

Jenkins Boost Camp

Jenkins Installation.....	3
Jenkins on Windows.....	3
Jenkins on Ubuntu.....	8
Jenkins on Mac.....	11
Jenkins the Basic.....	13
Create “Hello world” Job.....	13
Add Build Step.....	14
Start the First Build.....	15
Console Output.....	16
Intro GOL Project (or World without Jenkins).....	17
Git Installation and Clone GOL Repository.....	17
Maven Installation.....	19
Run Maven Build Manually.....	21
Install Tomcat Server as Service.....	22
Deploy GOL to Tomcat Server.....	25
Continuous Integration with Jenkins.....	25
Jenkins in the Big Picture of CI, CD and DevOps.....	25
Create GOL Job.....	26
Source Code Management.....	27
Check if git is installed in local machine.....	27
Specify source code repository.....	28
Check if setting working fine.....	29
Build Triggers.....	31
Build.....	36
Configure Build Step.....	36
Check Log and Workspace.....	38
Continuous Inspection with Jenkins.....	39
Job Configure for Test Report.....	39
Test Report on Job Dashboard.....	41

Continuous Delivery with Jenkins.....	41
Install “Deploy to Container Plugin”.....	42
Add Deploy Step.....	43
Check if Deploy Step Run Successfully.....	45
Continuous Monitoring with Jenkins.....	45
Create Slack Team and Install Jenkins CI App.....	45
Install Slack Notification Plugin to Jenkins.....	47
Configure GOL Using Slack Notification.....	48
Check if Notification Work.....	49
Distributed Build System with Jenkins.....	50
Jenkins Architecture.....	50
Windows Slave Agent.....	50
Enable Launch Slave Agents via Java Web Start.....	50
Add New Windows Slave.....	51
Launch Slave Agent.....	53
Running Job with Windows Slave Agent.....	56
Ubuntu Slave Agent.....	58
Enable Launch Slave Agents via Java Web Start.....	58
Add New Ubuntu Slave Agent.....	58
Launch Slave Agent.....	59
Jenkins Pipeline.....	61
What is Jenkins Pipeline ?.....	61
Create a Jenkinsfile.....	61
Create Pipeline Job.....	63
Build with Pipeline.....	65
.....	66
Running Pipeline On Specific Agent.....	66
Deploy to Tomcat with Pipeline.....	67
Jenkins in the Cloud.....	69
Create a Key to Connect between Master and Slave.....	70
Install Jenkins Server on Cloud.....	71
.....	72
Install Java on Slave Machine.....	74
Configure a New Slave Agent from Jenkins.....	74
Create a new credential.....	74
Create a new node.....	75
Create a Job Using Running on Cloud.....	77
Jenkins Security.....	78
Create gol_developer user.....	78
Matrix-based Security.....	78
Role-based Security.....	79
Install Role-based Plugin.....	79
Configure Global Role.....	80
Configure Project Role.....	81
Disable Security.....	82
Jenkins Backup and Restore.....	83
Install and Configure ThinBackup Plugin.....	83
Manually Backup.....	85

Restore from Backup.....	86
Jenkins Interview Questions.....	87
Jenkins Interview Question.....	87
What is Jenkins?.....	87
What is the requirement for using Jenkins?.....	87
What are the advantages of Jenkins?.....	87
How you can move or copy Jenkins from one server to another?.....	87
What are the commands you can use to start Jenkins manually?.....	88
How can create a backup and copy files in Jenkins?.....	88
How you can set up Jenkins job?.....	88
What are the two components Jenkins is mainly integrated with?.....	88
CI/CD Interview Question.....	88
What is continuous integration?.....	88
Why is Continuous Integration important?.....	89
What are the success factors for Continuous Integration?.....	89
Special Bonus.....	90

Jenkins Installation

In this chapter, we will jump into how to install and start up Jenkins in 3 environment : Window, Ubuntu and Mac.

In all environment, Jenkins will be installed as service, it mean Jenkins will automatically start up every time you machine power on or restart machine.

Jenkins on Windows

Check if java is installed : Java should be installed in Windows machine, to check if Java is installed, start a command prompt and typing in **java -version** Java version show up as below. Incase java not yet install, please install java before process to installation.

```

Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

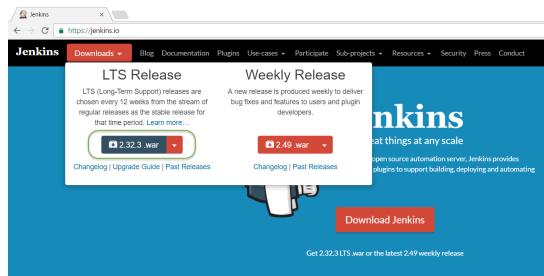
C:\Users\TAN>java -version
java version "1.8.0_111"
Java(TM) SE Runtime Environment (build 1.8.0_111-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.111-b14, mixed mode)

C:\Users\TAN>

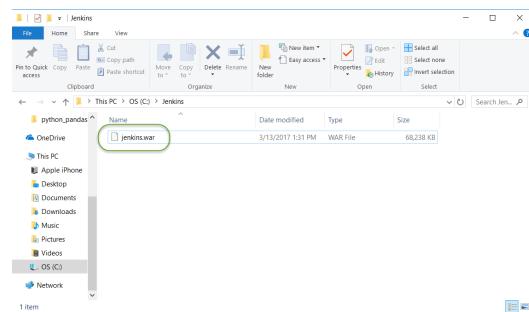
```

Install process

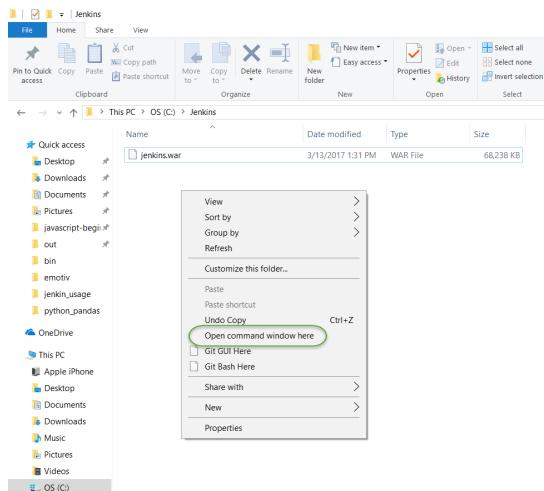
Step 1 : Go to <https://jenkins.io/> and download for stable version of Jenkins



Step 2 : Create a new folder for Jenkins (for example **C:\Jenkins**) and put file **jenkins.war** inside this folder

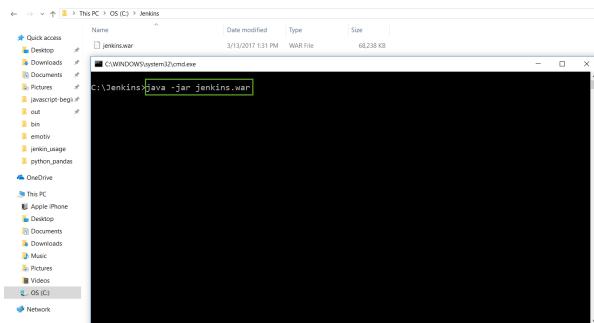


Step 3 : Press to shift and right click at same time to open context menu and select **Open command window here**



Step 4 : Typing in command prompt following to start installation **java -jar jenkins.war**

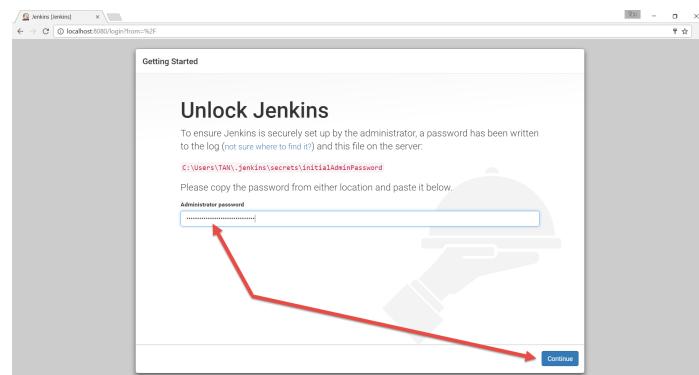
Then wait a while for complete running.



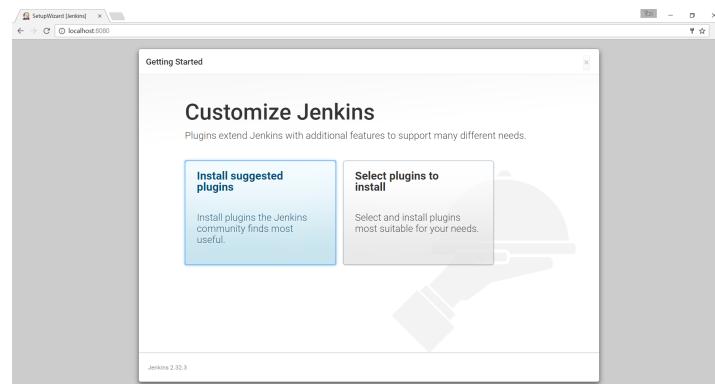
In the output log, you will see default password for first login of admin user, copy this password to notepad and save it for use on next step.

```
INFO:
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
6043ccce10c554047a3e951b60a4b427b
This may also be found at: C:\Users\TAN\.jenkins\secrets\initialAdminPassword
*****
```

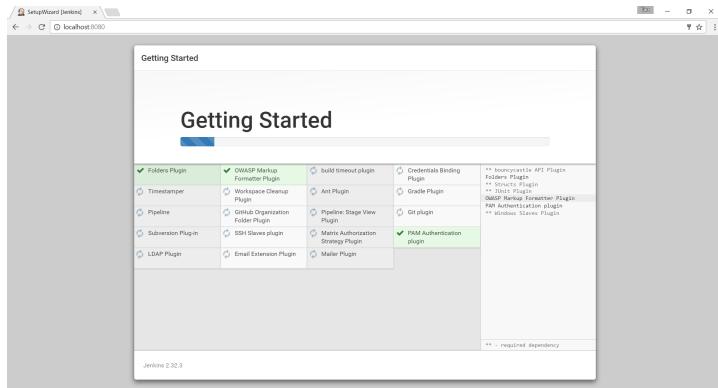
Step 5 : Now open browser and access to <http://localhost:8080> and then paste admin password to unlock Jenkins



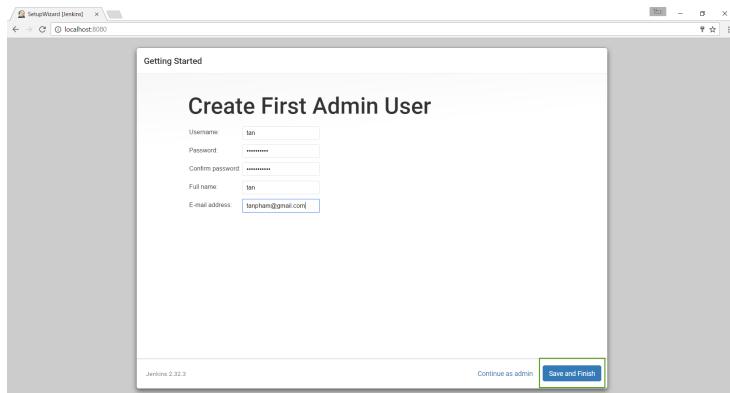
Wait a while and following screen show up, select default option **Install suggested plugins**



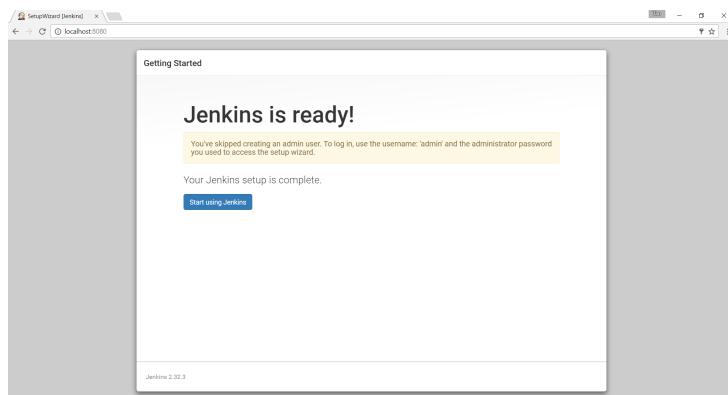
At this step default plugins will be installed.



Step 6 : Enter your user name, password for a new admin, then click to **Save and Finish**



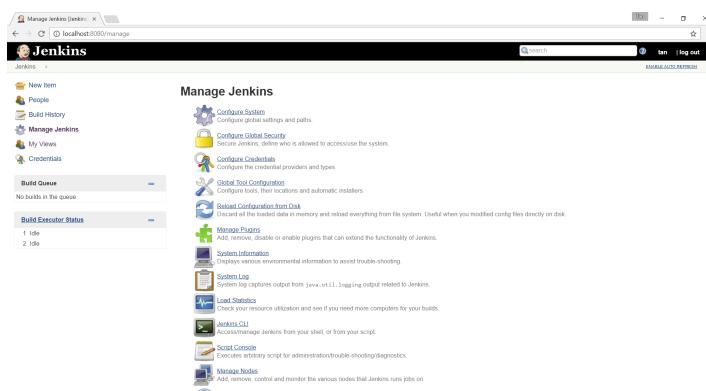
Step 7 : Click to **Start using Jenkins**, Congregation at this step you already complete for Jenkins installation



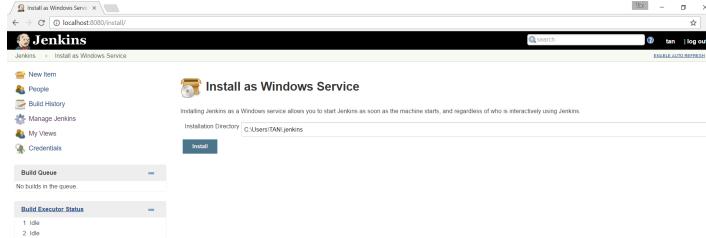
Jenkins auto login by new admin user already created



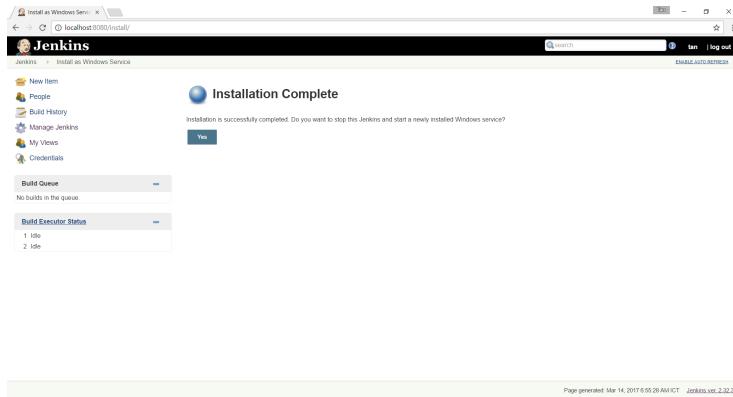
Step 8 : Click to *Manage Jenkins* from main screen, scroll down and click to *Install as Windows Service*



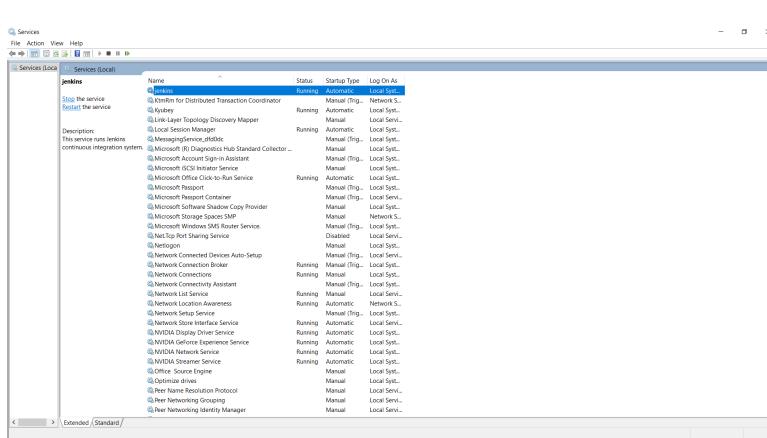
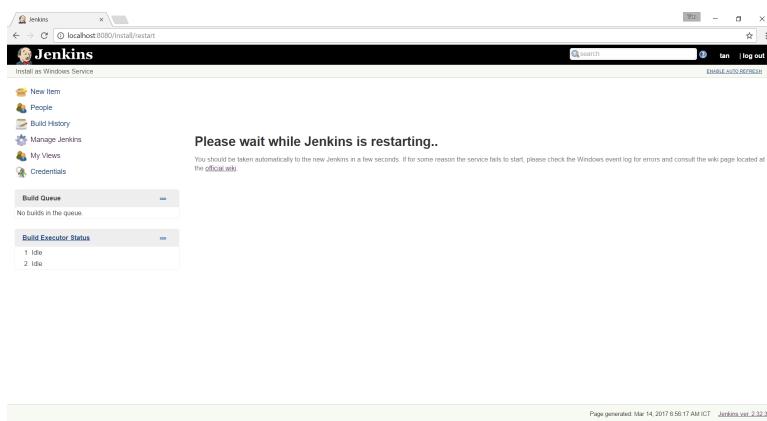
Click to *Install*



Click to *Yes*

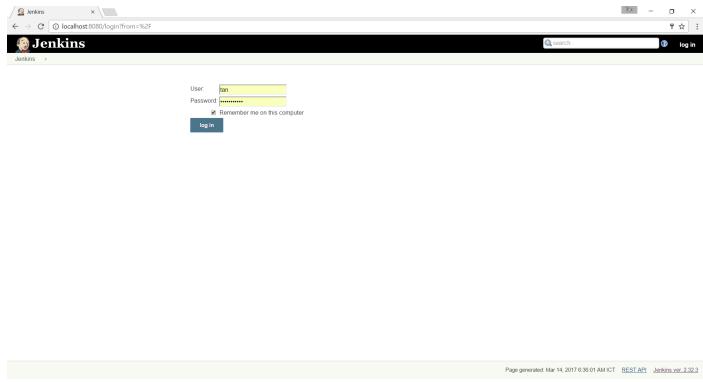


Wait some time and you will see jenkins service is running from windows service manager



That it, You complete install Jenkins on Windows as service. From now on every time Windows power up or restart, Jenkins will ready to access at <http://localhost:8080>

You can try to login Jenkins with above user created.



Jenkins on Ubuntu

Check if java is installed

Java should be installed in Ubuntu machine, to check if Java is installed, start a terminal and typing in **java -version** , If Java already installed, you will see Java version show up.

```
tan@ubuntu: ~
tan@ubuntu:~$ java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
tan@ubuntu:~$
```

Install Process

Step 1 : Open terminal and typing in following command

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

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

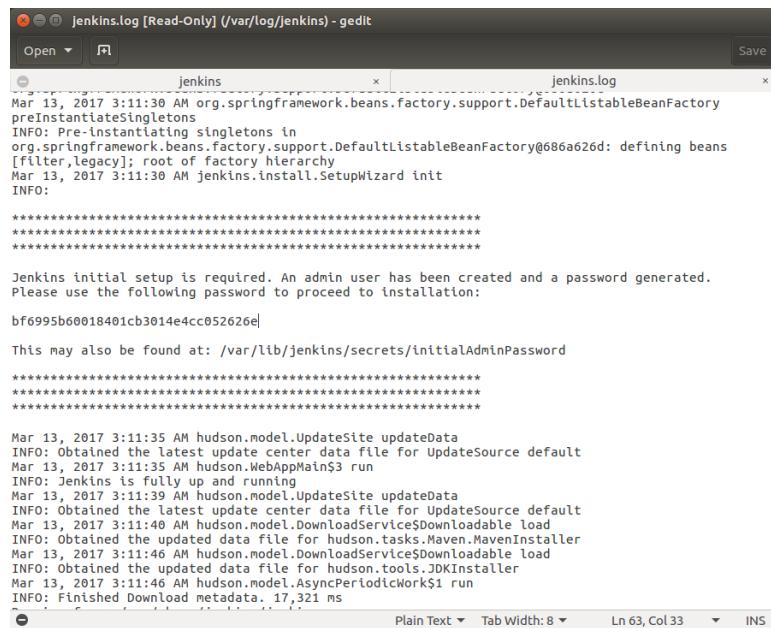
sudo apt-get update

sudo apt-get install jenkins
```

The above code do following job :

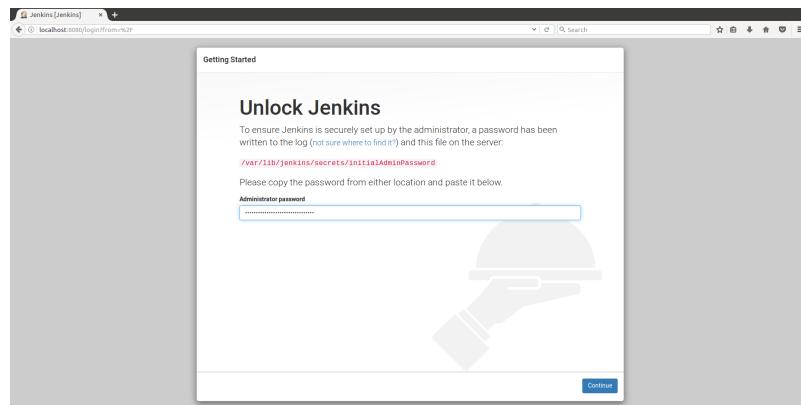
- Jenkins will be launched as a daemon up on start. See /etc/init.d/jenkins for more details.
- The 'jenkins' user is created to run this service.
- Log file will be placed in /var/log/jenkins/jenkins.log. Check this file if you are troubleshooting Jenkins.
- /etc/default/jenkins will capture configuration parameters for the launch like e.g JENKINS_HOME
- By default, Jenkins listen on port 8080. Access this port with your browser to start configuration.

Step 2 : Open the file **/var/log/jenkins/jenkins.log** and copy the default password which is automate generated during installation

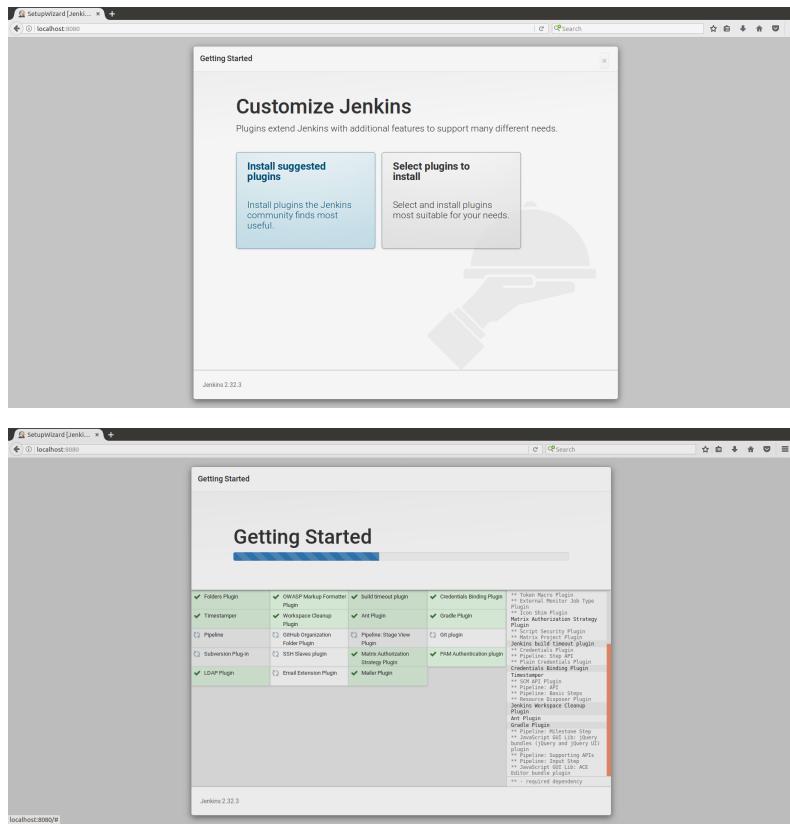


