



포팅 매뉴얼

안녕하세요!

웅골찬 팀의 “ 돈네 한바퀴 “ 포팅 매뉴얼입니다 ! 😊

1) 개발 환경

2) 설정 파일 및 환경 변수 정보

(1) Application.Yml

(2) Nginx 설치

(3) Nginx Conf 파일

(4) Docker 설정

(5) Docker - Compose 설정

(6) 도커 네트워크 설정

3) 방화벽

4) 빌드

Docker 파일과 Docker - compose 아를 파일 아주 중요!! 들여쓰기 조심!!

5) 자동 배포

Jenkins 설치

Jenkins 시작하기

1) 개발 환경

- Server : Ubuntu 20.04.6 LTS
- SpringBoot : 3.1.1
- Spring Security : 6
- JDK : OpenJDK 17
- Nginx : nginx/1.18.0 (Ubuntu)
- MariaDB : MariaDB Server 10.3.38
- Jenkins : 2.401.3 (latest)
- Docker : 24.0.5 (latest)
- Docker-Compose : v2.20.2
- Flutter : allowed latest version
- Android Studio : allowed any version
- IntelliJ : allowed any version
- Redis : allowed any version
- JWT : no version

2) 설정 파일 및 환경 변수 정보

(1) Application.Yml

```
server:
  port:
    5000

spring:
  jpa:
    hibernate:
      ddl-auto: update
    properties:
      hibernate:
        format_sql: true
        show_sql: true

datasource:
  driver-class-name: org.mariadb.jdbc.Driver
  url: jdbc:mariadb://ssafy-db:3306/donnearound?useUnicode=true
  username: ssafy
  password: {password}

data:
  redis:
    host: docker-redis
    port: 6379

fcm:
  service-account-file: /app/donnearound-java-access-key.json #비밀 키 경로

jwt:
  secret: {secret_key}

exchange:
  key: {password}
root@b8ef94a97242:/var/jenkins_home/workspace/BE_moneyallaround/backend/moneyallaround/src/main/resource
```

(2) Nginx 설치

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install nginx
```

```
sudo apt-get -y remove --purge nginx nginx-full nginx-common //삭제
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org. Commercial support is available at nginx.com.

Thank you for using nginx.

(3) Nginx Conf 파일

```
sudo vim /etc/nginx/conf.d/default.conf
```

```
upstream frontend {
    server localhost:3000;
}
upstream backend {
    server localhost:8080;
}
server {
    listen 80;
    server_name j9a705.p.ssafy.io;

    location /api {
        rewrite ^/api(/.*)$ $1 break;
        proxy_pass http://backend;
        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;
    }

    location / {
        proxy_pass http://frontend;
        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;
    }
}

// 저장 후 종료 하 세 요 !!!
: wq!
```

(4) Docker 설정

설치하기 및 시작

```
/ 를 기점으로 1줄 씩 입력 하세요
sudo apt-get -y install apt-transport-https ca-certificates curl gnupg-agent software-properties-common /
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) s
table" /
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io /

docker -v
docker version
```

(5) Docker - Compose 설정

설치하기

```
/ 를 기점으로 1줄 씩 입력 하세요

sudo apt install jq /
DCVERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r) /
DCDESTINATION=/usr/bin/docker-compose /
sudo curl -L https://github.com/docker/compose/releases/download/${DCVERSION}/docker-compose-$(uname -
s)-$(uname -m) -o $DCDESTINATION /
sudo chmod 755 $DCDESTINATION /
docker-compose -v
```

```
ubuntu@ip-172-26-2-39:~$ docker-compose -v
Docker Compose version v2.20.2
```

(6) 도커 네트워크 설정

```
docker network create deploy

docker network inspect $(docker network ls -q) // 네트워크 확인
docker container inspect ssafy-db //컨테이너 상세 정보 확인
```

