Experiment 05 - Jenkins Setup

Roll No.	Sreekesh Iyer
Name	24
Class	D15-A
Subject	DevOps Lab
LO Mapped	LO1: To understand the fundamentals of DevOps engineering and be fully proficient with DevOps terminologies, concepts, benefits, and deployment options to meet your business requirements LO3: To understand the importance of Jenkins to Build and deploy Software Applications on server environment

Aim: To understand Continuous Integration, install and configure Jenkins with Maven/Ant/Gradle to set up a build job.

Introduction:

Continuous Integration

Continuous integration (CI) is a development practice where development teams make small, frequent changes to code. An automated build verifies the code each time developers check their changes into the version control repository. As a result, development teams can detect problems early.

Continuous integration is the first part of CI/CD, a practice that enables application development teams to release incremental code changes to production quickly and regularly.

Importance of Continuous Integration

Reduced integration risk
More often than not, working on projects means multiple people are working on separate tasks or parts of the code. The more people, the riskier the integration. Depending on how bad the problem really is, debugging and solving the issue can be really painful and can potentially mean a lot of changes to the code. Integrating on a daily basis or even more frequently can help reduce these kinds of problems to a minimum.

Higher code quality
Not having to worry about the problems, and focusing more on the functionality of the code results in a higher quality product.

The code in version control works
If you commit something that breaks the build, you and your team get the notice immediately and the problem is fixed before anyone else pulls the "broken" code.

Reduced friction between team members
Having an impartial system in place reduces the frequency of quarrels between team members.

The quality of life improvement for testers
Having different versions and builds of the code can help isolate and trace bugs efficiently, and it makes life easier for the QA team.

Less time deploying
Deploying projects can be very tedious and time-consuming, and automating that process makes perfect sense.

Increased confidence and morale
People that don't work for fear of breaking something, are more likely to produce better results and can focus their energy and concentration on producing instead of worrying about the potential consequences of their actions.

Jenkins:

Jenkins is an open-source server that is written entirely in Java. It lets you execute a series of actions to achieve the continuous integration process, that too in an automated fashion.

This CI server runs in servlet containers such as Apache Tomcat. Jenkins facilitates continuous integration and continuous delivery in software projects by automating parts related to build, test, and deployment. This makes it easy for developers to continuously work on the betterment of the product by integrating changes to the project.

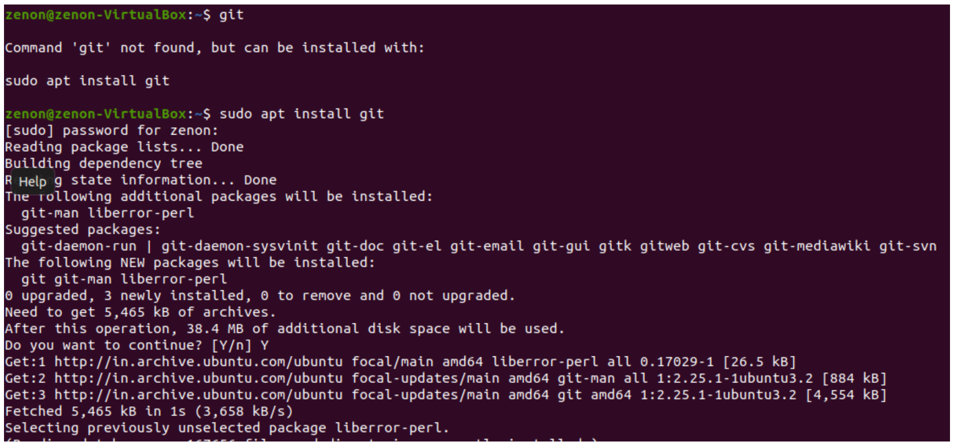
Jenkins automates the software builds in a continuous manner and lets the developers know about the errors at an early stage. A strong Jenkins community is one of the prime reasons for its popularity. Jenkins is not only extensible but also has a thriving plugin ecosystem.

Installation:

Before installing Jenkins, we need to install a few dependencies/prerequisites for it. We'd need Java, Maven and optionally, git.