```
jenkins.log [Read-Only] (/var/log/jenkins) - gedit
Open Save
jenkins Jenkins.log
Mar 13, 2017 3:11:30 AM org.springframework.beans.factory.support.DefaultListableBeanFactory
preInstantiateSingletons
INFO: Pre-instantiating singletons in
org.springframework.beans.factory.support.DefaultListableBeanFactory@686a626d: defining beans
[filter,legacy]; root of factory hierarchy
Mar 13, 2017 3:11:30 AM jenkins.install.SetupWizard init
INFO:
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
bf6995b0018401cb3014e4cc052626e
This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
*****
Mar 13, 2017 3:11:35 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Mar 13, 2017 3:11:35 AM hudson.WebAppMain$3 run
INFO: Jenkins is fully up and running
Mar 13, 2017 3:11:39 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Mar 13, 2017 3:11:40 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Mar 13, 2017 3:11:46 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Mar 13, 2017 3:11:46 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata... 17,321 ms
Plain Text Tab Width: 8 Ln 63, Col 33 INS
```

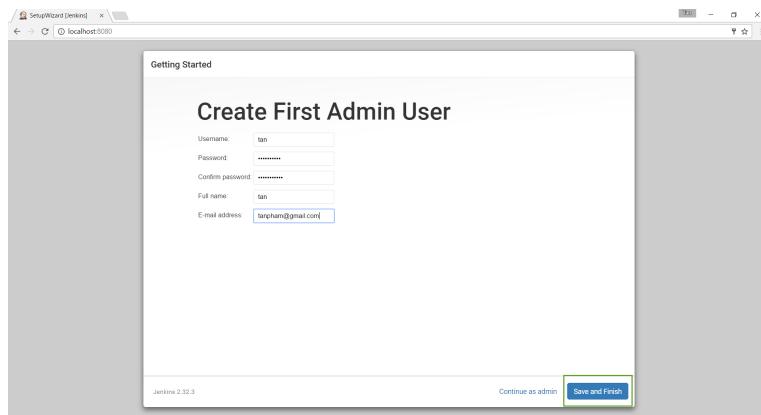
Step 3 : Open browser and access to Jenkins at <http://localhost:8080/> then paste the password above to unlock jenkins. Click to **Continue** button



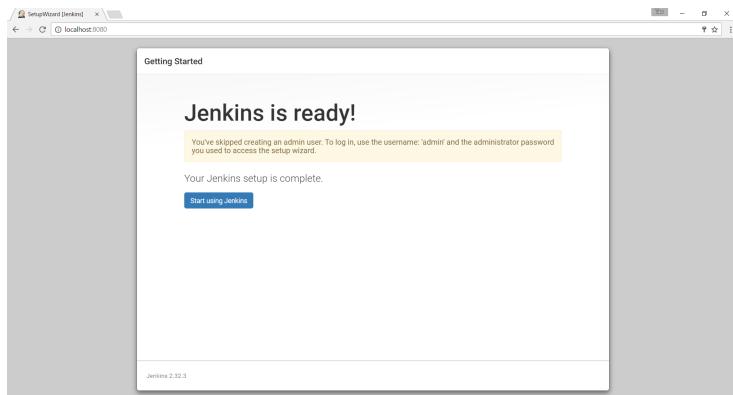
Step 4 : Click on *Install suggested plugins*



Step 5 : Enter your user name, password for a new admin.



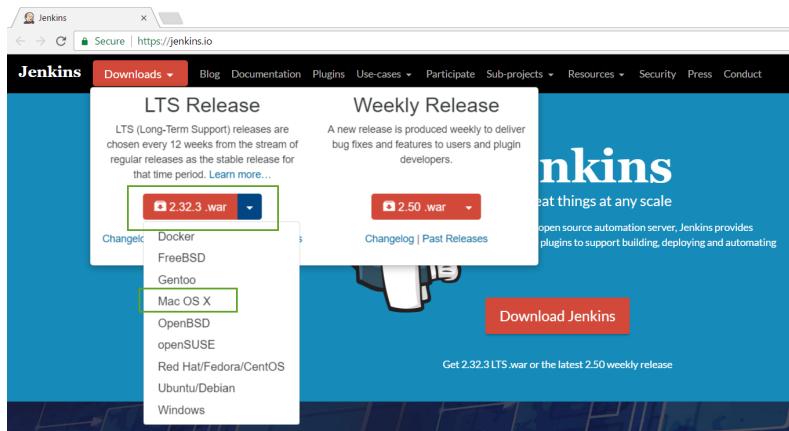
Step 6 : That it, now jenkins ready to use, click to *Start using Jenkins*



From now Jenkins already installed in Ubuntu as daemon service, so every time you power on machine, Jenkins will start and ready to access at <http://localhost:8080/>

Jenkins on Mac

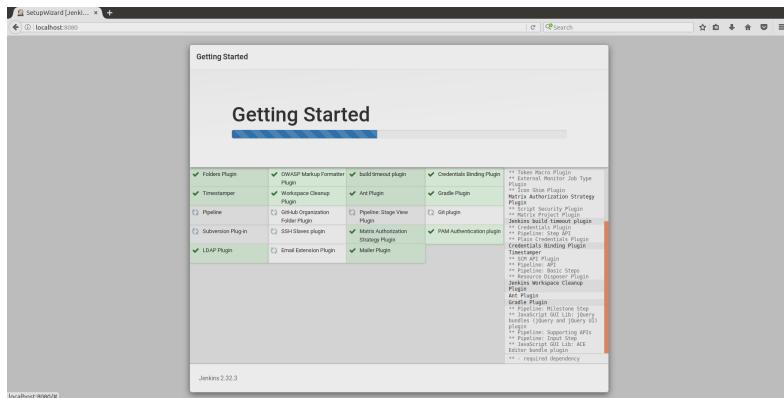
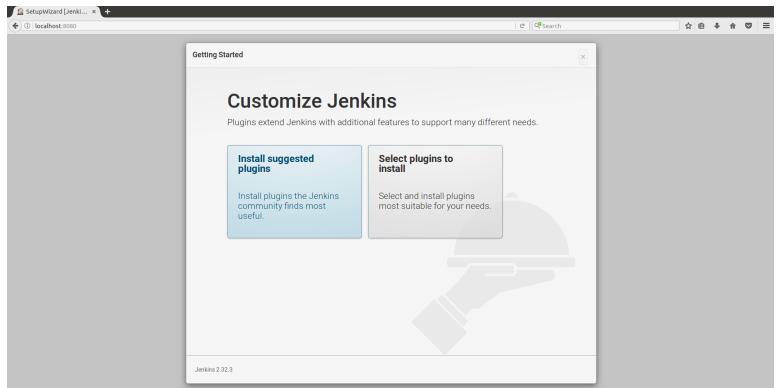
Step 1 : Download installer for Mac from <https://jenkins.io/> . Then run the installer file.



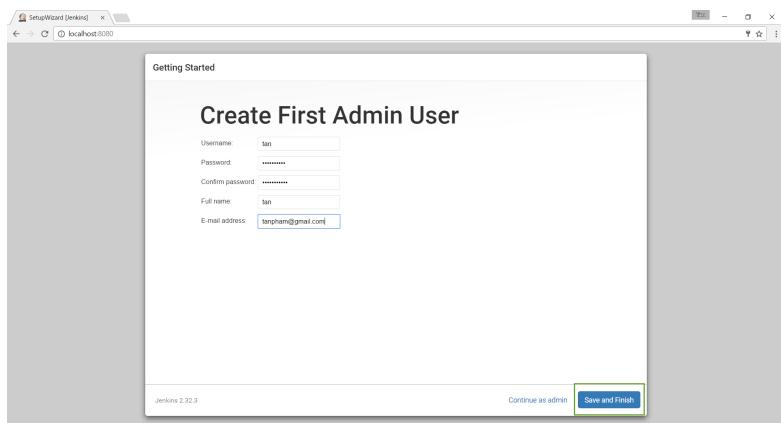
Step 2 : After complete install, you access Jenkins at <http://localhost:8080/>

Step 3 : Open file **/Users/Shared/Jenkins/Home/secrets/initialAdminPassword** and copy password use for unlock Jenkins

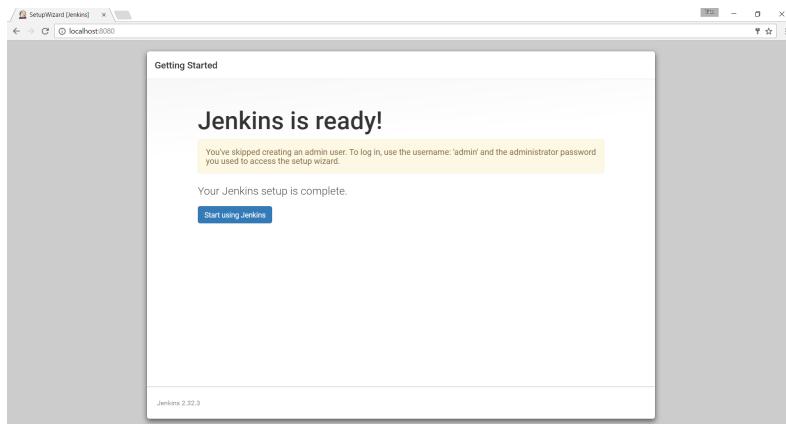
Step 4 : Click on **Install suggested plugins**



Step 5 : Enter your user name, password for a new admin.



Step 6 : That it, now jenkins ready to use, click to **Start using Jenkins**



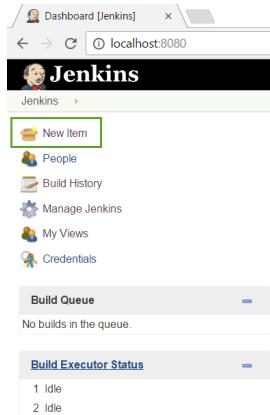
From now Jenkins already installed in Mac as a service, so every time you power on machine, Jenkins will start and ready to access at <http://localhost:8080/>

Jenkins the Basic

In this section I will explain Jenkins role in the world of continuous integration, continuous delivery and devops. And then we will go through steps in order to create “Hello World” Jenkins job.

Create “Hello world” Job

From Jenkins home page, click to **New Item**



In create project page, typing in “Hello World” to project name, select **Freestyle project** and then click to save button.

Enter an item name

Hello World

(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 slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system.

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

Add Build Step

In configure job page, select **Build** tab, click to **Add build step** and select **Execute Windows batch command** (in case you are using Linux please select **Execute shell**)

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Build Environment

- Delete workspace before build starts
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Use secret text(s) or file(s)

Build

Add build step ▾

- Copy artifacts from another project
- Execute Windows batch command**
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to <pending> on GitHub commit

Typing in command "echo Hello World". The purpose of this job is just echo to log message "Hello World" text.

Build

Execute Windows batch command

Command **echo Hello World**

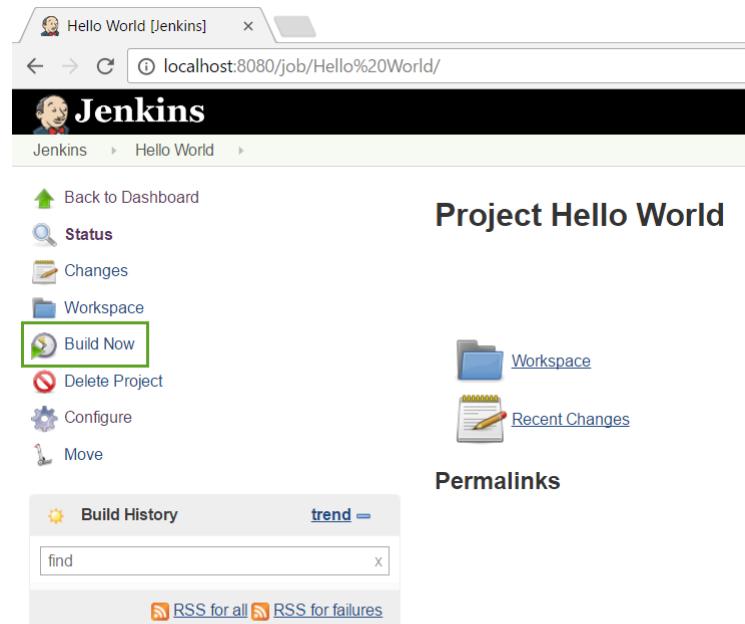
See the list of available environment variables Advanced...

Add build step ▾

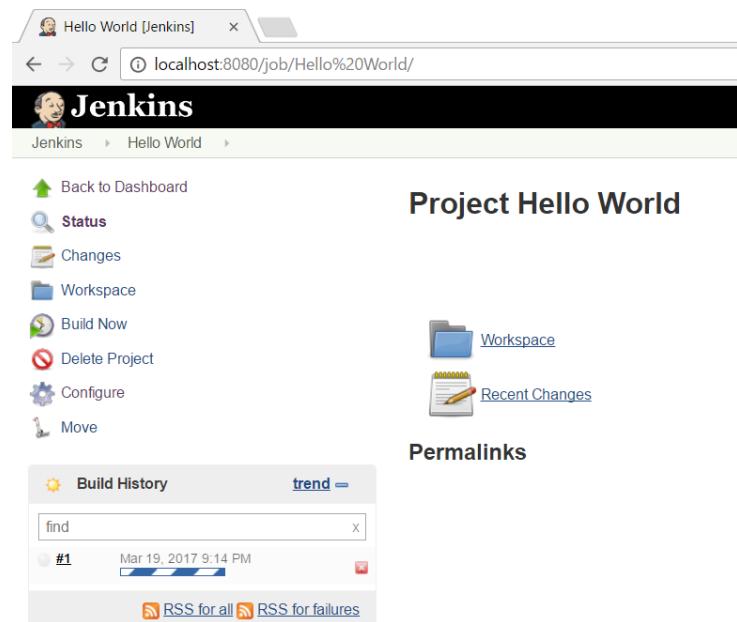
Click **Save** button and we complete the first job!

Start the First Build

From job dashboard, click **Build Now** and you will see job run. This job just do one thing, invoke the command "echo hello world"



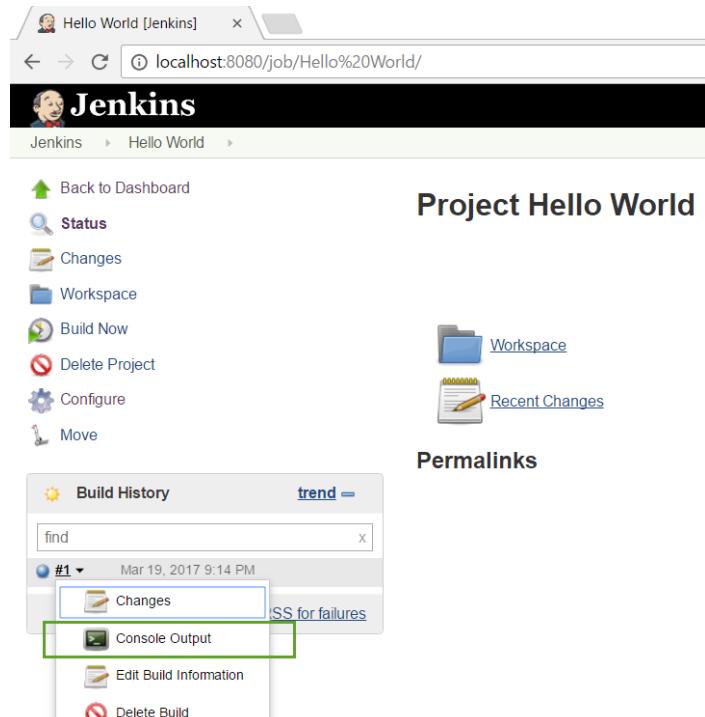
The screenshot shows the Jenkins Project Hello World dashboard. On the left, there is a sidebar with various links: Back to Dashboard, Status, Changes, Workspace, Build Now (which is highlighted with a green border), Delete Project, Configure, and Move. In the center, the title is "Project Hello World". Below the title are two links: "Workspace" and "Recent Changes". At the bottom, there is a "Permalinks" section and a "Build History" table. The table has a single row with a radio button labeled "#1", a timestamp "Mar 19, 2017 9:14 PM", and a progress bar.



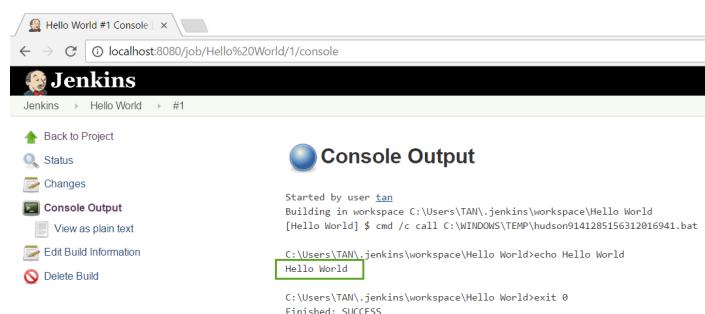
The screenshot shows the same Jenkins Project Hello World dashboard as the previous one, but now it displays a completed build. The "Build History" table shows a single row for build #1, which was run on March 19, 2017, at 9:14 PM. The progress bar is now fully blue, indicating a successful build. The rest of the interface remains the same, with the sidebar, central title, and permalinks section.

Console Output

Now job run completely, click to build number and select **Console Output**. You will see message “Hello World” show up inside build log. That it you just complete the first Jenkins job.



The screenshot shows the Jenkins interface for the 'Hello World' project. On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, and Move. The main area is titled 'Project Hello World'. Below the title, there are two sections: 'Workspace' and 'Recent Changes'. Under 'Recent Changes', there's a 'Permalinks' section. The 'Build History' section shows a single build (#1) from March 19, 2017, at 9:14 PM. The 'Actions' dropdown menu for this build is open, showing options: Changes, Console Output (which is highlighted with a green border), ISS for failures, Edit Build Information, and Delete Build.



The screenshot shows the Jenkins 'Console Output' page for build #1. The left sidebar includes links for Back to Project, Status, Changes, Console Output (which is selected and highlighted with a green border), View as plain text, Edit Build Information, and Delete Build. The main content area displays the build log:

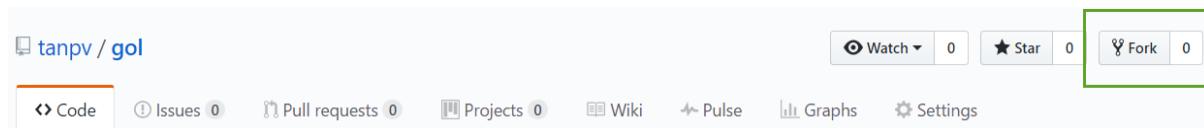
```
Started by user tan
Building in workspace C:\Users\TAN\jenkins\workspace\Hello World
[Hello World] $ cmd /c call C:\WINDOWS\TEMP\hudson9141285156312016941.bat
C:\Users\TAN\jenkins\workspace\Hello World>echo Hello World
Hello World
C:\Users\TAN\jenkins\workspace\Hello World>exit 0
Finished: SUCCESS
```

Intro GOL Project (or World without Jenkins)

In this book, project called GOL (game of life) is used to express the features related to Jenkins. So it will be helpful to review some information related to this project.

Git Installation and Clone GOL Repository

GOL project is hosted on Github at <https://github.com/tanpv/gol>. Please **fork** this repo to your Github account so you can try Jenkins features by yourself later.



To install Git, download software at <https://git-scm.com/>, and install with default option.

After install, open a cmd prompt and typing in **git --help** to check if install successfully

```
Command Prompt
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\TAN>git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           <command> [<args>]
```

Now create a folder name GOL and init it with command **git init**

```
C:\WINDOWS\system32\cmd.exe

C:\Users\TAN\GOL>git init
Initialized empty Git repository in C:/Users/TAN/GOL/.git/
```

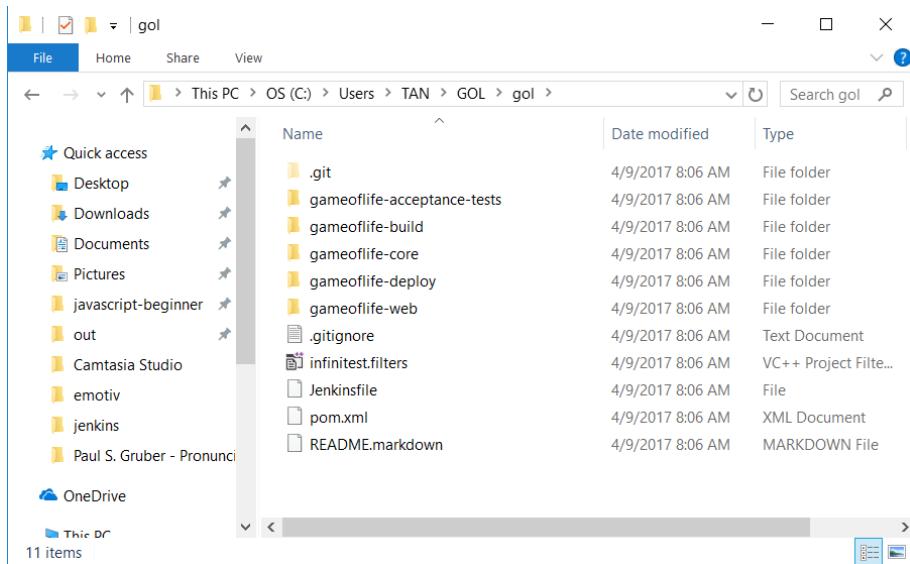
And finally, clone the repo from Github to local with command **git clone**
https://github.com/tanpv/gol.git

```
C:\WINDOWS\system32\cmd.exe - git clone https://github.com/tanpv/gol.git

C:\Users\TAN\GOL>git init
Initialized empty Git repository in C:/Users/TAN/GOL/.git/

C:\Users\TAN\GOL>git clone https://github.com/tanpv/gol.git
Cloning into 'gol'...
remote: Counting objects: 1399, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 1399 (delta 2), reused 0 (delta 0), pack-reused 1393 receiving objects
Receiving objects: 99% (1386/1399), 1004.01 KiB | 594.00 KiB/s
Receiving objects: 100% (1399/1399), 1.60 MiB | 594.00 KiB/s, done.
Resolving deltas: 100% (1017/1017), done.
```

And finally, you will see GOL project is completely clone to local



Maven Installation

First step, maven should be installed in machine which running Jenkins job. To check this, just open command prompt and typing in ***mvn --help***, you will see some all available maven command.

