Jenkins Usage Guide Book

Hyeonil Jeong<dev.nuti0102@gmail.com>

Table of Contents

Chapter 1. Introduction

본 문서는 Continuous Integration(지속적 통합) & Continuous Deliver(지속적 전달)툴인 Jenkins에 관하여 설명하는 목적으로 작성한다.
본 문서는 공식 가이드 문서와 사용자 경험을 바탕으로 작성하고, 초안은 2.46.2 버전을 기준으로 작성한다.

1.1. Continuous Integration

> In software engineering, continuous integration (CI) is the practice of merging all developer working copies to a shared mainline several times a day. Grady Booch first named and proposed CI in his 1991 method, although he did not advocate integrating several times a day. Extreme programming (XP) adopted the concept of CI and did advocate integrating more than once per day - perhaps as many as tens of times per day.
>
> — Wikipedia

지속적 통합(이하 CI)이란 소프트웨어를 개발하는 과정에서 개발자들이 개발한 소스코드 내용을 정기적으로 통합을 하는 것을 의미한다. 소스코드를 지속적으로 통합하고 빌드함으로써 최종 통합 과정에서 발생하는 위험 줄이고, 지속적으로 퀄리티 컨트롤을 적용하는 프로세스를 실행하여 제품의 품질을 높이는 것이 목적이다. 또한, CI는 소프트웨어 개발 과정에서 반복적으로 하는 수동 작업을 자동화 함으로써, 빌드에 소요되는 시간을 줄이는 것을 목적으로 한다.

CI의 이점

- 생산성 향상
- 테스트 자동화에 따른 빠른 버그 검출 및 해결
- 정적 코드 분석에 의한 소스코드 품질 향상
- 빠른 업데이트 제공

1.2. Continuous Delivery

CI가 지속적으로 통합하고, 빌드하는 것을 의미한다면, 지속적 전달(이하 CD)이란 코드 변경에 따라 지속적으로 배포하는 것을 의미하는 것이다. 지속적으로 배포하는 대상은 테스트 환경이 될 수도 있고, 프로덕션 환경이 될 수도 있다. CD를 `Continuous Deployment`(지속적 배포)라고 말하는 이들도 있는데, 지속적 전달과 지속적 배포의 차이는 프로덕션 환경에 배포하는 과정이 수동 배포(지속적 전달)인지, 자동 배포(지속적 배포)인지에 대한 차이가 있다.

CD의 이점

- 생산성 향상
- 테스트 환경 자동 배포에 따른 빠른 버그 검출 및 해결
- 빠른 업데이트 제공
- 소프트웨어 릴리즈 자동화

1.3. Jenkins

Jenkins는 Sun Microsystem에서 Hudson이라는 이름으로 출시된 프로젝트이지만, Sun Microsystem이 Oracle에 인수되면서 Oracle과의 상표권문제로 인해서 이름을 변경하며 Jenkins로 분리되어 나왔다. Jenkins는 Java기반의 오픈소스로 CI와 CD를 가능하게 하는 툴이다. 초기에는 CI툴로 자리를 잡았으나, Blue Ocean이라는 서브 프로젝트를 구축하면서, CD툴로도 자리를 잡아가고 있다.

Jenkins가 제공하는 기능

- 웹 인터페이스를 통한 간편한 설정
- 강력하고 편리한 Reporting 기능
- 지속적인 자동화 빌드
- 지속적인 자동화 테스트
- 커버리지 검사
- 코드 품질 검사
- 다양한 인증기반과 결합한 인증 및 권한 관리 기능
- Groovy script를 이용한 고수준의 Job Scheduling 기능
- 커맨드라인 인터페이스 제공
- 자동화된 배포 관리
- 분산빌드 기능
- 윈도우 커맨드 스케쥴링 실행기능

다양한 테스트 및 빌드 환경을 만들어 놓으면 Jenkins가 작업을 처리해 주기 때문에 Jenkins를 집사라고 표현하기도 한다. 각종 환경을 만드는 것은 사람이 직접해야하고, Jenkins를 효율적으로 활용하기 위해서는 다음과 같은 작업이 함께 이루어져야 한다.

빌드 자동화

빌드 자동화 스크립트를 만들어 놓음으로써 Jenkins에게 빌드를 명령하여 지속적으로 빌드할 수 있도록 한다.