1. First, let's install Git.

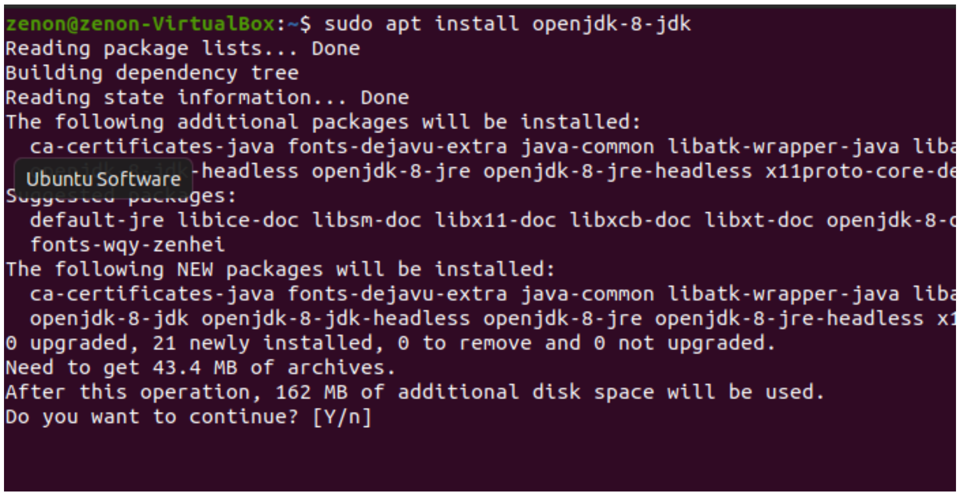
```
sudo apt install git
```



```
zenon@zenon-VirtualBox:~$ git

Command 'git' not found, but can be installed with:

sudo apt install git

zenon@zenon-VirtualBox:~$ sudo apt install git
[sudo] password for zenon:
Reading package lists... Done
Building dependency tree
R[illegible]g state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 5,465 kB of archives.
After this operation, 38.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://in.archive.ubuntu.com/ubuntu focal/main amd64 liberror-perl all 0.17029-1 [26.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 git-man all 1:2.25.1-1ubuntu3.2 [884 kB]
Get:3 http://in.archive.ubuntu.com/ubuntu focal-updates/main amd64 git amd64 1:2.25.1-1ubuntu3.2 [4,554 kB]
Fetched 5,465 kB in 1s (3,658 kB/s)
Selecting previously unselected package liberror-perl.
```

2. Install Java

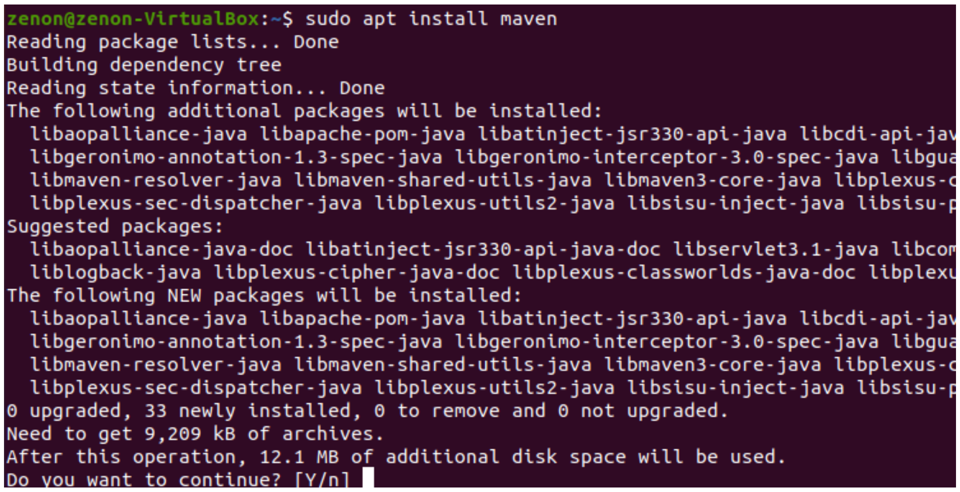
```
sudo apt install openjdk-8-jdk
```



```
zenon@zenon-VirtualBox:~$ sudo apt install openjdk-8-jdk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java liba
  [illegible]-headless openjdk-8-jre openjdk-8-jre-headless x11proto-core-de
S[illegible]ges:
  default-jre libice-doc libsm-doc libx11-doc libxcb-doc libxt-doc openjdk-8-
  fonts-wqy-zenhei
The following NEW packages will be installed:
  ca-certificates-java fonts-dejavu-extra java-common libatk-wrapper-java liba
  openjdk-8-jdk openjdk-8-jdk-headless openjdk-8-jre openjdk-8-jre-headless x1
0 upgraded, 21 newly installed, 0 to remove and 0 not upgraded.
Need to get 43.4 MB of archives.
After this operation, 162 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

3. Install Maven

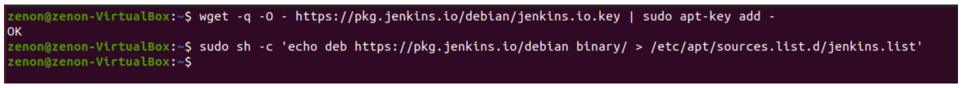
```
sudo apt install maven
```



