



WiKiKube

Play Data

DUYOUNG JANG, EUNVIT JO, YUNHO KANG, WONGYUN CHO

2021.06.18

Table of Contents

목표	2
프로젝트 기술 설명	2
개발환경	2
CI/CD	2
프로젝트 관리 및 라이프 싸이클	4
Requirements of The Project	4
Functional Requirements	5
Non-Functional Requirements	5
Tools	5
Front End	5
Back End	8
MVT Pattern	8
Model (MVC.Model)	9
Template (MVC.View)	9
View (MVC.Controller)	12
MySQL 테이블	14
Use Case	16
Architecture	17
AWS Architecture	17
Feedback form	18
Three-tier Applications Architectural Pattern	18
Gantt Chart	19
Reference	21

목표

이 프로젝트의 목적은 웹 애플리케이션을 구축하고 Docker, CI/CD, Elastic Kubernetes Service (EKS) 및 Terraform 을 사용하여 현대적인 I.T 인프라를 구현하는 방법을 보여주는 것입니다. 한국어로 된 원본 k8 문서가 있습니다. 그러나 번역의 일부가 누락되고 한국 사용자와의 의사 소통이 부족합니다. 그래서 우리는 한국 사용자들을 위한 한국 k8s 커뮤니티 웹 사이트 구축에 기여하기로 결정했습니다. 애플리케이션은 기본적으로 k8 이라고도하는 Kubernetes, 기본 개념과 필수 기능 및 학습 할 샘플 그리고 실습 튜토리얼을 제공하는 한국어로 된 문서입니다. 또한 더 나은 사용자 경험을 위해 사용자 피드백 및 댓글로 의사소통할 수 있습니다. 개발에 관해서는 프론트 엔드 에서 JavaScript, HTML5 및 Bootstrap 을 사용하고 백엔드 에서 Python Django 프레임 워크를 그리고 인프라는 EKS, Docker, Terraform, GitHub Action 및 Argo CD 를 이용하여 구축 했습니다. 연구와 문서작성 목적으로 AWS, K8, Docker, EKS Terraform 와 Django 공식 문서를 사용할 것입니다.

프로젝트 기술 설명

웹 프레임워크로 이용한 장고는 - 로그인, 회원가입, 인증, CORS, data parsing 등 라이브러리를 이용할 수 있어 개발 시간을 많이 단축 시켜 빌드 할 수 있어 많이 사용되는 웹 프레임워크 중에 하나이다.

K8s, IaC 및 CI / CD 는 오늘날 광범위하고 현대적인 기술이며, 따라서 수요가 많으며 중소기업에서 대기업에 이르기까지 번거로움 없이 I.T 인프라를 구축하는 데 기본적으로 사용됩니다.

Kubernetes 는 컨테이너화 된 애플리케이션의 배포, 확장 및 관리를 자동화하기 위한 오픈 소스 컨테이너 오케스트레이션 엔진입니다.

Infrastructure as Code(IaC)는 수동 프로세스가 아닌 코드 또는 스크립트를 통해 인프라를 관리하고 프로비저닝을 자동화 할 수 있습니다 .CI / CD 는 Code 부터 Production Deployment 까지의 문제에 대한 솔루션 인 앱 개발 단계에 자동화를 도입하여 고객에게 지속적으로 앱을 제공하는 방법입니다.

개발환경

CI/CD

Figure 1. Github Action 을 이용하여 Slack 으로 실시간으로 코드가 업데이트되면 팀원 전체가 알림을 받을 수 있습니다. Figure 2. Github Action 을 이용하여 Slack 과 Docker 에 연동 시켜 소스코드가 업데이트되면 아래 이미지와 같이 지속적 으로 Slack 알림이 오고, Docker Image 를 자동으로 생성 합니다. Figure 3. Argo CD 에 Github 과 연동하고 Auto Sync 로 동시에 소스가 업데이트되면 배포가 됩니다.