테스트 자동화

단위 테스트, UI 테스트 등 테스트를 자동화해놓음으로써 개인이 미처 발견하지 못한 오류를 검출하고 개인에게 알림이 갈 수 있도록 한다.

코드 표준 준수 여부 검사

정적 코드 분석을 통하여 코드 품질을 높일 수 있도록 한다.

Build Pipeline 구성

2개 이상의 모듈로 구성되는 프로젝트의 경우 각 모듈의 참조관계에 따라서 순차적으로 빌드할 수 있도록 한다.

Chapter 2. Installing Jenkin

본 장에서는 Jenkins의 기본 설치 방법 및 초기 설정방법에 대하여 기술한다.

2.1. Pre-install

2.1.1. System Requirements

최소 요구사항:

- Java 7
- 256MB free memory
- 1GB+ free disk space

소규모 팀 기준 권장 요구사항:

- Java 8
- 1GB+ free memory
- 50GB+ free disk space

2.2. Installation

2.2.1. Unix/Linux

Ubuntu와 같은 Debian 계열에서는 apt를 통하여 설치할 수 있다. 기본 저장소에서 가장 최근의 버전을 설치할 수 있고, LTS버전을 설치하려면 별도의 저장소 설정을 해주어야 한다.

```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key
add -
sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins
```

/etc/default/jenkins 파일을 통해 기본 설정을 수정할 수 있고, 포트 중복 방지를 위해서 기본 포트를 바꿔주는 것을 권장한다.

2.2.2. OS X

- jenkins 공식 홈페이지에서 패키지를 다운받는다.
- 패키지를 실행한다.

brew를 통해서도 설치를 할 수 있다.

- 최신 버전 설치

```
brew install jenkins
```

- LTS 버전 설치

```
brew install jenkins-lts
```

2.2.3. Windows

- jenkins 공식 홈페이지에서 패키지를 다운받는다.
- 패키지를 실행한다.

2.2.4. Docker

- Docker 저장소에서 jenkins 이미지를 pull 한다.

```
docker pull jenkins
```

- Docker 컨테이너를 실행한다.

```
docker run -d -p 9000:8080 -v $PWD/jenkins:/var/jenkins_home -t jenkins
```

- 위와 같이 실행할 시 로컬호스트의 9000포트가 컨테이너의 8080포트와 매핑이 된다.
- `-d` 옵션은 데몬 실행 옵션이니, 데몬으로 실행하기를 원하지 않을 경우 제외하면 된다.

2.2.5. Other

war로 패키지된 파일을 다운받아서 `Tomcat`이나 `Jetty`와 같은 컨테이너로 실행할 수 있다.

2.3. Post-installation(Setup Wizard)

2.3.1. Create Admin User and Password for Jenkins

최초로 Jenkins를 실행하고 브라우저를 통해 페이지 진입 시, 관리자 인증을 요구한다. 이 때 필요한 토큰은 Jenkins의 로그에 남아있으니, 확인후 입력해주면 된다.

```
*************************************************************

Jenkins initial setup is required. A security token is required to
proceed.
Please use the following security token to proceed to installation:

41d2b60b0e4cb5bf2025d33b21cb

*************************************************************
```

