😀

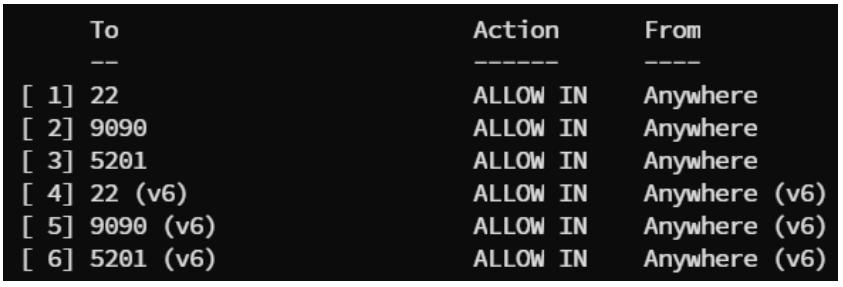
Spring CI/CD Jenkins

SpringBoot 2.7.16, Github

포트 등록

```
# 포트 등록 5201 spring, 9090 jenkins
sudo ufw allow 5201
sudo ufw allow 9090

#포트 정상 등록확인
sudo ufw status numbered
```



docker, docker compose 설치

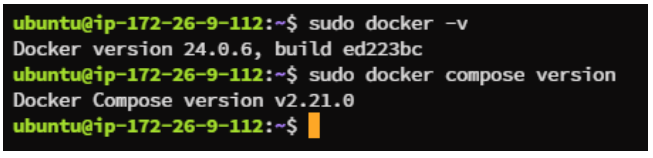
```
# docker, docker compose 설치 - https://docs.docker.com/engine/install/ubuntu/
sudo apt-get update
sudo apt-get install ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
  "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

# 설치 확인
sudo docker -v
sudo docker compose version
```



Yml 작성

```
sudo vim docker-compose.yml
```

```
version: '3'

services:
        jenkins:
                image: jenkins/jenkins:lts
                container_name: jenkins
                volumes:
                        - /var/run/docker.sock:/var/run/docker.sock
                        - /jenkins:/var/jenkins_home
                ports:
                        - "9090:8080"
                user: root
```

Jenkins 설치 후 컨테이너 접속

```
# Jenkins 설치
sudo docker compose up -d

# 확인
sudo docker ps

# 컨테이너 접속
sudo docker exec -it jenkins /bin/bash
```

```
ubuntu@ip-172-26-9-112:~$ sudo docker ps
CONTAINER ID   IMAGE                 COMMAND                  CREATED         STATUS         PORTS                                                                                  NAMES
3a1ce343dafa   jenkins/jenkins:lts   "/usr/bin/tini -- /u…"   56 seconds ago   Up 52 seconds   50000/tcp, 0.0.0.0:9090->8080/tcp, :::9090->8080/tcp   jenkins
ubuntu@ip-172-26-9-112:~$
```

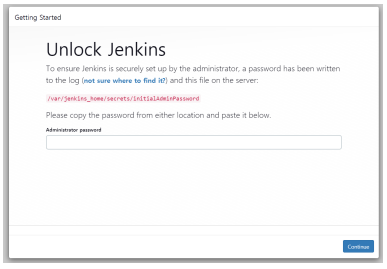
Jenkins 접속 후 초기 로그인

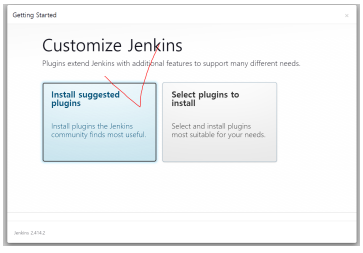
접속 url : http://j9s006a.p.ssafy.io:9090/

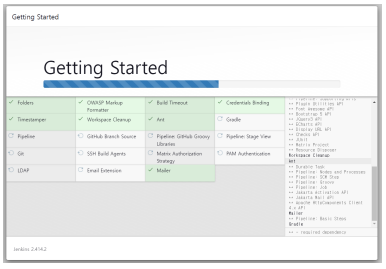
```
# 젠킨스 처음접속 비밀번호 확인
cat /var/jenkins_home/secrets/initialAdminPassword
```

```
root@3a1ce343dafa:/# cat /var/jenkins_home/secrets/initialAdminPassword
9a24959d880d40a89df6464c012ce794
root@3a1ce343dafa:/#
```

