



부록 2. 무중단 Jenkins CI 구축



여러분, 한창 도커의 재미에 푹 빠져 있으시죠? 😊

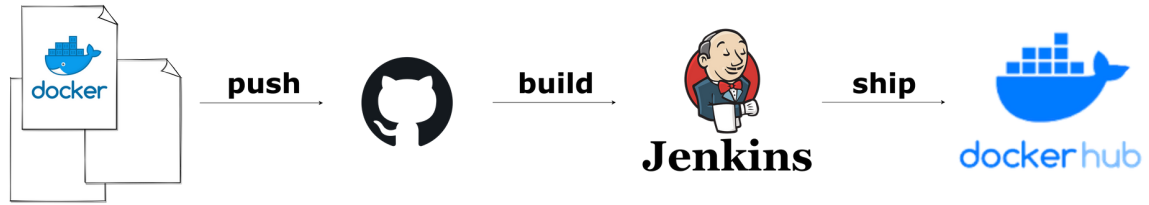
부록으로 간단한 CI 시스템 구축 실습을 준비했습니다. CI는 *Continuous Integration*의 약자로 소스 작성, 이미지 빌드/공개까지의 과정을 자동화 한 것을 가리킵니다. Oracle Cloud 에서 무료로 제공하는 VM에 Jenkins 컨테이너를 구동시키고 Github 변동사항을 트래킹하여 이미지를 자동으로 빌드하게 하는 것이죠. 이어서 빌드된 이미지는 자동으로 Docker Hub에 공개되도록 세팅합니다. 실제의 CI 시스템은 좀 더 복잡하지만 본 실습을 통해 CI 파이프라인이 구성되는 방식을 이해하는 데에 도움이 되길 바랍니다 🙌



Oracle Cloud 의 정책 변동에 따라 과금이 될 수 있으니 주의 부탁드립니다.

- [Oracle Cloud](#)
- [Jenkins](#)
- [Github](#)
- [Docker Hub](#)

1. [Jenkins 컨테이너 구동 및 플러그인 설치](#)
2. [컨테이너 SSH Public/Private Key 등록](#)
3. [Docker Hub 계정 등록](#)
4. [Github WebHook 등록](#)
5. [Jenkins 파일 생성](#)
6. [Jenkins Pipeline 생성](#)
7. [Jenkins Pipeline Build](#)
8. [Build 로그 확인](#)



그림은 실습에서 구축할 CI 체계를 간단하게 나타낸 것입니다. Oracle Cloud VM을 아직 생성하지 않으셨다면 하단의 링크를 참고하여 먼저 진행 부탁드립니다.

