Plog 포팅매뉴얼

태그

1. 개발환경

1.1. Frontend

- Node JS 20.15.0 (LTS)
- React 18.3.1
- zustand 4.5.4
- Axios 1.7.2
- Tailwind CSS 3.4.7

1.2. Backend

- Java
    - Java OpenJDK 17
    - Spring Boot 3.3.2
        - Spring Data JPA 3.3.2
        - Spring Security 3.3.2
        - JUnit 5.8.2
        - Lombok 1.18.26
    - Gradle 7.6
- Jakarta API:
    - Jakarta Persistence API: 3.1.0
    - Jakarta Servlet API: 6.0.0
- Swagger: Springdoc OpenAPI 2.0.4
- MySQL Driver: 8.0.33
- QueryDSL: 5.0.0
- Redis: Spring Boot Starter Data Redis

- AWS S3: AWS Java SDK S3 1.11.1000
- Scheduler: Spring Retry & Spring Aspects
- JSON: org.json 20210307
- OAuth 2.0: Spring Boot OAuth2 Client & Resource Server
- Mail: Spring Boot Starter Mail
- Guava: 29.0-jre
- Socket IO: 4.6.1
- WebSocket: Spring Boot Starter WebSocket

1.3. Server

- Ubuntu 20.04 LTS
- Docker-compose 2.6.1
- Nginx 1.27.0
- Docker 27.1.1
- Jenkins 2.452.3

1.4. Database

- MySQL 9.0.1
- Redis 7.4.0

1.5. UI/UX

- Figma

1.6. IDE

- Visual Studio Code 1.91.1
- IntelliJ IDEA 2024.01

1.7. 형상/이슈관리

- GitLab
- Jira

1.8. 기타 툴

- Postman 11.6.2
- Termius 9.2.0

2. 환경변수

2.1. Frontend

> REACT_APP_FIREBASE_API_KEY
> REACT_APP_FIREBASE_AUTH_DOMAIN
> REACT_APP_FIREBASE_PROJECT_ID
> REACT_APP_FIREBASE_STORAGE_BUCKET
> REACT_APP_FIREBASE_MESSAGING_SENDER_ID
> REACT_APP_FIREBASE_APP_ID
> REACT_APP_FIREBASE_MEASUREMENT_ID
> REACT_APP_WEB_PUSH_CERTIFICATE_KEY
> REACT_APP_API_BASE_URL

2.2. Backend

> build.date
> server.port
> server.address
> server.servlet.context-path
> server.servlet.encoding.charset
> server.servlet.encoding.enabled
> server.servlet.encoding.force

> spring.jpa.hibernate.naming.implicit-strategy
> spring.jpa.hibernate.naming.physical-strategy
> spring.jpa.hibernate.ddl-auto
> spring.jpa.properties.hibernate.dialect
> spring.data.web.pageable.one-indexed-parameters
> spring.datasource.url
> spring.datasource.driver-class-name

> spring.datasource.hikari.username
> spring.datasource.hikari.password
> jwt.secret
> jwt.expiration

> spring.mail.host
> spring.mail.port
> spring.mail.username
> spring.mail.password
> spring.mail.properties.mail.smtp.auth
> spring.mail.properties.mail.smtp.starttls.enable
> spring.mail.properties.mail.smtp.starttls.required
> spring.mail.properties.mail.smtp.connectiontimeout
> spring.mail.properties.mail.smtp.timeout
> spring.mail.properties.mail.smtp.writetimeout

> spring.data.redis.host
> spring.data.redis.port
> spring.data.redis.password

> spring.redis.realtime.host
> spring.redis.realtime.port
> spring.redis.realtime.password

> cloud.aws.credentials.accessKey
> cloud.aws.credentials.secretKey
> cloud.aws.s3.bucketName
> cloud.aws.region.static
> cloud.aws.stack.auto-

