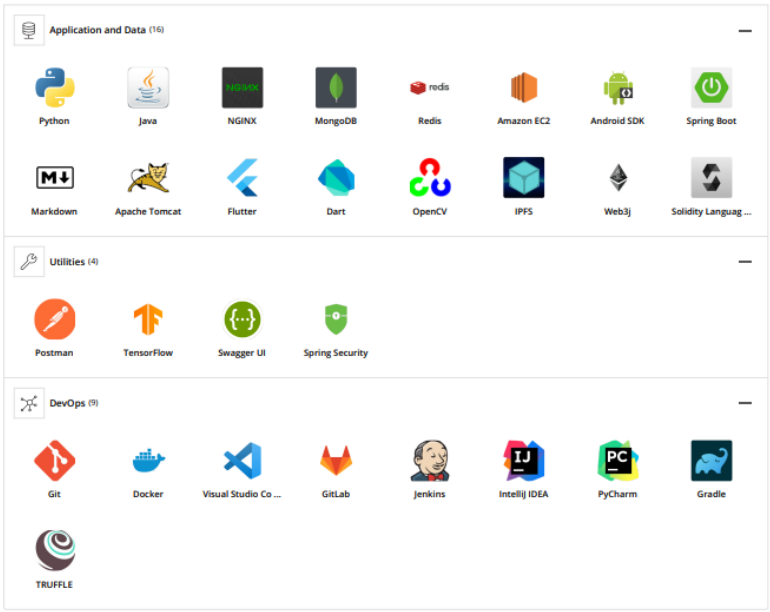
포팅메뉴얼

1. 프로젝트 구성도

기술 스택(stackshare)