3) 방화벽

UFW 설정하기 및 존재하지 않는다면 설치

```
sudo apt install ufw
```

```
sudo ufw default deny incoming // 모든 인바운드 연결 차단
sudo ufw default allow outgoing // 모든 아웃바운드 연결 허용
sudo ufw allow ssh // 22번 포트 허용
sudo ufw allow http // 80번 포트 허용
sudo ufw allow https // 443번 포트 허용

sudo ufw enable // 방화벽 켜기
```

```
ubuntu@bossniceshot:~$ sudo ufw status
Status: active

To Action From
--
80/tcp ALLOW Anywhere
22/tcp ALLOW Anywhere
443/tcp ALLOW Anywhere
3478/tcp ALLOW Anywhere
3478/udp ALLOW Anywhere
40000:57000/tcp ALLOW Anywhere
40000:57000/udp ALLOW Anywhere
57001:65535/tcp ALLOW Anywhere
57001:65535/udp ALLOW Anywhere
Nginx HTTP ALLOW Anywhere
```

4) 빌드

- *Back-spring*
Gradle 실행
Bootjar 실행
- *Front*
npm import —force
npm start

Docker 파일과 Docker - compose 야물 파일 아주 중요!! 들여쓰기 조심!!

Docker File - BE

```
# Stage 1: Build with Gradle
FROM gradle:8.1.1-jdk17 as builder
WORKDIR /workspace
COPY build.gradle settings.gradle /workspace/
COPY src /workspace/src/
RUN gradle build -x test --no-daemon

# Stage 2: Create a minimal JRE-based image for running the application
FROM eclipse-temurin:17-jdk-jammy
WORKDIR /app
COPY --from=builder /workspace/build/libs/*.jar app.jar

COPY ./src/main/resources/firebase/donnearound-java-access-key.json /app
CMD ["java", "-jar", "app.jar"]
```

- 멀티 스테이지 빌드 (젠킨스에서 직접 빌드하지 않음)
- 버전 문제 주의 (젠킨스와 JAVA의 버전이 맞지 않으면 전부 처음부터 설정해야 할 수도 있음)

Docker - compose.yml - BE

```
version: "3.8"
services:
  redis-docker:
    image: redis:latest
    container_name: docker-redis
    volumes:
      - docker-redis:/data
    ports:
      - 6380:6380
    networks:
      - deploy

  application:
    build:
      context: /var/jenkins_home/workspace/BE_moneyallaround/backend/moneyallaround/
      dockerfile: Dockerfile
    environment:
      SPRING_DATASOURCE_URL: jdbc:mariadb://ssafy-db:3306/donnearound?useUnicode=true
      SPRING_DATASOURCE_USERNAME: ssafy
      SPRING_DATASOURCE_PASSWORD: ssafy
    ports:
      - 5000:5000
    networks:
```

```
- deploy

networks:
  deploy:
    external: true

volumes:
  docker-redis:
```

- **docker run**
 - 도커 컨테이너 시작

```
docker run [options] image [command] [argument]
```

- **docker ps**
 - 실행 중인 도커 나열 / -a 를 붙이면 모든 컨테이너 나열 (실행 중, 중지)

```
docker ps -a
```

- **docker stop**
 - 특정 도커 컨테이너 중지

- **docker rm**
 - 특정 도커 컨테이너 삭제

- **docker images**
 - 로컬에 저장된 도커 이미지 나열

```
docker stop container_id
docker rm container_id
docker images -a

//그 외 명령어들 참고

docker rmi image_id //특정 도커 이미지 삭제
docker build -t image_name:tag <Dockerfile이 있는 위치> //도커 파일 기반으로 이미지 빌드
```

5) 자동 배포

Jenkins 설치

Jenkins 설치(Windows 환경)(1)

우선 Jenkins는 흔히 말하는 CI/CD 중 지속적 통합(Continuous Integration)을 구현하기 위한 서비스이다. 개발 중인 저장소(git, svn 등)에 업로드된 소스를 테스트, 빌드, 빌드 후 작업 등을 자동 동작하게 해 주어 (이 자체가 지속적 통합) 그만큼 개발자의 리소스 소모가 줄어든다. 1. Installer Download <https://www.jenkins.io/download/> Jenkins download and deployment Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software..

 <https://lock.tistory.com/2>

Click the Finish button to exit the Setup Wizard.