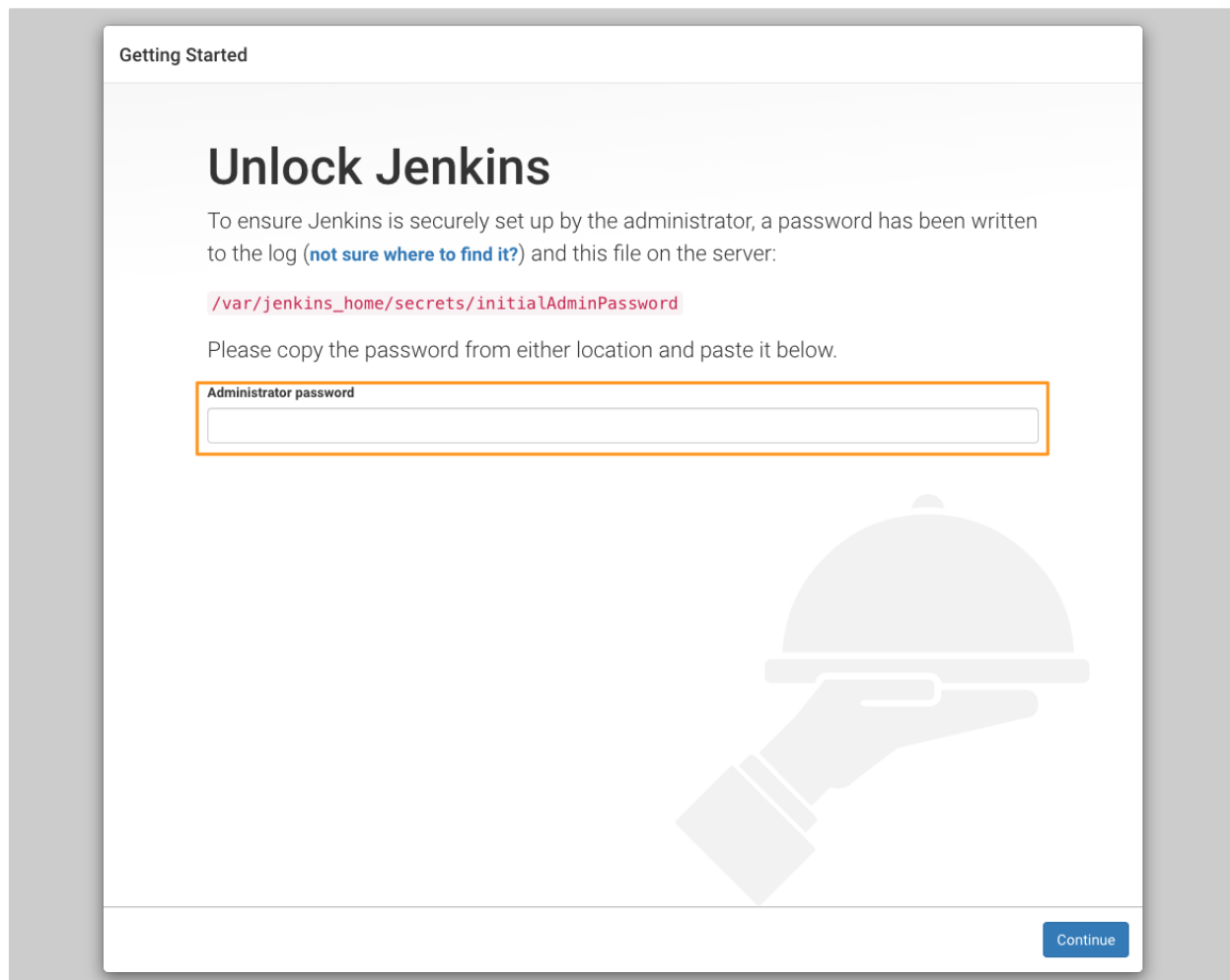
제목 없음

1. Jenkins 컨테이너 구동 및 플러그인 설치

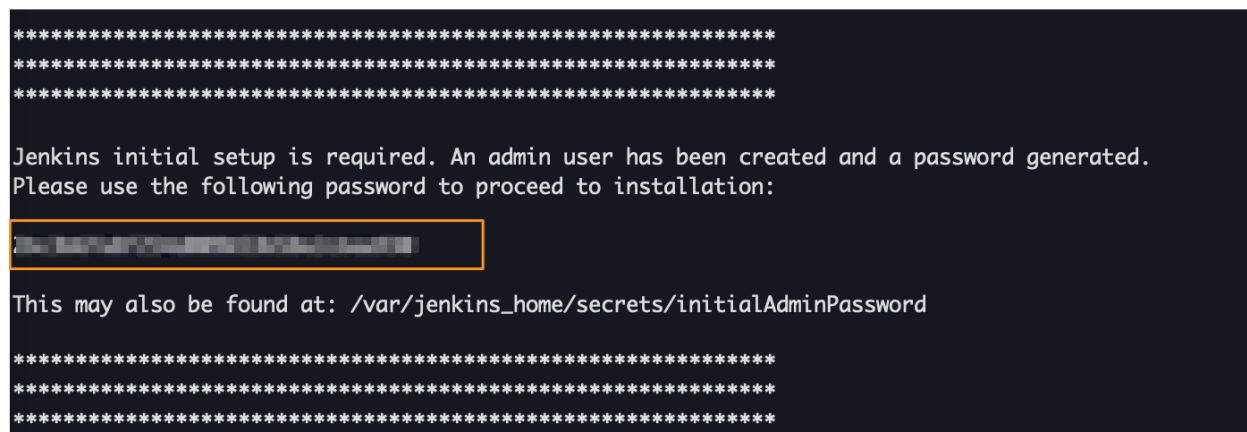
1) 다음의 명령어를 입력하여 Jenkins 컨테이너를 구동합니다. **8080**, **50000** 포트는 사전에 VM 서브넷 수신 규칙에 미리 추가합니다.

```
sudo docker run --name jenkinsci -u root -p 8080:8080 -p 50000:50000 -v $(which docker):/usr/bin/docker -v $HOME/.jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock jenkins/jenkins:lts
```