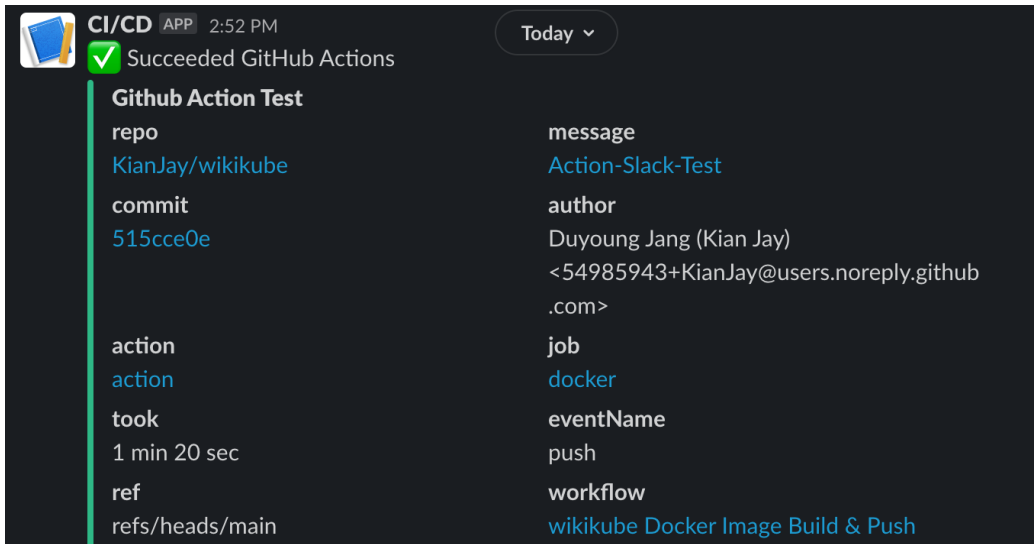


Figure 1. Slack notification

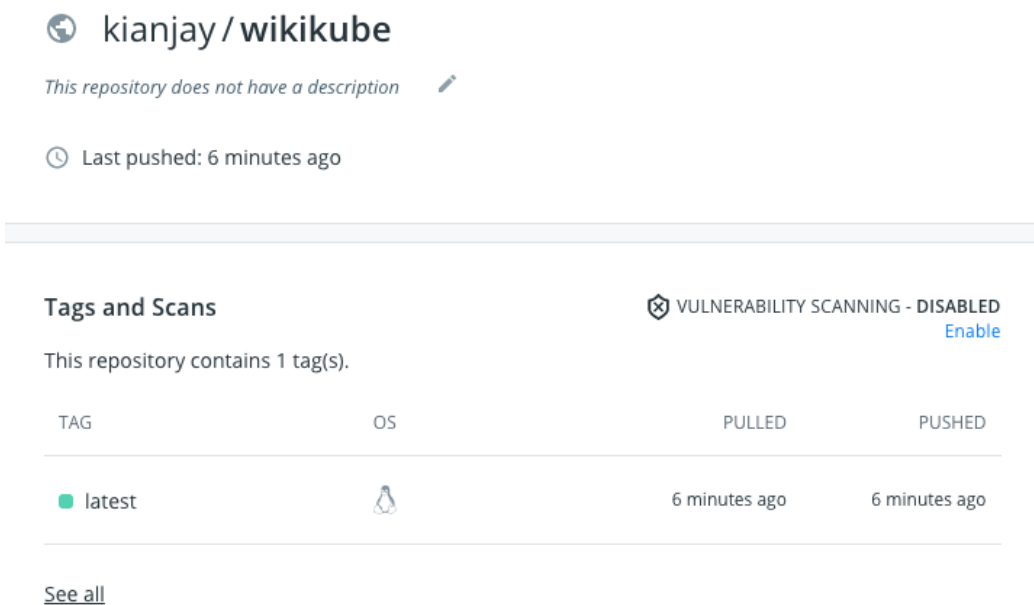


Figure 2. Docker image build

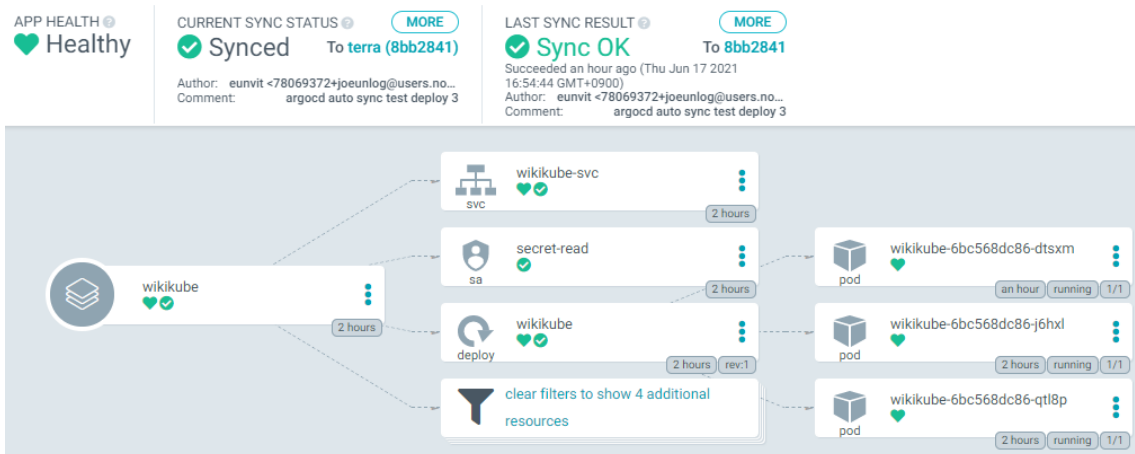


Figure 3. Argo CD

프로젝트 관리 및 라이프 싸이클

Figure 4. 팀원이 WikiKube 를 개발하고 운영하는 방식에 대해서 일일 보고한 내용 입니다.

- 월-금 아침 9 시 To-Do 리스트 및 이슈 리포트를 칸반에 작성합니다.
- 월-금 아침 10 시 10 분 부터 10 시 40 분까지 팀 회의 주로 내용은 오늘 할일과 이슈 정리를 합니다.
- 현재 개발 진행 상태를 칸반에 작성합니다
- 월-금 16 시부터 16 시 30 분까지 일일 보고서를 팀원들과 함께 작성합니다
- 문제점 또는 버그가 생기면 칸반에 작성해서 이슈를 팀원들에게 알립니다.
- 팀원들과 해결된 이슈는 해결된 이슈로 옮긴 후 케이스를 닫습니다.
- 완성된 기능 및 개발

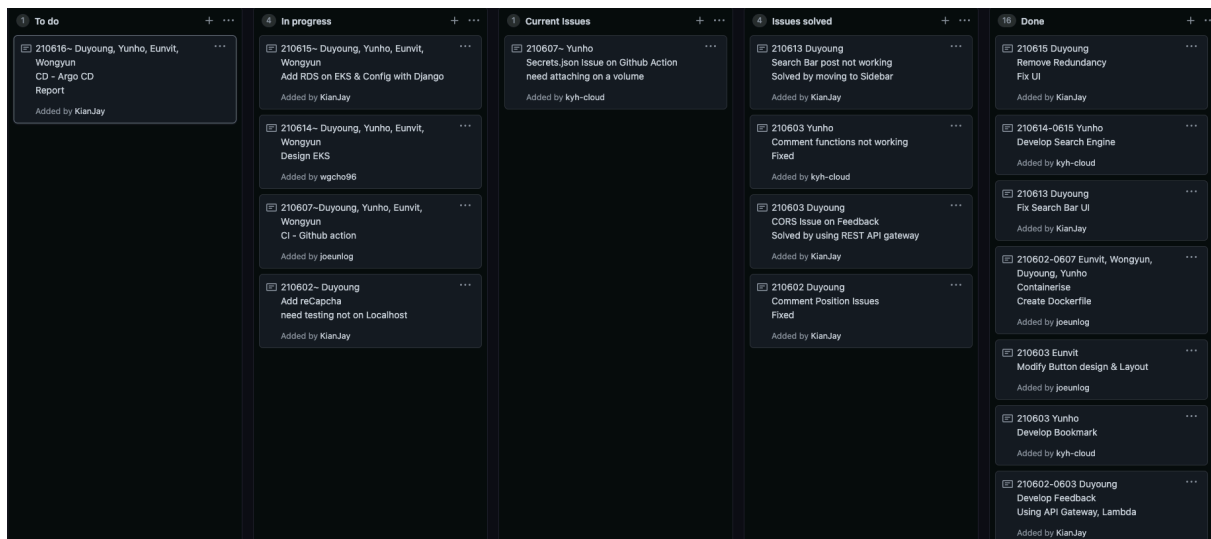


Figure 4. Github Project Kanban

Requirements of The Project

- AWS 클라우드 200 만원 지원금
- 25 일간의 프로젝트
- 어플리케이션 개발
- 모던 인프라 구성
- 아키텍처 디자인
- 팀 프로젝트
- 고 가용성 아키텍처

Functional Requirements

- 유저의 검색
- 유저의 피드백 보내기
- 유저의 회원가입
- 유저의 로그인
- 회원 가입된 유저가 비밀번호를 잊어 버렸을 때 비밀번호 재 생성
- 회원 가입된 유저들의 북 마크 추가, 수정, 삭제
- 회원 가입된 유저들의 댓글 쓰기, 수정, 삭제
- 관리자 콘텐츠 쓰기, 수정, 삭제
- 관리자 유저 관리

Non-Functional Requirements

- 문서 내용 정확도
- 어플리케이션 작동 퍼포먼스
- 유저 정보 보안성
- 문서 내용 유용 성

Tools

- Front-end: JavaScript, HTML5, CSS, Bootstrap
- Back-end: Python, Django, RDS MySQL
- Cloud: AWS EKS
- CI: Github Action, Dockerhub, Slack
- CD: Argo CD
- Management: Github Kanban, Zoom, Google Meet

Front End

홈페이지 부트스트랩과 jQuery 이용하여 테블릿, 모바일, PC 에서 Responsive 하게 디자인 하였습니다.

