캡스톤디자인 I 계획서

<table>
<tr><td rowspan="2">제 목</td><td>국문</td><td colspan="4">백엔드 서비스 프로바이더:컨테이너 오케스트레이션 플랫폼 개발</td></tr>
<tr><td>영문</td><td colspan="4">Backend Service Providers: Container Orchestration Platform Development</td></tr>
<tr><td>프로젝트 목표
(500자
내외)</td><td colspan="5">본 캡스톤디자인은 컨테이너화된 애플리케이션의 배포, 관리 및 확장을 자동화하는 컨테이너 오케스트레이션 플랫폼 구축을 목표로 한다. 모니터링 서비스를 적용하여 관리하기 용이하며 특정 제품에 종속되지 않고 단순한 구조를 지녀 다양한 규모의 조직이 효율적인 관리 및 운영이 가능하도록 한다. 개발된 컨테이너 오케스트레이션 플랫폼은 멀티플레이어 게임 서버 구축으로 성능, 안정성, 확장성을 검증하도록 한다. 개발자들은 유연하고 경제적인 서버 운영이 가능해질 것이며 서버 배포 시간 단축으로 비용절감 및 시장 출시 기간을 줄일 수 있을 것이다. 또한 다양한 클라우드 환경에서 애플리케이션의 배포와 관리를 단순화하고 최적화 하여, 클라우드 기반 서비스 제공업체를 포함한 다양한 산업 분야의 활용이 가능할 것이다. 특히, 컨테이너 오케스트레이션 툴을 사용함으로써 복잡한 인프라 관리에 어려움을 사용에 겪는 기업이나 개발자에게 도움을 주는 것을 목표로 한다.</td></tr>
<tr><td>프로젝트
내용</td><td colspan="5">현대의 디지털 환경은 다양하고 복잡한 워크로드를 신속하고 효율적으로 처리할 수 있는 능력을 요구하며, 이에 따른 세계 컨테이너 오케스트레이션 시장 규모가 증가하고 있다. 대표적인 도구로 쿠버네티스가 존재하며 확장성, 유연성, 자동화의 특징이 있다. 하지만 구조가 복잡하고 높은 학습 곡선을 가져 소규모 프로젝트나 조직에 오버엔지니어링이 된다는 문제가 있다. 이에 본 컨테이너 오케스트레이션 플랫폼 개발은 실제로 필요한 기능과 팀의 기술적 역량을 고려하여 다양한 규모의 기업이나 개발자의 부담을 줄이고자 한다. 또한 직관적인 GUI기반 관리 도구 제공으로 생산성을 향상시키고 게임 서버 구축을 통한 플랫폼의 성능, 안정성, 확장성을 평가하고자 한다. Go 및 Rust프로그래밍 언어를 사용하여 개발될 예정이며, Linux 운영체제상에서 실행될 것이다.</td></tr>
<tr><td>중심어(국문)</td><td>쿠버네티스</td><td>컨테이너</td><td>오케스트레이션</td><td colspan="2">게임 서버</td></tr>
<tr><td>Keywords
(english)</td><td>Kubernetes</td><td>Container</td><td>Orchestration</td><td colspan="2">Game Server</td></tr>
<tr><td>멘토</td><td>소속</td><td colspan="2">ETRI/AI컴퓨팅시스템SW연구실</td><td>이름</td><td>강성주</td></tr>
<tr><td rowspan="4">팀
구성원</td><td>학년</td><td>학 번</td><td>이 름</td><td colspan="2">연락처(전화번호/이메일)</td></tr>
<tr><td>4</td><td>20211939</td><td>허유정</td><td colspan="2">01022352464/20211939@edu.hanbat.ac.kr</td></tr>
<tr><td>4</td><td>20217140</td><td>김창인</td><td colspan="2">01071561720/20217140@edu.hanbat.ac.kr</td></tr>
<tr><td></td><td></td><td></td><td colspan="2"></td></tr>
</table>

컴퓨터공학과의 캡스톤디자인 관리규정과 모든 지시사항을 준수하면서 본 캡스톤디자인을 성실히 수행하고자 아래와 같이 계획서를 제출합니다.

