

2024.03.13

# 캡스톤 디자인 계획 발표

백엔드 서비스 프로바이더  
: 컨테이너 오케스트레이션 플랫폼 개발

팀명 : 악동개발자  
20211939 허유정  
20217140 김창인





## 개발 배경 및 필요성

- 1) 개발 배경
- 2) 현재 기술 상황 및 한계



## 개발 계획

- 1) 요구사항 정의서
- 2) 시스템 구조도
- 3) 응용 분야



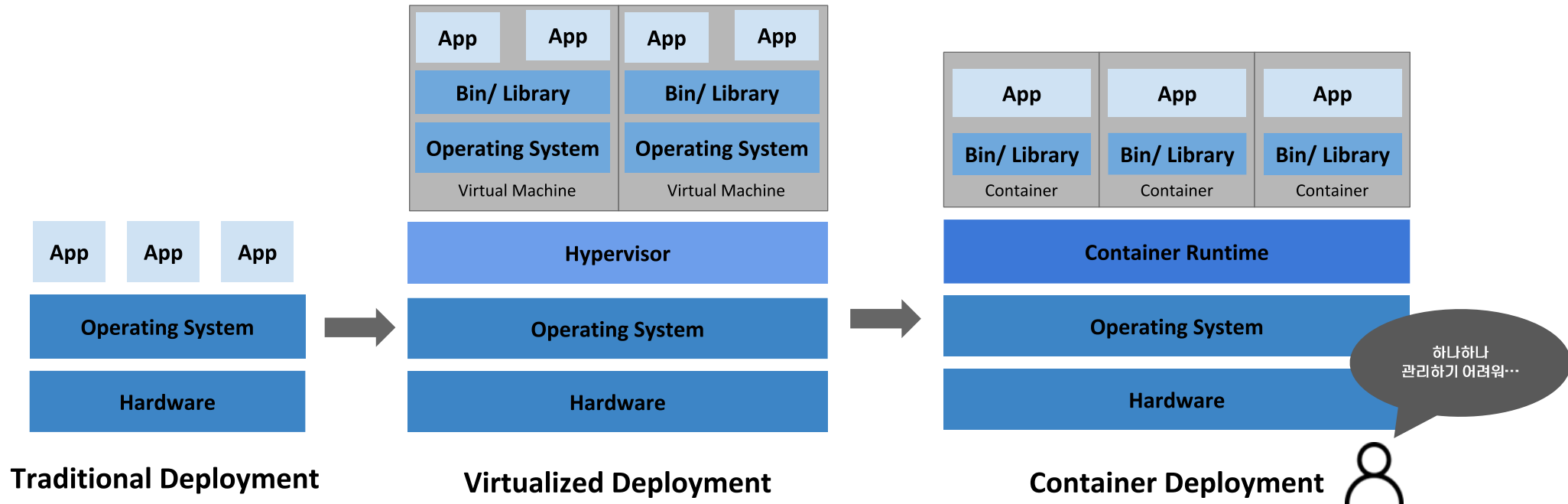
## 추진전략 및 기대효과

- 1) 추진전략
- 2) 역할분담
- 3) 진행계획
- 4) 기대효과 및 활용방안

## part1 개발 배경 및 필요성 - 개발배경



- 컨테이너 기술의 등장으로 애플리케이션의 빠른 배포와 확장, 일관된 환경 실행으로 서버 관리의 효율성 증대
- 컨테이너 관리 복잡성 증가로 효율적 운영을 위한 컨테이너 오케스트레이션 도구 필요성 대두



출처 : 쿠버네티스 공식 홈페이지

- 쿠버네티스 사용
  - 대표적인 컨테이너 오케스트레이션 사용을 통한 비교
  - 쿠버네티스 설치 과정에서 많은 시간이 들었으며 다양한 오류를 마주함
  - 실제로 필요한 기능과 기술적 역량을 고려 했을 때, 과도한 복잡성을 가지고 있음

```
yj@master:~$ kubectl get node -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-IMAGE	KERNEL-VERSION	CONTAINER-RUNTIME
master	Ready	control-plane	7m40s	v1.28.2	192.168.50.10	<none>	Ubuntu 20.04.6 LTS	5.15.0-91-generic	containerd://1.6.26
worker1	Ready	worker	5m48s	v1.28.2	192.168.50.88	<none>	Ubuntu 20.04.6 LTS	5.15.0-91-generic	containerd://1.6.26
worker2	Ready	worker	4m47s	v1.28.2	192.168.50.100	<none>	Ubuntu 20.04.6 LTS	5.15.0-91-generic	containerd://1.6.26

▲ 쿠버네티스 클러스터 구축