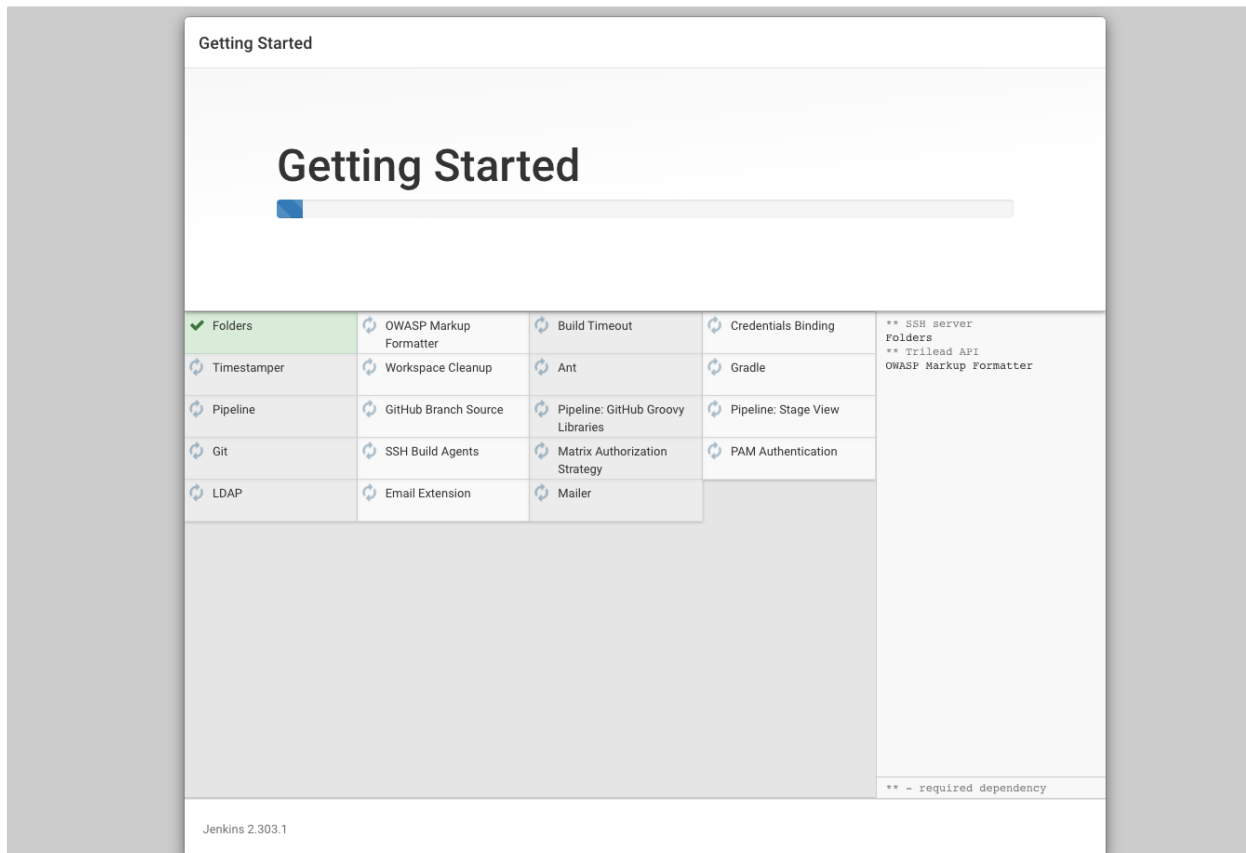
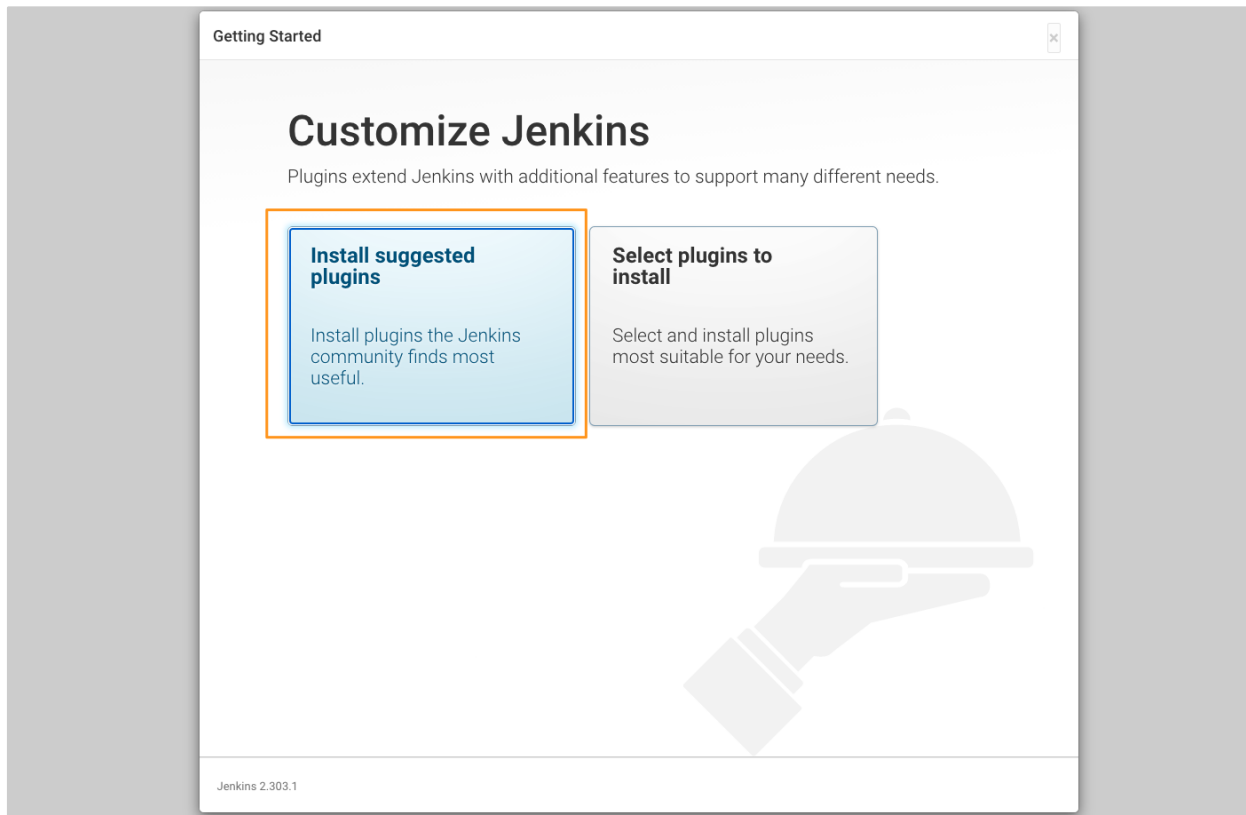
2) 브라우저에서 **<VM Instance 공용 IP>:8080** 로 접속합니다. 정상적으로 접속했다면 **Administrator Password**를 입력하는 창이 나옵니다.



3) Jenkins 를 구동한 터미널을 보면 **Administrator Password** 에 입력하는 값이 다음과 같이 출력되어 있으며, 이를 복사해 붙여 넣습니다.



4) 다음의 창에서 **Install suggested plugins** 를 선택합니다. 이어서 하단 이미지와 같이 설치가 진행되는데 시간이 다소 소요됩니다.



5) 설치가 완료되면 계정을 설정하는 창이 뜹니다. 알맞게 작성 후 **Save and Continue** 버튼을 클릭합니다.

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.303.1 Skip and continue as admin Save and Continue

6) Jenkins URL 의 값이 **<VM Instance 공용 IP>:8080** 으로 입력되어 있는 것을 확인하고 **Save and Finish** 버튼을 누릅니다.