> weather.api.key
> weather.api.url

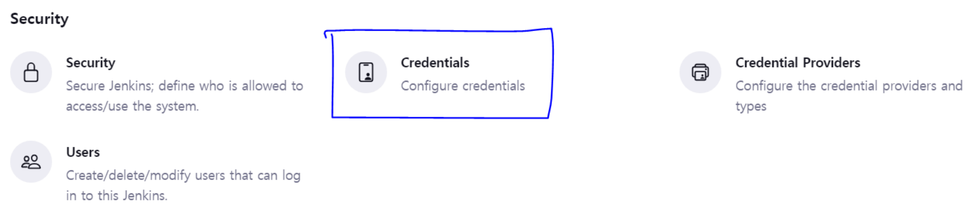
> spring.servlet.multipart.enabled

> server.url

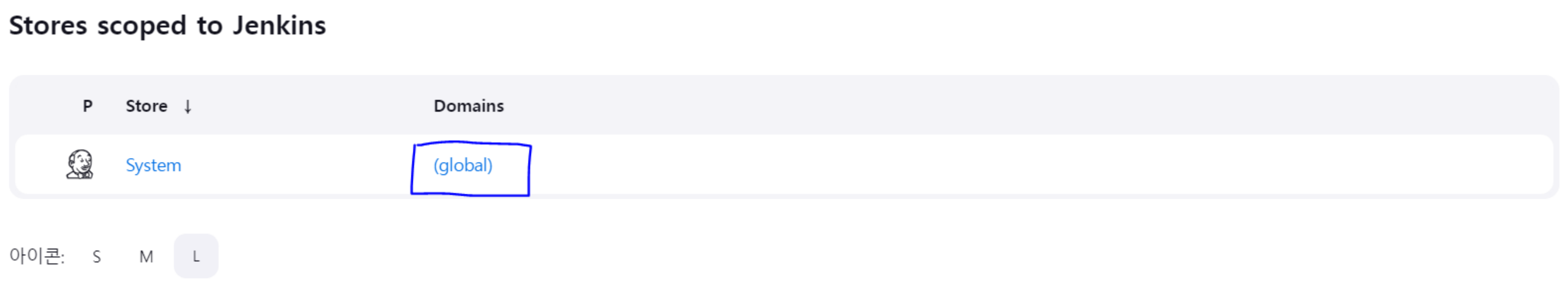
2.3. 민감 환경변수 관리

2.3.1. Frontend

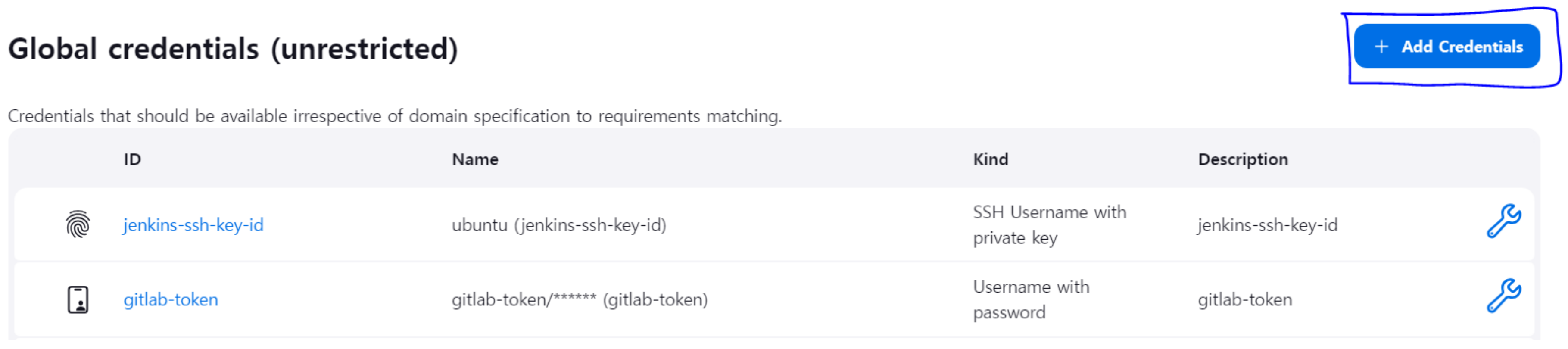
Jenkins credentials로 .env 수동 저장 및 관리(.gitignore에 추가하여 GitLab에 푸시되는 일이 없도록 함)


Security
Security
Secure Jenkins; define who is allowed to access/use the system.
Credentials
Configure credentials
Credential Providers
Configure the credential providers and types
Users
Create/delete/modify users that can log in to this Jenkins.

Jenins 설정의 Credentials로 간다.



(global) 도메인으로 만든다.

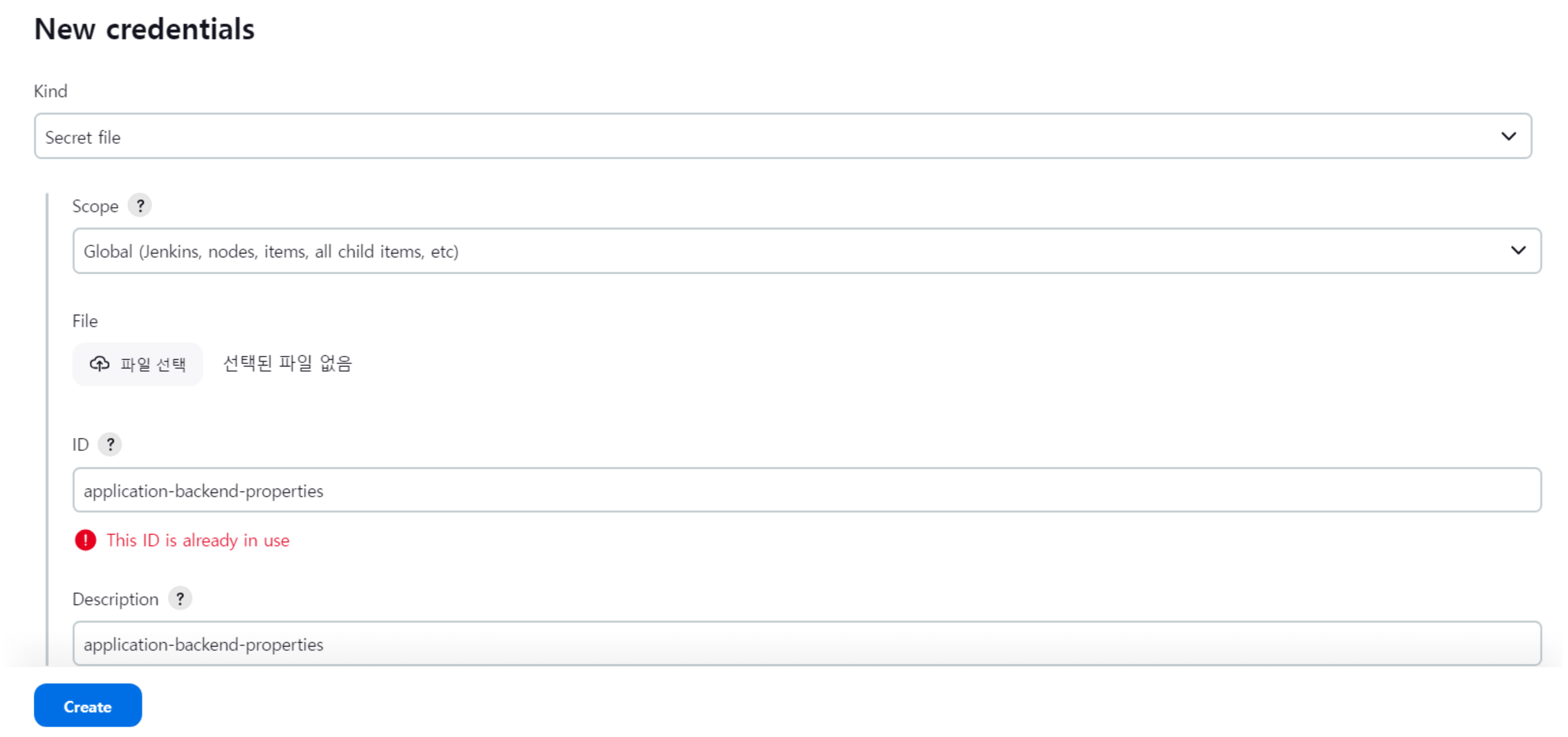


Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

	ID	Name	Kind	Description
	jenkins-ssh-key-id	ubuntu (jenkins-ssh-key-id)	SSH Username with private key	jenkins-ssh-key-id
	gitlab-token	gitlab-token/****** (gitlab-token)	Username with password	gitlab-token

Add Credentials로 새로운 Crdential을 만든다.



New credentials

Kind: Secret file

Scope: Global (Jenkins, nodes, items, all child items, etc)

File: 파일 선택 선택된 파일 없음

ID: application-backend-properties

This ID is already in use

Description: application-backend-properties

Secret 파일에 민감한 환경변수 파일을 넣고, 파이프라인 구성시 필요한 환경변수 파일을 복사해서 이미지를 빌드하는 형식으로 진행했다.

2.3.2. Backend

Jenkins credentials로 application.properties & certification.json 수동 저장 및 관리(.gitignore에 추가하여 GitLab에 푸시되는 일이 없도록 함)

Frontend와 동일하게 민감한 환경변수를 관리하고 진행했다.