시스템 아키텍쳐

2. Spring 서버 ec2 세팅

Server spec : t2.micro

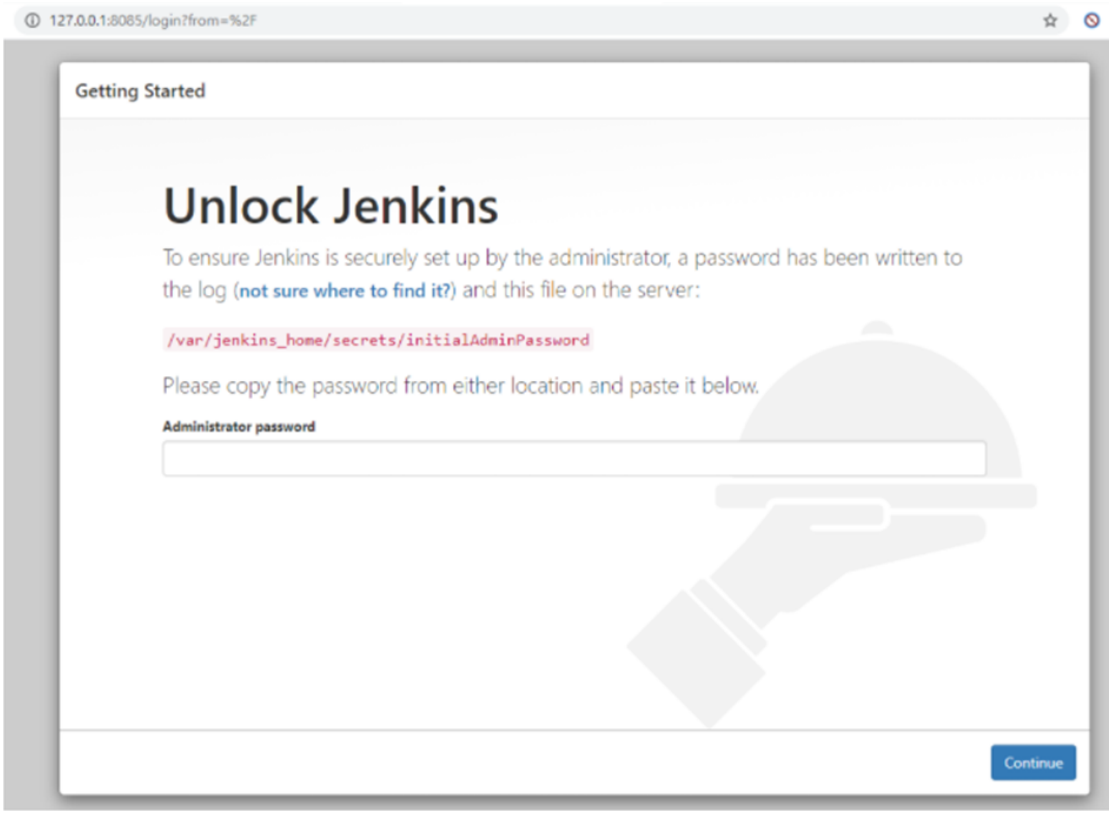
os : Amazon Linux 2

docker 설치

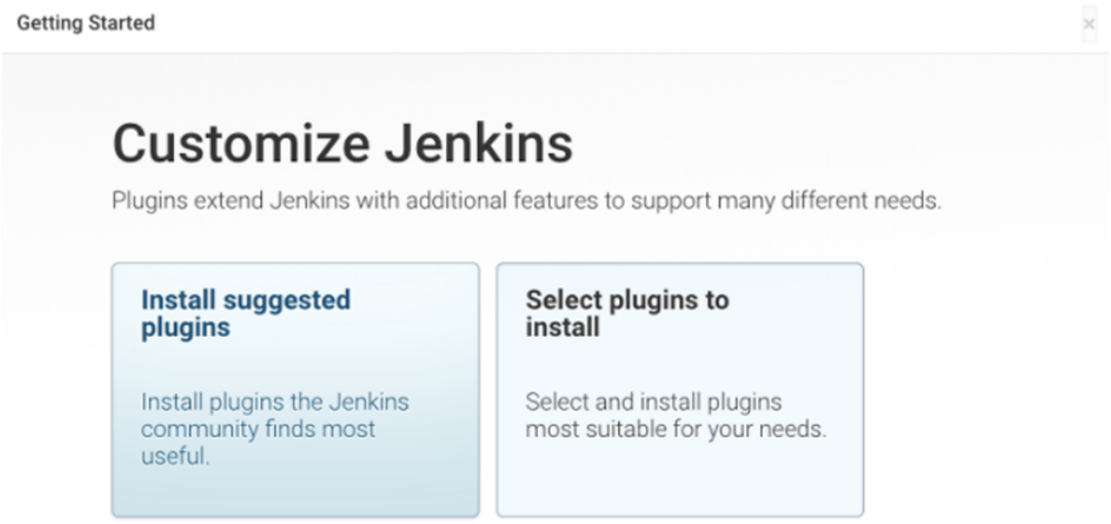
```
// docker 설치

// 패키지 업데이트
sudo apt update
// https관련 패키지 설치
sudo apt install apt-transport-https ca-certificates curl software-properties-common
// docker repository 접근을 위한 gpg 키 설정
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
// docker repository 등록
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
// 다시 업데이트
sudo apt update
// 도커 설치
sudo apt install docker-ce
// 설치 확인
docker --version
```

3. jenkins 서버 ec2 세팅

```
// Jenkins 설치


// 젠킨스 이미지 다운로드
docker pull jenkins/jenkins:lts
// 젠킨스 컨테이너 설치 및 실행
docker run -itd -p 8080:8080 -v /jenkins:/var/jenkins_home –name jenkins -u root jenkins/jenkins:lts
```