Getting Started

Instance Configuration

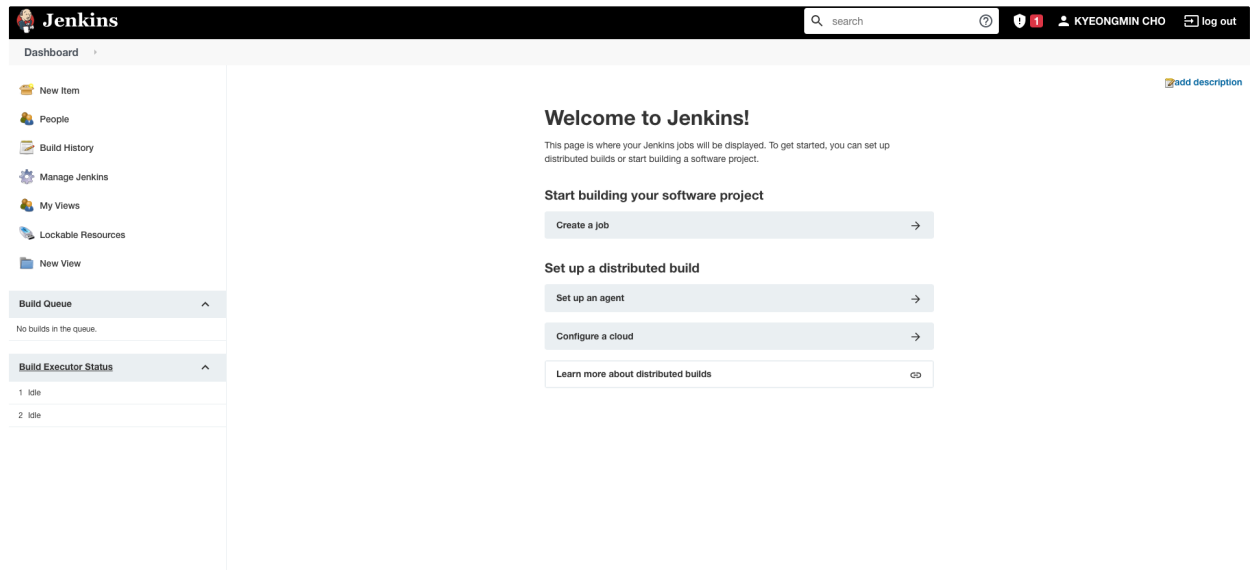
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.303.1 Not now Save and Finish

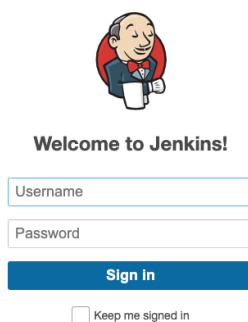
7) 다음의 화면이 나오면 정상적으로 Jenkins 설치가 완료된 것입니다.



8) `ctrl + c` 혹은 `cmd + c` 를 눌러 컨테이너를 빠져나온 후 백그라운드에서 Jenkins가 실행될 수 있도록 다음의 명령어를 실행합니다.

```
sudo docker start jenkinsci
```

9) 브라우저에서 `<VM Instance 공용 IP>:8080` 로 정상적으로 접속이 되는지 확인합니다. 설정했던 계정을 입력하면 7)과 같은 초기화면이 뜹니다.



10) Manage Jenkins - Manage Plugins 페이지로 이동합니다.

Manage Jenkins

Building on the controller node can be a security issue. You should set up distributed builds. See [the documentation](#).

[Set up agent](#)

[Set up cloud](#)

[Dismiss](#)

System Configuration



Configure System
Configure global settings and paths.



Global Tool Configuration
Configure tools, their locations and automatic installers.



Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security



Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.



Manage Credentials
Configure credentials



Configure Credential Providers
Configure the credential providers and types



Manage Users
Create/delete/modify users that can log in to this Jenkins

Status Information



System Information
Displays various environmental information to assist trouble-shooting.



System Log
System log captures output from java.util.logging output related to Jenkins.



Load Statistics
Check your resource utilization and see if you need more computers for your builds.



About Jenkins
See the version and license information.

11) Available 탭을 선택하고 Docker 를 검색한 결과 중, Docker Pipeline 을 체크하고 Install without restart 버튼을 누릅니다.

Updates

Available

Installed

Advanced

Install	Name	Version	Released
<input type="checkbox"/>	Docker Cloud Providers Cluster Management and Distributed Build docker This plugin integrates Jenkins with Docker	1.2.3	8 days 15 hr ago
<input type="checkbox"/>	Docker Commons api-plugin docker Library plugins (for use by other plugins) Provides the common shared functionality for various Docker-related plugins.	1.17	1 yr 1 mo ago
<input type="checkbox"/>	Docker Pipeline Deployment DevOps docker pipeline Build and use Docker containers from pipelines.	1.26	6 mo 4 days ago
<input type="checkbox"/>	Docker API api-plugin docker This plugin provides docker-java API for other plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	3.1.5.2	1 yr 4 mo ago
<input type="checkbox"/>	docker-build-step Build Tools docker This plugin allows to add various docker commands to your job as build steps.	2.8	2 mo 0 days ago
<input type="checkbox"/>	CloudBees Docker Build and Publish Build Tools docker This plugin enables building Dockerfile based projects, as well as publishing of the built images/repos to the docker registry.	1.3.3	6 mo 6 days ago

Amazon ECR

Install without restart

Download now and install after restart

Update information obtained: 8 hr 30 min ago

Check now

Installing Plugins/Upgrades

Preparation

- Checking internet connectivity
- Checking update center connectivity
- Success

Authentication Tokens API Success

Docker Commons Success

Docker Pipeline Success

Loading plugin extensions Running

[Go back to the top page](#)

(you can start using the installed plugins right away)

☐ Restart Jenkins when installation is complete and no jobs are running

2. 컨테이너 SSH Public/Private Key 등록

1) Jenkins가 Github Repository에 접근하기 위해서는 SSH key가 필요합니다. 컨테이너의 SSH key 생성을 위해 다음의 명령어를 실행합니다.

```
sudo docker exec -it jenkinsci ssh-keygen
```

```
ubuntu@docker-instance:~$ sudo docker exec -it jenkinsci ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/jenkins_home/.ssh/id_rsa):
Created directory '/var/jenkins_home/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/jenkins_home/.ssh/id_rsa
Your public key has been saved in /var/jenkins_home/.ssh/id_rsa.pub
The key fingerprint is:
```