Figure 5. 웹사이트 홈페이지 입니다 iDocs 템플릿을 참고하여 UI/UX 디자인 하였으며, 왼쪽 사이드바 검색창 을 포함하고, 중앙 메인 콘텐츠 그리고 상단에는 메인 네비게이션 홈 안에는, 어바웃, 피드백, Sign up, Login 으로 구성 되 있습니다.

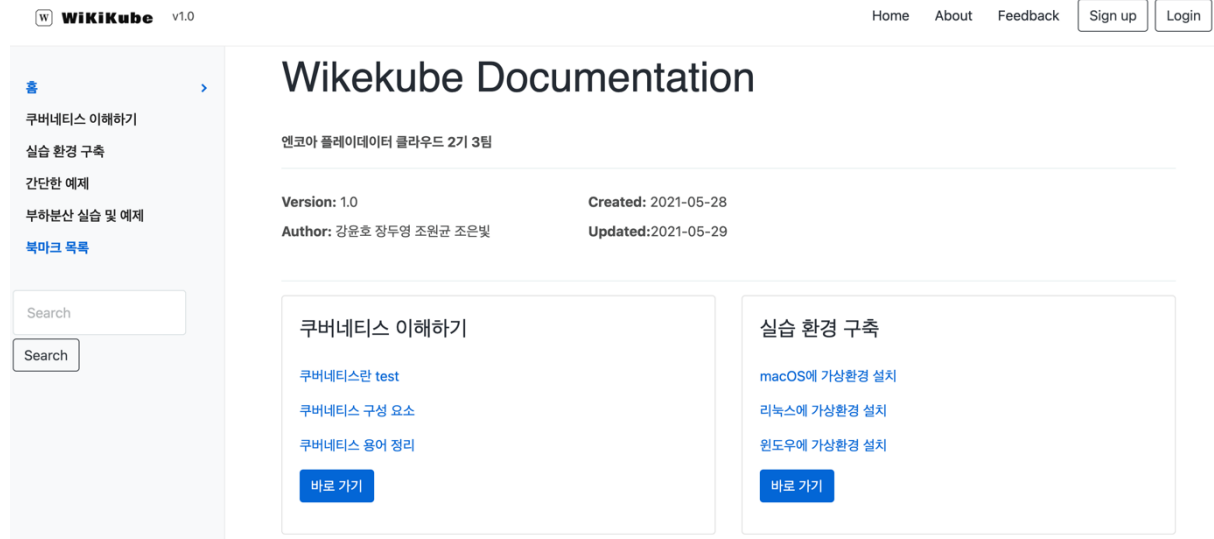


Figure 5. WikiKube 홈페이지

회원가입 화면

Figure 6. 상단에 있는 네비게이션 바 Sign Up 버튼을 회원 가입 페이지로 넘어 갑니다.

ID, First Name, Last Name, E-Mail, Password, Password Confirm 정해진 값에 맞는 인풋을 받고 Register 버튼을 누르면 회원가입이 완료됩니다.

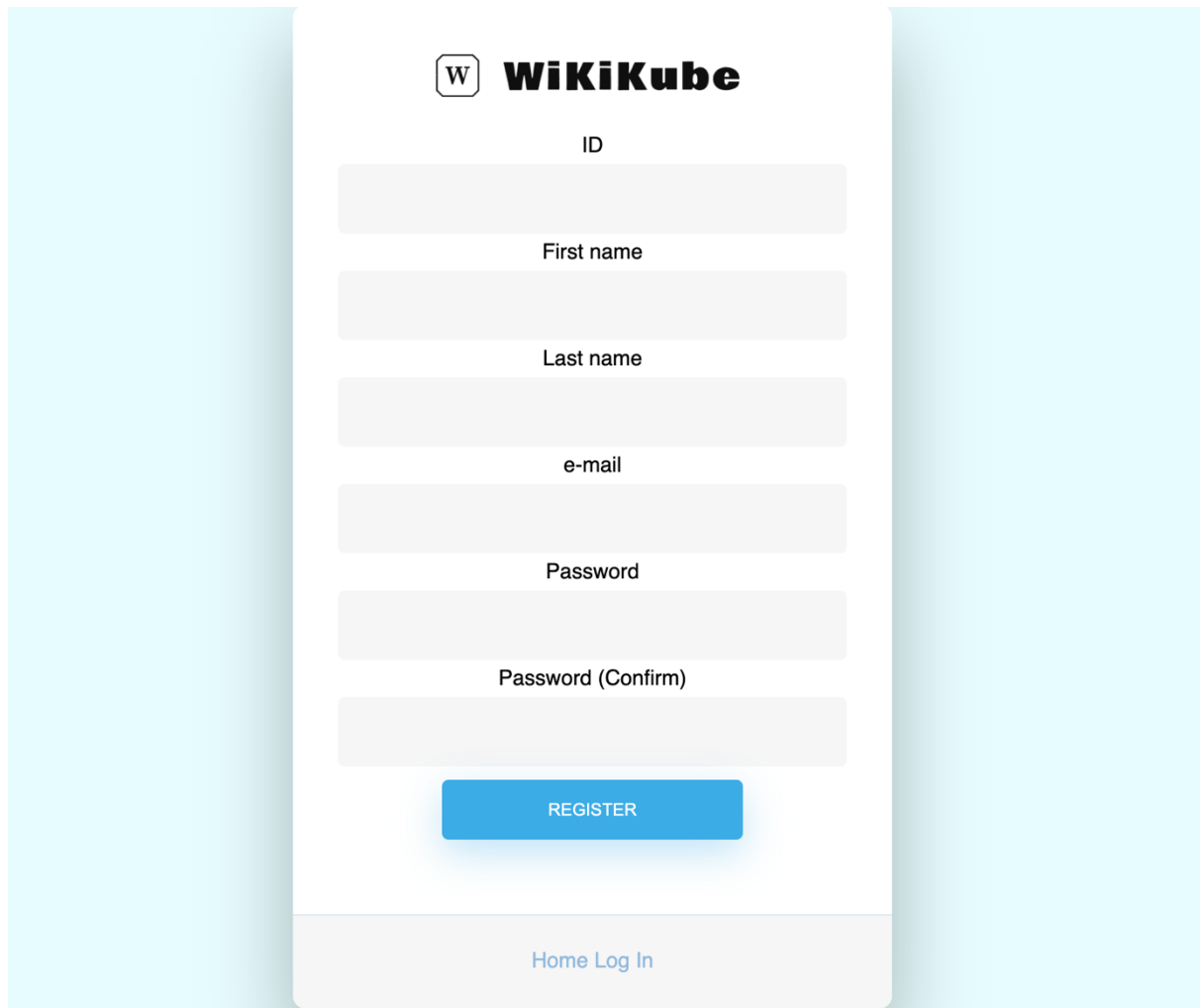
The image shows a registration form for 'WiKiKube'. At the top, there is a logo consisting of a square with the letter 'W' inside, followed by the text 'WiKiKube'. Below the logo, there are seven input fields, each with a label above it: 'ID', 'First name', 'Last name', 'e-mail', 'Password', and 'Password (Confirm)'. Each input field is a light gray rectangle. Below the 'Password (Confirm)' field is a blue button with the text 'REGISTER' in white. At the bottom of the form, there is a light gray bar containing the text 'Home Log In' in blue.