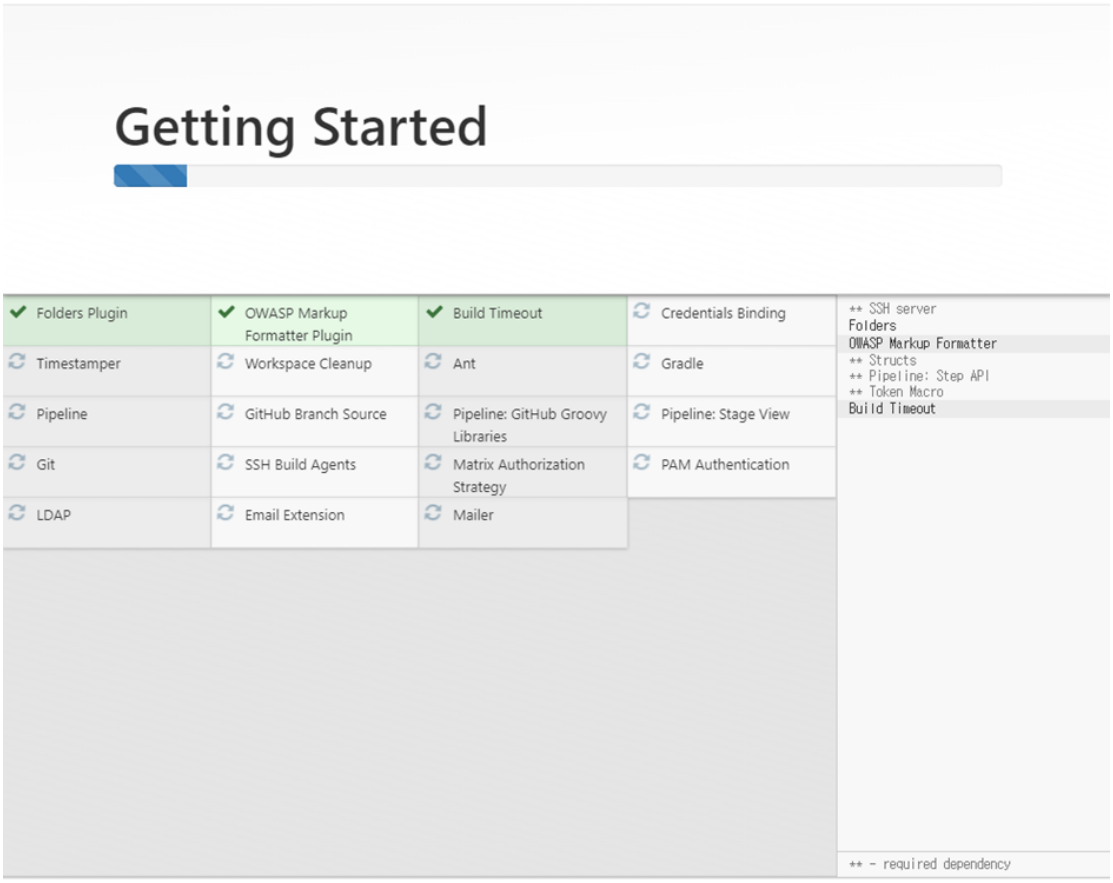


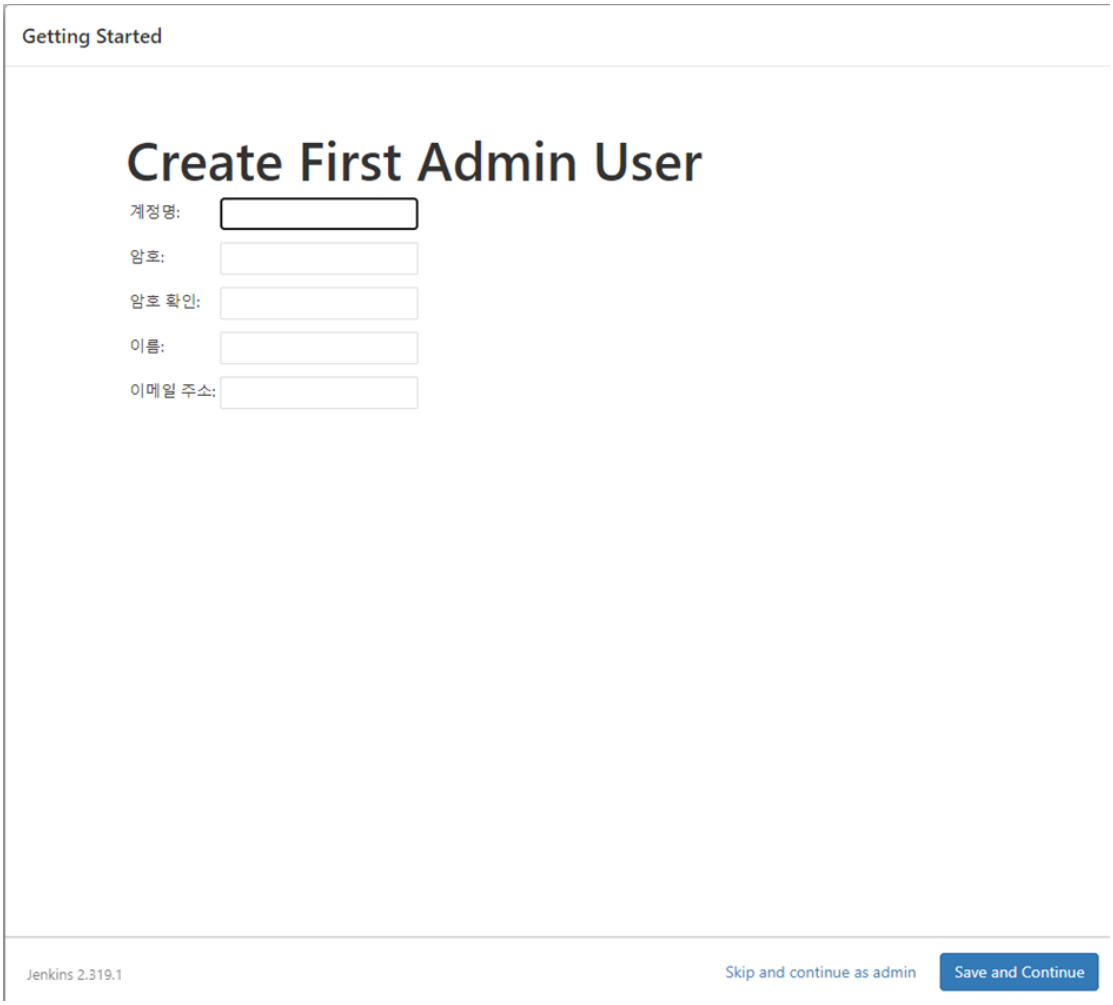

```
// 젠킨스 컨테이너에 접근해서 어드민 패스워드 찾기

docker exec -it jenkins /bin/bash

cat /var/jenkins_home/secrets/initialAdminPassword

// 혹은 jenkins 로그를 출력해서 초기 비번 확인
docker logs jenkins -f
```



