2022 년 전기 졸업과제 착수보고서

오픈소스 클라우드 플랫폼을 활용한 사용자 맞춤형 가상머신 관리 시스템

팀명 : 뜬구름

201624540 이봉훈

201624406 고준성

201624444 김영후

지도교수 : 염근혁 (인)

목차

1. 과제의 목표 3

1) 과제 배경 3

2) 과제 세부 목표 3

2. 대상 문제 및 요구사항 분석 4

1) 유사 시스템 분석 4

2) 문제점 분석 5

3) 요구사항 분석 7

4) 시스템 구성도 및 흐름도 9

3. 현실적 제약 사항 분석 결과 및 대안 13

1) 현실적 제약 사항 13

2) 대안 13

4. 개발 환경 및 사용 기술 14

1) 개발 환경 14

2) 사용 기술 14

III. Cinder 15

5. 개발 일정 및 역할 분담 20

1) 개발 일정 20

2) 역할 분담 20

2022 전기 졸업과제

1. 과제의 목표

1) 과제 배경

최근 언택트 시대가 도래하면서 물리적 인프라를 직접 구축하지 않고도 컴퓨팅 자원에 접근할 수 있게 하는 클라우드 컴퓨팅이 각광받고 있다. 이러한 시대에 발맞춰 많은 클라우드 벤더들은 사용자에게 여러 유형의 클라우드 기반 서비스를 제공하고 있고, 클라우드 사용자들은 각자 다양한 목적과 용도로 이러한 서비스를 사용하고 있다. 그 중에서도 클라우드를 통해 사용자 각자의 요구에 알맞은 IT 인프라 자원을 이용하려는 사용자가 늘고 있다.

하지만 사용자가 클라우드 컴퓨팅을 활용하기 위해서는 관련 용어 및 가상머신에 대한 전문적인 지식이 요구되므로 클라우드 관련 전문 지식이 부족한 사용자는 가상머신에 대한 설정을 잘 다루지 못하는 문제가 생긴다. 또한 사용자의 요구에 부합하는 가상머신을 생성하기 위해서 하드웨어, 소프트웨어 등에 대한 요구사항을 받아들이고 처리할 수 있는 구조도 지원되어야 한다.

이와 더불어 클라우드 인프라 구축 후에 발생하는 관리적 측면의 어려움이 있다. 사용자의 클라우드 자원 활용에 따른 변경이 있을 경우 클라우드 자원의 활용도가 달라지게 되므로, 이를 처리하는 방안과 구조가 필요하다. 또한 갑작스러운 정전이나 재해 등과 같은 외부 요인 혹은 시스템 자체 결함 등의 내부 요인으로 클라우드 시스템에 장애가 생기는 경우, 클라우드 사용자가 장애발생에 유연하게 대처하는 것도 어려운 점 중 하나이다.

따라서 본 과제는 사용자의 요구사항을 입력 받아 클라우드 가상머신 생성 시 필요한 설정, 요구사항의 변경이 있을 시 가상머신의 업데이트, 재해발생 시 가상머신의 재해 복구를 지원하는 통합적인 가상 머신 관리 시스템 구현을 목표로 한다.

2) 과제 세부 목표

① 사용자 하드웨어, 소프트웨어 요구사항을 반영한 가상 머신 생성

② 가상머신의 하드웨어, 소프트웨어 요구사항 변경에 따른 맞춤형 가상 머신 관리

③ 재해복구(Disaster Recovery)기능을 갖춘 동기화된 멀티 클라우드 환경 구성

2. 대상 문제 및 요구사항 분석

1) 유사 시스템 분석

① AWS(Amazon Web Services) Elastic Beanstalk

AWS Elastic Beanstalk 는 Java, .NET, PHP, Node.js, Python, Ruby, Go 및 Docker 를 사용하여 개발된 웹 애플리케이션 및 서비스를 Apache, Nginx, Passenger 및 IIS 와 같은 친숙한 서버에서 손쉽게 배포하고 확장할 수 있는 서비스이다. 인프라를 구성, 운영하고 애플리케이션 스택을 관리해주어 사용자가 관리에 시간을 들일 필요가 없어 생산성이 높다. 또한 간단한 오토 스케일링 설정 조정을 통해 적절한 규모 유지가 가능하며, 리소스 제어가 간단하다는 특징이 있다.

코드를 업로드하기만 하면 Elastic Beanstalk 가 용량 프로비저닝, 로드 밸런싱, Auto Scaling 부터 시작하여 애플리케이션 상태 모니터링에 이르기까지 배포를 자동으로 처리한다. 애플리케이션을 실행하는 데 필요한 AWS 리소스를 완벽하게 제어할 수 있으며 언제든지 기본 리소스에 액세스할 수 있다. 또한 특정 프로그래밍 언어, 프레임워크 및 웹 컨테이너용으로 개발된 웹 애플리케이션 실행을 지원하는 관리형 플랫폼을 제공한다. 각 플랫폼에 대해 하나 이상의 플랫폼 버전을 제공하고, 환경을 생성하는 플랫폼을 선택하면 하나 이상의 Amazon EC2(Elastic Compute Cloud) 인스턴스를 포함하여 애플리케이션에 필요한 리소스를 프로비저닝 한다.

② Amazon EBS(Elastic Block Store)

Amazon EBS 는 사용이 쉽고 확장 가능한 고성능 블록 스토리지 서비스로서 Amazon EC2 용으로 설계되었다. SAP, Oracle 및 Microsoft 제품과 같은 미션 크리티컬 애플리케이션을 포함하여, 가장 까다로운 고성능 워크로드에 맞춰 빠르게 확장할 수 있다. 가용 영역 내 복제를 포함한 99.999%의 가용성과 io2 Block Express 볼륨의 99.999% 내구성을 통해 장애로부터 보호할 수 있다. 또한 요금이 경제적인 볼륨부터 가장 빠른 IOPS 및 처리량을 제공하는 고성능 볼륨 중 선택하는 등 워크로드에 적합한 선택이 가능하다. 스냅샷 기능을 제공해서 EBS 의 현재 상태를 그대로 보존할 수 있으며, CloudWatch 를 통한 EBS 의 통계 열람이 가능하다. 또한 EBS 는 독립적이기 때문에 EC2 인스턴스를 제거해도 데이터가 유지된다.