2024 년 03월 07일

책 임 자 : (인)

희망 지도교수 : 최창범 교수님

1. 캡스톤디자인의 배경 및 필요성

가. 컨테이너 기술의 등장

현대의 디지털 환경은 다양하고 복잡한 워크로드를 신속하고 효율적으로 처리할 수 있는 능력을 요구한다. 이에 애플리케이션을 빠르게 배포하고 확장할 수 있는 수단인 컨테이너 기술이 등장하였으며 애플리케이션과 의존성을 패키징하여 소프트웨어를 일관된 환경에서 실행할 수 있는 서버 관리의 효율성이 증대되었다.[1]

나. 컨테이너 오케스트레이션의 등장

서버 컨테이너 수가 증가함에 따라 대규모로 컨테이너를 배포하고 관리하는 과정에서 복잡성을 겪게 되었으며 이를 관리하기 위해 컨테이너 오케스트레이션 도구의 필요성이 대두되었다. 컨테이너 오케스트레이션은 컨테이너의 배포, 관리, 확장 등을 자동화함으로써, 복잡한 컨테이너 환경을 효율적으로 운영할 수 있다.

다. 컨테이너 오케스트레이션 도구의 현재 상황과 한계

쿠버네티스는 컨테이너 오케스트레이션의 대표적인 도구로 확장성, 유연성, 자동화의 장점을 지닌다. 언어나 프레임 워크에 구애받지 않아 개발자와 운영자의 작업 부담을 줄여주지만 구조가 복잡하며 높은 학습 곡선으로 소규모 프로젝트나 조직에 오버엔지니어링이 된다는 문제가 있다.

라. 컨테이너 오케스트레이션 플랫폼 개발의 필요성

본 컨테이너 오케스트레이션 플랫폼 개발은 실제로 필요한 기능과 팀의 기술적 역량을 고려하여 작은 규모의 프로젝트나 스타트업에게도 부담을 줄이고자 한다. 이는 서버 배포 시간 단축으로 비용 절감이 가능할 것이며 직관적인 GUI 기반 관리 도구 제공으로 개발자의 작업 부담을 줄이고, 생산성을 향상 시킬 수 있을 것이다. 또한 컨테이너화 및 오케스트레이션 기술 교육 자료로 활용되어, 학생들에게 현대적인 인프라 관리 기술을 가르치는 데 유용한 도구로 활용될 수 있다.

2. 캡스톤디자인 목표 및 비전

가. 컨테이너 오케스트레이션 플랫폼 개발의 목표

본 캡스톤디자인은 컨테이너화된 애플리케이션의 배포, 관리 및 확장을 자동화하는 컨테이너 오케스트레이션 플랫폼 구축을 목표로 한다. 모니터링 서비스를 적용하여 관리하기 용이하며 특정 제품에 종속되지 않는 플랫폼을 제작을 목표로 한다. 구조를 단순하게 하여 다양한 규모의 조직이 효율적인 관리 및 운영이 가능하도록 한다. 또한 게임 서버 구축으로 간편한 배포, 관리, 확장이 가능한지 실용성 여부를 확인해보고자 한다.

나. 기술적 검증 및 목표

개발된 컨테이너 오케스트레이션 플랫폼의 기능성을 검증하기 위해 실시간 멀티플레이어 게임 서버를 구축하여 테스트를 진행할 예정이다. 이를 통해 플랫폼의 성능, 안정성, 확

장성을 평가하고, 최종적으로 고성능, 고가용성을 보장하는 게임 서버 인프라를 구현할 목표를 가진다.

