1장, 2장 : 희원

▼ 🎀1. 새로운 인프라 환경이 온다

▼ ✨1.1 컨테이너 인프라 환경이란

- 컨테이너 : 하나의 운영체제 커널에서 다른 프로세스에 영향 받지 않고 독립적으로 실행되는 프로세스 상태

1.1.1 모놀리식 아키텍처

- 하나의 큰 목적이 있는 서비스 또는 애플리케이션에 여러 기능이 통합돼 있는 구조
- 초기단계에서 설계 용이하지만 수정이 많아질 경우, 다른 서비스에 영향 미칠 가능성이 커지고, 기능이 많아질수록 서비스 관계가 복잡해질 수 있음
  - 웹툰 서비스 사용량 증가로 서버 증설할 경우, 뉴스와 블로그 등 다른 서비스도 확장하는 것은 비효율적
  - 특정 서비스에서 에러 발생하면 다른 서비스도 이용할 수 없는 상황 생길 수 있음
  - 이런 문제 해결을 위해 마이크로서비스 아키텍처 등장

1.1.2 마이크로서비스 아키텍처

- 개별 기능을 하는 작은 서비스를 각각 개발해 연결함
- 개발된 사이즈를 재사용하기 쉽고, 향후 서비스가 변경됐을 때 다른 서비스에 영향 미칠 가능성이 줄어들며 특정 서비스만 확장 가능
- 사용자의 요구 사항에 따라 가용성을 즉각적으로 확보해야 하는 IaaS 환경에 적합

1.1.3 컨테이너 인프라 환경에 적합한 아키텍처

- 컨테이너 인프라 환경은 마이크로서비스 아키텍처로 구현하기에 적합

▼ ✨1.2 컨테이너 인프라 환경을 지원하는 도구

1.2.1 도커

1.2.2 쿠버네티스

- 다수의 컨테이너 관리에 사용
- 컨테이너 인프라에 필요한 기능을 통합하고 관리하는 솔루션으로 발전

▼ 🎀2. 테스트 환경 구성하기

▼ ✨2.1 테스트 환경을 자동으로 구성하는 도구

2.1.1 버추얼박스 설치

2.1.2 베이그런트 설치

- 가상 시스템 환경을 관리하기 위한 도구
- 프로비저닝 : 사용자 요구에 맞게 시스템 자원을 할당, 배치, 배포해 두었다가 필요할 때 시스템에 사용할 수 있는 상태로 만들어줌

▼ 2.1.3 베이그런트 구성하고 테스트하기

- 가상 머신 이미지 내려받고 가상 머신 생성하기
  1. vagrant init
  2. Vagrantfile 확인 > config.vm.box = "sysnet4admin/CentOS-k8s"로 수정 (https://app.vagrantup.com/sysnet4admin/boxes/CentOS-k8s)

3. vagrant up
4. vm 실행해 가상머신 실행 확인
5. vagrant ssh : 설치된 CentOS에 접속
6. uptime(실행시간), cat /etc/redhat-release(운영체제의 종류 확인)
7. vagrant destroy -f : 강제 종료

> 💡 **자주 사용하는 베이그런트 명령**
>
> - vagrant init : 프로비저닝 위한 기초 파일 생성
> - vagrant up : Vagrantfile을 읽어 프로비저닝 진행
> - vagrant halt : 가상 머신 종료
> - vagrant destroy : 가상 머신 삭제
> - vagrant ssh : ssh로 접속
> - vagrant provision : 가상 머신에 변경된 설정 적용