2) 문제점 분석

① 현 유사 시스템의 문제점

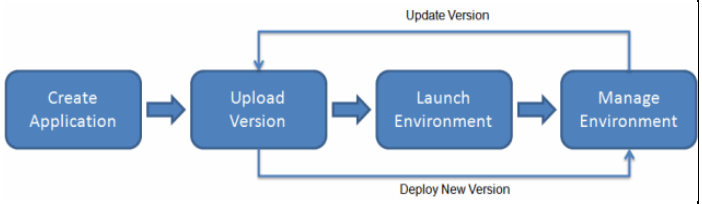


그림 1. AWS Elastic Beanstalk workflow

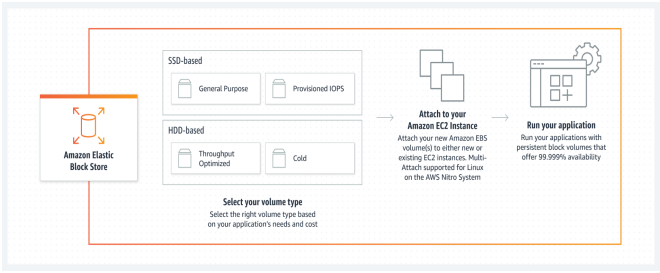


그림 2. Amazon Elastic Block Store 의 작동 방식

AWS 의 다양한 서비스들 중에서 AWS Elastic Beanstalk 는 애플리케이션 배포 시 필요한 인스턴스 생성에 맞춘 자동 프로비저닝을 제공하기에, 초반 인스턴스 생성 시 필요한 최소한의 정보에 배포할 애플리케이션의 소스코드가 포함되어야 한다는 제약사항이 존재한다.

그와 더불어 초반 beanstalk 환경 구축 시 load balancer 설정 등을 위한 AWS 시스템에 대한 전문지식이 필요하므로, 이러한 지식이 부족한 경우의 사용자는 이용에 제약이 있는 등의 문제점이 있다.

Amazon Elastic Block Store 역시 데이터 복원 등을 위해 사용을 시작할 때 설정 값을 직접 입력해줘야 한다는 제약이 존재한다.

또한 두 시스템 모두 사용자가 특정 클라우드 벤더에 종속되는 클라우드 벤더 락인(Lock-in) 문제를 안고 있다.

② 시스템의 필요성

I. 전문지식이 부족한 사용자도 쉽게 클라우드 가상머신을 구축할 수 있는 시스템이 필요하다.

- 예상되는 사용자의 요구사항을 카테고리로 분류하여 일반적인 사용자의 접근성을 높임
- 사용자가 선택한 요구사항을 기반으로 맞춤형 가상머신 생성

II. 가상머신에 대한 하드웨어, 소프트웨어 요구사항 변경 발생 시 그에 맞춘 가상머신의 변경을 지원할 필요가 있다.

- 사용자의 요구사항 변경 요청에 맞는 가상환경 변경 기능 개발

III. 재해발생(정전, 시스템 에러 등) 시에도 지속적인 가상 환경을 제공할 필요가 있다.

- 재해발생에 대비한 멀티 클라우드 플랫폼 상에서의 주기적 백업, 동기화 기능 개발
- 재해발생 시 사용가능한 클라우드 플랫폼으로 전환하여 사용자의 가상머신 이용 불가 시간 단축

3) 요구사항 분석

① 기능적 요구사항

표 1 은 시스템이 제공해야 할 기능들에 대한 요구사항을 나타낸다.

표 1. 기능적 요구사항

기능		설명
사용자 관리	시스템 접근 제어	- 사용자의 시스템 접근 및 사용자 등록을 수행하는 로그인, 로그아웃, 회원가입 기능
	사용자 가상환경 정보 출력	- 사용자가 생성한 가상머신들의 기본 정보(가상머신 이름, 가상머신 UUID, 용량, Power State)를 출력
	사용자 요구사항 입력	- 맞춤형 가상머신 구축이 가능하도록 미리 생성된 요구사항 카테고리로 하드웨어, 소프트웨어 등의 사용자 요구사항을 반영
사용자 요구사항 맞춤형 가상 머신 생성	맞춤형 Orchestration Template 생성	- 입력 받은 요구사항을 기반으로 Orchestration 기능을 수행할 수 있는 Template 을 생성
	Template 기반 가상머신 생성	- 생성된 맞춤형 Template 을 입력 값으로 클라우드 플랫폼별 Orchestration 컴포넌트를 이용하여 가상머신을 구축 - 클라우드 플랫폼마다 동일한 가상머신 생성(메인 가상머신과 백업용 가상머신으로 구성)
사용자 요구사항 변경 반영	가상머신 환경 업데이트	- 사용자의 요구사항 변경 요청 시 요구사항에 맞게 기존의 가상머신 환경에 대한 업데이트 실행
재해 복구 관리	백업 이미지 주기적 생성	- 메인 가상머신이 위치한 클라우드 플랫폼의 백업 컴포넌트를 이용하여 주기적으로 스냅샷을 이미지 형태로 저장
	백업용 가상머신 업데이트	- 생성된 백업 이미지를 기반으로 백업용 가상머신에 업데이트
	재해 알림	- 재해 발생 시 재해복구 시스템이 동작할 수 있도록 이벤트 알림
	알림 발생시 가상머신 전환	- 재해 발생을 감지하는 재해 알림 이벤트 발생 시 백업용 가상머신으로 사용 환경 전환

② 비기능적 요구사항

다음 표 2 는 시스템이 만족해야 할 비기능적 요구사항이다.