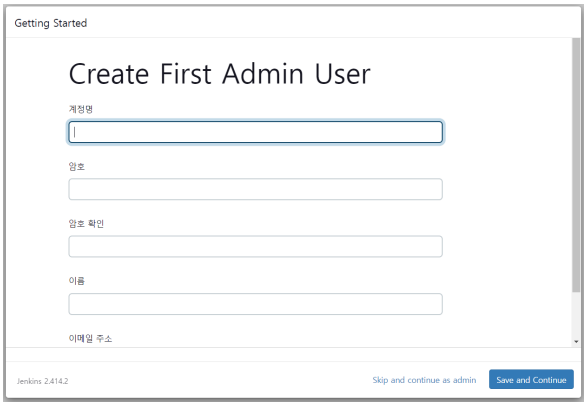
Administrator password 에 비밀번호 입력 후 Continue → Install suggested plugins 클릭 → 설치(시간 좀 걸림)

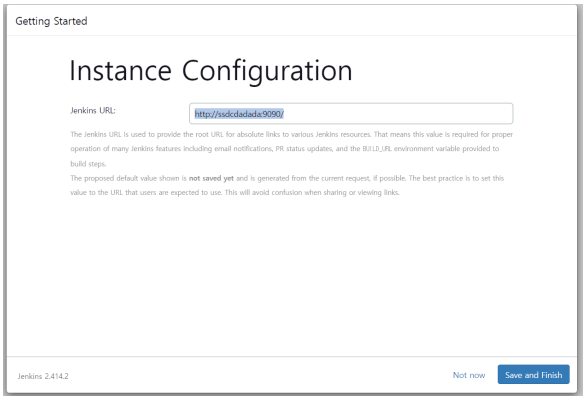






계정 생성 → 젠킨스 url 설정





계정명: dadada

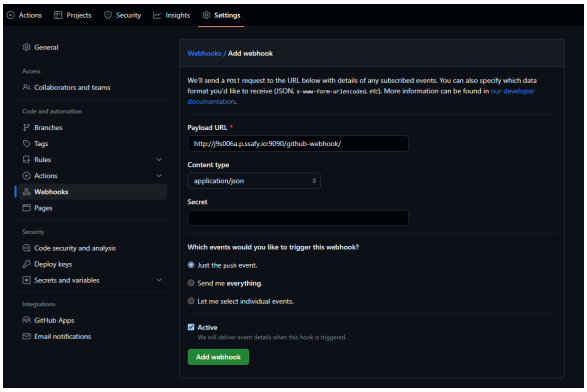
암호: Cb00N6ryX/BMJWplbYIL

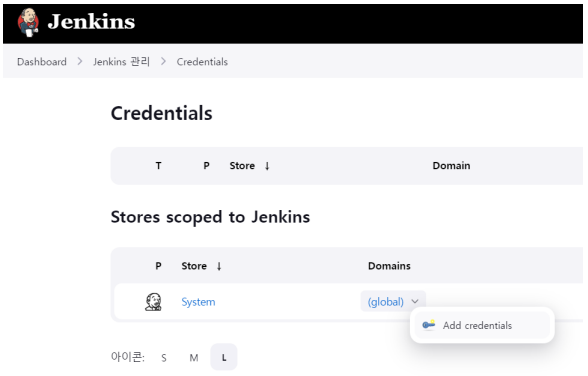
http://ssdcdadada:9090/

Jenkins-Github 연결

Webhooks 설정

http://j9s006a.p.ssafy.io:9090/github-webhook/

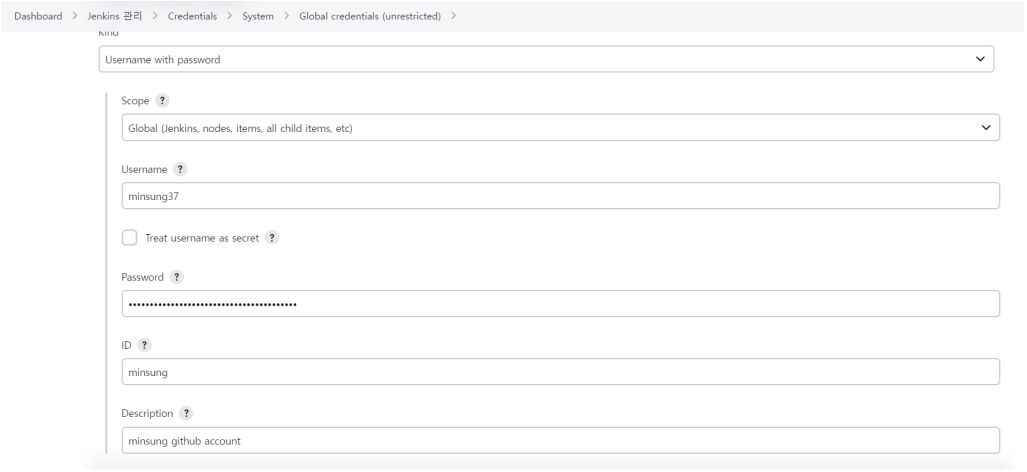




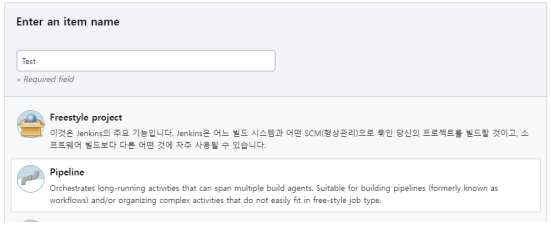
Github → Settings → Developer Settings → Personal access tokens 발급(repo, admin:org, admin:repo_hook 발급받음)