```
C:\Windows\SysWOW64\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\SysWOW64>mvn --help

usage: mvn [options] [<goal(s)>] [<phase(s)>]

Options:
-am,--also-make           If project list is specified, also
                          build projects required by the
                          list
-amd,--also-make-dependents  If project list is specified, also
                                build projects that depend on
                                projects on the list
-B,--batch-mode          Run in non-interactive (batch)
                         mode
-b,--builder <arg>       The id of the build strategy to
                         use.
-C,--strict-checksums    Fail the build if checksums don't
                         match
-c,--lax-checksums       Warn if checksums don't match
                         Ineffective, only kept for
                         backward compatibility
-cpu,--check-plugin-updates Define a system property
-D,--define <arg>        Produce execution error messages
-e,--errors               Encrypt master security password
-emp,--encrypt-master-password <arg> Encrypt server password
-ep,--encrypt-password <arg>
-f,--file <arg>          Force the use of an alternate POM
                         file (or directory with pom.xml).
-fae,--fail-at-end       Only fail the build afterwards;
```

In case maven not yet install follow these steps to do maven installation (for Window)

- Download maven from link <http://maven.apache.org/download.cgi>

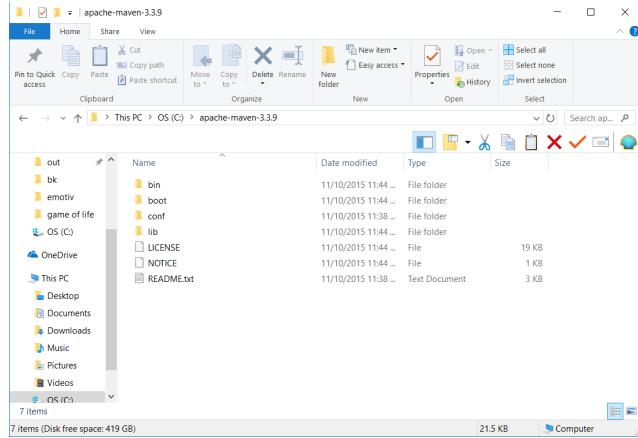
Files

Maven is distributed in several formats for your convenience. Simply pick a ready-made binary distribution archive and follow the [installation instructions](#). Use a source archive if you intend to build Maven yourself.

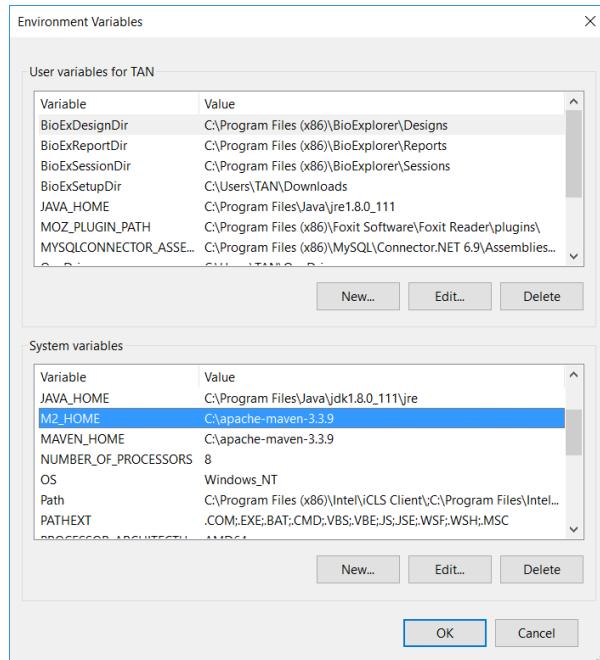
In order to guard against corrupted downloads/installations, it is highly recommended to verify the signature of the release bundles against the public [KEYS](#) used by the Apache Maven developers.

Link	Checksum	Signature
Binary tar.gz archive	apache-maven-3.3.9-bin.tar.gz	apache-maven-3.3.9-bin.tar.gz.asc
Binary zip archive	apache-maven-3.3.9-bin.zip	apache-maven-3.3.9-bin.zip.asc
Source tar.gz archive	apache-maven-3.3.9-src.tar.gz	apache-maven-3.3.9-src.tar.gz.asc
Source zip archive	apache-maven-3.3.9-src.zip	apache-maven-3.3.9-src.zip.asc

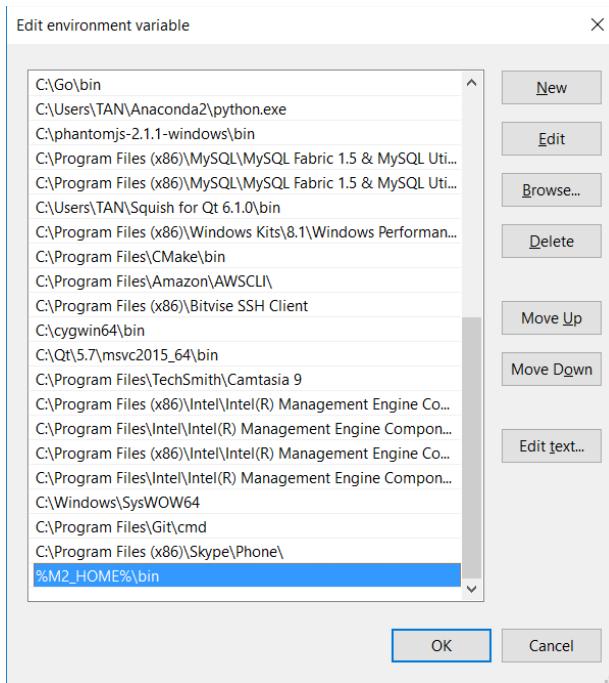
- Unzip zip file to one folder (for example I unzip to C:\apache-maven-3.3.9)



- Add M2_HOME and MAVEN_HOME variable point to above foder



- Add mavin bin to Path variable so you can call maven from any place



That it, to check maven installed successfully, open cmd and typing in **mvn --help**
Help content for mvn should show up correctly

```
C:\Windows\SysWOW64\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\SysWOW64>mvn --help

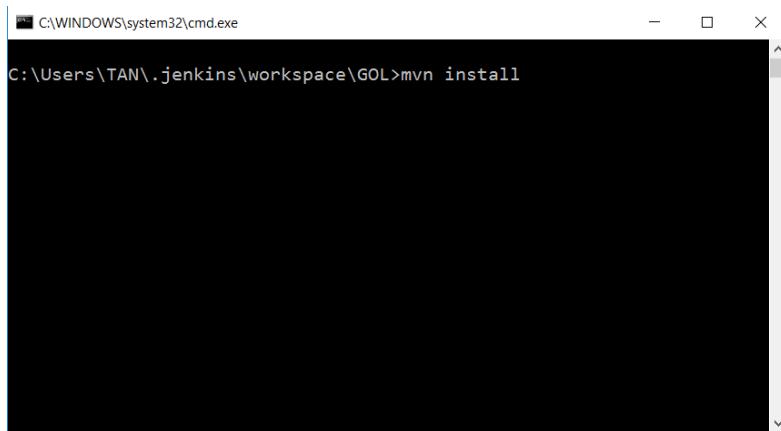
usage: mvn [options] [<goal(s)>] [<phase(s)>]

Options:
  -am,--also-make           If project list is specified, also
                           build projects required by the
                           list
  -amd,--also-make-dependents  If project list is specified, also
                           build projects that depend
                           on projects on the list
  -B,--batch-mode           Run in non-interactive (batch)
                           mode
  -b,--builder <arg>        The id of the build strategy to
                           use.
  -C,--strict-checksums     Fail the build if checksums don't
                           match
  -c,--lax-checksums        Warn if checksums don't match
  -cpu,--check-plugin-updates  Ineffective; only kept for
                           backward compatibility
  -D,--define <arg>         Define a system property
  -e,--errors               Produce execution error messages
  -emp,--encrypt-master-password <arg> Encrypt master security password
  -ep,--encrypt-password <arg> Encrypt server password
  -f,--file <arg>           Force the use of an alternate POM
                           file (or directory with pom.xml).
  -fae,--fail-at-end        Only fail the build afterwards;
```

Run Maven Build Manually

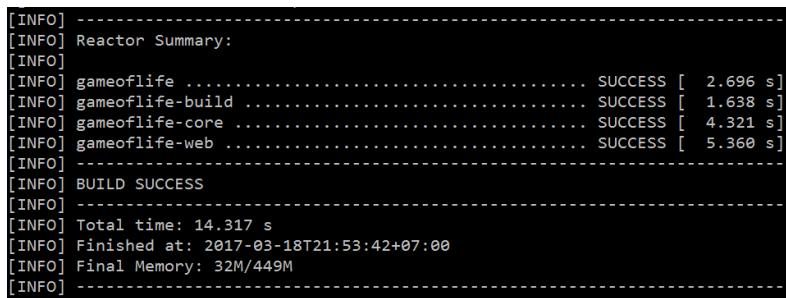
Now maven is ready, we can try to run the build for source code which we just clone from Github repository at **source code management** step.

Open **cmd** and change directory to repository GOL folder. And then typing in **mvn install**
This command will run java source unit test and then build this web application to a .war file



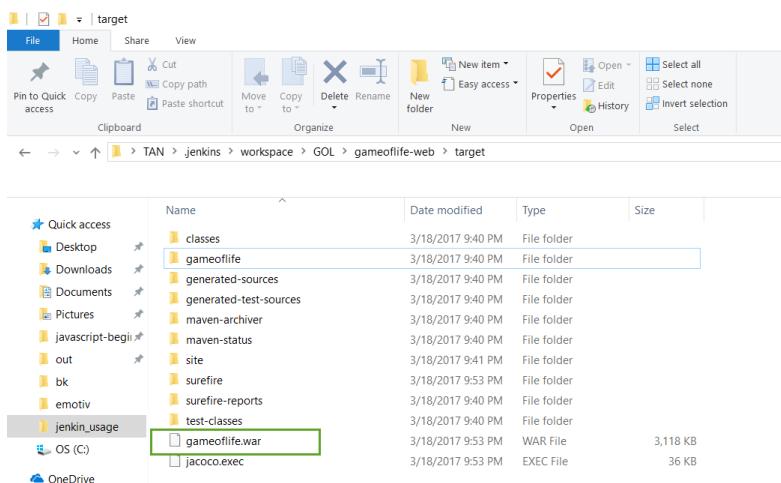
```
C:\WINDOWS\system32\cmd.exe
C:\Users\tan\.jenkins\workspace\GOL>mvn install
```

You will build result from command prompt



```
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] gameoflife ..... SUCCESS [ 2.696 s]
[INFO] gameoflife-build ..... SUCCESS [ 1.638 s]
[INFO] gameoflife-core ..... SUCCESS [ 4.321 s]
[INFO] gameoflife-web ..... SUCCESS [ 5.360 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 14.317 s
[INFO] Finished at: 2017-03-18T21:53:42+07:00
[INFO] Final Memory: 32M/449M
[INFO] -----
```

And can see actual .war file inside target folder of **gameoflife-web**. This folder could be used to deploy with web application server.



Install Tomcat Server as Service

To install Tomcat 7 server, you just follow these steps:

- Download Tomcat 7 from this page <http://tomcat.apache.org/download-70.cgi>, note that we will download version [Windows Service Installer](#) so Tomcat will be installed as window service

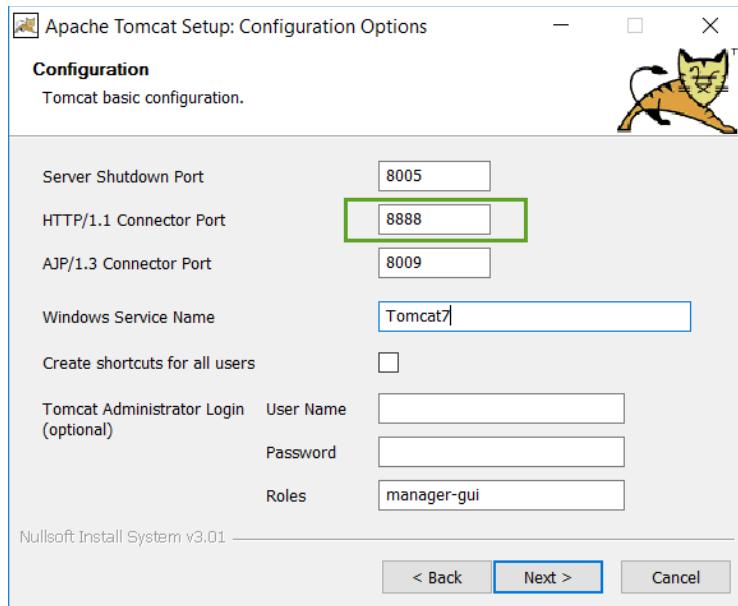
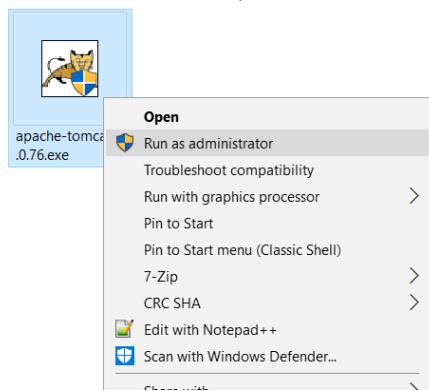
7.0.76

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, md5, sha1\)](#)
 - [tar.gz \(pgp, md5, sha1\)](#)
 - [32-bit Windows zip \(pgp, md5, sha1\)](#)
 - [64-bit Windows zip \(pgp, md5, sha1\)](#)
 - [32-bit/64-bit Windows Service Installer \(pgp, md5, sha1\)](#)
- Full documentation:
 - [tar.gz \(pgp, md5, sha1\)](#)
- Deployer:
 - [zip \(pgp, md5, sha1\)](#)
 - [tar.gz \(pgp, md5, sha1\)](#)
- Extras:
 - [JMX Remote jar \(pgp, md5, sha1\)](#)
 - [Web services.jar \(pgp, md5, sha1\)](#)
 - [JULI adapters.jar \(pgp, md5, sha1\)](#)
 - [JULI log4j.jar \(pgp, md5, sha1\)](#)
- Embedded:
 - [tar.gz \(pgp, md5, sha1\)](#)
 - [zip \(pgp, md5, sha1\)](#)

- Run installer and select port 8888 for running Tomcat (We already use 8080 for Jenkins installation)



- After install successfully, a Tomcat service will running and you can access Tomcat server from localhost

Services (Local)					
Apache Tomcat 7.0 Tomcat7	Name	Status	Startup Type	Log On As	
Stop the service	Apache Tomcat 7.0 Tomcat7	Running	Automatic	Local Syst...	
Restart the service	Time Broker	Running	Manual (Trigger Start)	Local Servi...	
Description:	Tile Data model server	Running	Automatic	Local Syst...	
Apache Tomcat 7.0.75 Server - http://tomcat.apache.org/	Themes	Running	Automatic	Local Syst...	
	Remote Desktop Services	Running	Manual	Network S...	
	Telephony	Running	Manual	Network S...	
	System Events Broker	Running	Automatic (Trigger Start)	Local Syst...	
	Supercat	Running	Automatic	Network S...	
	Dell SupportAssist Agent	Running	Automatic	Local Syst...	
	Storage Service	Running	Automatic (Delayed Start)	Local Syst...	
	Windows Image Acquisition (Running	Manual (Trigger Start)	Local Servi...	
	State Repository Service	Running	Manual	Local Syst...	
	Secure Socket Tunneling Protocol Service	Running	Manual	Local Servi...	
	SSDP Discover	Running	Manual	Local Syst...	

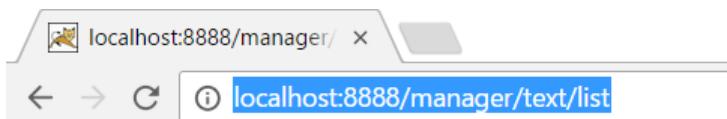
The screenshot shows the Apache Tomcat 7.0.75 web interface at localhost:8088. The page title is "Apache Tomcat/7.0.75". It features a banner with the text "If you're seeing this, you've successfully installed Tomcat. Congratulations!" and a cartoon cat icon. Below the banner are links for "Recommended Reading" including "Security Considerations HOW-TO", "Manager Application HOW-TO", and "Clustering/Session Replication HOW-TO". A sidebar on the left titled "Developer Quick Start" includes links for "Tomcat Setup", "First Web Application", "Realms & AAA", "JDBC Datasources", "Examples", and "Servlet Specifications". A "Server Status" button is also present.

- To control Tomcat from Jenkins, one user with role “manager-script” should be added to file **conf/tomcat-users.xml**. Then restart Tomcat server by service.

```
tomcat-users.xml [1]
1 <?xml version='1.0' encoding='cp1252'?>
2 <!DOCTYPE tomcat-users SYSTEM "http://tomcat.apache.org/xml/tomcat-users-dtd.dtd">
3 
4 <!-- Licensed to the Apache Software Foundation (ASF) under one or more
5     contributor license agreements. See the NOTICE file distributed with
6     this work for additional information regarding copyright ownership.
7     The ASF licenses this file to You under the Apache License, Version 2.0
8     (the "License"); you may not use this file except in compliance with
9     the License. You may obtain a copy of the License at
10 
11     http://www.apache.org/licenses/LICENSE-2.0
12 
13 Unless required by applicable law or agreed to in writing, software
14 distributed under the License is distributed on an "AS IS" BASIS,
15 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
16 See the License for the specific language governing permissions and
17 limitations under the License.
18 -->
19 
20 <tomcat-users>
21   <role rolename="manager-script"/>
22   <user username="admin" password="admin" roles="manager-script"/>
23 </tomcat-users>
```

Services (Local)					
Apache Tomcat 7.0 Tomcat7	Name	Status	Startup Type	Log On As	
Stop the service	Apache Tomcat 7.0 Tomcat7	Running	Automatic	Local Syst...	
Restart the service	Time Broker	Start	Manual (Trigger Start)	Local Servi...	
Description:	Tile Data model server	Stop	Automatic	Local Syst...	
Apache Tomcat 7.0.75 Server - http://tomcat.apache.org/	Themes	Pause	Automatic	Local Syst...	
	Remote Desktop Services	Resume	Automatic	Network S...	
	Telephony	Restart	Manual	Network S...	
	System Events Broker	All Tasks	Automatic (Trigger Start)	Local Syst...	
	Supercat	Refresh	Automatic	Local Syst...	
	Dell SupportAssist Agent	Properties	Automatic (Delayed Start)	Local Syst...	
	Storage Service	Help	Manual (Trigger Start)	Local Servi...	
	Windows Image Acquisition (Running	Automatic	Local Syst...	
	State Repository Service	Running	Manual	Local Syst...	
	Secure Socket Tunneling Protocol Service	Running	Manual	Local Servi...	
	SSDP Discover	Running	Manual	Local Syst...	

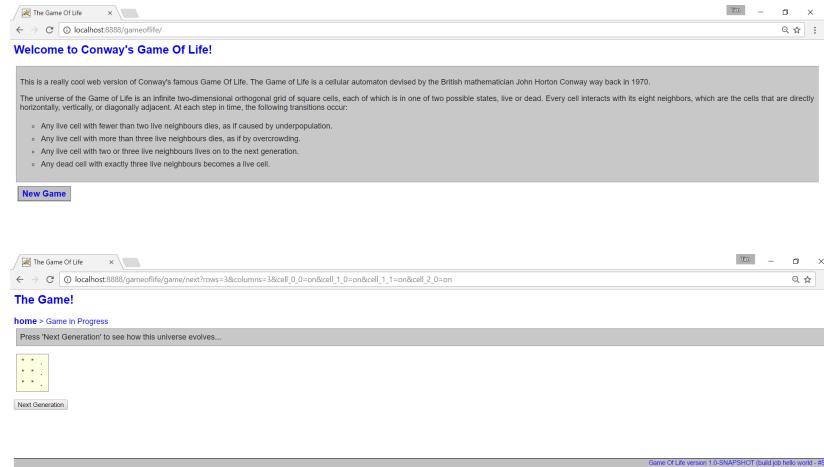
To check if admin user working, access to <http://localhost:8888/manager/text/list> and then put on user / password you just specify



```
OK - Listed applications for virtual host localhost
:running:0:ROOT
/gameoflife:running:0:gameoflife
/manager:running:0:manager
/docs:running:0:docs
```

Deploy GOL to Tomcat Server

Deploy this application to Tomcat server, and you will see some the app from browser and actually start a new game !.



Continuous Integration with Jenkins

This chapter will show you how to automate software build with Jenkins.

Jenkins in the Big Picture of CI, CD and DevOps

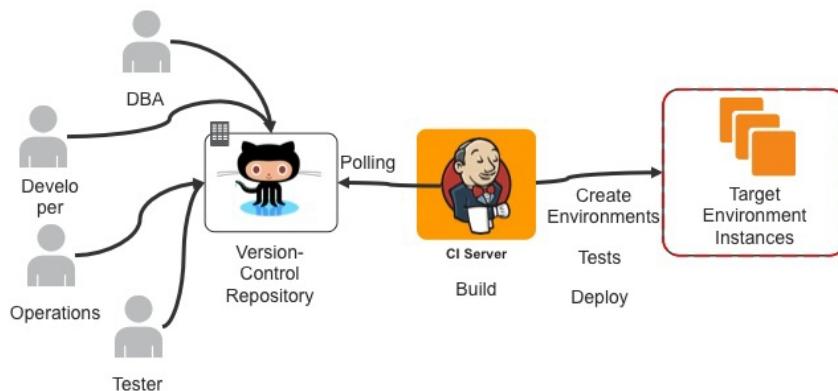


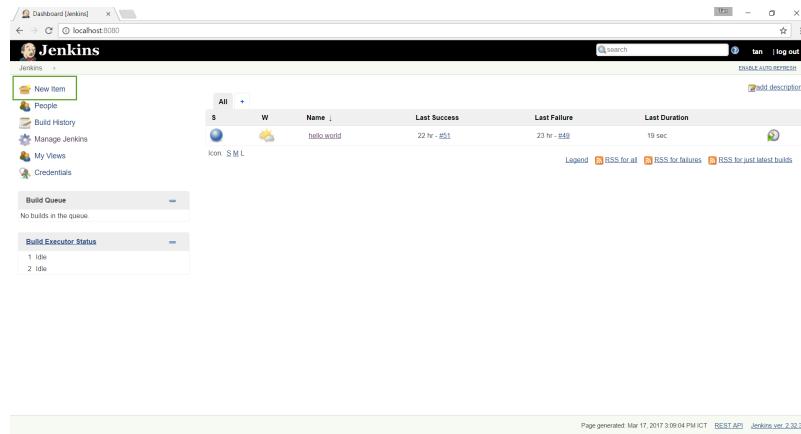
Image above show quite clear about Jenkins role in software development process.

- Developer do coding and commit their code to a version control repository like Github
- Jenkins actively check Github to see if have any change, in case has code change, Jenkins will pull the last code from Github to build machine (machine which run Jenkins job)

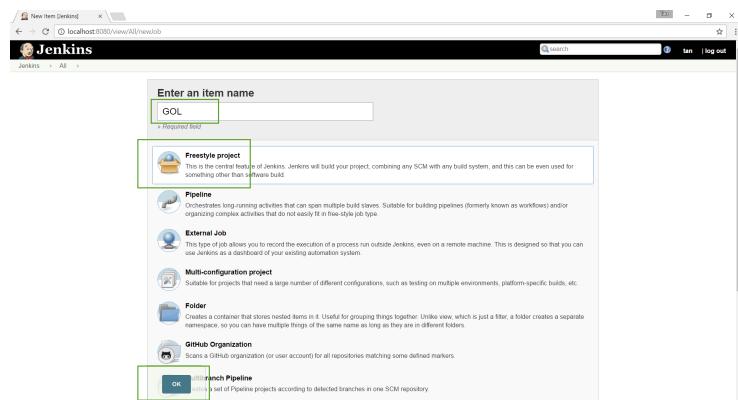
- Jenkins build software artifact from pulled source code (for example war file from .java source file)
- Jenkins run and show up unit test report
- Jenkins deploy artifact to target environment (for example a web server like : Tomcat, JBoss, WebSphere...)

Create GOL Job

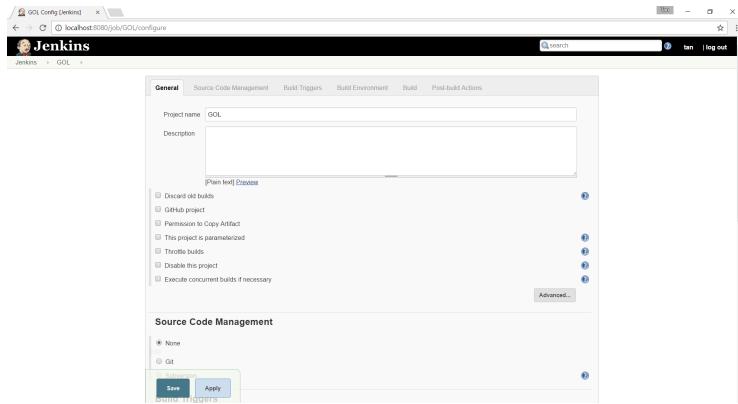
From home page, click to **New Item** link



Enter **GOL** for project name and then chose **Freestyle project** and finally click to **OK** button to create a new job with name is **GOL**



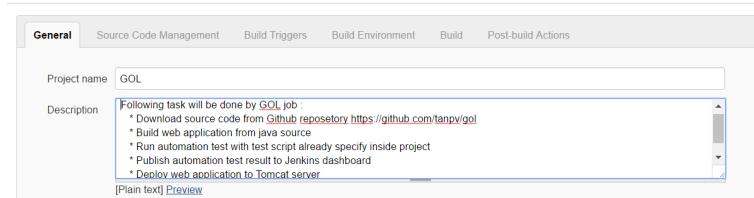
Then a new screen will show up allow us to configure everything for **GOL** job.



Description is place to describe what mission of this job and steps inside the job. Put follow text into **Description** session.

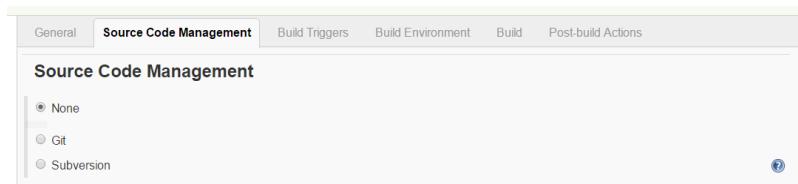
“ Following task will be done by GOL job :

- Check change on Github repo and automatically download source code from Github repository <https://github.com/tanpv/gol>
- Build web application from java source
- Run automation test with test script already specify inside project
- Publish automation test result to Jenkins dashboard
- Deploy web application to Tomcat server ”



Source Code Management

Purpose of this part is answer for question **Where to get source code ?** for our software build. From job configure, click into **Source Code Management** tab



Check if git is installed in local machine

In order to clone source code from Github, local machine should install Git.

To check if local machine already install **git** or not, open command prompt and typing **git --help**

```

C:\Windows\SysWOW64\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\SysWOW64>git --help
usage: git [--version] [--help] [-C <path>] [-c name=value]
           [-e exec-path=<path>] [-H html-path] [-m man-path] [--info-path]
           [-p | --paginate | --no-pager] [-n no-replace-objects] [-bare]
           [-g git-dir=<path>] [-w work-tree=<path>] [--namespace=<name>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv        Move or rename a file, a directory, or a symlink
  reset     Reset current HEAD to the specified state
  rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status