표 2. 비기능적 요구사항

요건	설명
운영	요구사항 반영시스템, 클라우드 환경 관리 시스템은 24 시간, 365 일 내내 사용 가능해야 하고 기준 시간에 맞게 주기적으로 백업용 이미지를 생성해야 함 (무정지 운영)
사용 편의성	해당 시스템을 이용하는 모든 사용자가 서비스를 이용할 수 있도록 UI 를 배치해야 함. 또한 클라우드 관리 비전문가도 클라우드 관리가 가능하도록 코드 단위의 접근이 필요하지 않도록 구현해야 함
이식성	멀티 클라우드 플랫폼 상에서 가상환경 관리를 지원하도록 해야함
보안성	등록된 사용자만이 시스템에 접근할 수 있어야 함
	사용자의 정보 및 가상환경 정보를 다른 사용자가 조회하거나 변경할 수 없도록 해야 함
상호 운용성	클라우드 리소스를 이용하기 위해 요구사항 반영 시스템과 클라우드 플랫폼이 연동되어 정보를 교환해야 함
성능	관리자가 사용자의 요구사항을 입력 받아 가상환경을 생성, 변경하기까지 1 분 이내의 시간이 소요되어야 함

4) 시스템 구성도 및 흐름도

① 시스템 구성도

그림 3 은 본 시스템의 전체 구성을 나타낸 그림이다. 구성 시스템은 아래와 같다.

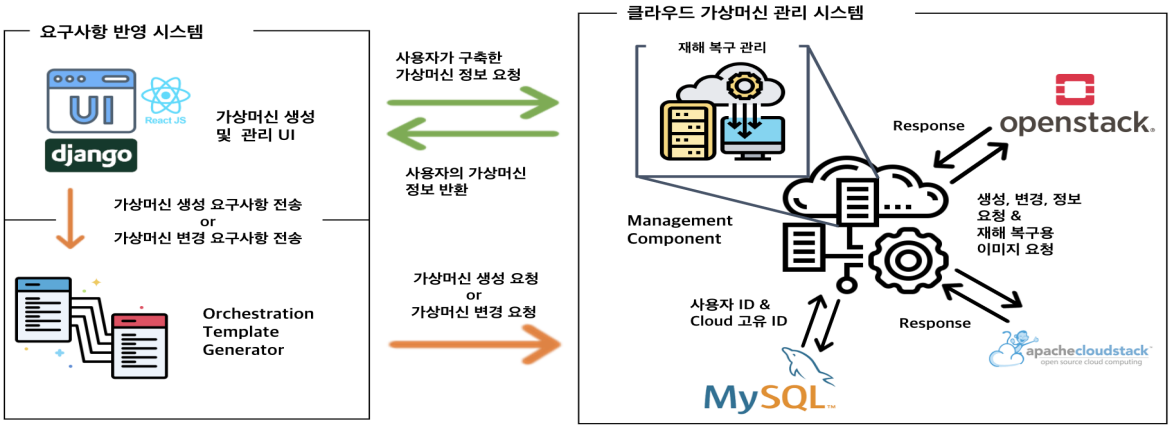


그림 3. 시스템 구성도

I. 요구사항 반영 시스템

요구사항 반영 시스템은 사용자의 가상머신 정보, 생성, 변경 요청을 입력 받아 클라우드 관리 시스템에 가상머신 정보에 대한 요청, 요구사항에 맞는 템플릿 생성을 수행한다. 요구사항 반영 시스템은 가상머신 생성 및 관리 UI 와 Orchestration Template Generator 로 이루어져 있다.

- 가상머신 생성 및 관리 UI 는 사용자의 요청을 입력 받고 클라우드 가상머신 관리 시스템 혹은 Orchestration Template Generator 는에 기능 수행을 요청하여 반환 값을 출력한다.
- Orchestration Template Generator 는 가상머신 생성 및 관리 UI 에게 사용자의 요구사항을 입력으로 받아온다. 입력된 요구사항을 바탕으로 클라우드 플랫폼이 요구사항에 맞는 가상머신을 생성할 수 있도록 IaC(Infrastructure as Code) 코드를 작성한다.

II. 클라우드 가상머신 관리 시스템

요구사항 반영 시스템에서 받은 요청을 수행하고 생성된 가상머신의 재해복구를 담당한다. 구성요소로 Management Component 와 클라우드 플랫폼, Database 가 있다.

- Management Component 는 시스템 구성요소들 중간에서 요청을 보내고 반환 값을 받아 요구사항 반영 시스템으로 전송하는 역할을 수행하며 재해복구 정책에 따라 클라우드 플랫폼에 요청한다.
- 클라우드 플랫폼은 사용자가 주로 사용하는 메인 클라우드와 재해 발생시 사용하는 백업용 클라우드로 구분된다. Management Component 의 요청을 받아 가상머신 생성 및 변경을 수행한다.
- Database 는 가상머신 사용자 ID, 가상머신 ID 정보와 백업용 이미지를 매핑하여 저장한다.

② 시스템 흐름도

I. 가상머신 정보 요청, 반환 흐름도

그림 4 는 사용자가 사용 중인 가상머신의 정보에 대한 요청을 했을 때 시스템의 흐름을 나타낸 것이다.