Install suggested plugins 설치

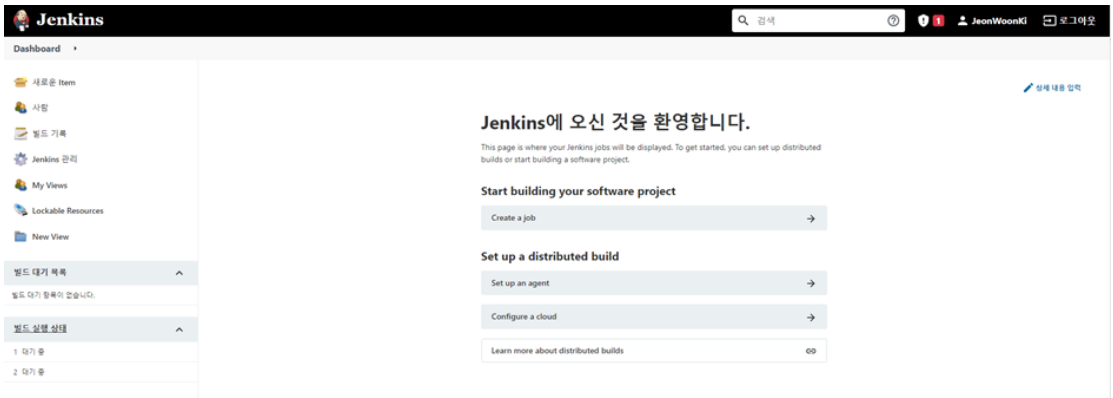
Getting Started





설치가 완료되면 계정의 정보 입력

4. Jenkins 파이프라인 생성



필요 플러그인 설치

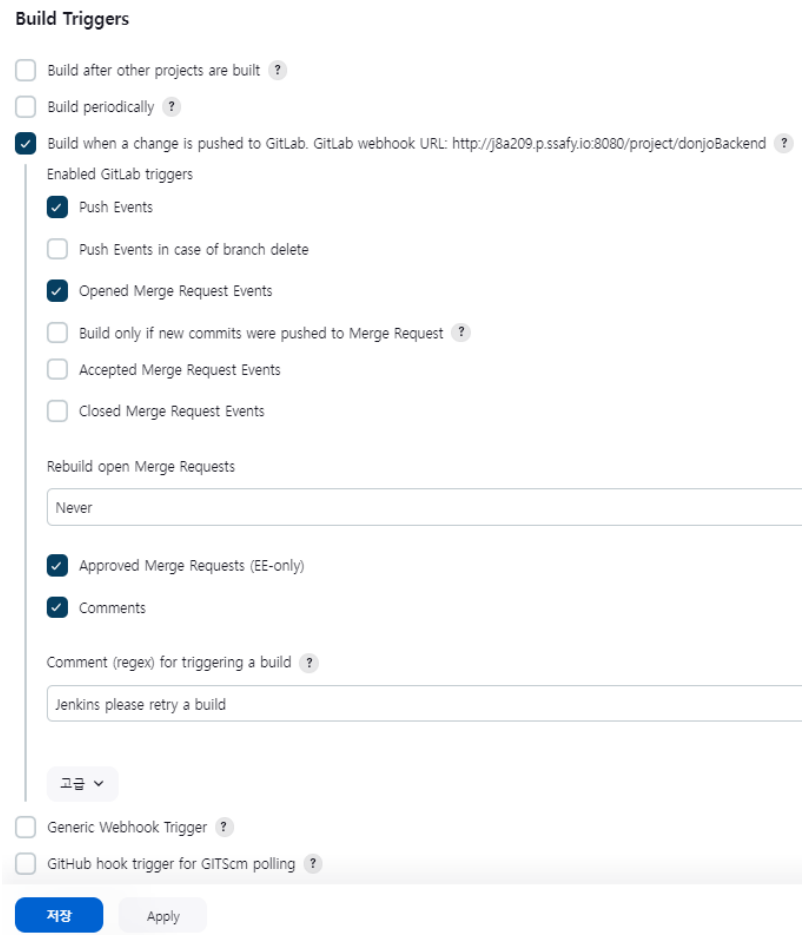
- Generic Webhook Trigger Plugin
- GitLab
- Gitlab API Plugin
- GitLab Authentication plugin
- Mattermost Notification Plugin
- Publish Over SSH

