

CLOUD WITH DL

PART I

ABOUT CLOUD

ABOUT CLOUD

◆ 컨테이너(Container)

- 화물운송에 주로 사용하는, 쇠로 만든 큰 상자, 협동일관운송에 사용될 목적으로 설계해 제조되는 표준화된 선적용기
- 선박·철도·트럭을 막론하고 옮겨 실을 때마다 내용물을 꺼냈다가 다시 포장할 필요 없이 컨테이너째로 싣고 내릴 수 있다는 의미



IT SW에 적용



ABOUT CLOUD

◆ 컨테이너(Container)

● SW 도입 배경

SERVER

서버 상태 관리

쉽게 관리 위해 문서화

▼
서버 관리 도구

▼
복잡한 관리 설정, 어려움

VM

가상머신 1개 - SW 1개



느림, 클라우드 연동 어려움



특정 제조사에 의존적

DOCKER

컨테이너 기반



어디서나 빠르게



어디서나 연동

ABOUT CLOUD

◆ 컨테이너(Container)

- 특징

- 가상머신과 비교해서 컨테이너 생성 쉽고 효율적
- 컨테이터 이미지 이용한 배포와 롤백 간단
- 언어나 프레임워크에 상관없이 애플리케이션 동일 방식 관리
- 개발/테스팅/운영 환경은 어디서나(Local PC, Cloud) 동일한 환경 구축 가능
- 특정 클라우드 제조사에 종속적이지 않음

ABOUT CLOUD

◆ 컨테이너(Container)

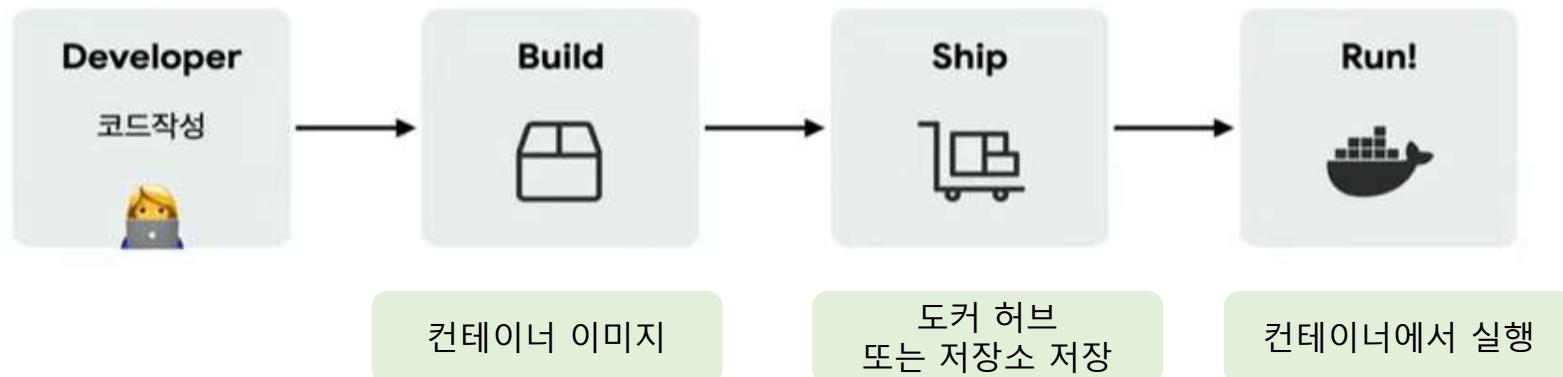
- 컨테이너라이제이션(Containerization) 모든 애플리케이션을 컨테이너로 관리하는 현상



ABOUT CLOUD

◆ 컨테이너(Container)

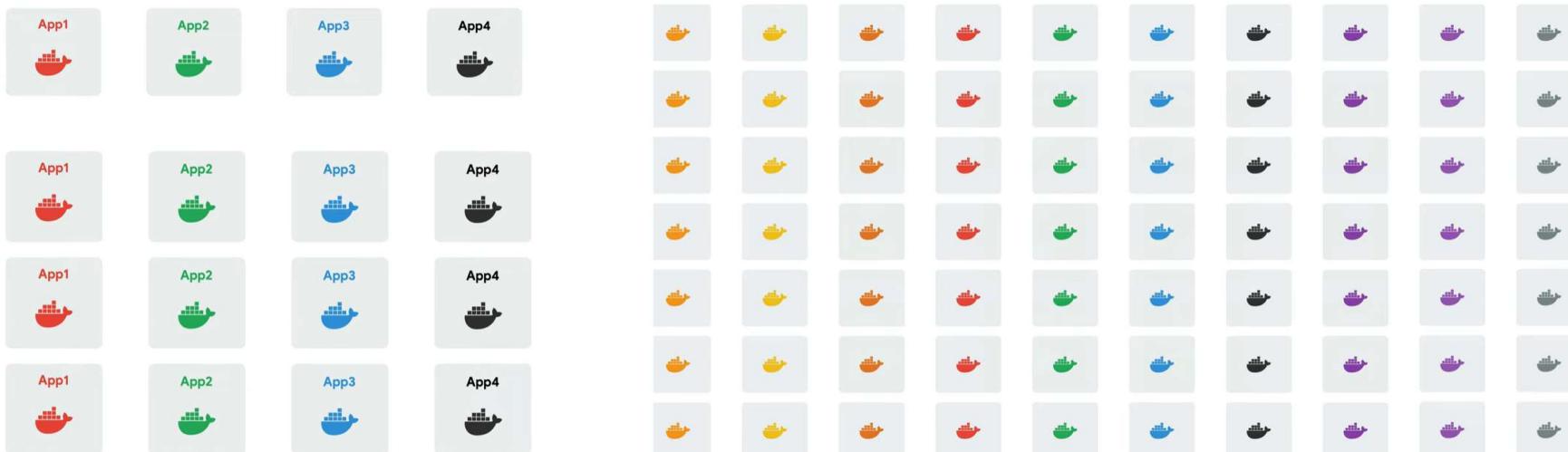
- 컨테이너라이제이션(Containerization) → **프로그램 개발 과정 정형화/표준화**



ABOUT CLOUD

◆ 컨테이너(Container)

- 컨테이너라이제이션(Containerization) → 너무 많은 컨테이너 관리 어려움!

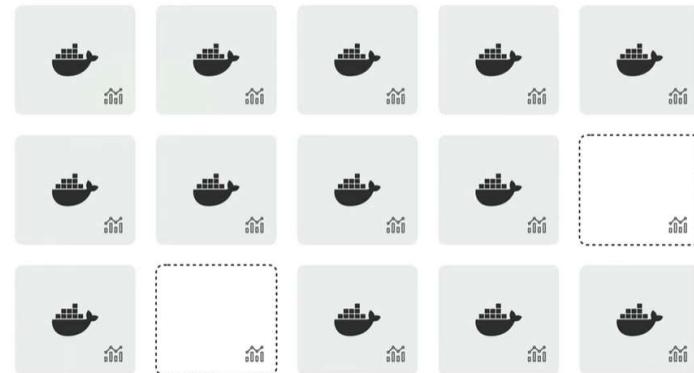


ABOUT CLOUD

◆ 컨테이너(Container)

- 도커 서버

- 많은 도커 서버의 컨테이너 실행 → 하나하나 IP 접속 후 실행 / 제어 / 관리
 - 컨테이너 실행 안된 도커 서버 찾기
 - 버전 업데이트
 - 이전으로 롤백할 컨테이너



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

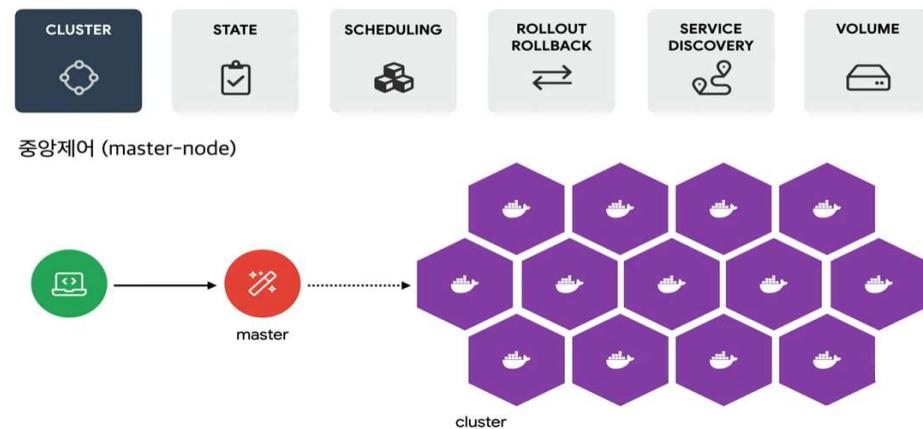
- 복잡한 컨테이너 환경을 효과적으로 관리하기 위한 도구



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 역할 - ① 중앙제어(Master - Node)
 - 관리자는 **Master**에 명령 → **Cluster**내 노드 제어
 - **Cluster**내 노드 통신
 - **부하 고려 설계(Node Scaling)**



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 역할 - ② 상태관리(STATE)

```
{  
  image: "app1"  
  replicas: 2  
}
```



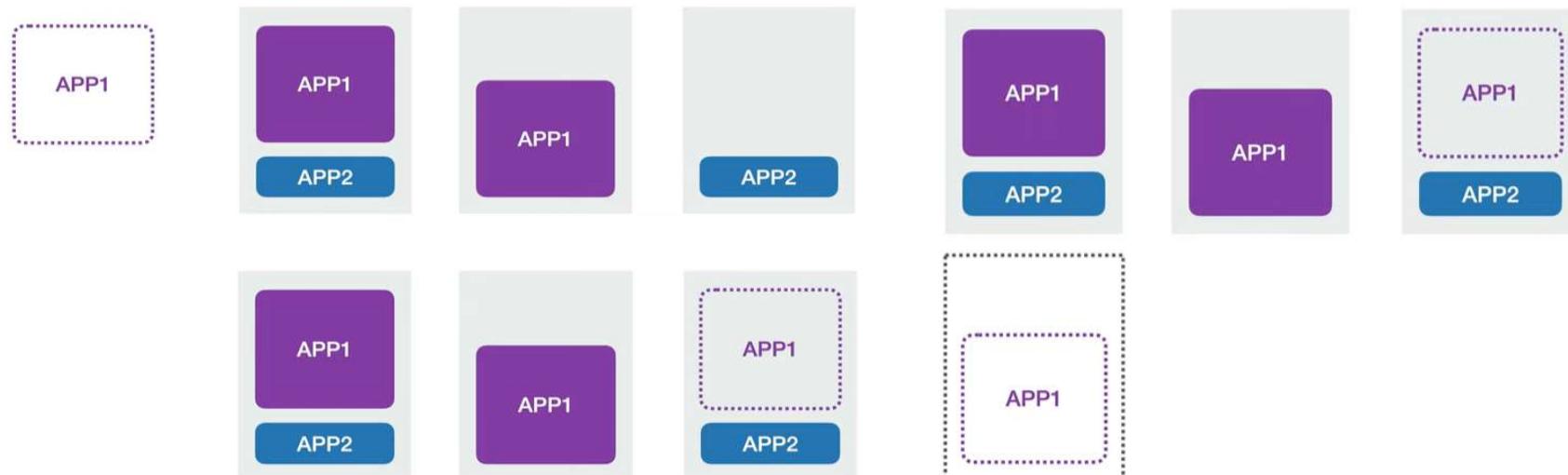
```
{  
  image: "app1"  
  replicas: 3  
}
```



