캡스톤 디자인 II 최종결과 보고서

프로젝트 제목(국문): 백엔드 서비스 프로바이더: 경량형 컨테이너 서비스 개발
프로젝트 제목(영문): Backend Service Provider: Lightweight Container Service Development

프로젝트 팀(원): 학번: 20211939 이름: 허유정
프로젝트 팀(원): 학번: 20217140 이름: 김창인

1. 중간보고서의 검토결과 심사위원의 '수정 및 개선 의견'과 그러한 검토의견을 반영하여 개선한 부분을 명시하시오.

- 학부생들에게 현대적인 인프라 관리 기술을 이해하기 위한 플랫폼으로 개발되었으면 좋을 것 같다. 실습 중심의 학습 환경 제공을 통해 학생들이 컨테이너 생성, 실행, 삭제 등의 과정을 직접 체험할 수 있도록 하여 실질적인 기술 이해에 사용되면 좋을 것 같다.

2. 기능, 성능 및 품질 요구사항을 충족하기 위해 본 개발 프로젝트에서 적용한 주요 알고리즘, 설계방법 등을 기술하시오.

본 캡스톤 디자인은 컨테이너 관리 시스템의 안정성과 효율성을 보장하기 위해 컨테이너 생애주기 관리와 자원 해제 및 복구 절차를 중심으로 설계를 진행함

컨테이너 생성 시에는 Linux 네임스페이스와 cgroups를 활용해 네트워크와 프로세스를 독립적으로 격리하고, chroot와 마운트 작업을 통해 컨테이너 내부 환경을 구성하였음. 또한, Veth(Virtual Ethernet Pair)를 생성하여 컨테이너와 호스트 간의 네트워크 연결을 제공함. 컨테이너 종료 시에는 PID 파일, 네트워크 인터페이스(veth), 마운트된 파일 시스템(/proc, /sys. /dev)를 자동으로 해제하는 과정을 구현하였으며, 이 과정에서 오류 발생 시 재시도 로직을 추가하여 시스템의 신뢰성을 강화함. 컨테이너 삭제 단계에서는 잔여 자원 및 데이터를 확인하여 시스템 상태를 정리함

이러한 설계와 구현은 컨테이너 관리 시스템이 다양한 환경에서 안정적으로 동작하도록 지원하며, 특히 자원 효율성과 시스템 안정성을 동시에 확보하는 데 중점을 둠

3. 요구사항 정의서에 명세된 기능 및 품질 요구사항에 대하여 최종 완료된 결과를 기술하시오.

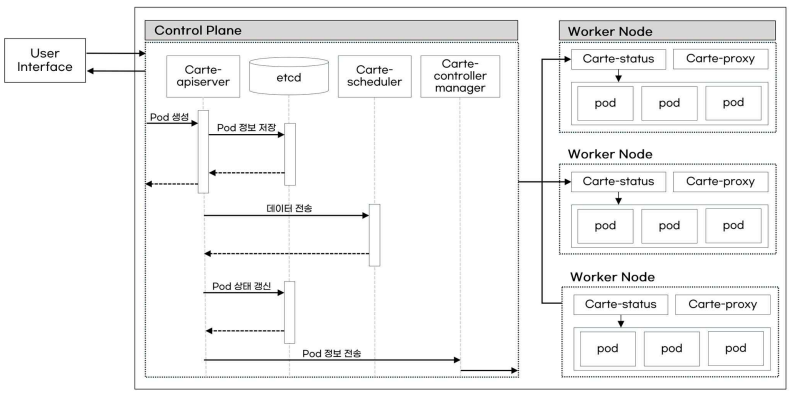


그림 1 시스템 구성도

Control Plane

- 컨테이너의 생성, 실행, 종료와 같은 작업 명령을 Worker Node에 전달하며, 컨테이너 상태를 모니터링함. 명령어 실행 결과를 확인하고, 로그를 기록하여 시스템의 전반적인 동작 상태를 파악하는 역할을 수행함.