```
zenon@zenon-VirtualBox:~$ sudo apt install maven
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-jav
  libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libgua
  libmaven-resolver-java libmaven-shared-utils-java libmaven3-core-java libplexus-c
  libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java libsisu-p
Suggested packages:
  libaopalliance-java-doc libatinject-jsr330-api-java-doc libservlet3.1-java libcom
  liblogback-java libplexus-cipher-java-doc libplexus-classworlds-java-doc libplexu
The following NEW packages will be installed:
  libaopalliance-java libapache-pom-java libatinject-jsr330-api-java libcdi-api-jav
  libgeronimo-annotation-1.3-spec-java libgeronimo-interceptor-3.0-spec-java libgua
  libmaven-resolver-java libmaven-shared-utils-java libmaven3-core-java libplexus-c
  libplexus-sec-dispatcher-java libplexus-utils2-java libsisu-inject-java libsisu-p
0 upgraded, 33 newly installed, 0 to remove and 0 not upgraded.
Need to get 9,209 kB of archives.
After this operation, 12.1 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

4. Add keys and Jenkins repository entries to your system to automate installation.

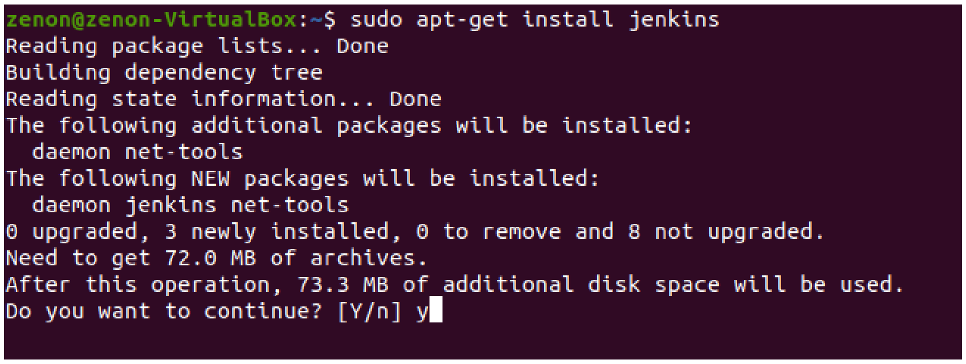
```
wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo
apt-key add -

sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ >
/etc/apt/sources.list.d/jenkins.list'
```



```
zenon@zenon-VirtualBox:~$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
OK
zenon@zenon-VirtualBox:~$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian binary/ > /etc/apt/sources.list.d/jenkins.list'
zenon@zenon-VirtualBox:~$
```

5. Update the local package index and install Jenkins.

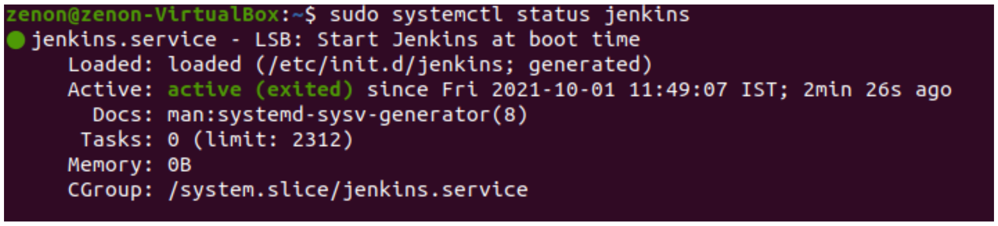
```
sudo apt-get update
sudo apt-get install jenkins
```



```
zenon@zenon-VirtualBox:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemon net-tools
The following NEW packages will be installed:
  daemon jenkins net-tools
0 upgraded, 3 newly installed, 0 to remove and 8 not upgraded.
Need to get 72.0 MB of archives.
After this operation, 73.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

6. Once done, Jenkins will automatically start on localhost, port 8080. Check Jenkins Status.

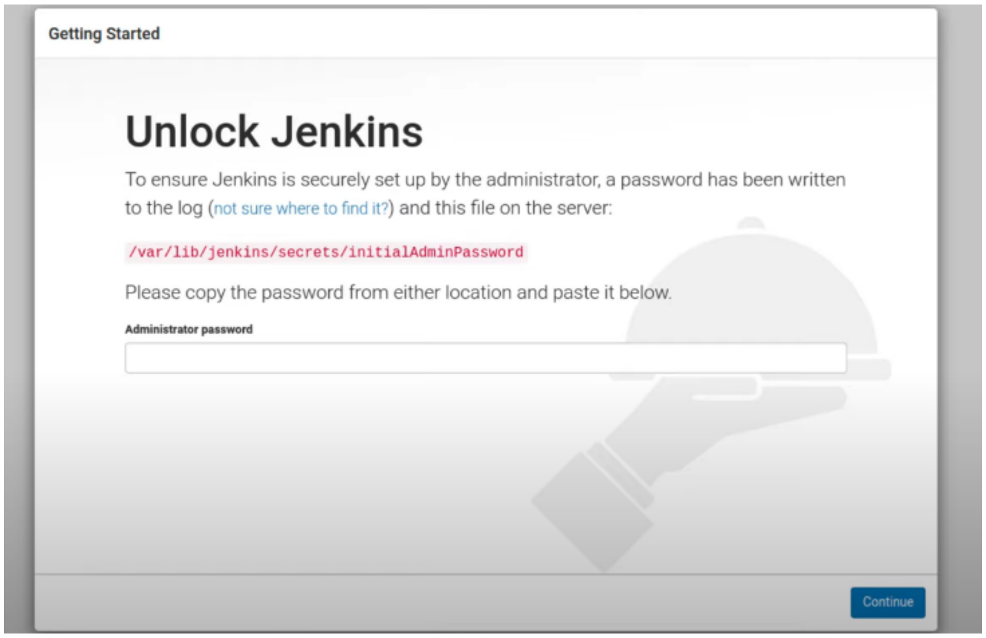
```
sudo systemctl status jenkins
```