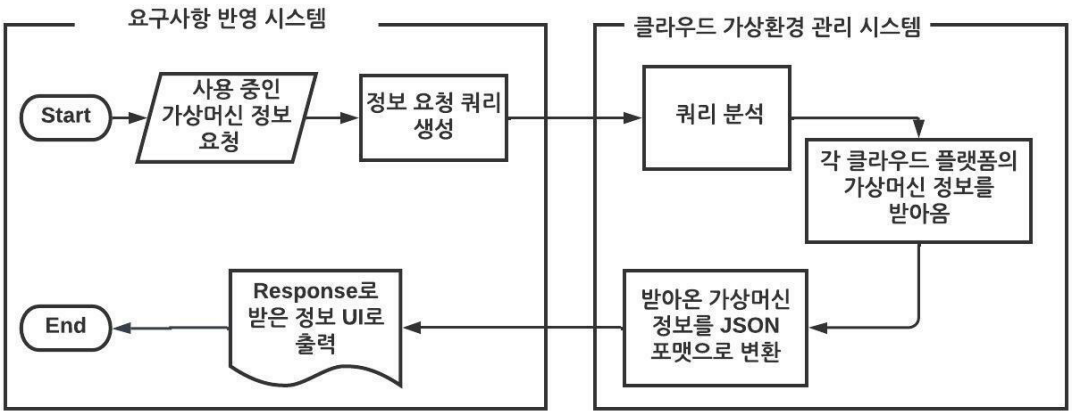


그림 4. 가상머신 정보 요청, 반환 흐름도

- 요구사항 반영 시스템의 UI 를 통해 사용자가 가상머신 정보에 대한 요청을 한다.
- 요구사항 반영 시스템은 요청을 받고 클라우드 가상환경 관리 시스템에게 보낼 정보 요청 쿼리를 생성한다.
- 클라우드 가상환경 관리 시스템은 받은 쿼리를 해석한 후, 각 클라우드 플랫폼의 가상머신에게서 정보를 받아온다.
- 받아온 정보를 JSON 포맷으로 변환 후 요구사항 반영 시스템에 보내주면 그 정보를 UI 에 맞게 매핑한 후 사용자에게 출력한다.

II. 가상머신 생성, 변경 요청 흐름도

그림 5 는 시스템 사용자가 새로운 가상머신 생성을 요청하거나 기존 가상머신의 요구사항 변경을 요청했을 때 수행되는 흐름도를 나타낸 것이다.

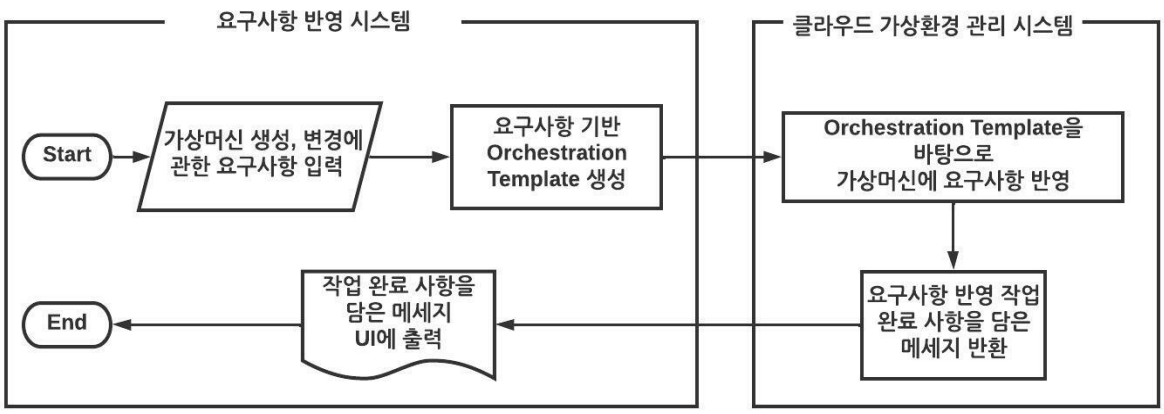


그림 5. 가상머신 생성, 변경 요청 흐름도

- 처음 가상머신 생성 또는 변경 요청을 받으면 요구사항 반영 시스템에서 사용자가 원하는 가상머신 하드웨어, 소프트웨어 요구사항을 카테고리 형식으로 입력 받는다.
- 입력된 요구사항을 바탕으로 Orchestration Template 생성 컴포넌트에서 맞춤형 Template 를 생성하고 가상머신의 생성 또는 변경을 적용하기 위해 클라우드 가상환경 관리 시스템과 통신한다.
- 클라우드 가상환경 관리 시스템은 요구사항 반영 시스템에서 받은 Template 을 입력 값으로 각 클라우드 플랫폼에게 가상머신 생성 또는 변경을 수행하도록 한다.
- 작업이 완료되면 각 클라우드 플랫폼의 수행 결과 메세지를 받아 요구사항 반영 시스템으로 반환한다.
- 요구사항 반영 시스템은 수행결과 메시지를 사용자가 확인할 수 있도록 UI 포맷으로 매핑하여 출력한다.

III. 재해 복구 management 흐름도

그림 6 은 본 시스템의 재해 복구 관리 기능을 나타낸 것이다. 이 기능은 사용자가 생성한 각 클라우드 플랫폼의 가상머신에 대한 재해 복구는 주기적인 요청과 응답을 통해 이루어진다.