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

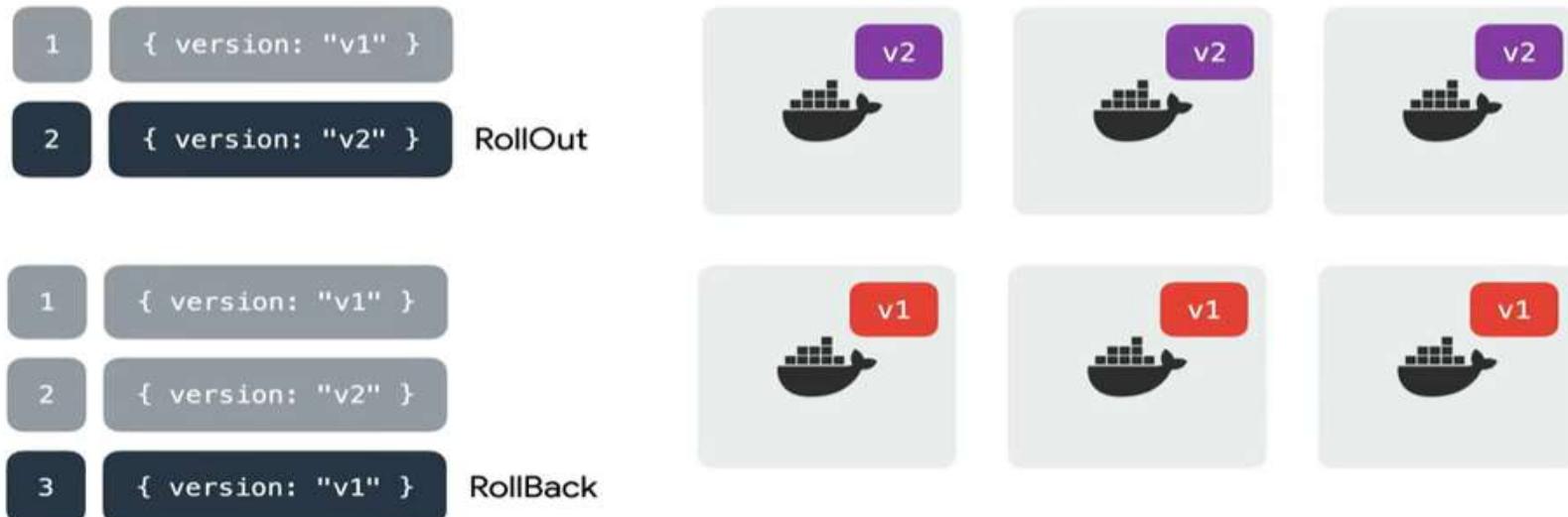
- 역할 - ③ 스케줄링(SCHEDULING)



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

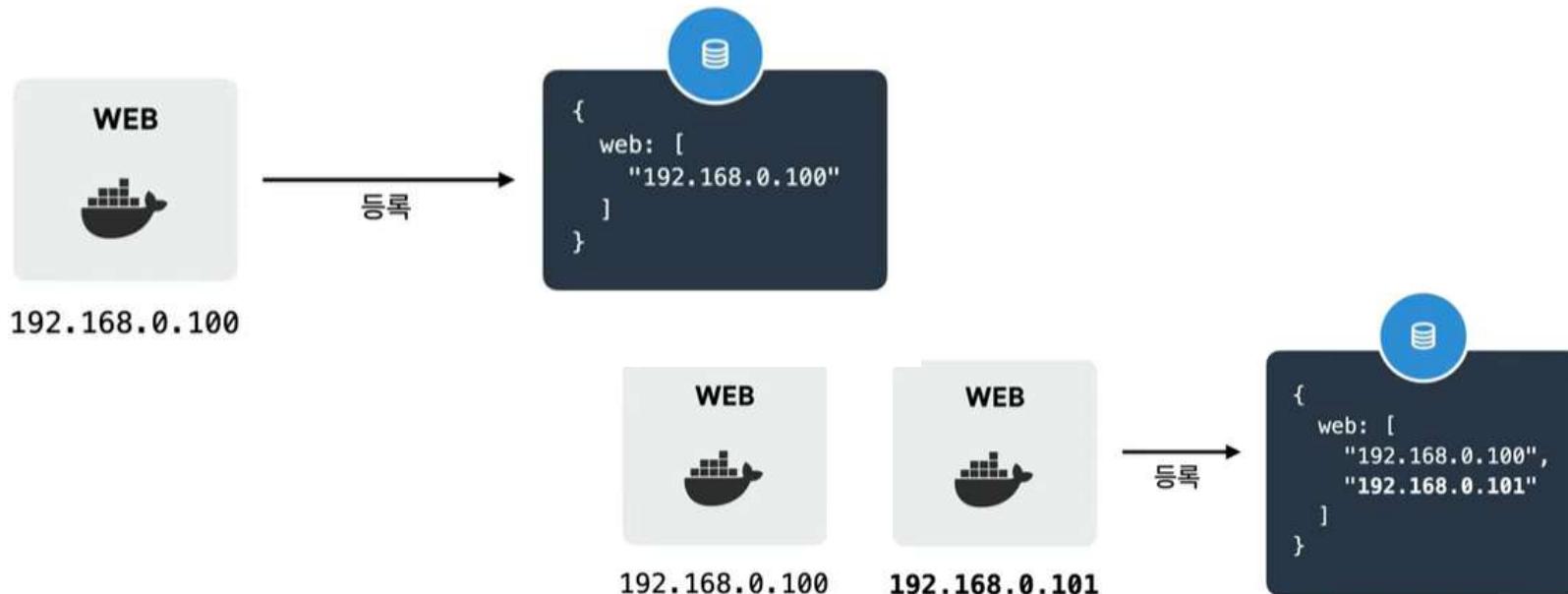
- 역할 - ④ 버전 배포 및 복구(SCHEDULING)



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 역할 - ⑤ 서비스 관리(SERVICE DISCOVERY)



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

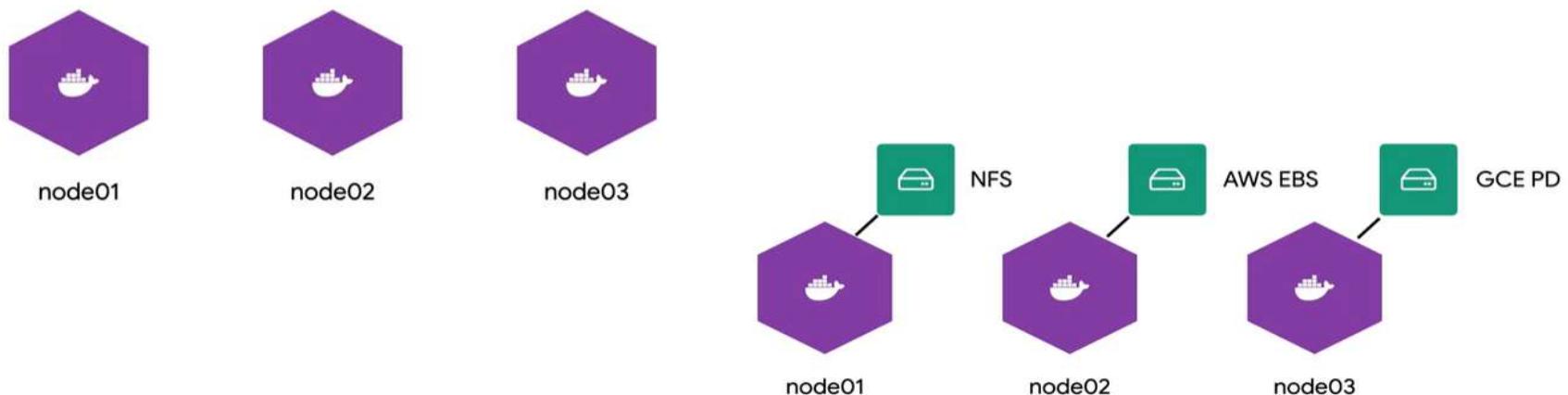
- 역할 - ⑤ 서비스 관리(SERVICE DISCOVERY)



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 역할 - ⑥ 볼륨(VOLUME) & 스토리지 연결/해제 제어
 - 볼륨 : 컨테이너에 제공하는 호스트의 공간



ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 다양한 컨테이너 오케스트레이션 Tools



MESOS



MARATHON

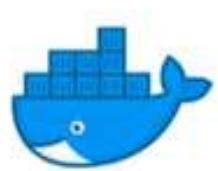


HashiCorp
Nomad

ABOUT CLOUD

◆ 컨테이너 오케스트레이션(Container Orchestration)

- 다양한 컨테이너 오케스트레이션 Tools



PART I

**ABOUT
KUBERNETES**

ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

- 구글에서 2014년 발표 후 Linux 재단에 의해 관리되는 Open Source
- 컨테이너를 쉽고 빠르게 배포/확장하고 관리를 자동화해주는 오픈소스 플랫폼



kubernetes

ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

- 특징 - 무한한 확장성

쿠버네티스 상에 구동되는 플랫폼들



Kubeflow



머신러닝



서비스메시



서버리스

ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

- 특징 - 무한한 확장성

쿠버네티스 네이티브 플랫폼들



Rancher (by SUSE)



Red Hat OpenShift (by IBM)



Tanzu (by VMware)