Swap 메모리 선언


```
df -h # 용량 할당
sudo fallocate -l 8G /swapfile # Swap 영역 할당 (일반적으로 서버 메모리의 2배)
sudo chmod 600 /swapfile # Swapfile 권한 수정
sudo mkswap /swapfile # Swapfile 생성
sudo swapon /swapfile # Swapfile 활성화
free -h # swap 영역이 할당 되었는지 확인
```

Jenkins nginx 설정

Reverse proxy - Nginx

Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software

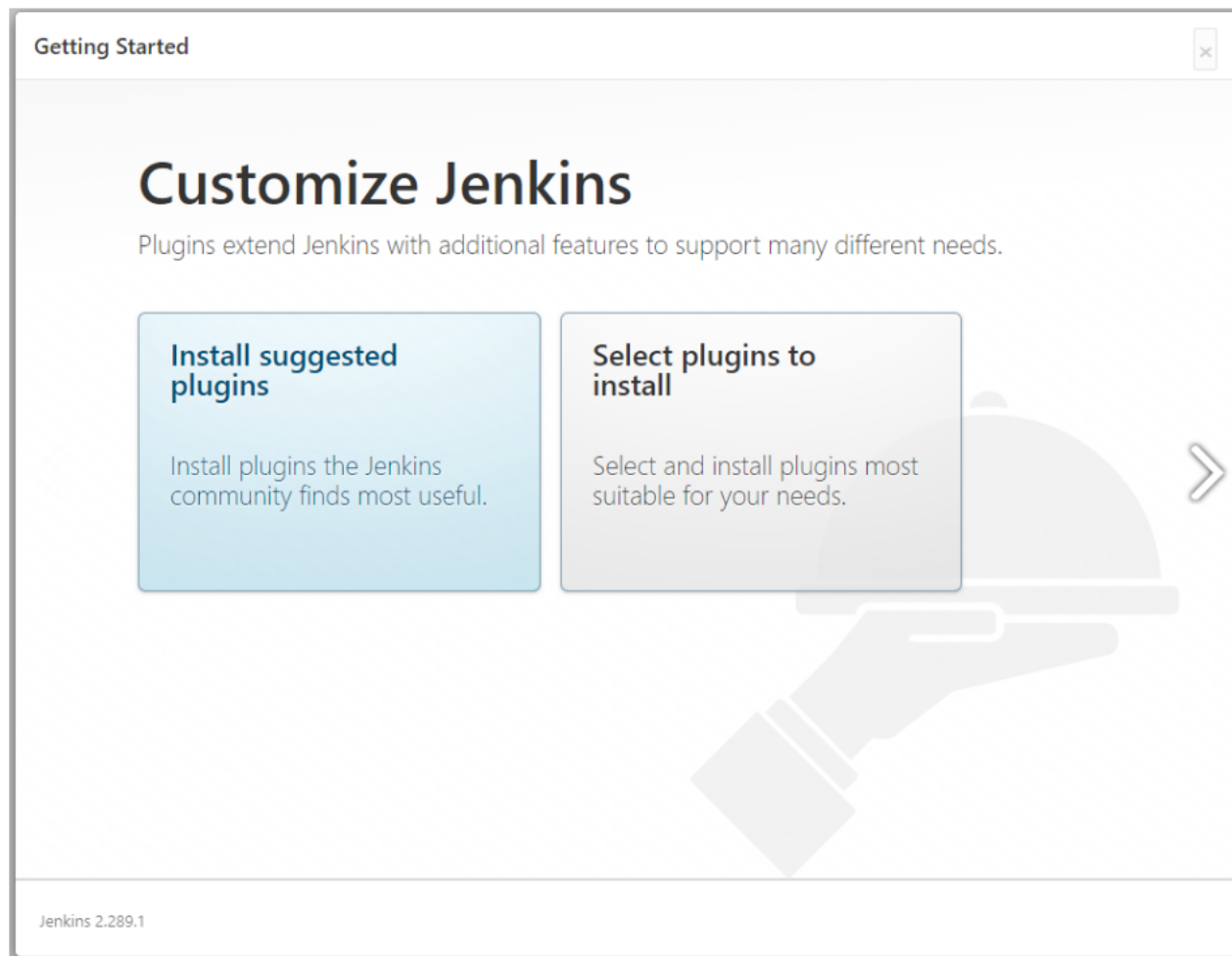
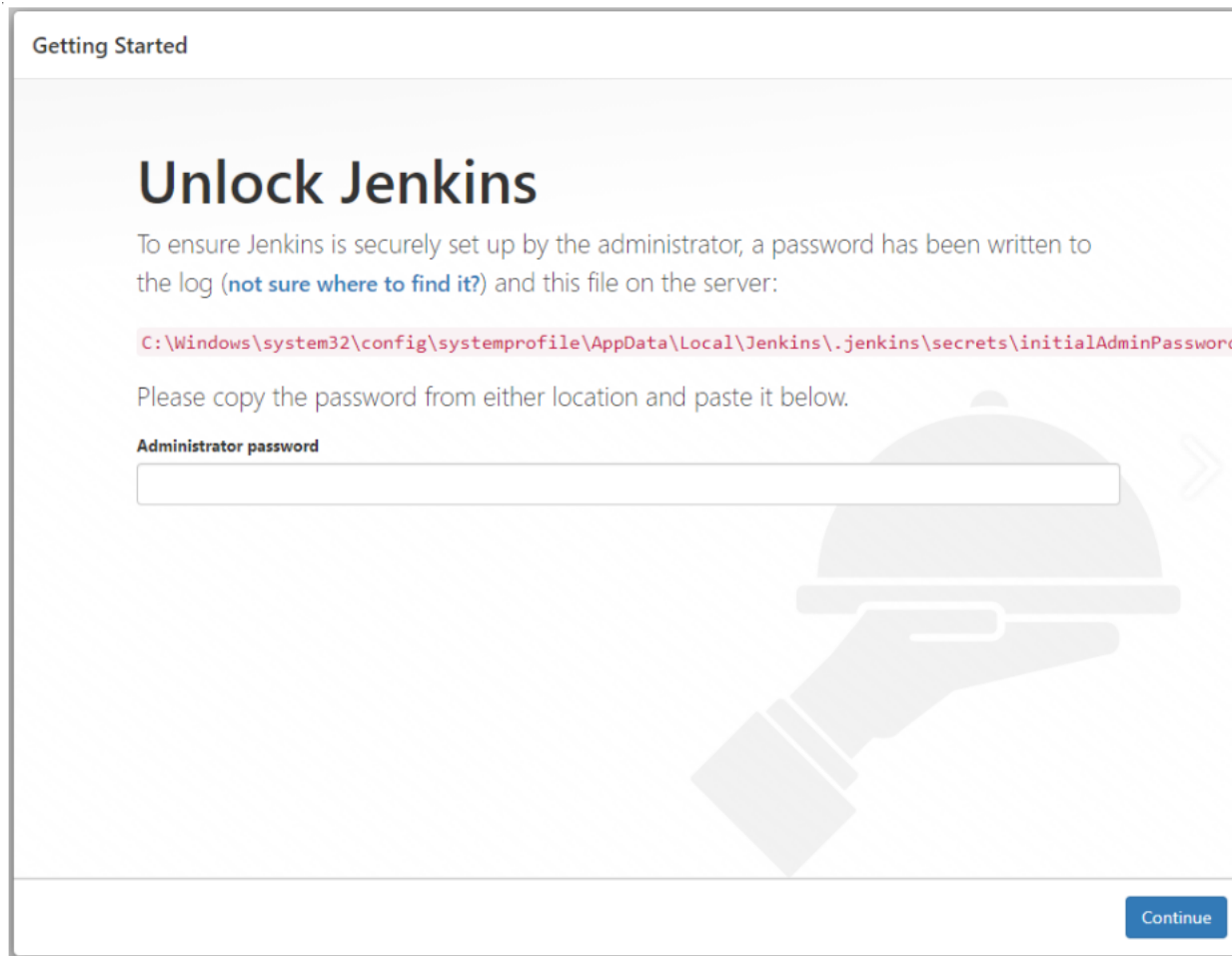
 <https://www.jenkins.io/doc/book/system-administration/reverse-proxy-configuration-with-jenkins/reverse-proxy-configuration-nginx/>

