캡스톤디자인Ⅱ 계획서

<table>
<tr><td rowspan="2">제 목</td><td>국문</td><td colspan="4">백엔드 서비스 프로바이더: 하향식 기반 방법론을 적용한 경량형 컨테이너 서비스 개발</td></tr>
<tr><td>영문</td><td colspan="4">Backend Service Providers: Container Runtime Environment and Orchestration Systems</td></tr>
<tr><td>프로젝트 목표
(500자 내외)</td><td colspan="5">본 캡스톤디자인2 프로젝트의 목표는 도커와 쿠버네티스와 같은 기존 컨테이너 관리 및 오케스트레이션 도구에 의존하지 않고, 직접 컨테이너 이미지 생성, 이미지 빌드, 런타임 환경 구축 및 오케스트레이션 기능을 구현하는 것이다. 이를 통해 컨테이너 기술의 핵심 원리를 이해하고, 사용자가 환경에 맞게 커스터마이징할 수 있는 유연한 컨테이너 환경을 제공을 목표로 한다. 이 프로젝트는 오픈 소스 기반으로 진행되며, 최적화된 성능과 자원 효율성을 달성하여 기업의 IT 비용 절감 및 다양한 산업 분야에서의 활용 가능성을 제시한다.</td></tr>
<tr><td>프로젝트 내용</td><td colspan="5">컨테이너 기술의 핵심 요소인 네임스페이스와 cgroups를 사용해서 구현하며 컨테이너 이미지 생성, 이미지 빌드, 런타임 환경 구축, 오케스트레이션 기능을 포함하는 시스템을 개발한다. 프로젝트는 크게 Carte Client와 Carte Daemon 두 부분으로 나뉘어 진행된다. Carte Client는 사용자 인터페이스와 명령어 입력, HTTP 요청 처리 등을 담당하며, Carte Daemon은 컨테이너의 생성, 실행, 중지, 삭제 등 관리 기능과 이미지 빌드, 오케스트레이션 기능을 수행한다. 설치 스크립트를 통해 시스템 설정 및 사용자 권한 관리를 자동화하며, YAML 설정 파일을 사용하여 컨테이너 배포 및 설정을 관리한다. TCP/IP 및 HTTP 기반의 통신 프로토콜을 사용하여 컨테이너 간 통신을 효율적으로 구현한다.</td></tr>
<tr><td>기대효과
(500자 이내)
(응용분야 및 활용범위)</td><td colspan="5">컨테이너 기술의 핵심 원리를 직접 구현함으로써 기술을 깊이 이해하고 학습할 수 있는 기회를 제공한다. 오픈 소스 기반의 프로젝트로써 다양한 개발자들이 참여하고 협력할 수 있는 환경을 제공하여 커뮤니티 발전에 기여할 수 있다. 또한 특정 애플리케이션 요구사항에 맞춘 최적화된 컨테이너 환경을 구축하여 성능과 자원 효율성을 높일 수 있을 것으로 기대한다. 경제적 이점으로, 상용 솔루션에 대한 의존도를 줄이고 자체 솔루션을 통해 다양한 산업 분야에서의 맞춤형 솔루션 개발에 활용될 수 있으며, 맞춤형 요구사항이 많은 환경에서 유용하게 사용될 수 있을 것이다.</td></tr>
<tr><td>중심어(국문)</td><td>컨테이너 런타임</td><td>컨테이너 오케스트레이션</td><td>네임스페이스</td><td>cgroups</td></tr>
<tr><td>Keywords (english)</td><td>Container Runtime</td><td>Container Orchestraion</td><td>Namespaces</td><td>cgroups</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/반</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 7 월 4 일

책 임 자 : 허유정 (인)

지도교수 : (인)

1. 캡스톤디자인의 배경 및 필요성

가. 컨테이너 기술의 배경

컨테이너 기술은 애플리케이션을 빠르게 배포하고 확장할 수 있어 현대 소프트웨어 개발과 배포의 핵심 요소로 자리 잡고 있다. 서버 컨테이너 수가 증가함에 따라 대규모로 컨테이너를 배포하고 관리하는 과정의 중요성이 증대되었으며 현대에서는 이러한 컨테이너 환경을 효율적으로 운영하고자 한다.

나. 컨테이너 기술의 국내·외 연구 및 산업 현황

다수의 연구가 컨테이너 기술의 성능 최적화, 보안 강화, 자원 관리 효율성 향상 등을 목표로 진행되고 있다. 리눅스 네임스페이스와 cgroups를 활용한 자원 격리 및 제한 기술은 활발히 연구되고 있으며 이를 사용한 도커와 쿠버네티스 사용이 보편화되었다.