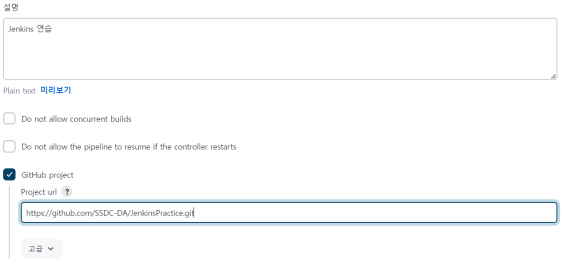
Dashboard → jenkins관리 → Credentials → Add credentials

Password 발급받은 Access token



Pipeline





project url : https://github.com/SSDC-DA/JenkinsPractice.git

GitHub hook trigger for GITScm polling 선택

Script

```
pipeline {
    agent any

    environment {
        GIT_URL = "https://github.com/SSDC-DA/JenkinsPractice.git"
        CONTAINER_NAME = "test-docker"
        IMAGE_NAME = "test"
        CONFIG_PATH = "/var/jenkins_home/backend-config"
        SPRING_RESOURCE_PATH = "src/main/resources"
        VOLUME_NAME = "spring-volume"
    }

    stages {
        stage('Git clone') {
            steps {
                git branch: 'develop',
                    url: "${GIT_URL}",
                    credentialsId: "minsung"
            }

            post {
                success {
                    sh 'echo "Successfully Cloned Repository"'
                }
                failure {
                    sh 'echo "Fail Cloned Repository"'
                }
            }
        }
```

```
        stage('Build And Test') {
            steps {
                // 설정파일 복사
                sh "cp ${CONFIG_PATH}/application.yml ${SPRING_RESOURCE_PATH}/application.yml"

              // gralew이 있어야됨. git clone해서 project를 가져옴.
                sh '''
                    chmod +x ./gradlew
                    ./gradlew clean build
                '''
            }
            post {
                success {
                    echo 'gradle build success'
                }

                failure {
                    echo 'gradle build failed'
                }
            }
        }

        stage('Docker delete') {
            steps {
                script {
                    try {
                        // 컨테이너가 존재하면 삭제합니다.
                        sh "docker stop ${CONTAINER_NAME}"
                        sh "docker rm -f ${CONTAINER_NAME}"
                    } catch (Exception e) {
                        // 컨테이너가 존재하지 않는 경우 에러가 발생할 수 있으므로, 에러를 무시합니다.
                        echo "Docker container ${CONTAINER_NAME} does not exist. Skipping deletion."
                    }

                    try {
                        // 이미지가 존재하면 삭제합니다.
                        sh "docker image rm ${IMAGE_NAME}"
                    } catch (Exception e) {
                        // 이미지가 존재하지 않는 경우 에러가 발생할 수 있으므로, 에러를 무시합니다.
                        echo "Docker image ${IMAGE_NAME} does not exist. Skipping deletion."
                    }
                }
            }


            post {
                success {
                    sh 'echo "docker delete Success"'
                }
                failure {
                    sh 'echo "docker delete Fail"'
                }
            }
        }

        stage('Dockerizing'){
            steps{
                sh 'echo " Image Bulid Start"'
                sh """
                    docker build -t ${IMAGE_NAME} .
                """
            }
            post {
                success {
                    sh 'echo "Bulid Docker Image Success"'
                }

                failure {
                    sh 'echo "Bulid Docker Image Fail"'
                }
            }
        }

        stage('Deploy') {
            steps {
                sh "docker run --name ${CONTAINER_NAME} -v ${VOLUME_NAME}:/app/profile -d -p 5201:8080 ${IMAGE_NAME}"
            }

            post {
```

```
                success {
                    echo 'deploy success'
                }

                failure {
                    echo 'deploy failed'
                }
            }
        }
    }
}
```

application.yml 작성

```
# 컨테이너 안에 vim 설치
apt-get update
apt-get install vim

cd /var/jenkins_home
mkdir backend-config
cd backend-config

vi application.yml
```

application.yml

```
SECRET: hello jenkins
```

컨테이너에 도커 설치

```
# 도커 컨테이너 도커 설치
apt install docker

# 권한 설정
chmod 666 /var/run/docker.sock

# /var/jenkins_home/workspace/Test@tmp/durable-89d78643/script.sh: 2: docker: not found 오류해결
apt-get install docker.io
```