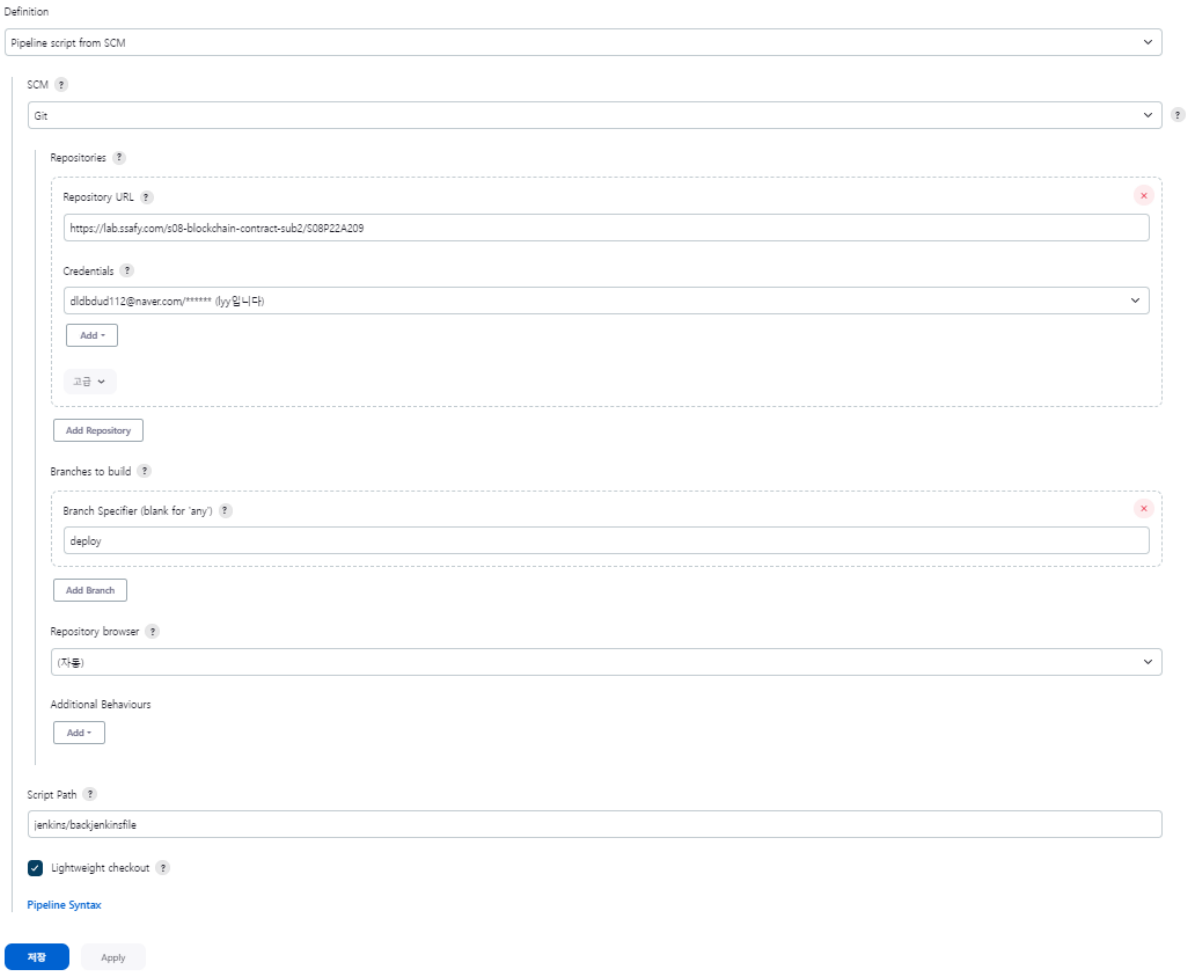
Credentials

T	P	Store ↓	Domain	ID	Name
		System	(global)	GitLabID	ts7681@naver.com/****** (깃랩 설정)
		System	(global)	Blockus	GitLab API token (Blockus pjt)
		System	(global)	lyy	dldbdud112@naver.com/****** (lyy입니다)