```
changelin@syslab:~$ sudo net-get update
```

```
Hit1: http://security.ubuntu.com/ubuntu focal-security InRelease
Hit2: http://ppa.launchpad.net/graphics-drivers/ppa/ubuntu focal InRelease
Hit3: http://fr.archive.ubuntu.com/ubuntu focal InRelease
Hit4: http://fr.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit5: http://fr.archive.ubuntu.com/ubuntu focal-backports InRelease
Reading package lists... Done
changelin@syslab:~$ sudo dpkg-query -f='${Package} ${Version} ${Architecture} ${Vendor} ${Source} ${Description}\n'
```

```
Package: nvidia-driver-545-open
Version: 545.06.02
Architecture: amd64
Vendor: NVIDIA Corporation
Source: nvidia-driver-545-open
Description: Third-party non-free nvidia-driver-545-open - distro non-free
```




```
yj@syslab:~$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
syslab	NotReady	control-plane,master	92s	v1.21.7

```
yj@syslab:~$ kubectl get pods -n kube-system -o wide
```

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
coredns-558bd4d5db-qcpl	0/1	Pending	0	96s	<none>	<none>	<none>	<none>
coredns-558bd4d5db-v9x4l	0/1	Pending	0	96s	<none>	<none>	<none>	<none>
etcd-syslab	1/1	Running	21	103s	192.168.50.10	syslab	<none>	<none>
kube-apiserver-syslab	1/1	Running	22	112s	192.168.50.10	syslab	<none>	<none>
kube-controller-manager-syslab	1/1	Running	19	111s	192.168.50.10	syslab	<none>	<none>
kube-flannel-ds-rv2vr	1/1	Running	0	91s	192.168.50.10	syslab	<none>	<none>
kube-proxy-pxlvr	1/1	Running	0	97s	192.168.50.10	syslab	<none>	<none>
kube-scheduler-syslab	1/1	Running	19	111s	192.168.50.10	syslab	<none>	<none>

▲ 실제 설치 과정에서 발생한 오류

	 Kubernetes	 Docker Swarm	 Apache Mesos
정의	컨테이너화 된 애플리케이션의 배포, 확장 및 관리 자동화	여러 도커 호스트를 하나의 가상 도커 호스트로 관리	대규모 클러스터 관리를 위한 플랫폼으로 분산 시스템 효율적 실행 가능
장점	<ul style="list-style-type: none"><li>- 수천개 컨테이너 관리가 가능한 높은 확장성</li><li>- 자동 복구, 롤링 업데이트, 서비스 검색 가능</li></ul>	도커 생태계와의 높은 호환성과 통합성	<ul style="list-style-type: none"><li>- 수만개의 노드 관리가 가능한 높은 확장성</li><li>- 자원 분할과 공유로 효율적인 자원 사용 가능</li></ul>
단점	초기 설정과 관리가 복잡하여 학습 곡선이 높음	대규모 시스템에서의 제약	쿠버네티스보다 높은 학습 곡선



기존 오케스트레이션 도구는 복잡성과 높은 학습 곡선으로 인해 **소규모 프로젝트나 조직에 부담**  
특정 제품에 종속되지 않고 추가적인 비용 없이 사용가능한 **경량화 된 컨테이너 오케스트레이션 플랫폼 필요**  
GUI 제공을 통해 사용자 **친화적이며 간편**한 사용이 가능한 플랫폼의 필요



- 경량화 된 컨테이너 오케스트레이션 플랫폼 개발을 목표로 하며 배포관리, 스케일링, 네트워킹 자동화 기능 부여
- 사용자 직관적이며 효율적인 관리 및 모니터링을 위한 웹 기반 UI 제공

요구사항 명	요구사항 설명	요구사항 ID
컨테이너 런타임 관리	실행 중인 컨테이너에 대한 기본적인 관리 기능을 제공(예. 시작, 정지, 재시작, 삭제)	REQ-001
스케일링	수동 또는 자동으로 컨테이너의 인스턴스 수를 조정	REQ-002
네트워킹	컨테이너 간 통신 및 외부 네트워크와의 연결을 관리할 수 있는 기능 제공	REQ-003
사용자 인터페이스	컨테이너를 실행 및 시스템을 모니터링 할 수 있는 직관적인 웹 기반 UI를 제공 이상 상태 감지 시 사용자에게 알림을 제공	REQ-004
테스트(개발)	게임 서버 구축을 통한 컨테이너 오케스트레이션 플랫폼 기능성 테스트 진행	REQ-005