Figure 6. 회원가입 화면

피드백 화면

Figure 7. 상단에 있는 네비게이션에서 Feedback 버튼을 누르면 창이 하나 생기고 올바른 값을 입력하면 입력된 정보를 WiKiKube 운영자 이메일로 보낼 수 있습니다.

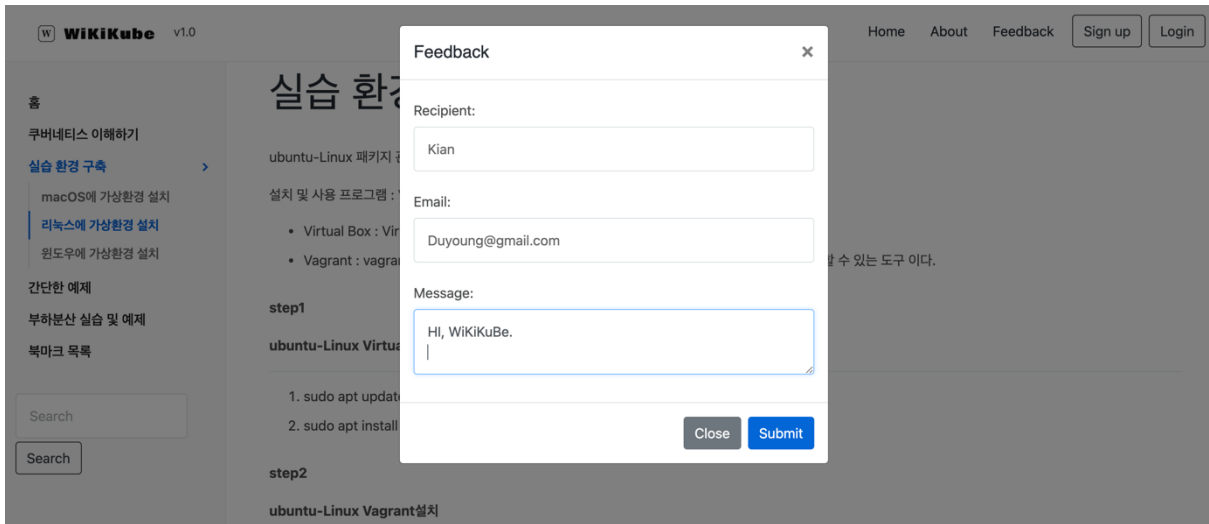


Figure 7. 피드백 폼

Back End MVT Pattern

Figure 8. 장고 프레임워크의 View 는 Template, Controller 는 View 이며, 이를 MVT 패턴이라고 합니다.. Model 은 데이터 베이스에 저장되는 데이터이고, Templates 사용자에게 보여지는 Front-End, View 는 프로그램 Logic 이 동작하여 데이터를 가져와서 처리한 결과를 템플릿에 전달하는 역할을 수행합니다.

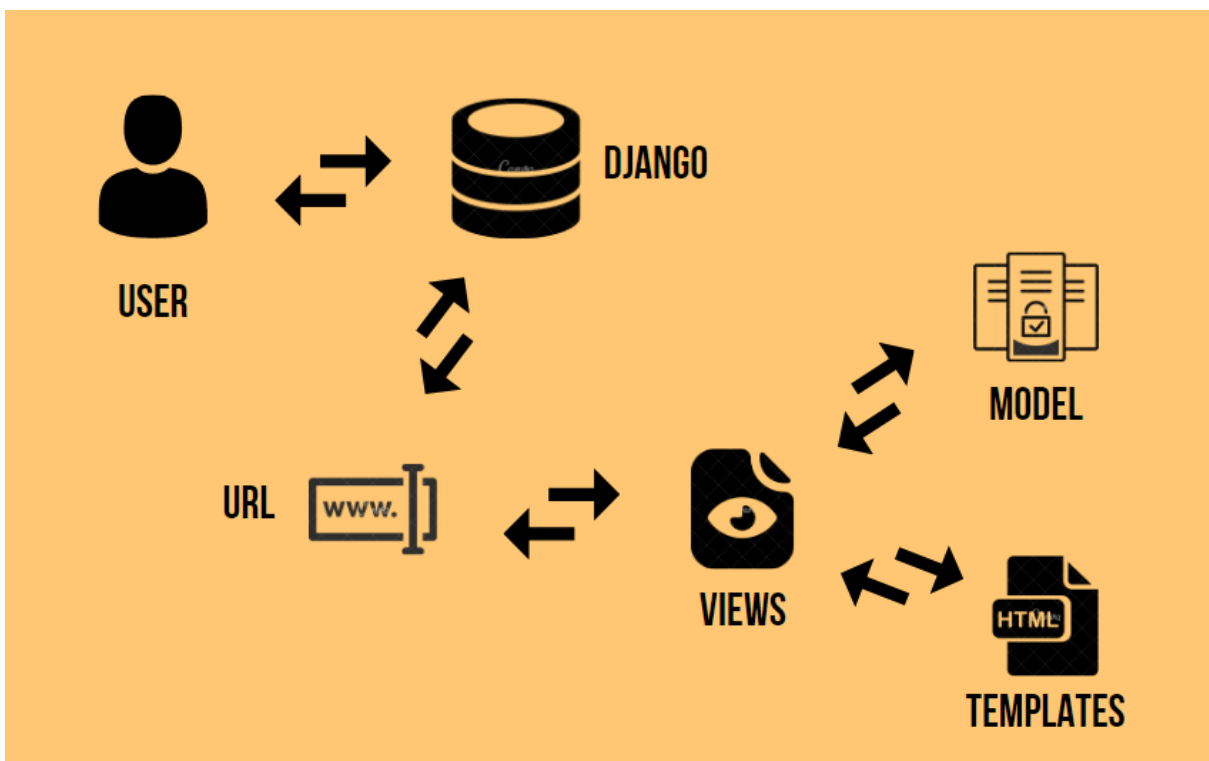


Figure 8. Django MVT 패턴

Model (MVC.Model)

User : 사용자

- Username : 사용자 ID
- First_name : 사용자의 first name
- Last_name : 사용자의 last name
- Email : 사용자의 email
- Password : 사용자의 암호

Post : 게시물

- Title : 제목
- Category : 카테고리
- Content : 내용

Comment : 댓글

- comment_content : 댓글 내용
- post_id : 게시물 고유번호
- user_id : 작성자 아이디
- com_create_date : 작성 날짜