▼ ✨2.2 베이그런트로 테스트 환경 구축하기

▼ 2.2.1 가상 머신에 필요한 설정 자동으로 구성하기

- 베이그런트는 루비라는 언어로 작성

1. Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config| #API버전 : 2
  config.vm.define 'm-k8s' do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    #프로바이더가 VM이라고 정의
    cfg.vm.provider "virtualbox" do |vb|
      #가상 머신 이름 , cpu 수, 메모리 크기, 소속된 그룹 명시
      vb.name = "m-k8s(github_SysNet4Admin)"
      vb.cpus = 2
      vb.memory = 2048
      vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
    end
    #호스트의 이름 설정
    cfg.vm.host_name = 'm-k8s'
    #호스트 전용 네트워크를 private_network로 설정해 eth1 인터페이스를 호스트 전용으로 구성하고 ip를 지정
    cfg.vm.network 'private_network', ip: '192.168.1.10'
    #호스트 60010번을 게스트 22번으로 전달되도록 구성
    #auto_correct: true로 설정해서 포트 중복시 자동으로 포트 변경
    #id: 'ssh'로 설정하지 않으면 localhost의 내용과 모든 IP의 60010포트로 오는 내용을 게스트의 22번으로 포워딩하게 됨. 기능 문제는 없으나 명시
    #id: 'ssh'로 설정하면 IP의 60010포트로 오는 내용만 게스트 22번으로 포워딩 함
    cfg.vm.network 'forwarded_port', guest: 22, host: 60010, auto_correct: true, id: 'ssh'
    #호스트와 게스트 사이에 디렉토리 동기화가 이뤄지지 않게 설정 (disabled: True)
    cfg.vm.synced_folder '../data', '/vagrant', disabled: true
  end
end
```

2. 가상머신 생성 : vagrant up
3. 가상머신 접속 : vagrant ssh
4. IP '192.168.1.10'가 제대로 설정됐는지 확인 : ip addr show eth1
5. exit

▼ 2.2.2 가상 머신에 추가 패키지 설치하기

1. Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrant.configure("2") do |config| #API버전 : 2
  config.vm.define 'm-k8s' do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    #프로바이더가 VM이라고 정의
    cfg.vm.provider "virtualbox" do |vb|
      #가상 머신 이름 , cpu 수, 메모리 크기, 소속된 그룹 명시
      vb.name = "m-k8s(github_SysNet4Admin)"
      vb.cpus = 2
      vb.memory = 2048
      vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
    end
    #호스트의 이름 설정
    cfg.vm.host_name = 'm-k8s'
    #호스트 전용 네트워크를 private_network로 설정해 eth1 인터페이스를 호스트 전용으로 구성하고 ip를 지정
    cfg.vm.network 'private_network', ip: '192.168.1.10'
    #호스트 60010번을 게스트 22번으로 전달되도록 구성
    #auto_correct: true로 설정해서 포트 중복시 자동으로 포트 변경
    #id: 'ssh'로 설정하지 않으면 localhost의 내용과 모든 IP의 60010포트로 오는 내용을 게스트의 22번으로 포워딩하게 됨. 기능 문제는 없으나 명시
    #id: 'ssh'로 설정하면 IP의 60010포트로 오는 내용만 게스트 22번으로 포워딩 함
    cfg.vm.network 'forwarded_port', guest: 22, host: 60010, auto_correct: true, id: 'ssh'
    #호스트와 게스트 사이에 디렉토리 동기화가 이뤄지지 않게 설정 (disabled: True)
    cfg.vm.synced_folder '../data', '/vagrant', disabled: true

    #추가 패키지 설치
    #shell 구문으로 경로에 있는 sh파일을 내부에서 호출해 설치되도록 함 (epel, vim 추가 기능)
    cfg.vm.provision "shell", path: "install_pkg.sh" #add provisioning script
  end
end
```

2. install_pkg.sh

```
#!/usr/bin/env bash
# install packages
yum install epel-release -y
yum install vim-enhanced -y
```

2. 추가한 프로비전 구문 실행 : vagrant provision
3. CentOS 접속 : vagrant ssh
4. EPEL 저장소 구성 확인 : yum repolist
5. vim 문법 하이라이트 적용됐는지 확인 : vi .bashrc
6. vagrant destroy -f로 가상머신 삭제

