**오늘 할 일 공유 플랫폼**

팀명 : 1황5김

팀장 : 김민식

팀원 : 김동영

김지원

**목 차**

[1. 프로젝트 개요 4](#_Toc98922908)

[1.1 프로젝트 목표 4](#_Toc98922909)

[1.2 팀원 및 역할 4](#_Toc98922910)

[1.3 프로젝트 일정 5](#_Toc98922911)

[1.4 프로젝트 환경 6](#_Toc98922912)

[1.5 부분별 기대효과 7](#_Toc98922913)

[1.5.1 현황 (AS-IS) 7](#_Toc98922914)

[1.5.2 개선 및 기대효과 (TO-BE) 7](#_Toc98922915)

[1.6 시스템 아키텍처 8](#_Toc98922916)

[1.6.1 전체 구성도 8](#_Toc98922917)

[1.6.2 인프라 구성도 8](#_Toc98922918)

[1.6.3 프로젝트 관리 방안 9](#_Toc98922919)

[2. 프로젝트 내용 1](#_Toc98922920)0

2.1 인프라 설계   
 2.1.1 EKS 클러스터 구축……………………………………………………………………………………..………...10

[2.1.2 Route 53.................................................................................................................................................13](#_Toc98922925)

[2.1.3 CI…….........................................................................................................................................................17](#_Toc98922925)  
2.1.3 CD…………………………………………………………………………………………………………………………..27

2.1.4 Monitorig………………………………………………………………………………………………………………..38

[2.2 애플리케이션 설계](#_Toc98922924)

[2.2.2 Front-end………………….. .. …… .. 4](#_Toc98922931)1

2.2.3 Back-end…………………………………………………………………………………………………………………41

[3. 테스트 및 결론](#_Toc98922932) 49

[4. 후기](#_Toc98922933) 50

**표 목 차**

[[표 1‑1] 팀원 및 역할 4](#_Toc98921842)

[[표 1‑2] 프로젝트 일정 5](#_Toc98921843)

[[표 1‑3] 프로젝트 환경 6](#_Toc98921844)

[표 1-4] EKS 클러스터 구성…………………………………………………………………………………………………10

[[표 2‑1] API 명세 47](#_Toc98921845)

[[표 2‑2] 후기……..………………………………………………………………………………………………………………...](#_Toc98921845) 50

# **프로젝트 개요**

* 1. 프로젝트 목표

**기존 goorm 쿠버네티스 전문가 양성과정을 수강하면서 추가적인 학습을 진행하는 학생들이 상당수 존재한다. 허나, slack과 notion만으로는 전체 수강생들의 공부 내역을 한 눈에 알아보기 힘든 점이 존재한다. 수강생들의 공부 내역 전체를 확인하지 못하면, 온라인과 장기간의 공부에 따른 피로감이 누적된 수강생들에게 더 이상의 학습 동기부여를 유발하지 못한다는 점으로 확산될 수 있다. 이러한 문제점을 개선하기 위해 수강생들이 각 어떤 목표를 가지고 오늘의 할 일을 진행하였는지 한 눈에 알아보기 쉽게 하여 수강생들의 학습 의지 증가를 유발할 수 있다. 또한, 다른 수강생의 학습 계획표를 탐구할 수 있어, 정규시간 이외에 추가적인 학습 방향을 제대로 잡지 못한 수강생들에게 도움을 줄 수 있다. 이에 최종적으로 학습에 대한 동기부여 유도를 기대할 수 있는 서비스를 기획하였다.**

* 1. 팀원 및 역할

아래 표는 팀 구성원의 담당 역할입니다.

|  |  |
| --- | --- |
| 이름 | 역할 |
| 김민식 | - 팀장  - 웹서비스 개발 담당 |
| 김동영 | - 웹서비스 개발 담당 |
| 김지원 | - 인프라 구성 담당 |

[표 ‑] 팀원 및 역할

* 1. 프로젝트 일정

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1주차 | | 2주차 | | 3주차 | | 4주차 | | 5주차 | |
| 자료 수집 |  |  |  |  |  |  |  |  |  |  |
| 인프라 구축 |  |  |  |  |  |  |  |  |  |  |
| 서비스 구축 |  |  |  |  |  |  |  |  |  |  |
| CI/CD 구축 |  |  |  |  |  |  |  |  |  |  |
| 모니터링/테스트 |  |  |  |  |  |  |  |  |  |  |
| 보고서 작성 |  |  |  |  |  |  |  |  |  |  |

[표 ‑] 프로젝트 일정

**1주차 - 주제에 대한 논의 및 결정, 요구사항 분석 및 설계, 인프라 구성도와 ERD 및 API와 화면 설계**

**2주차 - 인증서버 구축 및 API 개발, CI/CD 및 RDS 연동, ALB 구축**

**3주차 - Route 53 및 https 설정, Front-end 개발, API 구현**

**4주차 – Front-end / Back-end 연동, 오류수정 및 모니터링 구축**

**5주차 – 시연영상 제작, 문서정리, 보고서 작성**

* 1. 프로젝트 환경

아래 [표 1-3]의 프로그램들을 이용하여 프로젝트를 진행하였다.

|  |  |  |
| --- | --- | --- |
| **k8s Managed Service** |  | Elastic Kubernetes Service |
| **CI / CD** |  | Jenkins / ArgoCD |
| **Monitoring** |  | Prometheus / Grafana |
| **Web Framework** |  | Spring boot 3.0.4 |
| **Lauguage** | JS] 자바스크립트 공식사이트 | Java 17  JavaScript |

[표 ‑3] 프로젝트 환경

* 1. 부분별 기대효과
     1. 현황 (AS-IS)

✅ 10~19시 까지 이어지는 강의를 마치면 피곤하고 나태해짐

✅ To do list를 혼자 진행 시, 쉽게 포기 하거나 미루는 현상 발생

✅ 다른 수강생들은 어떤 공부를 얼마나 하는지에 대한 정보의 부재

✅ 강의시간 이후의 학습 방향을 잡지 못하는 교육생 증가

* + 1. 개선 및 기대효과 (TO-BE)