### [ 요구사항 정의서 세부사항 ]

#### 컨테이너 런타임 관리(REQ-001)

요구사항 명	요구사항 설명	요구사항 ID
컨테이너 시작	사용자가 직접 컨테이너를 시작할 수 있는 기능 추가 제공	REQ-001-001
컨테이너 정지	실행 중인 컨테이너를 정지할 수 있는 기능	REQ-001-002
컨테이너 재시작	정지된 컨테이너를 재시작하여 가동할 수 있는 기능	REQ-001-003
컨테이너 삭제	더 이상 필요하지 않은 컨테이너를 사용자가 직접 삭제할 수 있는 기능	REQ-001-004

#### 스케일링(REQ-002)

요구사항 명	요구사항 설명	요구사항 ID
수동 스케일링	사용자가 수동으로 컨테이너의 인스턴스 수를 조정할 수 있는 기능	REQ-002-001
자동 스케일링	시스템이 자동으로 한정된 리소스에 맞추어 인스턴스 수 조정할 수 있는 기능	REQ-002-002

### [ 요구사항 정의서 세부사항 ]

#### 네트워킹(REQ-003)

요구사항 명	요구사항 설명	요구사항 ID
컨테이너 간 통신 관리	컨테이너 간의 통신을 관리하고 보안을 유지할 수 있는 기능	REQ-003-001
외부 네트워크 연결 관리	컨테이너와 외부 네트워크 간의 연결을 관리하고 보안을 유지할 수 있는 기능	REQ-003-002

#### 사용자 인터페이스(REQ-004)

요구사항 명	요구사항 설명	요구사항 ID
웹 기반 UI 제공	모니터링이 가능한 직관적이고 사용성이 높은 웹 기반 UI 제공	REQ-004-001
이상 상태 알림	시스템이 컨테이너 관리 리소스의 이상 상태를 감지 시, 사용자에게 경고 및 알림 제공	REQ-004-002