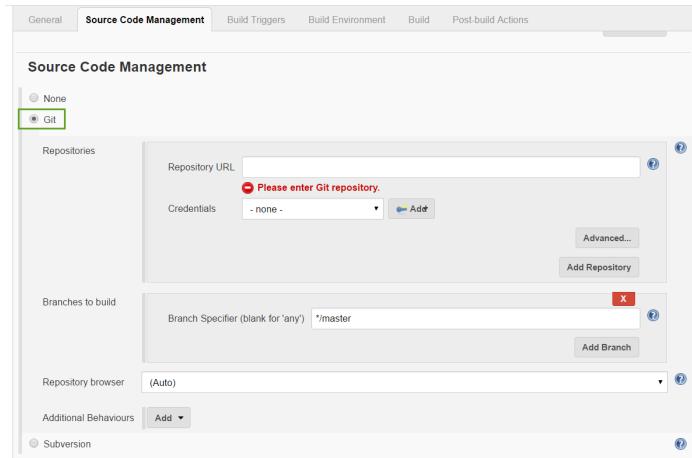
grow, mark and tweak your common history
  commit   Record changes to the repository
  merge    Join two or more development histories together
  rebase   Rewrite the local history
  tag      Create a tag or reference to a specific commit

```

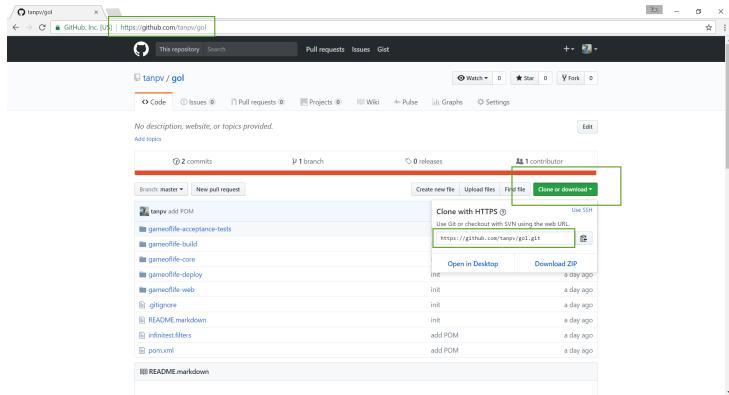
To download and install git, please refer to this link <https://git-scm.com/downloads>

Specify source code repository

Because we host project source code on Github, so you will choose **Git** as below.



To get repo link, go to Github <https://github.com/tanpv/gol> then click to **Clone or download** button, you will see the repo link. Copy this link.



Put the repo link <https://github.com/tanpv/gol.git> in to **Repository URL** section. The branch will be default as ***/master**

Click to **Save** button and that it, we already tell to Jenkins where to get source code.

Check if setting working fine

This is the time to try for first running and see if source code from Github is clone or not. Go back to home page then click to GOL job

The dashboard for GOL will show up, you will see describe which we just added before

The screenshot shows the Jenkins interface for the 'GOL' job. The left sidebar contains links for Back to Dashboard, Status, Changes, Workspace, Build Now (which is highlighted in blue), Delete Project, Configure, and Move. The main content area is titled 'Project GOL' and lists the following tasks:

- Following task will be done by GOL job:
- Download source code from Github repository <https://github.com/tanpv/gol>
- Build web application from java source
- Run automation test with test script already specify inside project
- Publish automation test result to Jenkins dashboard
- Deploy web application to Tomcat server

Below this, there are links for 'Workspace' and 'Recent Changes'. At the bottom, there are 'Permalinks' and RSS feed links for 'RSS for all' and 'RSS for failures'.

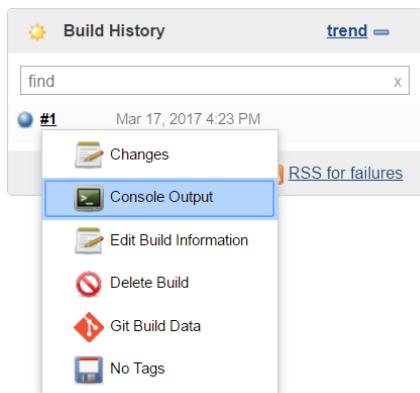
Click in to **Build Now** to activate job run immediately.

This screenshot is identical to the one above, but the 'Build Now' button in the sidebar has a green border around it, indicating it is currently selected or active.

You will see GOL start running as expected

The screenshot shows the 'Build History' page for build #1, which was run on Mar 17, 2017 at 4:23 PM. The progress bar is shown as a blue horizontal bar. At the bottom, there are RSS feed links for 'RSS for all' and 'RSS for failures'.

To see how job running, select **Console Output** from context menu while you click to build number



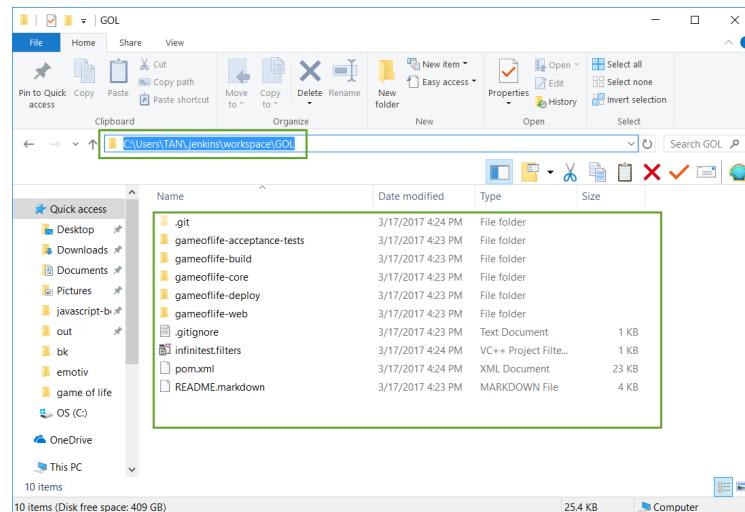
Then the job log will show up every thing happen in order to clone source code from Github repository. You can see that the build already success running.

```

Started by user tan
Building workspace C:\Users\tan\jenkins\workspace\GOL
Cloning the remote Git repository
Cloning repository https://github.com/tanpu/gol.git
> git.exe init C:\Users\tan\jenkins\workspace\GOL # timeout=10
Fetching upstream changes from https://github.com/tanpu/gol.git
> git.exe --version # timeout=10
> git.exe fetch --tags --progress https://github.com/tanpu/gol.git +refs/heads/*:refs/remotes/origin/*
> git.exe config --remote origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe config remote.origin.url https://github.com/tanpu/gol.git # timeout=10
Fetching upstream changes from https://github.com/tanpu/gol.git
> git.exe fetch --tags --progress https://github.com/tanpu/gol.git +refs/heads/*:refs/remotes/origin/*
> git.exe rev-parse "refs/remotes/origin/master^{commit}" # timeout=10
> git.exe rev-parse "refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 38b0942a6ec3857632fd392cd527f8526f06c25 (refs/remotes/origin/master)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 38b0942a6ec3857632fd392cd527f8526f06c25
First time build. Skipping changelog.
Finished: SUCCESS

```

Now we could go to local repository to see the source code already downloaded. Each job will have it's own folder inside **workspace** folder



That it, we complete the first step of this job.

Build Triggers

Project GOL

Following task will be done by GOL job :

- * Download source code from Github repository <https://github.com/tanpv/gol>
- * Build web application from java source
- * Run automation test with test script already specify inside project
- * Publish automation test result to Jenkins dashboard
- * Deploy web application to Tomcat server

Build History

#	Date
#2	Mar 17, 2017 4:54 PM
#1	Mar 17, 2017 4:23 PM

[RSS for all](#) [RSS for failures](#)

Permalinks

- [Last build \(#2\), 3 hr 56 min ago](#)
- [Last stable build \(#1\), 4 hr 26 min ago](#)
- [Last successful build \(#1\), 4 hr 26 min ago](#)
- [Last failed build \(#2\), 3 hr 56 min ago](#)
- [Last unsuccessful build \(#2\), 3 hr 56 min ago](#)
- [Last completed build \(#2\), 3 hr 56 min ago](#)

The first way and most simple way to start LOG job is just click in to **Build Now** button. But this way is manually and not so cool. We want to trigger Jenkins job running automatically, so to do this, click to **Configure** from LOG dashboard.

Project GOL

Following task will be done by GOL job :

- * Download source code from Github repository <https://github.com/tanpv/gol>
- * Build web application from java source
- * Run automation test with test script already specify inside project
- * Publish automation test result to Jenkins dashboard
- * Deploy web application to Tomcat server

Build History

#	Date
#2	Mar 17, 2017 4:54 PM
#1	Mar 17, 2017 4:23 PM

[RSS for all](#) [RSS for failures](#)

Permalinks

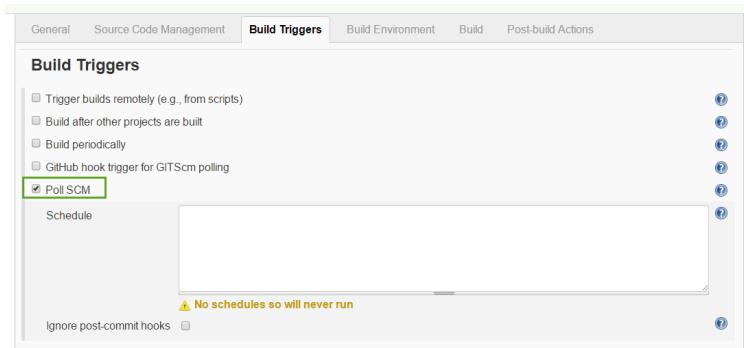
- [Last build \(#2\), 3 hr 56 min ago](#)
- [Last stable build \(#1\), 4 hr 26 min ago](#)
- [Last successful build \(#1\), 4 hr 26 min ago](#)
- [Last failed build \(#2\), 3 hr 56 min ago](#)
- [Last unsuccessful build \(#2\), 3 hr 56 min ago](#)
- [Last completed build \(#2\), 3 hr 56 min ago](#)

Then select to **Build Triggers** tab you will some option to configure so job could trigger automatically

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Have 5 ways to automate trigger Jenkins job as show above. In this book I will focus on **Poll SCM**, the good way in configure for a continuous system. Now you click on **Poll SCM**, you will see a text box show up for setting schedule



So following are steps show up how **Poll SCM** work :

- Base on schedule setting, Jenkins will actively check Github repository to see if have any change from repository
- If have any change on Github repo Jenkins job will trigger

Entry	Description	Equivalent To
@yearly (or @annually)	Run once a year at midnight in the morning of January 1	0 0 1 1 *
@monthly	Run once a month at midnight in the morning of the first of the month	0 0 1 * *
@weekly	Run once a week at midnight in the morning of Sunday	0 0 * * 0
@daily	Run once a day at midnight	0 0 * * *
@hourly	Run once an hour at the beginning of the hour	0 * * * *
@reboot	Run at startup	@reboot

* * * * * command to be executed

day of week (0 - 7) (0 or 7 are Sunday, or use names)
month (1 - 12)
day of month (1 - 31)
hour (0 - 23)
min (0 - 59)

Schedule with **Poll SCM** work follow cron schedule rule, above image show some cron schedule example. Basically, we want to catch the change on Github repository as soon as possible, so follow text will be add to **Schedule** * * * * *. This mean we want to check change on Github every minute !

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Build Triggers

- Trigger builds remotely (e.g., from scripts)
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Schedule:

Do you really mean "every minute" when you say * * * * *? Perhaps you meant "H * * * * to poll"**

Would last have run at Friday, March 17, 2017 9:41:54 PM ICT, would next run at Friday, March 17, 2017 9:41:54 PM ICT.

Ignore post-commit hooks

Click to **Save** and Jenkins will move you back to job dashboard



Now is the time to check if this setting really work ?

I will go to my GOL project repo on Github and change content of file **README.markdown** right from web browser (Github support commit change right from web browser)

This repository Search Pull requests Issues Gist

tanypv / gol

Code Issues Pull requests Projects Wiki Pulse Settings

No description, website, or topics provided. Add topics

2 commits 0 branches 0 releases 1 contributor

branch: master New pull request Create new file Upload files Find file Close or download

Latest commit 38d9842 a day ago

- tmpv add POM
- gamefile-acceptance-tests
- gamefile-build
- gamefile-core
- gamefile-deploy
- gamefile-web
- .gitignore
- .gitmodules
- infiniti.filters
- pom.xml
- README.markdown**

<https://github.com/tanypv/gol/edit/master/README.markdown>

This repository Search Pull requests Issues Gist

tanypv / gol

Code Issues Pull requests Projects Wiki Pulse Settings

branch: master tmpv init 24d2110 a day ago

1 contributor

32 lines (19 sloc) 2.99 kB

Raw Blame History

This is a simple demonstration application used in the [Jenkins: The Definitive Guide](#) book.

Building the project

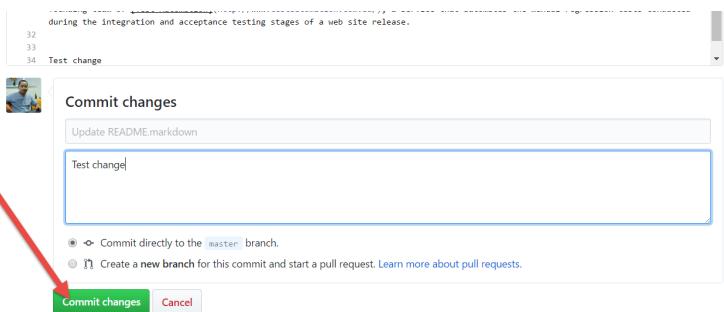
The project is a simple multi-module Maven project. To build the whole project, just run `mvn install` from the root directory.

Running the game

The application is a very simple online version of Conway's 'game of life'. To see what the game does, run `mvn install` as described above, then go to the gamefile-web directory and run `mvn jetty:run`. The application will be running on <http://localhost:9090>.

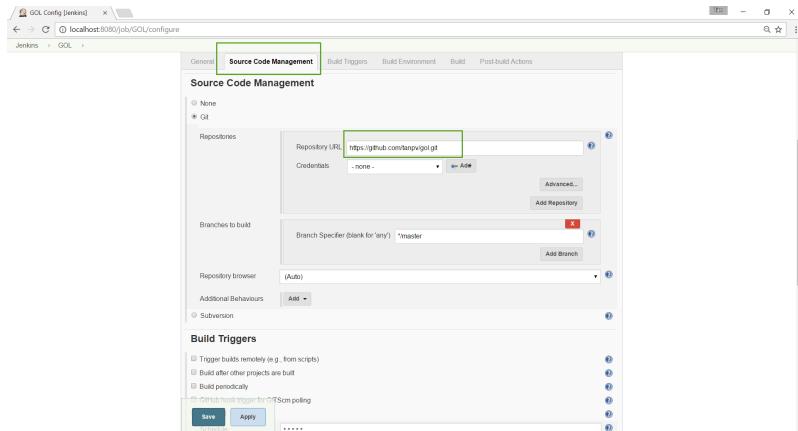
Running the acceptance tests

This documentation hasn't been written yet. Modules and Test modules. These are documented in our [Acceptance Tests](#) page.



Now come back to job dash board, wait for about 1 minute, you will see new job is planning and run.

That it, it work !!!, to practice by yourself, just clone GOL project to your account by **Fork** button and then change the GOL configuration point to your github account as show below.



Build

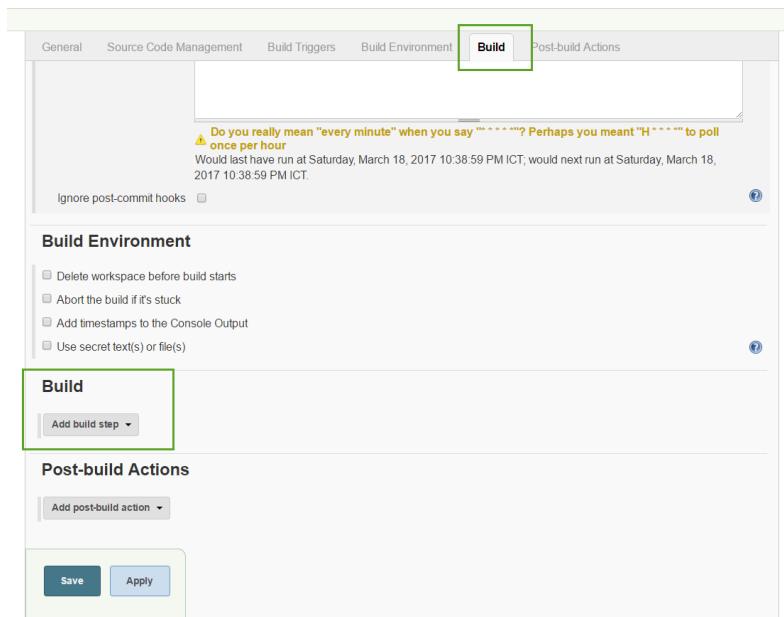
In this section, We will setting to build java web app with maven.

Configure Build Step

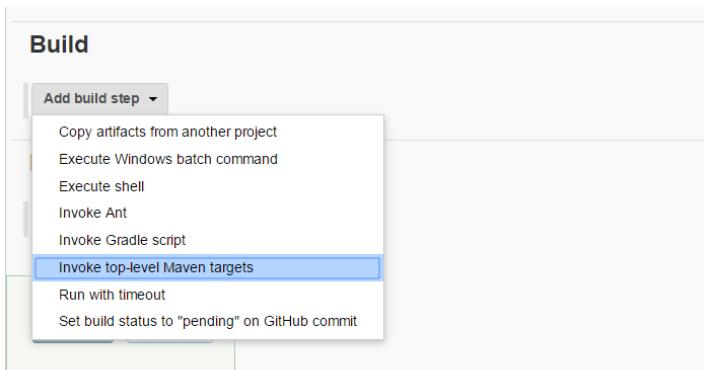
From GOL job dashboard, click to **Configure** link, the configure page will show up

The screenshot shows the Jenkins dashboard for the 'GOL' job. On the left, there's a sidebar with links: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure (which is highlighted with a green border), Git Polling Log, and Move. The main content area is titled 'Project GOL'. It lists the tasks to be performed by the job, which include downloading source code from a GitHub repository and building a Java web application. Below this, there are sections for 'Workspace' and 'Recent Changes', and a 'Permalinks' section with a list of build links. On the far left, there's a 'Build History' panel showing four builds: #4 (Mar 17, 2017 10:09 PM), #3 (Mar 17, 2017 10:07 PM), #2 (Mar 17, 2017 4:54 PM), and #1 (Mar 17, 2017 4:23 PM). At the bottom of the history panel are 'RSS for all' and 'RSS for failures' links.

From configure page, select **Build** tab, page will scroll down to **Add build step**



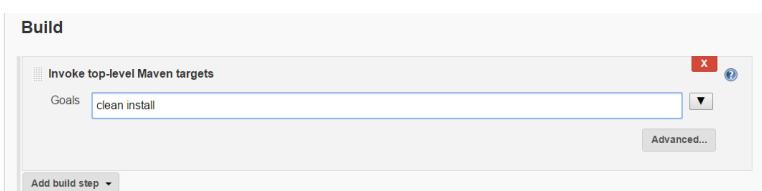
Click to **Add build step** and select **Invoke top-level Maven targets**



Section to adding maven command will show up as below



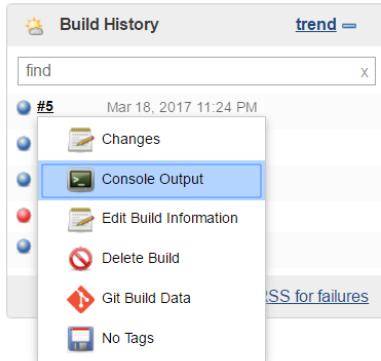
Put **Goals** maven command **clean install**



Then finally click in to **Save** button. That it we already complete configure for build maven step.

Check Log and Workspace

Now from dashboard of GOL, just click to **Build Now** to see how Jenkins job work by open console log



Scroll this log to bottom you will see build job run successfully