✅ 본인이 작성한 to do list를 이용자들 간 공유함으로써 적극적인 참여 유도

✅ 이용자들간 서로 소통이 가능한 게시판 구축

✅ 다른 수강생의 학습 진행 현황 확인 -> 동기부여 유발 및 학습 효과 증가 기대

✅ 다른 이용자의 추가 학습 계획을 탐구하여 자신의 학습 계획 자아성찰 가능

* 1. 시스템 아키텍처
     1. 전체 구성도

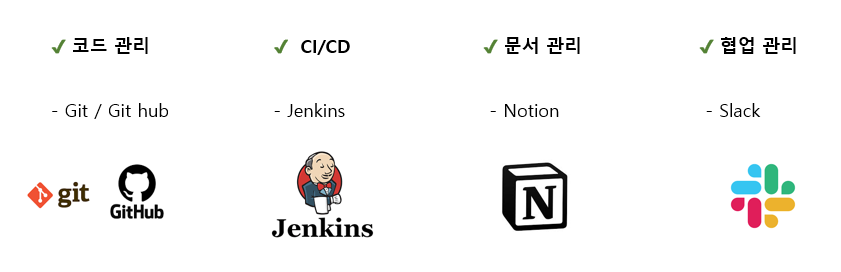
도표이(가) 표시된 사진

자동 생성된 설명

* + 1. 인프라 구성도

# 도표이(가) 표시된 사진 자동 생성된 설명

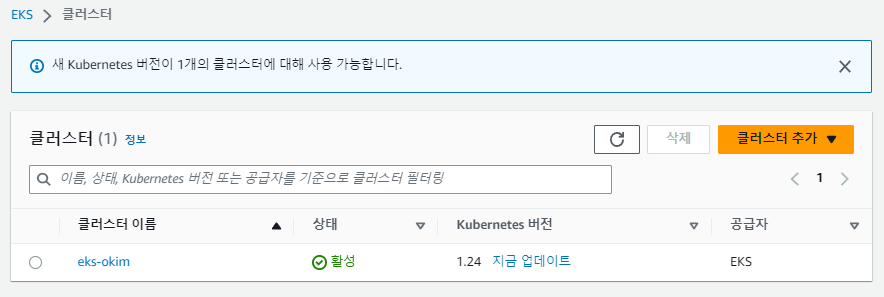
* + 1. 프로젝트 관리 방안



# **프로젝트 내용**

2. 1. 인프라 설계
      1. 네트워크 구성

a. EKS(Elastic Kubernetes Service) 클러스터 생성

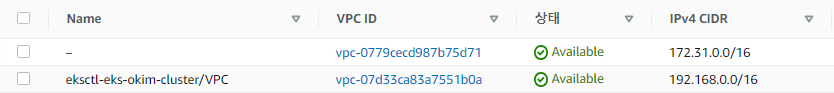


a-1. EKS spec

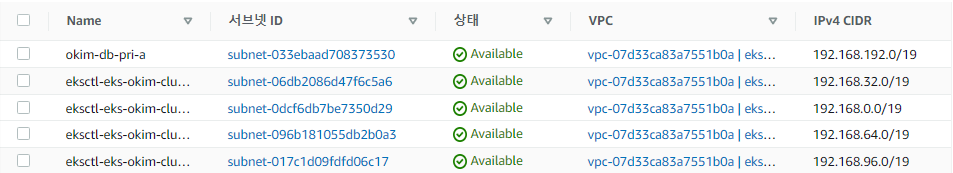
|  |  |
| --- | --- |
| var.cluster\_name | eks-okim |
| var.cluster\_version | 1.24 |
| worker\_group\_name | mynodes-t3 |
| worker\_group\_instance\_type | t3.medium |
| autoscaling\_group\_min\_size | 2 |
| autoscaling\_group\_desired\_capacity | 2 |
| autoscaling\_group\_max\_size | 5 |

