

Dev, Back-End/CI & CD

[JenKins] ⑤ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS)

TROLL's | 2021. 6. 4. 17:17

⑤
Jenkins를 이용해서
node.js CI/CD
자동 배포하는 방법
(NestJS)

JenKins를 이용해서 node.js 배포하는 방법 (NestJS) 4편에 이어서

5편을 포스팅하겠습니다.

▼ Jenkins를 통해 자동배포 (NodeJS) 시리즈

- ▶ [① 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 - 바로가기](#)
- ▶ [② 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 - 바로가기](#)
- ▶ [③ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 - 바로가기](#)
- ▶ [④ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 - 바로가기](#)
- ▶ [⑤ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 - 바로가기](#)



● EC2 SSH Authentication

```
ubuntu@ip-10-10-10-10:~$ sudo su - jenkins
jenkins@ip-10-10-10-10:~$
```

\$ sudo su -jenkins

1. Jenkins 서버로 SSH 접속해서 젠킨스 사용자로 유저를 변경합니다.

```
jenkins@ip- ~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
[Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Created directory '/var/lib/jenkins/.ssh'.
[Enter passphrase (empty for no passphrase):
[Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:

The key's randomart image is:
```

```
jenkins@ip- ~$
```

\$ ssh-keygen -t rsa

2. SSH키를 발행하고 계속 엔터를 누르고 진행합니다.

```
jenkins@ip- ~$ cat ~/.ssh/id_rsa.pub
jenkins@ip- ~$
```

\$ cat ~/.ssh/id_rsa.pub

3. 파란부분을 복사해줍니다.

```
[ubuntu@ip- ~]$ vim ~/.ssh/authorized_keys
[ubuntu@ip- ~]$
```

4. vim으로 해당 파일을 들어가서 위에서 복사한 것을 붙여넣기합니다.

```
ubuntu@ip- ~$ chmod 700 ~/.ssh
ubuntu@ip- ~$ chmod 600 ~/.ssh/*
ubuntu@ip- ~$
```

```
$ chmod 700 ~/.ssh
```

```
$ chmod 600 ~/.ssh/*
```

5. .ssh 폴더 권한을 줍니다.

```
[jenkins@ip- ~]$ ssh ubuntu@
```

```
Last login: Thu May 20 11:46:48 2021 from 1.230.111.97
ubuntu@ip-~$
ubuntu@ip-~$
```

\$ ssh ubuntu@NODE.APP.SERVER.IP

6. 젠킨스 서버로 가서 node서버에 접속이 가능한지 확인합니다.

위와 같이 ssh접속이 되면 성공한것입니다.

● 자동 배포 세팅

```
1  #!/bin/sh
2  ssh ubuntu@[REDACTED] <<EOF
3      cd ~/ovni
4      git pull origin master
5      curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.8/install.sh | bash
6      . ~/.nvm/nvm.sh
7      nvm install 16.2.0
8      npm install
9      npm install -g typescript
10     npm install -g ts-node
11     npm install -g pm2
12     pm2 install typescript
13     pm2 restart ecosystem.config.js
14     exit
15 EOF
```

7. node 앱에 배포를 위한 shell script가 필요하기 때문에 script 디렉토리 아래 deploy파일을 생성하여 위처럼 입력합니다. (확장자 없이)

GitHub 주소 : <https://github.com/choseongho93/Jenest/blob/master/script/deploy>

※ deploy파일 내용 설명

위에서 젠킨스 서버가 Node서버로 SSH접속할수있도록 입력해줍니다.

검정박스는 Node서버 IP주소를 입력하시면 됩니다.

기본 디렉토리가 /home/ubuntu로 되어있고, cd를 통해 프로젝트명으로 이동합니다.
git pull을 받고 npm을 설치해주는 과정을 거치고 ts를 js로 변환할수있도록 설치를 진행합니다.
node프로세스 관리도구인 pm2를 설치하여 run해주도록 ecosystem.config.js파일을 실행합니다.

```
$ chmod +x script/deploy
```

8. 방금 만든 depoy파일 권한을 부여합니다.

```
1  module.exports = {  
2    apps: [{  
3      name   : "onvi",  
4      script : "npm",  
5      args   : "start"  
6    }]  
7  }
```

9. 루트 디렉토리 아래 ecosystem.config.js파일을 만들어주고 위와 같이 내용을 넣어줍니다.