Bookmark: 북마크

- book_user : 북마크 사용자
- post_id : 북마크된 게시물 고유번호

Template (MVC.View)

index.html

- wikikube 사이트의 home 화면입니다
- 사이트 작성자와 배포 날짜 정보 및 게시물의 전체 레이아웃을 확인할 수 있습니다.

signup.html

- 사용자 가입을 위한 template으로, user model의 각 인자를 입력 받습니다
- Password는 확인을 위해 두 차례 입력하도록 되어있으며, 같은 값을 입력해야 가입에 성공할 수 있습니다.
- Password는 특정 양식에 맞게 입력해야 하며, 맞지 않게 입력한 경우 가입에 실패하고 에러 메시지와 password 작성 양식 안내 메시지가 뜹니다.
- 화면 상단의 로고 혹은 하단의 'Home' 링크를 클릭하여 index 화면으로 이동할 수 있습니다.
- 화면 하단의 'Log In' 링크를 클릭하여 로그인 화면으로 이동할 수 있습니다.

login.html

- 사용자가 로그인을 하기 위한 template으로, user model의 인자 중 username(ID), password(암호)를 입력 받습니다.
- 화면 상단의 로고를 클릭하여 index 화면으로 이동할 수 있습니다.
- 화면 하단의 'Sign up' 링크를 클릭하여 사용자 가입 화면으로 이동할 수 있습니다.
- 사용자가 비밀번호를 잃어버렸을 경우, 화면 하단의 'Forgot Password' 링크를 클릭하여 사용자 비밀번호 초기화 화면으로 이동할 수 있습니다.

forgetpw.html

- 사용자가 비밀번호를 잃어버렸을 경우, 초기화를 하기 위한 메일 발송 template입니다.
- 사용자 식별을 위해 first name, last name, email을 입력 받습니다.
- 'Reset Password' 버튼을 클릭하여 비밀번호 초기화 메일을 전송할 수 있습니다.

resetpassworddone.html

- 사용자 비밀번호 초기화 과정에서 메일을 성공적으로 발송했을 경우 표시되는 화면입니다.
- 화면 상단의 로고를 클릭하여 index 화면으로 이동할 수 있습니다.
- 화면 하단의 'Sign up' 링크를 클릭하여 회원가입 화면으로 이동할 수 있습니다.
- 화면 하단의 'Log In' 링크를 클릭하여 로그인 화면으로 이동할 수 있습니다.

password_reset_confirm.html

- 사용자 비밀번호 초기화 과정에서 새로운 비밀번호를 입력 받기 위한 template입니다.
- 비밀번호를 두 차례 입력하게 되어있으며, 같은 값을 입력해야 가입에 성공할 수 있습니다.
- Password는 특정 양식에 맞게 입력해야 하며, 맞지 않게 입력한 경우 새 암호 설정에 실패하

고 에러메시지가 뜹니다.

password_reset_complete.html

- 사용자 비밀번호 초기화 과정에서 비밀번호를 성공적으로 다시 설정할 경우 표시되는 화면입니다.
- 화면 상단의 로고를 클릭하여 index 화면으로 이동할 수 있습니다.
- 화면 하단의 'Sign up' 링크를 클릭하여 회원가입 화면으로 이동할 수 있습니다.
- 화면 하단의 'Log In' 링크를 클릭하여 로그인 화면으로 이동할 수 있습니다.

change pw.html

- 로그인 상태의 사용자가 비밀번호를 변경하기 위한 template입니다.
- 기존 password와 새로운 password를 입력 받고, 새로운 password는 확인을 위해 두 차례 입력 받도록 되어있습니다.
- 성공적으로 변경하면 index 화면으로 자동 이동 됩니다.

postDetail.html

- 인자로 받은 post로 게시글의 내용을 마크다운 형식으로 변환해서 표시해줍니다.
- 댓글리스트와 댓글입력창이 존재합니다.
 - addComment, deleteComment, movetoEditComment 모두 postDetail에서 실행됩니다.
- 북마크 설정/해제 버튼이 존재하는데 사용자의 북마크 설정유무에 따라 버튼이 달라집니다.
 - 북마크 설정시 addBookmark 실행, 해제시 delBookmark 실행 됩니다.

editComment.html

- movetoEditComment로 들어오게 되는 웹페이지 입니다
- 댓글을 수정할 때 사용되는 페이지고, 입력창에는 기존 댓글내용이 입력 돼 있습니다.
- Edit 버튼을 누르면 입력창에 입력한 내용을 인자로 가져가서 editComment 함수를 실행하고, Cancel 버튼을 누르면 댓글 수정을 취소하고 원래 게시글 페이지로 돌아 갑니다.

bookmark.html

- 사용자가 등록한 북마크의 list를 받아와서 화면에 세로로 나열합니다.
 - 이 때, 북마크로 등록된 게시글의 첫번째 줄을 표시해줍니다.

- 각각의 텍스트는 해당 게시글로 하이퍼링크가 돼있습니다

View (MVC.Controller)

addComment

1. 로그인 유무를 먼저 검사한다. 비로그인 상태라면 로그인 페이지로 이동하고 끝난다. 로그인 된 상태라면 POST 요청을 받습니다.
2. POST에 담긴 데이터를 활용하여 Comment object를 생성 합니다.
3. 현재 게시글 페이지를 새로고침하면서 마무리 합니다.

movetoEditComment

1. 로그인 유무를 먼저 확인한다. 비로그인 상태시 권한이 없다는 페이지를 새로고침하면서 권한이 없다는 메시지를 출력 합니다.
2. 로그인 상태가 확인되면, 댓글 작성자랑 현재 유저가 일치하는지 확인한다. 불일치시 페이지를 새로고침하면서 권한이 없다는 메시지를 출력 합니다.
3. 로그인 유저랑 댓글 작성자가 일치시 editComment.html로 이동 합니다.

editComment

1. POST요청을 통해서 댓글 내용을 받아서 Comment object를 수정해서 저장한 다음에 원래 게시글 페이지로 다시 돌아 갑니다.

deleteComment

1. 현재 로그인된 유저와 댓글 작성자가 동일한지 검사합니다. (로그인이 안되어 있더라도 검사가 가능).
2. 로그인 유저랑 작성자가 동일하면 해당 댓글에 대한 삭제작업을 진행하고 일치하지 않으면 권한이 없다는 메시지를 출력 합니다.

viewPost

1. category와 title에 해당하는 Post 객체와 해당 Post에 해당하는 Comment 객체를 불러 옵니다.
2. 현재 사용자의 로그인 유무를 검사한다. 로그인 상태일시 사용자의 Bookmark 객체도 함께 불러온다. 그리고 불러온 객체들을 context라는 딕셔너리에 담아서 postDetail.html 페이지를 불러 옵니다.