이에 클라우드 네이티브 애플리케이션의 증가에 따라 AWS, Google Cloud, Microsoft Azure와 같은 주요 클라우드 제공업체는 컨테이너 관리 서비스를 제공하고 있다.

다. 컨테이너 기술의 전망

컨테이너 기술의 사용은 지속적으로 확대될 것으로 예상되며, 이를 지원하는 도구와 서비스도 더욱 다양해질 것이다. 하지만, 도커와 쿠버네티스와 같은 대중적인 도구 외에도, 보다 간단하고 유연한 컨테이너 관리 솔루션에 대한 수요가 계속 증가할 것이다.

라. 현존하는 기술의 문제점

쿠버네티스는 컨테이너 오케스트레이션의 대표적인 도구로 효율적인 운영과 배포를 지원한다. 하지만 소규모 프로젝트나 단순한 애플리케이션에는 과도한 복잡성을 나타내며 특정 도구와 플랫폼에 대한 의존성이 커지는 문제가 있다. 이는 유연성을 제한하고, 새로운 요구사항에 대한 대응을 어렵게 만든다. 이에 기존 도구는 사용자의 직접 수정 및 확장이 어려워 특정 요구를 반영하는 커스터마이징에 제한적이라는 문제를 가진다.

마. 프로젝트의 필요성

본 Carte 프로젝트는 사용자 요구에 맞춘 커스터마이징을 가능하게 하여 특정 요구사항에 대한 대응을 원활하게 진행하고자 한다. 실제로 필요한 기능과 팀의 기술적 역량을 고려하여 작은 규모의 프로젝트나 스타트업에게도 부담을 줄이고자 한다.

이는 기존의 컨테이너 오케스트레이션 도구의 한계를 극복할 수 있으며 사용자 정의가 가능한 유연한 컨테이너 환경 제공이 가능할 것이다.

바. 최적화 가능성과 경제적 이점

본 프로젝트를 통해 특정 애플리케이션 요구사항에 맞춘 최적화된 컨테이너 환경을 구축할 수 있다. 이는 성능 향상, 자원 효율성 증대 등의 이점을 제공할 수 있다. 또한 오픈 소스 기반의 자체 솔루션을 제공함으로써, 도커와 같은 상용 솔루션에 대한 의존도를 줄이고, 다양한 기업의 IT 비용을 절감할 수 있는 기회를 제공할 것이다.

2. 캡스톤디자인 목표 및 비전

가. 프로젝트 목표

Carte 프로젝트는 도커와 쿠버네티스의 복잡성과 의존성 문제를 해결하고자 한다. 현재의 컨테이너 오케스트레이션 도구들은 설정이 복잡하고 특정 플랫폼에 의존적이며, 유연한 커스터마이징이 어렵다는 문제가 있다. Carte는 이러한 문제를 해결하기 위해, 직접 컨테이너 환경을 구축하고 관리할 수 있는 도구를 제공한다.

나. 개발 목표

컨테이너 기술의 핵심 원리인 네임스페이스와 cgroups를 직접 구현함으로써, 기존 도구의 동작 원리를 이해하고 이를 바탕으로 더 나은 솔루션을 개발할 수 있다. 이를 통해 개발자로써 컨테이너 기술의 기초를 보다 깊이 있게 학습하고자 한다.

다. 프로젝트 비전

기존의 컨테이너 관리 및 오케스트레이션 원리를 재검토하여 효율적이고 유연한 접근 방식이 가능할 것이다. 컨테이너 이미지와 런타임 엔진의 경량화, 컨테이너 노드간 통신을 TCP/IP 또는 HTTP 기반의 효율적인 통신 프로토콜을 구현할 계획이다.

오픈 소스 기반의 자체 솔루션을 통해, 다양한 사용 사례에 맞춘 최적화된 컨테이너 환경을 구축할 수 있으며, 이는 차세대 컨테이너 기술의 발전에 기여할 것이다.