```
zenon@zenon-VirtualBox:~$ sudo systemctl status jenkins
● jenkins.service - LSB: Start Jenkins at boot time
     Loaded: loaded (/etc/init.d/jenkins; generated)
     Active: active (exited) since Fri 2021-10-01 11:49:07 IST; 2min 26s ago
       Docs: man:systemd-sysv-generator(8)
      Tasks: 0 (limit: 2312)
     Memory: 0B
     CGroup: /system.slice/jenkins.service
```

If you can see Jenkins active, it means that you've successfully installed it.

7. Change initial Admin password



Getting Started

Unlock Jenkins

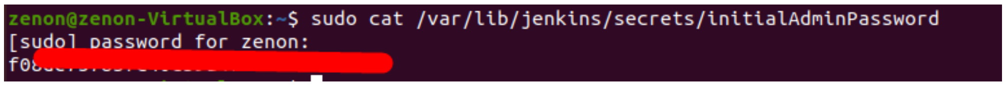
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

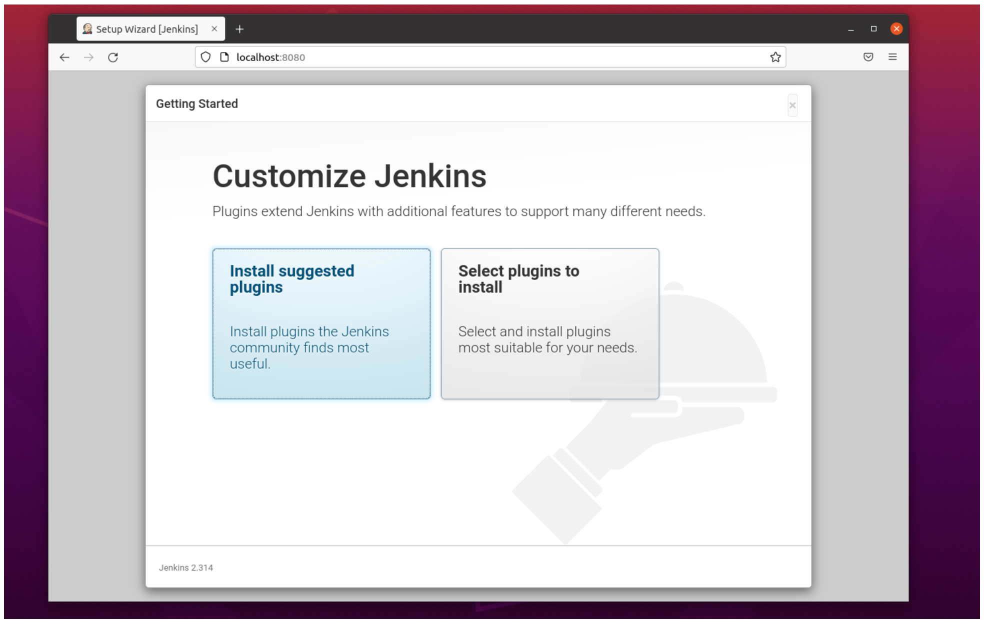
```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



```
zenon@zenon-VirtualBox:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
[sudo] password for zenon:
f0[redacted]
```

Use this password to get the first entry.