Worker Node

- 컨테이너 종료 시 자원 해제를 수행함. 컨테이너의 PID 파일 삭제, 네트워크 인터페이스(veth) 해제, 마운트된 파일 시스템(/proc, /sys, /dev) 해제를 자동으로 처리하여 자원 누수를 방지함.
- 자원 해제 작업 중 발생하는 오류를 기록하고, 재시도 로직을 통해 안정적으로 복구를 시도함. 이 과정에서 PID 파일, 네트워크 인터페이스 등이 정상적으로 삭제되지 않았을 경우 이를 확인하고 반복적으로 처리함.
- 컨테이너 생성 시 네임스페이스와 cgroups를 설정하여 컨테이너의 독립적인 환경을 제공함. 네트워크 인터페이스(veth)를 생성하여 호스트와의 연결을 지원하며, 컨테이너 내에서 격리된 실행 환경을 유지하도록 보장함.

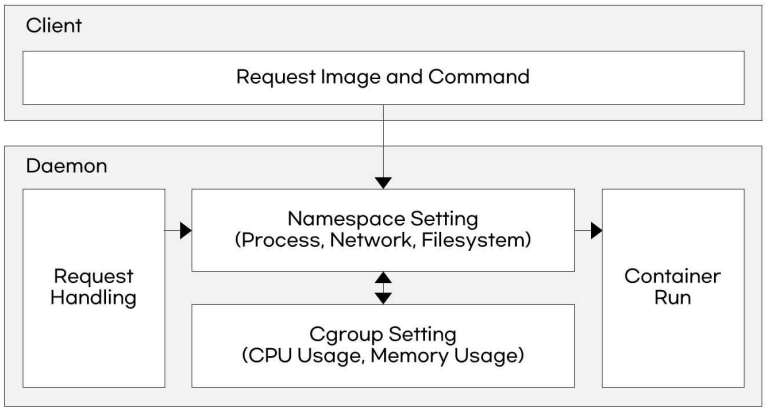


그림 2 Carte Engine

Namespace 설정을 통한 프로세스 격리

- Namespace를 활용하여 컨테이너별로 격리된 프로세스와 네트워크 환경을 제공함. 이를 통해 하나의 컨테이너에서 실행되는 프로세스가 다른 컨테이너 또는 호스트 시스템의 프로세스와 간섭하지 않도록 설정함. 컨테이너 실행 시 CLONE_NEWNS, CLONE_NEWPID, CLONE_NEWNET 등의 플래그를 적용하여 프로세스, 네트워크, 마운트 포인트의 독립성을 보장함.

Cgroup 설정을 통한 리소스 제한

- Cgroups(Control Groups)를 활용하여 CPU와 메모리와 같은 시스템 자원을 각 컨테이너별로 제한함. 이를 통해 특정 컨테이너가 과도하게 자원을 사용하는 상황을 방지하고, 다수의 컨테이너가 동시에 실행될 때 안정적인 자원 분배를 보장함.
- CPU의 경우, cpu.cfs_quota_us를 설정하여 일정 이상의 CPU 사용을 제한함. 메모리는 memory.limit_in_bytes를 활용하여 컨테이너가 사용할 수 있는 메모리 용량을 설정함. 이를 통해 시스템의 자원이 효율적으로 활용되도록 관리

4. 구현하지 못한 기능 요구사항이 있다면 그 이유와 해결방안을 기술하시오,

(작성요령: 전부 구현한 경우는 “이유”란에 “해당사항 없음”이라고 기재하고, 만약 요구사항대비 구현하지 못한 기능이 있다면 “이유”란에 그 사유를 기재함)

최초 요구사항	구현 여부(미구현, 수정, 삭제 등)	이유(일정부족, 프로젝트 관리미비, 팀원변동, 기술적 문제 등)
오토스케일링	미구현	주제 변경

5. 요구사항을 충족시키지 못한 성능, 품질 요구사항이 있다면 그 이유와 해결방안을 기술하시오.