[표 1-4] EKS 클러스터 구성

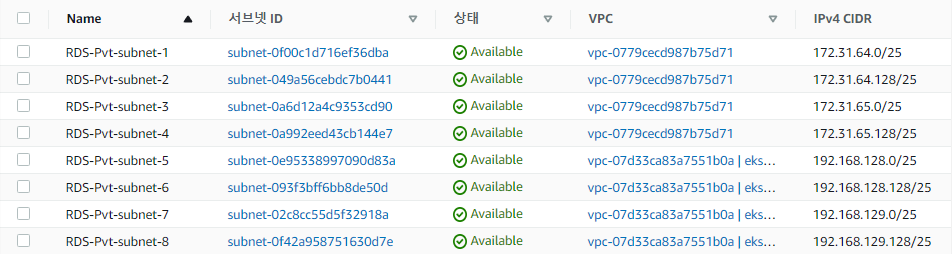
b. 운영환경 VPC 생성



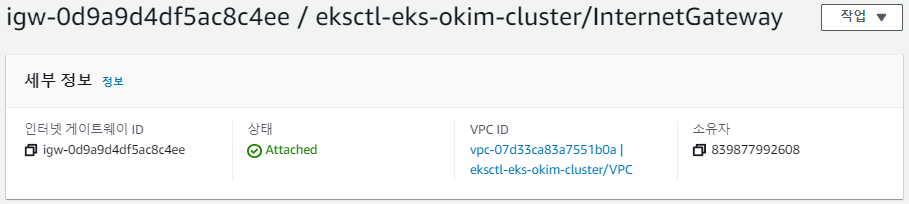
c. Subnet 구성



d. RDS Subnet 구성



e. Internet Gateway 생성



f. NAT Gateway 생성



g. Routing Table 생성



* + 1. Route 53

a. 웹서비스용 도메인 생성  
웹사이트이(가) 표시된 사진

자동 생성된 설명

- 웹서비스 접속을 이용자들이 쉽게 할 수 있도록 도메인 주소 생성

b. 레코드 생성

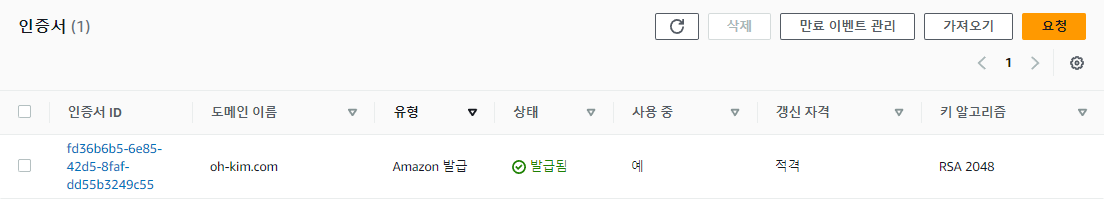
텍스트이(가) 표시된 사진

자동 생성된 설명

c. 인증서 발급 및 확인

텍스트이(가) 표시된 사진

자동 생성된 설명



d. 웹서비스용 도메인에 LoadBalnacer 설정

텍스트이(가) 표시된 사진

자동 생성된 설명

**e. LoadBalancer 보안그룹에 HTTPS 인바운드 규칙 추가**



**f. LoadBalancer에 https 연동**

**텍스트이(가) 표시된 사진

자동 생성된 설명**

- 웹서비스에 안전한 접속을 위해 https 연결 활성화

**g. http와 https 리다이렉팅**

**텍스트이(가) 표시된 사진

자동 생성된 설명**http로 - 접속해도 자동으로 https 주소로 변환하게끔 설정

**h. 인증서 서버와 api 서버의 도메인 생성**

****

****

**i. 인증서, api 서버 도메인에 LoadBalancer 설정**

**텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명**

- 인증서 서버와 api 서버를 각각의 로드밸런서에 연결하여 차후 웹서비스가 api를 가져올 때   
Mix content가 출력되는 것을 방지한다

* + 1. CI / CD

도표이(가) 표시된 사진

자동 생성된 설명

1. Jenkins에 github webhook 구성하기
   1. github – settings에 들어간다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. generate new token – classic

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. repo와 hook를 선택한다. 토큰을 잘 저장한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. webhook를 설정하고 싶은 레포지토리에 간다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. settings - wehbook - add webhook을 누른다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. http://젠킨스 주소/github-wehbook/ 을 추가한다.
  2. 젠킨스의 파이프 라인 - 구성에서 다음 항목들을 체크한다
* GitHub hook trigger for GITScm polling
* github project url

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. private 레포지토리인 경우 credential에서 다음 항목들을 구성해야 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Jenkins 슬랙 알람
   1. <https://my.slack.com/services/new/jenkins-ci> 접속
   2. 젠킨스 알림을 포스트할 채널을 선택하고 젠킨스 CI 통합 앱 추가

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 젠킨스 서버에서 젠킨스 관리- 플러그인 관리에서 slack notification 설치

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 슬랙 알림을 설정을 하기위해 젠킨스 관리 - 시스템 설정에 접속한다.

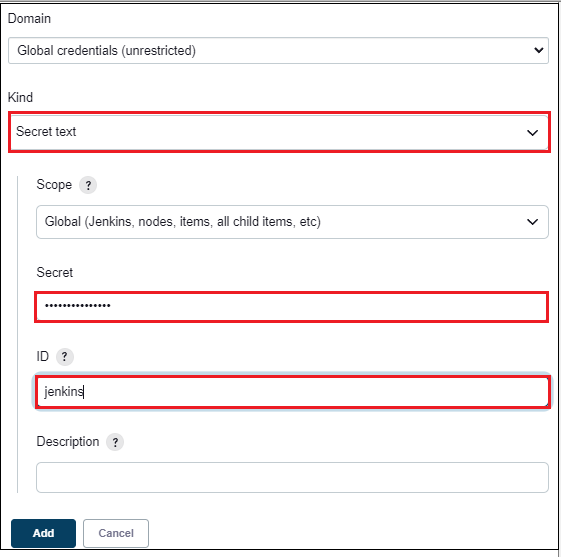
텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 맨 하단에 슬랙 카테고리가 있다. workspace는 슬랙의 매뉴얼 페이지에 있는 팀 하위 도메인을, credential에 secret text 형태로 통합 토큰 자격 증명 ID를 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명



* 1. 구성된 화면은 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. test connection를 눌러 테스트 후 저장한다.
  2. 왼쪽 상단의 새 item – pipeline

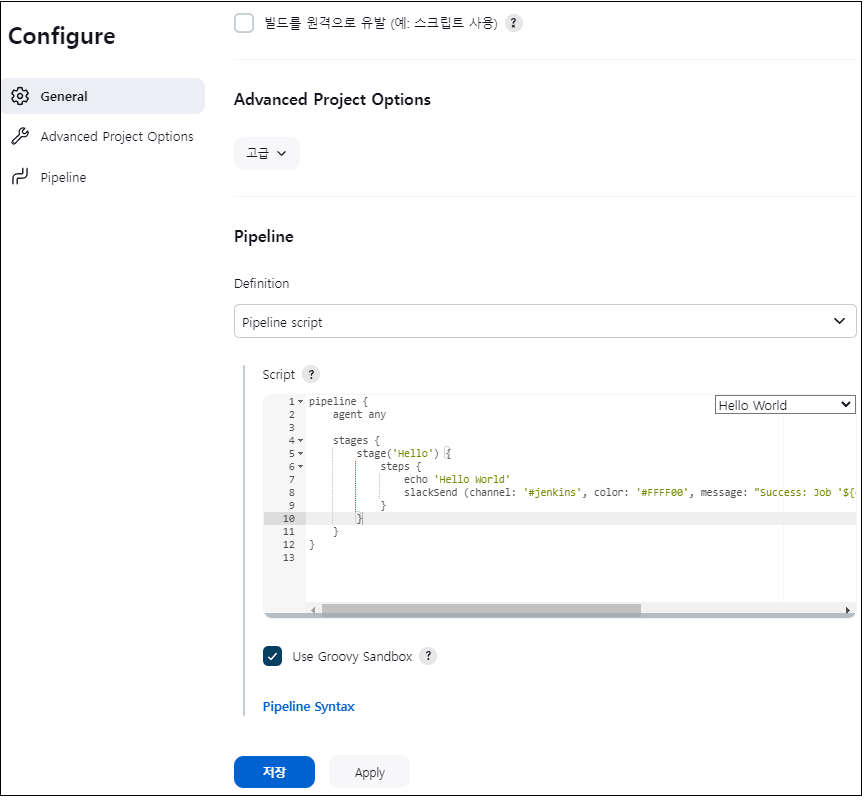
텍스트, 편지이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 간단한 슬램 알림 전송 기능이다. 알림을 보낼 채널, 색상, 메시지의 수정이 가능하다

slackSend (channel: '#jenkins', color: '#FFFF00', message: "Success: Job '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]' (${env.BUILD\_URL})")



* 1. 해당 스크립트를 저장한다. 지금 빌드를 눌러 파이프라인이 작동하는지 확인한다.