쿠버네티스

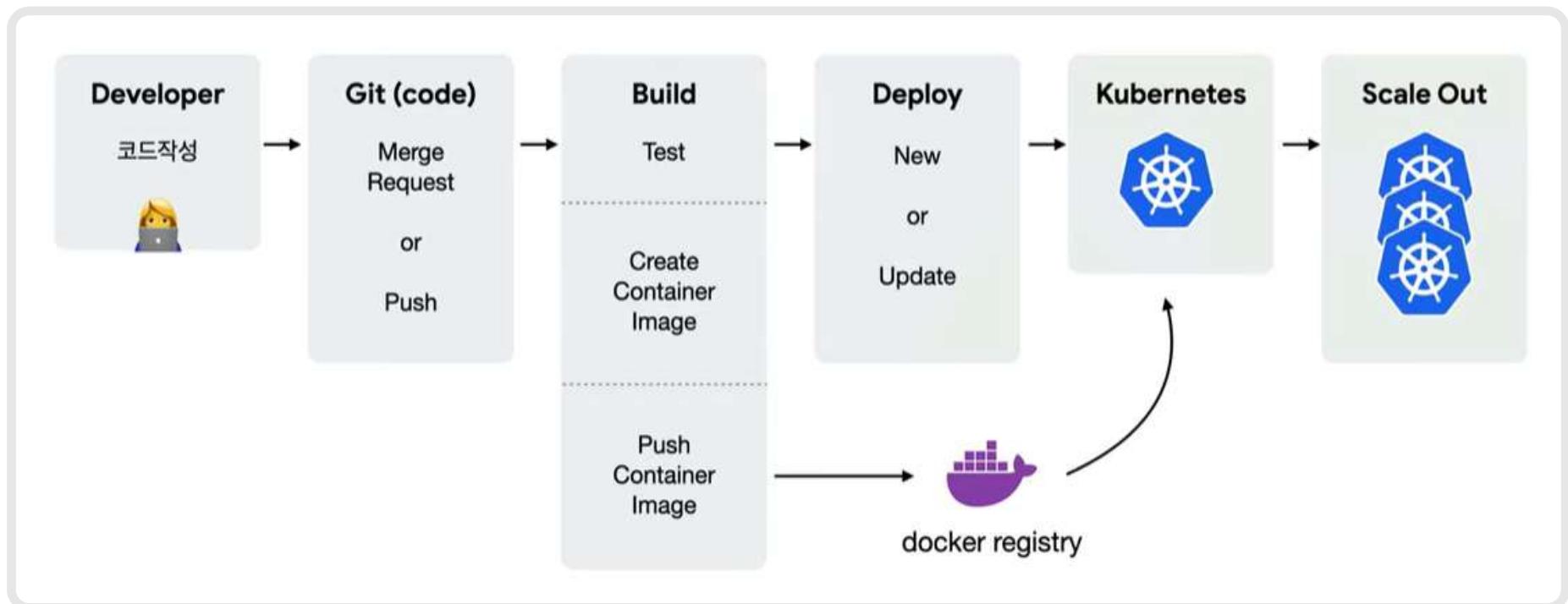
ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

- 클라우드 네이티브(Cloud Native) : 클라우드 환경에 적합한 컴퓨팅 기술
 - 핵심역할 : 쿠버네티스 즉, 사실상 표준(de facto)

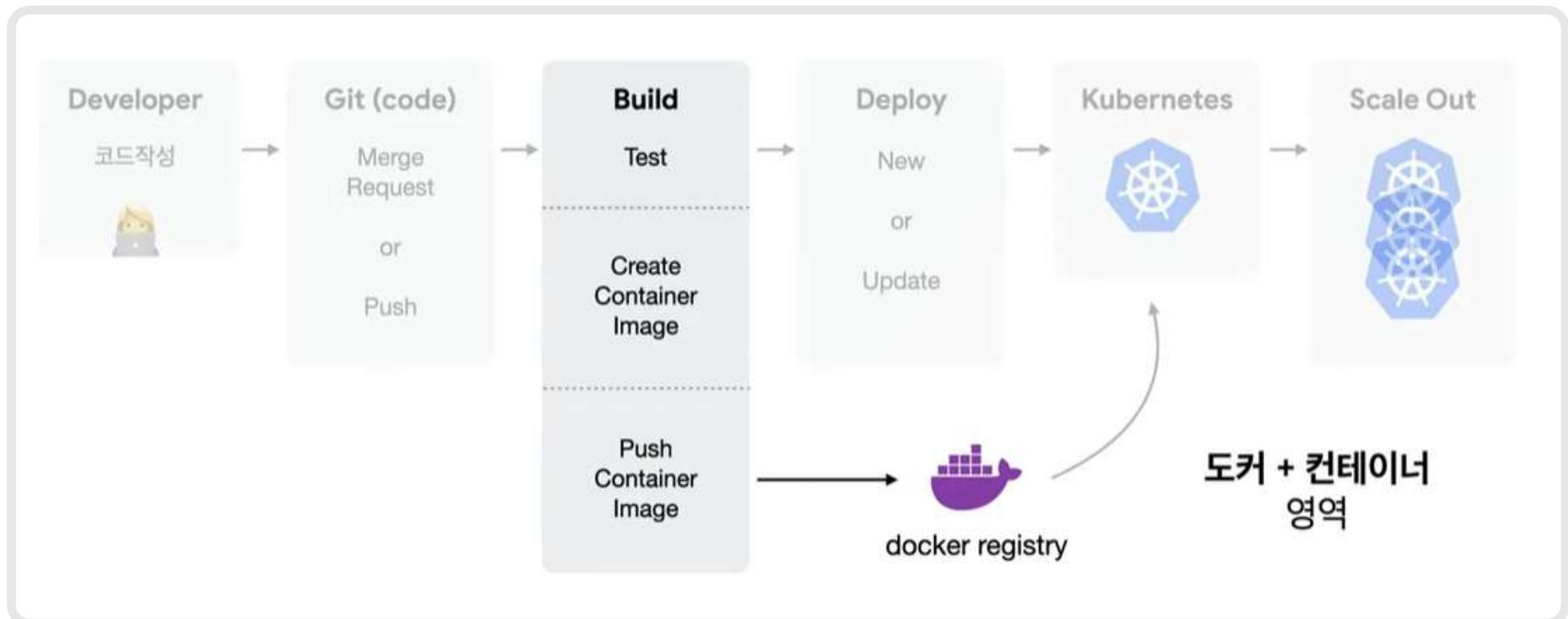
ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구



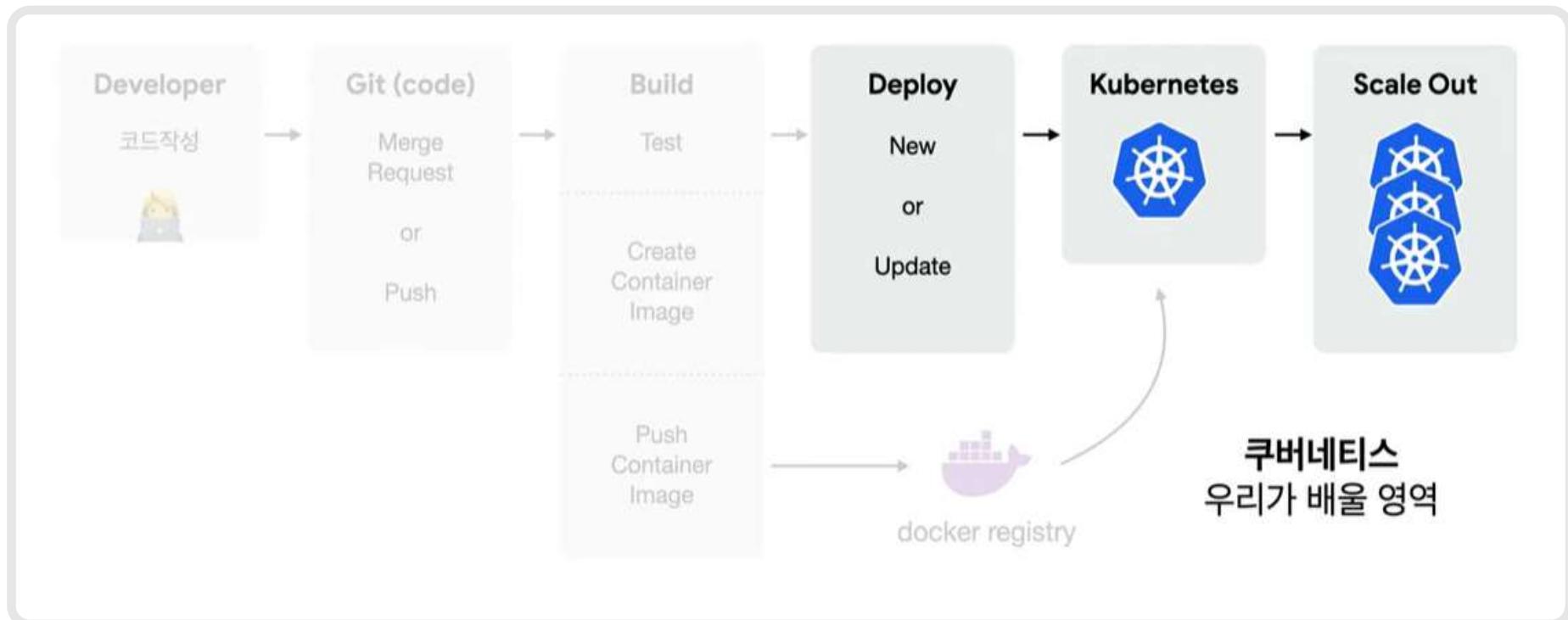
ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구



ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구



ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

도커 컨테이너 실행하기

쿠버네티스에 컨테이너 배포하기

외부 접속 설정 하기

스케일 아웃 하기

그외 고급기능 소개

도커와 도커컴포즈를 이용한
멀티 컨테이너 관리

ABOUT KUBERNETES

◆ 컨테이너 오케스트레이션 도구

도커 컨테이너 실행하기

쿠버네티스에 컨테이너 배포하기

외부 접속 설정 하기

스케일 아웃 하기

그외 고급기능 소개

실습(hands-on) 환경 만들기

kubectl 사용법

pod, deployment, service 등

기본 리소스 학습

ABOUT KUBERNETES

◆ 용어

master	마스터
node	노드 (구 minion 미니언)
k8s	쿠버네티브, 케이에잇츠
kubectl	큐브컨트롤, 큐브씨티엘, 큐브커들
etcd	엣지디, 엣시디, 이티시디
pod	팟, 파드, 포드
istio	이스티오
helm	헬름, 햄
knative	케이 네이티브

ABOUT KUBERNETES

◆ 용어

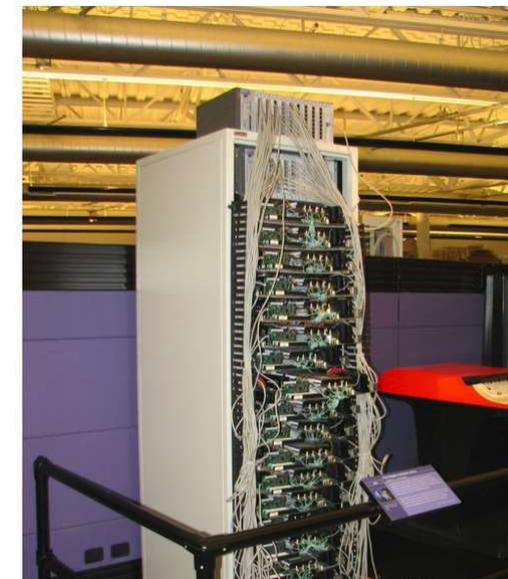
- 클라우드 네이티브 (Cloud Native)

클라우드 이전

리소스를 한 땀 한 땀 직접 관리

각 서버 역할 명시

SERVER	SERVER	SERVER
WEB-PROD	API-PROD	DB-PROD
192.168.0.101	192.168.0.102	192.168.0.103
web	api	db



ABOUT KUBERNETES

◆ 용어

- 클라우드 네이티브 (Cloud Native)



ABOUT KUBERNETES

◆ 용어

- 클라우드 네이티브 (Cloud Native)



클라우드 환경에서 어떻게 애플리케이션을 배포하는게 좋은걸까?