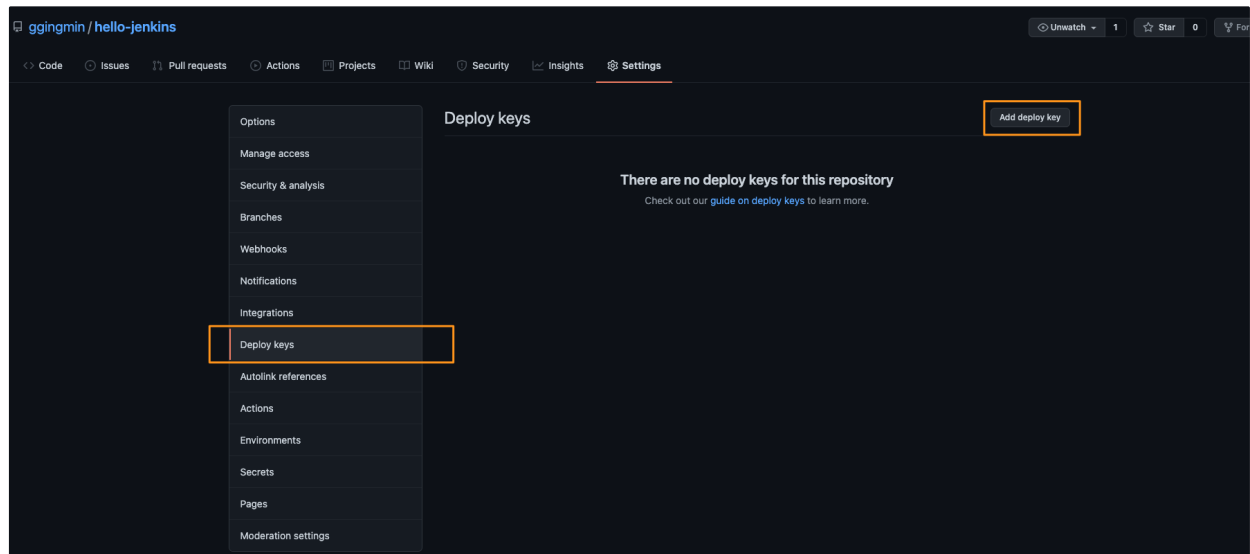
The key's randomart image is:

2) 정상적으로 SSH key가 생성 되었다면 Public Key 값을 조회하기 위해 다음의 명령어를 실행합니다.

```
sudo docker exec -it jenkinsci cat /root/.ssh/id_rsa.pub
```

```
ubuntu@docker-instance:~$ sudo docker exec -it jenkinsci cat /var/jenkins_home/.ssh/id_rsa.pub
ssh-rsa
```

3) 조회된 키를 Github Repository 에 등록하기 위해 **Settings - Deploy keys** 화면에서 **Add deploy key** 버튼을 누릅니다.



4) Title은 식별이 가능하게 작성해주시고 터미널에 출력된 **SSH Public key**를 복사해서 Key 필드에 붙여넣기 합니다.

Deploy keys / Add new

Title

jenkinsci

Key

ssh-rsa

[Redacted SSH Public Key]

☐ **Allow write access**
Can this key be used to push to this repository? Deploy keys always have pull access.

Add key

5) 다음은 SSH Private Key 값을 Jenkins Credential 에 추가해주어야 합니다. 명령어를 실행해서 Private Key 값을 조회합니다.

```
sudo docker exec -it jenkinsci cat /var/jenkins_home/.ssh/id_rsa
```

```
ubuntu@docker-instance:~$ sudo docker exec -it jenkinsci cat /var/jenkins_home/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
```

6) 브라우저의 Jenkins 대시보드에서 **Manage Jenkins** 를 클릭하고 이어 **Manage Credentials** 를 클릭합니다.

Manage Jenkins

Building on the controller node can be a security issue. You should set up distributed builds. See [the documentation](#).

[Set up agent](#) [Set up cloud](#) [Dismiss](#)

System Configuration

- Configure System**
Configure global settings and paths.
- Global Tool Configuration**
Configure tools, their locations and automatic installers.
- Manage Plugins**
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security

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


Status Information

- System Information**
Displays various environmental information to assist trouble-shooting.
- System Log**
System log captures output from java.util.logging output related to Jenkins.
- Load Statistics**
Check your resource utilization and see if you need more computers for your builds.
- About Jenkins**
See the version and license information.

7) Jenkins - Global credentials (unrestricted) - Add Credentials 를 차례대로 클릭합니다.

Credentials


T	P	Store ↓	Domain
Icon: S M L			
<h3>Stores scoped to Jenkins</h3>			
P	Store ↓	Domains	
	Jenkins	(global)	

Domain	Description
 Global credentials (unrestricted)	Credentials that should be available irrespective of domain specification to requirements matching.

Icon: [S](#) [M](#) [L](#)

Dashboard > Credentials > System > Global credentials (unrestricted) >

[Back to credential domains](#)
[Add Credentials](#)

 **Global credentials (unrestricted)**
 Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	De
This credential domain is empty. How about adding some credentials?			

Icon: [S](#) [M](#) [L](#)

8) Kind 는 **SSH Username with private key** 로 설정한 후, ID는 식별을 위한 적절한 값을 입력합니다. 하단 Private Key 항목에 터미널에 출력된 SSH Private Key 를 붙여넣고 OK 를 누릅니다.

Kind
SSH Username with private key

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

ID
jenkinsci-ssh-private-key

Description

Username

☐ Treat username as secret

Private Key
☒ Enter directly
Key

Enter New Secret Below

Passphrase

OK

3. Docker Hub 계정 등록

1) 브라우저의 Jenkins 대시보드에서 **Manage Jenkins** 를 클릭하고 이어 **Manage Credentials** 를 클릭합니다.

Manage Jenkins

Building on the controller node can be a security issue. You should set up distributed builds. See [the documentation](#).

[Set up agent](#)[Set up cloud](#)[Dismiss](#)

System Configuration



Configure System
Configure global settings and paths.