GitHub 주소 : <https://github.com/choseongho93/Jenest/blob/master/ecosystem.config.js>

● 테스트

```

1  import { Injectable } from '@nestjs/common';
2
3  @Injectable()
4  export class AppService {
5    getHello(): string {
6      return 'Hello World! ovni!';
7    }
8  }

```

10. src/app.service.ts 파일을 위와 같이 텍스트를 수정해줍니다.
GitHub에서 Push합니다.

Jenkins Dashboard > ovni >

Workspace of ovni on master

- .git
- node_modules
- script
- src
- test
- .gitignore (2021. 5. 20. 오전 11:57:36, 549 B 보기)
- Dockerfile (2021. 5. 20. 오전 11:57:36, 490 B 보기)
- ecosystem.config.js (2021. 5. 20. 오후 12:04:47, 115 B 보기)
- nest-cli.json (2021. 5. 20. 오전 11:57:36, 64 B 보기)
- package.json (2021. 5. 20. 오전 11:57:36, 1.92 KB 보기)
- package-lock.json (2021. 5. 20. 오후 12:22:36, 651.03 KB 보기)
- README.md (2021. 5. 20. 오전 11:57:36, 3.31 KB 보기)
- tsconfig.build.json (2021. 5. 20. 오전 11:57:36, 97 B 보기)
- tsconfig.json (2021. 5. 20. 오전 11:57:36, 339 B 보기)

(zip 파일로 압축)

Build History 추이

find

#4 2021. 5. 20. 오후 12:22

11. 푸시를 하면 젠킨스에서 좌측 하단에 배포중임을 알수있고, 그에따른 로그도 확인가능합니다.
(초록색이 뜨면 정상 배포되었다는 의미입니다.)



Hello World! ovni!

12. Node서버 IP주소:3000으로 접속하면 자동배포가 성공된것을 확인할 수 있습니다.

※ 최종 GitHub 주소 : <https://github.com/choseongho93/Jenest>

Ref: <https://soojae.tistory.com/25>

<https://12340zszs.tistory.com/33>

<https://cheese10yun.github.io/PM2/>

[dlawogus \(임재현\) - velog](#)

블로그 내용이 도움이 되셨다면 아래 광고를 클릭해주시면 감사하겠습니다. ^^

♡ 1



구독하기



카카오스토리



트위터



페이스북

'[Dev. Back-End](#)' > [CI & CD](#) 카테고리의 다른 글

[Jenkins] job scheduling 설정하는 방법 (batch) (0)

2021.07.24

[Jenkins] Global 환경변수 등록하는 방법 (0)	2021.07.23
Jenkins를 Docker Container로 구축시에 TimeZone 설정하는 방법 (0)	2021.07.21
[Jenkins] docker를 이용해서 jenkins 설치하는 방법 (ubuntu) (0)	2021.07.11
<u>[JenKins] ⑤ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS) (0)</u>	2021.06.04
[JenKins] ④ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS) (0)	2021.06.03
[JenKins] ③ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS) (0)	2021.06.02
[JenKins] ② 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS) (0)	2021.06.01
[JenKins] ① 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS) (0)	2021.05.29

태그 #cd, #CI, #github, #javascript, #NestJS, #node, #Typescript, #빌드, #자동배포, #젠킨스

'Dev. Back-End/CI & CD' Related Articles

**Jenkins를
Docker Container로
구축시에
TimeZone
설정하는 방법**

Jenkins를 Docker Container로 구축시에
TimeZone 설정하는 방법

**[Jenkins]
docker(도커)를
이용해서
jenkins 설치하는 방법**

[Jenkins] docker를 이용해서 jenkins 설치하는
방법 (ubuntu)

**④
Jenkins를 이용해서
node.js CI/CD
자동 배포하는 방법
(NestJS)**

**③
JenKins를 이용해서
node.js CI/CD
자동 배포하는 방법
(NestJS)**

[JenKins] ④ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS)

이름

암호

☐ Secret

여러분의 소중한 댓글을 입력해주세요.

댓글달기

[JenKins] ③ 젠킨스를 이용해서 node.js CI/CD 자동 배포하는 방법 (NestJS)