CLOUD NATIVE 하다

컨테이너!

서비스메시!

마이크로
서비스!

API!

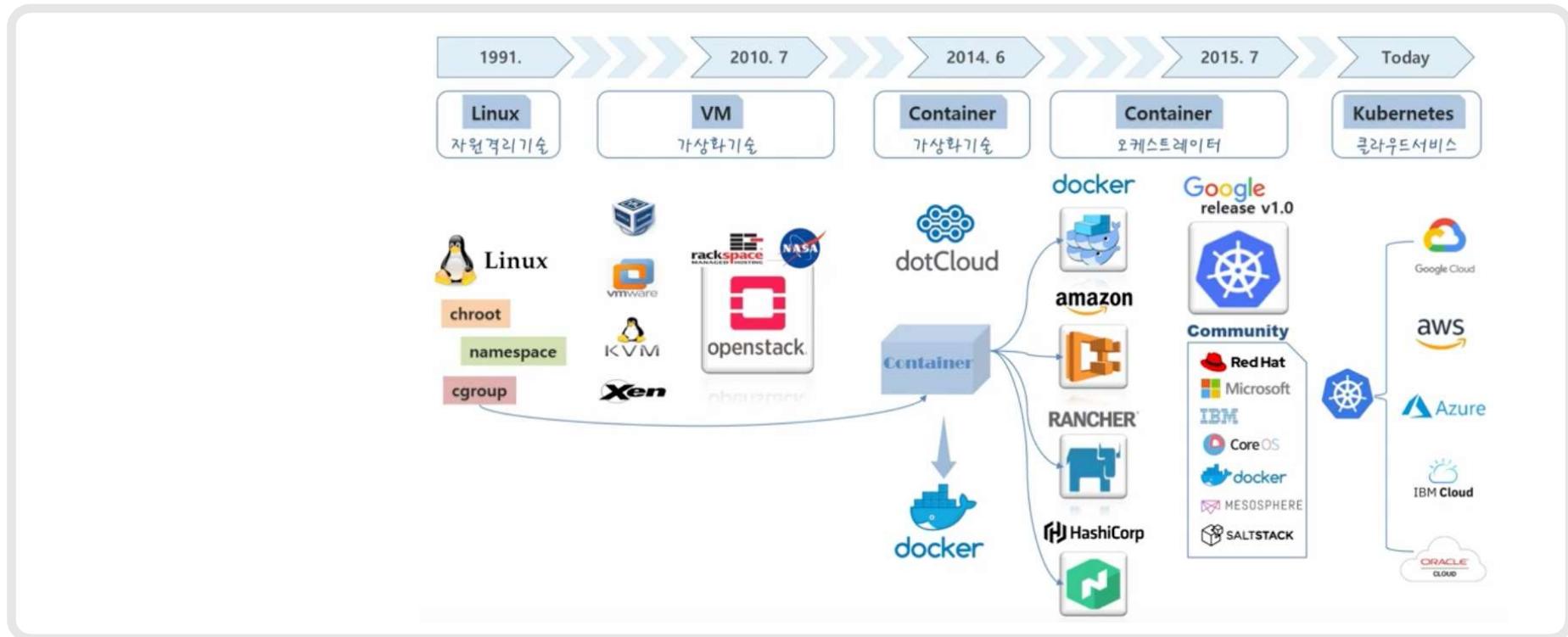
인프라
쓰고 버려!

DevOps!



ABOUT KUBERNETES

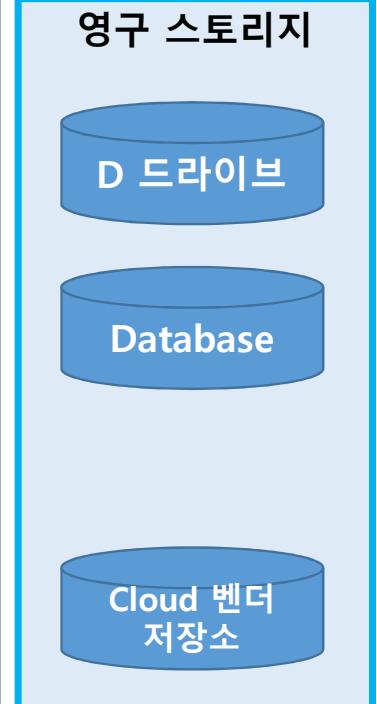
◆ 발전 역사



ABOUT KUBERNETES

◆ 쿠버네티스 구조

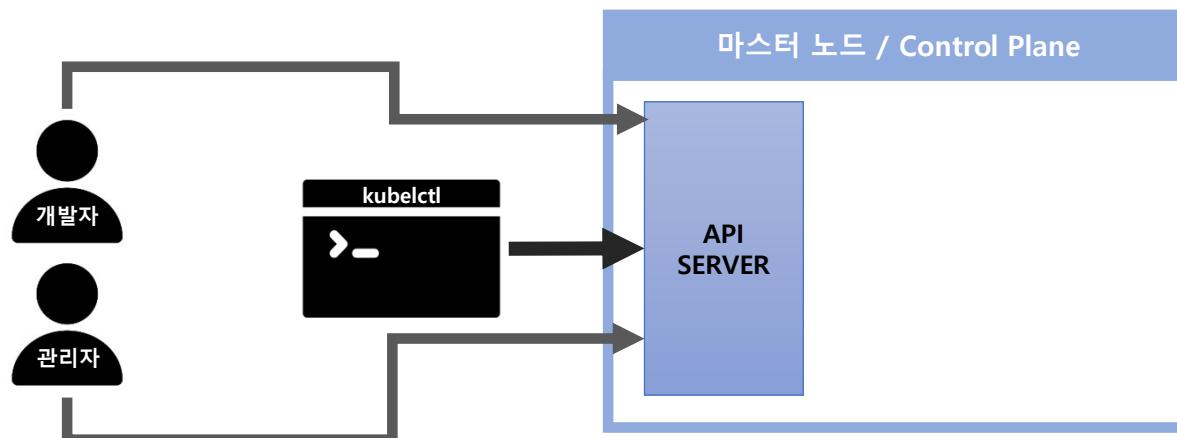
쿠버네티스 클러스터



ABOUT KUBERNETES

◆ 구성요소 - Master Node : API Server

- 쿠버네티스 클러스터의 중심 역할을 하는 통로
- 주로 상태 값을 저장하는 etcd와 통신
- 다른 요소들 또한 API 서버를 중심에 두고 통신하기 때문에 API 서버의 역할이 매우 중요
- 개발자가 kubectl 커맨드라인 이용해서 API 서버에 명령 전송 >> API서버는 적절한 위치로 명령 전달



ABOUT KUBERNETES

◆ 구성요소 - Master Node : ETCD

- 리눅스 구성 정보 가지고 있는 etc 디렉터리 + distributed 합성어
- 쿠버네티스 클러스터의 상태 저장
- 클러스터에 필요한 정보, 파드 같은 리소스 상태 정보 저장하는 곳
- 형태 : Key-Value

PART I

DEVELOPMENT ENVIRONMENT

DEVELOPMENT ENVIRONMENT

◆ INSTALL DOCKER

● WSL(Windows Subsystem for Linux) 설치

- 원도우에서 리눅스 환경처럼 Powershell을 Bash 처럼 사용
- Linux 명령어(sed, awk, vim, apt 등)를 사용 가능
- Linux 커널 이용 가능

특색	WSL 1	WSL 2
Windows와 Linux 간의 통합	✓	✓
빠른 부팅 시간	✓	✓
기존 가상 머신에 비해 적은 리소스 공간	✓	✓
현재 버전의 VMware 및 VirtualBox에서 실행	✓	✓
관리 형 VM	✗	✓
전체 Linux 커널	✗	✓
전체 시스템 호출 호환성	✗	✓
OS 파일 시스템에서의 성능	✓	✗

DEVELOPMENT ENVIRONMENT

◆ INSTALL DOCKER

- WSL(Windows Subsystem for Linux) 설치

- Windows PowerShell을 관리자 권한으로 실행
- 설치 명령어 : `$ wsl --install`
- 버전 2 설정 : `$ wsl --set-default-version 2`
- 커널 업데이트 : `$ wsl --update`

DEVELOPMENT ENVIRONMENT

◆ INSTALL DOCKER

● WSL(Windows Subsystem for Linux) 설치

- 리눅스 확인 : \$ wsl -l -v

```
NAME      STATE      VERSION
* Ubuntu   Running    2
```

- 리눅스 종료 : \$ wsl -t Ubuntu

- 리눅스 확인 : \$ wsl -l -v

```
NAME      STATE      VERSION
* Ubuntu   Stopped    2
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL DOCKER

- DOCKER 설치

- Window용 설치 파일 다운로드

<https://www.docker.com/products/docker-desktop/>

Docker Desktop Installer.exe



DEVELOPMENT ENVIRONMENT

◆ INSTALL DOCKER

● DOCKER 설치 확인

- Windows PowerShell을 관리자 권한으로 실행
- 명령어 : \$ wsl -l -v

NAME	STATE	VERSION
* docker-desktop	Running	2
docker-desktop-data	Running	2

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- MINIKUBE 사이트 이동 : <https://minikube.sigs.k8s.io/docs/start/>

Click on the buttons that describe your target platform. For other architectures, see the [release page](#) for a complete list of minikube binaries.

Operating system [Linux](#) [macOS](#) [Windows](#)

Architecture [x86-64](#)

Release type [Stable](#) [Beta](#)

Installer type [.exe download](#) [Windows Package Manager](#) [Chocolatey](#)

To install the latest minikube **stable** release on **x86-64 Windows** using **Windows Package Manager**:

If the [Windows Package Manager](#) is installed, use the following command to install minikube:

```
winget install minikube
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- MINIKUBE 설치 : <https://minikube.sigs.k8s.io/docs/drivers/>

- Windows PowerShell을 관리자 권한으로 실행
- 설치 명령어 입력 : \$ winget install **minikube**
- 확인 명령어 입력 : \$ **minikube version**

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- kubectl 설치

- Kubernetes 클러스터에 명령을 내리는 CLI(Command Line Interface)
- 기본 구조 : kubectl [command] [Type] [NAME] [flags]
 - command - 자원 실행 명령 (create, get, delete, edit)
 - TYPE : 자원 타입 (node, pod, service, ...)
 - NAME : 자원 이름
 - flags : 부가적 설정 옵션

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- kubectl 설치

- Windows PowerShell을 관리자 권한으로 실행
 - 설치 명령어 입력 : \$ winget install -e --id **Kubernetes.kubectl**
 - 확인 명령어 입력 : \$ kubectl version --client
- <https://kubernetes.io/docs/tasks/tools/install-kubectl-windows/>