3. EC2 세팅

3.1. Docker 설치

```
# 1. 리눅스 업데이트
sudo apt update -y && sudo apt upgrade -y

# 1.1 필수 패키지 설치
sudo apt-get install -y ca-certificates curl gnupg lsb-release

# 2. Docker의 공식 GPG 키를 추가할 디렉토리 생성
sudo mkdir -p /etc/apt/keyrings

# 3. Docker의 GPG 키 다운로드 및 바이너리 형식으로 변환하여 저장
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

# 4. Docker 저장소를 추가
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 5. 패키지 목록 업데이트
sudo apt-get update

# 6. Docker 패키지 설치
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 7. Docker 데몬을 시작하고 부팅 시 자동으로 시작하도록 설정
sudo systemctl start docker
sudo systemctl enable docker
```

3.2. Docker-compose 설정

```yaml
version: '3.8'

services:
	nginx:
		image: nginx:latest
		restart: unless-stopped
		volumes:
			- ./data/nginx/nginx.conf:/etc/nginx/nginx.conf
			- ./data/certbot/conf:/etc/letsencrypt
			- ./data/certbot/www:/var/www/certbot
		ports:
			- "80:80"
			- "443:443"
		command: "/bin/sh -c 'while :; do sleep 360h & wait $${!}; nginx -s reload; done & nginx -g \"daemon off;\"'"
		logging:
			driver: "json-file"
			options:
				max-size: "10m"
				max-file: "1"

	certbot:
		image: certbot/certbot
		restart: unless-stopped
		volumes:
			- ./data/certbot/conf:/etc/letsencrypt
			- ./data/certbot/www:/var/www/certbot
		entrypoint: "/bin/sh -c 'trap exit TERM; while :; do certbot renew; sleep
720h & wait $${!}; done;'"
		logging:
			driver: "json-file"
			options:
				max-size: "10m"
				max-file: "1"

	jenkins:
		image: jenkins/jenkins:lts
		restart: unless-stopped
		user: root # 필요한 경우 root로 진행
		volumes:
			- jenkins_home:/var/jenkins_home
			- ./data/jenkins:/var/jenkins_shared
			- /var/run/docker.sock:/var/run/docker.sock  # Docker 소켓 마운트
			- /usr/local/bin/docker-compose:/usr/local/bin/docker-compose  # Docker Compose 바이너리 공유
			- /usr/bin/docker:/usr/bin/docker  # Docker CLI 바이너리 공유
			- /home/ubuntu/plog:/home/ubuntu/plog  # 호스트의 프로젝트 디렉토리 공유
		environment:
			JENKINS_OPTS: --prefix=/jenkins
		logging:
			driver: "json-file"
			options:
				max-size: "10m"
				max-file: "1"

	backend-realtime:
		image: backend-realtime:latest
		restart: unless-stopped
		environment:
			- SPRING_PROFILES_ACTIVE=prod
		logging:
			driver: "json-file"
			options:
				max-size: "10m"
				max-file: "1"

	backend:
		image: backend:latest
		restart: unless-stopped
		environment:
			- SPRING_PROFILES_ACTIVE=prod
		logging:
			driver: "json-file"
			options:
				max-size: "10m"
				max-file: "1"

	frontend:
		image: frontend:latest
		restart: unless-stopped
		env_file:
			- .env
		environment:
			- NODE_ENV=production
```

```
                logging:
                        driver: "json-file"
                        options:
                                max-size: "10m"
                                max-file: "1"
                expose:
                        - "3000"

        mysql:
                image: mysql:latest
                container_name: mysql
                restart: unless-stopped
                environment:
                        MYSQL_ROOT_PASSWORD:
                        MYSQL_DATABASE:
                        MYSQL_USER:
                        MYSQL_PASSWORD:
                volumes:
                        - ./data/mysql_data:/var/lib/mysql
                expose:
                        - "3306"

        redis:
                image: redis:latest
                restart: unless-stopped
                command: redis-server --requirepass plog
                expose:
                        - "6379"

        redis-realtime:
                image: redis:latest
                restart: unless-stopped
                command: redis-server --port 6380 --requirepass plog
                expose:
                        - "6380"

volumes:
        jenkins_home:
        mysql_data:
```

3.3. Nginx 설정

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;

events {
    worker_connections  1024;
}

http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"'
                      '"$host" "$server_name" "$request_uri" "$uri" '
                      '"$request_body" "$args" "$upstream_addr" "$upstream_status"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    keepalive_timeout  65;

    # 요청 제한 설정
    limit_req_zone $binary_remote_addr zone=mylimit:1m rate=100r/s;

    server {
        listen 80;
        server_name i11b308.p.ssafy.io;
        server_tokens off;

        location /.well-known/acme-challenge/ {
            root /var/www/certbot;
        }
        location / {
            return 301 https://$host$request_uri;
        }
    }

    server {
        listen 443 ssl;
        server_name i11b308.p.ssafy.io;
        server_tokens off;

        ssl_certificate /etc/letsencrypt/live/i11b308.p.ssafy.io/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/i11b308.p.ssafy.io/privkey.pem;
        include /etc/letsencrypt/options-ssl-nginx.conf;
        ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

        # client_max_body_size 설정 추가
        client_max_body_size 50M;

        # Jenkins 리버스 프록시 설정 (8080 포트)
        location /jenkins {
            proxy_pass http://jenkins:8080/jenkins;
            proxy_set_header Host $host:443;
            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;
            proxy_set_header X-Forwarded-Port 443;

            proxy_http_version 1.1;

                        # 캐시 처리
            add_header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0";
            add_header Pragma "no-cache";
            add_header Expires "0";

            # CORS 헤더 추가
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept, Authorization';
            add_header 'Access-Control-Allow-Credentials' 'true';


        }

        # Jenkins의 50000 포트에 대한 리버스 프록시 설정
        location /jenkins-jnlp {
            proxy_pass http://jenkins:50000;
            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;

            add_header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0";
            add_header Pragma "no-cache";
            add_header Expires "0";
        }

        # backend 리버스 프록시 설정
        location /api/ {
            proxy_pass http://backend:8081/api/;
            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;

            add_header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0";
            add_header Pragma "no-cache";
            add_header Expires "0";

        }

        # backend-realtime 리버스 프록시 설정
        location /realtime/ {
            proxy_pass http://backend-realtime:8082/realtime/;

            # WebSocket 지원을 위한 헤더 추가
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "Upgrade";

            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;

            add_header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0";
            add_header Pragma "no-cache";
            add_header Expires "0";
        }

        # 요청 제한 설정 적용
        location / {
            limit_req zone=mylimit burst=1000 nodelay;

            # 요청이 너무 많을 경우 429 에러 반환
            limit_req_status 429;

            proxy_pass http://frontend:3000;
            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;