테이블이(가) 표시된 사진

자동 생성된 설명



1. Argo + Github 연동
   1. ArgoCD – Settings

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. Reposotories - connect Repo 클릭

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. VIA HTTPS, GIT, github URL, username과 password를 채운다

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

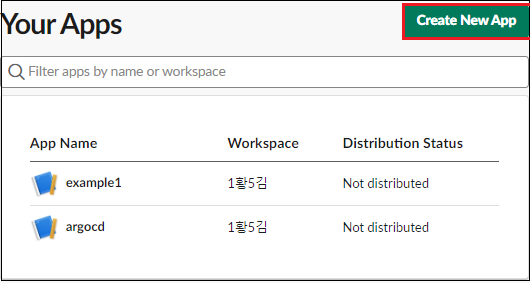
자동 생성된 설명

1. ArgoCD 슬랙 및 APP
   1. Argo CD Notifications 와 Triggers and Templates from the catalog 를 설치한다. 설치할 때 namespace는 argocd로 한다

$ kubectl apply -n argocd -f <https://raw.githubusercontent.com/argoproj-labs/argocd-notifications/release-1.0/manifests/install.yaml>

$ kubectl apply -n argocd -f <https://raw.githubusercontent.com/argoproj-labs/argocd-notifications/release-1.0/catalog/install.yaml>

* 1. 슬랙으로 접속해 새 앱을 생성한다
     1. <https://api.slack.com/apps?new_app=1>



* 1. From scratch를 선택한다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 앱의 이름과 workspace를 지정한다. app name: argoCD, workspace는 본인의 워크스페이스를 지정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. Oauth & Permissions를 선택한다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. Scopes - chat:write를 선택한다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. install app to workspace를 누른다

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

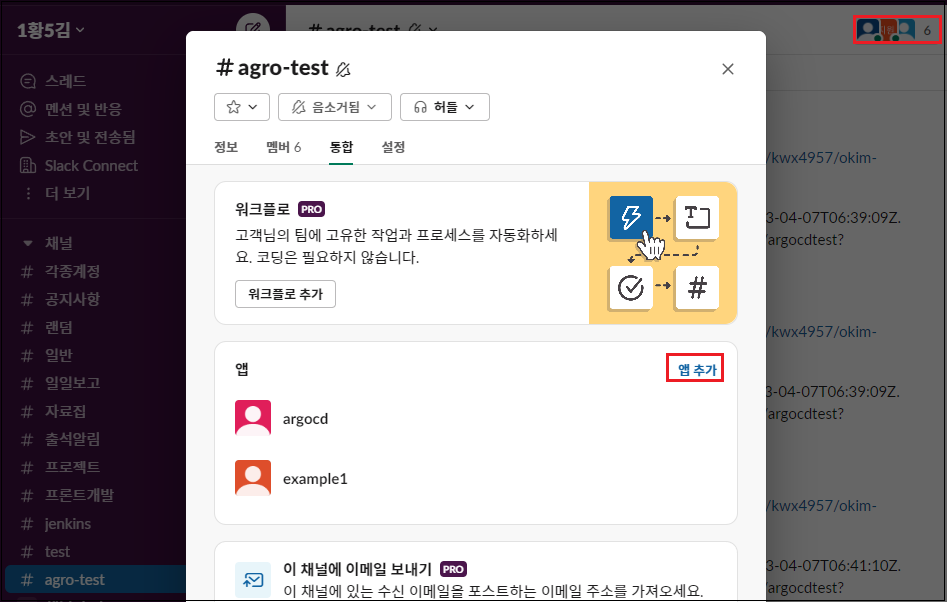
자동 생성된 설명

* 1. 생성된 토큰은 잘 복사해둔다

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. argo-test 채널로 클릭하여 해당 채널로 이동한다. 화면의 오른쪽 상단에 보이는 버튼을 누르면 다음과 같은 창이 뜬다



* 1. 앱 추가를 누른다. argocd 앱을 추가한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* + 1. 슬랙의 토큰을 넣는다

$ kubectl apply -f argocd-notifications-secret.yaml -n argocd

텍스트이(가) 표시된 사진

자동 생성된 설명

$kubectl edit cm argocd-notifications-cm -n argocd

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. argocd 홈페이지에서 APP 생성한다

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 모니터, 화면이(가) 표시된 사진

자동 생성된 설명

* 1. kubectl get app -n argocd # 현재 실행중인 argocd app을 보여준다
  2. 해당 앱이 싱크가 성공시, 시작시, 실패시 슬랙에 알림을 보내게 설정한다.