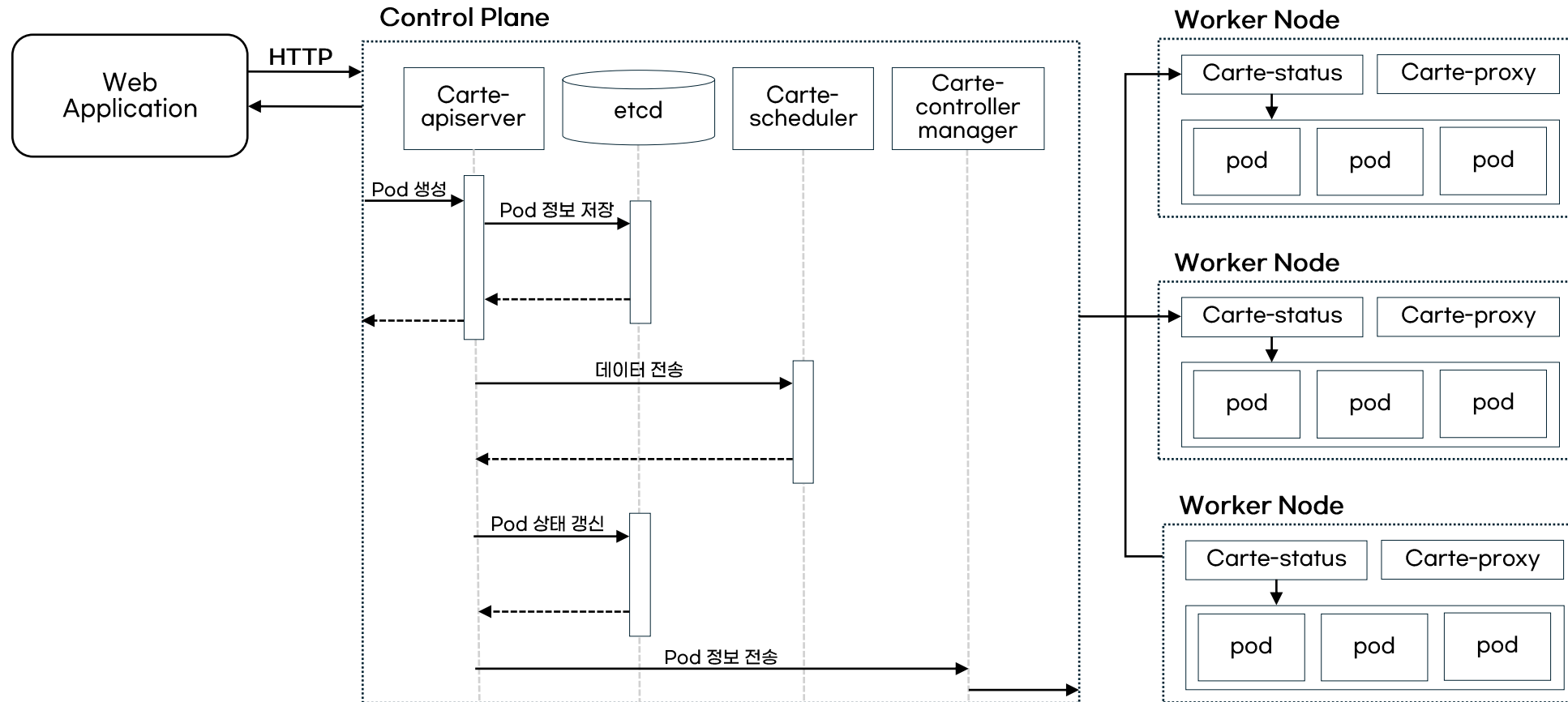




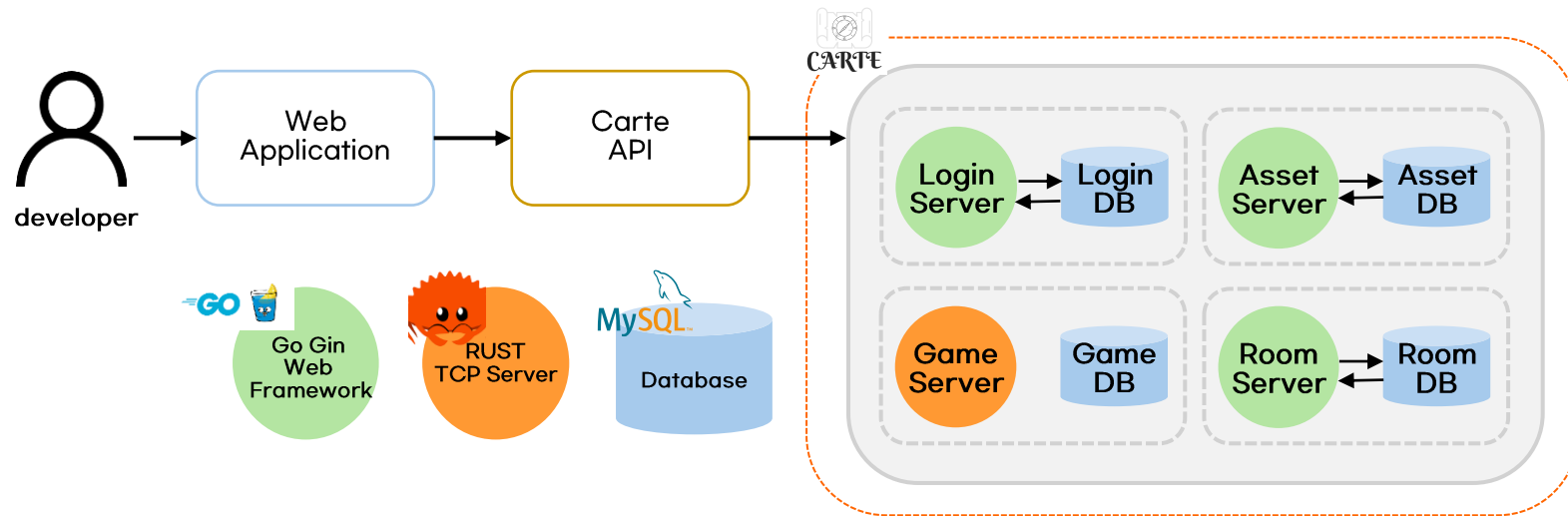
[ 요구사항 정의서 세부사항 ]