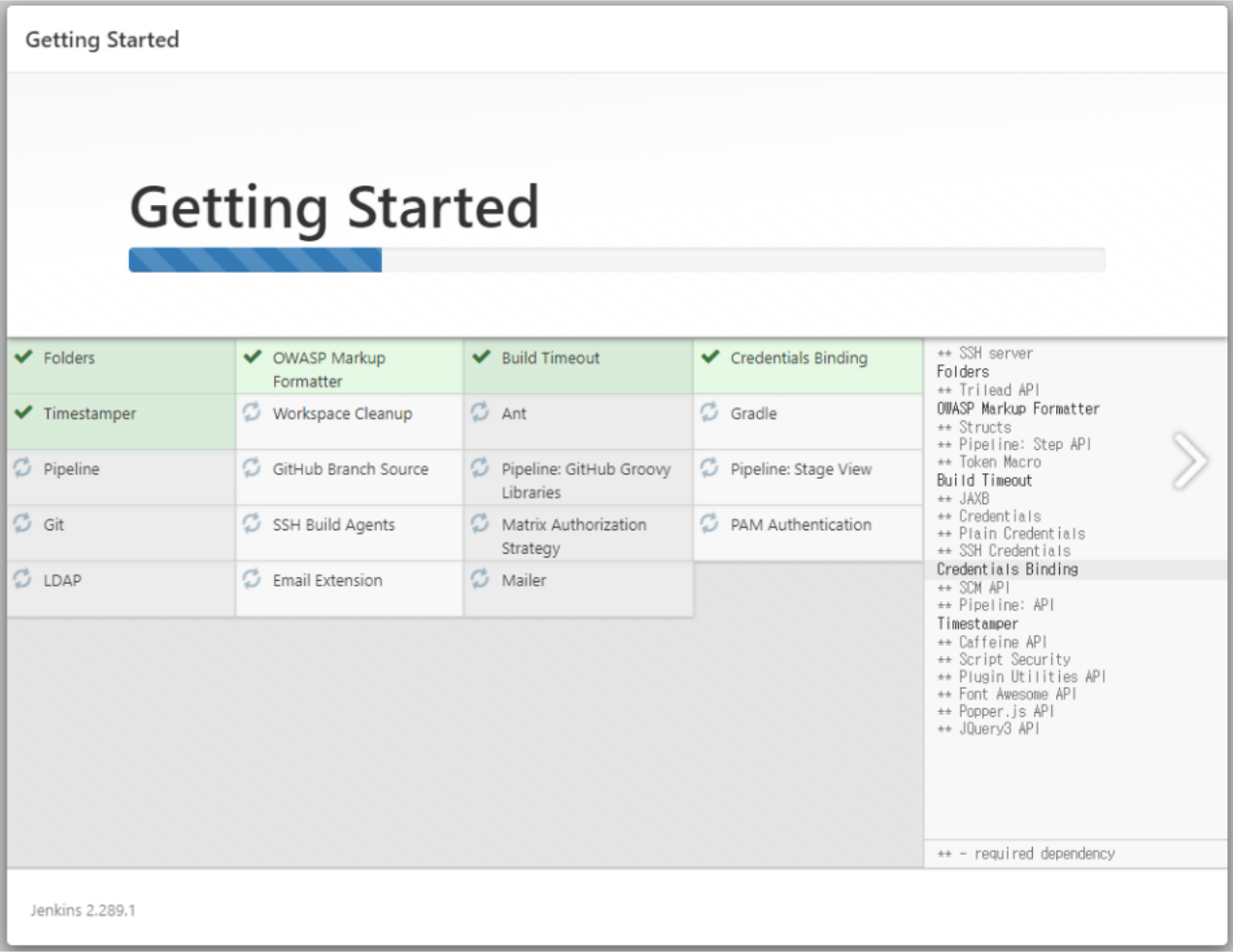


Jenkins

Jenkins 시작하기

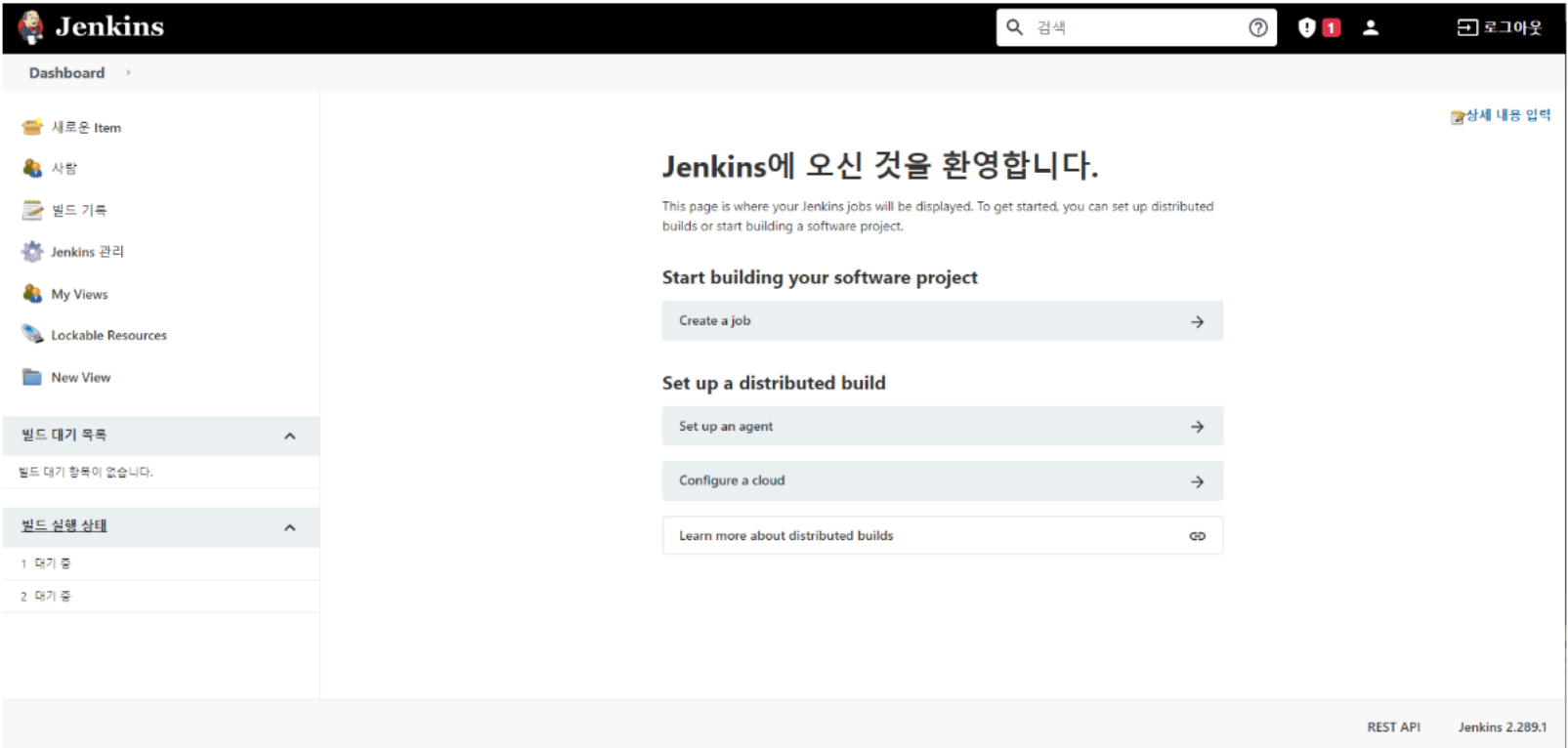
Docker logs jenkins 비밀번호 확인 후 복사





이후에 Admin User Setting 하고
앞서 설치 시 설정한 Port 로 설정된 접속 URL 창 입력한다

그러면 젠킨스 사용 가능 !!