            # 캐시 방지 헤더 설정
            add_header Cache-Control "no-store, no-cache, must-revalidate, proxy-revalidate, max-age=0";
            add_header Pragma "no-cache";
            add_header Expires "0";

            # CORS 헤더 추가
            add_header 'Access-Control-Allow-Origin' '*';
            add_header 'Access-Control-Allow-Methods' 'GET, POST, PUT, DELETE, PATCH, OPTIONS';
            add_header 'Access-Control-Allow-Headers' 'Origin, Content-Type, Accept, Authorization';
            add_header 'Access-Control-Allow-Credentials' 'true';

        }

    }

}
```

3.4. EC2 Port

Port 번호	내용
22	SSH
80	HTTP (HTTPS로 redirect)
443	HTTPS
3000	Nginx, React(Docker)
3306	MySQL
6379	Redis (Cache)
6380	Redis (Realtime)
8080	Jenkins
8081	Spring Boot (api) (Docker)
8082	Spring Boot (realtime) (Docker)

3.5. 방화벽(UFW) 설정

```
ufw 상태 확인
sudo ufw status
Status: active

To                         Action      From
--                         ------      ----
22                         ALLOW       Anywhere
8989                       ALLOW       Anywhere
443                        ALLOW       Anywhere
80                         ALLOW       Anywhere
587                        ALLOW       Anywhere
22 (v6)                    ALLOW       Anywhere (v6)
8989 (v6)                  ALLOW       Anywhere (v6)
443 (v6)                   ALLOW       Anywhere (v6)
80 (v6)                    ALLOW       Anywhere (v6)
587 (v6)                   ALLOW       Anywhere (v6)

사용할 포트 허용하기
sudo ufw allow 포트번호

ufw 활성화하기
sudo ufw enable

등록한 포트 삭제하기
sudo ufw status numbered
sudo ufw delete 4
```

4. CI/CD 구축

4.1. Jenkins 도커 이미지 + 컨테이너

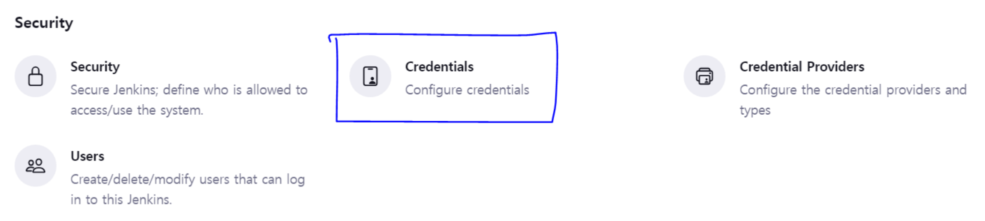
docker-compose.yml 내부