테스트(REQ-005)

요구사항 명	요구사항 설명	요구사항 ID
게임 서버 구축(Login)	플레이어 정보 저장 및 관리가 가능한 서버	REQ-005-001
게임 서버 구축(Asset)	플레이어가 제작한 Asset(제작한 도구)정보의 저장 및 관리가 가능한 서버	REQ-005-002
게임 서버 구축(Room)	플레이어가 제작한 Room(제작한 배경)정보의 저장 및 관리가 가능한 서버	REQ-005-003
게임 서버 구축(Game)	동시 사용자 접속 시, 멀티 플레이어의 통신 관리가 가능한 서버	REQ-005-004
사용자 경험 테스트	실제 사용자 환경에서의 제품 사용성과 사용자의 만족도를 평가하기 위한 테스트 수행	REQ-005-005



- 멀티플레이어 게임 서버 구축을 통해 개발된 컨테이너 오케스트레이션 플랫폼의 기능성 검증은 목표로 함
- 고성능, 고가용성을 보장하는 게임 서버 인프라로 복잡한 서버 관리 작업 없이 간편한 서비스 이용을 확인하기 위함
- Go 및 Rust 프로그래밍 언어와 TCP, HTTP프로토콜의 사용으로 효율적인 실시간 멀티플레이어 게임 서버 구축





프로젝트 관리

- 주기적인 스프린트 계획과 리뷰를 통한 진행 상황 모니터링
- 소프트웨어 아키텍처 설계를 통한 시스템 이해도 확립
- 애자일 개발 방법론 채택으로 프로젝트 유연성 확립



Github

- 소스 코드와 변경 사항의 효과적인 추적 및 관리 가능
- 풀 리퀘스트 생성으로 변경 사항 검토 및 통합
- 개발자들 간 소스 코드 공유 및 병합 간편화

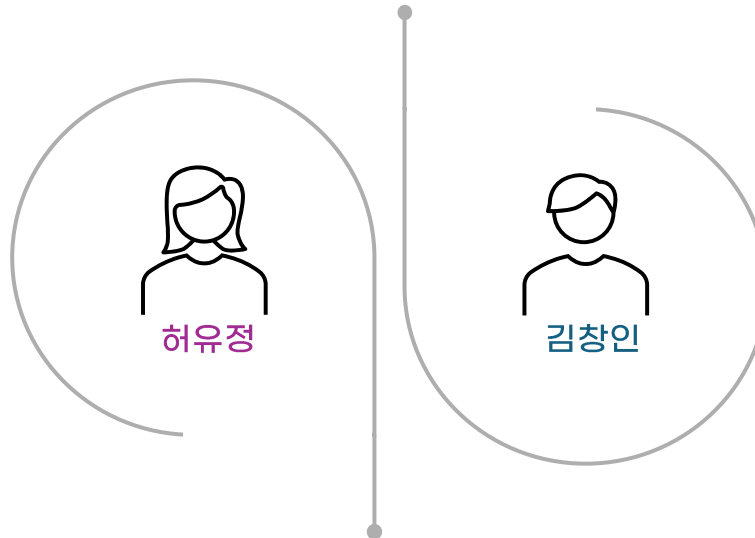


Microsoft Teams

- 계획 공유를 통한 일정 관리
- 문서 공유의 불편함 최소화
- 멘토와의 의사소통 강화

### Go 언어 활용

- 컨테이너 오케스트레이션 플랫폼  
컨테이너 런타임 및 네트워킹 관리  
의 기본적인 기능 관리
- 게임서버 구축
  - 비실시간 통신 관리
  - 데이터 스키마 구성



### Rust 언어 활용

- 컨테이너 오케스트레이션 플랫폼
  - 스케일링 및 고성능과 안정성이  
요구되는 기능 관리
  - 플랫폼 개발을 위한 환경 구축
- 게임서버 구축
  - 실시간 통신 관리

## part3 추진전략 및 기대효과 - 진행 계획



- 직관적인 GUI 기반 도구 제공을 통한 시스템 모니터링 및 관리가 가능하여 개발자의 작업 부담을 줄이고, 생산성을 향상시킬 것으로 기대
- 특허 및 실용신안 출원을 통해 개발된 기술의 지적 재산을 확보함으로써, 기술적 리더십을 확립하고 장기적인 사업화 기반 마련 가능
- 컨테이너화 및 오케스트레이션 기술 교육 자료로 활용하여 실습 중심의 학습을 통해 학생들에게 인프라 관리 기술을 가르치는데 유용한 도구가 될 수 있을 것으로 기대함