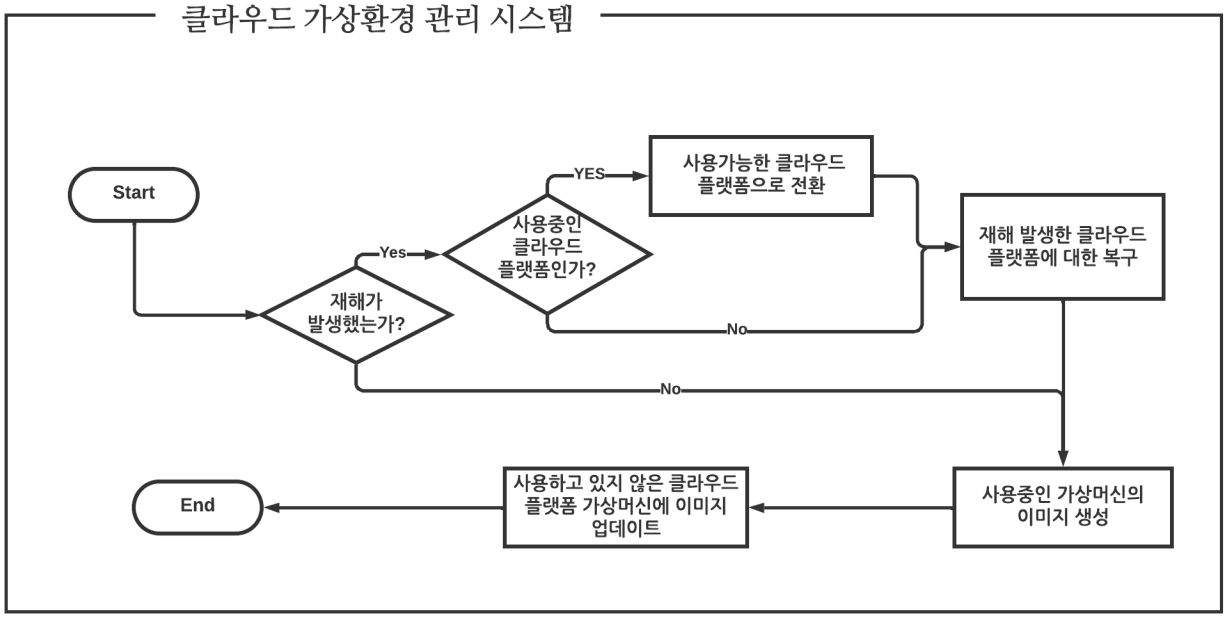


그림 6. 재해복구 management 흐름도

- 생성되어 있는 가상머신들에게 주기적으로 백업용 이미지와 재해 발생을 판단하기 위한 응답을 요청한다. 이러한 요청에 대한 응답 시간이 재해 발생 판별 기준 시간을 초과하였을 경우 재해 발생으로 간주한다.
- 재해가 사용자가 메인으로 사용하는 클라우드 플랫폼에서 발생한 경우 사용자의 작업환경을 백업용 가상머신으로 전환해준 후 재해 발생 클라우드 플랫폼에 대한 복구를 진행한다.
- 반면에 재해가 백업용 클라우드 플랫폼에서 발생한 경우는 작업환경의 전환 없이 재해 발생 클라우드 플랫폼에 대한 복구를 진행해준다.
- 다음 재해 여부와 상관없이 현재 가상머신의 이미지를 요청한다. 이미지를 바탕으로 이종 클라우드 플랫폼의 가상머신에 이미지를 업데이트한다.

위 과정의 반복 수행을 통해 본 시스템의 재해 복구 관리 기능은 사용자에게 끊임없는 가상머신 작업 환경을 제공한다.

3. 현실적 제약 사항 분석 결과 및 대안

1) 현실적 제약 사항

① 사용자의 요구사항이 다양하기 때문에 모든 요구사항을 반영하는 것은 힘들다.
② 멀티 클라우드 환경을 재해복구의 수단으로 사용하므로 단일 클라우드 인프라 재해 복구 대비 RTO(Response Time Objectives, 복구 시간 목표), RPO(Response Point Objectives, 복구 지점 목표)가 나쁘다.
③ Openstack 의 경우 다양한 기능별 컴포넌트들이 존재하지만 Cloudstack 은 지원하는 컴포넌트가 그에 준하지 않는다.

2) 대안

① Orchestration 적용 사항에 대한 세분화, 하드웨어, 소프트웨어 요구사항 입력 카테고리의 종류를 다양하게 만든다.
② 재해복구 RPO 에 대한 약점을 보완하기 위해 백업용 이미지 생성 주기를 30 분 이내로 짧게 설정하여 보완, RTO 에 대한 약점을 멀티 플랫폼 동시 운용을 통해 보완한다.
③ Openstack 과 Cloudstack 에서 유사한 기능을 수행하는 컴포넌트를 이용하여 시스템의 핵심 기능을 구현한다.

4. 개발 환경 및 사용 기술

1) 개발 환경

표 3 은 시스템의 개발에서 사용되는 관련 도구들을 보여준다.

표 3. 개발 환경

개발환경	
소프트웨어 형상관리(SCM)	GitHub
클라우드 플랫폼	OpenStack, CloudStack
개발 IDE	VSCode
DBMS	MySQL
Front-end	React JS
Back-end	Python, Django, Terraform
API	REST API

2) 사용 기술

① Openstack

Openstack 은 IaaS(Infrastructure as a Service)를 쉽게 구축할 수 있는 클라우드 컴퓨팅 오픈 소스 프로젝트로, 프로세싱, 스토리지, 네트워킹 가용자원을 제어하는 목적의 여러 개의 하위 프로젝트로 이루어져 있다. 관리자는 데이터 센터의 프로세싱, 스토리지, 네트워킹 자원들을 대시보드를 통해 제어할 수 있고, 사용자는 웹을 통해 필요한 기능을 사용할 수 있다. Openstack 의 대표적인 구성 요소로는 Nova, Neutron, Cinder, Keystone, Glance, Swift, Horizon, Heat, Freezer 등이 있다.

I. Nova

오픈스택 컴퓨트 서비스 Nova 는 IaaS(Infrastructure as a Service) 시스템의 핵심 서비스로서 클라우드 컴퓨팅 운영체제에서 리눅스의 커널과 같은 역할을 담당하고 있다. 오픈스택의 많은 서비스 중에서 일종의 컨트롤러와 같은 역할을 맡고 있다. Nova 서비스는 처음 컴퓨터의 자원 풀을 관리하고 자동화하기 위해 설계됐으며 고성능 컴퓨팅 설정, 소프트웨어가 설치되지 않은 베어 메탈 시스템 그리고 가상화 기술과도 널리 사용되고 있다.

II. Neutron

Neutron 은 네트워크와 IP 주소를 관리하기 위해 사용되는 오픈스택 네트워크 시스템이다. 오픈스택 네트워크는 클라우드 시스템 설치 시에 네트워크가 장애물이나 제약 요소가 더 이상 아니라는 사실을 보장해주고 사용자들이 네트워크 설정을 스스로 할 수 있도록 도와주는 역할을 한다. Neutron 은 네트워크 서비스를 추가로 설치하고 관리할 수 있는 확장된 프레임워크를 제공하는데 이에는 침입탐지 시스템(IDS), 로드 밸런싱(부하 분산), 방화벽, 그리고 VPN(가상 사설망) 등이 포함된다.