```
        jenkins:
                image: jenkins/jenkins:lts
                restart: unless-stopped
                user: root # 필요한 경우 root로 진행
                volumes:
                        - jenkins_home:/var/jenkins_home
                        - ./data/jenkins:/var/jenkins_shared
                        - /var/run/docker.sock:/var/run/docker.sock  # Docker 소켓 마운트
                        - /usr/local/bin/docker-compose:/usr/local/bin/docker-compose  # Docker Compose 바이너리 공유
                        - /usr/bin/docker:/usr/bin/docker  # Docker CLI 바이너리 공유
                        - /home/ubuntu/plog:/home/ubuntu/plog  # 호스트의 프로젝트 디렉토>리 공유
                environment:
                        JENKINS_OPTS: --prefix=/jenkins
                logging:
                        driver: "json-file"
                        options:
                                max-size: "10m"
                                max-file: "1"
```

4.2. Jenkins 설정

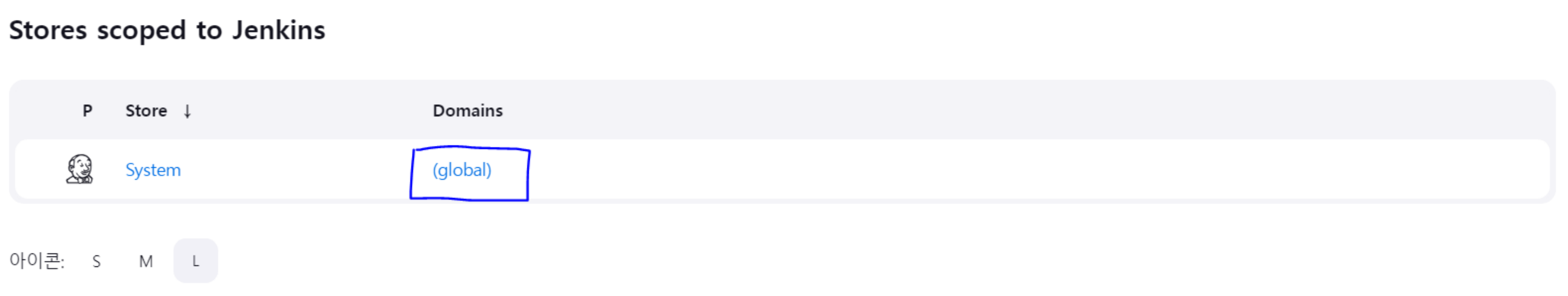
4.2.1 GitLab Credentials 설정

1. 아이디 → "Credentials" 클릭
2. "Store : System" → "(global)" → "+ Add Credentials" 클릭



Security

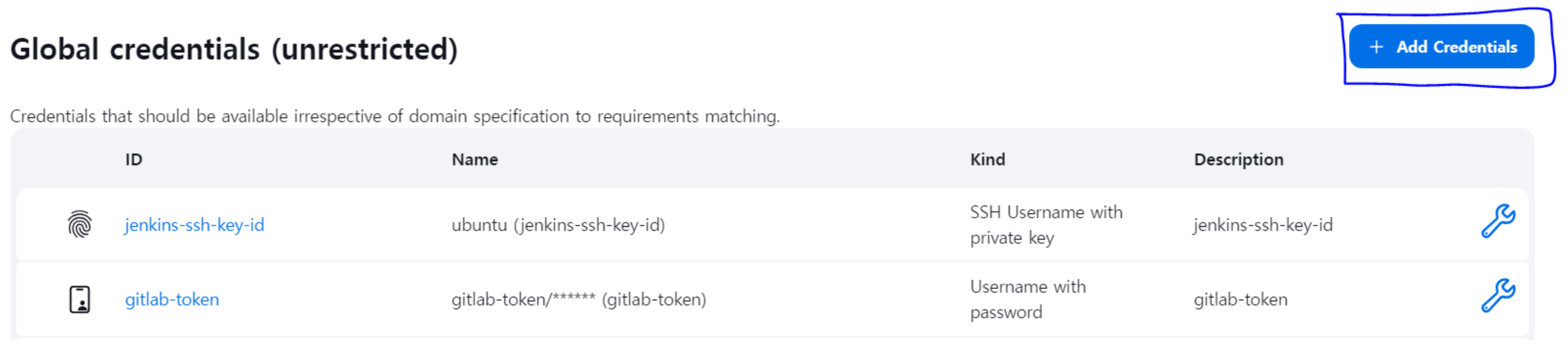
Security Secure Jenkins; define who is allowed to access/use the system.	Credentials Configure credentials
Credential Providers Configure the credential providers and types	
Users Create/delete/modify users that can log in to this Jenkins.	



Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

아이콘: S M L



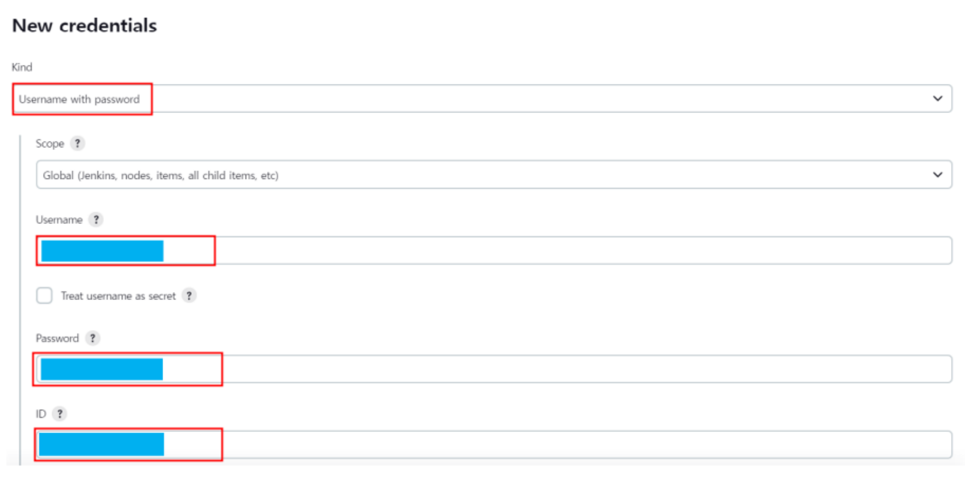
Global credentials (unrestricted)

+ Add Credentials