▼ 2.2.3 가상 머신 추가로 구성하기

- 가상 머신 3대를 추가로 설치하고 기존의 가상 머신과 추가한 가상 머신 간에 네트워크 통신이 원활하게 작동하는지 확인

1. Vagrantfile

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

Vagrant.configure("2") do |config|
  config.vm.define "m-k8s" do |cfg|
    cfg.vm.box = "sysnet4admin/CentOS-k8s"
    cfg.vm.provider "virtualbox" do |vb|
      vb.name = "m-k8s(github_SysNet4Admin)"
      vb.cpus = 2
      vb.memory = 2048
      vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
    end
    cfg.vm.host_name = "m-k8s"
    cfg.vm.network "private_network", ip: "192.168.1.10"
    cfg.vm.network "forwarded_port", guest: 22, host: 60010, auto_correct: true, id: "ssh"
    cfg.vm.synced_folder "../data", "/vagrant", disabled: true
    cfg.vm.provision "shell", path: "install_pkg.sh"
    cfg.vm.provision "file", source: "ping_2_nds.sh", destination: "ping_2_nds.sh"
    cfg.vm.provision "shell", path: "config.sh"
  end

  #=============#
  # Added Nodes #
```

```
  #=============#

  # 추가한 3대의 CentOs에 대한 구성 : 반복문 사용
  (1..3).each do |i|
    config.vm.define "w#{i}-k8s" do |cfg|
      cfg.vm.box = "sysnet4admin/CentOS-k8s"
      cfg.vm.provider "virtualbox" do |vb|
        vb.name = "w#{i}-k8s(github_SysNet4Admin)"
        vb.cpus = 1
        vb.memory = 1024
        vb.customize ["modifyvm", :id, "--groups", "/k8s-SM(github_SysNet4Admin)"]
      end
      cfg.vm.host_name = "w#{i}-k8s"
      cfg.vm.network "private_network", ip: "192.168.1.10#{i}"
      cfg.vm.network "forwarded_port", guest: 22, host: "6010#{i}",auto_correct: true, id: "ssh"
      cfg.vm.synced_folder "../data", "/vagrant", disabled: true
      cfg.vm.provision "shell", path: "install_pkg.sh"
    end
  end
end
```

2. install_pkg.sh

```
#!/usr/bin/env bash
# install packages
yum install epel-release -y
yum install vim-enhanced -y
```

3. ping_2_nds.sh

```
# ping 3 times per nodes
# 추가 설치한 CentOs 3대로 ping 보내 네트워크가 제대로 작동하는지 확인함
# -c 옵션 : 몇 번의 ping보낼 것인지 지정
ping 192.168.1.101 -c 3
ping 192.168.1.102 -c 3
ping 192.168.1.103 -c 3
```

4. config.sh

```
#!/usr/bin/env bash# modify permission
# ping이 업로드되고 난 후에 실행할 수 있도록 권한을 744로 줌
chmod 744 ./ping_2_nds.sh
```

- 파일 권한 744의미

구분	소유자	그룹 멤버	그 외
읽기	예	예	예
쓰기	예	아니오	아니오
실행	예	아니오	아니오

5. vagrant up
6. 가상머신 1번 ping test : vagrant ssh m-k8s > ./ping_2_nds.sh > exit

▼ ✨2.3 터미널 프로그램으로 가상 머신 접속하기

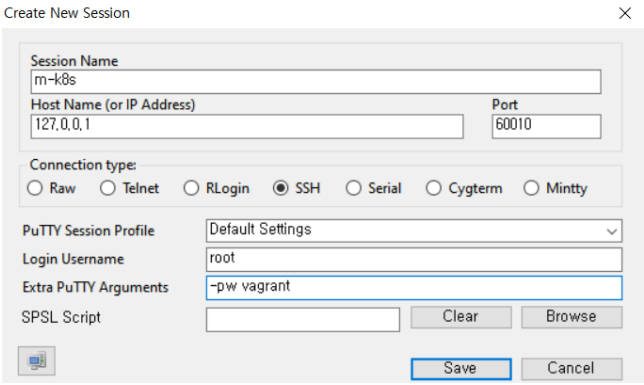
2.3.1 푸티 설치하기

- https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

2.3.2 슈퍼푸티 설치하기

2.3.2 슈퍼푸티로 다수의 가상 머신 접속하기

1. Putty Sessions >New Folder > k8s 입력 후 OK
2. k8s 디렉터리 > New에 다음과 같이 입력 후 save



3. Copy As로 가상 머신 세 개 더 생성
4. Tool > Options > Allow Plain text passwords on putty command line
5. Connect All 로 모든 가상 머신 한 번에 접속
6. 명령 실행되는지 확인 : commands 창에 hostname 입력