2022 전기 졸업과제

III. Cinder

오픈스택 블록 스토리지 Cinder 는 Nova 서비스가 제공하는 인스턴스에 지속적으로 사용이 가능한 블록 스토리지 장치를 제공한다. 여기서 블록 스토리지 시스템은 블록 장치를 생성하고 서버에 부착하고 분리하는 업무를 담당한다. 블록 스토리지 볼륨은 오픈스택 컴퓨트와 대시보드에 통합돼 사용이 가능한데 여기서 대시보드는 클라우드 사용자들이 스스로 스토리지 요구 사항을 관리할 수 있도록 접속을 허용한다.

IV. Keystone

오픈스택 인증 서비스 Keystone 은 중앙에서 사용자에 대한 인증 디렉토리를 제공하는데 이 디렉토리는 사용자가 접근 가능한 오픈스택 서비스와 매핑 된 정보를 저장하고 있다. Keystone 서비스는 전체 오픈스택 클라우드 운영 시스템에서 일반적인 운영 시스템처럼 동작하며 LDAP 처럼 백엔드에서 동작하는 디렉토리 서비스와 서로 통합할 수 있다. Keystone 서비스는 여러 형태의 인증 방법을 지원하는데 전통적인 방식의 사용자 ID 와 패스워드, 토큰 기반의 인증 시스템 그리고 AWS 스타일도 그 범주에 속한다. 또한 프로그래밍 방식으로 사용자와 서드파티 툴을 사용하면 이들이 어떤 자원에 접근할 수 있는지도 결정할 수 있다.

V. Glance

오픈스택 이미지 서비스 Glance 는 디스크와 인스턴스를 생성할 서버 이미지를 발견해서 등록하고 전송하는 역할을 수행한다. 여기서 저장된 이미지는 마치 Template 처럼 사용할 수 있다. 또한 무제한으로 백업이 가능하고 그 목록을 만들 수도 있다. Glance 서비스는 Swift 를 포함해 다양한 백엔드 시스템에 디스크와 서버 이미지를 저장할 수 있다. Glance API 는 이렇게 백엔드에 저장된 디스크 이미지에 대한 정보를 질의할 수 있는 RESTful 인터페이스를 제공한다. 그리고 클라이언트가 새로운 서버로 이러한 이미지를 내려받을 수 있는 기능도 제공한다.

VI. Swift

오픈스택 오브젝트 스토리지 서비스 Swift 는 확장이 용이한 뛰어난 스토리지 시스템이다. 사용자는 오브젝트와 파일들을 스토리지를 구성하는 여러 서버에 설치된 디스크 드라이브에 분산해서 저장한다. 여기서 오픈스택 소프트웨어 Swift 는 한 클러스터 전체에 걸쳐서 데이터의 복제와 무결성을 보장하기 위한 목적으로 사용된다. 만약 운영 중인 클러스터의 서버나 하드 드라이브에 문제가 생겨 시스템 장애가 발생한 경우 오픈스택 Swift 서비스는 현재 활성화된 노드로부터 그 콘텐츠를 클러스터 내부의 다른 위치로 복제한다.

VII. Horizon

오픈스택 대시보드 서비스 Horizon 은 관리자와 사용자들이 클라우드 기반의 자원에 대한 접근을 허용하고 이러한 서비스 및 자동화 기능을 제공하는 GUI 서비스이다. 사용자들은 이 대시보드를 통해 오픈스택이 제공하는 여러 자원들과 편리하게 의사소통 할 수 있다. 또한 개발자들은 오픈스택 자체에서 제공하는 API 또는 AWS EC2 와 호환이 가능한 API 를 사용하면 현재 사용 중인 자원들에 대한 접속을 자동화하거나 관리하기 위한 툴을 개발할 수 있다.

VIII. Heat

오픈스택 오케스트레이션 서비스 Heat 는 Template 을 이용해 오픈스택 자체에서 제공하는 REST API 와 Query API 를 통해서 여러 복잡한 클라우드 애플리케이션을 생성, 관리 및 조정하기 위해 사용된다.

IX. Freezer

오픈스택 재해복구 서비스 Freezer 는 가상머신을 주기적으로 스냅샷을 찍어 오브젝트 스토리지인 Swift 에 보관하다가 장애 시 Restore 기능을 이용하여 정상적인 상태의 가상머신으로 재생성하기 위해 사용된다. 가상머신은 File System, Oracle, Mysql 등 다양한 모드로 백업이 가능하다. 또한 주기적으로 Backup & Restore 를 수행할 수 있도록 스케줄 기능을 지원한다.

그림 7 은 오픈스택 아키텍처를 나타낸 그림이다.