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● kubectl 대표 명령어

apply	상태 적용 명령어	kubectl apply -f 파일명 또는 URL
get	리소스 목록 출력	kubectl get [TYPE]
describe	리소스 상태 자세히 출력	kubectl describe [TYPE]/[NAME]
delete	리소스 제거	kubectl delete [TYPE]/[NAME]
logs	컨테이너 로그 확인	kubectl logs [POD_NAME]
exec	컨테이너에 명령어 전달	kubectl exec [-it] [POD_NAME]
config	kubectl 설정 관리	kubectl config current-context kubectl config use-context minikube

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- **kubectl** - 쿠버네티스 클러스터 상태 정보 확인

```
$ kubectl cluster-info
```

```
so@k8s: ~/kube
so@k8s: $ kubectl cluster-info
Kubernetes control plane is running at https://127.0.0.1:51706
CoreDNS is running at https://127.0.0.1:51706/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

kubernetes control plane과 kubeDNS 실행 상태
정상적인 클러스터 구성

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- **kubectl** - 쿠버네티스 클러스터 상세 정보 확인

```
$ kubectl config view
```

```
so@k8s:~/kube$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://kubernetes.docker.internal:6443
    name: docker-desktop
- cluster:
    certificate-authority: /home/so/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Tue, 16 Apr 2024 22:26:15 KST
        provider: minikube.sigs.k8s.io
        version: v1.32.0
        name: cluster_info
    server: https://127.0.0.1:51706
    name: minikube
contexts:
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- **kubectl** - 쿠버네티스 클러스터의 노드 정보 확인

```
$ kubectl get nodes
```

```
so@k8s: ~  
so@k8s:~$ kubectl get nodes  
NAME     STATUS   ROLES      AGE    VERSION  
minikube Ready    control-plane   35m    v1.28.3  
so@k8s:~$
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● kubectl 설치

- kube 폴더 생성

- ① 사용자 Home 폴더 이동 : \$ cd ~
- ② kube 폴더 생성 : \$ mkdir .kube ← 숨김 폴더로 생성
- ③ kube 폴더 이동 : \$ cd .kube
- ④ Kube 클러스터 사용 위한 설정 : \$ New-Item config -type file

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 시작

- Windows PowerShell을 관리자 권한으로 실행
- 실행 명령어 입력 : \$ **minikube start --driver=docker** # --driver=가상화기술선택 옵션

```
PS C:\Users\wanecce> minikube start
[0331 20:02:20.733864  34932 main.go:291] Unable to resolve the current Docker CLI context
It": context not found: open C:\Users\wanecce\.docker\contexts\meta\37a8eec1ce19687d132fe29051
4133a33f0688f\meta.json: The system cannot find the path specified.
* Microsoft Windows 10 Home 10.0.19045.4170 Build 19045.4170 의 minikube v1.32.0
* 기존 프로필에 기반하여 docker 드라이버를 사용하는 중
* minikube 클러스터의 minikube 컨트롤 플레인 노드를 시작하는 중
* 베이스 이미지를 다운받는 중 ...
* Restarting existing docker container for "minikube"
* 쿠버네티스 v1.28.3 을 Docker 24.0.7 런타임으로 설치하는 중/
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 시작

- Windows PowerShell을 관리자 권한으로 실행
- 실행 명령어 입력 : \$ **minikube start --driver=docker** # --driver=가상화기술선택 옵션

```
PS C:\Users\wanecce> minikube start
[0331 20:02:20.733864  34932 main.go:291] Unable to resolve the current Docker CLI context
It": context not found: open C:\Users\wanecce\.docker\contexts\meta\37a8eec1ce19687d132fe29051
4133a33f0688f\meta.json: The system cannot find the path specified.
* Microsoft Windows 10 Home 10.0.19045.4170 Build 19045.4170 의 minikube v1.32.0
* 기존 프로필에 기반하여 docker 드라이버를 사용하는 중
* minikube 클러스터의 minikube 컨트롤 플레인 노드를 시작하는 중
* 베이스 이미지를 다운받는 중 ...
* Restarting existing docker container for "minikube"
* 쿠버네티스 v1.28.3 을 Docker 24.0.7 런타임으로 설치하는 중/
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 시작

```
PS C:\Users\wanecce> minikube start --driver=docker
W0416 21:09:55.328128  82100 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": context not found: open C:\Users\wanecce\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f\meta.json: The system cannot find the path specified.
* Microsoft Windows 10 Home 10.0.19045.4291 Build 19045.4291 의 minikube v1.32.0
* 기존 프로필에 기반하여 docker 드라이버를 사용하는 중

* Exiting due to PROVIDER_DOCKER_NOT_FOUND: The 'docker' provider was not found: exec: "docker": executable file not found in %PATH%
* 권장: Install Docker
* 문서: https://minikube.sigs.k8s.io/docs/drivers/docker/
```

- 실행 명령어 입력 : \$ **kubectl describe namespace default**

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 시작

- docker image를 다운받고 실행하여 kubernetes cluster를 생성
- kubectl과의 연결을 위해 ~/.kube/config 파일 setting까지 완료

```
PS C:\Users\wanece> minikube start
W0331 20:02:20.733864 34932 main.go:291] Unable to resolve the current Docker CLI context "default": context
It": context not found: open C:\Users\wanece\docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958
4133a33f0688f\meta.json: The system cannot find the path specified.
* Microsoft Windows 10 Home 10.0.19045.4170 Build 19045.4170 의 minikube v1.32.0
* 기존 프로필에 기반하여 docker 드라이버를 사용하는 중
* minikube 클러스터의 minikube 컨트롤 플레인 노드를 시작하는 중
* 베이스 이미지를 다운받는 중 ...
* Restarting existing docker container for "minikube"
* 쿠버네티스 v1.28.3 을 Docker 24.0.7 런타임으로 설치하는 중
* Configuring bridge CNI (Container Networking Interface) ...
* Kubernetes 구성 요소를 확인 ...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* 애드온 활성화 : storage-provisioner, default-storageclass
* 끝났습니다! kubectl이 "minikube" 클러스터와 "default" 네임스페이스를 기본적으로 사용하도록 구성되었습니다.
PS C:\Users\wanece>
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 상태확인

- 명령어 입력 : \$ minikube status

```
PS C:\Users\Wanece> minikube status
[0331 20:05:05.753323  34860 main.go:291] Unable to resolve the current Docker CLI context "default"
It": context not found: open C:\Users\Wanece\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d16
4133a33f0688f\meta.json: The system cannot find the path specified.
minikube
type: Control Plane
host: Running
kublet: Running
apiserver: Running
kubeconfig: Configured

PS C:\Users\Wanece>
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster CLI 경고

```
C:\Users\anecce>minikube status
W0404 09:07:38.777069    1276 main.go:291] Unable to resolve the current Docker CLI context "default": context "default"
: context not found: open C:\Users\anecce\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a3
3f0688f\meta.json: The system cannot find the path specified.
minikube
```

- 명령어 입력 : \$ docker context use default
`default`
Current context is now "default"

- 명령어 입력 : \$ minikube status

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● MINIKUBE Cluster 제어 명령어

- 일시정지 명령어 입력 : \$ minikube **pause**
- 재가동 명령어 입력 : \$ minikube **unpause**
- 종료 명령어 입력 : \$ minikube **stop**
- 삭제 명령어 입력 : \$ minikube **delete**

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

● 생성 가능 리소스 목록

- 일시정지 명령어 입력 : \$ kubectl [api-resources](#)

C:\Users\Wanece>kubectl api-resources	NAME	SHORTNAMES	API VERSION	NAMESPACE	KIND
	bindings		v1	true	Binding
	componentstatuses	cs	v1	false	ComponentStatus
	configmaps	cm	v1	true	ConfigMap
	endpoints	ep	v1	true	Endpoints
	events	ev	v1	true	Event
	limitranges	limits	v1	true	LimitRange

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- 스토리지 확인

- 일시정지 명령어 입력 : \$ kubectl get sc