$ kubectl patch app 앱이름 -n argocd -p '{"metadata": {"annotations": {"[notifications.argoproj.io/subscribe.on-sync-running.slack":](http://notifications.argoproj.io/subscribe.on-sync-running.slack%22:%22agro-test)”알림을 보낼 채널명"}}}' --type merge

$ kubectl patch app 앱이름 -n argocd -p '{"metadata": {"annotations": {"[notifications.argoproj.io/subscribe.on-sync-succeeded.slack":"](http://notifications.argoproj.io/subscribe.on-sync-succeeded.slack%22:%22agro-test)알림을 보낼 채널명"}}}' --type merge

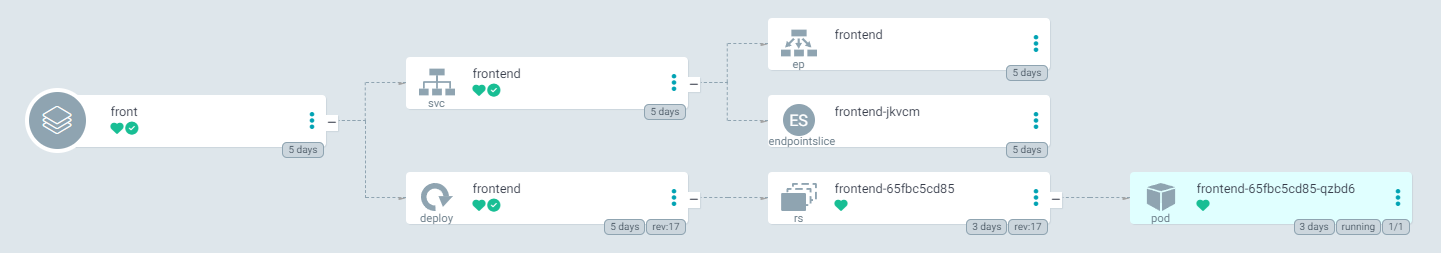
$ kubectl patch app 앱이름 -n argocd -p '{"metadata": {"annotations": {"[notifications.argoproj.io/subscribe.on-sync-failed.slack":"](http://notifications.argoproj.io/subscribe.on-sync-failed.slack%22:%22agro-test)알림을 보낼 채널명"}}}' --type merge

**Argo CD에서 배포중인 목록**

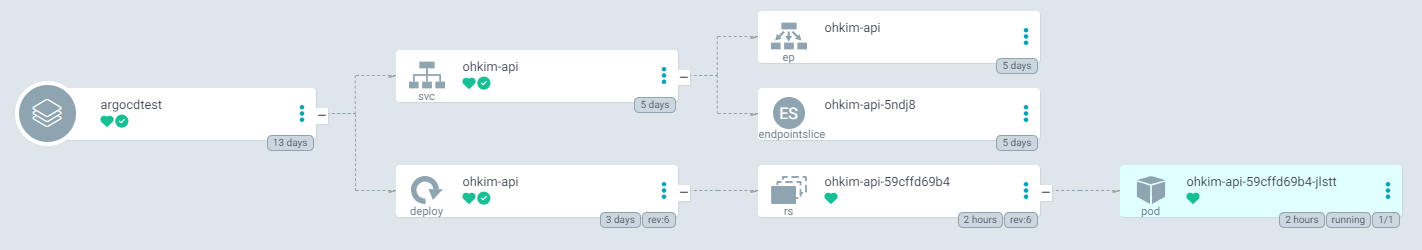
텍스트이(가) 표시된 사진

자동 생성된 설명

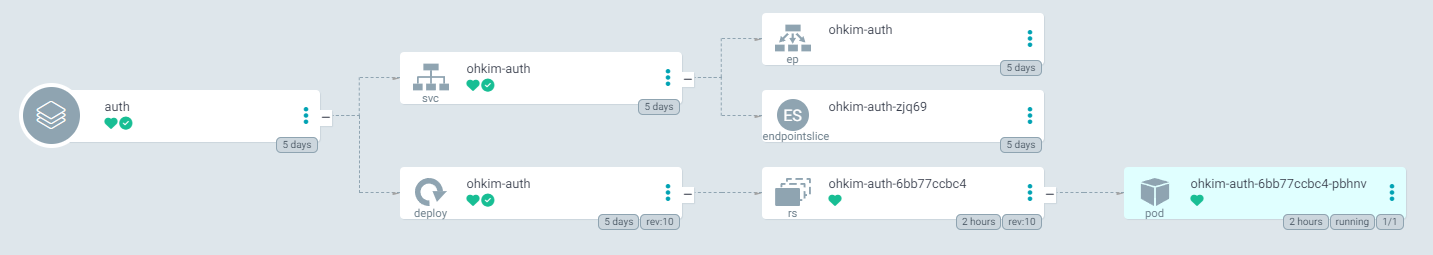
**- front deployment, service 구조**



**- api deployment, service 구조**



**- auth deployment, service 구조**



* + 1. Prometheus / Grafana

1. helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
2. helm repo update
3. kubectl create ns monitor
4. prometheus-grafana.yaml

grafana:

service:

type: LoadBalancer

1. helm install prometheus prometheus-community/kube-prometheus-stack -n monitor   
   -f prometheus-grafana.yaml

텍스트이(가) 표시된 사진

자동 생성된 설명

1. Prometheus-node monitoring

텍스트, 모니터, 스크린샷, 실내이(가) 표시된 사진

자동 생성된 설명

- 클러스터 노드 별 Pod 리소스 모니터링(CPU사용량, 메모리 사용량 등…)

1. Prometheus-kubelet monitoring

텍스트, 득점판, 스크린샷이(가) 표시된 사진

자동 생성된 설명

- 동작중인kubelet 개수, 파드, 컨테이너 등 확인

**2.1.4 elastic**

- elastic을 구성하여 <https://oh-kim.com> 에 대한 로그를 기록, 열람

a. elastic – Logs – Stream – message: GET 검색

텍스트이(가) 표시된 사진

자동 생성된 설명

b. elastic – Logs – Stream

텍스트이(가) 표시된 사진

자동 생성된 설명

* 1. 애플리케이션 설계
     1. Front-End

**a. 프로젝트 주소** <https://github.com/kwx4957/okim-front>

**b. 기술 스택**

1. Package manager : npm 9.5.0  
 2. SPA Framwork  
 2-1. Vue.js  
 ◾ Vuex : Vue.js 상태관리 라이브러리  
 ◾ Vue Router : Vue.js 라우팅 라이브러리  
 ◾ Vue-cli : Vue.js 공식 개발 도구, Vue.js build 도구로 활용