분류(성능, 속도 등) 및 최초 요구사항	충족 여부(현재 측정결과 제시)	이유(일정부족, 프로젝트 관리미비, 팀원변동, 기술적 문제 등)
		해당사항 없음

6. 최종 완성된 프로젝트 결과물(소프트웨어, 하드웨어 등)을 설치하여 사용하기 위한 사용자 매뉴얼을 작성하시오.

git을 다운 받은 후, 해당 폴더에서 Carte 명령어를 순차적으로 진행한다.
(우선적으로, Go언어 1.21 version을 설치해주어야한다)

1. 이미지 및 컨테이너 생성

```
./carte create ./image.sh/test.sh
```

2. Carte 명령 수행
- 지정된 이미지 사용과 컨테이너 실행 명령을 통한 컨테이너 독립적 동작
- Container Life Cycle을 기준으로 생성하였으며 기존 기술보다 명령어를 간소화시켜 사용자 입장에서 유용한 사용을 하도록 구성함

명령어	정의
Carte create	새로운 컨테이너 생성
Carte start	지정된 이미지 기반으로 컨테이너 실행
Carte stop	실행중인 컨테이너를 정지
Carte remove	정지된 컨테이너 삭제
Carte list_i	사용 가능한 이미지 확인
Carte list_c	실행중인 컨테이너 확인