Global Tool Configuration
Configure tools, their locations and automatic installers.



Manage Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.



Manage Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Security



Configure Global Security
Secure Jenkins; define who is allowed to access/use the system.



Manage Credentials
Configure credentials



Configure Credential Providers
Configure the credential providers and types



Manage Users
Create/delete/modify users that can log in to this Jenkins

Status Information



System Information
Displays various environmental information to assist trouble-shooting.



System Log
System log captures output from `java.util.logging` output related to Jenkins.



Load Statistics
Check your resource utilization and see if you need more computers for your builds.



About Jenkins
See the version and license information.

2) Jenkins - Global credentials (unrestricted) - Add Credentials 를 차례대로 클릭합니다.

Dashboard > Credentials > System > Global credentials (unrestricted)

[Back to credential domains](#)

[Add Credentials](#)

Global credentials (unrestricted)

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

ID	Name	Kind	De
This credential domain is empty. How about adding some credentials ?			

Icon: [S](#) [M](#) [L](#)

3) Kind 는 Username with password 로 설정한 후, Username 과 Password는 사용하고 계신 Docker Hub 계정정보를 입력합니다 ID는 인증정보를 식별할 수 있는 값을 넣습니다.

Kind: **Username with password**

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

Username: **ggingmin**

☐ Treat username as secret

Password:

ID: **docker_hub**

Description:

[OK](#)

4) 2개의 credential이 잘 생성되었는지 확인합니다.

Global credentials (unrestricted)

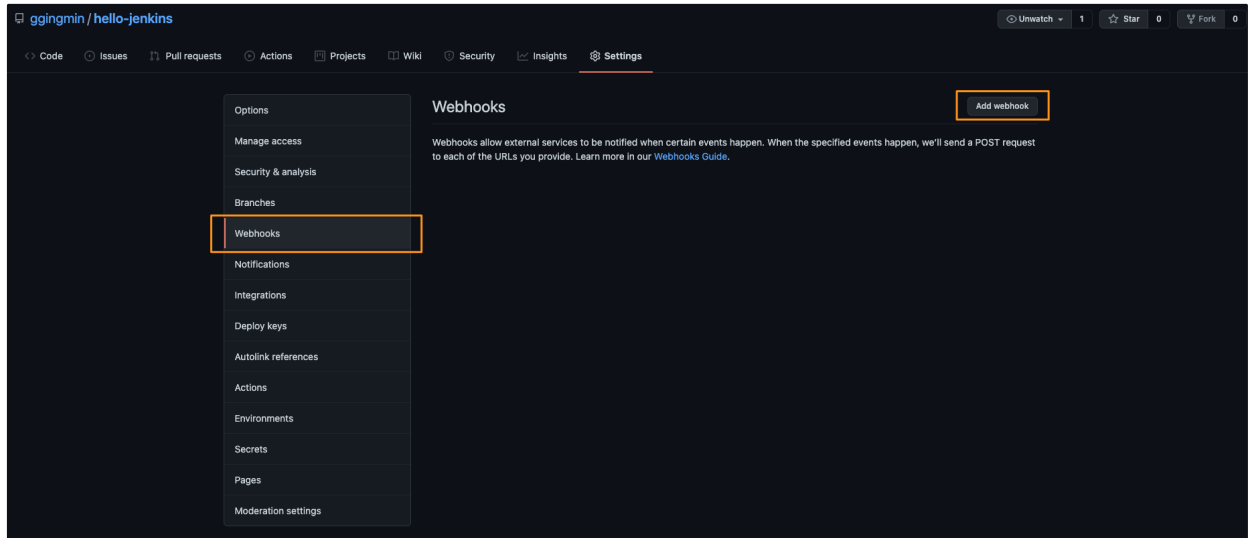
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
jenkinsci-ssh-private-key	jenkins	SSH Username with private key	
docker_hub	ggingmin/*****	Username with password	

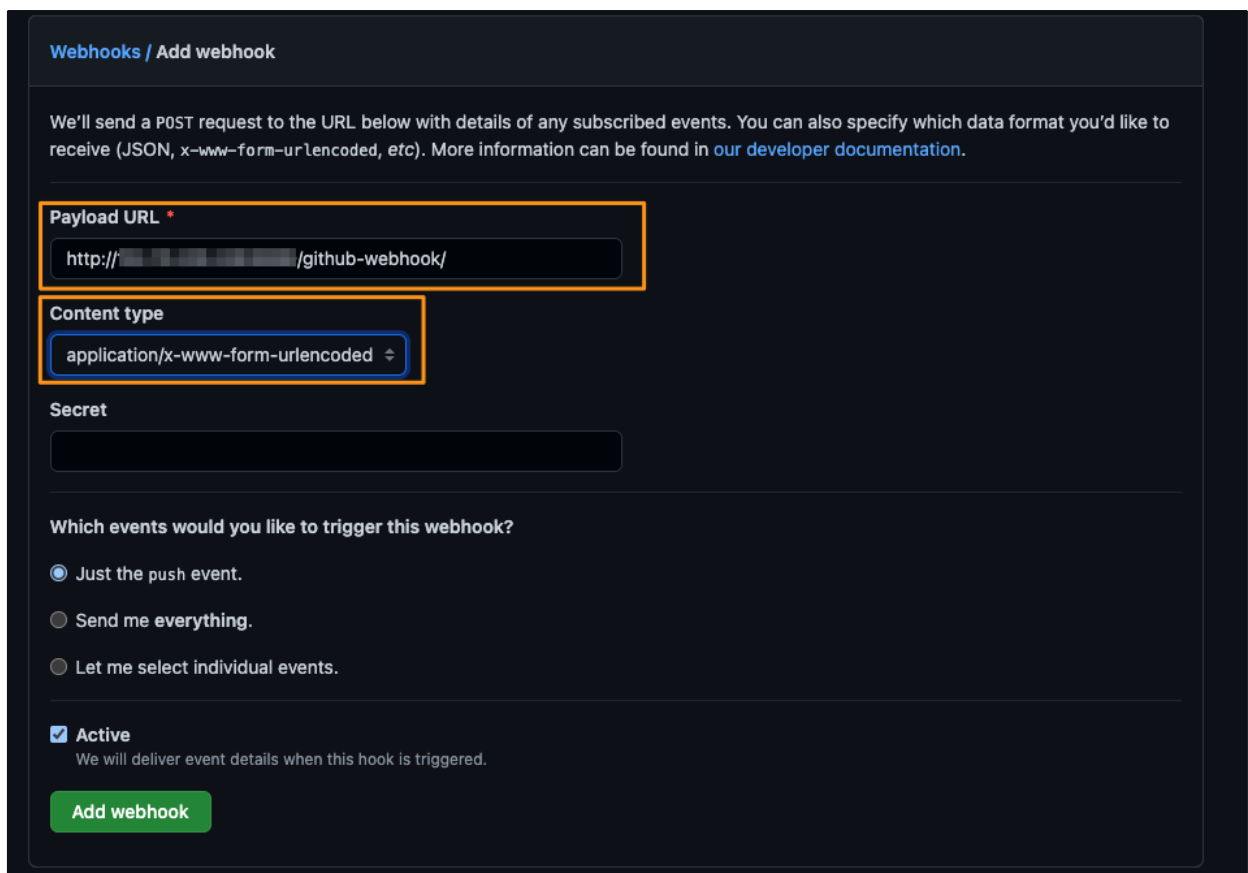
Icon: [S](#) [M](#) [L](#)