8. Choose to install suggested plugins.



Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

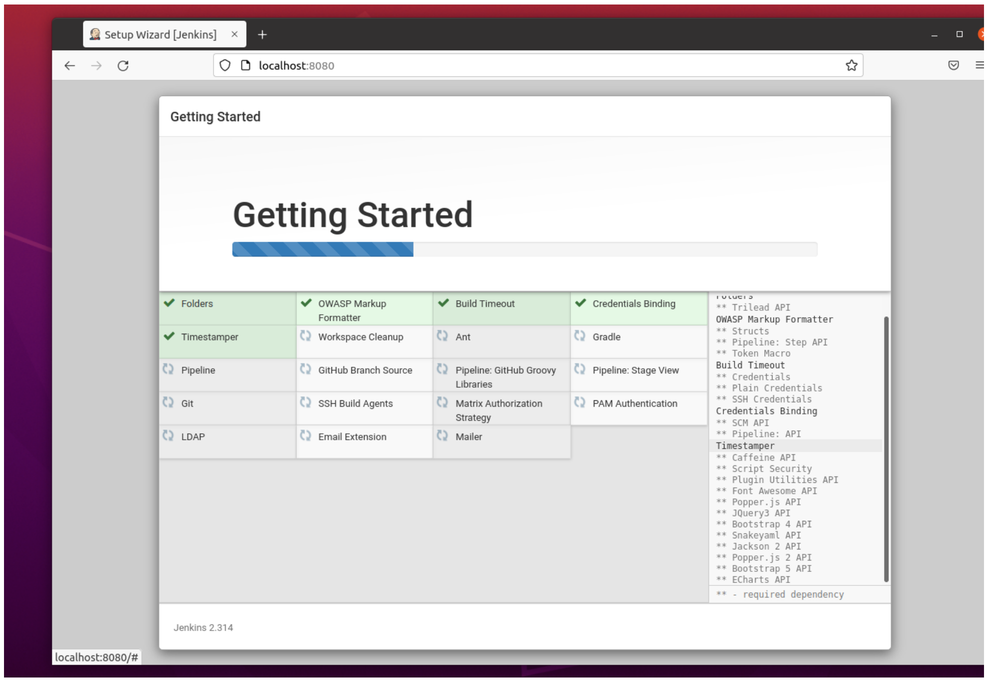
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.314

9. Wait until the installation is complete.



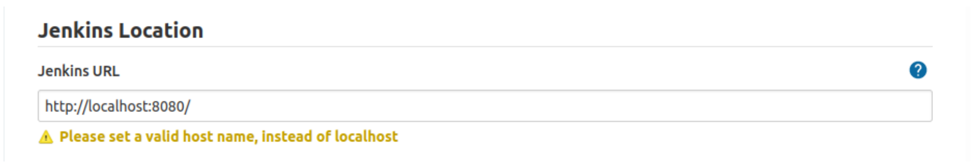
Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	

```
Folders
** Trilead API
OWASP Markup Formatter
** Structs
** Pipeline: Step API
** Token Macro
Build Timeout
** Credentials
** Plain Credentials
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
Timestamper
** Caffeine API
** Script Security
** Plugin Utilities API
** Font Awesome API
** Popper.js API
** JQuery3 API
** Bootstrap 4 API
** Snakeyaml API
** Jackson 2 API
** Popper.js 2 API
** Bootstrap 5 API
** ECharts API
** - required dependency
```