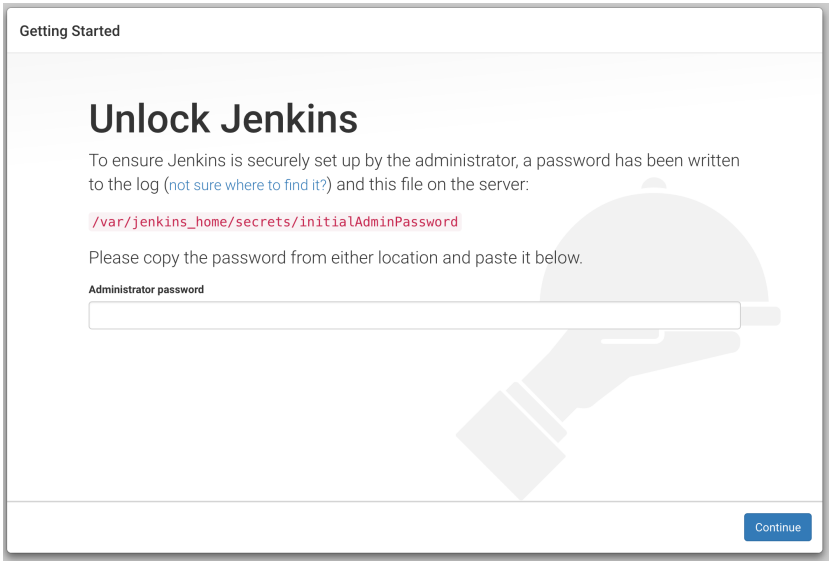

Figure 1. 관리자 비밀번호

2.3.2. Initial Plugin Installation

관리자 인증을 마치면 초기 Plugin 설치를 물어보는 페이지가 나타나는데, 처음 사용한다면 추천 플러그인을 설치할 것을 권장한다.

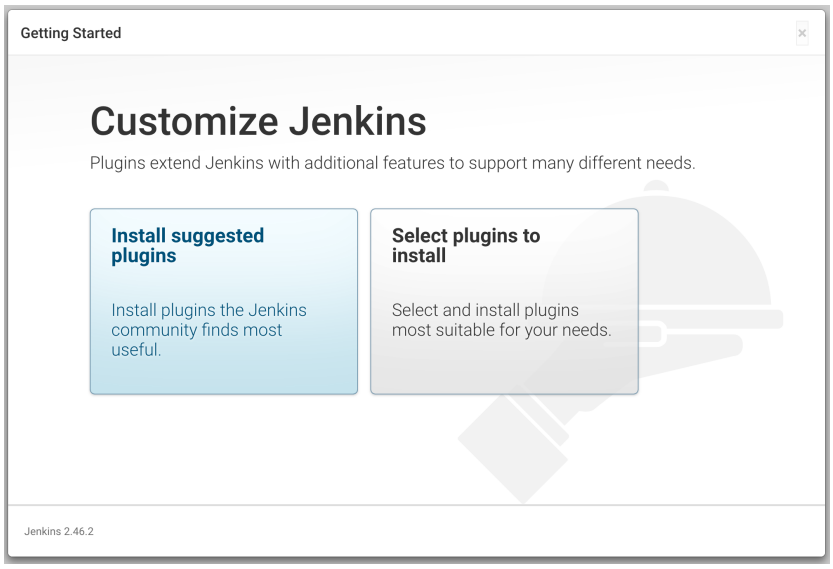

Figure 2. 초기 플러그인 설치

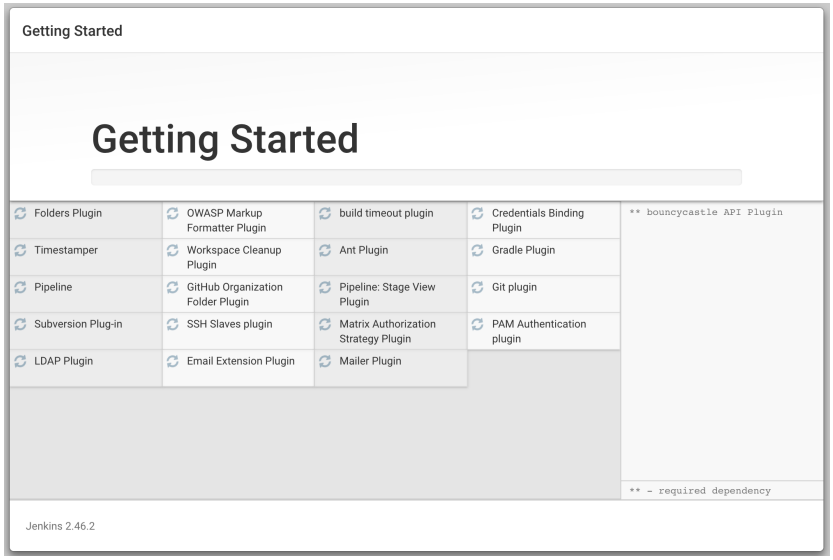

Figure 3. 추천 플러그인 설치

Chapter 3. System Configuration

본 장에서는 Jenkins 시스템 설정하는 방법에 대하여 기술한다.