```
Results :  
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0  
[INFO] [INFO] ... jacoco-maven-plugin@0.7.2.201409121644:report (jacoco-site) @ gameoflife-web ...  
[INFO] Analyzed bundle 'gameoflife-web' with 2 classes  
[INFO]  
[INFO] ... maven-thucydides-plugin@0.9.268:aggregate (thucydides-reports) @ gameoflife-web ...  
log4j:WARN No appenders could be found for logger (org.jboss.logging).  
log4j:WARN Please initialize the log4j system properly.  
[INFO] Reading requirements from net.thucydides.core.requirements.FileSystemRequirementsTagProvider@2b6adda7  
[INFO] Requirements found:  
[INFO] Requirements found:  
[INFO] Generating release reports for: []  
[INFO] ... maven-install-plugin@2.4.1:install [default-install] @ gameoflife-war ...  
[INFO] Installing C:\Users\TAN\jenkins\workspace\GOL\gameoflife-web\target\gameoflife-war.war  
C:\Windows\system2\config\systemprofile\.m2\repository\com\vaakale\gameoflife\gameoflife-war\1.0-SNAPSHOT\gameoflife-war-1.0-SNAPSHOT.war  
[INFO] Installing C:\Users\TAN\jenkins\workspace\GOL\gameoflife-web\pom.xml  
C:\Windows\system2\config\systemprofile\.m2\repository\com\vaakale\gameoflife\gameoflife-war\1.0-SNAPSHOT\gameoflife-war-1.0-SNAPSHOT.pom  
[INFO] -----  
[INFO] Reactor Summary:  
[INFO] gameoflife ..... SUCCESS [ 10.519 s]  
[INFO] gameoflife-build ..... SUCCESS [ 4.644 s]  
[INFO] gameoflife-core ..... SUCCESS [ 7.861 s]  
[INFO] gameoflife-web ..... SUCCESS [ 14.089 s]  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 38.349 s  
[INFO] Finished at: 2017-03-18T23:25:21+07:00  
[INFO] Final Memory: 38M/249M  
[INFO] -----  
Finished: SUCCESS
```

From GOL home page, click to **Workspace**



Jenkins > GOL >

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Git Polling Log

Move

You will access directory where build happen and actually could see the war file inside **Gameoflife-web/target/**. You could actually download war file from here.



Jenkins > GOL >

Back to Dashboard

Status

Changes

Workspace

Build Now

Delete Project

Configure

Git Polling Log

Move

Workspace of GOL on Windows_Agent

gameoflife-web / target /

classes
gameoflife
generated-sources/annotations
generated-test-sources/test-annotations
maven-archiver
maven-status/maven-compiler-plugin
site
surefire
surefire-reports
test-classes/com/wakaleo/gameoflife/webtests/controllers

gameoflife.war 3.04 MB [view](#)

jacoco.exec 17.93 KB [view](#)

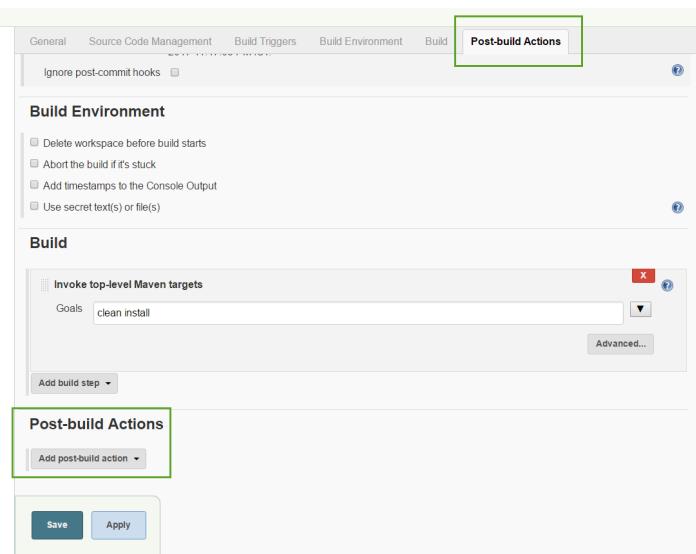
(all files in zip)

That it, We already finish setting up build step.

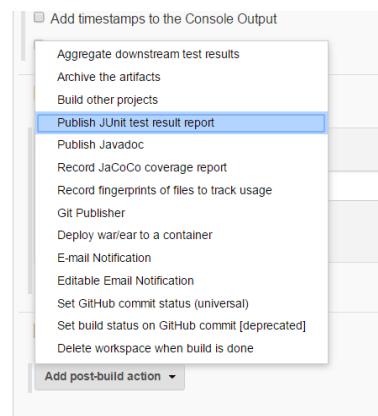
Continuous Inspection with Jenkins

Job Configure for Test Report

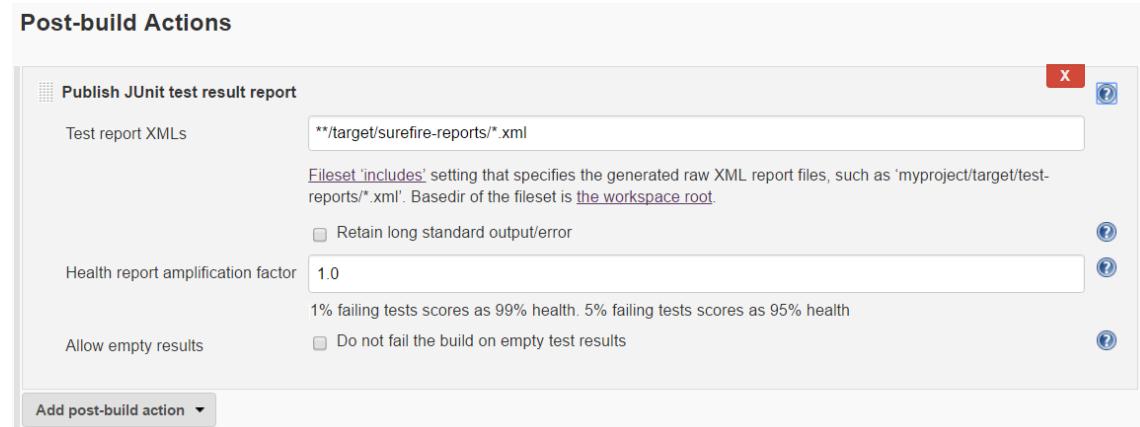
From job dashboard, click to **Configure**. Because showing unit test result on dashboard is done after job build successfully, so we click to tab **Post-build Actions**.



Click to **Add post-build action**, from dropdown menu, select **Publish JUnit test result report**



Input path to maven test result ****/target/surefire-reports/*.xml**



Click **Save** button to finish configure

Test Report on Job Dashboard

From job dashboard, click to **Build Now**. After job run successfully, you will see a link lead to test result. Click to **Latest Test Result**

The screenshot shows the Jenkins interface for the 'Project GOL' job. On the left, there's a sidebar with links like 'Status', 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure', 'Git Polling Log', 'Move', and 'RSS for all' and 'RSS for failures'. The main area is titled 'Project GOL' and contains a list of recent builds. The 'Latest Test Result (no failures)' link is highlighted with a red box. Below it, a 'Permalinks' section lists several build links.

The test report show up, from here you can see many statistic related to test.

This screenshot shows the 'Test Result' page for build #6. It displays a summary bar with 0 failures and 1 skip, and a total of 60 tests took 1.1 sec. Below this, there's a table for 'All Tests' showing test results by package. A chart titled 'Test Result Trend' is also present.

Continue run second time, you will see a chart on job dash board which show up how trending in test result or changing from build to build.

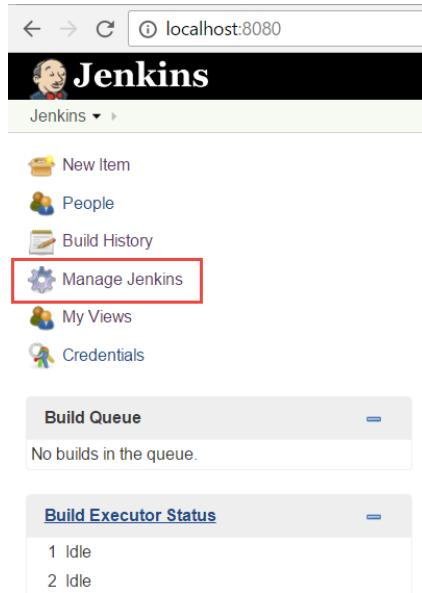
This screenshot shows the Jenkins dashboard for 'Project GOL'. It features the 'Test Result Trend' chart from the previous screen. The chart shows a single data series with values ranging from 0 to 60. Below the chart, there's a list of permalinks for various builds.

Continuous Delivery with Jenkins

This session will show you how to deploy web application (.war file) to a Tomcat web server.

Install “Deploy to Container Plugin”

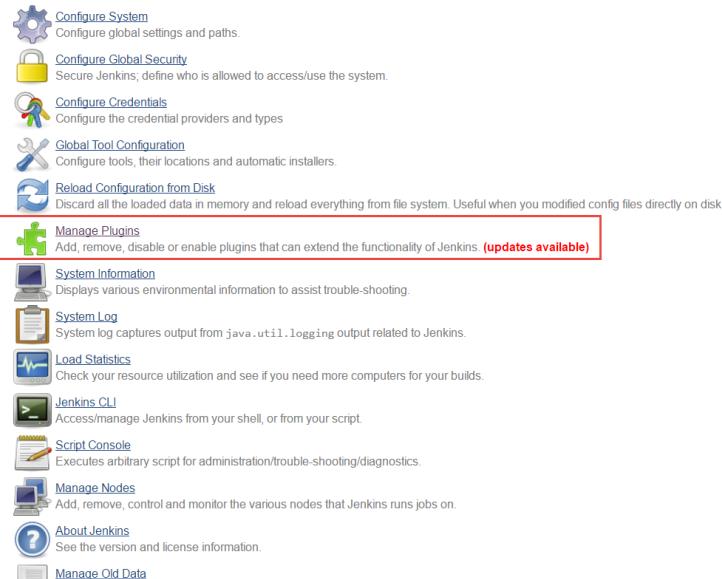
In order to deploy war file in Tomcat server, we need to install “Deploy to Container Plugin”. From Jenkins home page, click to **Manage Jenkins**



The screenshot shows the Jenkins home page at localhost:8080. The navigation bar includes links for New Item, People, Build History, Manage Jenkins (which is highlighted with a red box), My Views, and Credentials. Below the navigation bar are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The 'Manage Jenkins' link is located under the 'Manage Jenkins' heading.

Then click to **Manage Plugin**

Manage Jenkins



The screenshot shows the 'Manage Jenkins' page with various configuration options. The 'Manage Plugins' link (Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (updates available)) is highlighted with a red box. Other links include Configure System, Configure Global Security, Configure Credentials, Global Tool Configuration, Reload Configuration from Disk, System Information, System Log, Load Statistics, Jenkins CLI, Script Console, Manage Nodes, About Jenkins, and Manage Old Data.

Click to tab **Available**, typing “Deploy to Container Plugin”, then click to **Install without restart** button

The screenshot shows the Jenkins Plugin Manager interface. The title bar says "Update Center [Desired] < localhost:8080/pluginManager/available". The left sidebar has links for Back to Dashboard, Manage Jenkins, and Update Center. The main area has tabs for Updates, Available, Installed, and Advanced. The Available tab is selected, showing a list of available plugins. One plugin, "Deploy to container Plugin", is highlighted with a green border. Below the list are buttons for "Install without restart", "Download now and install after restart", and "Check now". A status message at the bottom says "Update information obtained 43 sec ago".

After install you will see it show up inside **Installed** tab

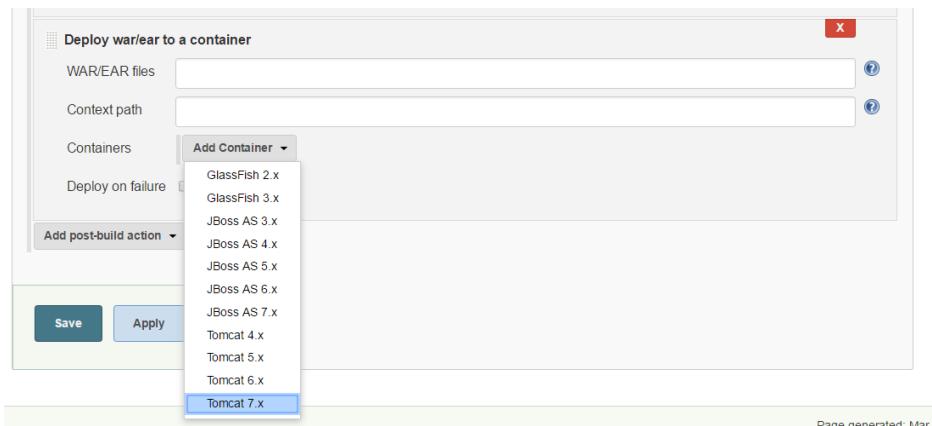
The screenshot shows the Jenkins Plugin Manager interface with the "Installed" tab selected. The title bar says "Update Center [Desired] < localhost:8080/pluginManager/installed". The left sidebar has links for Back to Dashboard, Manage Jenkins, and Update Center. The main area shows a list of installed plugins. The "Deploy to container Plugin" is listed under the "Enabled" section. It has a version of 1.10 and a "Uninstall" button. Other plugins listed include "bouncycastle API plugin", "Matrix Authorization Strategy Plugin", "Matrix Project Plugin", "OWASP Mercurie Formatter Plugin", and "Windows Slaves Plugin".

Add Deploy Step

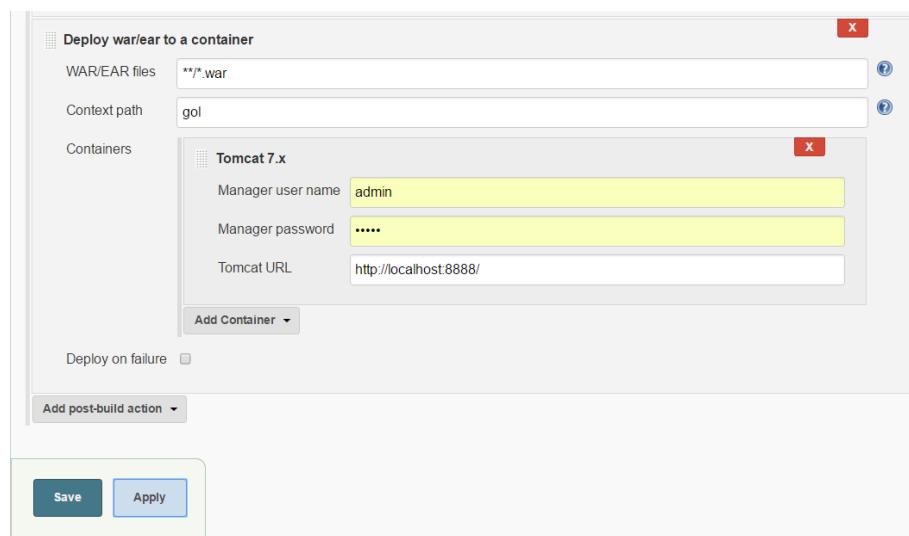
Open GOL job configuration, click to “Post-build Actions” tab, click to “Add post-build action”, and select item “Deploy war/ear to a container” from context menu

The screenshot shows the Jenkins Job configuration interface. The top navigation bar has tabs for General, Source Code Management, Build Triggers, Build Environment, Build, and Post-build Actions. The Post-build Actions tab is selected. In the main area, there is a panel titled "Invoke top-level Maven targets" with a "Goals" field containing "clean install". Below this is a "Add build step" dropdown menu. The "Deploy war/ear to a container" option is highlighted with a blue selection bar. To the right of the dropdown, there is a detailed description of the "Deploy war/ear to a container" step. At the bottom of the page are "Save" and "Apply" buttons.

Select “Tomcat 7.x” from **Containers**



Input path to **war file**, input **context path**, and finally input **admin user/password** which already created at step install Tomcat server.



Finally click to **Save** button.

Note that to control Tomcat from Jenkins, one user with role "manager-script" should be added to file **conf/tomcat-users.xml**. Then restart Tomcat server by service.

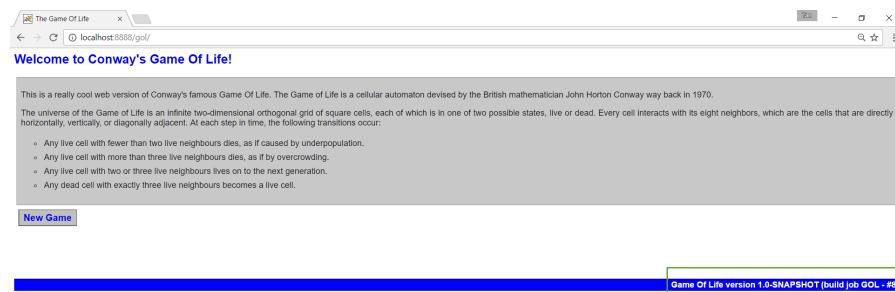
```
tomcat-users.xml [3]
1 <?xml version='1.0' encoding='cp1252'?>
2 <!--
3 Licensed to the Apache Software Foundation (ASF) under one or more
4 contributor license agreements. See the NOTICE file distributed with
5 this work for additional information regarding copyright ownership.
6 The ASF licenses this file to You under the Apache License, Version 2.0
7 (the "License"); you may not use this file except in compliance with
8 the License. You may obtain a copy of the License at
9
10 http://www.apache.org/licenses/LICENSE-2.0
11
12 Unless required by applicable law or agreed to in writing, software
13 distributed under the License is distributed on an "AS IS" BASIS,
14 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
15 See the License for the specific language governing permissions and
16 limitations under the License.
17 -->
18
19 <tomcat-users>
20   <role rolename="manager-script"/>
21   <user username="admin" password="admin" roles="manager-script"/>
22 </tomcat-users>
```

Check if Deploy Step Run Successfully

From GOL job dashboard, click into **Build Now**, wait some time until job complete and you will see consolog for step deploy GOL to Tomcat server

```
Recording test results
Deploying C:\Users\TAN\jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war to container Tomcat 7.x Remote
[C:\Users\TAN\jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war] is not deployed. Doing a fresh deployment.
Deploying [C:\Users\TAN\jenkins\workspace\GOL\gameoflife-web\target\gameoflife.war]
Finished: SUCCESS
```

And now you try to access GOL web app at <http://localhost:8888/gol/>

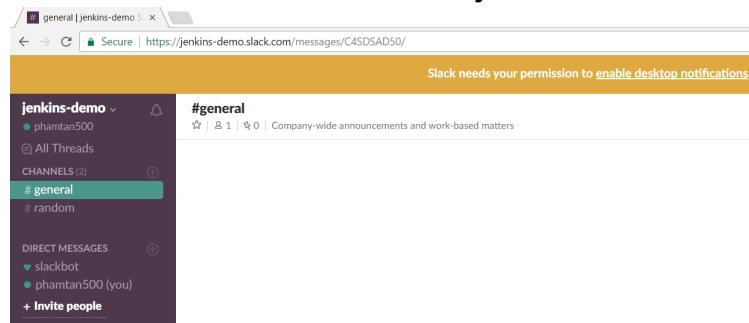


That it, deployment step working fine now. And the great thing is you know you are testing on what build on Jenkins.

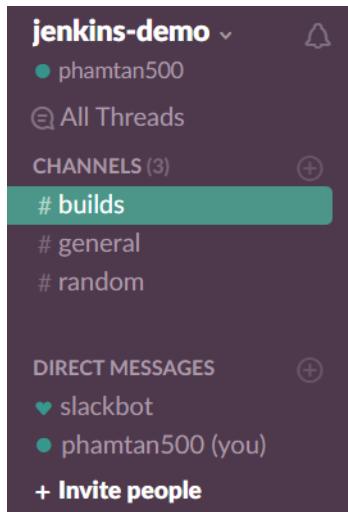
Continuous Monitoring with Jenkins

Create Slack Team and Install Jenkins CI App

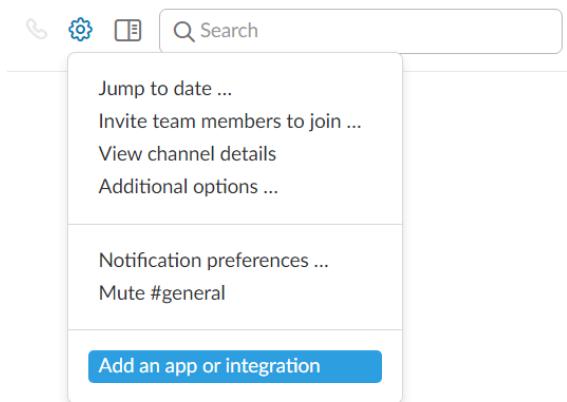
From Slack, create a new team call **jenkins-demo**



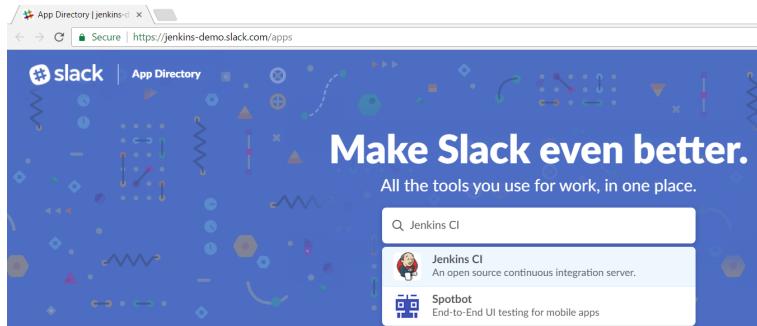
Create a new channel call **#build**, this channel will be used to notify message from Jenkins



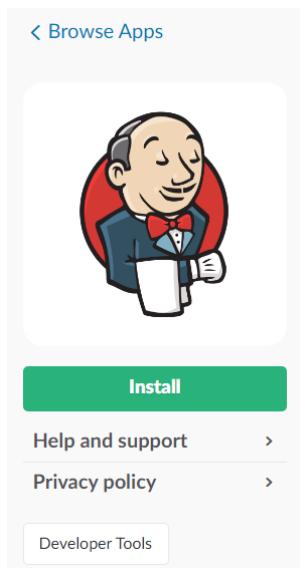
From **jenkins-demo** home page, click to **Add an app or integration**



Search for **Jenkins CI** application



Click to **Install** button



Select **#build** and click **Add Jenkins CI integration**

Jenkins CI
An open source continuous integration server.
Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs.
This integration will post build notifications to a channel in Slack.

Post to Channel
Start by choosing a channel where Jenkins notifications will be posted. #builds or [create a new channel](#)

Add Jenkins CI integration

Copy information related to team domain and token, this information will be used to configure Jenkins job later

- Step 3** After it's installed, click on Manage Jenkins again in the left navigation, and then go to Configure System. Find the Global Slack Notifier Settings section and add the following values:
- Team Domain: `jenkins-demo`
 - Integration Token: `Axe8dxyzI8jdMSde16FSmIw`

The other fields are optional. See the help text by clicking the question mark icon next to the fields for more information. Press the Save button when you're done.

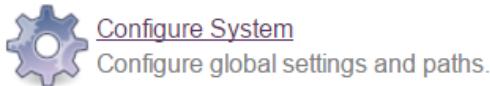
Global Slack Notifier Settings	
Team Domain	my-team
Integration Token	JEyqMFPKa5yUDKV9inCUaJDH
Channel	
Build Server URL	/

Install Slack Notification Plugin to Jenkins

Go to **Plugin Manager** and install **Slack Notification** plugin

The screenshot shows the Jenkins plugin manager interface. At the top, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A search bar at the top right contains the text 'slack notification'. Below the tabs, a table lists available plugins. One plugin is highlighted: 'Slack Notification Plugin' by 'A Build status publisher that notifies channels on a Slack team'. The table includes columns for 'Name' and 'Version' (2.2). At the bottom of the table, there are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'. A note below the table states 'Update information obtained: 1 hr 44 min ago'.

Go to **Manage Jenkins**, click to **Configure System**



Search for section **Global Slack Notifier Settings**, and put in information for **Team Subdomain** and **Integration Token** (This info get from previous step), then click to **Test Connection**, you will see connection successfully. Finally click to **Save** button.

This screenshot shows the 'Global Slack Notifier Settings' configuration page. It includes fields for 'Base URL', 'Team Subdomain' (set to 'jenkins-demo'), 'Integration Token' (set to 'Ax6dxyzl8JdMSde6FSmWwe'), and 'Integration Token Credential ID' (set to '-none-'). A warning message '⚠ Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID' is displayed above the credential dropdown. There are also fields for 'Is Bot User?' and 'Channel'. At the bottom are 'Success' and 'Test Connection' buttons.

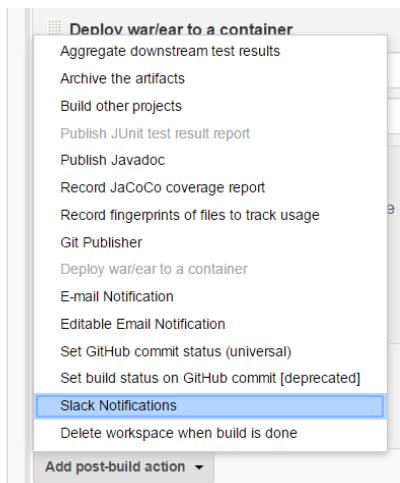
That it, now Slack notification is ready used inside Jenkins job.

Configure GOL Using Slack Notification

From Jenkins home page, click to **Configure** of GOL job

This screenshot shows the Jenkins home page with a list of jobs. One job, 'GOL', has its configuration context menu open. The menu items include 'Changes', 'Workspace', 'Build Now', 'Delete Project', 'Configure' (which is highlighted with a blue background), 'Git Polling Log', and 'Move'. Below the menu, there is a note 'Icon: S M L'.

From **Configure** page, click to tab **Post-build Actions**, then add a **Slack Notifications** step.



Select when the notify will be send

The configuration screen for Slack Notifications. It lists eight notification types with checkboxes:

- Notify Build Start (checked)
- Notify Aborted (unchecked)
- Notify Failure (checked)
- Notify Not Built (unchecked)
- Notify Success (checked)
- Notify Unstable (unchecked)
- Notify Regression (unchecked)
- Notify Back To Normal (unchecked)

Click to **Advanced** and put information for **Team Subdomain** and **Integration Token**. Finally click to **Save** button.

The advanced configuration screen for Slack Notifications. It includes fields for:

- Notification message includes: dropdown set to "nothing about commits".
- Base URL: input field containing "jenkins-demo".
- Team Subdomain: input field containing "jenkins-demo".
- Integration Token: input field containing "Axe8dxyzl8JdMSdel6FSmWwe". A warning message "⚠️ Exposing your Integration Token is a security risk. Please use the Integration Token Credential ID" is displayed above the token field.
- Integration Token Credential ID: dropdown set to "- none -".
- Is Bot User?: checkbox (unchecked).
- Project Channel: input field.
- Test Connection: button.

Check if Notification Work

From GOL home page click to **Build Now**



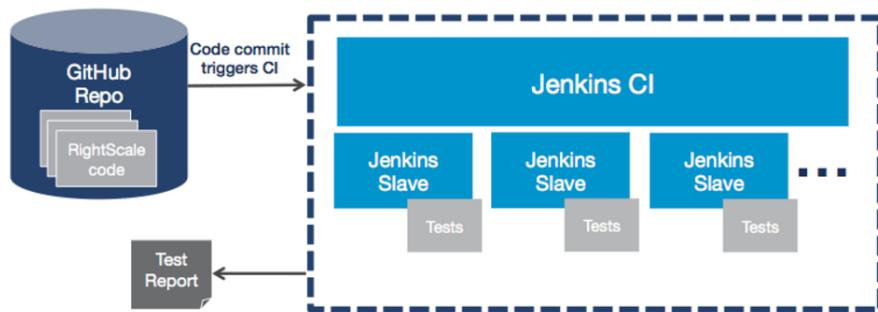
And notification message will be send to **#build** channel as expected