4. Github WebHook 등록

1) Settings - Webhooks 화면에서 Add webhook 버튼을 누릅니다.



2) Payload URL에 `http://<인스턴스 공용IP>:8080/github-webhook/` 를 입력하고 Content type 은 `application/x-www-form-urlencoded` 를 선택합니다.



5. Jenkins 파일 생성

1) 이미지를 빌드할 로컬 Repository에서 Jenkinsfile 을 생성한 후 아래와 같이 작성합니다.

```
node { stage('Clone') { checkout scm } stage('Build') { app =
docker.build("ggingmin/hello-jenkins") # 빌드할 이미지명은 자신의 Docker Hub 계정을 넣
어주어야 합니다. } stage('Push') {
docker.withRegistry('https://registry.hub.docker.com', 'docker_hub') {
app.push("${env.BUILD_NUMBER}") app.push("latest") } } }
```

2) Jenkinsfile을 Github Repository 에 push 합니다.

```
git add -A git commit -m "Jenkinsfile added" git push -u origin main
```


The screenshot shows the GitHub interface for the repository 'ggingmin/hello-jenkins'. The 'Code' tab is selected. At the top, there are navigation links: Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these, there are buttons for 'main' (selected), '1 branch', and '0 tags'. On the right, there are buttons for 'Go to file', 'Add file', and 'Code'. The main content area shows a commit by 'ggingmin' titled 'Jenkinsfile added' with commit hash '93d4496' and timestamp '17 seconds ago'. Below the commit title, there is a table listing the files included in the commit:

File	Commit Message	Time
Dockerfile	first commit	1 hour ago
Jenkinsfile	Jenkinsfile added	17 seconds ago
favicon.ico	first commit	1 hour ago
index.html	first commit	1 hour ago

At the bottom, there is a prompt to 'Add a README' with a button labeled 'Add a README'.

6. Jenkins Pipeline 생성

1) New Item 메뉴로 들어가서 item 이름을 작성하고 Pipeline 을 선택한 후 OK 를 누릅니다.



Jenkins

Dashboard

New Item

People

Build History

Manage Jenkins

My Views

Lockable Resources

New View


Build Queue


No builds in the queue.


Enter an item name


docker-pipeline


» Required field



Freestyle project
 This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.


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.


Multi-configuration project
 Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.


Folder
 Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.


GitHub Organization
 Scans a GitHub organization (or user account) for all repositories matching some defined markers.


Multibranch Pipeline
 Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

2) Build Trigger 항목에서 **GitHub hook trigger for GITScm polling** 를 체크합니다.

Build Triggers

☐ Build after other projects are built

☐ Build periodically








☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Disable this project

☐ Quiet period

☐ Trigger builds remotely (e.g., from scripts)

3) Definition 항목에서 **Pipeline script from SCM** 을 선택하고 SCM은 **Git**, Repository URL은 Github 저장소 주소를 넣습니다.

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/ggingmin/hello-jenkins.git

Credentials

- none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

Add Branch

4) 하단의 Branch는 */main으로 입력하고 Script Path가 **Jenkinsfile** 로 되어 있는지 확인하고 Save 버튼을 누릅니다.

Branches to build

Branch Specifier (blank for 'any')

main

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add

Script Path

Jenkinsfile

☒ Lightweight checkout

[Pipeline Syntax](#)

7. Jenkins Pipeline Build

1) 이제 모든 세팅이 완료되었습니다. **Build Now** 를 누르면 Github Repository의 소스를 자동으로 빌드하여 Docker Hub에 공개합니다. 로컬 git repository에서 일부 수정사항을 commit 하고 Github repository에 push 하면 별도로 Build Now 버튼을 누르지 않아도 Pipeline 이 자동으로 수행됩니다.