3. 캡스톤디자인 내용

가. 서론

	kubernetes	Docker Swarm	Apache Mesos
설명	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
장점	- 수천개 컨테이너 관리가 가능한 높은 확장성 - 자동 복구, 롤링 업데이트, 서비스 검색 가능	- 도커 생태계와의 높은 호환성과 통합성	- 수만개의 노드 관리가 가능한 높은 확장성 - 자원 분할과 공유로 효율적인 자원 사용 가능
단점	초기 설정과 관리가 복잡하여 학습 곡선이 높음	대규모 시스템에서의 제약	쿠버네티스보다 높은 학습 곡선

표 1 [벤치마킹] 컨테이너 오케스트레이션 플랫폼

현대의 디지털 환경에서는 복잡한 워크로드를 수용하고 관리 할 수 있는 능력을 요구하고 있으며, 이에 따른 세계 컨테이너 오케스트레이션 시장 규모가 증대되고 있다. 많은 클라우드 기반 오픈 소스 컨테이너 오케스트레이션 도구가 존재하며 애플리케이션의 확장성 및 기능을 향상시키는데 도움이 된다. 하지만 초기 설정과 관리가 복잡하여 학습 곡선이 높다는 문제가 있다.[2]

	Rancher	Azure Kubernetes Service(AKS)	Portainer CE	harshicorp nomad
설명	kubernetes환경을 지원하는 멀티 클러스터 컨테이너 관리 플랫폼	kubernetes 서비스를 사용해 완성된 클러스터 제공	도커를 배포하고 구성할 때 사용하는 웹 UI기반 툴	대규모 애플리케이션 배포 및 관리가 가능한 오케스트레이터
장점	계정 관리 시스템 모니터링 가능	Azure가 Master Node 관리를 하여 개발자 부담 최소화	편하고 직관적인 모니터링 클러스터 쉬운 배포 가능	가벼운 스케쥴러
단점	초기 설정에도00 많은 서버 리소스 요구	사용량에 따른 비용 발생	대규모 컨테이너 배포에 부적합	대부분의 기능 유료 사용

표 2 [벤치마킹] 컨테이너 관리 플랫폼

복잡한 학습곡선으로 컨테이너화된 응용 프로그램을 쉽게 실행할 수 있는 다양한 도구가 존재하며 Rancher, Portainer CE, AKS(Azure Kubernetes SErvice), harchicorp nomad를 그 예로 들 수 있다. 이러한 도구는 컨테이너 관리 플랫폼으로 컨테이너 오케스트레이션 플랫폼을 사용하기 위한 서비스로 볼 수 있다. 하지만 컨테이너 오케스트레이션 기반의 추가적인 기능을 제공하여 독립적이지 않으며 특정 제품에 종속하여 사용량에 따라 비용이 발생한다는 문제가 있다.

본 캡스톤디자인은 기존 도구의 문제 방안을 파악하기 위한 벤치마킹을 수행하였으며 특정 제품에 종속하지 않고 추가적인 비용 없이 사용가능한 컨테이너 오케스트레이션 플랫폼 자체 구축을 목표로 한다. 또한 다양한 규모의 조직이 효율적인 관리 및 운영이 가능하도록 컨테이너 오케스트레이션 필요 기능을 파악하고 개발 방법을 모색할 계획이다. 추가적으로 게임 서버 구축을 통해 컨테이너 오케스트레이션 실용성을 파악할 예정이다.

나. 본론

요구사항 명	요구사항 설명	요구사항 ID
컨테이너 런타임 관리	실행 중인 컨테이너에 대한 기본적인 관리 기능을 제공한다 (예: 시작, 정지, 재시작, 삭제)	REQ-001
스케일링	수동 또는 자동으로 컨테이너의 인스턴스 수를 조정한다	REQ-002
네트워킹	컨테이너 간 통신 및 외부 네트워크와의 연결을 관리할 수 있는 기능을 제공한다	REQ-003
사용자 인터페이스	사용자가 손쉽게 이미지를 빌드하고, 컨테이너를 실행하며, 시스템을 모니터링할 수 있는 직관적인 웹 기반 UI를 제공한다. 또한 이상 상태 감지 시 사용자에게 알림을 제공한다	REQ-004
테스트(개발)	게임 서버 구축을 통한 컨테이너 오케스트레이션 플랫폼 기능성 테스트를 진행한다	REQ-005

표 3 요구사항 정의서

본 캡스톤디자인에서는 쿠버네티스의 복잡성을 해결하고 보다 간편한 사용성을 제공하는 컨테이너 오케스트레이션 도구를 개발하고자 한다. 프로젝트는 Go 및 Rust 프로그래밍 언어를 사용하여 개발될 예정이며, Linux 운영체제 상에서 실행될 것이다. 이 플랫폼은 사용자 친화적인 웹 기반 사용자 인터페이스를 통해, 사용자가 효율적으로 컨테이너를 관리하고 모니터링 할 수 있도록 한다.