```
jenkins APP 11:52 AM
| Slack/Jenkins plugin: you're all set on http://localhost:8080/
jenkins APP 12:03 PM
| GOL - #9 Started by user tan (Open)
| GOL - #9 Success after 1 min 6 sec (Open)
```

That it, from now on, GOL build and deploy job will be automatically watched by Slack Notification.

Distributed Build System with Jenkins

Jenkins Architecture



Normally, job will do not run on machine which install Jenkins but running on slave machines and these slaves machine are controlled by master machine where Jenkins is installed.

Windows Slave Agent

Enable Launch Slave Agents via Java Web Start

From Jenkins home page, click to **Manage Jenkins**

The screenshot shows the Jenkins home page at localhost:8080. The navigation bar includes links for New Item, People, Build History, Manage Jenkins (which is highlighted with a green border), My Views, and Credentials. Below the navigation bar, there are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The 'Manage Jenkins' link is the target of the instruction.

Then click to **Configure Global Security**

Manage Jenkins

The screenshot shows the 'Manage Jenkins' page. It lists three options: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon, which is highlighted with a green border), and 'Configure Credentials' (key icon). The 'Configure Global Security' link is the target of the instruction.

Select **Random** for option **TCP port for JNLP agents**

The screenshot shows the 'Configure Global Security' page at localhost:8080/configureSecurity/. It features a lock icon and the title 'Configure Global Security'. Below that, there is a section for 'TCP port for JNLP agents' with three radio button options: 'Fixed' (disabled), 'Random' (selected and highlighted with a green border), and 'Disable'.

Then finally click **Save** button

Add New Windows Slave

From Jenkins home page, click to **Manage Jenkins**

The screenshot shows the Jenkins dashboard. The sidebar on the left includes links for New Item, People, Build History, Manage Jenkins (which is highlighted with a green box), My Views, and Credentials. Below the sidebar are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is currently empty.

On **Manage Jenkins** page, scroll down and click to **Manage Nodes**

The screenshot shows the 'Manage Jenkins' page. The sidebar on the left includes links for New Item, People, Build History, Manage Jenkins (highlighted with a green box), My Views, and Credentials. Below the sidebar are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area lists various Jenkins management options, with 'Manage Nodes' highlighted with a green box.

From there you can see all computer currently exist in system. We just have one master machine. To create new node, click to **New Node**

The screenshot shows the 'Nodes' page. The sidebar on the left includes links for Back to Dashboard, Manage Jenkins, New Node (highlighted with a green box), and Configure. Below the sidebar are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is a table showing one node named 'master'.

Input agent name and click to **OK** button

The screenshot shows the 'New Node' configuration dialog. It has a 'Node name' field containing 'Windows_Agent' and an 'OK' button highlighted with a green box. There is also a note about permanent agents below the input field.

Input **Remote root directory**, select **Launch agent via Java Web Start**, Finally click to **Save** button

Jenkins > Nodes > Windows_Agent

Name: Windows_Agent

Description:

of executors: 1

Remote root directory: C:\jenkins_slave

Labels:

Usage: Use this node as much as possible

Launch method: Launch agent via Java Web Start

Availability: Keep this agent online as much as possible

Node Properties:

- Environment variables
- Tool Locations

Save

Launch Slave Agent

From computer which we want to install as slave (for demo, I use same machine for both master and slave) access to link <http://localhost:8080/computer/>

You could see that currently Window_Agent is disconnected with Jenkins.

S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
1	master	Windows 10 (amd64)	In sync	419.23 GB	10.46 GB	419.23 GB	0ms
	Windows_Agent		N/A	N/A	N/A	N/A	Time out for last 1 try

Data obtained: 4 min 56 sec | Refresh status

Build Queue: No builds in the queue

Build Executor Status: 1 idle, 2 idle

Windows_Agent (offline)

Click to **Windows_Agent** and then click to **Launch** button

Windows_Agent [jenkins] > Jenkins > Nodes > Windows_Agent

Status:

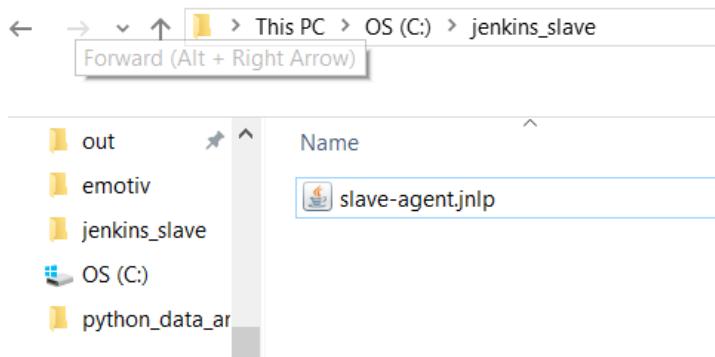
Connect agent to Jenkins one of these ways:

- Launch:** Launch agent from browser
- Run from agent command line

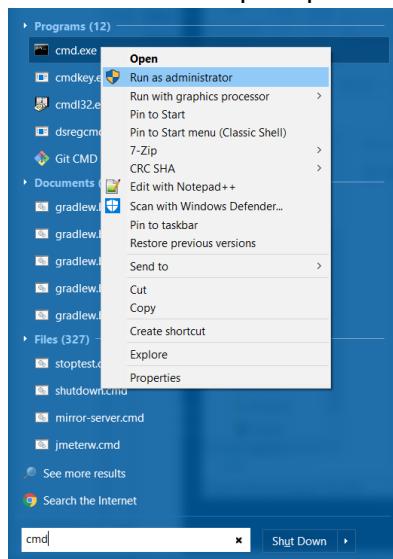
Projects tied to Windows_Agent: None

Mark this node temporarily offline

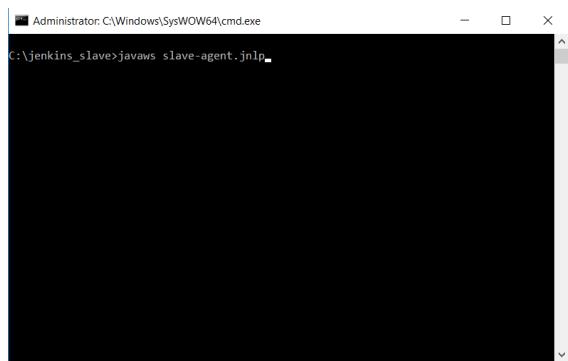
File **slave-agent.jnlp** will be downloaded to local, copy this file to **C:\jenkins_slave**



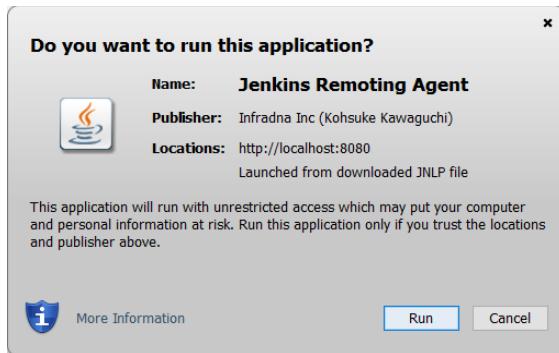
Start a command prompt as administrator.



Change directory to **C:\jenkins_slave** then execute command **javaws slave-agent.jnlp**



A dialog will show up, click to **Run** button

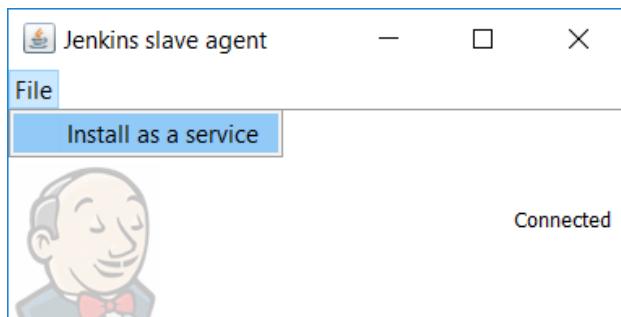


A dialog will show up and message that connected successfully

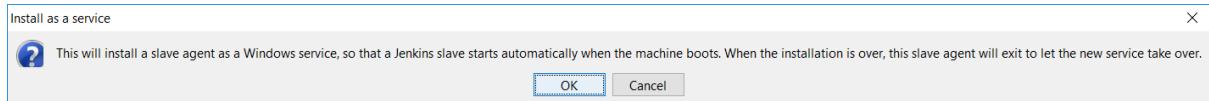


Go back to link <http://localhost:8080/computer/> and refresh, you will see Window_Agent up and running successfully.

Now Window_Agent ready for use. But we will do one more step to setting up slave agent as a Windows service. From **Jenkins slave agent** dialog, click to **File** and then select **Install as a service**



Then click **OK** for question



Current dialog will close up and new service for slave agent is installed successfully.

Name	Status	Startup Type	Log On As
jenkins	Running	Automatic	Local Syst...
Jenkinsslave-C_jenkins_slave	Running	Automatic	Local Syst...
Local Session Manager	Running	Automatic	Local Syst...
Microsoft Office Click-to-Run Service	Running	Automatic	Local Syst...
Network Connection Broker	Running	Manual (Trigger Start)	Local Syst...
Network Connections	Running	Manual	Local Syst...
Network List Service	Running	Manual	Local Servi...
Network Location Awareness	Running	Automatic	Network S...

From now on, everytime machine power up, this machine will automatically connect to jenkins master and ready for running job.

Running Job with Windows Slave Agent

From GOL job dashboard, click to **Configure**

From **General** tab, click to **Restrict where this project can be run**, then enter "Windows_Agent" in **Label Expression**. This mean we only want GOL job running on machine represented by "Windows_Agent". Then finally click to **Save** button.

General

Project name: GOL

Description:

Following task will be done by GOL job :

- * Download source code from Github repository <https://github.com/tanpv/gol>
- * Build web application from java source
- * Run automation test with test script already specify inside project
- * Publish automation test result to Jenkins dashboard
- * Deploy web application to Tomcat server

[Plain text] [Preview]

Discard old builds

GitHub project

Permission to Copy Artifact

This project is parameterized

Throttle builds

Disable this project

Execute concurrent builds if necessary

Restrict where this project can be run

Label Expression: Windows_Agent

Label Windows_Agent is serviced by 1 node

Advanced...

Source Code Management

Save Apply

Now try to run GOL from it's dashboard by click on **Build Now**
 From Jenkins home page, you could see GOL running with **Windows_Agent** machine

Dashboard [Jenkins] x Dashboard [Jenkins]

localhost:8080

Jenkins

Jenkins ▾ ▶

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Credentials

Build Queue

No builds in the queue.

Build Executor Status

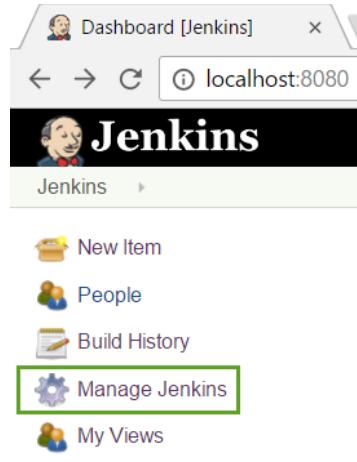
master	Windows_Agent
1 Idle	1 GOL
2 Idle	#9

Ubuntu Slave Agent

Enable Launch Slave Agents via Java Web Start

Add New Ubuntu Slave Agent

From jenkins home page, click to **Manage Jenkins**, then click to **Manage Nodes**



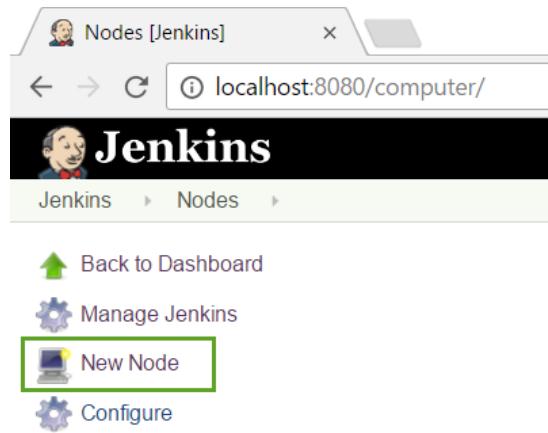
The screenshot shows the Jenkins home page with the following navigation links:

- New Item
- People
- Build History
- Manage Jenkins** (highlighted with a green box)
- My Views
- Credentials

Below the navigation bar, there is a section titled "Manage Nodes" with the following description:

 [Manage Nodes](#)
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

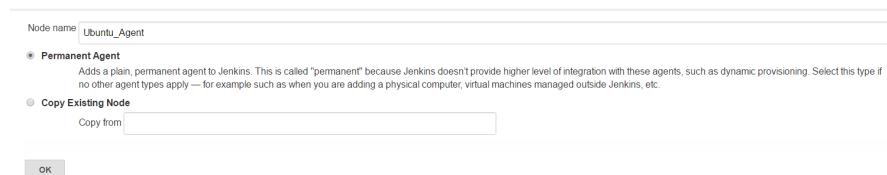
From **Nodes** screen, click to **New Node**



The screenshot shows the "Nodes" screen with the following navigation links:

- Back to Dashboard
- Manage Jenkins
- New Node** (highlighted with a green box)
- Configure

Put the name “Ubuntu_Agent”, click to **Save** button



The screenshot shows the "New Node" configuration dialog with the following fields:

Node name:

Permanent Agent
A plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Copy Existing Node
Copy from:

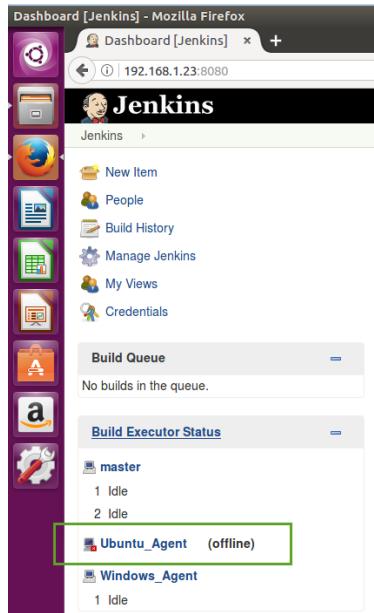
Specify **Remote root directory**, then click **Save** button

The screenshot shows the Jenkins 'Create New Item' page for a new node configuration. The 'Name' field is set to 'Ubuntu_Agent'. The 'Remote root directory' field is highlighted with a green box and contains the value '/home/tan/jenkins_slave'. The 'Save' button at the bottom is also highlighted with a green box.

Name: Ubuntu_Agent
Description:
of executors: 1
Remote root directory: /home/tan/jenkins_slave
Labels:
Usage: Use this node as much as possible
Launch method: Launch agent via Java Web Start
Availability: Keep this agent online as much as possible
Node Properties:
Environment variables
Tool Locations
Save

Launch Slave Agent

Go to Ubuntu machine which you want it become a jenkins agent and click to **Ubuntu_Agent**



Click to **Launch** button

Agent Ubuntu_Agent

Connect agent to Jenkins one of these ways:

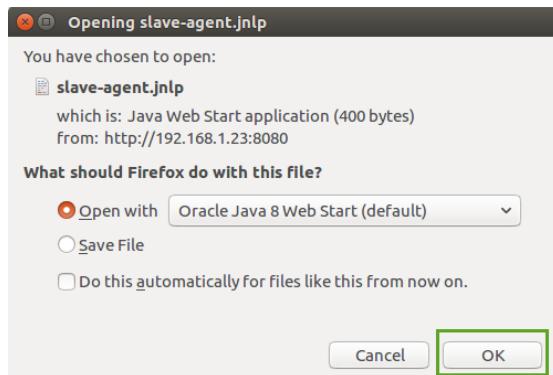
- Launch agent from browser (highlighted)
- Run from agent command line:
java -jar slave.jar -jnlpUrl http://192.168.1.23:8080/computer/Ubuntu_Agent/slave-agent.jnlp -secret 4510c86124503805a21cd54b67bd246a9efb21505426c399af208f3b66a51e

Projects tied to Ubuntu_Agent

None

Mark this node temporarily offline

Click to **OK** button



One dialog will show up that agent and master already connect



Now you will see **Ubuntu_Agent** up and running from Jenkins home page. And you could create job which need to running on Ubuntu machine.

Build Executor Status

Slave	Status
master	1 Idle 2 Idle
Ubuntu_Agent	1 Idle
Windows_Agent	1 Idle

Jenkins Pipeline

What is Jenkins Pipeline ?

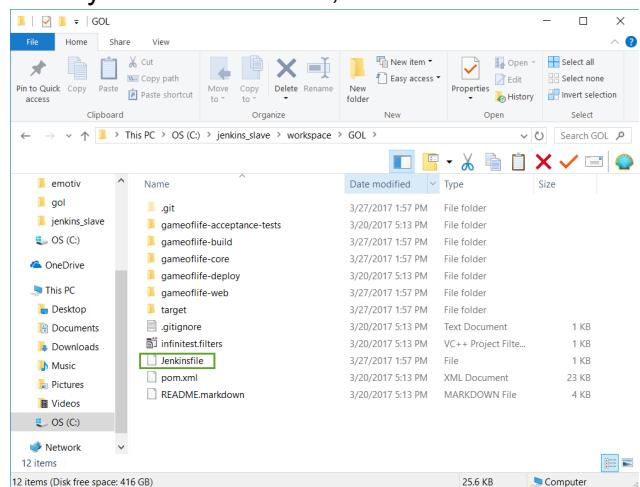
Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the [Pipeline DSL](#).

Following are common steps to use pipeline:

- Create a **Jenkinsfile** which describe what kind of command will run while pipeline running.
- Add this **Jenkinsfile** to the root of project on Github repository. **Jenkinsfile** use Groovy grammar.
- Create a new job with type **Pipeline**
- Run pipeline job

Create a Jenkinsfile

From your local machine, create a new file with name **Jenkinsfile**



Put following code into **Jenkinsfile**, this code basically will run maven command to build web application.

```
// start of pipeline
pipeline {
    // where pipeline job will run
    agent any
    // start of stages : build, test, deploy ...
    stages {
        // start of stage : build
        stage('build') {
            // start of running steps inside one stage
            steps {
                // invoke command to build with maven
```

```
        bat 'mvn clean install'
    }
}
}
}
```

Access to GOL github repo from browser, click to **Upload files**, then drag drop **Jenkinsfile** which already created. Finally click to **Commit changes**

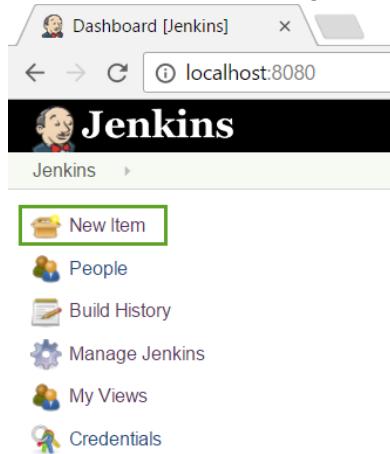
The screenshot shows a GitHub repository page for 'tanpv/gol'. At the top, there's a Jenkinsfile commit with several commits listed under it. Below the repository header, there's a 'Create new file' button and a 'Upload files' button highlighted with a green border. A file upload dialog box is open, prompting 'Drag additional files here to add them to your repository' or 'Or choose your files'. At the bottom of the dialog, there's a 'Commit changes' button.

Wait some time and you will see the file show up inside GOL repository

The screenshot shows the same GitHub repository page for 'tanpv/gol'. The Jenkinsfile commit is now visible in the commit list, along with other files like .gitignore, README.markdown, infinitest.filters, and pom.xml. The Jenkinsfile commit has a timestamp of '3 minutes ago'.

Create Pipeline Job

From Jenkins home page click to **New Item**



Typing job name **GOL_Pipeline**, select **Pipeline** job style, and click to **Save** button

A screenshot of the 'Enter an item name' dialog. The input field contains 'GOL_Pipeline'. Below the input field is a list of job types: 'Freestyle project', 'Pipeline', 'External Job', 'Multi-configuration project', 'Folder', 'GitHub Organization', and 'Multibranch Pipeline'. The 'Pipeline' option is highlighted with a green border. At the bottom right of the dialog is a blue 'OK' button.

From job dashboard **GOL_Pipeline**, click to **Configure**

The screenshot shows the Jenkins interface for the 'Pipeline GOL_Pipeline' job. On the left, a sidebar contains links like 'Back to Dashboard', 'Status', 'Changes', 'Build Now', 'Delete Pipeline', 'Configure' (which is highlighted in green), 'Move', 'Full Stage View', and 'Pipeline Syntax'. The main area is titled 'Pipeline GOL_Pipeline' and displays the message 'No data available. This Pipeline has not yet run.' Below this, there are sections for 'Build History' (with a search bar) and 'Permalinks' (including RSS feeds for all and failures).

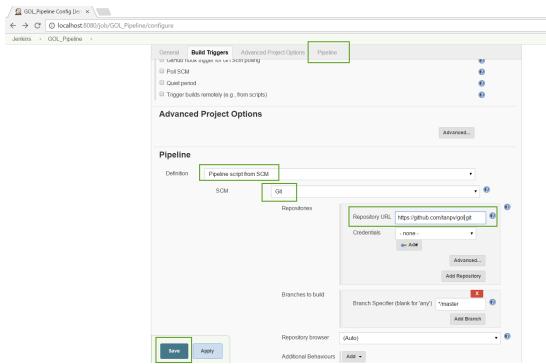
Select **Build Triggers** tab, select **Post SCM** check box, then put in following string to check change every minute on github “* * * * *

The screenshot shows the 'Build Triggers' configuration for the 'Pipeline GOL_Pipeline' job. The 'Post SCM' checkbox is selected, and a note below it says: 'Do you really mean "every minute" when you say "*****"? Perhaps you meant "H*****" to poll once per hour.' Other trigger options shown include 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GitHub polling', 'Schedule' (with a dropdown menu showing '*****'), 'Ignore post commit hooks', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'.

Select **Pipeline** tab, then select **Pipeline script from SCM**

The screenshot shows the 'Pipeline' configuration for the 'Pipeline GOL_Pipeline' job. In the 'Definition' dropdown, 'Pipeline script from SCM' is selected, while 'Pipeline script' and 'Pipeline as code' are also listed. There is an 'Advanced...' button at the top right of the Pipeline section. At the bottom, there are 'Save' and 'Apply' buttons.

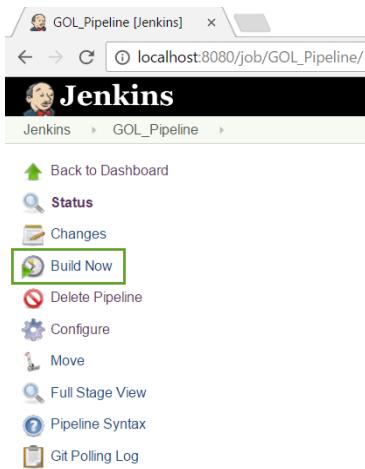
From **SCM** dropdown, select **Git**, then put in GOL project repository url <https://github.com/tanpv/gol.git> , finally click to **Save** button.



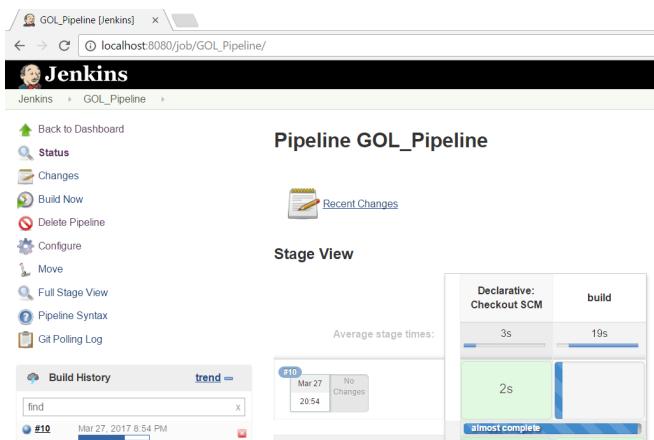
That it, we already complete create the first pipeline job with Jenkins!

Build with Pipeline

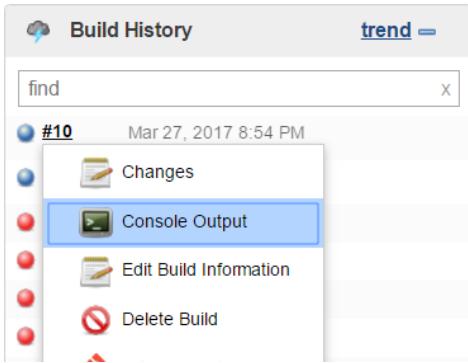
From GOL dashboard, click to **Build Now**



You will see job run with quite nice progress show up



Click to **Console Log**, build log will show up with build success for GOL project.



```
[INFO] -----
[INFO] Reactor Summary:
[INFO]
[INFO] gameoflife ..... SUCCESS [ 2.677 s]
[INFO] gameoflife-build ..... SUCCESS [ 1.581 s]
[INFO] gameoflife-core ..... SUCCESS [ 5.542 s]
[INFO] gameoflife-web ..... SUCCESS [ 5.501 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 15.575 s
[INFO] Finished at: 2017-03-27T20:55:16+07:00
[INFO] Final Memory: 39M/441M
[INFO] -----
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Running Pipeline On Specific Agent

Edit **Jenkinsfile** on Github with following content, basically this code will force pipeline to only running on Windows_Agent

```
// start of pipeline
pipeline {
    // where pipeline job will run
    agent {
        // force pipeline job to running on windows_agent
        label "Windows_Agent"
    }
    // start of stages : build, test, deploy ...
    stages {
        // start of stage : build
        stage('build') {
            // start of running steps inside one stage
            steps {
                // invoke command to build with maven
                bat 'mvn clean install'
            }
        }
    }
}
```

Click to **Commit changes**

Commit changes

Update Jenkinsfile

Force pipeline running on Windows_Agent

Commit directly to the `master` branch.

Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes **Cancel**

You will see GOL_Pipeline automatically run due to it saw code change from Github

localhost:8080/job/GOL_Pipeline/

Jenkins

GOL_Pipeline

Back to Dashboard Status Changes Build Now Delete Pipeline Configure Move Full Stage View Pipeline Syntax Git Polling Log

Pipeline GOL_Pipeline

Recent Changes

Stage View

Average stage times:

Declarative: Checkout SCM	build
3s	23s

Build History trend =

#11	Mar 29	No Changes	12s
#10	Mar 27	No	

And from **Console Output**, you can see that GOL_Pipeline job is running with "Windows_Agent"

Console Output

```
Started by an SCM change
Obtained Jenkinsfile from git https://github.com/tanpv/gol.git
[Pipeline] node
Still waiting to schedule task
Waiting for next available executor on Windows_Agent
Running on Windows_Agent in C:\jenkins_slave\workspace\GOL_Pipeline
```

Deploy to Tomcat with Pipeline

Edit **Jenkinsfile** on GOL repository with following code, this code add deployment to Tomcat server step.

```
// start of pipeline
pipeline {
```

```

// where pipeline job will run
agent {
    // force pipeline job to running on windows_agent
    label "Windows_Agent"
}

// start of stages : build, test, deploy ...
stages {

    // start of build stage
    stage('build') {
        // define step to run
        steps {
            // invoke command to build with maven
            bat 'mvn clean install'
        }
    }

    // start of deploy state
    stage('deploy') {
        // define step to run
        steps {
            //invoke command to stop tomcat service
            bat 'sc stop Tomcat7'
            bat 'ping 127.0.0.1 -n 6'
            // copy war file from build target to webapp Tomcat folder
            bat 'xcopy /y
C:\\jenkins_slave\\workspace\\GOL_Pipeline\\gameoflife-
web\\target\\gameoflife.war "C:\\Program Files\\Apache Software
Foundation\\Tomcat 7.0\\webapps"'
            //invoke command to start tomcat service
            bat 'sc start Tomcat7'
        }
    }
}

```

Commit change of **Jenkinsfile** to Github right from browser

Commit changes

Update Jenkinsfile

Add an optional extended description...