3. 캡스톤디자인 내용

가. 시스템 구성도

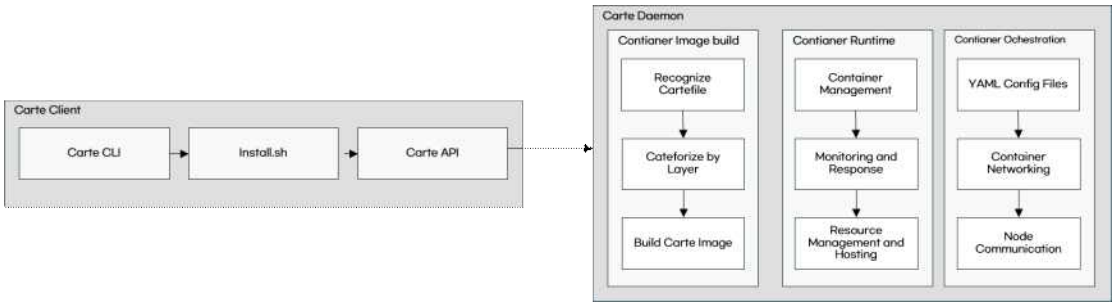

그림 2 시스템 구성도

그림1은 전체적인 프로젝트 구성도를 나타낸 것이며, Carte Client와 Carte Daemon에 대한 기능을 포함한다.Client에서 사용자 명령어를 통한 컨테이너 생성, 실행, 중지, 삭제, 통신을 하는 CLI를 제공하고 설치 스크립트로 시스템 설정, 권한 부여, 데몬 설치 및 실행 설정을 수행한다. HTTP 모듈을 통해 Daemon과 통신하며 응답 처리를 사용자에게 제공하며 Daemon에서 컨테이너 이미지빌드, 런타임, 오케스트레이션 기능을 수행할 수 있다.

나. 컨테이너간 통신

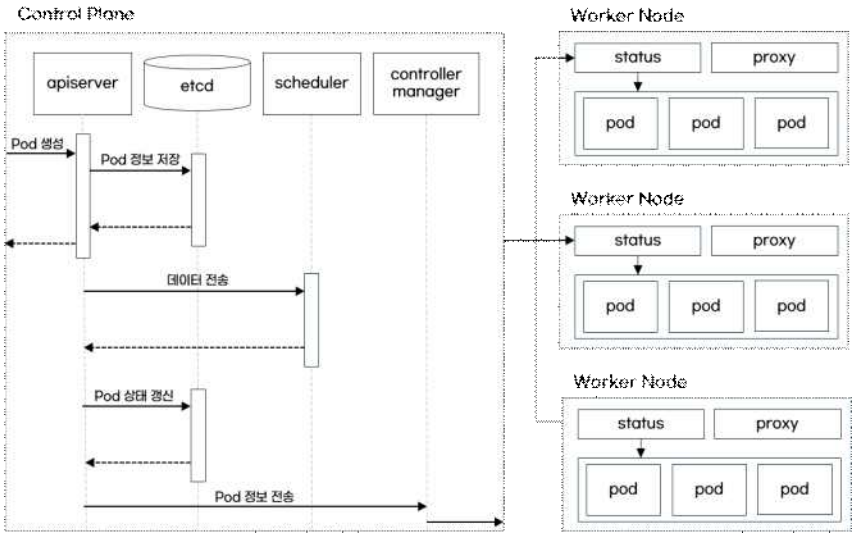

그림 3 컨테이너간 통신

그림2는 컨테이너간 통신에 대한 세부적으로 나타낸 것이며, 각 노드는 컨트롤 플레인과 통신하여 작업을 수행한다. YAML 설정 파일을 사용하여 컨테이너 배포 및 설정 관리가 가능하다. 컨트롤 플레인은 작업 지시 및 상태 모니터링을 수행하며, 노드는 작업을 실행하고 결과를 보고하는 것이다. 또한 각 Pod에 고유한 주소를 할당하여 통신 프로토콜을 규정한다.

다. 요구사항 명세서

기능 요구사항

요구사항 고유번호		SFR-001		
요구사항 명칭		컨테이너 이미지 빌드		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	애플리케이션을 컨테이너 이미지로 빌드하는 기능		
	세부내용	① 다양한 언어와 프레임워크로 작성된 애플리케이션을 컨테이너 이미지로 빌드할 수 있음 ② 애플리케이션 코드와 종속성을 포함하여 컨테이너 이미지 생성 ③ 이미지 버전 관리 및 레지스트리 저장 기능 ④ 레이어 계층을 분류하여 최종 빌드 이미지 생성 ⑤ 이미지 빌드 자동화 및 배포 파이프라인 연동		

요구사항 고유번호		SFR-002		
요구사항 명칭		컨테이너 런타임		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	실행 중인 컨테이너에 대한 기본적인 관리 기능		
	세부내용	① 컨테이너의 생성, 시작, 중식, 삭제의 런타임 관리 기능 ② 컨테이너 이상 상태 감지시, 적절한 조치 취할 수 있음 ③ 다양한 컨테이너 이미지 호스팅 및 관리 기능 ④ 컨테이너 실패할 경우 자동 재시작 기능 ⑤ 컨테이너 간의 리소스 관리와 격리 제공으로 성능, 안정성 부여		