플랫폼의 핵심 기능인 컨테이너 런타임 관리, 스케일링, 네트워킹, 사용자 인터페이스 관리로 사용자는 컨테이너의 라이프사이클을 쉽게 제어하고, 필요에 따라 리소스를 동적

으로 조정 가능하다.[3] 또한, 복잡한 네트워크 설정과 통신 규칙을 간편하게 구성할 수 있는 기능을 제공함으로써, 컨테이너 간 및 외부 네트워크와의 연결을 용이하게 할 수 있다.

프로젝트의 중요한 부분 중 하나는 테스트 단계로, 개발된 컨테이너 오케스트레이션 플랫폼의 기능성을 검증하기 위해 실시간 멀티플레이어 게임 서버를 구축하고 테스트한다.

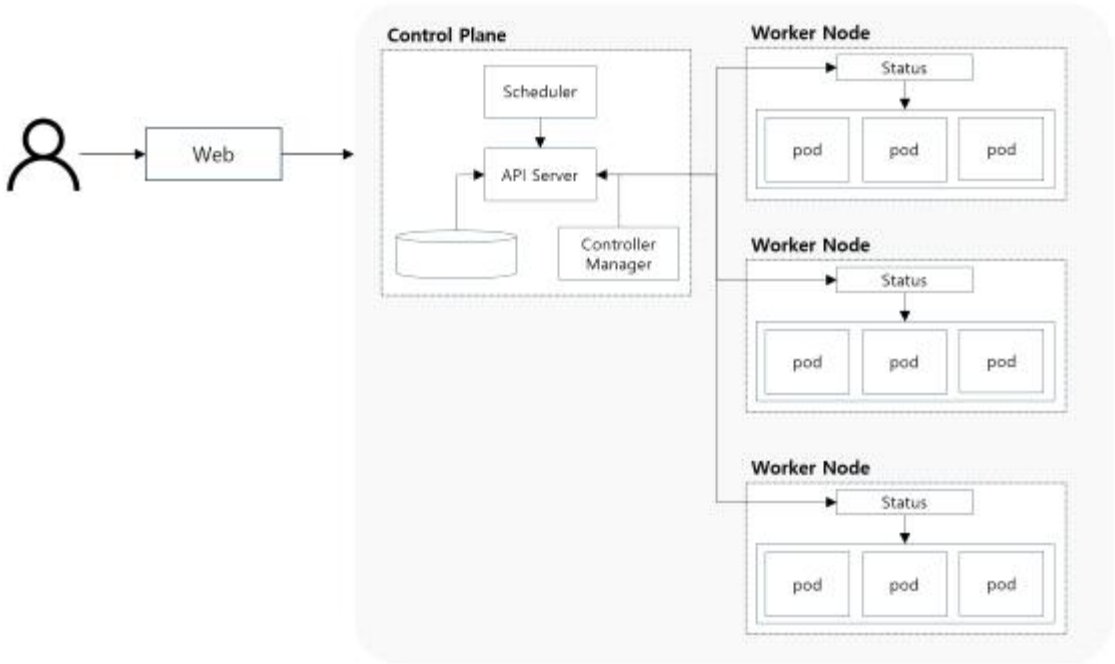

그림 1 플랫폼 도식화

1. 컨테이너 오케스트레이션 플랫폼 개발
- 서비스의 배포, 관리, 확장을 자동화
- Go 및 Rust 프로그래밍 언어 사용
- Linux 운영체제 사용

