261/0.443/0.611/0.143 ms
PING 192.168.1.103 (192.168.1.103) 56(84) bytes of data.
64 bytes from 192.168.1.103: icmp_seq=1 ttl=64 time=0.375 ms
64 bytes from 192.168.1.103: icmp_seq=2 ttl=64 time=0.565 ms
64 bytes from 192.168.1.103: icmp_seq=3 ttl=64 time=0.570 ms

--- 192.168.1.103 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 0.375/0.503/0.570/0.092 ms
```

핑테스트

2.3 터미널 프로그램(putty)로 가상머신 접속하기

- vagrant ssh <가상머신이름> 은 여러 개의 가상머신에 접근 할 때는 유용한 방법x. 푸티로 여러대 한번에 접속할 수 있음.





- command에 명령어 입력하니 4개의 가상머신 동시에 실행