비로그인 상태일 경우 Bookmark 객체는 불러오지 않고 기존에 불러온 객체들로만 context 를 구성해서 postDetail.html 페이지를 불러 옵니다.

showBookmark

1. 현재 사용자의 로그인 유무를 먼저 확인한다. 로그인이 안돼있을 경우 로그인 페이지로 사용자를 안내 합니다. 로그인이 돼있을 경우, 사용자의 아이디로 등록돼 있는 Bookmark 객체들을 필터링해서 가져 옵니다. 그리고 bookmark.html에서 북마크된 게시글들의 첫번째 줄 내용을 목록으로 표시 합니다.

addBookmark

1. 로그인이 된 상태이고, 현재 게시글을 사용자가 북마크로 등록하지 않은 상태이면 postDetail.html에서 버튼이 표시 됩니다.
2. 버튼을 누르면, 현재 사용자의 id와 게시글 고유번호를 불러와서 새로운 Bookmark 객체를 생성 합니다.
3. postDetail.html 페이지를 새로고침해서 보여 줍니다.

delBookmark

1. 로그인이 된 상태이고, 현재 게시글을 사용자가 북마크로 등록한 상태이면 postDetail.html에서 버튼이 표시 됩니다.
2. 버튼을 누르면, 현재 사용자의 id와 게시글 고유번호를 불러와서 기존 Bookmark 객체를 삭제 합니다.
3. postDetail.html 페이지를 새로고침해서 보여 줍니다.

CreateUserView

1. signup template과 CreateUserForm을 이용하여 signup 페이지를 생성 합니다.
2. CreateUserForm을 이용하여 입력한 인자들을 user model로 db에 저장 합니다.

index

1. Index 페이지를 출력 합니다.

UserPasswordResetView

1. forgetpwForm을 생성하여 forgetpw.html template의 form과 연동하여 페이지를 출력 합니다.
2. 사용자가 인자를 입력하고 요청을 보낼 경우, 해당 인자를 기반으로 user model의 데이터 중에 일치하는 데이터가 있는지 검색 합니다.
3. 일치하는 데이터가 있는 경우 Django lib의 contrib/auth/PasswordResetView를 이용하여 해당 유저에게 비밀번호 변경을 할 수 있는 url이 담긴 메일을 전송 합니다.
4. 일치하는 데이터가 없는 경우, 같은 페이지에서 에러메시지가 출력 되도록 합니다.

UserPasswordResetDoneView

1. Django lib의 contrib/auth/PasswordResetDoneView를 상속하며, PasswordResetView를 통해 메일이 정상적으로 전송되었을 때, resetpassworddone.html template을 이용하여 화면을 구성하도록 합니다.

UserPasswordResetConfirmView

1. Django lib의 contrib/auth/PasswordResetConfirmView를 상속하며, 사용자가 비밀번호 초기화를 위한 메일의 url을 통해 접속했을 때, password_reset_confirm.html template을 이용하여 비밀번호 초기화 화면이 구성되도록 합니다.
2. SetPasswordForm을 이용하며, 이 Form이 양식에 맞게 잘 입력된 경우 해당 사용자의 User model data를 새로운 비밀번호로 변경합니다.

UserPasswordResetCompleteView

1. Django lib의 contrib/auth/ PasswordResetCompleteView를 상속하며, 사용자가 입력한 비밀번호로 User model data가 잘 변경됐을 때, resetpasswordcomplete.html template으로 화면을 구성합니다.

change_password

1. Django lib의 contrib/auth/ PasswordChangeForm을 이용하여 이전 패스워드와 새로운 패스워드를 입력 받습니다.
2. 이전 패스워드는 사용자를 확인하는 용도로 쓰고, 새로운 비밀번호로 사용자 데이터를 갱신합니다.

search

1. 사용자 입력을 이용하여 게시물 목록에서 해당 입력이 포함된 게시물을 받아옵니다.
2. 게시물들을 화면에 표시하기 좋게 처리하여 검색결과 화면으로 전달합니다.

MySQL 테이블

Figure 9. 은 MySQL 테이블 아키텍처를 보여줍니다. 4 개의 테이블로 게시글, 유저, 북마크 그리고 댓글 테이블로 구성되 있습니다.

게시글 정보를 가지고 있는 테이블

PK : 게시글 작성시 고유의 id 가 생성 됩니다.

post_id/ Type : INT

FK : admin 을 제외한 사용자가 게시글 올리는걸 방지 하기위해 사용 됩니다.

user_id/ Type : INT

유저 정보를 가지고 있는 테이블

PK : 회원가입시 고유의 id 생성 됩니다.

User_id Type : INT

북마크 정보를 가지고 있는 테이블

PK : 즐겨찾기 추가시 고유의 id 가 부여됩니다

Bookmark_id Type : INT

FK : admin 유저의 어떤 게시글을 즐겨찾기 했는지 알 수 있습니다.

user_id Type : INT

Post_id Type : INT

댓글 정보를 가지고 있는 테이블

PK : 댓글 입력시 입력 한 댓글마다 고유의 id 가 부여 됩니다.

Comment_id Type : INT

FK : admin 유저의 어떤 게시글에 댓글을 달았는지 알 수 있음

user_id Type : INT

Post_id Type : INT

post

Num	PK	AI	FK	Null	Logical Name	Name	Type	Default	Comment
1	✓	✓	+	+	post_id	post_id	INT		
2	+	+	+	+	user_id	user_id	INT		
3	+	+	+	+	post_kategorie	post_kategorie	VARCHAR(50)		
4	+	+	+	+	post_title	post_title	VARCHAR(45)		
5	+	+	+	+	post_contents	post_contents	VARCHAR(45)		
6	+	+	+	+	post_create_time	post_create_time	DATETIME		

user

Num	PK	AI	FK	Null	Logical Name	Name	Type	Default	Comment
1	✓	✓	+	+	user_id	user_id	INT		
2	+	+	+	+	user_password	user_password	VARCHAR(45)		
3	+	+	+	+	user_email	user_email	VARCHAR(45)		
4	+	+	+	+	user_first_name	user_first_name	VARCHAR(45)		
5	+	+	+	+	user_last_name	user_last_name	VARCHAR(45)		
6	+	+	+	+	user_create_time	user_create_time	DATETIME		
7	+	+	+	+	user_permissions	user_permissions	INT	0	admin? user=0

bookmark

Num	PK	AI	FK	Null	Logical Name	Name	Type	Default	Comment
1	+	+	+	+	user_id	user_id	INT		
2	+	+	+	+	post_id	post_id	INT		
3	+	+	+	+	post_kategorie	post_kategorie	VARCHAR(45)		
4	+	+	+	+	bookmark_create_time	bookmark_create_time	DATETIME		
5	✓	✓	+	+	bookmark_id	bookmark_id	INT		

comment

Num	PK	AI	FK	Null	Logical Name	Name	Type	Default	Comment
1	+	+	+	+	user_id	user_id	INT		
2	+	+	+	+	post_id	post_id	INT		
3	+	+	+	+	post_title	post_title	VARCHAR(50)		
4	+	+	+	+	comment_create_time	comment_create_time	DATETIME		
5	✓	✓	+	+	comment_id	comment_id	INT		

Figure 9. MySQL 테이블

Use Case

Figure 10. 일반 유저는 검색, 댓글 읽기, 문서 읽기, 피드백 보내기 회원 가입 할 수 있습니다.