Jenkins 2.314

10. Proceed with the default Jenkins URL or change if required.



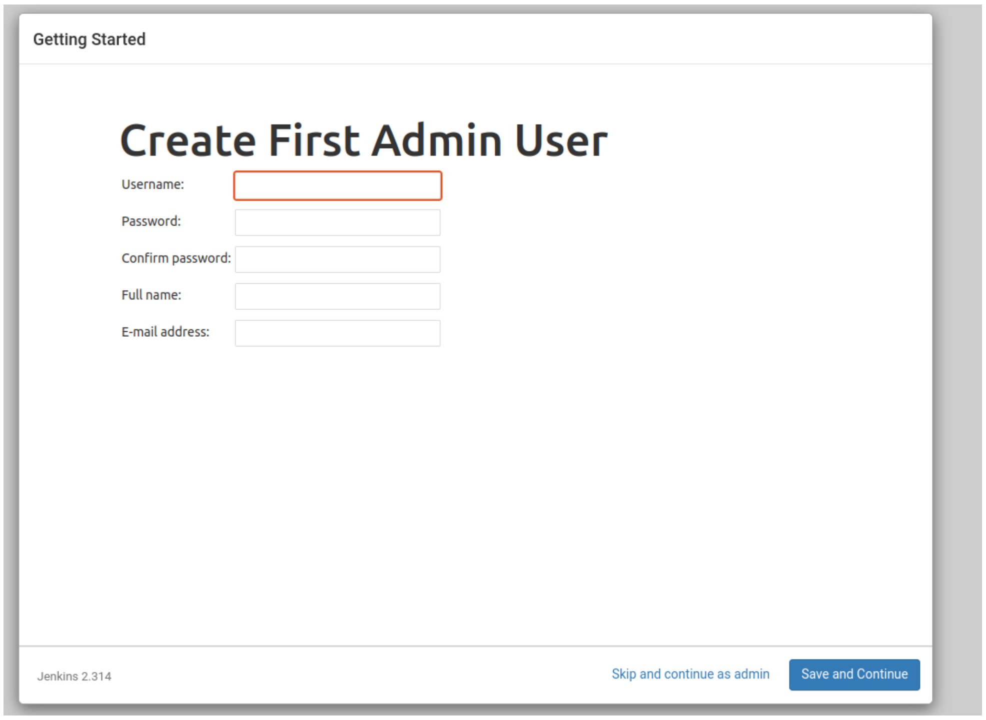
Jenkins Location

Jenkins URL

http://localhost:8080/

⚠ Please set a valid host name, instead of localhost

11. Create first admin user.



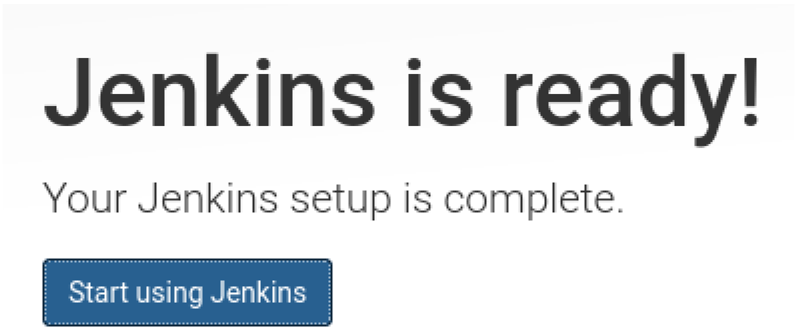
Getting Started

Create First Admin User

Username:	
Password:	
Confirm password:	
Full name:	
E-mail address:	

Jenkins 2.314

12. Jenkins installation is complete. Click on Start to go to the Jenkins console.

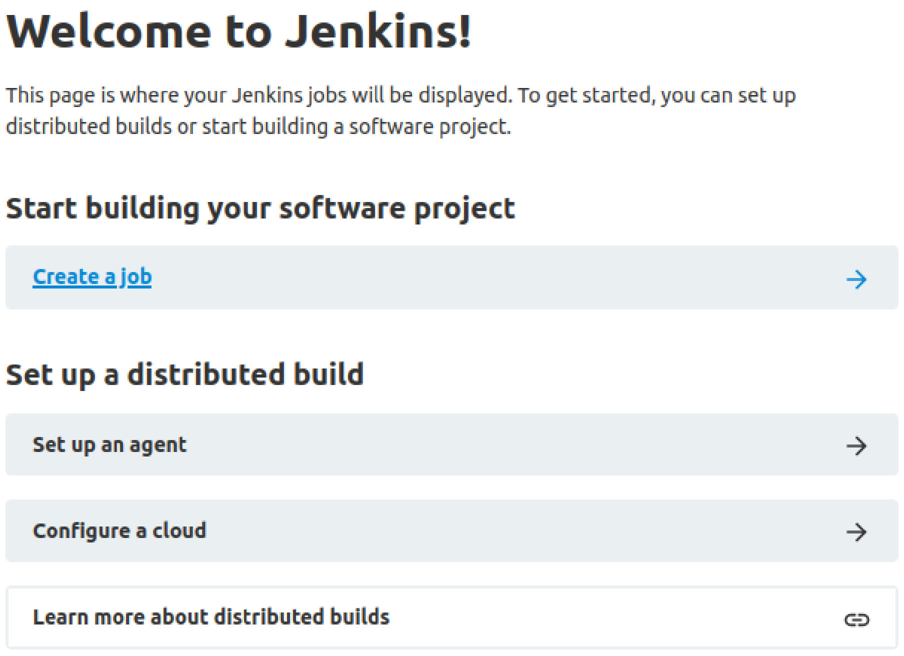


Jenkins is ready!

Your Jenkins setup is complete.

Build Job:

1. In the Jenkins console, click on Create a Job to start.



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

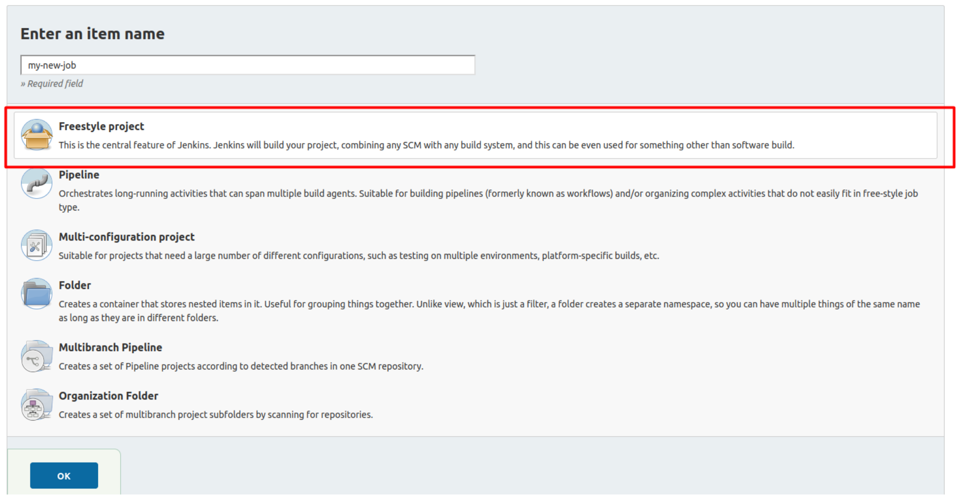
Set up a distributed build

Set up an agent →

Configure a cloud →

Learn more about distributed builds

2. Enter Item name and choose freestyle project.



Enter an item name

my-new-job

» 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.