Credentials that should be available irrespective of domain specification to requirements matching.

	ID	Name	Kind	Description
	jenkins-ssh-key-id	ubuntu (jenkins-ssh-key-id)	SSH Username with private key	jenkins-ssh-key-id
	gitlab-token	gitlab-token/****** (gitlab-token)	Username with password	gitlab-token

3. "Kind"에 "Username with password" 입력 → "Username"에 GitLab ID 혹은 원하는 ID 입력(gitlab-token) → "Password"에 Gitlab Personal Access Tokens 입력 → "ID"에 임의 아이디 입력(gitlab-token) → 생성
*** Personal Access Token은 Gitlab > User Settings > Access Tokens에서 생성



New credentials

Kind
Username with password

Scope ?
Global (Jenkins, nodes, items, all child items, etc)

Username ?
[redacted]

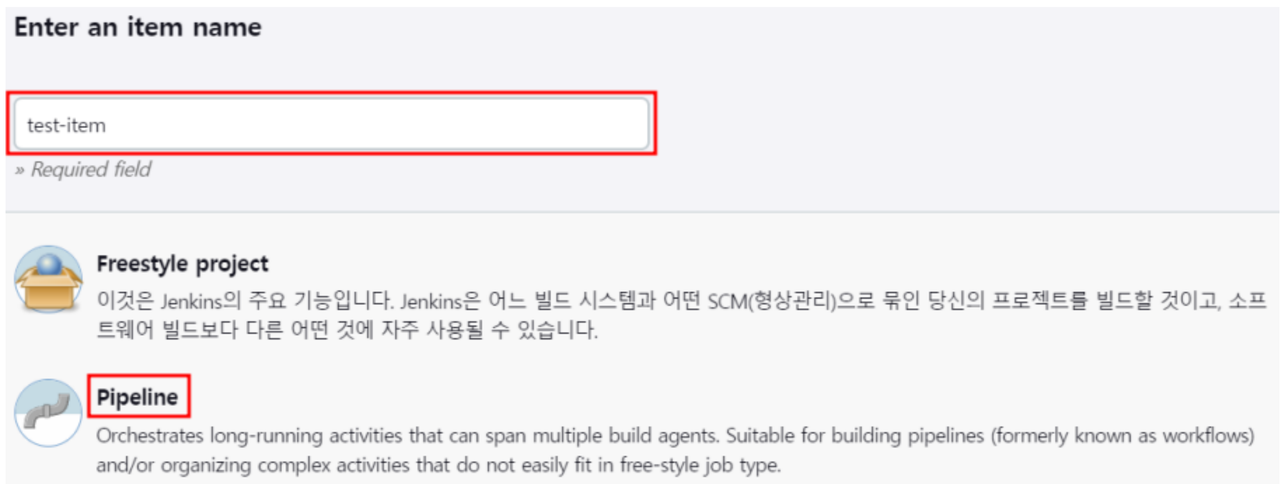
Treat username as secret ?

Password ?
[redacted]

ID ?
[redacted]

4.2.2 Jenkins Item 생성

1. “새로운 Item” 클릭
2. “Enter an item name”에 임의 Item 이름 입력 → “Pipeline” 클릭



Enter an item name

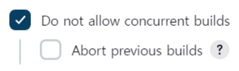
test-item

» Required field

Freestyle project
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.

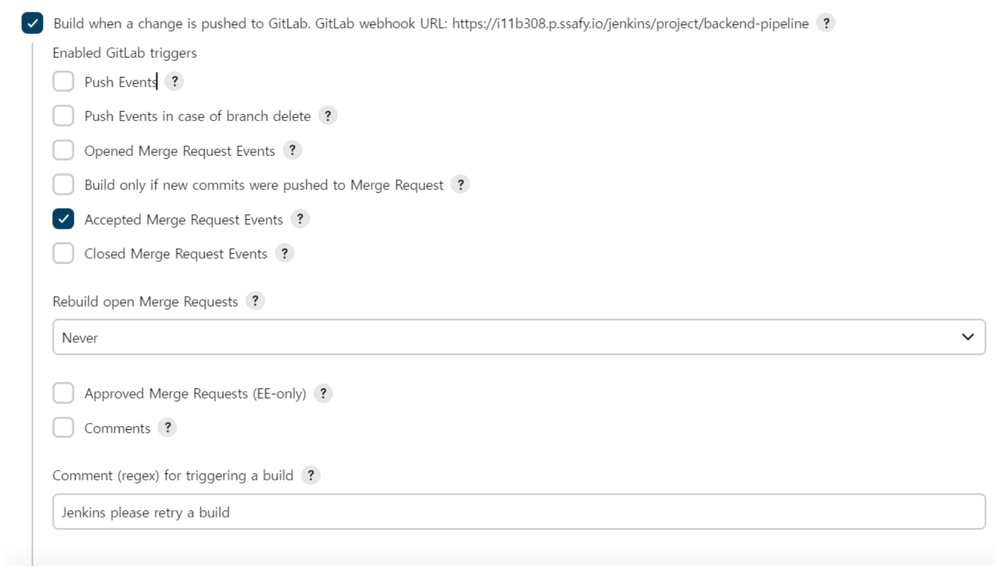
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

3. “General” → “Do not allow concurrent builds” 클릭
   (한 빌드를 진행중이면 동시에 빌드를 진행하지 않게 한다)



- [x] Do not allow concurrent builds
  - [ ] Abort previous builds

4. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭
   (WebHook 설정 : GitLab 특정 브랜치 merge 시 자동 빌드 + 배포 설정)
   (해당 URL 복사 → WebHook 설정 시 사용 예정)



- [x] Build when a change is pushed to GitLab. GitLab webhook URL: https://i11b308.p.ssafy.io/jenkins/project/backend-pipeline

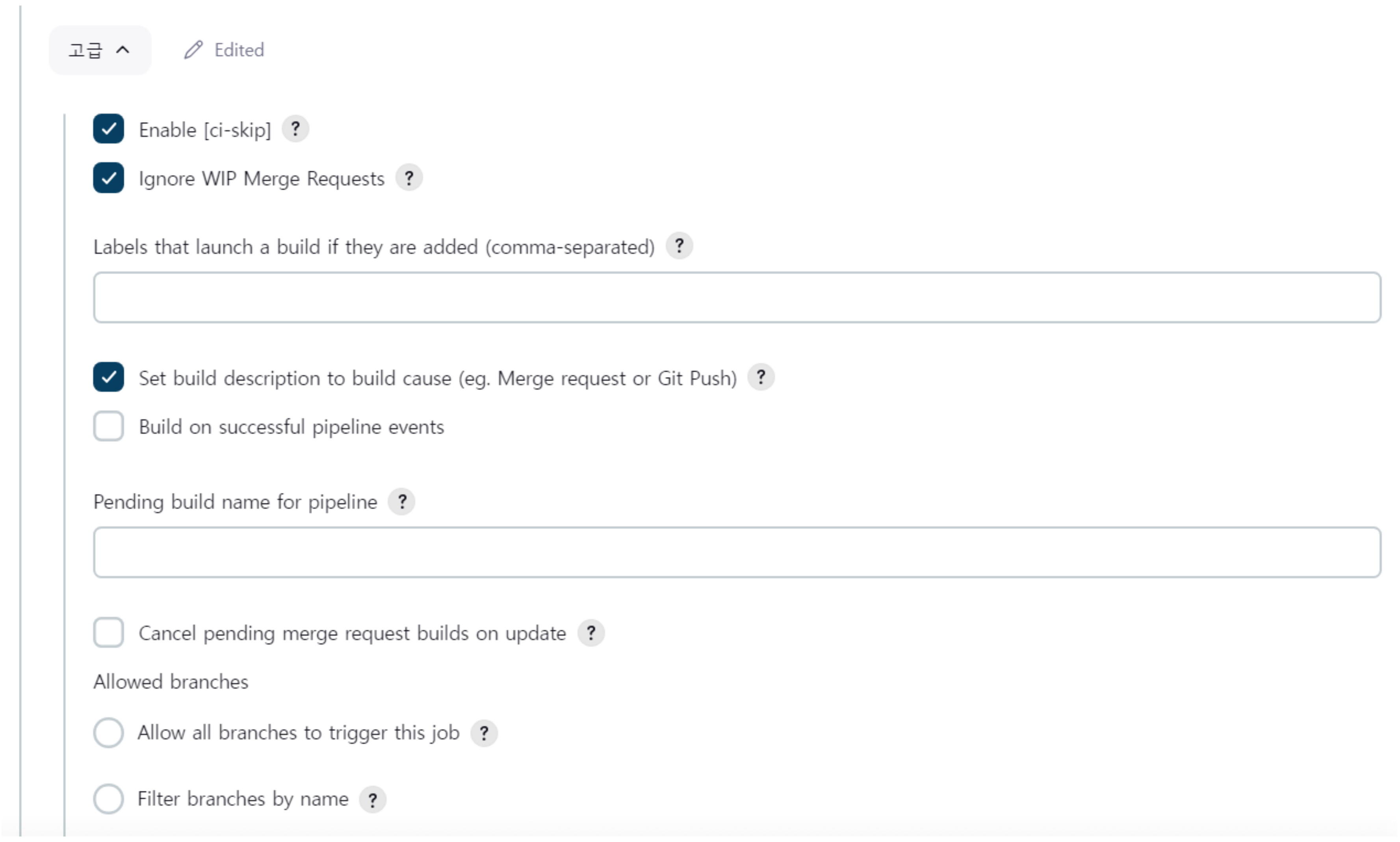
Enabled GitLab triggers