회원가입 한 유저는 댓글, 북 마크 쓰기, 수정, 삭제 그리고 비밀번호 재설정 기능을 추가로 사용할 수 있습니다. 관리자는 문서를 쓰기, 삭제 그리고 회원가입 한 유저를 관리 할 수 있습니다.

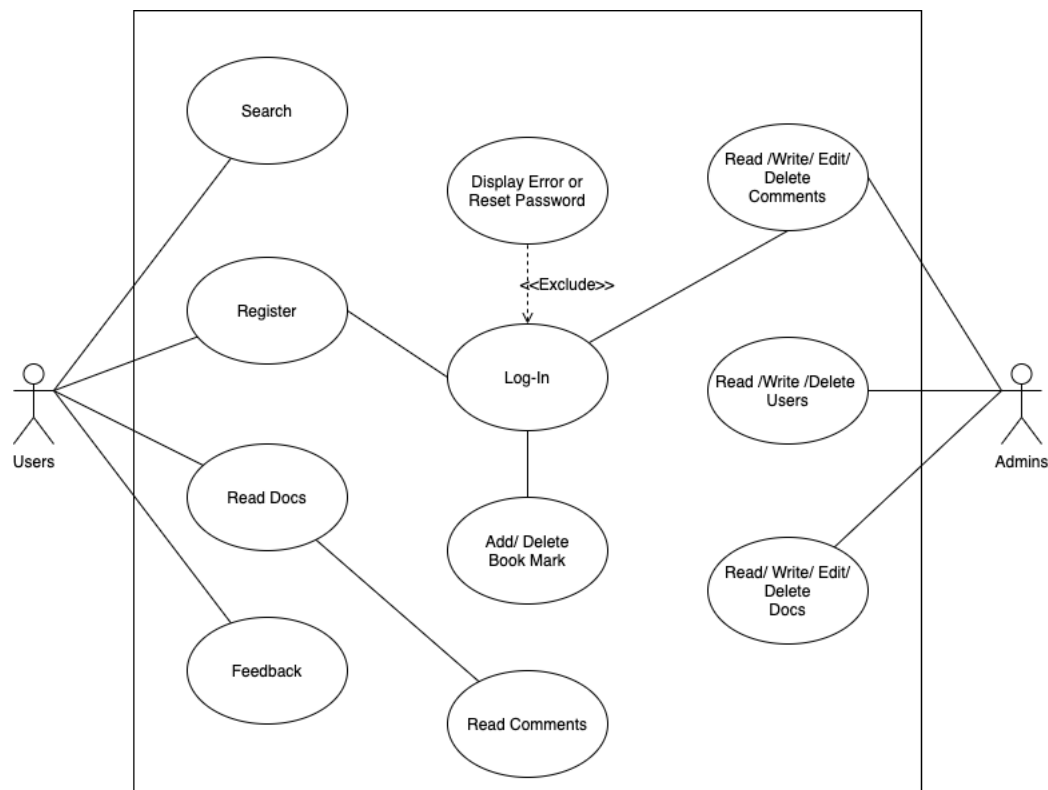


Figure 10. Use Case

Architecture

AWS Architecture

Figure 11. AWS 고 가용성 아키텍처 입니다. 각 노드 그룹은 3 개의 가용영역에 걸쳐서 존재하고, 노드 그룹은 프라이빗 서브넷에 위치해 있습니다. 퍼블릭 서브넷에 NAT Gateway 를 설치하여 인터넷 통신이 가능합니다. 또한 퍼블릭 서브넷에 Bastion host 를 설치하여 노드 그룹에 대한 보안을 강화하며, 부하분산을 트래픽을 유동 적으로 각 각 다른 서버로 분배 해줍니다. 각 노드에는 WiKiKube 가 실행되고 있으며 RDS MySQL 을 DB 로 사용하여 데이터 베이스를 관리합니다. DB 는 쿠버네티스 노드 그룹이 있는 프라이빗 서브넷에 위치하고 있으며, 하나의 RDS 가 활성화 된 상태이고 나머지는 대기상태로 존재 합니다.

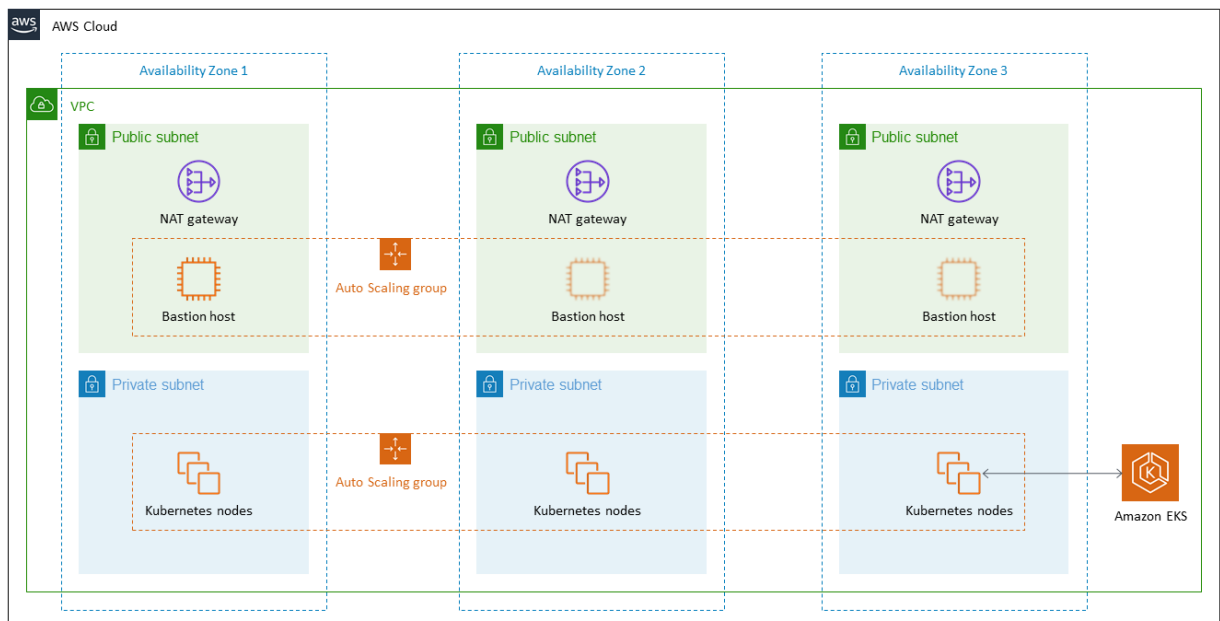


Figure 11. AWS 고 가용성 아키텍처

- 세 개의 가용 영역에 걸쳐 있는 고가용성 아키텍처.*
- AWS 에서 자체 가상 네트워크를 제공하기 위해 AWS 모범 사례에 따라 퍼블릭 및 프라이빗 서브넷으로 구성된 Virtual Private Cloud(VPC).*
- 퍼블릭 서브넷의 경우: 프라이빗 서브넷의 리소스에 대한 아웃바운드 인터넷 액세스를 제공하는 관리형 NAT 게이트웨이.*
- 퍼블릭 서브넷 중 하나의 경우: 프라이빗 서브넷의 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에 대한 인바운드 SSH(보안 셸) 액세스를 제공하는 Auto Scaling 그룹의 Linux

배스천 호스트.* 배스천 호스트는 Kubernetes 클러스터를 관리하기 위한 Kubernetes kubectl 명령줄 인터페이스로도 구성됩니다.