```
C:\Users\lanece>kubectl get sc
NAME      PROVISIONER          RECLAIMPOLICY  VOLUMEBINDINGMODE  ALLOWVOLUMEEXPANSION  AGE
standard  (default)  k8s.io/minikube-hostpath  Delete        Immediate        false            3d15h
```

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- **애드온 확인** : 클러스터 내부에서 필요한 기능들을 위해 실행되는 포드들

- 디플로이먼트, 리플리케이션 컨트롤러등에 의해 관리
- 네임스페이스 : kube-system
- 종 류 : 네트워킹, DNS, 대시보드

- **addon 확인 명령어** : \$ minikube addons list

DEVELOPMENT ENVIRONMENT

◆ INSTALL MINIKUBE

- 애드온 확인 : 클러스터 내부에서 필요한 기능들을 위해 실행되는 포드들

디플로이먼트, 리플리케이션 컨트롤러등에 의해 관리

사용하는 네임스페이스 : kube-system입니다. 애드온에는 몇가지 종류들이 있습니다출처:
<https://arisu1000.tistory.com/27828> [아리수:티스토리]

- User 설정 정보 확인 명령어 : \$ kubectl config get-users

DEVELOPMENT ENVIRONMENT

◆ hello-world 애플리케이션 K8s 클러스터에 배포

- 매니페스트 파일(manifest file)

K8s에서 클러스터에 애플리케이션을 어떻게 디플로이 하고, 클라이언트에게 액세스를 어떻게 처리해야 할지에 대한 내용을 저장하고 있는 파일로 K8s 구동 핵심 파일₩

- 내용 : 자원 할당과 네트워크 설정 정보
- 파일 형식 : YAML 파일

DEVELOPMENT ENVIRONMENT

◆ hello-world 애플리케이션 K8s 클러스터에 배포

- 매니페스트 파일(manifest file)

- 디플로이먼트 (deployments)

- K8s에서 pod 관리와 네트워킹 목적으로 함께 묶여 있는 하나 이상의 컨테이너 그룹
 - pod의 헬스체크와 생성 및 스케일링을 관리 역할 담당

- 서비스 (service)

- K8s의 pod는 컨테이너 내부의 네트워크 주소를 할당받기 때문에 외부에 접속하기 위해 서는 해당 pod의 서비스를 외부 IP로 노출시켜야 함
 - 이러한 정보를 담고 있는 것이 서비스

DEVELOPMENT ENVIRONMENT

◆ hello-world 애플리케이션 K8s 클러스터에 배포

- 매니페스트 파일(manifest file)

- 디플로이먼트 (deployments)

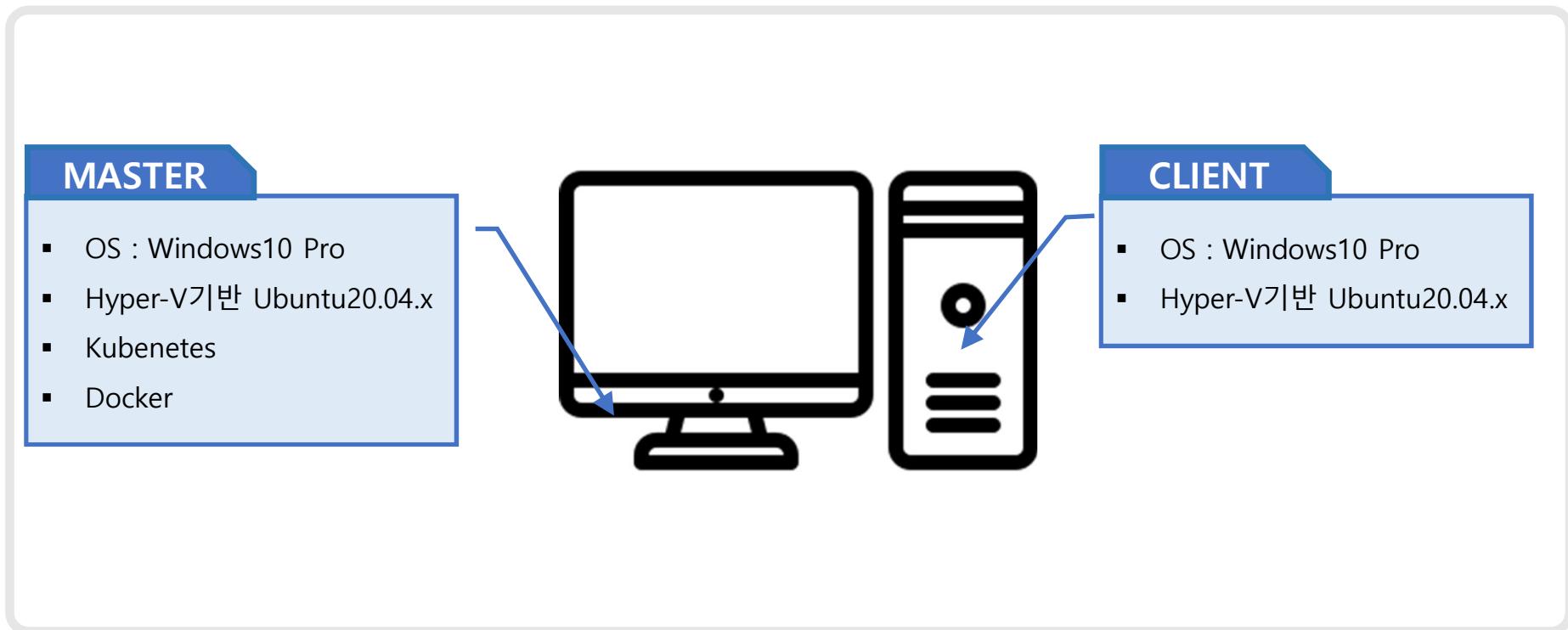
- 명령어 : \$ curl -LO https://github.com/coreos/kube-prometheus/archive/v0.5.0.zip

- 서비스 (service)

WINDOW PRO HYPER-V

DEVELOPMENT ENVIRONMENT

◆ 개발환경 구성도



DEVELOPMENT ENVIRONMENT

◆ KUBENETES 환경구축 - Hyper-V 설정

- Hyper-V 기능 체크

- 제어판 -> 프로그램 -> 프로그램 및 기능 -> Windows 기능 켜기/끄기 -> Hyper-V

PART I

OBSERVABILITY

OBSERVABILITY

◆ 관찰 가능성(Observability)

- 시스템이 점점 분산되면서 구축하고 운영하는 방법론 빠르게 진화
- 다양한 IT 인프라 환경의 공존과 복잡성 증대 → 서비스와 인프라 가시성 중요

2018년 진화

- '외부 출력값만을 사용해 시스템의 내부 상태를 관찰할 수 있다' 기본 개념
- '내부 상태에서 수집한 데이터 사용하여 시스템 동작 이해 및 디버깅 가능'
- 시스템은 내부 동작에 관한 풍부한 맥락 정보 제공, 더 깊은 시스템 문제 밝힐 수 있음

OBSERVABILITY

◆ 관찰 가능성(Observability) VS 모니터링(Monitoring)

❖ 관찰 가능성

- 핵심 요소 : 측정(Metrics), 추적(Traces), 로그(Log), 경고(Alert)
- 로그, 지표 등 조직 전체 데이터 수집

❖ 모니터링

- 모니터링 도구 사용 데이터 분석
 - 시스템의 상태 가용성, 변화 확인
 - 사용 통계 변화 조사 > 배포 전반 성능 변화 검증

OBSERVABILITY

◆ 관찰 가능성(Observability) 항목

- ❖ **지표(Metric)** CPU, 메모리, 네트워크, 디스크 공간 등 시스템 현재 상태 수치화
시스템의 최대 용량과 현재 상태 간 차이 확인 >>> 시스템 사용 여부 보장

- ❖ **로그(Log)** 앱 및 시스템에서 텍스트 로그 수집 방법
컨트롤 플레인 로그, 애플리케이션 피드 로그 조합
쿠버네티스 시스템 가용성 진단 및 애플리케이션 버그 분류

OBSERVABILITY

◆ 관찰 가능성(Observability) 항목

❖ 추적 (Trace) 분산된 추적 수집

マイ크로서비스 사용되는 분산 클라우드 네이티브 시 중요

병목 현상이나 문제 시 추적을 통해 서비스 간 호출의 구간별 분류 확인

❖ 경고 (Alert) 특정 이벤트 발생 시 자동화된 전달 방법 설정

지표, 로그 모두 설정 가능

문자 메시지, 이메일, 타사 애플리케이션 등 다양한 매체 통해 전달

쿠버네티스 미지원으로 OpenSource 도구 사용

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server

- 설치 명령어 입력 : \$ kubectl apply -f <https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml>

```
C:\Users\anece>kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
serviceaccount/metrics-server created
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader created
clusterrole.rbac.authorization.k8s.io/system:metrics-server created
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader created
clusterrolebinding.rbac.authorization.k8s.io/system:auth-delegator created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server created
service/metrics-server created
deployment.apps/metrics-server created
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io created
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server

- Pod 활성화 명령어 입력 : \$ minikube addons enable metrics-server

```
C:\Users\lanece>minikube addons enable metrics-server
* metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
- Using image registry.k8s.io/metrics-server/metrics-server:v0.6.4
* 'metrics-server' 애드온이 활성화되었습니다
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server

- 활성화 확인 명령어 입력 : \$ minikube addons list

kubevirt	minikube	disabled	3rd party (KubeVirt)
logviewer	minikube	disabled	3rd party (unknown)
metallb	minikube	disabled	3rd party (MetallLB)
metrics-server	minikube	enabled ✓	Kubernetes
nvidia-device-plugin	minikube	disabled	3rd party (NVIDIA)
nvidia-driver-installer	minikube	disabled	3rd party (Nvidia)

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server - Node CPU 사용량 측정

- 명령어 입력 : \$ kubectl **top nodes**

```
C:\Users\Wanece>kubectl top nodes
NAME      CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
minikube  219m        5%     1099Mi         13%
```

C:\Users\Wanece>

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server - Node 정보 확인

- 명령어 입력 : \$ kubectl **describe nodes**

Conditions:			
Type	Status	LastHeartbeatTime	LastTransitionTime
MemoryPressure	False	Thu, 04 Apr 2024 11:11:54 +0900	Thu, 04 Apr 2024 11:11:28 +0900
DiskPressure	False	Thu, 04 Apr 2024 11:11:54 +0900	Thu, 04 Apr 2024 11:11:28 +0900
PIDPressure	False	Thu, 04 Apr 2024 11:11:54 +0900	Thu, 04 Apr 2024 11:11:28 +0900
Ready	True	Thu, 04 Apr 2024 11:11:54 +0900	Thu, 04 Apr 2024 11:11:44 +0900

상태
확인

Reason	Message
KubeletHasSufficientMemory	kubelet has sufficient memory available
KubeletHasNoDiskPressure	kubelet has no disk pressure
KubeletHasSufficientPID	kubelet has sufficient PID available
KubeletReady	kubelet is posting ready status

문제 발생 원인

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 지표 서버 Metrics-Server - Node 정보 확인

- 명령어 입력 : \$ kubectl **describe nodes**

```
Namespace          Name
----              ---
kube-system       coredns-5dd5756b68-vfqpl
kube-system       etcd-minikube
kube-system       kube-apiserver-minikube
kube-system       kube-controller-manager-minikube
kube-system       kube-proxy-4b42
kube-system       kube-scheduler-minikube
kube-system       storage-provisioner

Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
Resource          Requests   Limits
----              ----      -----
cpu               750m (18%)  0 (0%)
memory           170Mi (2%)  170Mi (2%)
ephemeral-storage 0 (0%)    0 (0%)
hugepages-1Gi    0 (0%)    0 (0%)
hugepages-2Mi    0 (0%)    0 (0%)
```

	CPU Requests	CPU Limits	Memory Requests	Memory Limits	Age
coredns-5dd5756b68-vfqpl	100m (2%)	0 (0%)	70Mi (0%)	170Mi (2%)	49s
etcd-minikube	100m (2%)	0 (0%)	100Mi (1%)	0 (0%)	61s
kube-apiserver-minikube	250m (6%)	0 (0%)	0 (0%)	0 (0%)	61s
kube-controller-manager-minikube	200m (5%)	0 (0%)	0 (0%)	0 (0%)	61s
kube-proxy-4b42	0 (0%)	0 (0%)	0 (0%)	0 (0%)	49s
kube-scheduler-minikube	100m (2%)	0 (0%)	0 (0%)	0 (0%)	61s
storage-provisioner	0 (0%)	0 (0%)	0 (0%)	0 (0%)	58s

파드 메모리 요청 상태
및 시스템의 가용 상태

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

- 지표 서버 Metrics-Server - Pod CPU 사용량 측정

- 명령어 입력 : \$ kubectl top pods -n kube-system

```
so@LAPTOP-KGH195H9:~$ kubectl top pods
NAME                           CPU(cores)   MEMORY(bytes)
nginx-deployment-5d5c5c44c7-9kkdp   0m          4Mi
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 로깅 Logging - Control Plane

스케줄러, API 서버 등 같은 쿠버네티스 컨트롤 플레인 구성 요소에서 생성된 로그

【Master Node】

- 스케줄러 로그 : /var/log/kube-scheduler.log
- 컨트롤러 메시지 로그 : /var/log/kube-controller-manager.log
- API 서버 로그 : /var/log/kube-apiserver.log

【Worker Node】

- kubelet 로그 : /var/log/kubelt.log
- kube proxy : /var/log/kube-proxylog

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

- 로깅 Logging - Application

스케줄러, API 서버 등 같은 쿠버네티스 컨트롤 플레인 구성 요소에서 생성된 로그

【Pod Log】

- 명령어 입력 : \$ kubectl logs <파드 이름> <컨테이너 이름>

【Worker Node】

- kubelet 로그 : /var/log/kubelt.log
- kube proxy : /var/log/kube-proxylog

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 대시보드 (Dashboard)