- [ ] Push Events
- [ ] Push Events in case of branch delete
- [ ] Opened Merge Request Events
- [ ] Build only if new commits were pushed to Merge Request
- [x] Accepted Merge Request Events
- [ ] Closed Merge Request Events

Rebuild open Merge Requests: Never

- [ ] Approved Merge Requests (EE-only)
- [ ] Comments

Comment (regex) for triggering a build: Jenkins please retry a build

5. “Build when a change is pushed to GitLab” 하위의 “고급...” 클릭



- [x] Enable [ci-skip]
- [x] Ignore WIP Merge Requests

Labels that launch a build if they are added (comma-separated)

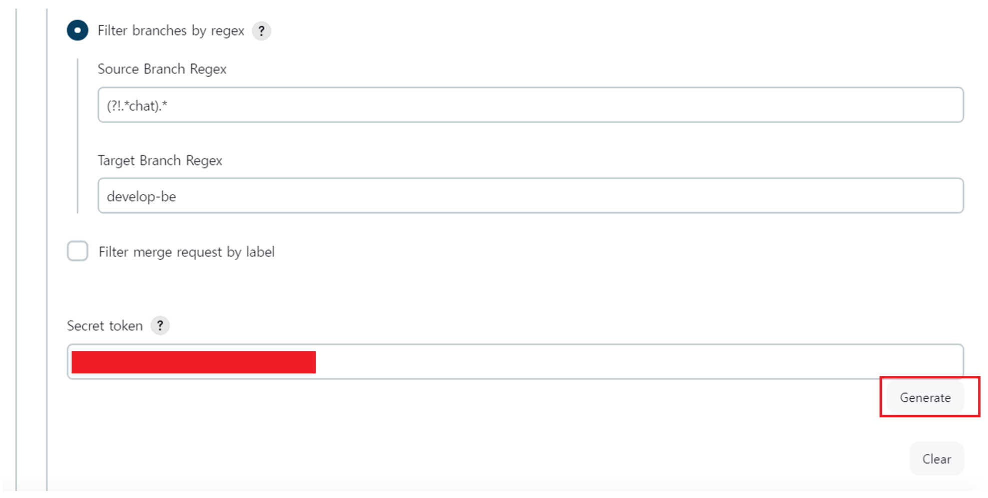
- [x] Set build description to build cause (eg. Merge request or Git Push)
- [ ] Build on successful pipeline events

Pending build name for pipeline

- [ ] Cancel pending merge request builds on update

Allowed branches
- ○ Allow all branches to trigger this job
- ○ Filter branches by name

6. 특정 브랜치에서 타겟 브랜치로 머지를 할 경우 빌드 + 배포가 진행되도록 설정
   Secret token의 “Generate” 클릭 후 생성된 토큰값 복사



● Filter branches by regex ?

Source Branch Regex

(?!.*chat).*

Target Branch Regex

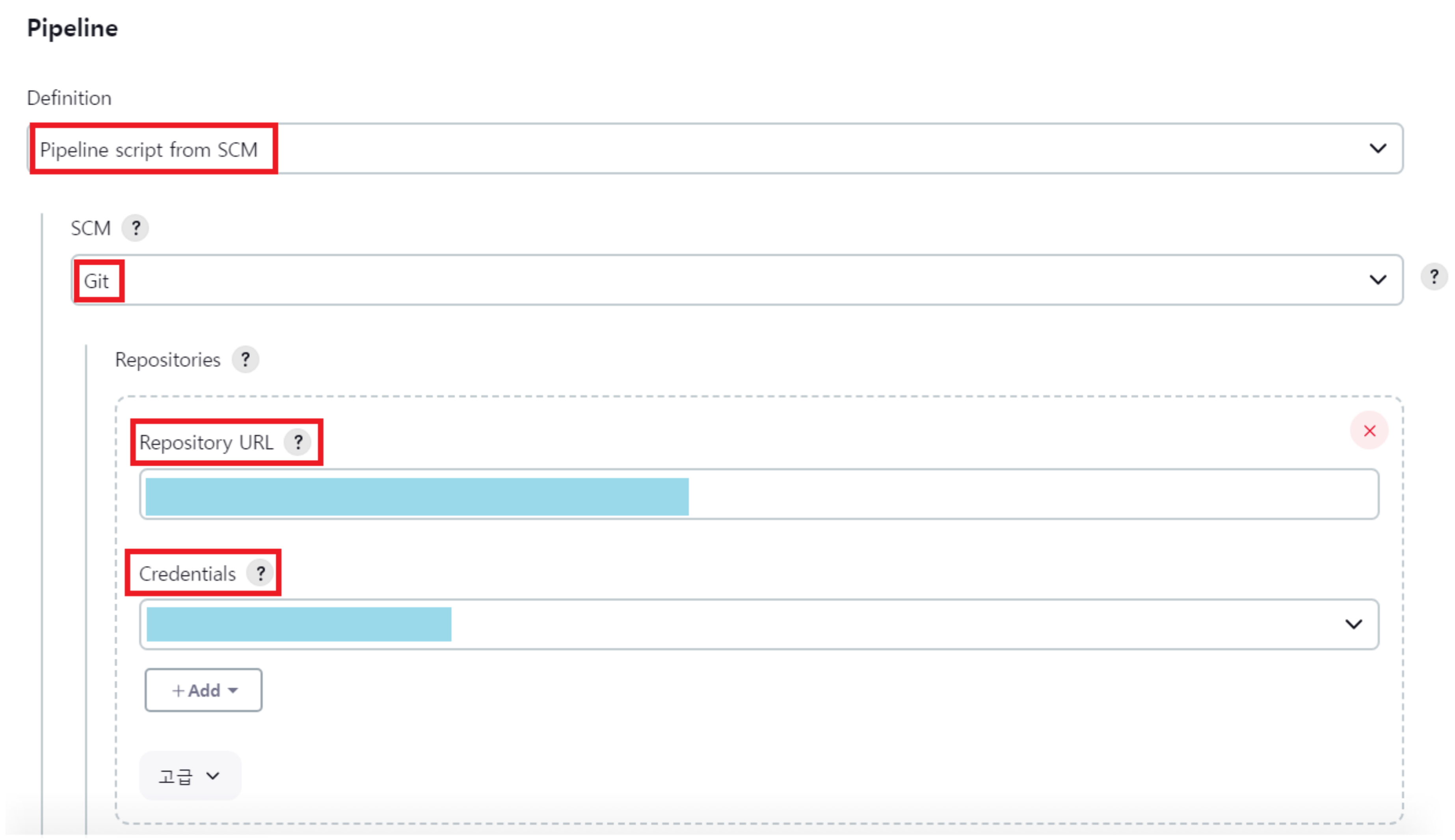
develop-be

☐ Filter merge request by label

Secret token ?

[redacted]

7. “Pipeline” → “Definition”에 Pipeline script from SCM 설정 → “SCM”에 “Git” 설정 → “Repository URL”에 프로젝트 GitLab URL 입력 → “Credentials”에 사전에 추가한 Credentials 입력



Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

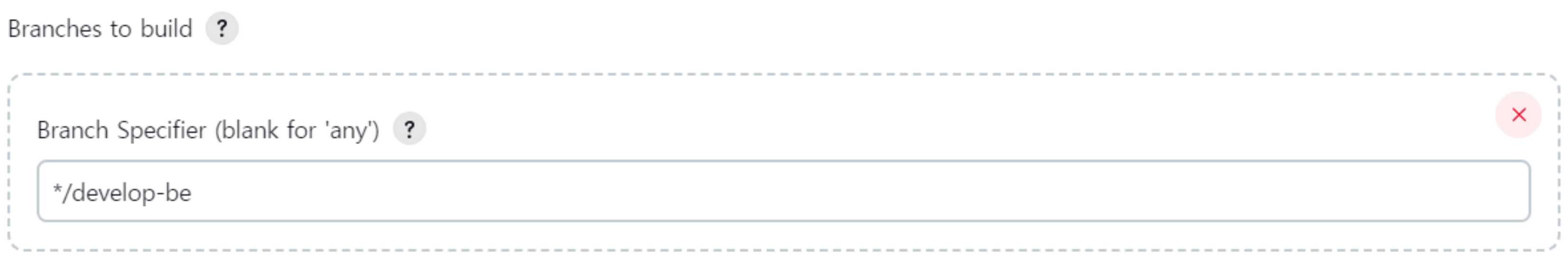
Repository URL ?

[redacted]

Credentials ?

[redacted]

8. “Branch Specifier”에 빌드 할 브랜치명 입력 (master일 시 “*/master)

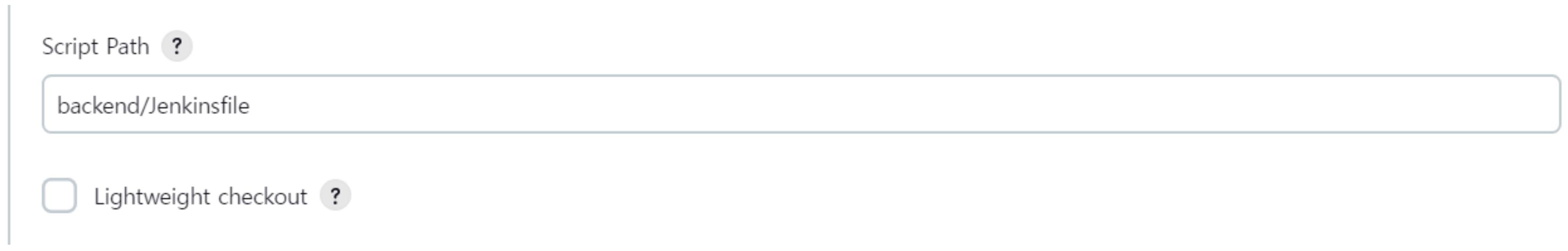