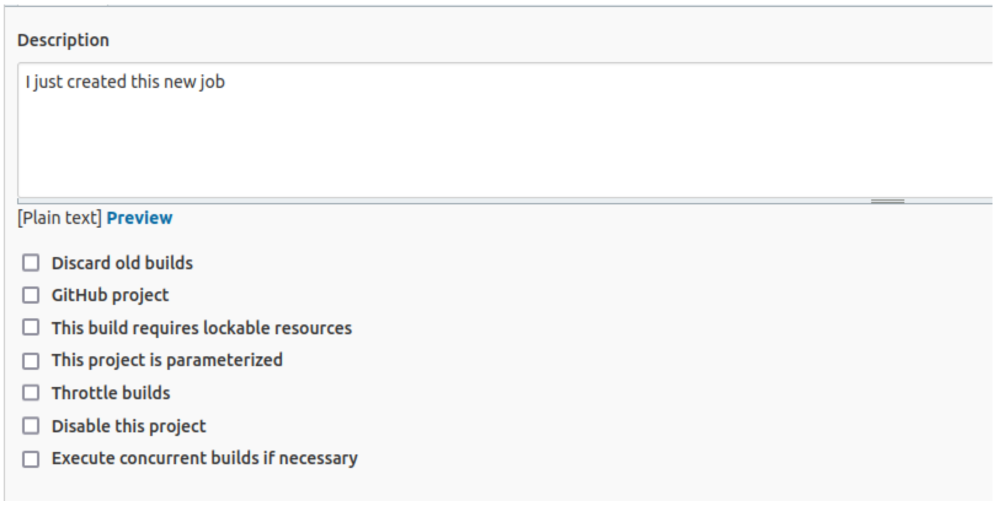
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

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

Organization Folder
Creates a set of multibranch project subfolders by scanning for repositories.

OK

3. Write a description under the general section. Optionally, add a Git Repository if needed.



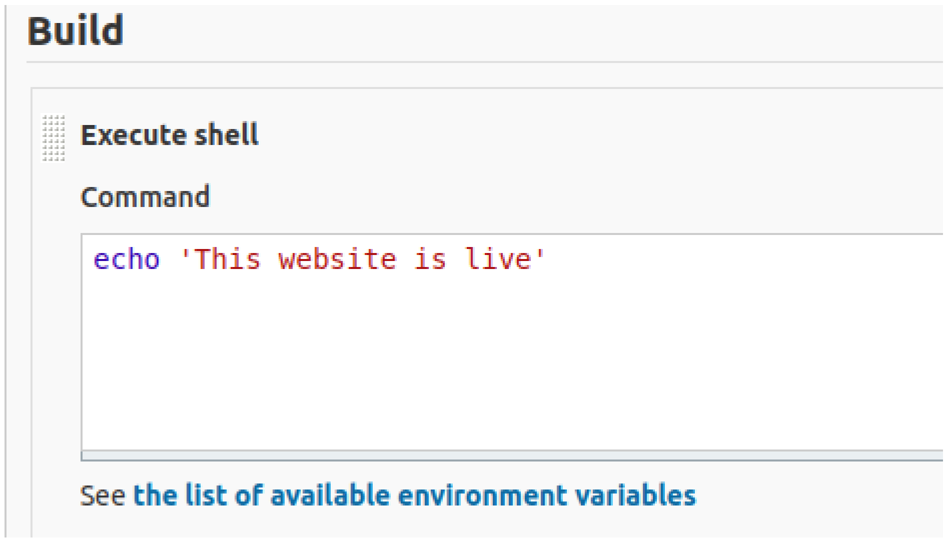
Description

I just created this new job

[Plain text] Preview

- [ ] Discard old builds
- [ ] GitHub project
- [ ] This build requires lockable resources
- [ ] This project is parameterized
- [ ] Throttle builds
- [ ] Disable this project
- [ ] Execute concurrent builds if necessary