Back to Dashboard

Back to Dashboard

Status

Changes

Build Now

Configure

Delete Pipeline

Full Stage View

Rename

Pipeline Syntax

GitHub Hook Log

Build History

find

#4 Aug 28, 2021, 2:38 AM

#3 Aug 28, 2021, 2:36 AM

#2 Aug 28, 2021, 2:35 AM

#1 Aug 28, 2021, 2:32 AM

Atom feed for all

Atom feed for failures

Pipeline docker-pipeline

Recent Changes

Stage View

Average stage times:

(Average full run time: ~56s)

Clone

Build

Push

#4

Aug 28 11:38

No Changes

1s

1s

49s

#3

Aug 28 11:36

1 commit

2s

4s

30s

#2

Aug 28 11:35

1 commit

2s

5s

30s

#1

Aug 28 11:32

No Changes

3s

27s

46s

2) 모든 단계가 완료된 후에 Docker Hub 페이지에 접속하면 다음과 같이 빌드된 이미지가 순차적으로 push 된것을 알 수 있습니다.

ggingmin / hello-jenkins

This repository does not have a description

Last pushed: 4 minutes ago

Docker commands

Public View

To push a new tag to this repository,

docker push ggingmin/hello-jenkins:tagname

Tags and Scans

VULNERABILITY SCANNING - DISABLED

Enable

This repository contains 5 tag(s).

TAG

OS

PULLED

PUSHED

latest

6 minutes ago

4 minutes ago

4

6 minutes ago

5 minutes ago

3

6 minutes ago

6 minutes ago

2

7 minutes ago

8 minutes ago

1

10 minutes ago

10 minutes ago

See all

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available on Pro and Team plans.

Upgrade to Pro

Learn more

8. Build 로그 확인

1) Build History에서 로그를 확인할 대상을 클릭합니다.

[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build Now](#)
[Build Now](#) **Configure**
[Delete Pipeline](#)
[Full Stage View](#)
[Rename](#)
[Pipeline Syntax](#)
[GitHub Hook Log](#)

Build History [trend](#) ^
 X
[#4](#) Aug 28, 2021, 2:38 AM
[#3](#) Aug 28, 2021, 2:36 AM
[#2](#) Aug 28, 2021, 2:35 AM
[#1](#) Aug 28, 2021, 2:32 AM
[Atom feed for all](#) [Atom feed for failures](#)

Pipeline docker-pipeline



Stage View

Average stage times:
(Average full run time: ~49s)

	Clone	Build	Push
#4 Aug 28 11:38 No Changes	1s	1s	24s
#3 Aug 28 11:36 1 commit	2s	4s	30s
#2 Aug 28 11:35 1 commit	2s	5s	30s
#1 Aug 28 11:32 No Changes	3s	27s	46s

2) Console Output 을 클릭합니다.

[Back to Project](#)
[Status](#)
[Changes](#)
[Console Output](#)
[Edit Build Information](#)
[Delete build '#4'](#)
[Git Build Data](#)
[Replay](#)
[Pipeline Steps](#)
[Workspaces](#)
[Previous Build](#)

Build #4 (Aug 28, 2021, 2:38:25 AM)



Started by user [KYEONGMIN.CHO](#)



Revision: e7fac042c4eb772ee1a7ceb3d86d69884a18f645
Repository: <https://github.com/ggngmin/hello-jenkins.git>

• refs/remotes/origin/main

3) 다음과 같이 전체 빌드 로그를 확인할 수 있습니다.

Back to Project

Status

Changes

Console Output

View as plain text

Edit Build Information

Delete build '#4'

Git Build Data

Replay

Pipeline Steps

Workspaces

Previous Build

Console Output

```
Started by user KYEONGMIN CHO
Obtained Jenkinsfile from git https://github.com/ggngmin/hello-jenkins.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/jenkins_home/workspace/docker-pipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Clone)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/jenkins_home/workspace/docker-pipeline/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/ggngmin/hello-jenkins.git # timeout=10
Fetching upstream changes from https://github.com/ggngmin/hello-jenkins.git
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git fetch --tags --force --progress -- https://github.com/ggngmin/hello-jenkins.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main`{commit}` # timeout=10
Checking out Revision e7fac042c4eb772eela7ceb3d86d69884a18f645 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f e7fac042c4eb772eela7ceb3d86d69884a18f645 # timeout=10
Commit message: "index.html edited"
> git rev-list --no-walk e7fac042c4eb772eela7ceb3d86d69884a18f645 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] isUnix
[Pipeline] sh
+ docker build -t ggngmin/hello-jenkins .
Sending build context to Docker daemon 75.78kB

-----
cbfd8f461fd0: Layer already exists
45d993692050: Layer already exists
cf38f2c3ba0d: Layer already exists
fc03e3cb8568: Layer already exists
24934e5e6c61: Layer already exists
e2eb06d8af82: Layer already exists
1ea998b95474: Layer already exists
4: digest: sha256:26f1be5134f67ead4b5670505673db1673e39d19501782762112644a958ee43e size: 1984
[Pipeline] isUnix
[Pipeline] sh
+ docker tag ggngmin/hello-jenkins registry.hub.docker.com/ggngmin/hello-jenkins:latest
[Pipeline] isUnix
[Pipeline] sh
+ docker push registry.hub.docker.com/ggngmin/hello-jenkins:latest
The push refers to repository [registry.hub.docker.com/ggngmin/hello-jenkins]
cf38f2c3ba0d: Preparing
cbfd8f461fd0: Preparing
45d993692050: Preparing
1ea998b95474: Preparing
95b99a5c3767: Preparing
fc03e3cb8568: Preparing
24934e5e6c61: Preparing
e2eb06d8af82: Preparing
fc03e3cb8568: Waiting
24934e5e6c61: Waiting
e2eb06d8af82: Waiting
95b99a5c3767: Layer already exists
cbfd8f461fd0: Layer already exists
1ea998b95474: Layer already exists
cf38f2c3ba0d: Layer already exists
24934e5e6c61: Layer already exists
45d993692050: Layer already exists
e2eb06d8af82: Layer already exists
fc03e3cb8568: Layer already exists
latest: digest: sha256:26f1be5134f67ead4b5670505673db1673e39d19501782762112644a958ee43e size: 1984
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```