Jenkins를 설치하면 동시 실행 가능 executor의 수, 메일 서버, VCS 등 여러가지 정보를 설정할 수 있고, 플러그인 설치에 따라 더욱 다양한 정보를 설정할 수 있다. 각자 확인 후 필요한 정보들을 입력해주면 된다.

Chapter 4. Managing Security

본 장에서는 Jenkins 보안 설정하는 방법에 대하여 기술한다.

Jenkins는 웹 환경에서 동작하기 때문에 보안과 관련된 다양한 설정을 제공한다. Jenkins를 안전하게 사용하기 위해서 보안 설정을 필수로 해줄 것을 권장한다.

4.1. Enabling Security

Enable Security설정에 따라 보안 사용 여부를 지정할 수 있다. 보안 설정은 반드시 활성화 할 것을 권장한다.

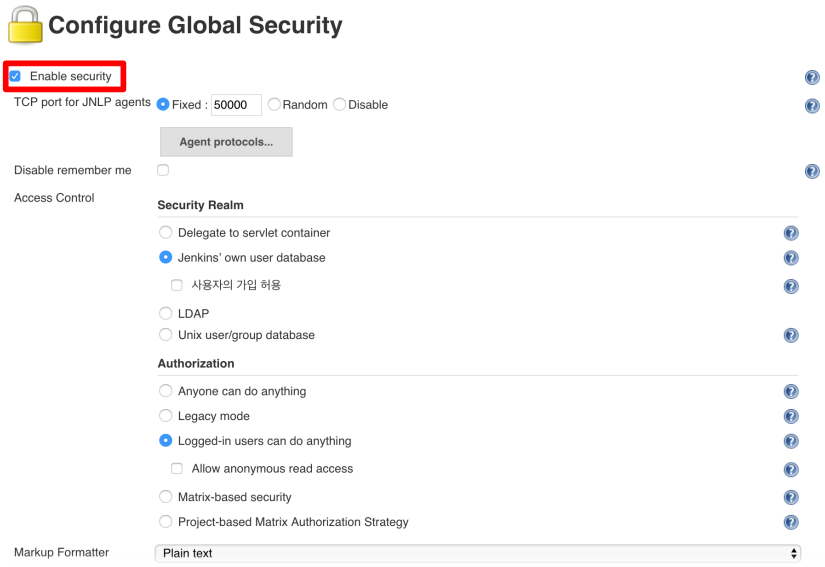

Figure 4. 보안 설정

4.1.1. JNLP TCP Port

Jenkins는 JNLP 프로토콜을 통해 Agent를 실행할 수 있도록 TCP 포트를 사용한다.

JNLP 기반 Agent를 위해 다음과 같은 두 가지 옵션을 제공한다.

Random
: JNLP 포트를 랜덤으로 선택한다.

Fixed
: JNLP 포트를 관리자가 임의로 지정한다.

4.1.2. Access Control

Access Control은 Jenkins의 주된 보안 메카니즘이다. Access Control설정을 위해 주된 두가지 설이 필요하다.

1. **Security Realm**은 사용자 정보를 가져을 방법과 위치를 알려주는 보안영역이다. 일반적으로 "인증"이라고 한다.
2. **Authorization**은 사용자 혹은 그룹이 어던 측면까지 액세스할 수 있는지 설정하는 영역이다. 일반적으로 "권한"이라고 한다.

Security Realm과 Authorization 설정에 따라 완화된 혹은 엄격한 인증 및 권한 부여를 구성할 수 있다.

또한, Role-based Authorization Strategy와 같은 플러그인을 통해 Access Control의 기능을 확장할 수 있다.

Security Realm

Jenkins는 기본적으로 몇가지 Security Realm을 제공한다.

Delegate to servlet container
: 인증 절차를 Jenkins를 실행하고있는 Servlet Container에 위임한다. 이 옵션을 사용한다면, Servlet Container의 인증 문서를 확인하길 바란다.

Jenkins' own user database
: 인증 절차를 다른 외부 시스템에 위임하지 않고 Jenkins 소유의 사용자 데이터베이스를 사용한다. 이 설정은 2.0부터 지원을 하고, 소규모 환경에 적합하다.

LDAP
: 모든 인증 과정을 LDAP 서버에 위임한다. 이 옵션은 이미 LDAP 환경이 구축된 대규모 조직에 적합하다.