Git Lab 연동은 다음 페이지 참고

Jenkins - Git 연동 (1) - 연결

1. Git 설치 후 Jenkins 연동 <https://git-scm.com/> 깃 설치 안된 상태이면, 해당 사이트 들어가서 설치. 설치 후, Jenkins 관리 -> Global Tool Configuration 들어가서 설치된 깃 정보 입력. 2. 깃 계정 AccessToken 확인 깃 Access Tokens 확인 깃 로그인 후 계정 아이콘 드롭 메뉴 -> Preferences -> 왼쪽 사이트 메뉴의 Access Tokens (혹은 아이콘 드롭메뉴 -> Settings -> 상위 메뉴의 Access Tokens) 혹은 Access Token 페이지 들어가서, 토큰 생성 혹은, 이렇게 토큰 생성. Scopes는 다 체크해주고, Expires at 은 미 설정 시 기한 제한 없음

<https://lock.tistory.com/6>

Jenkins-Token

Expires at

Scopes

- ☒ api Access your API
- ☒ read_user Read user information

Jenkins Script - BE

```
pipeline{
  agent any

  options{
    timeout(time: 1, unit: 'HOURS')
  }

  stages{
    stage('git clone') {
      steps {
        git url: "https://lab.ssafy.com/s09-fintech-finance-sub2/S09P22A705.git",
          branch: "backend",
          credentialsId: "moneyallaround"
      }
    }
  }
}
```

```

        sh "ls -al"
    }
}

stage('set backend enviornment'){
    steps{
        dir("./backend/moneyallaround"){
            sh '''
                cp /var/jenkins_home/util/BE/moneyallaround/Dockerfile /var/jenkins_home/workspace/BE_moneyallaround/backend/moneyallaround/Dockerfile
                cp /var/jenkins_home/util/BE/moneyallaround/docker-compose.yml /var/jenkins_home/workspace/BE_moneyallaround/backend/docker-compose.yml
                cp /var/jenkins_home/util/BE/moneyallaround/application.yml /var/jenkins_home/workspace/BE_moneyallaround/backend/moneyallaround/src/main/resources/application.yml
                cp /var/jenkins_home/util/BE/moneyallaround/donnearound-java-access-key.json /var/jenkins_home/workspace/BE_moneyallaround/backend/moneyallaround/src/main/resources/firebase/donnearound
            '''
            // sh "chmod +x ./gradlew"
            // sh "./gradlew clean"
            // sh "./gradlew build -x test"
        }
    }
}

stage('Docker down'){
    steps{
        dir("/var/jenkins_home/workspace/BE_moneyallaround/backend"){
            echo "Docker compose down"
            sh "docker-compose -f docker-compose.yml down --rmi all"
            sh "docker ps -a"
        }
    }
}

stage('Docker build'){
    steps{
        echo "docker compose build"
        dir("/var/jenkins_home/workspace/BE_moneyallaround/backend"){
            sh "docker-compose -f docker-compose.yml build --no-cache"
        }
    }
    post{
        success{
            echo "Success to build"
        }
        failure{
            echo "Docker build failed. clear unused file"
            sh "docker system prune -f"
            error 'pipeline aborted'
        }
    }
}

stage('Docker up'){
    steps{
        echo "docker compose up"
        sh "docker-compose -f /var/jenkins_home/workspace/BE_moneyallaround/backend/docker-compose.yml up -d"
    }
}

stage('Docker clear'){
    steps {
        sh "sudo docker system prune -f"
    }
}
}
}

```

- 파이어베이스 SDK 파일에 JSON 형태로 키가 들어있기 때문에 보안 상으로 굉장히 중요해서 git ignore에 추가
- 그렇기 때문에 젠킨스에서 빌드될 때 해당 파일이 같이 되지 않아서, 원활한 서비스 운영을 위해 EC2에 파일을 따로 업로드 하고 빌드 하는 설정 필수 (위 스크립트에 포함되어 있음)