3. CSS  
 ◾ Tailwind CSS : CSS 유틸리티 프레임워크  
 ◾ DaisyUI : Tailwind CSS를 확장하여 디자인 요소를 제공하는 라이브러리

**c. 화면 구성**

◾ 메인화면

◾ 로그인화면

◾ 프로필수정화면

◾ 회원가입화면

◾ 그룹별 TASK 들을 볼 수 있는 화면

◾ 유저별 TASK 들을 볼 수 있는 화면

◾ TASK 의 세부적인 정보를 볼 수 있는 화면

◾ TASK를 작성하고 내부 항목들을 수정할 수 있는 화면

**d. 디렉토리 구조**

├── Dockerfile -> Vue.js Docker 이미지를 생성하기 위한 스크립트 파일

├── README.md

├── babel.config.js -> Babel 을 사용할 때 사용자 정의 설정을 제공하는 파일

├── dist -> 최종 빌드 결과물

├── jsconfig.json -> Js 프로젝트의 모듈 해결 방법과 파일 구조를 구성하도록 돕는 파일

├── nginx.conf -> Dockerfile 빌드시 복사해 넣을 nginx 설정 파일

├── node\_modules -> 의존성 라이브러리 저장하는 디렉토리

├── package-lock.json

├── package.json -> 패키지 관리 파일

├── postcss.config.js -> PostCSS 사용할 때 사용자 정의 설정을 제공하는 파일

├── public -> 프로젝트 소스 파일

├── src -> 프로젝트 소스 파일

├── tailwind.config.js -> tailwind css 사용할 때 사용자 정의 설정을 제공하는 파일

└── vue.config.js -> Vue CLI 을 사용하여 Vue.js 프로젝트를 구성할 때 사용하는 설정 파일. 이 파일을 사용하여 Vue CLI의 동작 방식, 웹팩 설정, 개발 및 프로덕션 빌드 구성 등을 사용자 정의할 수 있음

**현재 디렉토리 : ./src**

├── App.vue : Vue Root Component(최상위 컴포넌트)