요구사항 고유번호		SFR-003		
요구사항 명칭		컨테이너 오케스트레이션(네트워킹, 스케일링)		
요구사항 분류		기능	응락수준	필수
요구사항 상세설명	정의	컨테이너 배포, 관리, 확장을 위한 오케스트레이션 기능 제공		
	세부내용	① 컨테이너 네트워킹: 컨테이너 간 통신, 부하 분산, 서비스 디스커버리 ② 컨테이너 스케일링: 자동 스케일링, 수평/수직 확장 ③ 컨테이너 배포: 롤링 업데이트		

인터페이스 요구사항

요구사항 고유번호		SIR-001		
요구사항 명칭		CLI		
요구사항 분류		인터페이스	응락수준	필수
요구사항 상세설명	정의	사용자가 명령줄에서 컨테이너 관리 기능을 사용할 수 있는 CLI 제공		
	세부내용	① 컨테이너 이미지 생성 및 확인 등의 기본 명령 지원 ② 컨테이너 생성, 시작, 중지, 삭제 등의 기본 명령 지원 ③ 컨테이너 오케스트레이션 기능 CLI 인터페이스 제공 ④ 사용자 정의 스크립트 작성 및 실행 기능 지원		

보안 요구사항

요구사항 고유번호	SER-002		
요구사항 명칭	네트워크 보안		
요구사항 분류	보안	응락수준	선택
요구사항 세부내용	① 외부와의 네트워크 통신 보안 정책 관리 기능 ② 컨테이너 간의 통신을 안전하게 보호하기 위한 네트워크 보안 프로토콜 ③ 네트워크 트래픽 암호화 지원 ④ 네트워크 보안 정책 정의 및 적용할 수 있는 기능 제공을 통한 시스템 전체적인 보안 수준 유지 ⑤ 보안 감사를 통한 네트워크 사용에 대한 모든 활동 추적 및 보안 위협 탐지		

프로젝트 관리 요구사항

요구사항 고유번호	PMR-001		
요구사항 명칭	버전관리		
요구사항 분류	프로젝트 관리	응락수준	선택
요구사항 세부내용	- 소스코드 및 구성 파일의 버전 관리 기능 - 다중 브랜치 병합 기능을 통한 여러 개발자 동시 작업 가능 - 팀원 간의 협업을 위한 기능 포함으로 코드 공유 및 협업 작업에 용이 - 프로젝트 릴리스 및 배포를 위한 버전 태깅 및 릴리스 노트 생성 기능		

테스트 요구사항

요구사항 고유번호	TER-001		
요구사항 명칭	운영 테스트		
요구사항 분류	테스트	응락수준	필수
요구사항 세부내용	- 실제 구현한 메타버스 게임서버를 통해 운영 테스트를 진행 - 게임서버는 4개의 웹 서버와 4개의 디비로 구성 - 이미지 빌드 단계부터 배포까지 테스트		

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	- 이미지 레이어 계층 분류 - 이미지 생성, 빌드 - 컨테이너 스케일링 - 명령줄 인터페이스 구성	- 이미지 레이어 계층 분류 - 컨테이너 런타임 환경 구축 - 컨테이너 네트워킹 - 설치 스크립트 정의

5) 추진일정

요구사항	추진 일정	비고
컨테이너 이미지 빌드	6월, 7월	
컨테이너 런타임	6월, 7월, 8월	
컨테이너 오케스트레이션	8월, 9월, 10월	
CLI	6월, 7월, 8월, 9월, 10월	
네트워크 보안	미정	고려 기능 사항
운영 테스트	10월, 11월	

5. 캡스톤디자인 결과의 활용방안

가. 오픈 소스

오픈 소스 기반의 프로젝트로서, 다양한 개발자들이 참여하고 협력할 수 있는 환경을 제공한다. 이를 통해 지식 공유와 협업이 촉진되고, 커뮤니티 발전에 기여할 수 있다.

나. 기술적 효과

Carte는 기존의 도커와 쿠버네티스와는 다른 새로운 접근 방식을 통해 컨테이너 기술의 발전을 도모한다. 또한 특정 요구사항에 맞춘 최적화된 컨테이너 환경을 구축할 수 있어, 다양한 산업 분야에서의 활용 가능성이 높다.

다. 경제적 효과

간편한 설치와 유연한 관리 기능을 통해, 개발 및 운영의 생산성을 향상시킬 수 있다. 이는 프로젝트 일정 단축과 비용 절감으로 이어질 수 있다.

상용 솔루션에 대한 의존도를 줄이고, 기업의 IT 비용을 절감할 수 있다. 이는 특히 중소기업과 스타트업에게 경제적인 이점을 제공한다.

6. 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. 한국통신학회 논문지, 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