- [1]P. Sharma, et al., “Containers and virtual machines at scale: A comparative study,” in Proc. 17th Int. Middleware Conf. ACM, Trento, Italy, Nov. 2016
- [2]최원석, 정혜진, 나연묵. (2019). 컨테이너 기반 클러스터 환경에서 효율적인 자원관리를 위한 오케스트레이션 방법. 한국차세대컴퓨팅학회 논문지, 15(2), 71-78.
- [3]박영기, 양현식, 김영한. (2019). 컨테이너 네트워킹 기술의 성능비교. 한국통신학회논문지, 44(1), 158-170, 10.7840/kics.2019.44.1.158
- [4]RedHat.What is Kubernetes?Retrievedfrom <https://www.redhat.com/ko/topics/containers/what>
- [5]kubernetes:<https://github.com/kubernetes/kubernetes>



2024.03.13

# 감사합니다



## [ 벤치마킹 세부사항\_컨테이너 관리 플랫폼(1) ]

	Amazon Elastic Container Service(Amazon ECS)	Amazon Elastic Kubernetes Service(Amazon EKS)	Azure Kubernetes Service(AKS)
정의	<ul style="list-style-type: none"> <li>- Docker 컨테이너 지원</li> <li>- Amazon EC2 인스턴스 클러스터에서 응용 프로그램 실행 가능한 컨테이너 관리 서비스</li> </ul>	쿠버네티스 응용 프로그램 및 관리 인프라를 실행하기 위한 컨테이너 오케스트레이션	<ul style="list-style-type: none"> <li>- 쿠버네티스 서비스를 사용하여 컨테이너화된 어플리케이션</li> <li>- 쉬운 배포/관리 가능</li> </ul>
장점	<ul style="list-style-type: none"> <li>- k8s보다 간단한 개념</li> <li>- AWS서비스에서의 통합 간편</li> </ul>	<ul style="list-style-type: none"> <li>- 계획적으로 애플리케이션을 격리함으로써 보안 성능을 향상</li> <li>- 쿠버네티스 Control Plane설치 및 운영 없이 AWS에서 관리 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 완성된 클러스터 제공으로 편리한 유지보수</li> <li>- Azure의 Master관리로 개발자의 부담 낮춤</li> <li>- Node Scaling의 자동화</li> </ul>
단점	<ul style="list-style-type: none"> <li>- 사용량에 따라 비용 발생</li> <li>- AWS 제품에 종속</li> </ul>	<ul style="list-style-type: none"> <li>- 사용량에 따라 비용 발생</li> <li>- AWS 제품에 종속</li> </ul>	<ul style="list-style-type: none"> <li>- 사용량에 따라 비용 발생</li> <li>- AWS 제품에 종속</li> </ul>

## [ 벤치마킹 세부사항\_컨테이너 관리 플랫폼(2) ]

	Rancher	Portainer CE	Harshicorp nomad
정의	<ul style="list-style-type: none"> <li>- 쿠버네티스 업그레이드, 백업 및 배포 가능</li> <li>- kubernetes 환경을 지원할 수 있는 멀티 클러스터 컨테이너 관리 플랫폼</li> </ul>	<p>쉽게 도커를 배포하고 구성할 때 사용하는 웹 UI 기반 관리 툴</p>	<p>클라우드 플랫폼에서 컨테이너 대규모 배포 및 관리 가능한 오케스트레이터</p>
장점	<ul style="list-style-type: none"> <li>- 자체적인 일반 계정 시스템 보유</li> <li>- 앱 카달로그 지원</li> <li>- 지표 기반의 모든 클러스터 관리 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 도커 환경을 GUI로 보여줌으로써 간편한 사용 가능</li> </ul>	<ul style="list-style-type: none"> <li>- 작업스케줄링에 중점</li> <li>- 구조적 단순함</li> </ul>
단점	<ul style="list-style-type: none"> <li>- 많은 서버 리소스 요구</li> <li>- 쿠버네티스 관리 중점</li> </ul>	<p>대규모 컨테이너 배포에 부적합</p>	<ul style="list-style-type: none"> <li>- 특정 기능이나 확장성 측면에서 제한적</li> <li>- 유료 구독 필요</li> </ul>