표 1 Carte 명령어

- 컨테이너 실행

```
./carte start testContainer
```

- 컨테이너 정지

```
./carte stop testContainer
```

- 이미지 리스트 확인

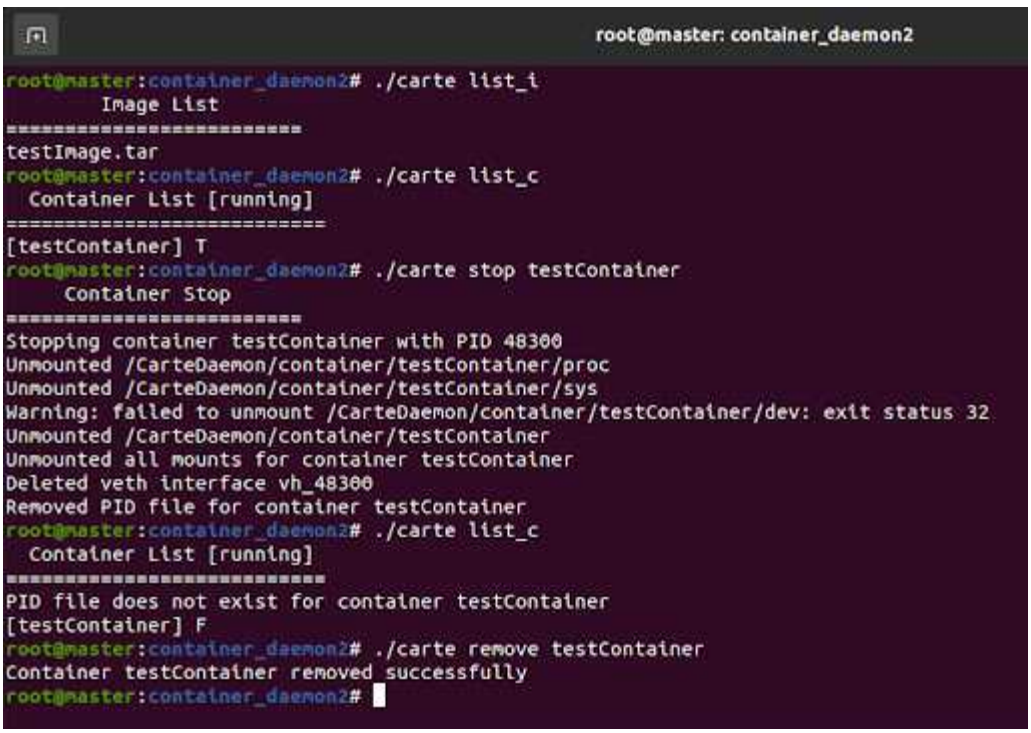
```
./carte list_i
```

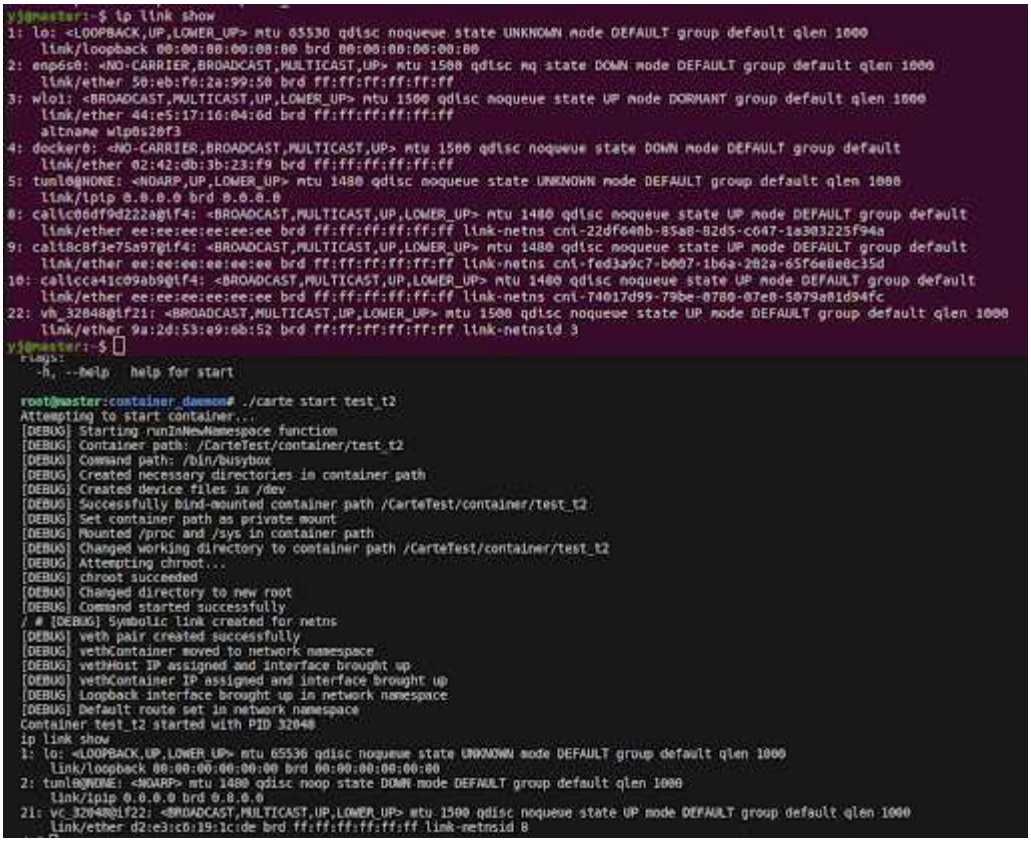
- 컨테이너 리스트 확인(실행여부 포함)

```
./carte list_c
```

- 컨테이너 삭제

```
./carte remove testContainer
```





7. 캡스톤디자인 결과의 활용방안

1) 클라우드 애플리케이션 배포 및 관리

다양한 클라우드 환경에서 애플리케이션의 배포와 관리를 단순화하고 최적화하여, 클라우드 기반 서비스 제공업체뿐만 아니라 다양한 산업 분야에서 광범위하게 활용될 수 있다.

2) 교육 분야에서의 활용

컨테이너화 및 오케스트레이션 기술 교육 자료로 활용될 수 있으며, 실습 중심의 학습을 통해 학생들에게 현대적인 인프라 관리 기술을 가르치는 데 유용한 도구가 될 수 있다,

본 과제의 결과물은 기술적, 경제적 가치를 넘어 사회적 영향력을 발휘할 수 있는 다양한 활용 가능성을 내포하고 있다. 특히, 컨테이너 오케스트레이션 툴을 통해 복잡한 인프라 관리의 문제를 해결하고자 하는 기업이나 개발자들에게 큰 도움을 줄 것으로 기대된다.