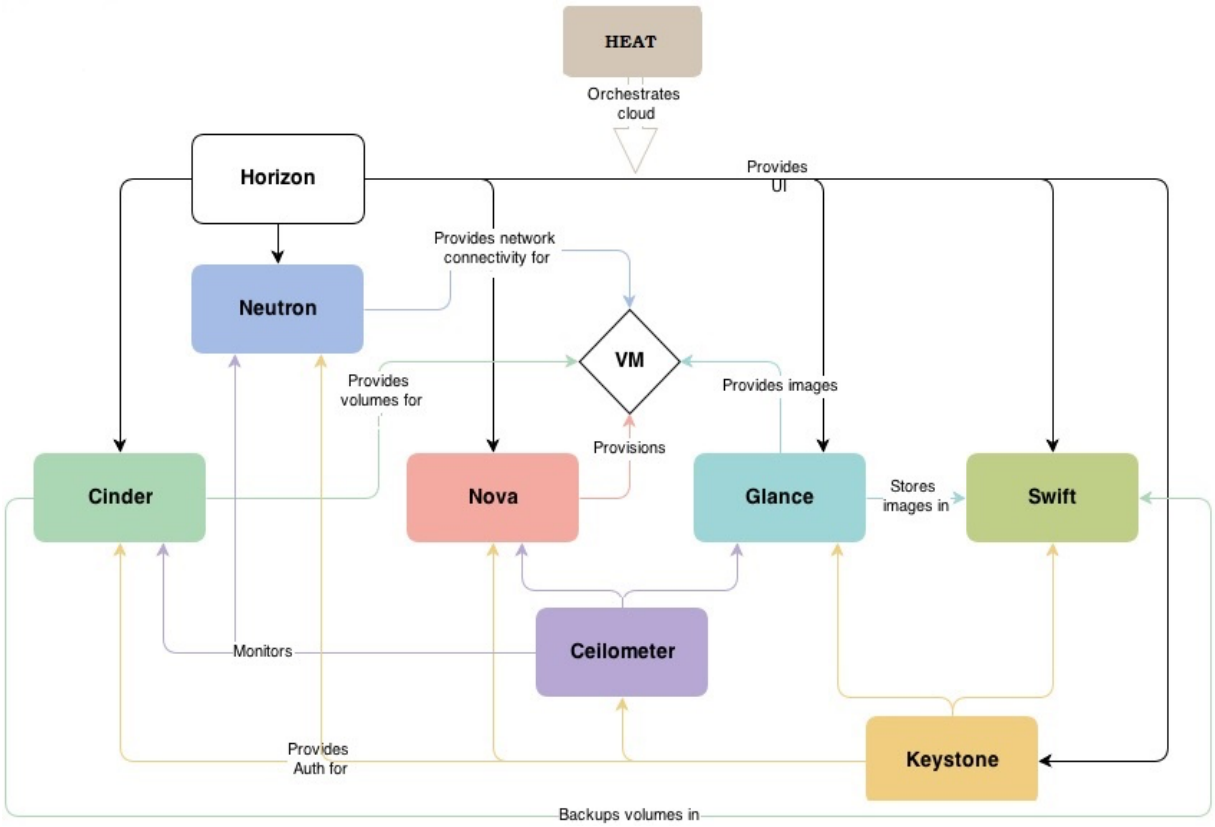


그림 7. 오픈스택의 아키텍처

② Apache Cloudstack

Apache Cloudstack 은 높은 가용성과 뛰어난 확장성을 지닌 IaaS 클라우드 컴퓨팅 플랫폼으로서 대규모 가상머신 네트워크를 배포하고 관리하도록 설계된 오픈소스 소프트웨어이다. Cloudstack 은 Vmware, KVM, citrix, XenServer 등의 하이퍼바이저를 지원한다. 또한 GUI, CLI 를 제공하며 RESTful API 로 관리할 수 있다. 또한 AWS 의 EC2, S3 와 호환되는 API 를 제공하므로 하이브리드 클라우드 환경 구축 및 배포가 가능하다. 또한 각 VM 을 배포할 때 네트워크 및 스토리지 설정이 자동으로 구성되어서 VM 들 간 일관성이 높고 가용성이 높은 클라우드 환경을 구축할 수 있다. Cloudstack 은 많은 회사에서 On-Premise 클라우드 혹은 Hybrid 클라우드 솔루션으로 사용되고 있다.

그림 8 은 Cloudstack 의 구조를 나타낸 것이다.

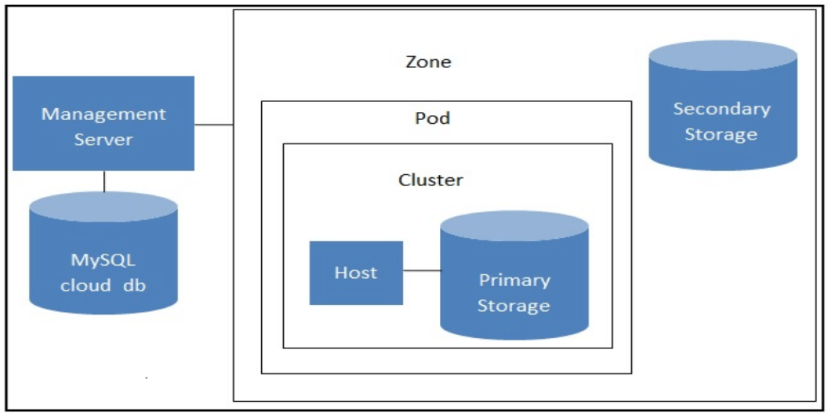
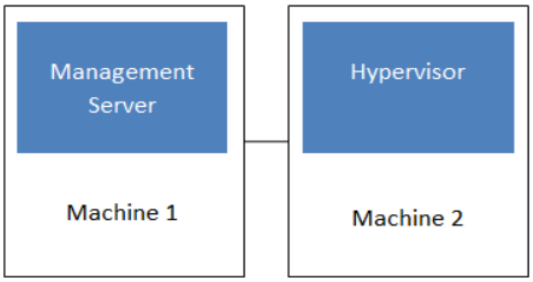


그림 8. Cloudstack 의 아키텍처

Cloudstack 의 배포구조는 management server 와 management server 의 관리를 받는 resources 들로 구성된다. management server 는 굉장히 단순한 구조를 가지고 있으며 심지어 하나의 노드가 management server 와 resources 역할을 동시에 수행할 수 있다. management server 는 시스템의 안정성 및 지속성을 위해 Apache Tomcat Server 와 MySQL 데이터베이스 환경에서 실행된다.

그림 9 는 Cloudstack 의 배포 구조를 간략히 나타낸 것이다.



Simplified view of a basic deployment

그림 9. 클라우드스택의 배포 구조

③ Terraform

Terraform 은 하시코프에서 오픈소스로 개발된 클라우드 인프라 자동화를 지향하는 코드로서의 IaC(Infrastructure as Code)도구이다. 여기서 IaC 는 코드로 인프라를 관리한다는 개념으로 Terraform 에서는 하시코프 설정 언어(Hashicorp Configuration Language)을 사용하여 Template 형태로 클라우드 리소스를 선언한다. 이렇게 코드로 인프라를 관리할 경우 환경의 변경 및 배포와 같은 반복되는 작업에 걸리는 시간을 줄일 수 있고 관리 작업의 효율을 높일 수 있다. AWS 에서 자체적으로 만든 IaC 도구인 AWS Cloud Formation 은 AWS 만 지원하지만 Terraform 의 경우 AWS, 구글 클라우드 플랫폼, 마이크로소프트 애저(Azure) 와 같은 주요 클라우드 서비스를 비롯하여 다양한 클라우드 서비스들을 프로바이더 방식으로 제공하고 있다. 이를 통해 Terraform 만으로 멀티 클라우드의 리소스들을 선언하고 관리하는 것이 가능하다.