├── api : 서버와 통신하기 위한 각종 api 모듈들 모음

├── assets : 서비스에 필요한 각종 이미지들 모음

├── components : 컴포넌트

├── index.css : tailwind css 파일

├── main.js : App.vue instance 생성 및 html 에 mount 설정하는 파일

├── router : 각종 라우팅 정보가 모여있는 디렉터리

├── services : 비지니스 로직을 처리하기 위한 로직이 담겨있는 디렉터리

├── store : vue 상태관리를 위한 디렉터리

├── utils : 각종 유틸리티 들을 모아놓은 디렉터리

└── views : components 들이 조합되어 이루어질 기본 바탕이 되는 views 폴더

**e. 배포 방식**

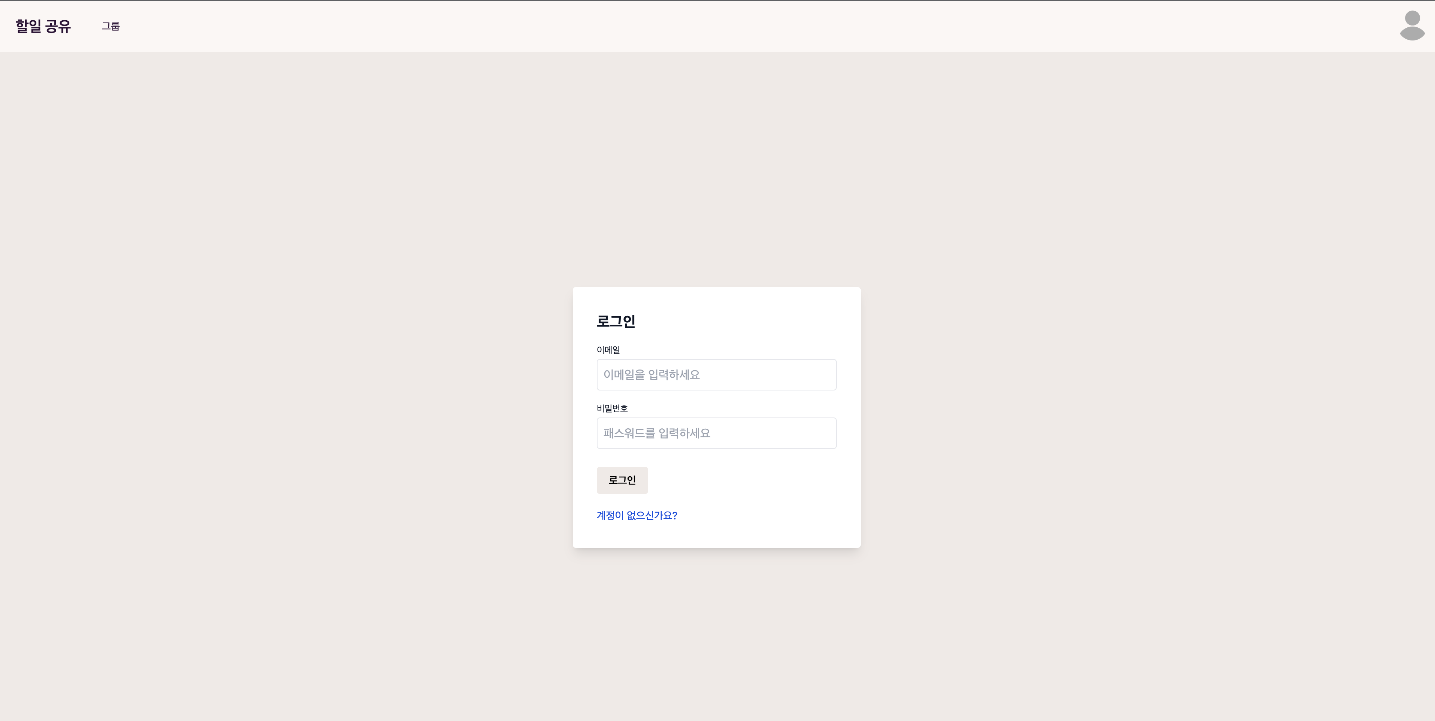


a. 메인화면

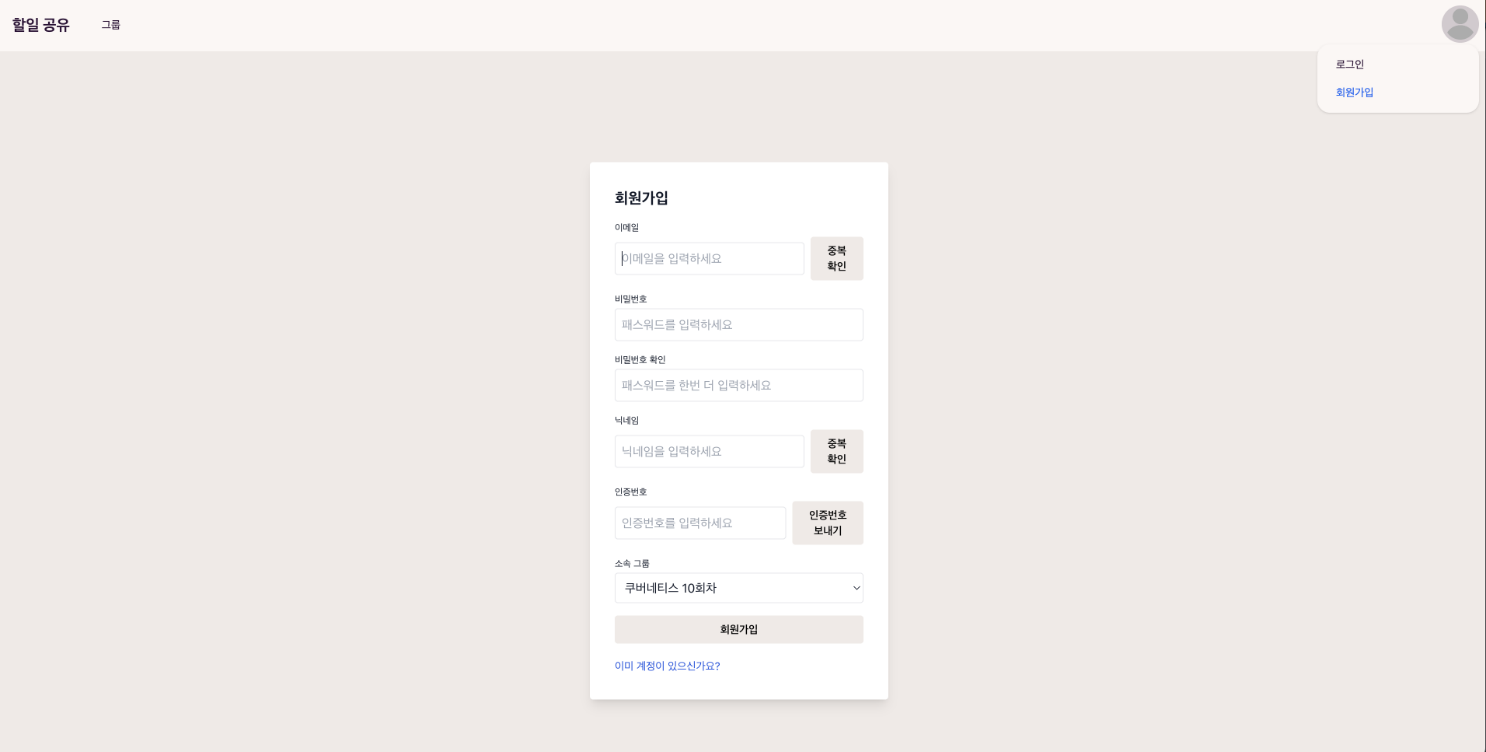
텍스트, 편지이(가) 표시된 사진

자동 생성된 설명

**b. 로그인 화면**

****

**c. 회원가입 화면**

****

**d. 그룹 현황**

텍스트, 전자제품, 잭, 스크린샷이(가) 표시된 사진

자동 생성된 설명

**e. 프로필 수정**



**f. 할 일 목록**

텍스트, 전자제품, 잭, 주방가전이(가) 표시된 사진

자동 생성된 설명

**2.2.2 Back-end**

**a. Github 주소**

◾ 인증서버 : <https://github.com/kwx4957/okim-auth>  
◾ API 서버 : <https://github.com/kwx4957/okim>

**b. 기술 스택**

1. 웹 프레임워크

◾ Spring Boot 3.0.4, Spring MVC, Spring Security

# 2. 라이브러리

◾ Spring Data JPA, Hibernate 6.1.7, Querydsl Jpa 5.0

# 3. 데이터베이스

◾ MySQL 8.0

◾ Redis

4. 테스팅 라이브러리

◾ Junit5

5. 빌드도구

◾ Gradle

6. 도구 및 툴

◾ Postman

◾ Docker

7. 기타

◾ AWS S3 : 회원 프로필 이미지 저장

**c. API 명세**

|  |  |  |  |
| --- | --- | --- | --- |
| **카테고리** | **METHOD** | **PATH** | **설명** |
| 테스크 | GET | /api/v1/groups/{group\_id}/tasks | 특정 그룹의 테스크 리스트 조회 |
| 테스크 | GET | /api/v1/groups/{group\_id}/tasks/status?day={day\_of\_week} | 특정 그룹의 유저 테스크 완료 리스트 조회 |
| 테스크 | GET | /api/v1/user/{user\_id}/tasks | 특정 유저의 테스트 리스트 조회 |
| 테스크 | GET | /api/v1/tasks/{task\_id} | 특정 유저의 테스크 조회 |
| 테스크 | POST | /api/v1/task | 테스크 생성 |
| 그룹 | GET | /api/v1/groups | 전체 그룹 리스트 조회 |
| 그룹 | GET | /api/v1/groupname | 그룹명 리스트 조회 |
| 아이템 | POST | /api/v1/item | 아이템 생성 |
| 아이템 | PUT | /api/v1/items/{itemId}/done | 아이템 완료/미완 수정 |
| 아이템 | PUT | /api/v1/items/{itemId} | 아이템 수정 |
| 아이템 | DELETE | /api/v1/items/{itemId} | 아이템 삭제 |
| 유저 | GET | /api/v1/user/{user\_id} | 유저 정보 조회 |
| 유저 | POST | /api/v1/user/login | 유저 로그인 |
| 유저 | POST | /api/v1/user/signup | 유저 회원가입 |
| 유저 | GET | /api/v1/email/{email}/check | 이메일 중복 확인 |
| 유저 | POST | /api/v1/email | 이메일 인증번호 보내기 |
| 유저 | GET | /api/v1/email/validation | 이메일 인증번호 확인 |
| 유저 | GET | /api/v1/ninckname | 닉네임 중복확인 |
| 유저 | PUT | /api/v1/user/{user\_id} | 유저 프로필 수정 |