GUI기반 로그 보기, 리소스 편집 등의 kubectl 모든 기능 제공

- 명령어 입력 : \$ minikube dashboard

```
so@LAPTOP-KGH195H9:~$ minikube dashboard
? Enabling dashboard ...
  - Using image docker.io/kubernetesui/dashboard:v2.7.0
  - Using image docker.io/kubernetesui/metrics-scraper:v1.0.8
? Some dashboard features require the metrics-server addon. To enable all features please run:
  minikube addons enable metrics-server

? Verifying dashboard health ...
? Launching proxy ...
? Verifying proxy health ...
? Opening http://127.0.0.1:43607/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard
n your default browser...
? http://127.0.0.1:43607/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 대시보드 (Dashboard)

<http://127.0.0.1:43607/api/v1/namespaces/kubernetes-dashboard/services/http:kubernetes-dashboard:/proxy/>

The screenshot displays two identical views of the Kubernetes Dashboard, each showing the details of an 'nginx-deployment' pod named '5d5c5c44c7-9kkdp'. The interface includes a sidebar with navigation links like '워크로드', '파드', '서비스', and '인그레스'. The main area is divided into two sections: 'CPU Usage' and 'Memory Usage', both of which currently show a message: '차트를 표시하기 위해 추가 데이터를 기다리는 중...'. Below these sections is a table titled '파드' (Pods) with the following data:

이름	이미지	레이블	노드	상태	재시작	CPU 사용량 (cores)	메모리 사용량 (bytes)	생성 시간
nginx-deployment-5d5c5c44c7-9kkdp	nginx	app: webserver pod-template-hash: 5d5c5c44c7	minikube	Running	0	0.00m	4.39Mi	41 minutes ago

OBSERVABILITY

◆ 관찰 도구(Observation Tool)

● 프로메테우스 Prometheus

- 사운드클라우등서 최초 개발 → 현재 CNCF(오픈소스)에 속한 모니터링 솔루션
- 모니터링 기능이 설치된 컨테이너가 파드 안에 존재
- CNCF가 호스트하는 오픈 소스 소프트웨어 모니터링 도구
- 운영 비용이나 서버 측 머신 리소스가 필요
- 여러 가지 지표(메트릭)을 수집/그래프화/모니터링 가능
- Pull 방식 → 서버에 에이전트가 떠있으면 프로메테우스에서 주기적으로 에이전트에 접속해서 데이터를 가져오는 방식
- 서버에 에이전트를 설치하고 메트릭 데이터를 수집해서 모니터링 서버로 보내면 상태를 보여주는 방식

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

● 설치 전 준비

- URL : <https://github.com/prometheus-operator/kube-prometheus>

Kubernetes 버전에 맞는 버전 클릭 후 사이트 이동

The following Kubernetes versions are supported and work as we test against these versions in their respective branches. But note that other versions might work!

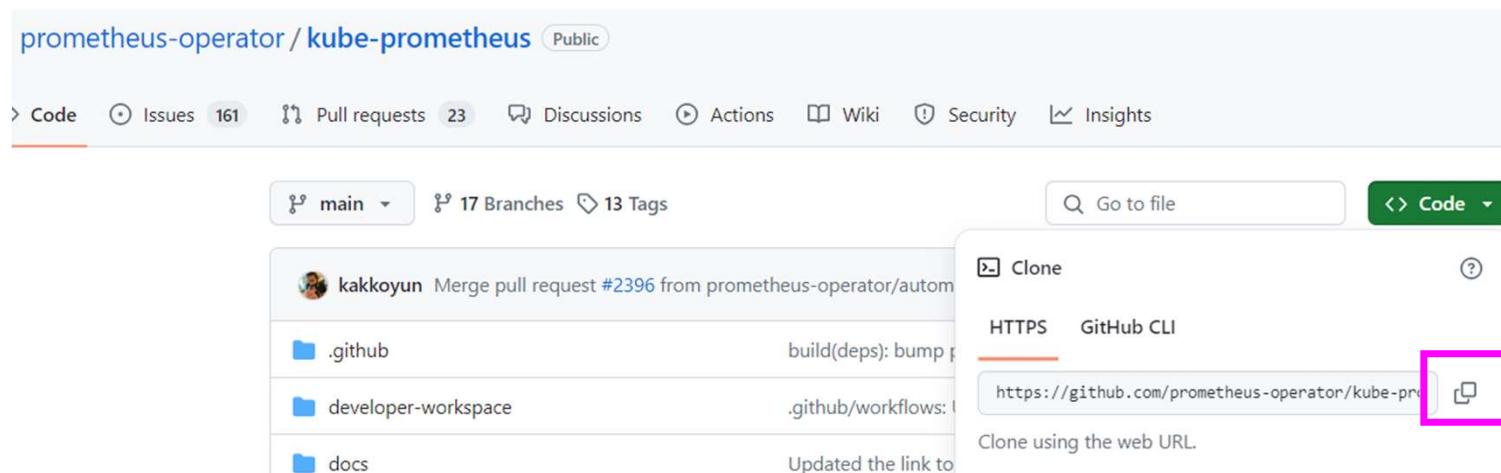
kube-prometheus stack	Kubernetes 1.22	Kubernetes 1.23	Kubernetes 1.24	Kubernetes 1.25	Kubernetes 1.26	Kubernetes 1.27	Kuberr 1.2
release-0.10	✓	✓	X	X	X	X	X
release-0.11	X	✓	✓	X	X	X	X
release-0.12	X	X	✓	✓	X	X	X
release-0.13	X	X	X	X	✓	✓	✓
main	X	X	X	X	X	✓	✓

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 전 준비

- Kubernetes 버전에 맞는 버전의 다운로드 URL 복사



OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 전 준비

- ① Prometheus 다운로드

- 명령어 : \$ git clone <https://github.com/prometheus-operator/kube-prometheus.git>

```
so@k8s:~$ git clone https://github.com/prometheus-operator/kube-prometheus.git
Cloning into 'kube-prometheus'...
remote: Enumerating objects: 19542, done.
remote: Counting objects: 100% (6611/6611), done.
remote: Compressing objects: 100% (513/513), done.
remote: Total 19542 (delta 6417), reused 6111 (delta 6095), pack-reused 12931
Receiving objects: 100% (19542/19542), 10.34 MiB | 15.27 MiB/s, done.
Resolving deltas: 100% (13349/13349), done.
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 전 준비

- ② CRC 스팩 설정 폴더 이동

- 이동 명령어 : \$ cd **kube-prometheus**

```
so@k8s:~$ ls -l
total 24
drwxr-xr-x 12 so so 4096 Apr 17 00:23 kube-prometheus
-rw-r--r--  1 so so  382 Apr  4 15:29 monitoring_app.yaml
-rw-r--r--  1 so so  414 Apr  4 15:29 monitoring_service.yaml
-rw-r--r--  1 so so  317 Apr  4 12:49 nginx.yaml
-rw-r--r--  1 so so  317 Apr  4 12:51 nginx.yaml
-rw-r--r--  1 so so  216 Apr  4 12:53 webserver.yaml
so@k8s:~$
```

OBSERVATION

사용자정의자원 CRD
생성 파일 위치 확인
manifests/setup

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 전 준비 <https://velog.io/@hkjs96/Kubernetes-%EB%AA%A8%EB%8B%88%ED%84%B0%EB%A7%81>

- 경로 이동 명령어 : kube-prometheus \$ cd manifests/setup
- ③ 폴더 내부 구조 : kube-prometheus /manifests/setup \$ ls -l

```
so@k8s:~/my-prometheus-grafana$ ls -l
total 44
-rw-r--r-- 1 so so 1006 Apr 17 01:10 grafana.yaml
-rw-r--r-- 1 so so 1811 Apr 17 01:10 kube-state-cluster-role.yaml
-rw-r--r-- 1 so so 932 Apr 17 01:10 kube-state-deployment.yaml
-rw-r--r-- 1 so so 330 Apr 17 01:10 kube-state-svc.yaml
-rw-r--r-- 1 so so 98 Apr 17 01:10 kube-state-svcaccount.yaml
-rw-r--r-- 1 so so 652 Apr 17 01:10 prometheus-cluster-role.yaml
-rw-r--r-- 1 so so 5344 Apr 17 01:10 prometheus-config-map.yaml
-rw-r--r-- 1 so so 930 Apr 17 01:10 prometheus-deployment.yaml
-rw-r--r-- 1 so so 710 Apr 17 01:10 prometheus-node-exporter.yaml
-rw-r--r-- 1 so so 308 Apr 17 01:10 prometheus-svc.yaml
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 전 준비

- ④ 프로메테우스 CRD(Custom Resource Definition) 생성

- CRD 생성 명령어 : \$ **kubectl create -f manifests/setup**

monitoring 네임스페이스 생성

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 전 준비

- ④ 프로메테우스 CRD(Custom Resource Definition) 생성

```
clusterrolebinding.rbac.authorization.k8s.io/prometheus-operator serverside-applied
Warning: spec.template.spec.nodeSelector[beta.kubernetes.io/os]: deprecated since v1.14; use "kubernetes.io/os" instead
deployment.apps/prometheus-operator serverside-applied
service/prometheus-operator serverside-applied
serviceaccount/prometheus-operator serverside-applied
resource mapping not found for name: "alertmanagers.monitoring.coreos.com" namespace: "" from "manifests/setup/prometheus-"
": no matches for kind "CustomResourceDefinition" in version "apiextensions.k8s.io/v1beta1"
ensure CRDs are installed first
```

yaml 파일에서

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

● 설치 전 준비

④ 프로메테우스 CRD(Custom Resource Definition) 생성

```
so@k8s: ~/kube-prometheus-0.5.0/manifests/setup
  GNU nano 6.2
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  annotations:
    controller-gen.kubebuilder.io/version: v0.2.4
  creationTimestamp: null
  name: thanosrulers.monitoring.coreos.com
spec:
```

```
so@k8s:~/kube-prometheus-0.5.0/manifests/setup$ sed -i "s/apiextensions.k8s.io/kubernetes.io/g" *.yaml  
so@k8s:~/kube-prometheus-0.5.0/manifests/setup$ sed -i "s/v1beta1/os/g" *.yaml  
so@k8s:~/kube-prometheus-0.5.0/manifests/setup$
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

● 설치 준비

① Prometheus를 올릴 Kubernetes 환경 준비

Prometheus 스택 배포하기 위한 "monitoring" namespace 생성

- 네임스페이스 생성 : \$ kubectl **create ns monitoring**
namespace/monitoring created
- 네임스페이스 확인 : \$ kubectl **get ns**

NAME	STATUS	AGE
default	Active	93m
kube-node-lease	Active	93m
kube-public	Active	93m
kube-system	Active	93m
kubernetes-dashboard	Active	56m
monitoring	Active	5s
staging	Active	81m

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 준비

② Prometheus 설치 파일 준비 및 폴더 이동

- 명령어: \$ git clone <https://github.com/hali-linux/my-prometheus-grafana.git>
- 명령어: \$ cd **my-prometheus-grafana**