Commit directly to the `master` branch.
 Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit changes **Cancel**

Wait for about 1 minute and GOL_Pipeline job will automatically build

The screenshot shows the Jenkins Pipeline GOL_Pipeline dashboard. On the left, there's a sidebar with various Jenkins management links like Back to Dashboard, Status, Changes, Build Now, Delete Pipeline, Configure, Move, Full Stage View, Pipeline Syntax, and Git Polling Log. The main area is titled "Pipeline GOL_Pipeline". It features a "Stage View" grid with three columns: Declarative: Checkout SCM, build, and deploy. Below the grid, two builds are listed: #30 (Mar 29, 06:07, No Changes, 20s) and #29 (Mar 29, 06:03, 1 commits, 3s, 16s, 6s). To the left of the grid, there's a "Build History" section showing a single entry for build #29 on Mar 29, 2017 at 6:07 AM.

From **Console Output** will see deploy to Tomcat7 step

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (deploy)
[Pipeline] bat
[GOL_Pipeline] Running batch script
```

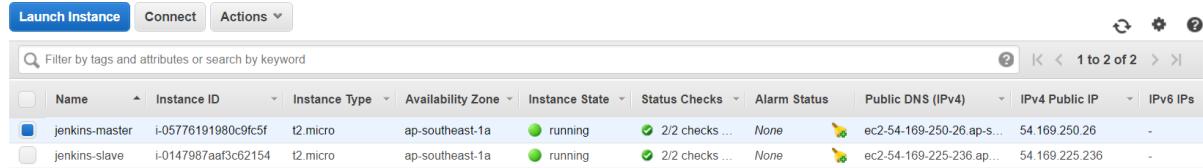
Check the site and you will see it work fine at <http://localhost:8888/gol>

The screenshot shows a web browser window titled "The Game Of Life". The address bar shows "localhost:8888/gol/". The page content is titled "Welcome to Conway's Game Of Life!". It contains a brief introduction about the Game of Life, the rules, and a "New Game" button. At the bottom, there's a footer note: "Game Of Life version 1.0-SNAPSHOT (build job GOL - #28)".

Note for auto build trigger after commit

Jenkins in the Cloud

From AWS console create 2 Ubuntu machine, one machine will be Jenkins master and one machine will be Jenkins slave

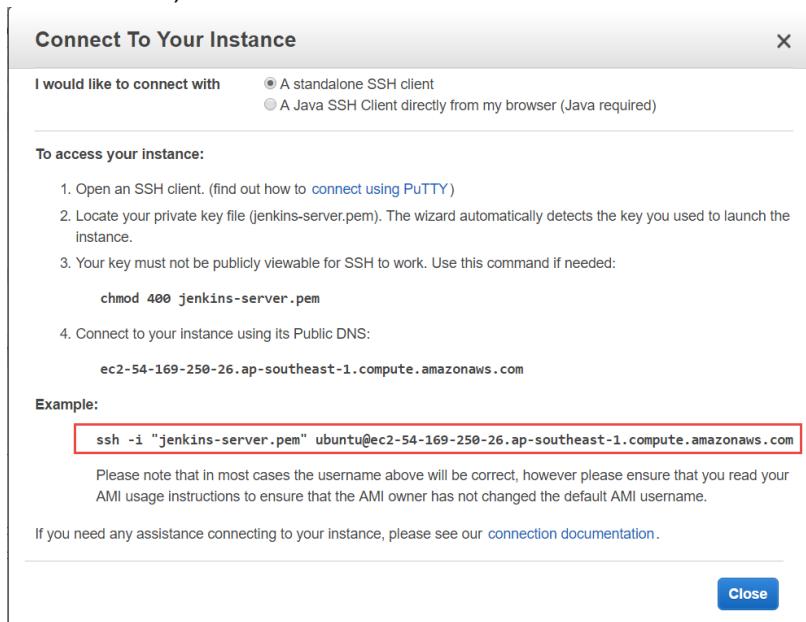


Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs
jenkins-master	i-05776191980c9fc5f	t2.micro	ap-southeast-1a	running	2/2 checks ...	None	ec2-54-169-250-26.ap-s...	54.169.250.26	-
jenkins-slave	i-0147987aaf3c62154	t2.micro	ap-southeast-1a	running	2/2 checks ...	None	ec2-54-169-225-236.ap...	54.169.225.236	-

Create a Key to Connect between Master and Slave

The purpose is master should successfully connect to slave with ssh command.

Using ssh and key pair file in order to connect to master machine (I already install ssh on my local window).



Create a publish key from master machine with command

ssh-keygen -t rsa the private key will be stored at **/home/ubuntu/.ssh/id_rsa**

```
ssh -i "jenkins-server.pem" ubuntu@ec2-54-169-250-26.ap-southeast-1.compute.amazonaws.com
ubuntu@ip-172-31-16-113:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
```

Store the private key to a file for later use

```

1 -----BEGIN RSA PRIVATE KEY-----
2 MIIEpAIBAAQCAQEAYBz1wbtt1lV1tLcs2nupKVuDkHEgGrtrbGdSd2gUD0wiA47m
3 1wy5N9jEzzVOREvhSJdRar85dQalwFJQswjeWge8M8uSzYCQWsCe/K17+TbirwK6
4 L3po0peEuAamAEqFUHysHJTUL310X5RCJ/pifBEIizZedFC4OjznDxBGdJ1SxMyf
5 dwSUm4mY+sSPOXzfhl1fKx3z1jIb6OiNEeUo07rdVYsD4iLW1sTPPTqtZuoXCRnz
6 ieCBz8V8cGuYBhzPM30LhB48dlmyP1su07WObg+08rIV4XsJ1q5Z/aEUKBgFgwm
7 Fzu9p5Rs+abR8skWxp+pW3ufoP3TRw9BVtf/oQIDAQABaoIBAHFe/thgEYc4tChP
8 SokJMCAvKY1GmeKrNcP3kV/2xg6YmlbuCozmieKlo2x7IKC3sIqgETSy3Gcb5tS
9 x0d3CcnFwhkyBxyH7sCLikcomgTp1GLESUkmt4XdjEsZSSgPEB12X6fdpYyz6XtQ
10 214d23f7vtCrqi/6n0ItemA12kYdYiGi4oF4RdrIWeyk5eQLPp0C/r0eIkz9+u9ls
11 9kgmXWttypdIp8m0zo5yVZ8zVfnoyLjZjuTsP2JRocPWDvWFlnP+7P2Yy2psQxK
12 XUM0VCL0BG0ABHWVWKjil6uVR2txVrFAWVvKUeJo9H30F2Lq98sCeXGuk9yZE8+B
13 2epewzUCgYEAE8N6Ayrc4dI5qGd/tL0v33ZGtt1JF0GISFYhADbx bik9+d1V4OGv
14 aMGQL22Yk152Lzrax9LMpM1nP1pNGleEI6IwFaWe+wUzkWsQzgYIAm7fuy+ky1F
15 e2feGEk2z8n6WiIvKhw1JX1f80wlaURmw2oh1mc1/C6semftnWQK9sCgYEAIK8L
16 sLMaqj3cXL1bTsk25tn061DMtKSWHJNDch7xPhaoWDQgQCfYw9MozzfnTMR/Od6d
17 poUf7IYe03f0o8iCt7WXVoKMyPQnyB9+cEHDAoFDWdA5C5zDYuT9ELPC54ddtoo
18 e1/F/Bp1EQ4zwaVRF/QUiU/tkU1ej+8kAZSquTMCGyEAhwnBT9+zjoPRZ0WEtPgx
19 ky04KpSpABcki99zSggCE8ycRsnkoFQAnYh6CIu3DUvS1EeLwJwhR3Dr5j6mh1OS
20 eVdbwemH5El5bgKeUy8jwvlu7Vjf8wdKj3Rie+BsKjobbGck50Ft+y1AfxxvU
21 9H7sa0nIo8Y078rTU1PdhycGyeA0C0WCDqe9T+/tjbSboefCY8gbkyEUjzxGdjT
22 JgXa8qfiV500DbevTBn/0/SQPiAHuAkinp+ng8grsXv3rqg2FiybUcTpGcepkfN
23 Zuu3iXg1CIA71hY3WVFyMysFB8bwSYAgeJY1E7BT5rLJw4KA4NG+pUSulEKfaV
24 c8ak50kCgYBG1jwcZGLtKSB8eTuU34frexgno4dqo3maQj42CBD0q82WrcMYvQ3
25 KuH8qIf/KiucszWXkv+XMAcSsoqlgudKTbx4XNe3dP5K3eHVdMttbXUQF/vN8cIW
26 CFrXVDBBW0gs0Dsuv0FyqLKOobaJVzprQbB2F6kgWbKqOiy7LkXNXw==
27 -----END RSA PRIVATE KEY-----

```

Open the publish key on master machine with command `cat ~/.ssh/id_rsa.pub`

```

ubuntu@ip-172-31-16-113:~$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDIPXBu1OKVwVMtyzae6kpW40QcSAau2tsZ1J3aBQPTCIDjubXDLk32MTPNU5ES+Fl1Fqvz11BqXAU1CzCN5aB7wzy5LNgJBawJ78qXv5NuKvArovenmjR6k54BqYAs0VqFkw1NQvexrF1eIn+jQ0Qin15s0ULg6PoCPEEZ0mVLezJ93B3Sb1zj6x185fN+Hw8Hf0Mhv0610R55jTut1ViwPiItbWx90q1m6hcJGf0J4IFnxXwa5gHm8zc4uEHHJ2wbI+V5TTtY5
ubuntu@ip-172-31-16-113:~$
```

Save this content to file `~/.ssh/authorized_keys` on slave machine with `echo` command

```

ubuntu@ip-172-31-22-93:~$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQDIPXBu1OKVwVMtyzae6kpW40QcSAau2tsZ1J3aBQPTCIDjubXDLk32MTPNU5ES+Fl1Fqvz11BqXAU1CzCN5aB7wzy5LNgJBawJ78qXv5NuKvArovenmjR6k54BqYAs0VqFkw1NQvexrF1eIn+jQ0Qin15s0ULg6PoCPEEZ0mVLezJ93B3Sb1zj6x185fN+Hw8Hf0Mhv0610R55jTut1ViwPiItbWx90q1m6hcJGf0J4IFnxXwa5gHm8zc4uEHHJ2wbI+V5TTtY5" >> ~/.ssh/authorized_keys

```

Now from master machine, try to ssh connection to slave machine. It should be successfully connectd.

```

ubuntu@ip-172-31-16-113:~$ ssh ubuntu@172.31.22.93
The authenticity of host '172.31.22.93 (172.31.22.93)' can't be established.
ECDSA key fingerprint is SHA256:OmzfpJbCxhUGo2I28WyDA8CVZKwCz/JgchzcYNixSsE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.31.22.93' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.4.0-72-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

 Get cloud support with Ubuntu Advantage Cloud Guest:
   http://www.ubuntu.com/business/services/cloud

20 packages can be updated.
0 updates are security updates.

```

Install Jenkins Server on Cloud

Running following command from master ssh console

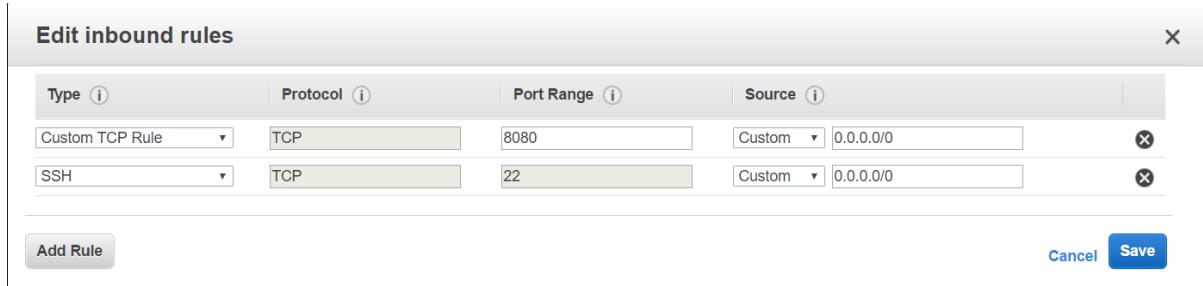
```

$ wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ >

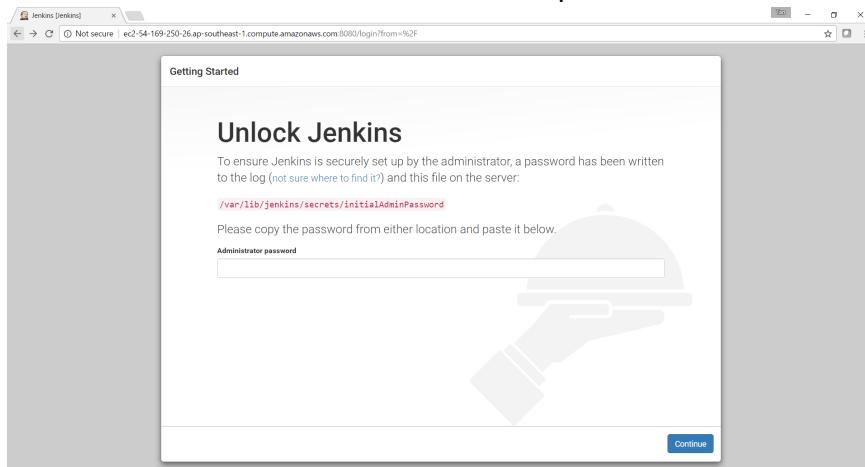
```

```
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install jenkins
$ sudo service jenkins start
```

Add port 8080 to inbound rules to master machine



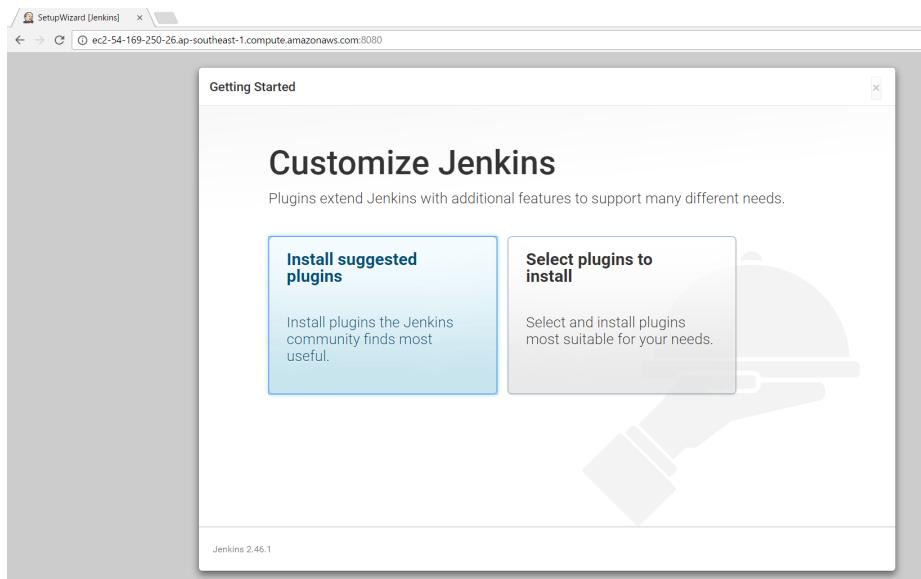
Access to Jenkins from local browser with port 8080



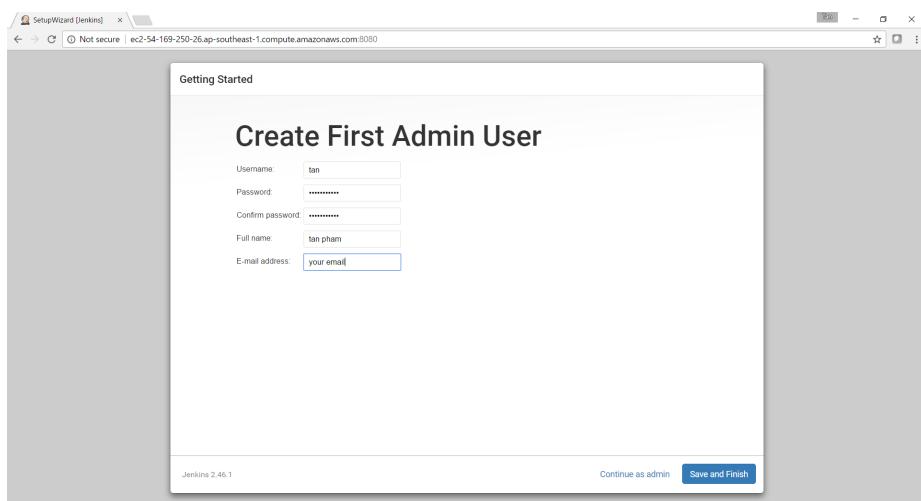
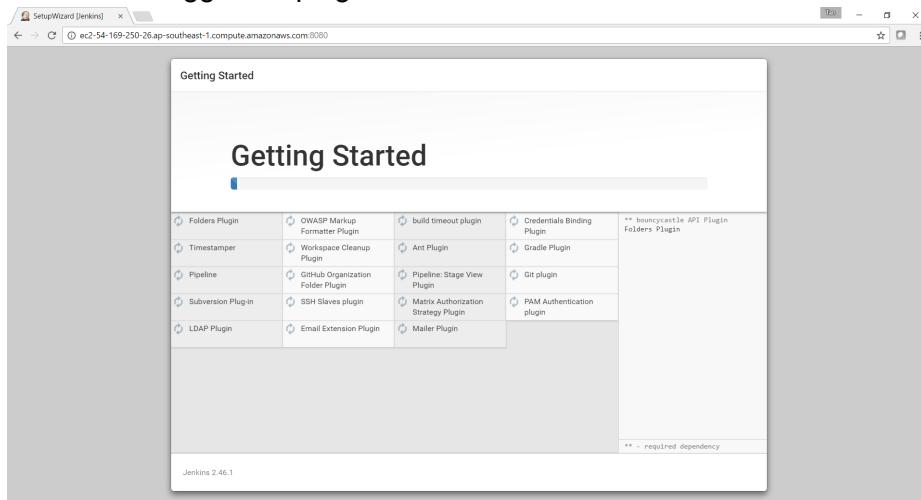
From command prompt using command to get default admin pass word
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

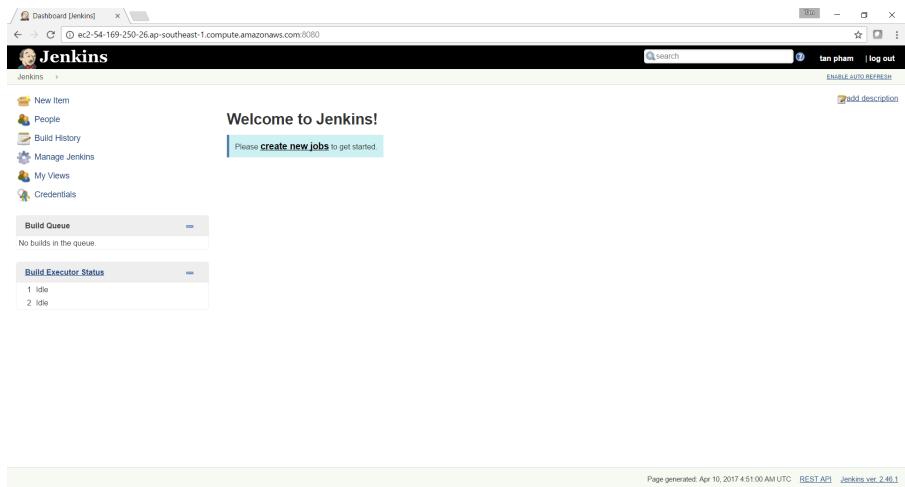
```
ssh -i "jenkins-server.pem" ubuntu@ec2-54-169-250-26.ap-southeast-1.compute.amazonaws.com
ubuntu@ip-172-31-16-113:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3181dd29d8b842039f97fc7d367ec56e
ubuntu@ip-172-31-16-113:~$
```

Put default password to browser to unlock Jenkins



Install with suggested plugins





Install Java on Slave Machine

SSH into slave nodes, upgrade the packages, and install a Java Runtime Environment:

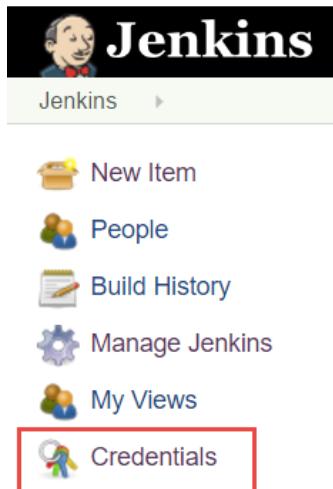
```
$ sudo apt-get update  
$ sudo apt-get upgrade  
$ sudo apt-get install default-jre  
$ java -version
```

After install successfully, java version should be show correctly. That's the only package Jenkins needs for slaves by default.

Configure a New Slave Agent from Jenkins

Create a new credential

Click to **Credential** from Jenkins home page



From **Global credentials** click to **Add Credential**

Jenkins

Back to credential domains Add Credentials

Global credentials (unrestricted)

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

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

Icon: S M L

Add the master private key in to **Key**, then **OK**

Jenkins

Back to credential domains Add Credentials

Kind: SSH Username with private key
Scope: Global (Jenkins, nodes, items, all child items, etc)
Username: ubuntu
Private Key: Enter directly
Key: -----BEGIN RSA PRIVATE KEY-----
-----END RSA PRIVATE KEY-----
Passphrase:
ID:
Description: ubuntu

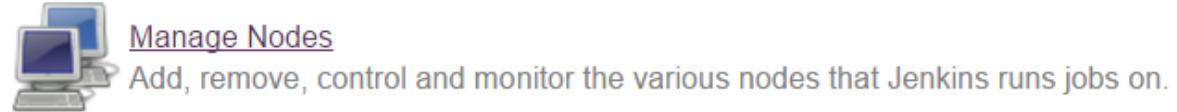
OK

Create a new node

Click to **Manage Jenkins**

-  [New Item](#)
-  [People](#)
-  [Build History](#)
-  [Manage Jenkins](#)
-  [My Views](#)
-  [Credentials](#)

Click to **Manage Node**



Then **New Node**

Jenkins > Nodes >

[Back to Dashboard](#)

[Manage Jenkins](#)

[New Node](#) (highlighted with a red box)

[Configure](#)

Then put in node name

Node name: ubuntu

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

[OK](#)

Put slave ip at **Host** and use all option as selected, finally click to **Save**

Name	ubuntu
Description	ubuntu on cloud
# of executors	1
Remote root directory	/home/ubuntu
Labels	ubuntu
Usage	Use this node as much as possible
Launch method	Launch slave agents via SSH
Host	172.31.22.93
Credentials	ubuntu (ubuntu) Add
Host Key Verification Strategy	Non verifying Verification Strategy
Advanced...	
Availability	Keep this agent online as much as possible

Node Properties

- Environment variables
- Tool Locations

[Save](#)

Click to Lanch Node, and connect successfully

```

Jenkins > Nodes > ubuntu
[INFO] [13:34:29] [SSH] Checking java version of Java
[INFO] [13:34:29] [SSH] Java version returned 1.8.0_111.
[INFO] [13:34:29] [SSH] Starting crypto client...
[INFO] [13:34:29] [SSH] Copying latest slave.jar...
[INFO] [13:34:29] [SSH] Copied 719,269 bytes.
[INFO] [13:34:29] [SSH] Executing command: cd "/home/ubuntu" && java -jar slave.jar
[INFO] [13:34:29] [SSH] Starting slave process: cd "/home/ubuntu" && java -jar slave.jar
<== Jenkins REPORTING CAPACITY==>channel started
Slave agent version 1.7
This is a Unix agent
Evacuated stdout
Agent successfully connected and online

```

S	Name ↓	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	master	Linux (amd64)	In sync	5.64 GB	0 B	5.64 GB	0ms 
	ubuntu	Linux (amd64)	In sync	5.88 GB	0 B	5.88 GB	3012ms 
	Data obtained	41 sec	41 sec	41 sec	41 sec	41 sec	41 sec

[Refresh status](#)

Create a Job Using Running on Cloud

Create a job which restricted node running with **ubuntu**

[General](#) [Source Code Management](#) [Build Triggers](#) [Build Environment](#) [Build](#) [Post-build Actions](#)

Project name: demo_job

Description:

[Plain text] [Preview](#)

Discard old builds 

GitHub project 

This project is parameterized 

Throttle builds 

Disable this project 

Execute concurrent builds if necessary 

Restrict where this project can be run 

Label Expression: ubuntu 

⚠ There's no agent/cloud that matches this assignment. Did you mean 'ubuntu' instead of 'ubun'?

[Advanced...](#)

Build

[Execute shell](#) 

Command: `ls -l`

See [the list of available environment variables](#)

[Advanced...](#)

[Add build step ▾](#)

Run the job and see if the job running

Console Output