LDAP 옵션이 제공되지 않는다면, plugin:ldap[LDAP plugin] 플러그인을 설치해서 사용할 수 있다.

Unix user/group database
: 인증 절차를 Unix 운영체제에 위임한다. 이 옵션은 Unix 그룹과 사용자를 재사용할 수 있다.

다음과 같은 플러그인을 통해 추가적인 Security Realm을 사용할 수 있다

- plugin:active-directory[Active Directory]
- plugin:github-oauth[GitHub Authentication]
- plugin:crowd2[Atlassian Crowd 2]

Authorization

Security Realm은 누가 Jenkins에 액세스 가능한지를 정한다. 그와 다르게 Authorization은 누가 무엇을 액세스할 수 있는지를 정한다. Jenkins는 기본적으로 몇가지 Authorization을 제공한다.

Anyone can do anything
: 모든 사용자가 Jenkins의 모든 기능을 활용할 수 있다. 로컬 테스트 용도가 아니라면 이 설정을 **사용하지 않는 것을 권장**한다.

Legacy mode
: 사용자가 "admin"일 경우 시스템 전체의 제어 권한을 갖고, 익명 사용자를 포함한 기타 사용자는 읽기 권한만 갖는다. 로컬 테스트 용도가 아니라면 이 설정을 **사용하지 않는 것을 권장**한다.

Logged in users can do anything
: 인증된 사용자가 시스템 전체 제어 권한을 갖는다. 추가 옵션으로 익명 사용자에게 읽기 권한을 부여할 지 여부를 정할 수 있다. 인증 과정을 강제로 지정할 때 유용하다.

Matrix-based security
: 특정 사용자에게 특정 제어 권한을 부여할 수 있다.

Project-based Matrix Authorization Strategy
: Matrix-based security의 확장으로 각각의 프로젝트에서 특정 사용자에게 특정 제어 권한을 부여한다.

4.1.3. Markup Formatter

Jenkins는 사용자 입력으로 HTML이나 Javascript를 입력하는 것을 허용한다. 기본 설정으로는 < 및 &와 같은 안전하지 않은 문자를 이스케이프처리하는 **Plain Text** 설정을 사용한다. **Safe HTML** 설정을 사용하면 사용자와 관리자가

프로젝트 설명과 같은 곳에 HTML을 사용할 수 있다.

4.2. Cross Site Request Forgery

CSRF 보호 설정 사용 여부를 지정할 수 있다.

4.3. Agent/Master Access Control

Conceptually, the Jenkins master and agents can be thought of as a cohesive system which happens to execute across multiple discrete processes and machines. This allows an agent to ask the master process for information available to it, for example, the contents of files, etc. For larger or mature Jenkins environments where a Jenkins administrator might enable agents provided by other teams or organizations, a flat agent/master trust model is insufficient. The Agent/Master Access Control system was introduced [2: Starting with 1.587, and 1.580.1, releases] to allow Jenkins administrators to add more granular access control definitions between the Jenkins master and the connected agents. As of Jenkins 2.0, this subsystem has been turned on by default.

4.3.1. Customizing Access

For advanced users who may wish to allow certain access patterns from the agents to the Jenkins master, Jenkins allows administrators to create specific exemptions from the built-in access control rules. By following the link highlighted above, an administrator may edit Commands and File Access Agent/Master access control rules.

Commands

"Commands" in Jenkins and its plugins are identified by their fully-qualified class names. The majority of these commands are intended to be executed on agents by a request of a master, but some of them are intended to be executed on a master by a request of an agent. Plugins not yet updated for this subsystem may not classify which category each command falls into, such that when an agent requests that the master execute a command which is not explicitly allowed, Jenkins will err on the side of caution and refuse to execute the command. In such cases, Jenkins administrators may "whitelist" [3: en.wikipedia.org/wiki/Whitelist] certain commands as acceptable for execution on the master. 20

Advanced

Administrators may also whitelist classes by creating files with the .conf extension in the directory JENKINS_HOME/secrets/whitelisted-callables.d/. The contents of these .conf files should list command names on separate lines. The contents of all the .conf files in the directory will be read by Jenkins and combined to create a default.conf file in the directory which lists all known safe command. The default.conf file will be re-written each time Jenkins boots. Jenkins also manages a file named gui.conf, in the whitelisted-callables.d directory, where commands added via the web UI are written. In order to disable the ability of administrators to change whitelisted commands from the web UI, place an empty gui.conf file in the directory and change its permissions such that is not writeable by the operating system user Jenkins run as.