**d. 배포방식**

**1.** 컨테이너로 애플리케이션을 배포

**2.** Dockerfile 에 컨테이너로 애플리케이션을 배포하기 위한 스크립트 작성

**3.** 실제 배포 시, Dockerfile 이 포함된 소스코드를 github으로 부터 다운받아 Dockerfile 을 이용해 애플리케이션 컨테이너 이미지 제작 및 배포함

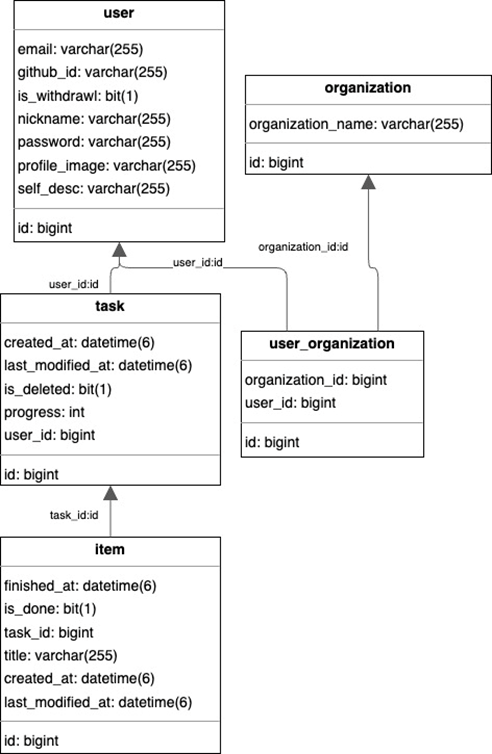
**4.** Dockerfile 의 세부적인 정보는 아래와 같음

a. FROM eclipse-temurin:17-jdk-alpine

b. COPY /build/libs/\*.jar app.jar

c. ENTRYPOINT [“java”, “-jar”, “/app.jar”]

**e. DB 스키마**

****

user : 클라이언트의 회원 정보를 관리하는 테이블(이메일, 비밀번호, 프로필 이미지 링크 등…)

task : 유저가 작성한 TASK(할 일)을 기록, 관리하는 테이블

organization : 그룹을 관리하는 테이블 (예, 쿠버네티스 10회차, 11회차), 사용자는 각 그룹에 포함되어 있다.

item: TASK (할 일) 하위의 item(항목) 을 관리하는 테이블, 각 항목은 완료여부를 컬럼으로 가지고 있으며, 사용자가 항목별로 완료여부를 체크할 때 사용된다.

user\_organization: 그룹(organization)과 사용자(user) 를 관리하는 테이블

**3. 테스트 영상 및 결론**

Application :

회원가입 -> 이메일인증 -> 로그인 -> 프로필 수정 -> 태스크 작성 -> 그룹 별 태스크 확인 -> 사용자별 태스크 목록 조회 -> 나의 할일 보기 -> 그룹 현황보기 -> 종료

CI/CD : <https://drive.google.com/file/d/19E42gG-Y0vLAj2CYRl9srxnubRiBimfb/view>

소스코드 수정 -> 깃헙 업로드 -> Jenkins 잡 시작 -> 소스코드 빌드 -> 슬랙 알람 -> 도커이미지 빌드 -> 이미지 업로드 -> 배포파일 리소스버전 수정 -> 슬랙 알람 -> 리소스 업데이트

Monitoring: <https://drive.google.com/file/d/17-PJ5PAdW2Jx70leedFRxqcc870_HIar/view>

Eksctl ClusterConfig로 Kubernetes 버전, 가용영역, 인스턴스 타입을 정하고 클러스터, 노드를 생성 후  
cloudwatch, autoscaler, albingress 등 AddonPolicy를 추가하여 사용 가능하도록 했다. Deployment, service, ingress, RDS, HPA Autoscaling, Route53, Https, Cloudwatch, LoadBalancer 등을 통해 인프라를 구축하고  
Spring / Vue로 만든 Front-end, Api-server, Auth-server 애플리케이션을 도커라이징하여 Jenkins / ArgoCD 를 통해 배포 후 Prometheus / Grafana를 사용해 클러스터와 노드의 리소스 사용량, kubelet 로그 등을 모니터링, ELK를 통해 애플리케이션을 모니터링 했다.

# 참고자료

[1] Route 53를 사용 중인 도메인에 대한 DNS 서비스로 설정

<https://docs.aws.amazon.com/ko_kr/Route53/latest/DeveloperGuide/migrate-dns-domain-in-use.html>

[2] ACM 인증서 발급

<https://docs.aws.amazon.com/ko_kr/acm/latest/userguide/gs-acm-request-public.html>

[3] Kubernetes Ingress

<https://kubernetes.io/ko/docs/concepts/services-networking/ingress/>

[4] Kubernetes Service

<https://kubernetes.io/ko/docs/concepts/services-networking/service/>

[5] Kubernetes Deployment

<https://kubernetes.io/ko/docs/concepts/workloads/controllers/deployment/>

[6] https 리다이렉션

<https://medium.com/@yangga0070/aws-%EB%A1%9C%EB%93%9C%EB%B0%B8%EB%9F%B0%EC%84%9C-http-https-%EB%A6%AC%EB%8B%A4%EC%9D%B4%EB%A0%89%EC%85%98-37c1039be0ab>

[7] Jenkins

[https://www.jenkins.io/doc/book/installing/kubernetes /](https://www.jenkins.io/doc/book/installing/kubernetes%20/)

[8] ArgoCD

<https://argo-cd.readthedocs.io/en/stable/user-guide/helm/>