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

● 설치 준비

② Prometheus 설치 파일 준비 및 폴더 이동

- 명령어: my-prometheus-grafana \$ ls -l

```
so@k8s:~/my-prometheus-grafana$ ls -l
total 48
-rw-r--r-- 1 so so 1006 Apr 17 01:10 grafana.yaml
-rw-r--r-- 1 so so 1811 Apr 17 01:10 kube-state-cluster-role.yaml
-rw-r--r-- 1 so so 932 Apr 17 01:10 kube-state-deployment.yaml
-rw-r--r-- 1 so so 330 Apr 17 01:10 kube-state-svc.yaml
-rw-r--r-- 1 so so 98 Apr 17 01:10 kube-state-svccount.yaml
-rw-r--r-- 1 so so 642 Apr 17 01:25 prometheus-cluster-role.yaml
-rw-r--r-- 1 so so 652 Apr 17 01:10 prometheus-cluster-role.yaml.bak
-rw-r--r-- 1 so so 5344 Apr 17 01:10 prometheus-config-map.yaml
-rw-r--r-- 1 so so 930 Apr 17 01:10 prometheus-deployment.yaml
-rw-r--r-- 1 so so 710 Apr 17 01:10 prometheus-node-exporter.yaml
-rw-r--r-- 1 so so 308 Apr 17 01:10 prometheus-svc.yaml
so@k8s:~/my-prometheus-grafana$
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 진행

- ③ Prometheus 관련 설정 파일 적용

- [Prometheus의 Kubenetes API 접근 권한 부여]

- 명령어 : \$ kubectl create -f [prometheus-cluster-role.yaml](#)

- [Prometheus 수집 지표 및 주기, 알림 정의]

- 명령어 : \$ kubectl create -f [prometheus-config-map.yaml](#)

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 진행

- ③ Prometheus 관련 설정 파일 적용

- [Prometheus Server 파드 생성]

- 명령어 : \$ kubectl create -f [prometheus-deployment.yaml](#)

- [노드 정보 내보내기(node-exporter) 즉, kubernetes Node 정보 : 데몬셋 파드]

- 명령어 : \$ kubectl create -f [prometheus-node-exporter.yaml](#)

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 진행

- ③ Prometheus 관련 설정 파일 적용

- [Prometheus 노드 정보 노출 서비스 생성]

- 명령어 : \$ kubectl create -f [prometheus-svc.yaml](#)

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 진행

- ③ 쿠버네티스의 자원 사용량에 대한 에이전트

- [Prometheus 노드 정보 노출 서비스 생성]

- \$ kubectl apply -f kube-state-cluster-role.yaml
 - \$ kubectl apply -f kube-state-deployment.yaml
 - \$ kubectl apply -f kube-state-svcaccount.yaml
 - \$ kubectl apply -f kube-state-svc.yaml

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 진행

- ④ Prometheus 관련 네이스페이스 확인

- 명령어 : \$ kubectl get pod -n **monitoring**
- 상세 명령어 : \$ kubectl get pod -n **monitoring** -o wide

```
so@k8s:~/my-prometheus-grafana$ kubectl get pod -n monitoring -o wide
NAME                           READY   STATUS    RESTARTS   AGE     IP           NODE
node-exporter-brknt            1/1     Running   0          63m    10.244.0.9   minikube
prometheus-deployment-568f7f568f-s588k  1/1     Running   0          63m    10.244.0.8   minikube
so@k8s:~/my-prometheus-grafana$
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- 설치 진행

- ⑤ Prometheus 관련 서비스 확인

- 명령어 : \$ kubectl get service -n monitoring -o wide

```
so@k8s:~/my-prometheus-grafana$ kubectl get service -n monitoring -o wide
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE   SELECTOR
prometheus-service   NodePort  10.104.140.27 <none>       8080:30003/TCP  65m  app=prometheus-server
so@k8s:~/my-prometheus-grafana$
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 프로메테우스 Prometheus

- 설치 완료 및 구동 확인

- ⑥ Prometheus 서비스 구동

- 명령어 : \$ minikube service --url **prometheus-service** -n monitoring
http://10.104.140.27:30003

```
so@k8s:~$ kubectl get service -n monitoring
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
prometheus-service   NodePort   10.104.140.27 <none>        8080:30003/TCP   23m

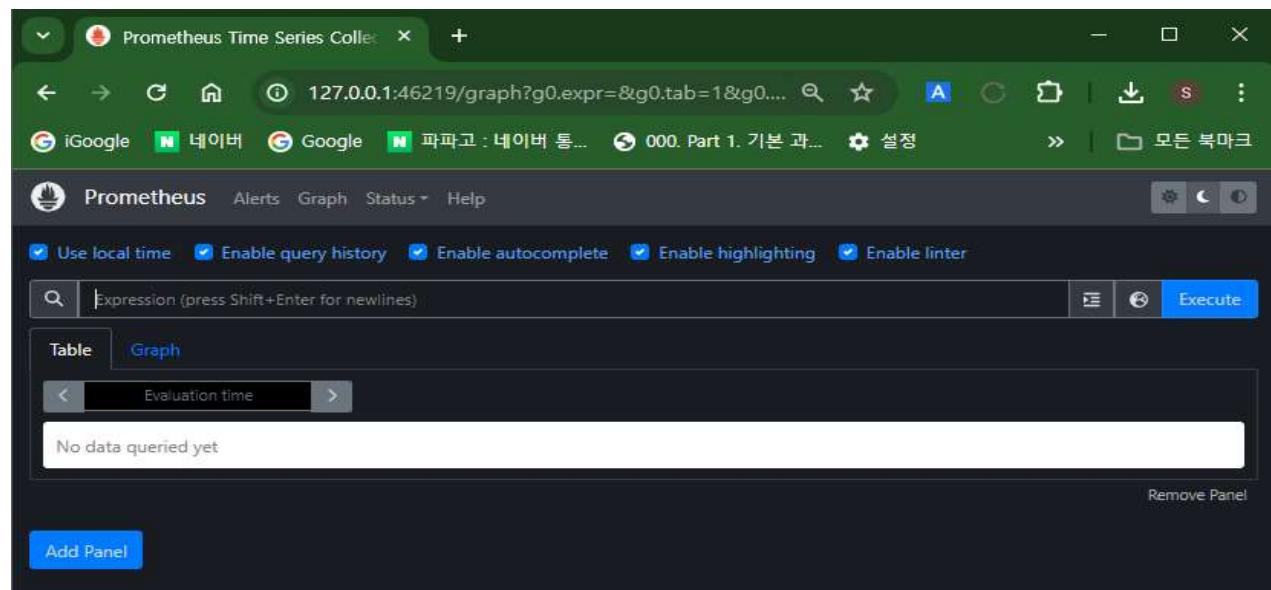
so@k8s: ~$ minikube service --url prometheus-service -n monitoring http://10.104.140.27:30003
http://127.0.0.1:46219
! Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - [프로메테우스 Prometheus](#)

- [Prometheus 서비스](#)

⑥ 서비스 구동



OBSERVABILITY

◆ 관찰 도구(Observation Tool)

- 그라파나 Grafana

- 2014년 Torked Odegaard 주도로 처음 릴리즈
- Prometheus에서 수집된 메트릭 이용해 조회/시각화/경고 기능 지원

OBSERVABILITY

◆ 관찰 도구(Observation Tool) - 그라파나 Grafana

● 설치 준비

① Prometheus를 올릴 Kubernetes 환경 준비

Prometheus 스택 배포하기 위한 "monitoring" namespace 생성

- 네임스페이스 생성 : \$ kubectl **create ns monitoring**
namespace/monitoring created
- 네임스페이스 확인 : \$ kubectl **get ns**

NAME	STATUS	AGE
default	Active	93m
kube-node-lease	Active	93m
kube-public	Active	93m
kube-system	Active	93m
kubernetes-dashboard	Active	56m
monitoring	Active	5s
staging	Active	81m

PART I

TROUBLESHOOTING

TROUBLESHOOTING

◆ 관찰 가능성(Observability)

- 시스템이 점점 분산되면서 구축하고 운영하는 방법론 빠르게 진화
- 다양한 IT 인프라 환경의 공존과 복잡성 증대 → 서비스와 인프라 가시성 중요

2018년 진화

- '외부 출력값만을 사용해 시스템의 내부 상태를 관찰할 수 있다' 기본 개념
- '내부 상태에서 수집한 데이터 사용하여 시스템 동작 이해 및 디버깅 가능'
- 시스템은 내부 동작에 관한 풍부한 맥락 정보 제공, 더 깊은 시스템 문제 밝힐 수 있음

TROUBLESHOOTING

◆ 분산 애플리케이션(Dapp:Distributed Apps) 장애

❖ 분산 애플리케이션

- 여러 개의 독립적이고 작은 부분들로 기능을 나누어 동작하는 애플리케이션
 - 서로 다른 문제를 처리하는 개별 마이크로서비스 컴포넌트들로 구성된 애플리케이션
 - 예] 클라우드 네이티브 환경에서 개별 컴포넌트들이 클러스터에서 컨테이너로 실행
-
- 쿠버네티스 구성 요소 ➔ 2개 이상의 복제본 실행 전체로 배포 : 분산 애플리케이션

TROUBLESHOOTING

◆ 분산 애플리케이션(Dapp:Distributed Apps) 장애

❖ 장애 요인

- 네트워크 안정성
- 네트워크 지연시간
- 대역폭 제한
- 네트워크 보안 취약
- 네트워크 토폴로지(topology) 변경
- 다수 관리자 및 충돌 규칙
- 상이한 애플리케이션 네트워킹 구현 환경

TROUBLESHOOTING

◆ 분산 애플리케이션(Dapp:Distributed Apps) 장애

❖ 쿠버네티스 클러스터 문제 - 파드 배치 실패

[현상] 파드 무기한 Pending 상태

[일반적 원인] 자원 부족

- | | |
|---------------|--|
| ➤ 노드 상태 확인 | : 파드 배치 가능한지 노드 상태 확인
노드가 준비 상태인지, 사용 중이지는 않은지, 파드 필요 요구 사항
충족시킬 만큼 충분한 자원(예: CPU, 메모리) 가지고 있는지 확인 |
| ➤ 파드 스펙 확인 | : 파드 스펙(예: 요청한 리소스, 레이블 선택기 등) 정의 바른지 확인 |
| ➤ 이벤트 로그 확인 | : 어떤 이유로 파드가 Pending 상태에 머물러 있는지 확인 |
| ➤ 컨테이너 이미지 확인 | : 파드 내부의 컨테이너 이미지가 올바르게 정의되어 있는지 확인 |

TROUBLESHOOTING

◆ 분산 애플리케이션(Dapp:Distributed Apps) 장애

❖ 쿠버네티스 클러스터 문제 - 파드 배치 실패

[현상] 파드 Running 상태지만 오작동

[일반적 원인] 컨테이너 구성 또는 이미지 문제

- 로그 확인 : 파드 내부 컨테이너 로그 확인하여 컨테이너 오작동 원인 파악
- 파드 이벤트 확인 : 이벤트 로그 확인으로 원인 파악
- 컨테이너 설정 확인 : 컨테이너 설정(예: 환경 변수, 볼륨 마운트 등)이 올바른지 확인
- 이미지 업데이트 확인 : 파드의 이미지 버전이 최신 버전인지 확인 및 업데이트