Branches to build ?

Branch Specifier (blank for 'any') ?

*/develop-be

9. “Script Path”에 Jenkinsfile 경로 입력



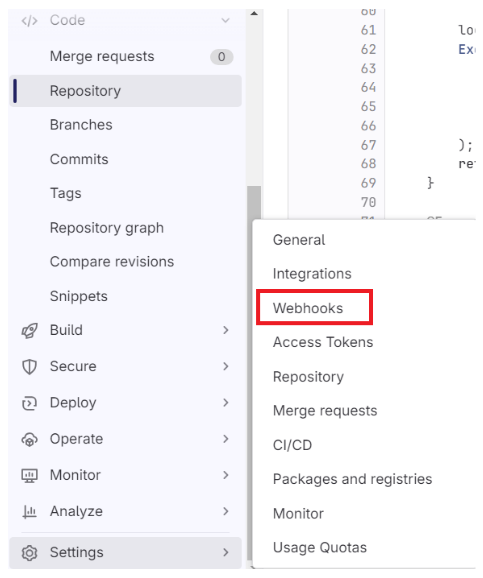
Script Path ?

backend/Jenkinsfile

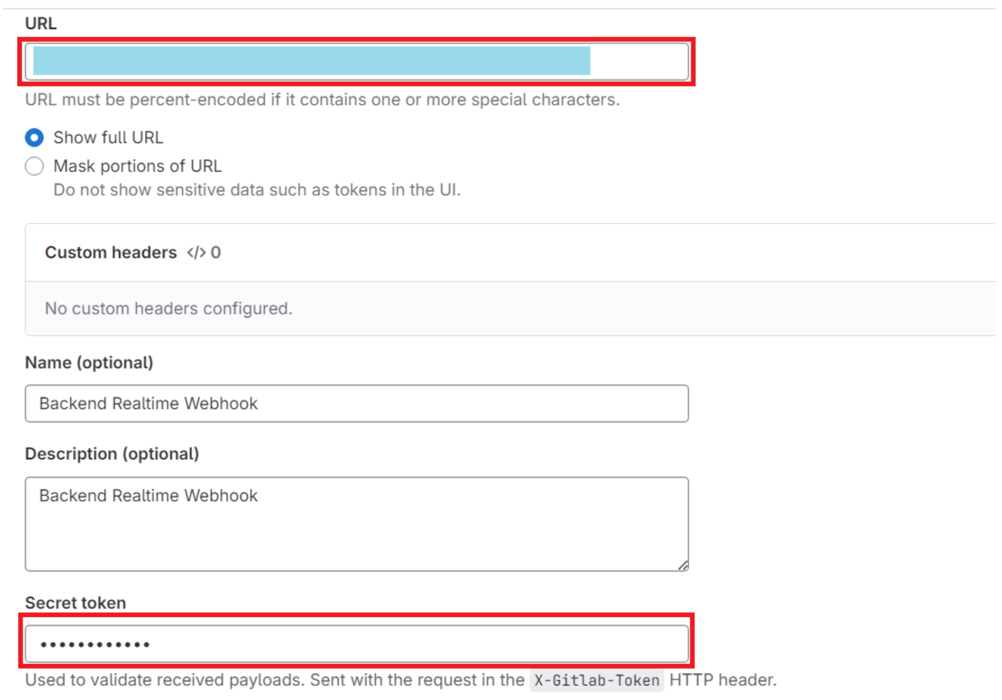
☐ Lightweight checkout ?

4.2.3. GitLab Webhook 설정

1. 프로젝트 GitLab → “Settings” → “Webhooks” 클릭



2. "URL"에 사전에 복사해놓은 Jenkins URL 입력 → "Secret token"에 사전에 복사해놓은 Secret token 입력 → "Merge request events" 클릭 후 WebHook 적용 브랜치 입력 (Jenkins Branch Specifier과 일치하여야 함)



URL

[redacted]

URL must be percent-encoded if it contains one or more special characters.

- Show full URL
- Mask portions of URL
  Do not show sensitive data such as tokens in the UI.

Custom headers </> 0

No custom headers configured.

Name (optional)

Backend Realtime Webhook

Description (optional)

Backend Realtime Webhook

Secret token

••••••••••••

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

4.2.4. 빌드 및 배포

> Option 1. 상기 WebHook 설정한 브랜치로 merge

> Option 2. Jenkins 홈 화면 → Jenkins Item 클릭 → "지금 빌드" 클릭

5. Redis 설정

5.1. Redis 설정

docker-compose.yml 내부

```
        redis:
                image: redis:latest
                restart: unless-stopped
                command: redis-server --requirepass plog
                expose:
                        - "6379"

        redis-realtime:
                image: redis:latest
                restart: unless-stopped
                command: redis-server --port 6380 --requirepass plog
                expose:
                        - "6380"
```

5.2. Docker redis-cli

```
docker exec -it plog-redis-1 /bin/bash
redis-cli
AUTH plog
```