File Access Rules

The File Access Rules are used to validate file access requests made from agents to the master. Each File Access Rule is a triplet which must contain each of the following elements: 1. allow / deny: if the following two parameters match the current request being considered, an allow entry would allow the request to be carried out and a deny entry would deny the request to be rejected, regardless of what later rules might say. 2. operation: Type of the operation requested. The following 6 values exist. The operations can also be combined by comma-separating the values. The value of all indicates all the listed operations are allowed or denied. ◦ read: read file content or list directory entries ◦ write: write file content ◦ mkdirs: create a new directory ◦ create: create a file in an existing directory ◦ delete: delete a file or directory

◦ stat: read metadata of a file/directory, such as timestamp, length, file access modes. 3. file path: regular expression that specifies file paths that matches this rule. In addition to the base regexp syntax, it supports the following tokens: ◦ <JENKINS_HOME> can be used as a prefix to match the master's JENKINS_HOME directory. ◦ <BUILDDIR> can be used as a prefix to match the build record directory, such as /var/lib/jenkins/job/foo/builds/2014-10-17_12-34-56. ◦ <BUILDID> matches the timestamp-formatted build IDs, like 2014-10-17_12-34-56. The rules are ordered, and applied in that order. The earliest match wins. For example, the following rules allow access to all files in JENKINS_HOME except the secrets

folders: # To avoid hassle of escaping every '\' on Windows, you can use / even on Windows. deny all <JENKINS_HOME>/secrets/.* allow all <JENKINS_HOME>/.* Ordering is very important! The following rules are incorrectly written because the 2nd rule will never match, and allow all agents to access all files and folders under JENKINS_HOME: allow all <JENKINS_HOME>/.* deny all <JENKINS_HOME>/secrets/.*

Advanced

Administrators may also add File Access Rules by creating files with the .conf. extension in the directory JENKINS_HOME/secrets/filepath-filters.d/. Jenkins itself generates the 30-default.conf file on boot in this directory which contains defaults considered the best balance between compatibility and security by the Jenkins project. In order to disable these built-in defaults, replace 30- default.conf with an empty file which is not writable by the operating system user Jenkins run as. On each boot, Jenkins will read all .conf files in the filepath-filters.d directory in alphabetical order, therefore it is good practice to name files in a manner which indicates their load order. Jenkins also manages 50-gui.conf, in the filepath-filters/ directory, where File Access Rules added via the web UI are written. In order to disable the ability of administrators to change the File Access Rules from the web UI, place an empty 50-gui.conf file in the directory and change its permissions such that is not writeable by the operating system user Jenkins run as.

4.3.2. Disabling

While it is not recommended, if all agents in a Jenkins environment can be considered "trusted" to the same degree that the master is trusted, the Agent/Master Access Control feature may be disabled. Additionally, all the users in the Jenkins environment should have the same level of access to all configured projects. 22 An administrator can disable Agent/Master Access Control in the web UI by un-checking the box on the Configure Global Security page. Alternatively an administrator may create a file in JENKINS_HOME/secrets named slave-to-master-security-kill-switch with the contents of true and restart Jenkins. Most Jenkins environments grow over time requiring their trust models to CAUTION evolve as the environment grows. Please consider scheduling regular "check- ups" to review whether any disabled security settings should be re-enabled.

Chapter 5. Managing Users

Chapter 6. Managing Plugins

Chapter 7. Managing Nodes

Chapter 8. Freestyle Project

Chapter 9. Pipeline Project

Chapter 10. Multi-branch Pipeline Project

Chapter 11. Blue Ocean

Chapter 12. Pipeline Syntax

Chapter 13. Jenkins with Java

Chapter 14. Jenkins with PHP

Chapter 15. Jenkins with Javascript

Chapter 16. Jenkins with .NET

Chapter 17. Conclusions

Chapter 18. Terms

Chapter 19. References

- http://www.moreagile.net/2014/01/jenkins-cicontinuous-integration-2.html
- https://jenkins.io/doc/