Jenkins 관리 → Manage Credentials → Credentials 등록

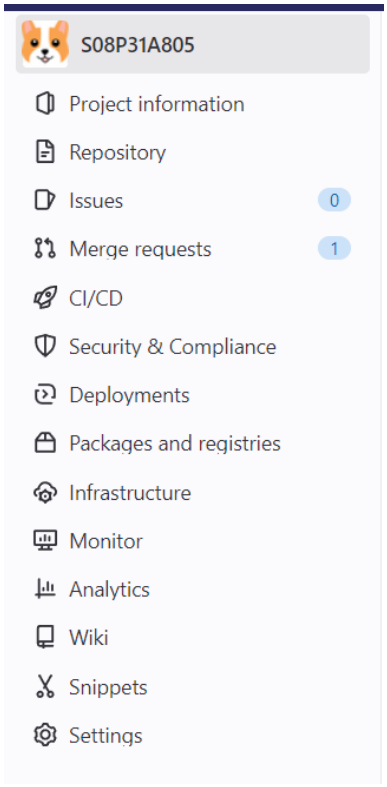
Item 생성(Pipeline)

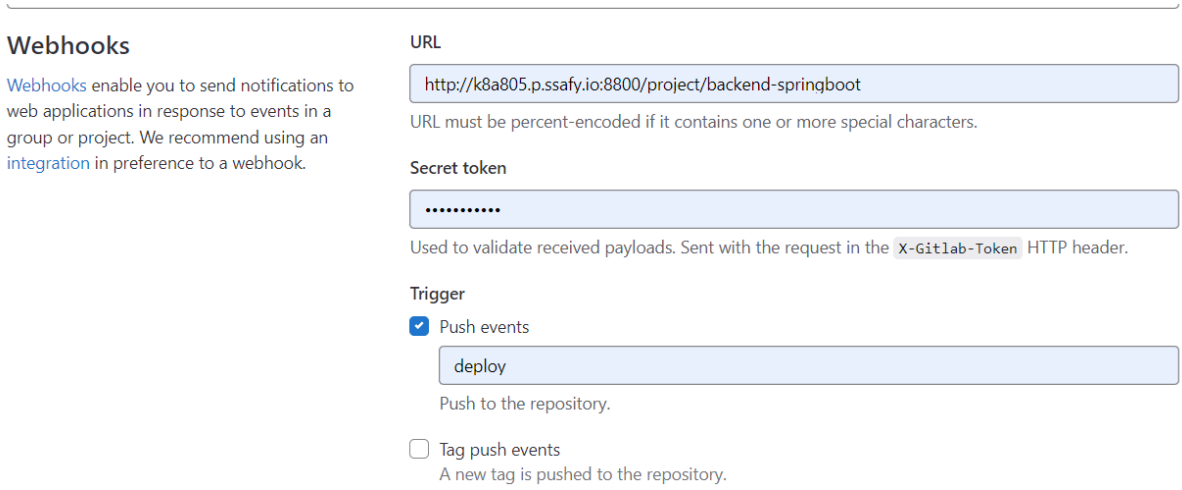




설정 셋팅하기

5. Webhook 연결(Build Triggers URL과 Secret token 복사)





GitLab → Settings → WebURL Secret token 복사 붙여넣기

6. 프로젝트 이미지 생성 후 컨테이너 띄우기

Backend Pipeline

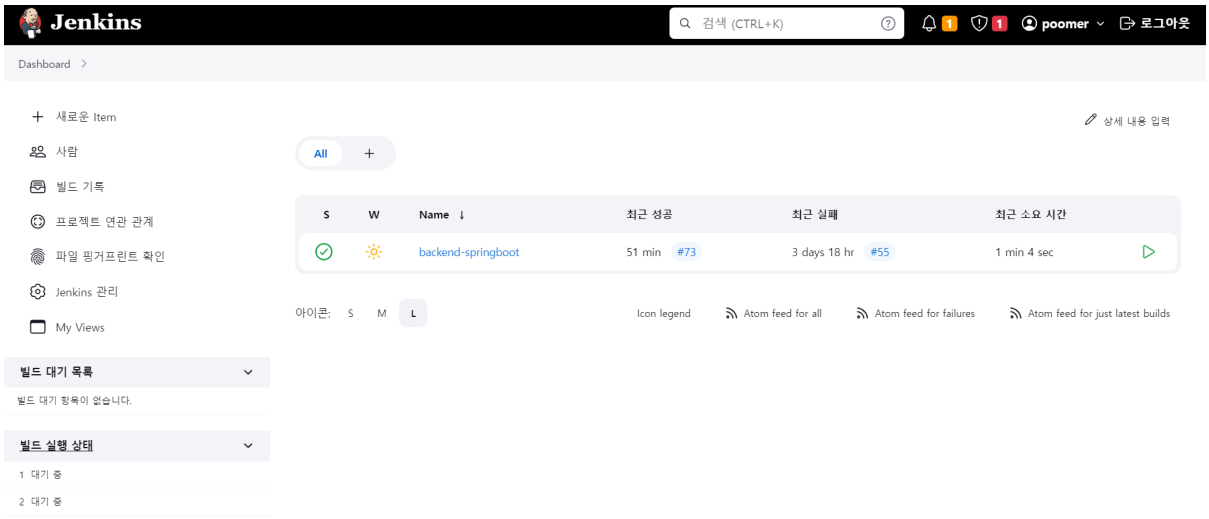
```
pipeline{
  agent any
  stages{
    stage('백엔드 자동 배포') {
      stages {
        stage('gradlew 권한'){
          steps{
            dir('backend'){
              sh "chmod +x gradlew"
            }
          }
        }
        stage('gradle 빌드'){
          steps{
            dir('backend'){
              sh './gradlew clean build --refresh-dependencies'
            }
          }
        }
```

```
        stage('백엔드 이미지 생성'){
          steps{
            dir('backend'){
              sh "./gradlew bootBuildImage"
            }
          }
        }
        stage('백엔드 컨테이너 삭제'){
          steps{
            catchError{
              sh "docker rm --force backend"
            }
          }
        }
        stage('백엔드 컨테이너 생성') {
          steps {
            sh "docker run -d -p 8080:8080 -v /etc/localtime:/etc/localtime:ro -e TZ=Asia/Seoul -e LC_ALL=C.UTF-8 --name backend backend:0.0.1-SNAPSHOT"
          }
        }
      }
    }
  }
}
```

Backend DockerFile

```
FROM openjdk:11-jdk
WORKDIR /app
COPY . .
RUN ls /app/src/main/resources/ # 이 줄 추가
ARG JAR_FILE=build/libs/backend-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} docker-springboot.jar