```
Started by user tan_pham
Building remotely on ubuntu in workspace /home/ubuntu/workspace/demo_job
[demo_job] $ /bin/sh -xe /tmp/hudson909968911449839448.sh
+ ls -l
total 0
Finished: SUCCESS
```

Jenkins Security

Create gol_developer user

From **Manage Jenkins**, click to **Manage Users**



Click to **Create User**

A screenshot of the Jenkins "Users [Jenkins]" page. The URL is "localhost:8080/securityRealm/". A "Jenkins" logo is at the top. Below it is a breadcrumb trail: "Jenkins > Jenkins' own user database". There are three buttons: "Back to Dashboard", "Manage Jenkins", and "Create User", with "Create User" being highlighted with a green border.

Add information for new user

Create User

Username:	<input type="text" value="GOL_developer"/>
Password:	<input type="password" value="....."/>
Confirm password:	<input type="password" value="....."/>
Full name:	<input type="text" value="GOL_developer"/>
E-mail address:	<input type="text"/>

Create User

Then new user is created



Matrix-based Security

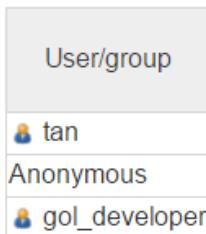
Click **Configure Global Security**



Click to **Matrix-based security**

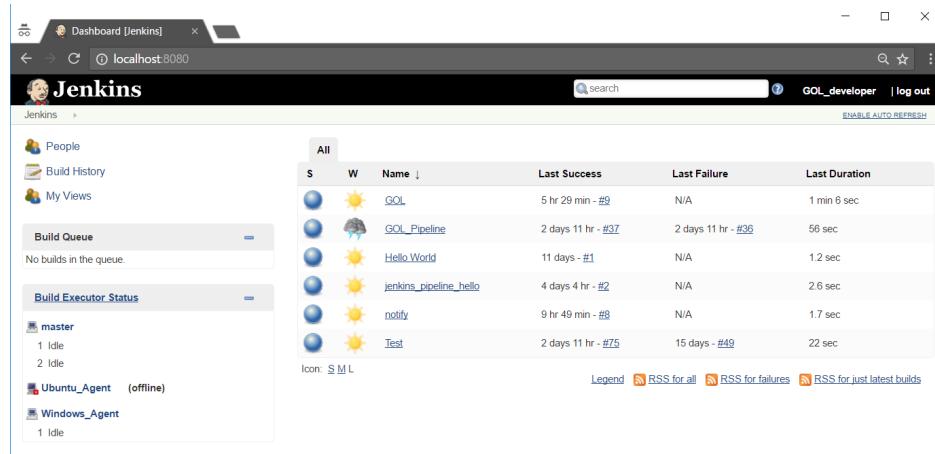
Matrix-based security

Setting all permission for **your user**, and try to setting for user **gol_developer** overall read and job read permission. Finally click to **Save** button.



The screenshot shows the Jenkins 'User/group' configuration page. It lists three users: 'tan' (with a user icon), 'Anonymous' (with a user icon), and 'gol_developer' (with a user icon). The 'gol_developer' entry is highlighted in blue, indicating it is selected or being edited.

Logout from your account, then try to login with **gol_developer** user. Now GOL_developer could read the jobs only.



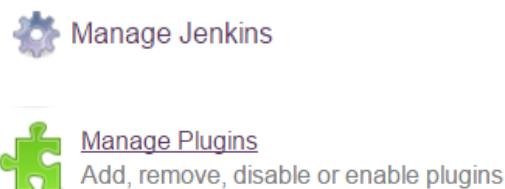
The screenshot shows the Jenkins dashboard at localhost:8080. The left sidebar includes links for People, Build History, My Views, Build Queue (empty), Build Executor Status (master: 1 Idle, Ubuntu_Agent (offline): 1 Idle, Windows_Agent: 1 Idle). The main content area displays a table of builds under the 'All' tab. The table columns are S (Status), W (Weather icon), Name, Last Success, Last Failure, and Last Duration. The data includes:

S	W	Name	Last Success	Last Failure	Last Duration
●	☀	GOL	5 hr 29 min - #8	N/A	1 min 6 sec
●	☁️	GOL_Pipeline	2 days 11 hr - #37	2 days 11 hr - #36	56 sec
●	☀	Hello_World	11 days - #1	N/A	1.2 sec
●	☀	jenkins_pipeline_hello	4 days 4 hr - #2	N/A	2.6 sec
●	☀	notify	9 hr 49 min - #8	N/A	1.7 sec
●	☀	Test	2 days 11 hr - #75	15 days - #49	22 sec

Role-based Security

Install Role-based Plugin

From home page click to **Manage Jenkins**, then **Manage Plugin**.



The screenshot shows the 'Manage Jenkins' section followed by 'Manage Plugins'. Under 'Available' filters, the 'Role-based Authorization Strategy' plugin is listed. Its details show it's version 2.3.2, with options to 'Install without restart' or 'Download now and install after restart'.

At **Manage Plugin** page, select tab **Available**, then typing into filter **Role-based Authorization Strategy**, click to **Install without restart**



The screenshot shows the 'Manage Jenkins' section followed by 'Manage Plugins' under the 'Available' tab. The 'Role-based Authorization Strategy' plugin is selected for installation. The 'Install without restart' button is highlighted with a green border.

Configure Global Role

From **Manage Jenkins**, configure to **Configure Global Security**, and then at session **Authorization** select **Role-based Strategy** option. Then click to **Save** button.

Authorization

- Anyone can do anything
- Legacy mode
- Logged-in users can do anything
- Matrix-based security
- Project-based Matrix Authorization Strategy
- Role-Based Strategy

Now from **Manage Jenkins**, you will see new item call **Manage and Assign Roles**



Click to **Manage and Assign Roles**, UI allow to **Manage Roles and Assign Roles** will show up

Manage and Assign Roles

-  [Manage Roles](#)
Manage Roles
-  [Assign Roles](#)
Assign Roles
-  [Role Strategy Macros](#)
Provides info about macro usage and available macros

Click to **Manage Roles**, at **Global roles**, add **dev** role with read permission at **Overall** and full permission with Job, then click to **Save**

Global roles		Overall																		Job																	
Role	User/group	Administer	Configure/Update Center	Read	Run Scripts	Upload Plugins	Create	Delete	Manage Domains	Update	View	Build	Configure	Connect	Create	Delete	Disconnect	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace											
admin	tan	<input checked="" type="checkbox"/>																																			
dev	GOL_developer	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>											

Back to screen **Manage Roles and Assign Roles**, then click to **Assign Roles**. Now assign GOL_Developer to **dev** role. Click to **Save**

Global roles

User/group	admin	dev
tan	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Anonymous	<input type="checkbox"/>	<input type="checkbox"/>
GOL_developer	<input type="checkbox"/>	<input checked="" type="checkbox"/>

That it, now try to logout from Jenkins and login with user GOL_developer. You will see GOL_developer could see all the job, but could not see **Manage Jenkins**

The screenshot shows the Jenkins dashboard with several jobs listed in a table:

S	W	Name	Last Success	Last Failure	Last Duration
●	☀️	GOL	18 hr · 89	N/A	1 min 6 sec
●	🌧️	GOL_Pipeline	2 days 23 hr · 832	2 days 23 hr · 832	56 sec
●	☀️	HelloWorld	12 days · 83	N/A	1.2 sec
●	☀️	jenkins_pipeline_hello	4 days 16 hr · 82	N/A	2.6 sec
●	☀️	notdy	22 hr · 88	N/A	1.7 sec
●	☀️	Testl	2 days 23 hr · 872	15 days · 869	22 sec

Legend: RSS for all, RSS for failure, RSS for just latest builds.

Configure Project Role

Currently, GOL_developer could see and access all jobs, You want to configure so GOL_developer only could see GOL project.

From **Manage and Assign Roles** screen, click to **Manage Roles**, at **Project roles** section, add GOL

The 'Project roles' section shows a table with columns: Role Pattern, Credentials, Job, Run, SCM. Under 'Role to add', the 'GOL' pattern is selected. The 'Pattern' dropdown also has 'GOL' selected. A 'Save' button is visible at the bottom.

And then select all permission for GOL role. Click **Save** button.

The 'Project roles' table shows the GOL role with all checkboxes checked under the Job, Run, and SCM columns.

From **Global roles** section, remove all permission related to job and only keep overall read permission.

The 'Global roles' table shows two users: admin and dev. The 'Overall' row contains checkboxes for Administer, Configure, UpdateCenter, Read, RunScripts, and UploadPlugins. The 'admin' row has checked boxes for Administer, Configure, UpdateCenter, Read, and UploadPlugins. The 'dev' row has checked boxes for Read and UploadPlugins.

Now come back to **Assign Role** screen, then at the **Project Role** section, add GOL_developer user, and select GOL role for this user. Save configure.

Project roles

User/group	GOL
GOL_developer	<input checked="" type="checkbox"/>
Anonymous	<input type="checkbox"/>

Logout from your admin and login back with GOL_developer. Now you can see that GOL_developer only see GOL project.

The screenshot shows the Jenkins dashboard with a single project named "GOL". The project status is marked with a green circle (S) and a sun icon. The build time is listed as "20 hr - #9". There is no entry in the "Last Failure" column. The "Last Duration" is "1 min 6 sec". Below the dashboard, there is a legend with three items: "RSS for all" (RSS feed icon), "RSS for failures" (RSS feed icon), and "RSS for just latest builds" (RSS feed icon).

Disable Security

In some case, you do not need any security (like a personal build system), you just want to disable Jenkins security. To do this, go to place where Jenkins is installed, edit file

config.xml

The screenshot shows a Windows File Explorer window displaying the contents of the Jenkins configuration directory at `C:\Users\TAN\jenkins\config.xml`. The `config.xml` file is highlighted with a blue selection bar. Other files visible include `jenkins.exe`, `queue.xml.bak`, `identity.key.enc`, `secret.key`, `jenkins.install.InstallUtil.lastExecVersion`, `secret.key.not-so-secret`, `.owner`, `jenkins.install.UpgradeWizard.state`, `jenkins.err.log`, `jenkins.out.log`, `jenkins.wrapper.log`, `redirect.log`, `jenkins.war`, and `jenkins.exe.config`.

Change value of attribute `useSecurity` to `false`

The screenshot shows the `config.xml` file open in Notepad++. The code is as follows:

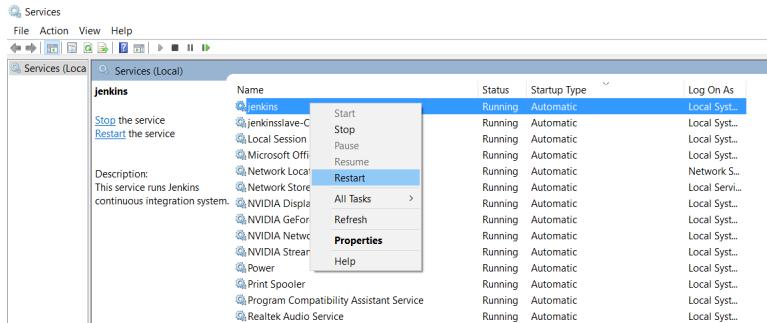
```

1 <?xml version='1.0' encoding='UTF-8'?>
2 <hudson>
3   <disabledAdministrativeMonitors/>
4   <version>1.0</version>
5   <numExecutors>2</numExecutors>
6   <mode>NORMAL</mode>
7   <useSecurity>false</useSecurity>
8   <authorizationStrategy class="hudson.security.GlobalMatrixAuthorizationStrategy"/>
9   <securityRealm class="hudson.security.HudsonPrivateSecurityRealm">
10    <disableSignup>true</disableSignup>
11    <enableCaptcha>false</enableCaptcha>
12  </securityRealm>

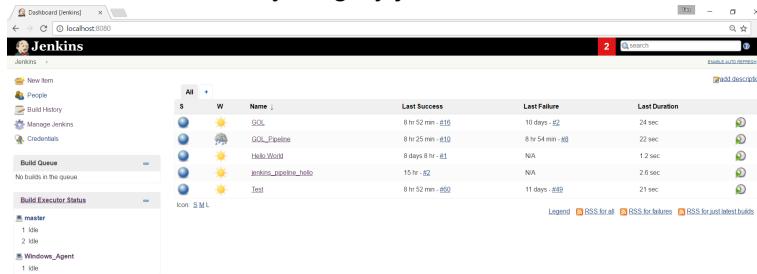
```

The `<useSecurity>` tag on line 7 is highlighted with a green selection bar, and the word `true` is replaced by `false`.

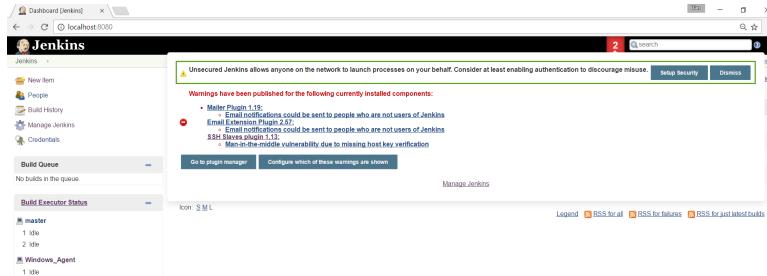
Restart Jenkins by service after change security configure



Now, You can do anything by just access to Jenkins without of need to login



And You can see a warning message from Jenkins show that security is disabled. And suggest You go for configure security



https://www.youtube.com/watch?v=rdypdeG_idE

Jenkins Backup and Restore

Install and Configure ThinBackup Plugin

From **Plugin Manager**, select **Available** tab, search for **thinBackup** plugin, select check box, then click to **Install without restart**

Updates Available Installed Advanced

Filter

Name	Version
ThinBackup	1.9

Install without restart Download now and install after restart Check now

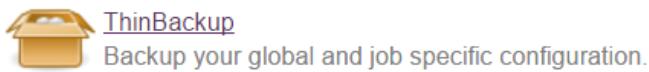
Installing Plugins/Upgrades

Preparation

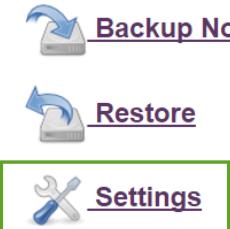
- Checking internet connectivity
- Checking update center connectivity
- Success

ThinBackup  Success

From **Manage Jenkins**, click to **ThinBackup**, then click to **Settings**



ThinBackup



Normally we not need to backup build result, so uncheck **Backup build results**. Specify **0 0 * * 0** mean do full backup every 7 days (This is for demo, you can specify timeline that suite with your jenkins system). Finally, click to **Save** button

thinBackup Configuration

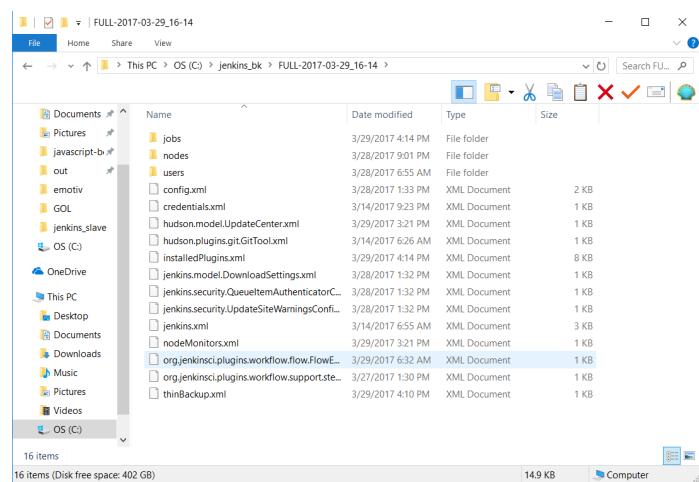
Backup settings

Backup directory	C:\jenkins_bk
Backup schedule for full backups	0 0 * * 0 <small>⚠ Cron schedule warning: Spread load evenly by using 'H 0 * * 0' rather than '0 0 * * 0'</small>
Backup schedule for differential backups	
Max number of backup sets	-1
Files excluded from backup (regular expression)	
<input checked="" type="checkbox"/> Wait until Jenkins/Hudson is idle to perform a backup	
Force Jenkins to quiet mode after specified minutes	120
<input type="checkbox"/> Backup build results	
<input type="checkbox"/> Backup 'userContent' folder	
<input type="checkbox"/> Backup next build number file	
<input type="checkbox"/> Backup plugins archives	
<input type="checkbox"/> Backup additional files	
<input type="checkbox"/> Clean up differential backups	
<input type="checkbox"/> Move old backups to ZIP files	

Save

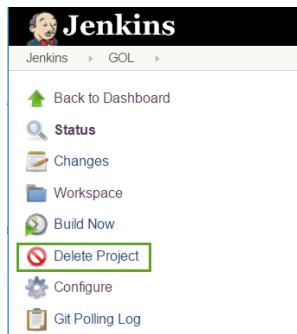
Manually Backup

Now click to **Backup Now** you will see full backup folder is created inside C:\jenkins_bk



Restore from Backup

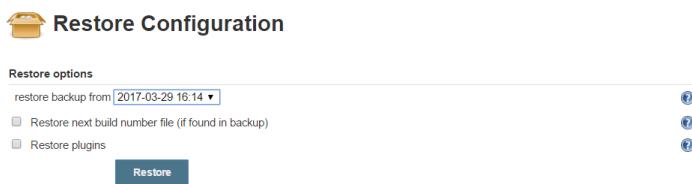
Try restore function by delete a job and then restore it. From home page click to GOL job, then click to **Delete Project**



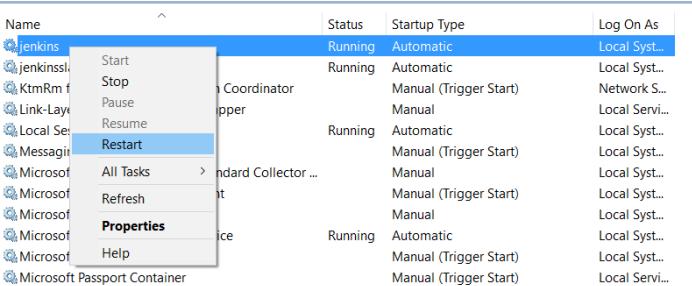
From **ThinBackup** click to **Restore**



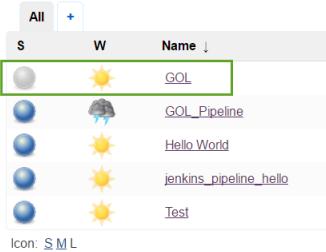
Select version want to restore and click to **Restore** button



Restart Jenkins by window service



Now back to Jenkins home page, You will see GOL job restored successfully.



Jenkins Interview Questions

Jenkins Interview Question

What is Jenkins?

Jenkins is an open source tool with plugin built for continuous integration purpose. The principle functionality of Jenkins is to keep a track of version control system and to initiate and monitor a build system if changes occur. It monitors the whole process and provides reports and notifications to alert.

What is the requirement for using Jenkins?

To use Jenkins you require

- A source code repository which is accessible, for instance, a Git repository
- A working build script, e.g., a Maven script, checked into the repository

What are the advantages of Jenkins?

Advantage of Jenkins include

- At integration stage, build failures are cached
- For each code commit changes an automatic build report notification generates
- To notify developers about build report success or failure, it is integrated with LDAP mail server
- Achieves continuous integration agile development and test driven development
- With simple steps, maven release project is automated
- Easy tracking of bugs at early stage in development environment than production

How you can move or copy Jenkins from one server to another?

- Slide a job from one installation of Jenkins to another by copying the related job directory
- Make a copy of an already existing job by making clone of a job directory by a different name
- Renaming an existing job by renaming a directory.

What are the commands you can use to start Jenkins manually?

To start Jenkins manually, you can use either of the following

- (`Jenkins_url/restart`): Forces a restart without waiting for builds to complete
- (`Jenkins_url/safeRestart`): Allows all running builds to complete

How can create a backup and copy files in Jenkins?

Jenkins saves all the setting, build artifacts and logs in its home directory, to create a backup of your Jenkins setup, just copy this directory. You can also copy a job directory to clone or replicate a job or rename the directory.

How you can set up Jenkins job?

To create a project that is handled via jobs in Jenkins. Select New item from the menu, once this done enter a name for the job and select free-style job. Then click OK to create new job in Jenkins. The next page enables you to configure your job.

What are the two components Jenkins is mainly integrated with?

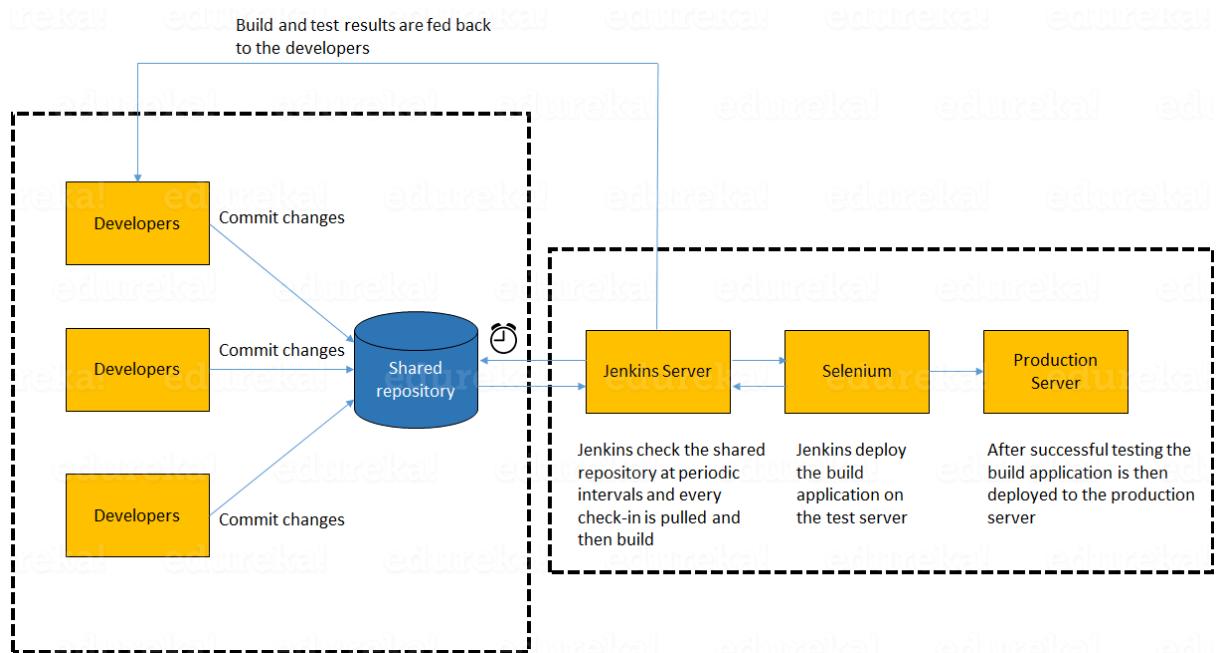
Jenkins is mainly integrated with two components

- Version Control system like GIT, SVN
- And build tools like Apache Maven.

CI/CD Interview Question

What is continuous integration?

It is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.



In the diagram shown above:

1. Developers check out code into their private workspaces.
2. When they are done with it they commit the changes to the shared repository (Version Control Repository).
3. The CI server monitors the repository and checks out changes when they occur.
4. The CI server then pulls these changes and builds the system and also runs unit and integration tests.
5. The CI server will now inform the team of the successful build.
6. If the build or tests fails, the CI server will alert the team.
7. The team will try to fix the issue at the earliest opportunity.
8. This process keeps on repeating.

Why is Continuous Integration important?

Two important reasons:

- Defects found early cost less to fix : When a defect is found immediately after a developer codes it, it takes 10x times less time to fix it compared to finding the defect a month later.
- Reduced Time to Market : Software is always tested. So, it is always ready to move to further environments.

What are the success factors for Continuous Integration?

Here you have to mention the requirements for Continuous Integration. You could include the following points in your answer:

- Maintain a code repository
- Automate the build

- Make the build self-testing
- Everyone commits to the baseline every day
- Every commit (to baseline) should be built
- Keep the build fast
- Test in a clone of the production environment
- Make it easy to get the latest deliverables
- Everyone can see the results of the latest build
- Automate deployment

Special Bonus

List of BEST DEVELOPMENT COURSE on Udemy with cheapest coupon from author!

<http://devcourses.co>

- Web development bootcamp
- Game development with C# and Unity
- Game, app development for iOS
- Game, app development for Android
- Data science with Python
- SQL and database