2. 테스트용 게임 서버 구축
- TCP, HTTP 프로토콜 사용
- Go 및 Rust 프로그래밍 언어 사용
- 관계형 데이터베이스 시스템 사용

2-1) Login Server
- 플레이어의 정보 저장 및 관리
2-2) Asset Server
- 플레이어가 제작한 Asset 정보 저장 및 관리
2-3) Scence Server(Room Server)
- 플레이어가 제작한 Scence 정보 저장 및 관리
2-4) Game Server
- 멀티플레이어 통신 관리

다. 결론

본 캡스톤디자인의 최종 결과물은 고성능, 고가용성을 보장하는 게임 서버 인프라와 사용자 친화적인 관리 플랫폼으로 소규모 개발 팀도 복잡한 서버 관리 작업 없이 게임 서비스를 원활하게 사용 가능할 것이라 기대한다.

4. 캡스톤디자인 추진전략 및 방법

가. 추진전략

1) 프로젝트 관리 및 협업 전략

가) 주기적인 스프린트 계획과 리뷰를 통한 진행 상황 모니터링
나) 소프트웨어 아키텍처 설계를 통한 시스템 이해도 확립
다) 애자일 개발 방법론 채택으로 프로젝트 유연성 확립
라) 디자인 패턴을 활용한 유지보수

2) 협업 도구의 활용

가) Github을 활용한 프로젝트 동시 개발
(1) 소스 코드와 변경 사항의 효과적인 추적 및 관리 가능
(2) 풀 리퀘스트 생성으로 변경 사항 검토 및 통합
(3) 개발자들 간 소스 코드 공유 및 병합 간편화
(4) 실시간 프로젝트 진행 상황 공유

나) Notion을 활용한 문서 협업
(1) 계획 공유를 통한 일정 관리
(2) 문서 공유의 불편함 최소화
(3) 작업의 투명성을 높여 팀원 간의 의사소통 강화

다) Microsoft Teams를 활용한 의사소통
(1) Github과의 연결을 통한 알림 활용
(2) 멘토와의 문서공유 및 의사소통 강화

3) 멘토 활용

멘토	최창범 교수님	강성주 책임 연구원
멘토 정보	한밭대학교 지도교수님	ETRI/AI컴퓨팅시스템SW연구실
멘토 활용	학문적 지도 및 이론에 대한 연구 방향 및 내용에 대한 조언	컨테이너 기술과 오케스트레이션 도구의 실무 적용에 대한 지식 및 경험

멘토진의 전문 지식과 경험을 활용하여 기술적 문제 해결, 효율적인 프로젝트 관리, 주기적인 피드백 및 평가, 그리고 신속한 시장 변화에 대응하는 능력에 도움을 얻고 프로젝트의 성공적인 완수를 목표로 한다.

4) 역할 분담

학생	허유정	김창인
역할	- Go언어 활용으로 컨테이너 오케스트레이션 플랫폼의 핵심 기능 개발 담당 - 클라이언트로부터의 요청 정보 처리 및 데이터 스키마 구성 - 게임 서버 구축에서의 비실시간 통신 관리	- Rust언어 활용으로 고성능 및 안정성이 요구되는 컨테이너 오케스트레이션 플랫폼 구성요소 개발 - 컨테이너 오케스트레이션 플랫폼 개발을 위한 리눅스 환경 구축 - 게임 서버 구축에서의 실시간 통신 관리

나. 수행 방법

(1) 벤치마킹
(2) 요구사항 정의서
(3) 시스템 아키텍처 설계
(4) 사용 언어 및 운영체제 확립
(5) 컨테이너 오케스트레이션 플랫폼 구축
  (가) 컨테이너 생성 및 런타임 관리
  (나) 무중단 배포 관리
  (다) 로드밸런싱 관리
  (라) 모니터링 관리
(6) 게임 서버 구축
  (가) Login Server
  (나) Asset Server
  (다) Room Server
  (라) Game Server
(7) 실증 및 테스트

5. 참고문헌

[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