- Kubernetes 컨트롤 플레인을 제공하는 Amazon EKS 클러스터.
- 프라이빗 서브넷의 경우: Kubernetes 노드 그룹.

Feedback form

Figure 12. 아래의 다이어그램에서 고객은 "피드백"양식을 통해 피드백을 제출합니다.

정보는 세 단계로 진행됩니다.

- 피드백 양식은 입력된 정보를 수집하고 Amazon API Gateway Restful Service 에 게시합니다.
- Amazon API Gateway 는 수집 된 정보를 AWS lambda 함수에 전달합니다.
- AWS Lambda 함수는 자동으로 이메일을 생성하고 Amazon SES 사용하여 메일 서버로 보냅니다.

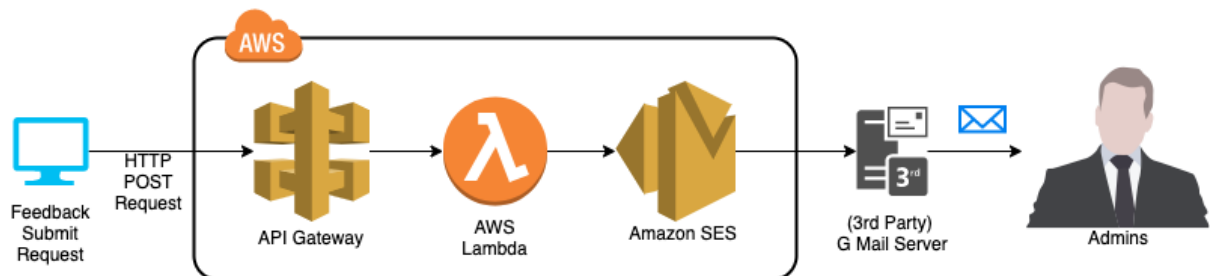


Figure 12. 피드백 아키텍처

Three-tier Applications Architectural Pattern

Figure 13. Three Tier Web Application Architecture 은 3 개의 계층의 웹어플리케이션 아키텍처입니다.

Presentation 계층은에서는 사용자가 직접마주하게 되는 계층입니다. Front-end 을 담당하고, 용자 인터페이스와 관계없는 데이터를 처리하는 로직 등은 포함하지 않습니다. WiKiKube 에서는 Javascript, HTML5, CSS, Bootstrap 그리고 Assets 이 이 계층에 해당 됩니다.

Application 계층은 비즈니스 로직 계층 또는 트랜잭션 계층이라고도 하는데, 비즈니스 로직은 워크스테이션으로부터의 클라이언트 요청에 대해 마치 서버처럼 행동한다. 차례로 어떤 데이터가 필요한지를 결정하고, 메인 프레임 컴퓨터 상에 위치하고 있을 세 번째 계층의 프로그램에 대해서는 마치 클라이언트처럼 행동합니다. back-end Python, Django Framework 가 해당 됩니다.

- 프레젠테이션코드나 데이터 관리 코드는 포함 되지 않습니다.

-주로 어플리케이션 서버를 뜻합니다.(물리적 : WAS 서버)

Data 계층은 데이터베이스와 액세스해서 읽거나 쓰는 것을 관리하는 프로그램을 포함합니다. 애플리케이션의 조직은 이것보다 더욱 복잡해질 수 있지만, 3 계층 관점은 대규모 프로그램에서 일부분에 관해 생각하기에 편리한 방법입니다. RDS MySQL 이 이 계층에 포함됩니다.

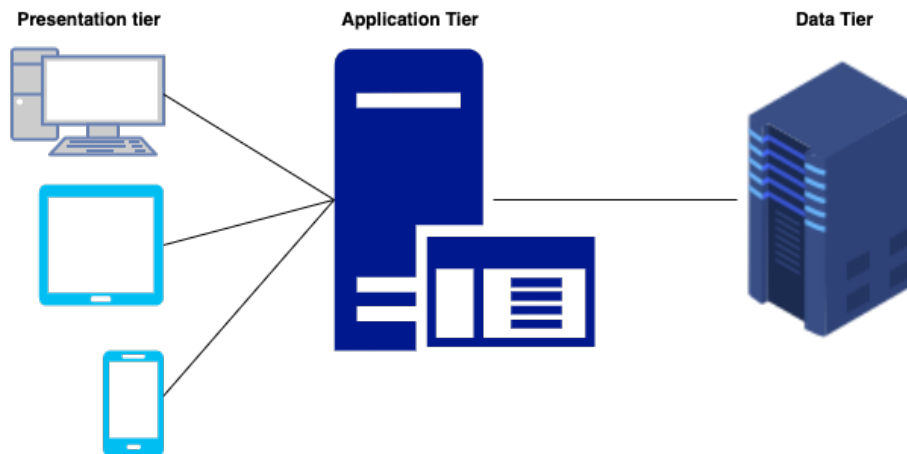


Figure 13. Three Tier Web Application Architecture

Gantt Chart

Figure 14. 크게 6 가지로 나뉜 단계별 프로젝트 진행사항 타임 테이블 입니다.

5 월 24 일 - 6 월 02 일일 웹사이트 (프론트, 백 엔드) 개발

6 월 02 일 – 6 월 07 일 도커 이미지 생성 도커파일 작성

6 월 04 일 – 6 월 14 일 EKS 클러스터 배포

6 월 10 일 – 6 월 16 일 클라우드 와치 배포 및 클러스터 모니터링

5 월 28 일 – 6 월 17 일 CI/CD 개발

주요 일정	24	25	26	27	28	31	1	2	3	4	7	8	9	10	11	14	15	16	17	18	21
웹사이트																					
도커 이미지 생성																					
EKS																					
모니터링																					
CI/CD																					
PPT 및 프로젝트 문서화																					

Figure 14. Gantt Chart

Reference

EKS Doc 08.06.2021

<http://eks.io/>

AWS EKS 01.06.2021

<https://aws.amazon.com/quickstart/architecture/amazon-eks/>

iDoc Front-End 28.05.2021

<https://github.com/harnishdesign/iDocs>

K8s Doc 24.05.2021

<https://kubernetes.io/ko/docs/home/>

Django Doc 23.05.2021

<https://www.djangoproject.com/>