그림 10 은 테라폼의 아키텍처를 나타낸 그림이다.

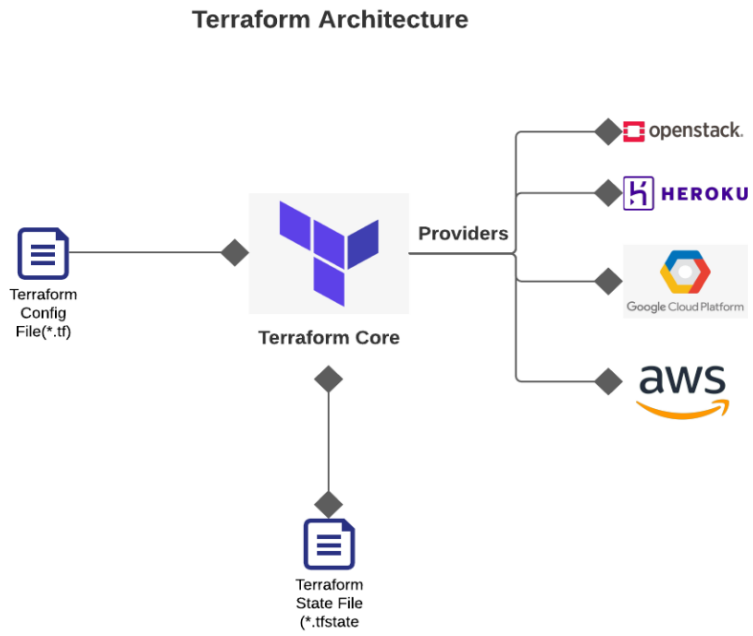


그림 10. 테라폼의 아키텍처

④ Django

Django 는 파이썬 기반 웹 애플리케이션 프레임워크이다. 좀 더 쉽게 웹 개발을 할 수 있게 개발에 일반적으로 사용되는 다양한 도구를 제공한다. Django 의 구성으로 관계형 데이터베이스로 구축해주는 모델(Model), HTTP 요청을 처리하는 웹 Template 시스템인 뷰(View), URL 의 라우팅을 처리하는 URL 컨트롤러(Controller)가 있고 이러한 모델, 뷰, 컨트롤러로 구성된 MVC 디자인 패턴을 따른다. 하지만 Django 에서는 이러한 MVC 디자인 패턴의 컨트롤러의 기능을 프레임워크 자체에서 하기 때문에 모델(Model), Template(Template), 뷰(View)로 분류해 MTV 프레임워크라고 부른다.

I. 모델

모델은 데이터에 관한 정보를 담고 있고 데이터에 대한 접근, 검증, 작동과 데이터 사이의 관계를 정의한다. 모델을 정의할 때 필드의 종류를 지정하는데, 이것이 데이터베이스에게 컬럼 타입을 알려주고 HTML 폼으로 표시될 때의 입력 타입도 내포하는 역할을 한다.

II. 뷰

뷰는 어떤 데이터가 표시될 것인지를 정의하는 역할을 한다. HTTP 의 응답을 반환해야 하고 이러한 응답의 종류는 웹 페이지, 리디렉션, 문서 등의 다양한 형태가 가능하다.

III. Template

Template 은 MVC 패턴의 뷰에 대응되며 사용자에게 보여지는 화면을 의미한다. 즉 데이터가 어떻게 표시되는지를 정의하고 사용자에게 실제로 보여지는 웹 페이지나 문서 등을 다룬다.

그림 11 은 장고의 아키텍처를 나타낸 그림이다.

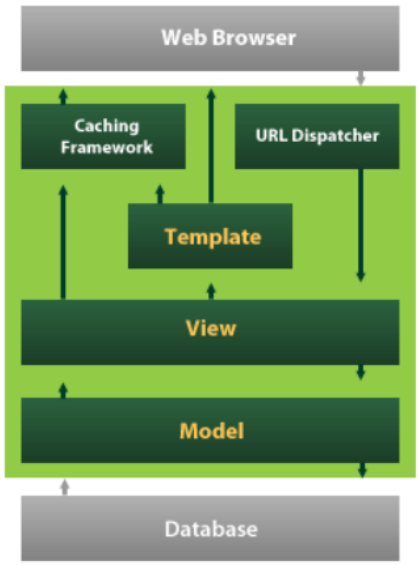


그림 11. 장고의 아키텍처

5. 개발 일정 및 역할 분담

1) 개발 일정

표 4. 개발 일정

기간 / 수행내용	5 월		6 월				7 월					8 월				9 월				
	3	4	1	2	3	4	1	2	3	4	5	1	2	3	4	1	2	3	4	5
필요 지식 습득	All																			
DB 구축 및 사용자 별 가상환경 정보 반환 기능 구현					고준성															
맞춤형 가상환경 Template 생성 컴포넌트 구현					이봉훈															
변경된 요구사항 가상환경에 적용 기능 구현					김영후															
재해발생 시 알림 기능 구현										고준성										
										이봉훈										
멀티 클라우드 플랫폼 이용 재해복구 기능 구현													고준성							
													이봉훈							
웹 기반 대시보드 컴포넌트 구현										김영후										
테스트 및 보완															All					
최종보고서 작성																		All		

2) 역할 분담

표 5. 역할 분담

이름	역할
공통	필요 지식 습득, 멀티 클라우드 개발 환경 구축, 테스트 및 보완, 중간보고서, 최종보고서 작성
이봉훈	맞춤형 가상환경 Template 생성 컴포넌트 구축, 멀티 클라우드 플랫폼 이용 재해복구 기능 구현
고준성	DB 구축 및 사용자 별 가상환경 정보 반환 기능 구현, 재해발생 시 알림 기능 구현, 멀티 클라우드 플랫폼 이용 재해복구 기능 구현
김영후	변경된 요구사항 가상환경에 적용 기능 구현, 웹 기반 대시보드 컴포넌트 구축