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app/docker-springboot.jar"]
```

Pipeline 생성 결과



7. Redis생성

```
// Redis Docker 이미지 다운로드
docker pull redis
// Redis Docker 컨테이너 생성 및 실행
docker run -d --name my-redis -e TZ=Asia/Seoul -p 6379:6379 redis redis-server --requirepass poompwd1234
// Docker 컨테이너 리스트 출력
docker ps -a
```

8. Nginx 설정

```
server {
    listen 80 ;
```

```
    listen [::]:80 ;
    server_name k8a805.p.ssafy.io;

    # 리다이렉트를 위한 설정
    return 301 https://$server_name$request_uri;
}

server {
    listen [::]:443 ssl;
    listen 443 ssl;
    server_name k8a805.p.ssafy.io;


    client_max_body_size 25M;

    # SSL 인증서와 관련된 설정
    ssl_certificate /etc/letsencrypt/live/k8a805.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/k8a805.p.ssafy.io/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    # 실제 서비스를 제공하는 설정
    root /home/ubuntu/landing;
    index index.html index.htm index.nginx-debian.html;
    location / {
        try_files $uri $uri/ =404;
    }

    location /webui {
        proxy_pass http://localhost:5001;
        proxy_set_header Host $host;
    }

    location ~ ^/api2 {
        proxy_pass http://localhost:8700;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~ ^/(api|swagger|webjars|configuration|swagger-resources|v2|v3|csrf) {
        proxy_pass http://localhost:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

9. Blockchain 엔드포인트 설정

Infura를 통해 Sepolia 체인의 엔드포인트를 발급 받습니다.