4. Leave Build Triggers as is and write a shell command to be executed on the build.



Build

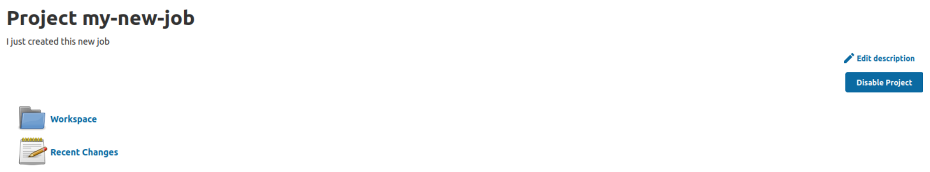
Execute shell

Command

```
echo 'This website is live'
```

See the list of available environment variables

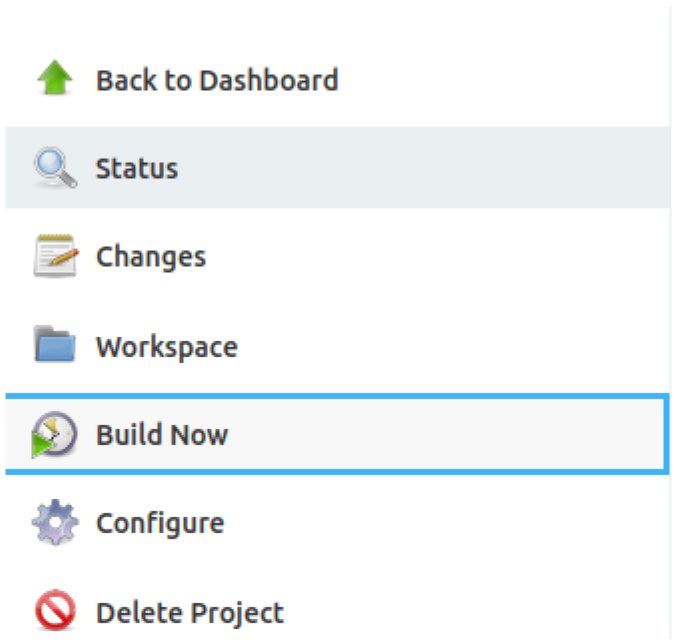
5. The New project is created.



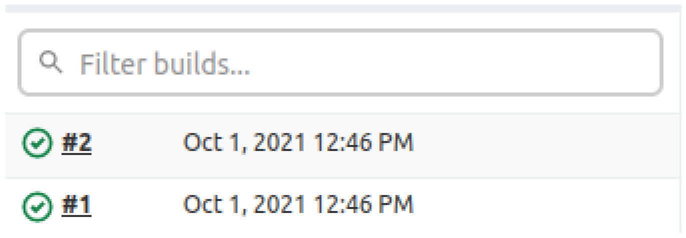
Project my-new-job

I just created this new job

6. On your left, click on build to start building.

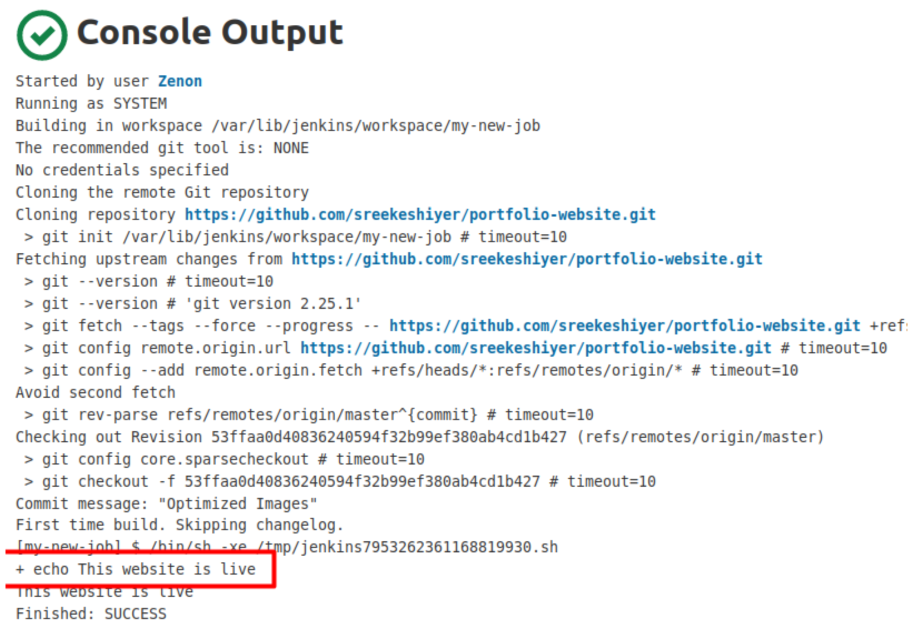


Click on one of your builds to see the console output.



#2	Oct 1, 2021 12:46 PM
#1	Oct 1, 2021 12:46 PM

7. Look for your executed command in the console output.



Console Output

```
Started by user Zenon
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/my-new-job
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/sreekeshiyer/portfolio-website.git
 > git init /var/lib/jenkins/workspace/my-new-job # timeout=10
Fetching upstream changes from https://github.com/sreekeshiyer/portfolio-website.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/sreekeshiyer/portfolio-website.git +ref
 > git config remote.origin.url https://github.com/sreekeshiyer/portfolio-website.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 53ffaa0d40836240594f32b99ef380ab4cd1b427 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 53ffaa0d40836240594f32b99ef380ab4cd1b427 # timeout=10
Commit message: "Optimized Images"
First time build. Skipping changelog.
[my-new-job] $ /bin/sh -xe /tmp/jenkins7953262361168819930.sh
+ echo This website is live
This website is live
Finished: SUCCESS
```

That is it, we successfully created our first build job on Jenkins!

Conclusion

In this experiment, we installed Jenkins along with its prerequisites on our Ubuntu machine and created our first freestyle build job on Jenkins.